Hierarchical structures for collision checking between virtual characters

Published in Computer Animation and Virtual Worlds 2014; **25**:333–342

Sybren A. Stüvel[1], Nadia Magnenat-Thalmann[2], Daniel Thalmann[2], Arjan Egges[1], and A. Frank van der Stappen[1]

[1]VHTLab, Universiteit Utrecht
[2]Institute for Media Innovation, Nanyang Technological University

June 4, 2014

Figure 1: A crowded pavement.

Abstract

Simulating a crowded scene like a busy shopping street requires tight packing of virtual characters. In such cases collisions are likely to occur, and the choice in collision detection shape will influence how characters are allowed to intermingle. Full collision detection is too expensive for crowds, so simplifications are needed. The most common simplification, the fixed-width, pose-independent cylinder, does not allow intermingling of characters, as it will either cause too much empty space between characters or undetected penetrations. As a possible solution to this problem, we introduce the *bounding cylinder hierarchy* (BCH), a bounding volume hierarchy that uses vertical cylinders as bounding shapes. Since the BCH is a generalization of the single cylinder, we expect that this representation can be easily integrated with existing crowd simulation systems. We compare our BCH with commonly used collision shapes, namely the single cylinder and oriented bounding box tree, in terms of query time, construction time and represented volume. To get an indication of possible crowd densities, we investigate how close characters can be before collision is detected, and finally propose a critical maximum depth for the BCH.

[1]e-mail:{s.a.stuvel,j.egges,a.f.vanderstappen}@uu.nl
[2]e-mail:{nadiathalmann,danielthalmann}@ntu.edu.sg

1 Introduction

Crowd simulation plays an ever increasing role in different fields. When a crowd consists of densely packed characters, such as shown in Figure 1, collision detection on all characters can become computationally expensive. A brute-force approach to collision detection would check every primitive of one character with all primitives of the other. For a real-time crowd this becomes unattainable, and a smarter approach is needed. Most agent-based crowd simulation systems use a simplification where crowd agents are modeled as a cylinder sliding on a ground plane, animating a character inside this cylinder using a walk cycle [PAB07]. For sparse to medium-density crowds, this method is often sufficient. However, when the density of the crowd increases such that the movement of an agent may be impaired, a more precise representation of the space that virtual characters occupy is needed.

Collision detection typically occurs in two phases [vdB04]. The *broad phase* finds pairs of objects that may collide and eliminates pairs that are far away from each other, usually employing space partitioning structures such as voxel grids [Rey06] or space partitioning trees [VLnG10]. The *narrow phase* takes these pairs of objects, and performs the actual collision test. It is in this phase that the techniques described in this paper are applied.

To calculate a physically plausible collision response, the force of the collision needs to be known. Collision is usually only detected after two objects have some measure of interpenetration, and the penetration depth is then used as a measure of the collision force [HTKG04]. Calculation of this penetration depth requires a volumetric representation of the colliding objects. However, most applications use a boundary representation where objects consist of a polyhedral mesh. In this case we are limited to detecting intersections of the boundaries of the objects.

In this paper we introduce the *bounding cylinder hierarchy* (BCH). We set criteria for collision handling between virtual characters in a crowd, based on query performance, construction speed and represented volume. We use this to compare the following collision detection structures: the oriented bounding box tree, our BCH, and the shape most widely used in crowd simulation, the single cylinder. Based on the outcome of a comparative experiment, we show which collision structure is suitable for which use case. We represent characters by a polyhedral mesh in which the vertices are defined in a object-local coordinate frame.

Related work is discussed in the next section. Section 3 formalizes bounding volume hierarchies for collision detection, and defines the bounding cylinder hierarchy (BCH) and OBB tree within that formal definition. Section 4 details our experimental comparison between the single cylinder, the BCH and OBB tree. In Section 5 we investigate different properties of the BCH. We discuss possible future work in Section 6, and conclude in Section 7.

2 Related Work

Collision detection is a widely researched topic. Applications range from robotics via computer aided design and manufacturing to crowd simulation. It is common that simplified shapes are used in order to reduce computational complexity, such as a set of bounding boxes for the head, the torso and the limbs. By using algebraic definitions, such shapes can be intersected to approximate the penetration depth. However, finding the correct algebraic shapes that tightly fit a given mesh can be cumbersome.

A different approach to speed up the narrow phase is the use of model partitioning techniques such as bounding volume hierarchies (BVHs) that allow for quick determination of non-intersection. Commonly used BVHs are sphere trees [Hub96], axis-aligned bounding box trees [vdB97], oriented bounding box trees [GLM96] and k-DOP trees [KHM+98].

Other authors have also compared different collision shapes. Gottschalk et al. [Got00] proved that oriented bounding box (OBB) trees are superior over sphere trees and axis-aligned bounding box (AABB) trees for surface based BVHs. Their method is widely used, and will be used for comparison with the bounding circle hierarchy. Van den Bergen showed that AABB trees are easy to adjust after the object has been deformed, so for real-time adaptation of the collision hierarchy AABB trees are preferred [vdB97]. Both box-based methods use a bounding shape with straight edges and square corners, which does not form a good match for the human shape. This in itself is not an issue as

the hierarchy contains all the necessary details, but when the maximum recursion depth is limited (as we investigate in Section 5) this becomes relevant. Sphere trees do not have such square corners, and have been proven useful for the estimation of penetration depth [OD99]. Collision tests on spheres are simple as they are independent of the character's rotation. However, a sphere bounding a virtual character would contain much empty space.

For a more detailed survey on collision detection techniques we refer to the work of Kockara et al. [KHI+07].

In this paper we extend the work of Stüvel et al. [SES+13] by introducing the *bounding cylinder hierarchy* for discrete collision detection. The cylinder is the prevalent shape in crowd simulation, and is widely accepted as a rough approximation of the human shape. By employing a hierarchical refinement strategy, we ensure that the BCH represents much tighter fit than possible with a single cylinder. The cylinder's rotational symmetry allows for fast intersection tests and efficient storage.

3 Hierarchical structures for collision detection

In the previous section, we have described a family of bounding volume hierarchies that are commonly used for collision or interference detection. This section presents a formal definition of such structures, providing us with a common reference frame to compare members of this family. We introduce the bounding cylinder hierarchy as a hierarchical generalization of the commonly used cylinder, and redefine the oriented bounding box tree within the terms of our reference frame. Section 4 will use these definitions in a comparative experiment.

For an object P the ingredients for a hierarchical collision structure $\mathcal{H}(P)$ are:

1. A family of shapes such as cylinders, boxes or spheres;
2. A finite tree structure, where every node ν contains a bounding volume $B(\nu)$ of the aforementioned shape family, and represents a sub-object $P(\nu) \subset P$;
3. A subdivision strategy, defining nodes in tree layer $i+1$ given the nodes in layer i;
4. A stop criterion for the subdivision.

Let ν be any node in the tree, and $C(\nu) = \{\mu_1, \ldots, \mu_k\}$ denote its child nodes. We impose the following requirements for the hierarchical structures we consider in this paper, where $int(X)$ denotes the interior of X.

1. $P(\nu) = \bigcup_{\mu \in C(\nu)} P(\mu)$
2. For all $\mu, \mu' \in C(\nu)$, $\mu \neq \mu'$: $int(P(\mu)) \cap int(P(\mu')) = \emptyset$

So in other words, $P(\nu)$ is partitioned into the sub-objects associated with the members of $C(\nu)$. A crucial property of bounding volumes is that for any query object Q and node ν, if $Q \cap B(\nu) = \emptyset$ then $Q \cap P(\nu) = \emptyset$.

The above properties rule out certain bounding volume hierarchies, such as using bounding boxes for torso (root node) and head, arms and legs (child nodes), or the elliptical and cylindrical collision shapes by Dube et al. [DTT11]

3.1 Bounding Cylinder Hierarchy

In this section we introduce the Bounding Cylinder Hierarchy (BCH). It refines the cylindrical representation commonly used in crowd simulation. Note that in this paper we use *bounding* shapes for collision detection of characters (and parts of characters), which is not always the case in the field of animation; often approximations are used that do not necessarily contain the entire character, in order to artificially allow increased crowd densities at the expense of realism. We define the structure $\mathcal{BCH}(P)$ as:

1. A vertical cylinder is used as the bounding shape $B(\nu)$.
2. The tree is binary; the root represents the entire object P with its smallest enclosing cylinder.
3. We consider the projections of $P(\nu)$ against the two horizontal global coordinate axes; the axis for which the projection has the largest extent defines the normal of a separation plane. $P(\mu_1)$ and $P(\mu_2)$ are defined as a partition of $P(\nu)$ by that plane – details are given below.

   $B(\mu_i)$ is defined as the smallest enclosing cylinder of $P(\mu_i)$.

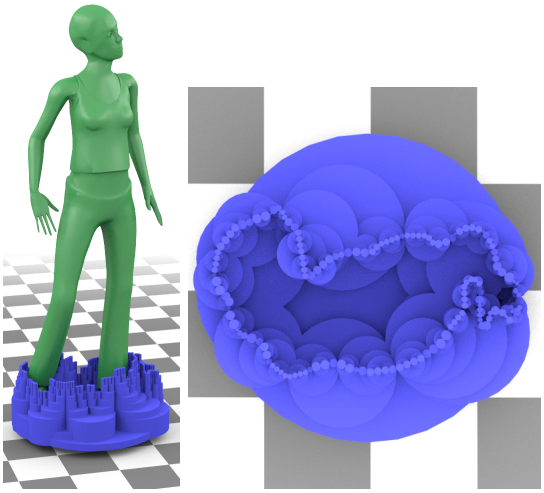

Figure 2: Visualization of the *contour* BCH.

4. When the radius of $B(\nu)$ is less than a predefined threshold, subdivision stops.

$P(\mu_1)$ and $P(\mu_2)$ are separated by the vertical plane, hence their interiors are disjoint. As $P(\mu_1) \cup P(\mu_2) = P(\nu)$, it follows that $P(\nu)$ is partitioned properly. We have investigated three methods of representing $P(\nu)$ and chosen appropriate subdivision schemes accordingly:

Contour: In this approach we only consider the contour (from the top view) of $P(\nu)$. The separation plane is then defined by its normal (as described earlier), and the centroid of the contour polygon. The centroid is commonly used, as it often results in a more balanced tree and better query performance. We have chosen to use the centroid of the projection rather than of the mesh vertices, in order to prevent bias to more detailed or heavily overlapping areas.

The contour boundary is interpreted as a sequence of line segments $S = \{S_0, \ldots, S_n\}$. For each segment the centre point is calculated; depending on the side of the separating plane the centroid lies on, it is used to define $P(\mu_1)$ or $P(\mu_2)$, taking care that neither set will be empty.

When S only contains a single line segment, and the segment is longer than twice the threshold radius, the line segment is split into two halves and decomposed as described above. This results in very little overlap between the cylinders associated with the leaf nodes of the hierarchy, as shown in Figure 2.

Contour (centre): This approach is almost identical to the *contour* approach, differing only in that it uses the centre point rather than the centroid.

All projected triangles: In this method we eliminate the need to calculate the contour explicitly, by simply including all triangles into the hierarchy. This will allow for faster construction at the cost of an increase of the query time. This approach mimics the decomposition used by Gottschalk et al.[GLM96]. A triangle is never subdivided into smaller pieces, but is placed into a subdivision $P(\mu_i)$ depending on the position of its centroid with respect to the separating plane. The separation plane is chosen as described above, except that the centroid of the mesh vertices is used. Subdivision stops when a node contains a single triangle. Since they are not subdivided, we cannot get arbitrarily small cylinders. However, we do have the ability to perform efficient, exact intersection tests as every leaf in the tree contains exactly one triangle.

Single cylinder: This method could be described as the *all projected triangles* method, but using an infinitely large number of triangles for the stop criterion. This results in a hierarchy with one node, containing only a single bounding cylinder. As this form is so common, we handle it as a special case.

A binary split was chosen in favour of a split into four parts, as the latter can result in longer, thinner subdivisions that are not a good match for a cylindrical bounding shape. In our implementation we consider the bounding cylinders to be of infinite height, effectively ignoring the height of the character. This is common practice in crowd simulations. Future development could include the height information in intersection tests.

3.2 Oriented Bounding Box tree

The OBB tree by Gottschalk et al.[GLM96] is a well-known and widely used method for collision

detection, and thus forms a good comparison for the BCH. It follows the formal definition given at the start of this section:

1. An oriented box is used as the bounding shape.
2. The tree is binary; the root represents the entire object P with its oriented bounding box.
3. Triangles of the mesh are stored in the leaves, based on which side of a separating plane their centroid lies. For the exact spatial subdivision rules we refer to [GLM96].
4. When a node contains only a single triangle, subdivision stops.

At every internal node, every triangle in $P(\nu)$ is represented by exactly one child node. This ensures that $P(\nu)$ is partitioned properly. At the leaf nodes, the OBB collision test performs triangle-triangle intersection tests. We define three techniques to construct the OBB tree:

Full: The OBB tree is populated with the triangles of the mesh. This is an exact representation of the mesh.

Contour: We calculate the contour of the mesh, as in Section 3.1. The line segments that make up this contour are used to populate the OBB tree.

All projected triangles: We use all triangles projected onto the ground plane to populate the hierarchy, as in Section 3.1. Our hypothesis is that this will be less efficient to query than the *contour* case but faster to construct. It may be faster to query than the *full* case.

4 Comparison

In order to compare the BCH and OBB tree, we perform two timing experiments. Each experiment uses two character meshes, one male and one female (see Figure 4) of approximately 2850 triangles each. We use motion capture recordings totalling 212 seconds of walking, turning, side-stepping and idling motions. For each of the two meshes, we randomly select 100 motion capture frames, resulting in a total of 200 randomly posed character meshes. For the experiments, those are regarded simply as a collection of triangles; the fact that they were posed using an articulated structure was of no relevance, and we do not use any metrics that depend on such a structure. Tests are run multiple times to reduce jitter and average out external factors, on an Intel Core i7 3630QM 3.2 GHz laptop. The BCH radius threshold is set at 1 cm.

We represent characters by their boundaries and do not see them as solids; we consider two objects as colliding when their boundaries intersect. This means that we will not be able to detect the case where one character completely envelopes another. In our intended scenario this will not be an issue, as collision will be detected before this can happen, if at all.

4.1 Experiments

In our first experiment we measure the time required to construct the representations for each of the 200 random poses. The time to load the mesh data from disk is excluded from the test. The results are shown in the left bar chart of Figure 3. The "BCH single cylinder" column shows the time needed to calculate the bounding cylinder for the posed mesh.

The second experiment measures the duration of intersection tests. Each of our 500 test cases is created by selecting two random poses, a random translation over the ground plane and a rotation about the up-vector. The process of randomization is repeated until a true collision is detected using the exact *full* OBB method. This presents us with a worst case test set, as negative tests often do not require descending the entire tree where positive tests do. All tests are binary, i.e. only report whether an intersection is found.

We run 10000 test iterations, where each iteration tests all 500 test cases sequentially, preventing over-optimistic results due to caching. To illustrate the effect of caching, we have performed the experiment in a different order by running each individual test case 10000 times. This resulted in an approximately 15% better performance, but as this would not reflect a real use case, it will not be included in our results.

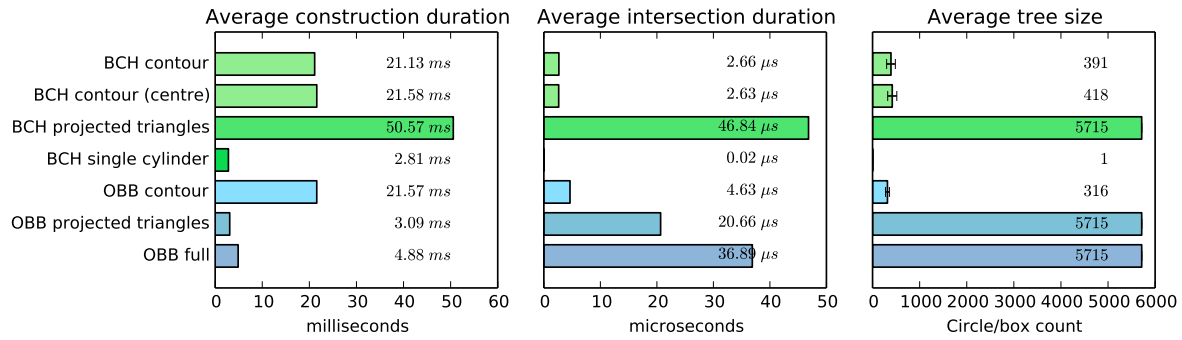

Figure 3: Comparison between the BCH and OBB tree variants.

4.2 Conclusion

The experiments show a query time of 2.66 μs for the *BCH contour* method, and 4.63 μs for the *OBB contour* method; in our scenario the BCH is 74% faster to query than the OBB tree. In general, the amount of overlap between cylinders of the same hierarchical layer of a BCH is higher than the overlap between boxes in the corresponding OBB tree. However, the simplicity of intersection tests between cylinders makes up for this when using our *contour* approach. The more or less fat shape of the projected mesh is a reasonably good fit for the cylinders, whereas the OBB tree is more efficient for long, thin shapes such as individual triangles. This is clearly shown in the difference in intersection speed between the BCH and OBB *all projected triangles* cases, where the trees are of the same size and both well balanced, but due to the lesser amount of overlap the OBB tree is more than twice as fast to query.

There is no significant difference between the *centre* and *centroid* BCH methods, with respect to construction and query duration. Since using the centroid results in a smaller tree, the rest of this paper will use this approach.

For interactive, multi-character purposes, use of the BCH requires a preprocessing step in which the data structures are calculated, as this cannot be done at interactive rates for multiple characters (yet). However, the resulting structure is nearly 8 times faster to query than the 20.66 μs of the fastest-to-construct *projected triangles* OBB method.

The OBB tree provides an exact collision test, whereas the precision of the BCH depends on the threshold radius. For cases where exactness is more important than performance the OBB tree would be a more suitable representation. We measure this in Section 5.

The implementation by Gottschalk et al. was used in all three OBB cases. The *contour* tests were performed using this 3D implementation, where the contour line segments are input as degenerate triangles. A pilot experiment showed that this did not cause significantly different results compared to using non-degenerate triangles. Our experimental results thus reflect a worst case scenario; an alternative implementation optimized for two-dimensional shapes could provide different results but was not available at the time of writing.

The BCH reports a collision when the cylinder at a leaf node collides with another cylinder at some leaf node. By reporting a collision when a given maximum recursion depth is reached instead, we can allow for a *level of detail* behaviour. A possible use case would be in a crowd simulation scenario. Close to the camera more precision is required than further away in the crowd, as most of the collisions will be occluded. Crowd density could also be used to influence the maximum recursion depth, reverting to a single cylinder for low-density crowds. A similar technique could be applied to the OBB tree, although at coarser levels the corners of the bounding boxes may induce unnatural behaviour and visual artefacts. At the coarsest level the BCH collapses to a single cylinder, which has already been proven to be useful for crowd simulations.

These experiments are purely quantitative, and do not consider the quality of the results. We in-

Figure 4: The two characters we used for testing. We removed non-manifold edges such as hair strands to allow for volume calculations.

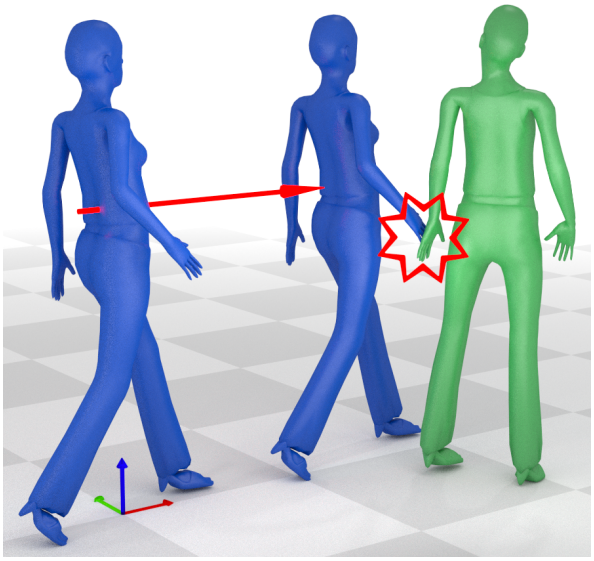

Figure 5: The distance test, where one character is slid towards the other until collision is detected.

tend to perform a qualitative experiment in the future.

5 Properties of the BCH

In the previous section we have shown that the BCH is a suitable collision structure for virtual characters. This section investigates the BCH further. One of the goals of this paper is to look at ways to effectively deal with crowds of high density. We look at the effect of employing a maximum recursion depth for intersection tests, to see how much closer meshes can get to each other by using a more detailed representation.

In real life, a collision between two people occurs when there is a collision between the two occupied volumes. The tighter the collision shapes fits this volume, the smaller the chance of false positives. In Section 5.2 we investigate the represented volumes of the BCH and compare those to the commonly used single cylinder.

These experiments have been performed using the same 200 randomly posed meshes as described in Section 4. Also for these experiments, those were regarded simply as a collection of triangles.

5.1 Minimum distance

Our premise was that a better collision shape would allow for tighter packing of virtual characters as there are less false positives detected. In this section we investigate the minimal distance between two characters before we detect a collision. Often the root joint is used as *the* position of a character, and the planar difference in root joint positions denotes the distance between them. However, the choice of root joint can be arbitrary, and the resulting distance is depending on the pose. Instead, we use the minimal distance between the two meshes – when this distance is zero, there is a collision.

We use a coordinate frame where the X- and Y-axes span the ground plane. In our experiment we randomly orient two posed meshes. We place one at the origin, and the other at $(2, y)$ where $y \in [-0.5, 0.5]$ is chosen such that the meshes will collide. We then slide the first mesh along the positive X-axis, until they collide, as shown in Figure 5. We chose this approach over explicitly calculating the distance at which a collision would occur, as our approach requires no knowledge of the actual collision detection algorithm and is very simple to implement. We repeated this process for all 200 meshes in a total of 500 random orientations, and

for several collision detection approaches.

The bar chart in Figure 6a shows the average minimum distance before collision for the minimum bounding cylinder, the *contour* BCH and *contour* OBB tree. The *full* OBB method is not included in the figure, as this is an exact collision detection scheme, so the distance is zero with zero standard deviation.

At an average minimum distance of 5 cm, the *BCH contour* method allows characters to get significantly closer to each other than the 20 cm allowed by a single cylinder. The situation shown in Figure 1 is impossible when representing characters with a single cylinder. However, even in this dense situation the contours do not intersect.

On average the *OBB contour* allows for a minimum distance of 4 cm, which is 1 cm closer than the *BCH contour* allows. This is explained by the fact that the BCH is an approximation; in our experiment we used a threshold radius of 1 cm, which is reflected in these results.

To investigate the *level of detail* possibilities of the BCH we limit the recursion depth of collision tests. Figure 6b shows the distance at which a collision is detected given a maximum recursion depth d. The single cylinder always has the same distance as $d = 0$. A higher d results in a tighter fit of the mesh, except in some specific cases (for example the small peak in one of the test cases at $d = 3$). At $d = 9$ the distance is only 1% larger than at $d = \infty$, so for many practical purposes this represents a critical maximum recursion depth. As the number of reachable nodes in the tree is $O(2^d)$, imposing such a limit will increase query performance.

5.2 Represented volume

When checking for collisions using simplified shapes, one aims to use a collision shape that approximates the actual shape in a sufficiently precise manner. To see how well a cylinder matches a human shape, we have compared for all 200 random poses: the smallest enclosing axis-aligned cylinder and the volume represented by the BCH. We normalized the volumes by the volume of the mesh, to make it easier to compare between meshes of different sizes.

We posed the meshes using linear blend skinning, also known as skeletal subspace deformation. This is a conventional skinning method that does not preserve volume [GB08]. However, we do not use heavily deforming poses; we measured a standard deviation of 0.01 m^2. The average of the measured volumes was used for normalization. We define the volume represented by a BCH as the volume of the union of the cylinders stored in the leaf nodes of the *all projected triangles* representation. For simplicity we use a discretization by drawing the bases of the cylinders onto a high-resolution image (> 6 megapixels/m^2), counting the drawn pixels and multiplying with the character's height.

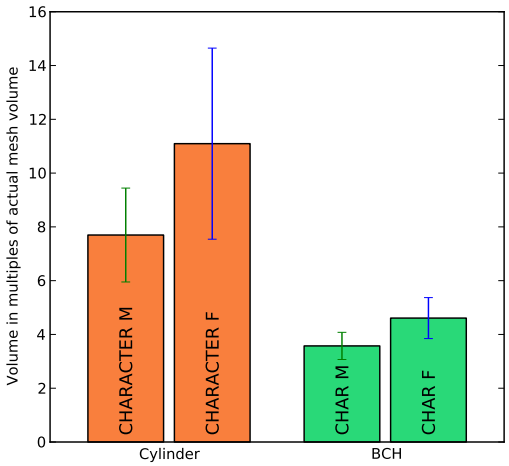

Figure 7: Volumes occupied by the minimal bounding cylinder and by the BCH.

The results can be seen in Figure 7, with numeric data in Table 1. From this, we can observe that the cylinder may not be the best representation of the human shape, as it represents a volume approximately seven times larger than the actual character.

The radii most often used in literature are around 0.25 m [KGO09, vTCG12, HFV00], or around 0.40 m [Ger10, AMTT12], and sometimes up to 2.00 m [GKLM11]. We used character models that are slightly taller than the world average (1.74 m for the female character, 1.88 m for the male character). The average radii we have found for the random poses are consistent with what is used in literature. However, we can safely say that for average-sized characters a radius of 0.25 m will cause unnoticed intersections, whereas a radius of much more than 0.40 m will cause more spacing between characters than strictly necessary.

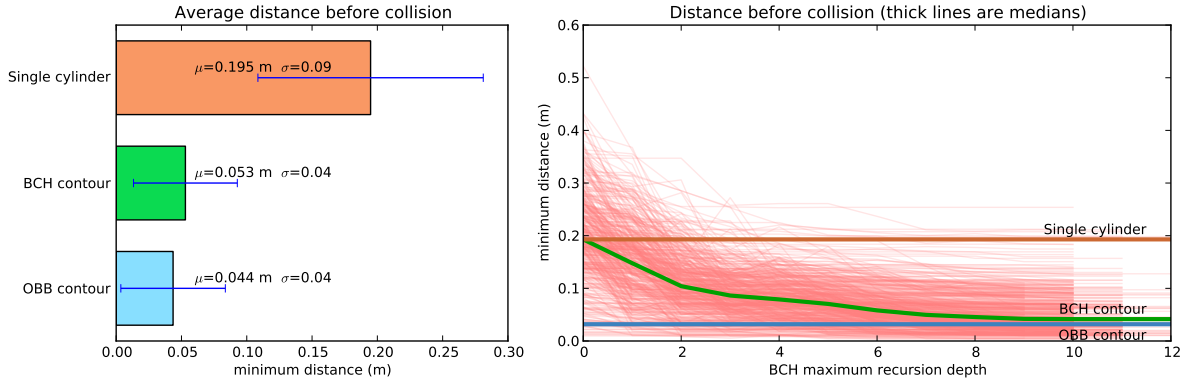

Figure 6: Minimum distance between meshes before collision. (a) The bar chart shows the average distance between two characters at which a collision would occur. (b) The graph shows a thin red line for each test case and thick lines for medians. The dark green, curved line shows the median minimal distance obtained with the BCH.

		Female character		**Male character**	
Cylinder radii (m)	average	0.34		0.42	
	median	0.31		0.40	
	stddev	0.05		0.05	
Volumes (normalized)		**Cylinder**	**BCH**	**Cylinder**	**BCH**
	average	11.09	4.61	7.70	3.57
	median	9.50	4.27	6.99	3.34
	stddev	3.55	0.76	1.75	0.51

Table 1: The observed radii of the minimal spanning cylinders, and the difference in volumes occupied by those cylinders and the BCH. All volumes have been normalized by the mesh volume.

6 Future work

In our current implementation, the vertical plane used in the construction of the BCH is always aligned with a horizontal coordinate axis, as suggested by Kolowski [J.T98]. Using a similar covariance analysis as used by Gottschalk in the OBB construction could result in a more optimal subdivision and faster query times, at the expense of slightly longer construction times.

The cylinders in the BCH span the full height of the pose. To facilitate a closer fit, we could horizontally slice the mesh, and create a BCH for each slice. Possible slices would be for the legs, torso and head.

We have limited our comparison to static poses. The BCH for one pose is completely unrelated to the BCH of the next pose of the character. We are interested in investigating an optimization technique to find a set of animated cylinders that fit an entire animation. Not only would this allow for more efficient storage, but such temporal cohesion would also make interpolation between motions possible. The method by Van Basten et al. [VBSE11] is an example in which predefined animation segments are interpolated and where such preprocessing is possible.

We performed a quantitative study of collision shapes. We intend to follow up with a qualitative study, in order to find out how well people can recognise collisions in static poses, and at what inter-character and camera-character distances. We suspect that using animations rather than static poses may skew results, depending on whether a collision response is shown or not.

7 Conclusion

We have introduced a bounding cylinder hierarchy for efficient collision checking with a flexible level of detail. We have compared it with the commonly used OBB tree, and seen that in the scenario we tested – humanoid character collisions – the BCH provides a better query-time performance.

We suspect that the BCH is more resistant to an increase in detail of the model than the OBB tree. The latter typically contains all triangles of the mesh, and thus depends on the mesh complexity. The BCH complexity is bound by the threshold radius, providing tunable mesh simplification for collision checking.

In comparison to single-cylinder systems, we have consistently used the smallest bounding cylinder of the posed character. This resulted in a worst case scenario, as often in crowd simulations a fixed pose-independent radius is used. In such a case, visualizing a dense crowd is guaranteed to show undetected interpenetration of body parts, which we want to avoid. We have shown that the volume that is represented by the BCH is much smaller than that of a single bounding cylinder, and that the BCH also allows characters to move much closer together before collision is detected.

Authors' biographies

Sybren A. Stüvel is a PhD student at the Virtual Human Technology Lab in the Department of Information and Computing Sciences, Utrecht University, The Netherlands. His main research topic is the animation of crowds of virtual characters. He obtained his Master's degree in Computer Science in 2010 at the same research group, and developed a method for the generation of human locomotion based on footstep positions. He is also the co-author of the IEEE Handbook of Digital Games, and part of the COMMIT project.

Professor Nadia Magnenat Thalmann has pioneered research into virtual humans over the last 30 years. She obtained several Bachelor's and Master's degrees in various disciplines (Psychology, Biology and Biochemistry) and a PhD in Quantum Physics from the University of Geneva in 1977. From 1977 to 1989, she was Professor at the University of Montreal in Canada.

In 1989, she moved to the University of Geneva where she founded the interdisciplinary research group MIRALab (www.miralab.ch). Her global research area is Virtual Human and Social Robot. She has acquired a great experience of collaborative research through her strong participation to more than 50 European Research Projects.

Together with her PhD students, she has published more than 500 papers and books on research topics such as 3D clothes, hair, body gestures, emotions modelling, and on medical simulation.

She is Editor-in-Chief of The Visual Computer Journal published by Springer Verlag, Co- Editor-in-Chief of the Journal Computer Animation and Virtual Worlds published by Wiley and Associate Editor of many other scientific journals.

During her Career, she has received several Awards such as "Woman of the Year", an early recognition in Montreal in 1987. Among the recent ones, she was awarded a Doctor Honoris Causa in Natural Sciences from the Leibniz University of Hanover in Germany in 2009, the Distinguished Career Award from the European Association for Computer Graphics in Norrkoping, Sweden in 2010, an Honorary Doctorate of the University in Ottawa in 2010 and a Career Achievement Award from the Canadian Human Computer Communications Society in Toronto in 2012 and in 2013, the Humboldt Research Award.

Besides directing her research group MIRALab in Geneva, she is presently Visiting Professor and Director of the Institute for Media Innovation (IMI) at Nanyang Technological University in Singapore.

Prof. Daniel Thalmann is with the Institute for Media Innovation at the Nanyang Technological University in Singapore. He is a pioneer in research on Virtual Humans. His current research interests include Real-time Virtual Humans in Virtual Reality, crowd simulation, and 3D Interaction. He is coeditor-in-chief of the Journal of Computer Animation and Virtual Worlds, and member of the editorial board of 6 other journals. Daniel Thalmann was member of numerous Program Committees, Program Chair and CoChair of several conferences including IEEE VR, ACM VRST, ACM VRCAI, CGI, and CASA. Daniel Thalmann has published more than 500 papers in Graphics, Animation, and Virtual Reality. He is coeditor of 30 books, and coauthor of several books including 'Crowd Simulation' (second edition 2012) and 'Stepping Into Virtual Reality' (2007), published by Springer. He received his PhD in Computer Science in 1977 from the University of Geneva and an Honorary Doctorate (Honoris Causa) from University Paul- Sabatier in Toulouse, France, in 2003. He also received the Eurographics Distinguished Career Award in 2010 and the 2012 Canadian Human Computer Communications Society Achievement Award.

Arjan Egges is an Associate Professor at the Virtual Human Technology Lab in the Department of Information and Computing Sciences, Utrecht University in the Netherlands. He obtained his PhD at MIRALab - University of Geneva, Switzerland on the topic of emotion and personality models, in combination with automatically generated face and body motions using motion capture data. His current research focuses on crowd animation and motion perception as a part of the COMMIT and TARDIS projects. He is also a member of the Games and Learning Alliance (GaLA). He teaches several courses related to games and computer animation. Arjan is also an associate editor of the Computer Animation and Virtual Worlds journal published by Wiley and he is one of the founders of the annual Motion in Games conference.

A. Frank van der Stappen received the M.Sc. degree from Eindhoven University of Technology, Eindhoven, The Netherlands, in 1988 and the Ph.D. degree from Utrecht University, Utrecht, The Netherlands, in 1994. He is currently an Associate Professor with the Department of Information and Computing Sciences, Utrecht University. His research interests include manipulation, motion planning, grasping, simulation, animation, and geometric algorithms.

Acknowledgements

This research is supported by the Dutch nationally funded project COMMIT. Part of this work has been performed during the stay of Sybren A. Stüvel and Arjan Egges at Nanyang Technological University, Singapore. Contour calculations were performed using the Clipper library [Ang14]. All graphs were made with Python, Matplotlib and Numpy. All other images were made with Blender.

References

[AMTT12] B.F. Allen, N. Magnenat-Thalmann, and D. Thalmann. Politeness improves interactivity in dense crowds. *Computer Animation and Virtual Worlds*, 23(6):569–578, 2012.

[Ang14] Angus Johnson. Clipper - an open source freeware library for clipping and offsetting lines and polygons. http://www.angusj.com/delphi/clipper.php, 25 March 2014. Accessed 2014-03-25.

[DTT11] C. Dube, M. Tsoeu, and J. Tapson. A model of the humanoid body for self collision detection based on elliptical capsules. In *IEEE international conference on Robotics and Biomimetics*, pages 2397–2402, 2011.

[GB08] J. Gain and D. Bechmann. A survey of spatial deformation from a user-centered per-

spective. *ACM Trans. Graph.*, 27(4):107:1–107:21, November 2008.

[Ger10] R. Geraerts. Planning short paths with clearance using explicit corridors. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1997–2004. IEEE, 2010.

[GKLM11] S.J. Guy, S. Kim, M.C. Lin, and D. Manocha. Simulating heterogeneous crowd behaviors using personality trait theory. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 43–52, 2011.

[GLM96] 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*, pages 171–180. ACM, 1996.

[Got00] S. Gottschalk. *Collision queries using oriented bounding boxes*. PhD thesis, The University of North Carolina, 2000.

[HFV00] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407(6803):487–90, September 2000.

[HTKG04] B. Heidelberger, M. Teschner, R. Keiser, and M. Müller M. Gross. Consistent penetration depth estimation for deformable collision response. In *Vision, Modeling, and Visualization 2004: Proceedings*. IOS Press, 2004.

[Hub96] P.M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[J.T98] J.T. Kolowski. *Efficient Collision Detection for Interactive 3D Graphics and Virtual Environments*. PhD thesis, State Univ. of New York, 1998.

[KGO09] I. Karamouzas, R. Geraerts, and M. Overmars. Indicative routes for path planning and crowd simulation. In *Proceedings of the 4th International Conference on Foundations of Digital Games*, pages 113–120. ACM, 2009.

[KHI+07] S. Kockara, T. Halic, K. Iqbal, C. Bayrak, and R. Rowe. Collision detection: A survey. In *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 4046–4051. IEEE, 2007.

[KHM+98] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-dops. *Visualization and Computer Graphics, IEEE Transactions on*, 4(1):21–36, 1998.

[OD99] C. O'Sullivan and J. Dingliana. Real-time collision detection and response using sphere-trees. *Spring Conference on Computer Graphics*, pages 83 – 92, 1999.

[PAB07] N. Pelechano, J.M. Allbeck, and N.I. Badler. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 99–108. Eurographics Association, 2007.

[Rey06] C. Reynolds. Big fast crowds on PS3. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, pages 113–121, 2006.

[SES+13] S.A. Stüvel, A. Egges, F.A. van der Stappen, N. Magnenat-Thalmann, and D. Thalmann. Collision checking with occupancy spaces. Poster presented at Motion in Games 2013, Dublin, Ireland, 2013.

[VBSE11] B.J.H. Van Basten, S.A. Stüvel, and J. Egges. A hybrid interpolation scheme for footprint-driven walking synthesis. In *Proceedings of Graphics Interface*, pages 9–16, 2011.

[vdB97] G. van den Bergen. Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools*, 2(4):1–13, 1997.

[vdB04] G. van den Bergen. *Collision detection in interactive 3D environments*. Elsevier, 2004.

[VLnG10] G. Vigueras, M. Lozano, J.M. Ordu na, and F. Grimaldo. A comparative study of partitioning methods for crowd simulations. *Applied Soft Computing*, 10(1):225–235, 2010.

[vTCG12] W.G. van Toll, A.F. Cook, and R. Geraerts. Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds*, 23(1):59–69, 2012.