REAL TIME COLLISION DETECTION FOR COMPLEX SIMULATIONS BASED ON HYBRID MULTI RESOLUTION APPROXIMATION OF CAD MODELS

I.F. Ceruti[(a)], G. Dal Maso[(b)], D. Rovere[(c)], P. Pedrazzoli[(d)], C. R. Boër[(e)]

[(a)] Technology Transfer System s.r.l. - TTS
[(b) (c) (d) (e)] University of Applied Science of Southern Switzerland - SUPSI

ceruti@ttsnetwork.com; giovanni.dalmaso@supsi.ch; diego.rovere@supsi.ch;
paolo.pedrazzoli@supsi.ch; claudio.boer@supsi.ch

ABSTRACT
This paper presents models and ideas at the foundation of a new collision detection engine for manufacturing systems simulation. The fast and reliable collision detection method proposed is based on an hybrid approach (usage of mixed type of bounding volumes shape and positioning) capable to satisfy the requirement of a quick and consistent detection while dealing with huge 3D models.

Keywords: complex simulation environment, collision detection, real time, multi resolution approximation.

1. INTRODUCTION

The importance of computer animated simulations is ever increasing both in the design and in the production phase of manufacturing systems.

A comprehensive simulation, whether applied to a single machining centre or to an entire plant, must be capable to represent various aspects of the reality. In a modular 3D simulation library (as the one this paper refers to, described in (Pedrazzoli 06)) these aspects are taken in charge by dedicated engines.

Among those, the collision detection engine deals with the interactions between moving objects. A collision detection engine for real time simulation of manufacturing environments must satisfy the constraints of short response time, low resource usage and the capability to deal with huge simulation models. These models are mostly exported directly from the constructive drawings coming out from CAD software in order to cut the simulation set-up costs and increase the reliability of the simulation results. Thus, the resulting meshes are characterized by an high level of detail and a huge number of triangles.

This paper deals with the design and development of a fast and reliable collision detection engine based on an hybrid approach (mixed type of bounding volumes shape and positioning) capable to satisfy all these requirements.

The article is based on the following structure: after a survey on collision detection state of the art, the advantages and limitations of the different approaches adopted to meet the requirements for collision detection engine will be summarized. In the third part the selected approach at the base of the engine will be extensively explained. Section four details industrial test cases results.

2. STATE OF THE ART

2.1. Problem background

The collision detection engine addressed by this paper is conceived in order to be plugged into a framework for the 3D simulation of complex manufacturing environments, representing either a single machining centre or a complete production plant.

In the specific case, the simulation is connected to a real CNC and acts as a substitution of the real environment receiving the same inputs usually sent to the real machines and emulating their behaviour. This kind of application is mainly used in two different phases of the machine/plant lifecycle:

1. During the development of the machine/plant management logics in order to speed up the debug phase of the CNC and PLC without facing the risk to damage the real structures (mainly to avoid auto collision situations).
2. In production in order to perform an off-line check of complex part programs, thus avoiding possible collisions of the machining heads with the fixtures or with the work piece.

Therefore the collision detection performed during these simulations must be responsive still remaining as adherent as possible to the tested mechanical structures. For this reason, the developed applications based on this framework are required to directly manage the models coming from the constructive drawings.

This approach allows the reuse of machine and plants 3D models from the design phase, cutting down the simulation set up times and the related costs, thus increasing the complexity of the collision check. In fact, the great number of different existing CAD systems requires the models to be exported to a neutral mesh exchange format like VRML (Carey 97), with the result,

in the case of highly detailed models, of big non optimized meshes.

This process, along with the huge number of contiguous moving parts, could dramatically increase the response time of the collision detection engine making it unusable for real-time simulations.

During the development of the here presented collision detection engine several approaches described in literature have been used to automatically create collision structures. Their pros and cons are described in detail hereinafter. Common problems detected in all of these approaches are, on one side the under-exploitation of the knowledge deriving from the kinematic structures involved in the simulation and, on the other, the missing adaptation to the precision level required by each part of the machine/plant.

For this reason a new hybrid, semi automatically defined, multi resolution approximation of CAD models is here proposed; this is based on a mixed structure of primitives which gives the user the opportunity to take advantage of the knowledge deriving from the motion logics, the exclusion rules and the required precision levels for each of the controlled objects. A 3D application has been developed to support the user during the process of collision structures definition.

2.2. Previous work

Collision detection problems have been extensively studied in literature (Kockara 07, Lin 98). (Hubbard 93) introduces a first classification of collision detections: *broad-phase* collision detection allows detecting objects that should be subsequently tested, while and *narrow-phase* collision detection determines the exact collision. This scheme is broadly used in literature, to classify collision detection algorithms (Ericson 05).

Algorithms for the broad-phase are mainly of three kinds: exhaustive, sweep and prune and hash table based. Exhaustive approach simply leads all pairs to be tested against each other. Sweep and prune (Baraff 03, Lin 94) sorts objects through their projection onto the coordinate axes and chooses the ones which projection overlaps. Hierarchical hash tables (Mirtich 96) divides the space into cells and selects objects that overlap the same cell.

Algorithms for narrow phase collision detection can be categorized into four groups: feature-based, simplex-based, image-space based, volume-based and spatial data structures (Moore 98).

Feature-based algorithms, such as Lin-Canny (Lin 91), V-Clip (Mirtich 98), and SWIFT (Ehmann 00a, Ehmann 00b), directly works on the geometric primitives of the object, such as points, vertices and faces. Simplex-based operates on the convex hull of its points: a well known simplex-based algorithm is GJK (Bergen 03).

Image space base (ISB) techniques are computed by image-space occlusion queries which are convenient to be implemented on the graphics hardware, the GPU, and common hardware, the CPU (Benes 05). Cinder (Knott 03) is a well known example of ISB, while frontiers on this type of algorithms can be found in (Heidelberger 04) and (Govindaraju 04).

Volume based algorithms are conceptually based on the same idea of the ISB techniques; however, they use different methods to compute layered depth images (Heidelberger 04) and distance fields. This group of algorithms are also suitable for GPU implementations. In (Gundelman 03) can be found a volume based algorithm example.

There are two types of spatial data structures for collision detection: spatial subdivision and bounding volume hierarchies (BVH). The first approach recursively divides the space, while the second recursively or iteratively partitions the object itself. With spatial partitioning, splitting of polygons (that is commonly used) causes the increasing of tree depth and the lost of performance. In addition, since cell size of the spatial partitioning cannot cover objects' primitives tightly, when objects are close, contact status determination is difficult.

On the contrary, BVHs provide smaller and tighter hierarchies, and are more suitable for general shapes than simplex based and feature based algorithms. BVH can be called as discrete representation of level of details of objects. At the first level, hierarchy includes one bounding volume which is a very coarse representation of an object. Further levels include more detailed representations of the object. The leaf level (or finest level) of the hierarchy generally includes the object primitives (lines, triangles, or tetrahedra). Bounding volume (BV) does not necessarily enclose its children's bounding volume; instead it must enclose the geometry of an object included in the children BVs.

The selection of the root volume to be descended is called *traversal rule*: generally largest volume is chosen in order to lower the chance of finding overlapping. Non-overlapping BVs are discarded from further consideration (pruning). At last in the traversal, when two leaf nodes from two distinct volumes are reached, then there are two possibilities: testing whether two primitives are colliding (pair-wise test) or testing one primitive with the other's leaf bounding volume (primitive-volume test). If objects' primitives are colliding, a pair-wise test should be performed anyway. Thus, there is a trade-off between the number of iterations and the complexity in the overlap tests.

(Gottschalk 00) states that recursively traversing BVHs is often a bad choice since the number of primitives and the hierarchies can be quite large. This problem can be solved by using iterative traversal technique with FIFO queue (Dingliana 00), with the introduction of a priority on the pair-wise BV tests.

Typical BVs are: axis-aligned bounding boxes (AABB) and spheres, usually chosen for the simplicity of such structure intersection test, OBBs (Oriented Bounding Boxes) and k-DOP (Discrete Orientation Polytopes), used for their fitting approximation of the object, and recently also cylinder (Ketchel 06) for particular geometrical bodies. In addition, with Welzl's algorithm (Fischer 03), finding better fitted spheres,

makes spheres preferable over other topological BVs. In overall, there is a trade-off between BVH complexity and performance (Gottschalk 00). Complex topology choices establish tighter fitted BVs and so fewer overlap tests but causes performance lost. On the other hand, less complex hierarchies provide faster overlap test but less tight BVs.

Recent attempts have been made to summarize different approaches in an unique algorithm. In (Yoon 04), the CHPM, Clustered Hierarchy of Progress Mesh, represents a model with a dual hierarchy: one for a coarse-grained selective refinement, the second for fine-grained local refinement. This technique also exploits the idea of error bounded detection, for approximate but fast intersection spotting.

3. ADOPTED APPROACHES

The here-described technique is the result of an evolution: different approaches have been adopted and developed to face the problem described in the previous sections. Some of them are already known in literature, while in the present work others are studied and developed to face the described problem. The aim of this section is to present an overview of these algorithms, highlighting the advantages and problems they showed when concretely used. These problems, strictly bonded to the selected application field (the real time simulation environments for big models), leaded up to the replacement of the algorithm with a more suitable one.

3.1. Literature Techniques

3.1.1. Triangles

The first idea was to use directly the triangles coming from the simulation model, surrounded by an OBB as approximation. This allowed achieving a unchallenged precision of the response without having any support structure.

Despite this, triangle technique has clearly visible problems: apart from requiring a huge number of test to be performed (each triangle have to be tested against each other triangle), the triangle-triangle test is one of the most time-consuming test, bringing the collision test far from being real time.

3.1.2. BVT (Bounding Volume Tree)

The usage of BVH was considered the solution to be adopted to speed up the collision tests. The model meshes were surrounded by binary trees of both spheres and OBBs: this approach allows performing fast rejection tests when objects were far apart, preserving the precision detail of triangles. The BVT structure is built automatically from the model mesh, assigning triangles to tree leaves, with a controlled tree depth.

Also this approach shows intrinsic lacks. The collision detection performances heavily downgrade when objects are close to each other: almost all leaves triangles need to be tested: in these cases, very frequent in considered applications, the bounding volume structure is just an undesired overhead. An accurate analysis, as a matter of fact, showed that the efficiency of the test was strictly linked to mesh layout. Most of them generate bounding volumes trees with leaves load balancing problems: many triangles are encapsulated with few leaves or, on the other hand, many triangles are shared between leaves.

This brings redundant triangle-triangle test and pointless tree branches, with children having the same informative value of their parents. As a consequence, also the control parameter has been shown to be completely useless, because of its inability to tune efficiently the created structure. The resulting collision detection time was unpredictable, resulting, in most cases, unacceptably long.

3.1.3. Example

The following figures show an example of problems described in the previous section.

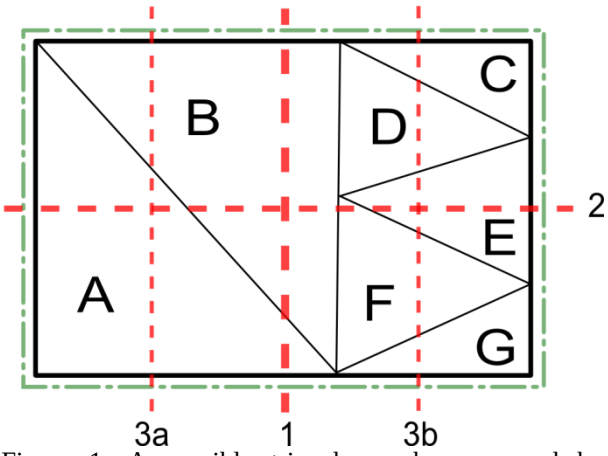

Figure 1: A possible triangle mesh, composed by triangles A B C D E F G, which BV (line-dot) is subsequently cut along the maximum extension. The second cut is common for both first level children, and have been indicated once for simplicity.

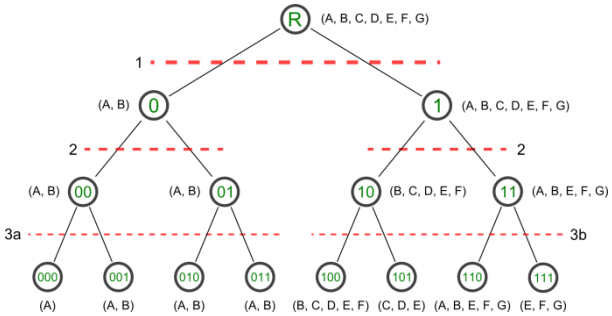

Figure 2: A hierarchical structure representing the mesh in [Figure 1]. For each node the related triangles and the generating cut are indicated.

In [Figure 1] a mesh of an object has been taken into account. The OBB has been subsequently divided into children using a simple strategy: the longest extension axis is split in two parts. Moreover a tree with a depth of four is considered enough for the example,

thus three cuts have been performed (the second cut is common for both first level children).

The hierarchy shown in [Figure 2] contains examples of all problems affecting hierarchical structures built on mesh, as for example BVT. The resulting structure is an heavily unbalanced tree: the right branch (1) of the root node (R), as a matter of fact, contains much more primitives for node than the left branch (0). Moreover, considering the leaves of this tree, many primitives are shared between nodes (e.g. B for all nodes except 000, 101, and 111). Node 01 and its children are a typical case of useless branch: it replicates the informative content of its parent referring triangles A and B.

3.2. Developed Techniques

3.2.1. BVMix (Bounding Volume Mix)

As a possible solution to trees with uselessly long branches, a fixed hierarchy of different bounding volumes has been adopted, also including space partition concepts.

This technique, titled BVMix, gives a multi resolution approximation of the given mesh. The model is represented, as a first approximation, by a sphere and by its OBB: this couple of bounding volumes assures a fast rejection when objects are far apart. The OBB is then subdivided in twenty-seven children called *clusters* obtained cutting the main OBB homogeneously on each of its axes. Each cluster contains a grid of spheres (*leaf spheres*) obtained by subdivision, in a tuneable number, of the shortest side: each of these spheres is associated with the triangles of the mesh concerning to the sphere volume. At the end each empty volume, both cluster and sphere, is removed from the hierarchy.

BVMix, preserving the triangle precision and the BVT hierarchical idea and fast rejection idea, is characterised by a control parameter (number of cluster side subdivision) able to fit the complexity of the model. This algorithm gave also the possibility to approximate the test, ending it at the sphere level: a fast response, especially in machine debug preliminary phases, is in most cases a desired feature for a simulation collision detection algorithm. Despite the new approximation capabilities, and the performances improvement in general cases, compared to other techniques adopted, also BVMix suffers some problems.

As for BVT, also for BVMix efficiency depends on mesh layout because triangles are still the final actors of the collision test, and the problem of triangles shared between leaves (spheres) is still unresolved. Last but not least, also in case of usage for approximated collision test, the user has no control on the approximation generated by the grid of spheres, not even tuning properly the control parameter.

3.2.2. Fitting Boxes Algorithm

Analysing the up-to-now described approaches, it's noticeable that problems for achieving goals of collision detection engine, with simulations mesh model, comes from the model mesh layout. Triangles, apart from being the most precise way for detecting collision, are the real bottleneck of every algorithm for collision detection. Thus the hereinafter described approach is based on the idea to sacrifice the precision of triangles in favour of a good approximation able to give a fast and conservative response to collision test.

The last approach described in this section is based on an approximation of each model with a two level hierarchy of boxes: we called it the Fitting Boxes Algorithm. The first level of FBA hierarchy is simply the model OBB, called *master box*. The second level is a set of OBB (aligned with the master box) called sub boxes: each sub box is obtained by refinement of the master box or of another sub box.

The refinement operation substitutes a box with a set of boxes sharing orientation, and whose dimension is a percentage of the generating box, containing at least one triangle of the model. The FBA building process can be made both by the user (who indicates which box has to be refined and the refinement percentages) or automatically (just indicating a percentage the algorithm refines the master box): this process leads to a controllable resolution of the model.

This structure is then used from the collision detection engines for the test phase: excluding the triangle from test phase considerably enhance test response time in each respective situation of objects involved. Moreover FBA evidently did not suffer of problems such as load balancing of the hierarchy (also because it exploit a two level hierarchy), shared triangles, or dependence of its efficiency from the level of detail of the mesh of the model involved in test.

Notwithstanding his effective features, that made FBA more usable simulations than previous approaches performed, also FBA has a main disadvantage: automatic built structures does not always perform a good approximation of the starting model. Also “hand-made” structure approximation can be either poor or needs a great number of boxes to reach a good approximation: curvatures on object are a clear example of this lack, and are intrinsic in the choice of box as approximating structure.

4. HMR APPROACH

None of our previous approaches took into account the peculiarities of the models used in the simulation environment. Machine CAD drawings are composed of a great number of shapes, whose geometry is different the ones available in classic 3D applications (e.g. trees, people, animals, vehicles). The presence of shapes with a lot of rough edges represent a problem when trying to apply automatic mesh reduction algorithms like edge collapse (Cignoni 00).

The approach we are describing exploits a key factor in simulation environments: the presence of physical laws that constraints the movements of the parts composing the machine model. The knowledge of the kinematics allows defining the degrees of freedom

of each part. For example, the limitation of one part movement can easily determine its bounding box in each allowed position. This can reduce the complexity of the approximating model (at least in a coarse-grained level of detail).

A very important aspect in machine simulation is the possibility to define a tolerance in the collision detection precision: some critical parts don't need to get too much close to each other for safety reasons (e.g. to avoid contacts for vibrating parts): in this case the simulation must report a collision even if the model meshes do not geometrically intersect.

4.1. Description

The exploitation of the simulation model knowledge and the inability to use automatic mesh reduction algorithms lead to an approach that defines and exploits a hierarchical multi resolution (HMR) approximating description of the models. This section describes how to define the approximating volumes, the hierarchical relationships and, for moving parts, the association with their degrees of freedom.

An object occupied volume can be defined using a set of geometric primitives optimized for intersection test: spheres, boxes and cylinders. These primitives are well suited for approximating most of the shapes found in machine CAD models. For example a screw can be approximated with two cylinders (one for the head and one for the shaft) or with a single cylinder, depending on the precision required for that part.

Models could eventually contain part that, even if made by simple shapes, cannot be well described with the available primitives. For this reason the here-described approach also includes the possibility to define a triangle mesh starting from the object convex hull. Given that triangle intersection tests are the most expensive, the number of approximating triangles mesh has to be low. The impact of the presence of triangle meshes can moreover be limited using an enclosing simple volume, like a sphere or a box, for early exclusion.

The advantages of hierarchical techniques can be achieved, in this approach, through the organization of these geometries in trees. This structure is traversed with the classical collision detection mode: test fails when a child fails, while succeeds when a leaf intersects.

The key differentiating point, with respect to traditional approaches, is the possibility of mixing different types of geometries allowing choosing the best fitting shapes for each node composing the tree. It is also possible to balance the depth of the tree depending of the required precision weighted against the complexity of the model.

The position and orientation of the geometry of each node is associated with that of a node of the simulation environment scene graph. This association is a key point in the flexibility of the tree structure: a node can be associated with a different transformation from the one linked to his parent or children one. This allows the independent motion of each tree node.

The resulting structure can be seen as a scene graph used for collision detection rather than for rendering. The fact that this structure is explicitly built gives the possibility to access any node and modify its properties. For example the enclosing bounding volume of a multiple drill head can be adjusted depending of the status of the drill tools: when the simulation control logic receives the command to extend or retract the tools, it can also expand or shrink the enclosing bounding volume to take into account tools positions.

Moreover, with this flexible hierarchy is also possible to realize culling and Boolean operations on geometries: a parent-child relation defines, as a matter of fact, the volume resulting from the difference between their approximating volumes. A scene graph like structure allows verifying and fine tuning the resolution of the approximation: where a greater detail is needed the structure can be expanded, leaving the representation coarse-grained in the zone of low interest. In this way it is possible to balance the precision and the speed of the collision detection.

Finally, a great advantage of this structure is the low memory footprint, as each node is the best fitting shape and the depth and the trees don't exceed the necessary.

4.2. Collision structure editor

A collision structure editor has been developed to support the user during the definition of the collision structures. In this 3D application it is possible to load models from VRML files: keeping the hierarchical structure of shapes of this format, it is possible to easily indentify and select model parts. Once the user has selected one or more parts, he can create one of the available bounding volumes (box, sphere and cylinder). The user can also define a triangle mesh as bounding volume. The application computes the best fitting bounding volume for the selection and, in the case of the triangle mesh, creates its convex hull.

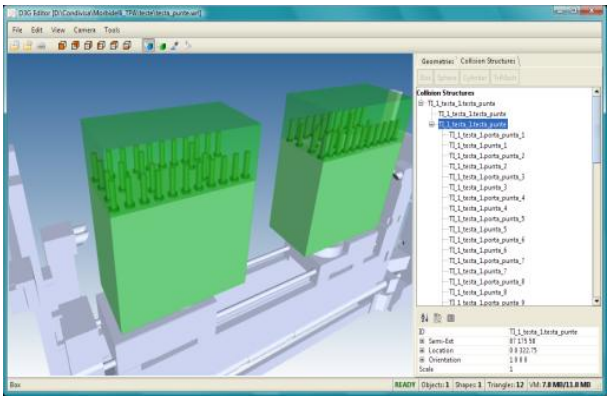

Figure 3: Collision structure editor

The bounding volumes created are the nodes of HMR hierarchy. These nodes can be arranged in the

desired structure using drag and drop operations on the tree view on the right side of the application.

The user can manually edit the properties of the primitives, such as extensions, location and orientation, to adjust the automatically computed values. That means that it is thus possible to define a bounding volume bigger that the fitting one: this is particularly useful when tolerances on movements are to be considered. Objects vibrations and dilatations are only simple examples of requirements bringing to slight enlargement of automatic fitting bounding volumes.

5. TEST CASES

In this section the results obtained by the described algorithms in two different applicative scenarios are reported. In both cases simulations were made on a common desktop PC (Pentium 4 3GHz, 2GB RAM, NVIDIA GeForce 6800 128MB video card) under a Windows XP SP3 operating system. The collision detection engine and algorithms are included, as pluggable modules, in a Java 1.6 simulation environment: thus all algorithms are implemented in Java. Performances measures, reported below, refers to this hardware and software configuration.

5.1. Machine scenario

The described test case concerns a real industrial scenario of a configurable hole punching machine.

5.1.1. Scenario description

This kind of tool-machine actually has up to 24 heads (12 for each of the two sides of the wood panel) and each head has up to 29 tools. The machine also has 5 bridges and 5 supporting structures to keep the panel in place, and 2 machine moving structures. The heads are mounted in pair on an axis that can move along the **x** direction, while each head can move along the **y** and **z** directions and rotate ±90 degrees around its axis. On the head each tool can be extended or retracted: thus the overall shape of the moving head is complex and dynamic: a schema of the machine is reported in [Figure 4].

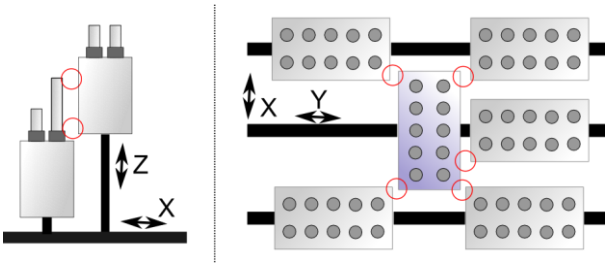

Figure 4: Test case machine heads schema. The red circles highlight possible collision in moving objects in the scene.

This scenario is particularly challenging for each collision algorithm because it has 732 different objects to be checked (267546 combinations of object to be tested), for a total of about 200 thousands triangles.

This amount of colliding pairs can be divided in collision groups (groups with objects worth to be checked for collision) to speed-up the overall test. For example the tools of a head cannot enter in contact with each other as they are firmly mounted on the head, and can thus belong to different collision groups.

A single head, as the blue one in [Figure 4], can collide with each of the heads of the other side of the panel, with the head on the same **y** axis and with the adjacent heads on the **x** axis. Tools can enter in contact with the heads and tools of the other side and with the adjacent tools and heads on their side, bringing the same number of objects to be checked as for the head. Moreover, the first and last heads have only one adjacent head on the **x** axis. The number of pairs of objects to be tested under these hypotheses is around 152 thousands.

Using the previously described method, the knowledge that heads and tools are moving together can be exploited. HMR allows seeing tools as children of a common parent bounding volume: these and the head are thus children of a common root bounding volume. This root can collide with the five adjacent ones and with the ones acting on the other side of the panel. Thus the number of objects to be checked in the first level of detection is around 400: only if collision is detected at first level HMR further checks each child of one hierarchy against each one of the other. Considering that most of the time we don't have collisions don't occur (as the CNC is able to generate safe paths in most of the cases), having a low first level number of objects, can strongly reduce the total number of checks.

The described machine, and its amount of colliding objects, has been used as reference environment for the performed test campaign and as benchmark for all described algorithms. Moreover, a common simulation has been set up: all the machine axes have been moved with a triangular speed motion law, bringing all the objects involved to collide at least once with the other movable objects.

5.1.2. Results

Benchmark simulation is formed of many steps including the update of the environment after the axis motion and a collision detection check. Number of primitive tests and time spent while testing are collected for each algorithm: results are in [Table 1].

Table 1: Results of tested methods for hole punching machine test case: triangles method results are omitted since not significant.

	BVT	*BVMix*	*FBA*	**HMR**
Box - Box	16565K	30707K	18654K	1675K
Sph - Sph	/	2115M	/	/
Tri - Tri	5115M	9279M	/	/
Collision	34099	34099	42062	32185
Time (s)	852,26	1673,69	5,81	4,47
FPS	0,12	0,06	17,21	22,39

Results obtained with Triangles algorithm are here omitted because not significant: time spent for collision isn't comparable with any other method tested.

Results show, as previously stated, that HMR including knowledge on machine in its collision structure is capable to perform better than other methods for collision detection. In particular, its frame rate makes HMR an *interactive* method for collision detection. It is worth notice that this result it is not reached with exact algorithms (Triangles, BVT or BVMix).

As a matter of fact BVMix, in this test case, performs worse than BVT: this is understandable considering that for the rough-edge geometries used BVT (using boxes as base bounding volumes) is capable to better approximate geometries than BVMix (that has spheres as hierarchy leaves). BVMix thus being more suitable for general geometries, as already explained, demotes its performance with squared bodies.

It is moreover useful to notice the number of detected collisions: HMR is supposed to detect more collisions, because it doesn't use the exactness of model triangles.

Grouping tools under a common bounding volume allows HMR to handle the case that when tools are retracted inside the head body, they cannot collide with anything outside the heads (parent bounding volume is sized to include only tool outside volume). For other methods, instead, they always collide with anything they are checked with. This, once again, makes HMR faster thanks to included machine knowledge and explaining the difference in the number of detected collisions.

5.2. Robot arms scenario

This test case concerns a common industrial application: the usage of robot arms acting simultaneously on the environment, for example in an assembly cell like the one in the simulation pictured in [Figure 5].

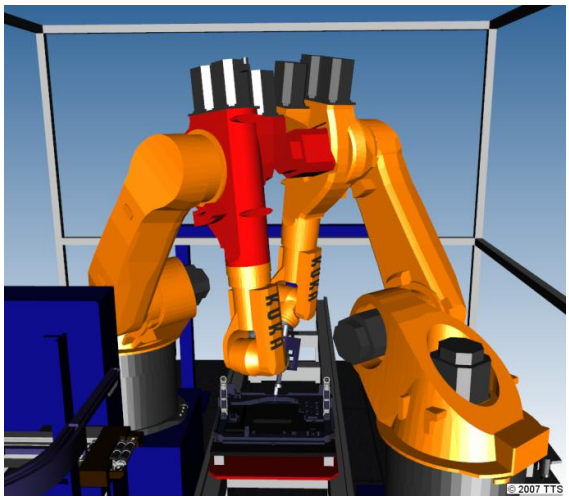

Figure 5: Robot arm scenario. Two robotic arms (KUKA KR 30 HA) colliding during an assembly task.

5.2.1. Scenario description

In this test case two robot arms are assembling a piece, acting on the two side of a conveyor system.

While assembling, it is possible that any of the links of each arm collide both with another of the same arm (*self-collision*) and with links of the other arm. A first kind of collisions is usually avoided thanks to physical joint limits or through properly writing each arm task; the other kind is usually managed in the simulation phase, especially when the number of arms involved increases.

This robotic scenario, even if common in industrial applications, is not particularly challenging for collision detection algorithms: each arm is formed by 6 links and 14 pieces to be checked for collision for a total of about 34 thousands triangles.

5.2.2. Results

Each simulation step requires the motion of all the joints in a task, forcing each arm link to collide with each other many times. Also for this test case, results obtained with Triangles algorithm are omitted because not significant.

Table 2: Results of tested methods for robot arms test case. Triangles method results are omitted since not significant.

	BVT	*BVMix*	*FBA*	**HMR**
Box - Box	307K	3026K	766K	197K
Sph - Sph	/	22M	/	/
Tri - Tri	954M	178M	/	/
Collision	6284	6284	7328	8060
Time (s)	160,501	33,451	0,222	0,109
FPS	6,23	29,89	4,5K	9,2K

This tests case, besides confirming what previously stated, shows also how BVT performances heavily can downgrade depending on the model, also if compared with another exact algorithm as BVMix.

The arm model is composed by many small triangles coupled with some long ones: when a BVT leaf is selected for primitive test, it participates with many elements, many of which are useless for detecting collisions. BVMix leaf spheres generally enclose much less primitives to be tested only when needed: being small are more punctual bounding volumes.

6. CONCLUSION

In this paper a hierarchical fast and reliable collision detection method based on an hybrid approach is presented. This novel technique, called HMR, is capable both to take into account the geometry structure of a model (using different kinds of bounding volumes), and to include knowledge on the model kinematics peculiarities and physical constraints.

The capability to include model knowledge, rising from the organization of bounding volumes in

hierarchies, allows HMR to achieve interactive collision detection with a low memory usage.

The stratification of bounding volumes represents for HMR a smart multi resolution approximation of moving objects: this representation comes directly from the human experience. Objects representation and organization in parent-child hierarchies is created by the user with the help of a visual editor.

Further investigations will be directed on exploiting HMR abilities to use more knowledge, e.g. reusing knowledge related to a single part of the model, to speed up the collision detection on other model parts. Another possible improvement will allow HMR to deal with other bounding volumes types.

REFERENCES

Baraff, D., Witkin, A., Anderson, J., Kass, M., 2003. Physical based modelling. *SIGGRAPH Course Notes*.

Benes, B., Villanueva, N. G., 2005. GI_COLLIDE-Collision Detection with Geometry Images. *Proceedings of the Spring Conference on Computer Graphics*, 95-102

Bergen, G., 2003. Collision Detection. *Interactive 3D Environments. Interactive 3D Technology Series*, Morgan Kaufmann ed.

Carey, R. & Gavin, B., 1997. *The Annotated VRML 2.0 Reference Manual.* Addison-Wesley

Cignoni, P., Costanza, D., Montani, C., Rocchini, C., R. Scopigno, 2000. Simplification of Tetrahedral Meshes with Accurate Error Evaluation. *Proceedings of the conference on Visualization*, 85-92

Dingliana, J., O'Sullivan, C., 2000. Graceful Degradation of Collision Handling in Physically Based Animation. Proceedings Eurographics Computer Graphics Forum, 19: 239-247

Ehmann, S. A., Lin, M., 2000. Accelerated Proximity Queries between Convex Polyhedra by Multi-level Voronoi Marching. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3 : 2101-2106

Ehmann, S. A., Lin, M., 2000. *Swift: Accelerated Proximity Queries between Convex Polyhedra by Multi-level Voronoi Marching*. Tech report. Department of Computer Science, University of North Carolina at Chapel Hill.

Ericson, C., 2005. *Real-Time Collision Detection.* Morgan Kaufman.

Fischer, K., Gartner, B., 2003. The Smallest Enclosing Ball of Balls: Combinatorial Structure and Algorithms. *Proceedings of 19th Annual Symposium on Computational Geometry (SCG)*, 291-301

Gottschalk, S., 2000. *Collision Queries using Oriented Bounding Boxes*. PhD Thesis. Department of Computer Science, University of North Carolina.

Govindaraju, N., Lin, M., Manocha:, D., 2004. *Fast and Reliable Collision Detection Using Graphics Hardware*. Tech report. Department of Computer Science, University of North Carolina at Chapel Hill.

Gundelman, E., Bridson, R., Fedkiw, R, 2003. Nonconvex Rigid Bodies with Stacking. *Proceedings of ACM SIGGRAPH*

Heidelberger, B., Teschner, M., Gross, M., 2004. Detection of collisions and self-collisions using image-space techniques. *Journal of WSCG*, 12 : 1-3

Hubbard, P. M., 1993. Interactive Collision Detection. *Proceedings of IEEE Symphosium on Research Frontiers in Virtual Reality*, 24-31.

Ketchel, J. S., Larochelle, P. M., 2006. Collision Detection of Cylindrical Rigid Bodies for Motion Planning. *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, 1530-1535.

Knott, D., Pai, D., 2003. Cinder: Collision and Interference Detection in Real-Time Using Graphics Hardware. *Proceeding of Graphics Interface*, 73-80.

Kockara, S., Halic, T., Iqbal, K., Bayrak, C., Rowe, R., 2007. Collision detection: a survey. *IEEE International Conference on Systems, Man and Cybernetics*, 4046-4051

Lin, M., Canny, J. A, 1991. Fast algorithm for incremental distance calculation. *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*

Lin, M., Gottschalk, S., 1998. Collision Detection between Geometric Models: A Survey. *Proceedings of IMA Conference on Mathematics of Surfaces*, 37-56

Lin, M., Manocha, D., 1994. *Efficient Contact Determination between Geometric Models*. Tech report. Department of Computer Science, University of North Carolina at Chapel Hill.

Mirtich, B., 1996. *Impulse Based Dynamic Simulation of Rigid Body Systems*. PhD Thesis. University of California, Berkley

Mirtich, B., 1998. V-Clip: Fast and Robust Polyhedral Collision Detection. *ACM Transactions on Graphics*, 17 : 177-208

Moore, M., Williams, J., 1998. Collision Detection and Response for Computer Animation. *Computer Graphics*, 22 : 289-298

Pedrazzoli, P., Sacco, M.; Jönsson, A., Boër, C.R., 2006. Virtual Factory Framework: key enabler for future manufacturing. *Digital enterprise technology: perspectives and future challenges*, Springer, 1, 83-90

Yoon, S., Salomon, B., Lin, M., Manocha, D., 2004. Fast collision detection between massive models using dynamic simplification. *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 71 : 136 -146