# $k$-IOS: Intersection of Spheres for Efficient Proximity Query

Xinyu Zhang and Young J. Kim

***Abstract*— We present a new bounding volume structure, $k$-IOS that is an intersection of $k$ spheres, for accelerating proximity query including collision detection and Euclidean distance computation between arbitrary polygon-soup models that undergo rigid motion. Our new bounding volume is easy to implement and highly efficient both for its construction and runtime query. In our experiments, we have observed up to 4.0 times performance improvement of proximity query compared to an existing well-known algorithm based on swept sphere volume (SSV) [1]. Moreover, $k$-IOS is strictly convex that can guarantee a continuous gradient of distance function with respect to object's configuration parameter.**

## I. Introduction

Performing proximity query is vital in algorithmic robotics. For instance, maintaining minimum distances from obstacles can assist robots to perceive a high risk region of potential collisions and thereby generate proper reactions before imminent collision occurs. A fast proximity query algorithm is a key to providing a sufficient time for robots to respond in time. Even though various algorithms for efficient proximity queries exist, however, they are still the bottleneck of many robotic algorithms, and faster proximity queries are desired. Moreover, advanced dynamic proximity query algorithms such as continuous collision detection [2], [3], [4] or penetration depth computation [5], [6] still requires fast proximity queries such as collision detection and distance calculation.

Bounding volume (BV) or hierarchies of BV (BVH) has been successfully used for accelerating proximity query in literature. Bounding volumes are often used to cull the geometric components that are far apart and that do not contribute the final proximity result. The most commonly used bounding volumes in literature are illustrated in Fig. 1. Designing a good bounding volume depends on sophistication of constructing a bounding volume, the cost of intersection and distance tests between bounding volumes, and the accuracy of these tests. In general, the more complicated a bounding volume is, the more expensive the cost of bounding volume construction and query become. Moreover, there is a trade-off between the cost of BV construction and query, and the accuracy of proximity results.

It is unnecessary for BVH-based proximity approaches to perform the precise intersection tests or distance computation between bounding volumes, as long as the correctness of the final results between underlying, bounded geometries is guaranteed. Thus, it is sufficient to devise conservative, efficient tests between bounding volumes. Since most of existing bounding volumes are designed to perform overlap tests or distance computation precisely, a new bounding volume structure is desired to perform only conservative tests that are simple, fast and preserving the correctness of the final result in BVH-based proximity query. Note that this conservativeness merely produces mores false-positives.

Meanwhile, in optimization-based collision avoidance and trajectory optimization [7], continuous distance gradients are required between bounding volumes that bound the motion of a robot in order to achieve a fast convergence. As explained in [8], a distance gradient continuity can be guaranteed only with strictly convex bounding volumes. A discontinuity in distance gradient may happen when the minimum distance can be realized by more than one witness point pairs. Since the gradient of distance is essentially the direction of closest vector, an abrupt change in the closest vector may also cause a problem in haptics or dynamics. Non-strictly convex bounding volumes such as AABBs or OBBs may have such a discontinuity problem. As shown in Fig. 1, we can classify existing bounding volumes into two categories: (a) non-strictly convex or non-convex and (b) strictly convex. Sphere, ellipsoid, and our new bounding volume $k$-IOS belong to strictly-convex bounding volumes.

**Main Contributions:** In this paper, we propose a new, strictly convex bounding volume $k$-IOS for fast proximity queries including collision detection and Euclidean distance computation. Our algorithm is designed for arbitrary rigid polygon-soup models. Our new bounding volume $k$-IOS has the following characteristics:

- easy to construct and implement;
- tight fit of the underlying polygonal-soup model;
- very efficient for proximity query by using simple, fast, conservative collision and distance tests;
- easy for SIMD parallelization due to its intrinsic parallel-friendliness and
- strictly convex and guaranteeing continuous distance gradients between $k$-IOS.

## II. Related Work

In general, most of existing bounding volumes are represented as an intersection of a set of half spaces; these half spaces can be either a planar or non-planar surface. For example, both AABBs (Axis-Aligned Bounding Box) and OBBs (Oriented Bounding Box) are constructed by the intersection of six (planar) half spaces. Cylinders are the intersection of a cylindrical tube and two (planar) half spaces. We refer readers to [9] for other commonly used bounding volumes.

X. Y. Zhang and Y. J. Kim are with the Department of Computer Science and Engineering at Ewha Womans University in Seoul, Korea. {zhangxy|kimy}@ewha.ac.kr

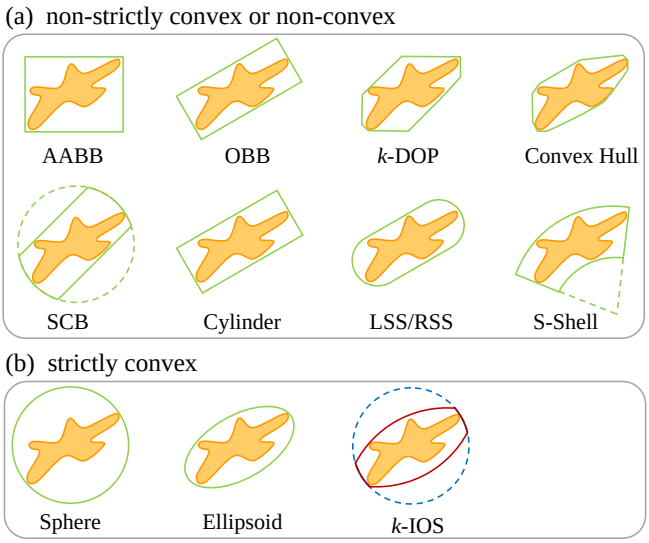


Fig. 1. Various Bounding Volume Shapes. Our new bounding volume shape is an intersection of $k$ spheres ($k$-IOS)

Among these bounding volumes, AABBs [10], OBBs [1] and $k$-DOPs ($k$-Discrete Oriented Polytopes) [11] are known to be efficient for collision detection, but not as fast as SSV [1] for distance computation. And they are not strictly convex either. Spheres [12] are strictly convex, but they do not fit the underlying geometries tightly. Ellipsoids [13] are strictly convex as well, but slower than SSV in proximity query. It is not clear whether Slab Cut Ball (SCB) [14] or Spherical Shell (S-Shell) [15] are suitable for distance computation. Moreover, S-Shell is a non-convex shape. STP-BV (Sphere and Torus Patches) [8] is strictly convex, but it suffers from the robustness problem during construction. Moreover, it does not guarantee strict convexity for a model consisting of very thin triangles. Due to its sophisticated constructing procedure, it is not clear how to extend it to BVH construction for non-convex objects.

SSV is the most well-known bounding volume for distance computation [16] for complicated polygonal models, which was implemented in PQP library [1]. As a result, PQP and its variants have been used in many motion planners such as MPK [17], OOPSMP [18], OMPL [19] and ROS [20]. SSV utilizes three types of bounding volume shapes: point swept sphere (PSS), line swept sphere (LSS) and rectangle swept sphere (RSS), to provide varying tightness for the underlying objects. In particular, a fast implementation of distance computation requires sophisticated procedures such as early rejection using Voronoi regions [21], and separating axis tests [22]. Such a highly specialized algorithm was introduced in PQP [1] and it is reported four times faster than GJK algorithm [23]. However, LSS and RSS are not strictly convex.

Other algorithms such as Lin-Canny [21] and GJK are applicable to general convex polyhedra. Using convex decomposition [24], these algorithms can be extended to non-convex objects. However, they cannot be used for arbitrary polygon-soup models.

## III. Construction

### A. Definition

Our bounding volume is defined as an intersection of $k$ spheres with different radii and denoted by $k$-IOS. In other words, it is

$$k\text{-IOS} = \cap S_i (i = 0, 1, ..., k-1)$$

where $S_i$ is the $i$th sphere that encloses the underlying geometries and $k$ is the total number of spheres. Simply speaking, $S_0$ plays a role of a central sphere bounding the entire geometry and other spheres ($0 < i < k$) correspond to cutting off the void space in $S_0$. And $k$-IOS utilizes spherical surfaces to define half spaces. Like AABB, we can align the major axes of $k$-IOS with the principal axes (axis-aligned $k$-IOS) or like OBB, align it with the local principal axes of the geometries. It corresponds to axis-aligned $k$-IOS (oriented $k$-IOS). Or like $k$-DOPs, the axes of $k$-IOS can be aligned with fixed directions (fixed direction $k$-IOS).

Fig. 2 demonstrates an example of axis-aligned 3-IOS. That is 3-IOS $= S_0 \cap S_1 \cap S_2$. Here, $S_0$ (blue solid circle) is a regular bounding sphere that encloses the underlying geometries tightly. $S_1$ and $S_2$ (blue dotted circles) are two bigger bounding spheres to cut off the void space of $S_0$. The final bounding volume is the intersection of the three spheres highlighted in red.

In this paper, we focus on oriented $k$-IOS because it is more efficient than others for distance computation. Other types of $k$-IOS can be constructed in a similar manner.

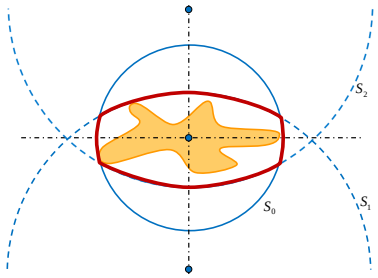


Fig. 2. Axis aligned 3-IOS (intersection of 3-spheres) consists of three spheres: $S_0$ (blue solid circle), $S_1$ and $S_2$ (blue dotted circles) for underlying geometries (orange). The shape in red is a 3-IOS.

### B. Constructing k-IOS

For an oriented $k$-IOS, the number of spheres $k$ could be 1, 3 or 5, determined by the dimensions of the underlying geometries.

Here, we present two methods of constructing $k$-IOS for a given set of triangles. The first method utilizes $k$ spheres to approximate a (smallest) bounding ellipsoid that encloses the underlying geometries and the second method applies principal component analysis (PCA) [22] for constructing $k$-IOS.

*1) Method I:* We first compute the smallest ellipsoid [25] enclosing all the vertices (green ellipsoid in Fig. 3). This enclosing ellipsoid can be arbitrarily oriented. We denote the center of the ellipsoid as $\vec{o}_0$ and the principal directions of the ellipsoid as $\vec{e}_0$, $\vec{e}_1$ and $\vec{e}_2$, respectively. In the local coordinate system of the ellipsoid, we denote the equatorial radii as $a$ and $b$ along the $\vec{e}_0$ and $\vec{e}_1$ axes, respectively, and the polar radius as $c$ along $\vec{e}_2$ axis. Without loss of generality,

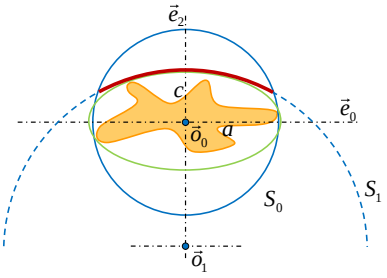


Fig. 3. Method I: Constructing a $k$-IOS using ellipsoid.

we assume $a \geq b \geq c$. Then, at the center of the ellipsoid, we use a sphere $S_0$ to enclose all the vertices of the given triangles. If we denote the radius of $S_0$ as $r_0$, obviously,

$$r_0 \leq a \tag{1}$$

Then, we use other two spheres ($S_1$ and $S_2$) with the same radius $r_1 = r_2$ to approximate the ellipsoid along $\vec{e}_2$ where $S_1$ approximates the upper half-ellipsoid and $S_2$ approximates the lower half-ellipsoid. So the ellipsoid is enclosed by the intersection of the two spheres. The radius of the sphere can be obtained by calculating the maximum of its radius of curvature on the $\vec{e}_0\vec{e}_1$-plane.

$$r_1 = r_2 = \frac{a^2}{c}$$

where $r_1 = r_2 > a$. Their centers are located at

$$\vec{o}_{1,2} = \vec{o}_0 \pm (r_1 - c) \cdot \vec{e}_2$$

It guarantees that each sphere encloses the ellipsoid entirely. Similarly, we can use other two spheres ($S_3$ and $S_4$) with radii $r_3$ and $r_4$ to approximate the half-ellipsoids along $\vec{e}_1$. For simplicity, we can let $r_3$ and $r_4$ be equal to $r_1$ and $r_2$. Their centers are located at

$$\vec{o}_{3,4} = \vec{o}_0 \pm (r_3 - b) \cdot \vec{e}_1$$

However, determining the smallest enclosing ellipsoid [26], [25] is non-trivial and could be computationally expensive since it has a time complexity $O(n^2)$ where $n$ is the number of points. Alternatively, we can approximate a bounding ellipsoid by using PCA and then resize the resulting ellipsoid to tightly enclose the underlying geometries.

*2) Method II:* The second method is more simple than the first one. We first compute a regular bounding sphere ($r_0$ and $\vec{o}_0$) that encloses the underlying geometries. Then, we perform PCA to find principal axes of the underlying geometries. As shown in Fig. 4, the PCA result is displayed

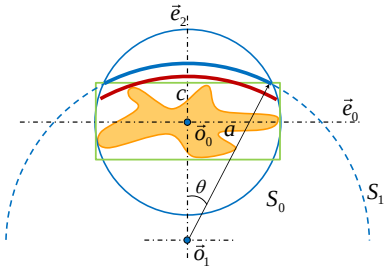


Fig. 4. Method II: Constructing a $k$-IOS using PCA.

as a green box. The radius of $S_1$ and $S_2$ is calculated as follows.

$$r_1 = r_2 = \frac{\sqrt{{r_0}^2 - c^2}}{\sin\frac{\pi}{6}}$$

where $\frac{\pi}{6}$ is an empirical value obtained by our experiments, which provides a better trade-off between the tightness of $k$-IOS and the conservativeness of proximity query using $k$-IOS. In general, a smaller empirical value produces a bigger spheres that can lead to a tighter bounding volume, but may incur more BV tests during runtime query due to its conservativeness. However, if this value is too big, the resulting sphere becomes rather small. As a result, this sphere does not help much to cut off the void space in $S_0$ and it causes a less tight $k$-IOS.

Then, the centers of $S_1$ and $S_2$ are

$$\vec{o}_{1,2} = \vec{o}_0 \pm (r_1 \cos\frac{\pi}{6} - c) \cdot \vec{e}_2$$

$S_1$ is displayed as a dotted blue circle (arc) in Fig. 4. However, $S_1$ may not fit the underlying geometries tightly from above. This can be solved by moving $S_1$ downwards. Therefore, we compute a bounding sphere with respect to $\vec{o}_1$ and its radius $r'$. Then $S_1$ can move downwards by the difference of $r'$ and $r_1$. The new center of $S_1$ is shifted to

$$\vec{o}_1^* = \vec{o}_1 + (r_1 - r') \cdot \vec{e}_2$$

The new $S_1$ is shown as a red arc in Fig. 4. $S_2$ can be computed in the same way.

Moreover, two other spheres on another axis can be determined by the same means. We compute the centers of $S_3$ and $S_4$ as follows, assuming their radii are $r_3$ and $r_4$.

$$\vec{o}_{3,4} = \vec{o}_0 \pm \left(\sqrt{{r_1}^2 - (a^2 + c^2)} - b\right) \cdot \vec{e}_1$$

Then, we move $S_3$ downwards to the enclosed geometries from above as we did for $S_1$. $S_4$ can be computed in the same way. Based on the fact that the medium splitting is usually used for constructing bounding volume hierarchies, we can let $r_4 = r_3 = r_2 = r_1 = \max(r_i)$. In this way, the memory usage can be further saved as well.

### C. Determine k

The total number of spheres, $k$, varies with respect to the relative dimensions of ellipsoid or PCA result as follows.

1) $k = 5$ if $\frac{a}{b} > 1.5$ and $\frac{a}{c} > 1.5$
2) $k = 3$ if $\frac{a}{b} < 1.5$ and $\frac{a}{c} > 1.5$
3) $k = 1$ if $\frac{a}{b} < 1.5$ and $\frac{a}{c} < 1.5$

where 1.5 is another empirical value obtained by our experiments. This result is based on the fact that the ellipsoid (Method I) becomes close to a spheroid if $a \approx b$. Thus, we have $r_1 \approx a$ ($r_1 > a$). Meanwhile considering the formula (Eq. 1), $r_0 \approx a$ ($r_0 \leq a$) if $a \approx b$. We obtain $r_1 \approx r_0$. It implies that the two spheres ($S_1$ and $S_2$) are, in fact, similar to the sphere $S_0$ and their centers are close to the $\vec{o}_0$. In such a case,$S_1$ and $S_2$ are not much needed to approximate the ellipsoid. Thus, we may merely keep $S_0$ instead of using all the three spheres.

### D. Some Geometric Properties of k-IOS

Our bounding volume has the following properties:

**Property 1** *A k-IOS is strictly convex bounding volume.*

**Property 2** *There is a unique pair of closest points (witness) between two separating and touching k-IOS.*

**Property 3** *The closest points (witness) realizing the minimum distance between two separating and touching k-IOS vary smoothly with respect to their relative configuration.*

**Property 4** *The minimum distance between two separating and touching k-IOS is a $C^1$ function with respect to their relative configuration. In other words, the minimum distance has a continuous gradient.*

**Property 5** *A k-IOS is piecewise $C^\infty$.*

These properties are very straightforward to show and the proofs directly follow the theorems in [8]. Non-unique closest point pairs (witness) can often occur in other non-strictly convex bounding volumes such OBB, $k$-DOP, Convex Hull, SSV, etc., for example, when an edge or a face is parallel to another edge or face. This causes the minimum distance to be realized by more than one point pairs. This does not affect the global minimum distance value, but it does affect the location of closest points. As a result, a gradient discontinuity occurs due to a sudden jump between witnesses.

## IV. Runtime Proximity Queries

BVH-based proximity query typically does not require exact overlap tests or distance computation between bounding volumes. For instance, a lower bound value of distance between two bounding volumes is sufficient to determine that these bound volume cannot contribute to the actual distance by comparing the lower bound against an upper bound of the distance of the entire geometry [1]. For collision detection between complicated geometry, knowing that a pair of BV pairs do not overlap provides a sufficient condition to cull these pairs for further consideration [1].

Conservative tests are sufficient and are able to guarantee the correctness of the results. Thus, we present two conservative tests, one for overlap and the other for distance computation.

### A. Overlap Tests

Given two BVs $A$ and $B$, the conservative overlap tests involve $k^A \times k^B$ sphere-sphere overlap tests, where $k^A$ spheres from $A$ and $k^B$ spheres from $B$. Whenever a sphere-sphere pair is reported as DISJOINT, we immediately say that there is no overlap between $A$ and $B$. Note that the overlap test is conservative and the result may generate false-positive answers.

**Algorithm 1** `ConservativeOverlap`(IOS A, IOS B)

```
1: for i = 0 to k^A − 1 do
2:   for j = 0 to k^B − 1 do
3:     if disjointSphereSphere(S_i^A, S_j^B) then
4:       return false
5:     end if
6:   end for
7: end for
8: return true
```

### B. Distance Query

Since it is expensive to compute the exact distance between $k$-IOS and it is unnecessary to use it with BVH, here we present a method to calculate a lower bound to the actual distance. As shown in Eq. 2, our lower bound is the maximum among the distances between all sphere pairs. Note that using a lower bound of distance between $k$-IOS, instead of the actual distance, does not affect the correctness of distance computation when a set of $k$-IOS are used to build the BVH, but may increase the number of bounding volume tests during BVH traversal.

$$LB(d) = \max\left[dist(S_i^A, S_j^B)\right] (0 \leq i < k^A, 0 \leq j < k^B) \quad (2)$$

where $S_i^A$ and $S_j^B$ are the spheres enclosing $A$ and $B$, respectively.

**Proof**: Let a point $p \in A$ and a point $q \in B$. The definition of our BV states that $A \subset S_i^A$ and $B \subset S_j^B$. Obviously, $p \in S_i^A$ and $q \in S_j^B$. The distance between any two points $p$ and $q$ is greater than or equal to the closest distance between a pair of spheres (based on the definition of the distance between two spheres).

$$|p - q| \geq dist\left(S_i^A, S_j^B\right) (0 \leq i < k^A, 0 \leq j < k^B) \quad (3)$$

We assume that the closest distance between two spheres is negative if they overlap. The closest distance between two BVs is the minimum among the distances $|p-q|$ and it is still greater than or equal to the closest distance between the pair of spheres.

$$d = \min |p - q| \geq dist\left(S_i^A, S_j^B\right) \quad (4)$$

The maximum among the closest distances of all the pairs is a lower bound of the distance between the two BVs. □

**Algorithm 2** `distLowerBound`(IOS A, IOS B)

```
d = 0
for i = 0 to k^A − 1 do
    for j = 0 to k^B − 1 do
        d = max(d, distSphereSphere(S_i^A, S_j^B))
    end for
end for
return d
```

**Streaming SIMD Implementation**: Many modern microprocessors have SIMD (single instruction, multiple data) instruction sets. These SIMD instructions execute a single operation on multiple values (4, 8 or 16 scalars) in parallel. Instruction-level parallel overlap tests for spheres has been suggested in [9] using streaming SIMD. Four sphere-sphere tests can then be performed in parallel. In our implementation, we extend this idea to $k$-IOS for proximity query. For the given two $k$-IOS consisting of $k^A$ and $k^B$ spheres, respectively, the $k^A \times k^B$ sphere-sphere tests can be packed into $\lfloor k^A \times k^B)/4 \rfloor$ [1] SIMD sphere-sphere tests. If the rest sphere-sphere pairs is greater than 1 (i.e. $[(k^A \times k^B)$ MOD $4]>1$), they are packed into another SIMD sphere-sphere test; otherwise a non-SIMD sphere-sphere test is performed.

## V. Implementation & Experimental Results

We have implemented $k$-IOS-based proximity algorithm on Windows using C++ language and Visual Studio 2008 with Streaming SIMD Extensions (SSE) enabled. Proximity queries were run on a PC with 2.67Hz Intel Core i7 CPU and 3.0G memory.

### A. Experimental Results

We first tested our new bounding volume with the rigid benchmarks consisting of polygonal soup models, also used in the PQP library [1]. We computed a BVH based on $k$-IOS for each model or each link of an articulated model. For these benchmarks, we repeatedly performed collision detection and Euclidean distance calculation for each benchmarking scenario.

1) Falling torus (Fig. 5-left). A bent torus(1332 triangles) falls through the center of a knobby torus (3240 triangles). The average timing for collision detection and distance computation is 0.135ms and 0.494ms, respectively. The memory usages of the two models are 9.8 and 9.9 floats per bounding volume, respectively.
2) Spinning bunny and torus (Fig. 5-right). A bunny and a torus are set apart in space and spin independently. The bunny and the torus consist of 2240 and 1332 triangles. The average timing for collision detection and distance computation is 0.419$\mu$s and 0.491ms, respectively. The average memory usages of two models are 9.8 and 9.6 floats per BV, respectively

We also tested our new bounding volume with the articulated robot benchmarks that consists of polygonal soup links. The robot motions were generated using Webots, a robot simulation software developed by Cyberbotics [27]. Then, the motions were used in our algorithm for performing Euclidean distance computation. The minimum distances between the links of articulated robots can assist the robots perceiving a high risk region of collisions so as to avoid it.

Fig. 5. (left) falling torus. (right) spinning torus and bunny.

1) IPR robot collaboration (Fig. 6-left). Two IPR robots collaboratively work together in the same work space. Each robot has 7 articulated links. $k$-IOS takes 9.6ms on average to compute the minimum distances for 33 links and PQP takes 20ms on average. The maximum performance improvement of $k$-IOS against PQP is 4. Note that some stationary links and the ones far apart are excluded from the query.
2) Aldebaran Nao robots in RobotCup (Fig. 6-right). Two Aldebaran Nao robots interact with each other on a Robotcup field. Each robot has 19 links. The minimum distances are tracked for each link of the robot in order to assist the robots perceiving a high risk region of collisions. $k$-IOS takes 98ms on average to compute the minimum distances for 33 links and PQP takes 215ms on average. The maximum performance improvement of $k$-IOS against PQP is 2.8.

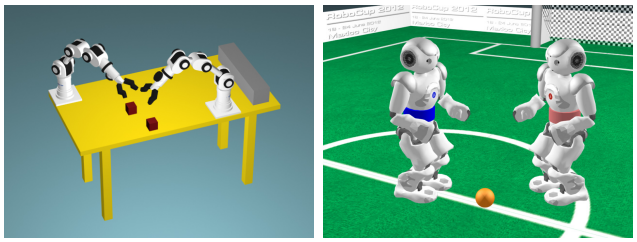

Fig. 6. (left) Two IPR robots collaboratively work together in the same work space. (right) Two Aldebaran Nao robots interact with each other on the RoboCup field.

### B. Comparisons and Discussions

We made a performance comparison between $k$-IOS and SSV[2] for proximity queries. To the best of our knowledge, SSV is the only public proximity library applicable to polygon-soup models for collision and distance queries. Table I shows the performance comparisons for collision detection and distance computation using the PQP benchmarks. For the set of benchmarks in Fig. 5, we observe up to 2.2 times performance improvement over SSV [1]. Though $k$-IOS requires more bounding volume tests than

[1]

[2] `http://gamma.cs.unc.edu/SSV`

PQP mainly due to its conservativeness for overlap tests and distance computation, its SIMD sphere-sphere tests are much simpler and faster than those used in SSV. Moreover, $k$-IOS is easier to implement, and more amenable to parallelization than SSV, and scalable with respect to the number of spheres (i.e. $k$).

TABLE I
THE TIMING STATISTICS AND THE NUMBER OF BV TESTS.

| Benchmarks | BV Types | Timings (ms) | # BV Tests |
|---|---|---|---|
| Falling | $k$-IOS | 0.135 | 798 |
| | PQP | 0.269 | 568 |
| Spinning | $k$-IOS | 0.494 | 1896 |
| | PQP | 0.495 | 1194 |

Moreover, for articulated robots shown in Fig. 6, $k$-IOS can achieve up to 4.0 times performance improvement compared to SSV.

With the increase of $k$, the number of spheres, $k$-IOS can bound the underlying geometries more tightly. However, since every sphere's location is explicitly represented, the memory usage of arbitrarily located $k$-IOS is relative expensive. Therefore, fixed direction $k$-IOS, including axis-aligned $k$-IOS, can be a potentially alternative in terms of memory usage. In contrast to $k$-DOPs, fixed direction $k$-IOS is not necessary to be recomputed even if the underly geometries are rotated. Moreover, the cost of overlap tests and distance computation between $k$-IOS increases quadratically with respect to the number of spheres. But with high performance SIMD architecture such as GPU, sphere-sphere tests can be performed in parallel with easy implementation.

## VI. CONCLUSION

We present a new bounding volume: $k$-IOS for efficient proximity query. It is easy to construct, implement and parallelizable on modern CPUs.

There are a few interesting future directions that deserve to be pursued. The current distance query between $k$-IOS is based on a conservative test, which is a lower bound of the minimum distance. This works fine with BVH structure. To achieve gradient continuity between $k$-IOS, however, the exact distance between them must be calculated. This can be solved by considering the intersection of two spherical shells (the result is a circle) during the distance query. We leave the gradient continuity and its application to optimization-based collision avoidance as a future work.

Another future work is to investigate the application of $k$-IOS in other proximity-related fields such as ray tracing in computer graphics. $k$-IOS provides an advantage for fast intersection tests against rays since ray tracing boils down to simple intersection tests between $k$ spheres and rays. Extending $k$-IOS to handle deformable models could be another interesting future work.

## ACKNOWLEDGEMENT

This research was supported in part by IT R&D program of MKE/MCST/KOCCA (2008-F-033-02) and NRF in Korea (NRF-2011-013-D00112).

## REFERENCES

[1] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast proximity queries with swept sphere volumes," in *International Conference on Robotics and Automation*, 2000, pp. 3719–3726.
[2] X. Y. Zhang, M. Lee, and Y. J. Kim, "Interactive continuous collision detection for non-convex polyhedra," *Pacific Graphics*, 2006.
[3] X. Y. Zhang, S. Redon, M. Lee, and Y. J. Kim, "Continuous collision detection for articulated models using Taylor models and temporal culling," *SIGGRAPH*, vol. 26, no. 3, p. 15, 2007.
[4] M. Tang, Y. J. Kim, and D. Manocha, "C2a: Controlled conservative advancement for continuous collision detection of polygonal models," *ICRA*, 2009.
[5] L. Zhang, Y. J. Kim, G. Varadhan, and D. Manocha, "Generalized penetration depth computation," *Comput. Aided Des.*, vol. 39, pp. 625–638, 2007.
[6] C. Je, M. Tang, Y. Lee, M. Lee, and Y. J. Kim, "Polydepth: Real-time penetration depth computation using iterative contact-space projection," *ACM Transactions on Graphics*, 2011.
[7] A. Escande, A. Kheddar, and S. Miossec, "Planning support contact-points for humanoid robots and experiments on hrp-2," in *IROS*, 2006, pp. 2974–2979.
[8] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distance for humanoids free-collision optimized-postures," in *Proc. 7th IEEE-RAS International Conference on Humanoid Robots*, 2007, pp. 188–195.
[9] C. Ericson, *Real-Time Collision Detection*. Morgan Kaufmann, 2004.
[10] G. van den Bergen, "Efficient collision detection of complex deformable models using AABB trees," *Graphics Tools*, vol. 2, no. 4, pp. 1–13, 1997.
[11] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of $k$-DOPs," *IEEE Trans. Visualization and Computer Graphics*, vol. 4, no. 1, pp. 21–36, 1998.
[12] I. J. Palmer and R. L. Grimsdale, "Collision detection for animation using sphere-trees," *Computer Graphics Forum*, vol. 14, no. 2, pp. 105–116, 1995.
[13] S. Liu, C. C. L. Wang, K.-C. Hui, X. Jin, and H. Zhao, "Ellipsoid-tree construction for solid objects," in *ACM symposium on Solid and physical modeling*, 2007, pp. 303–308.
[14] T. Larsson and T. Akenine-Moller, "Bounding volume hierarchies of slab cut balls," *Comput. Graph. Forum*, vol. 28, no. 8, pp. 2379–2395, 2009.
[15] S. Krishnan, A. Pattekar, M. C. Lin, and D. Manocha, "Spherical shell: a higher order bounding volume for fast proximity queries," in *WAFR*, 1998, pp. 177–190.
[16] C. Lauterbach, Q. Mo, and D. Manocha, "gproximity: Hierarchical gpu-based operations for collision and distance queries," in *Eurographics*, 2010.
[17] MPK-Team, *Motion Planning Kit*, http://ai.stanford.edu/∼mitul/mpk/.
[18] E. Plaku, K. Bekris, and L. E. Kavraki, "Oops for motion planning: An online opensource programming system," in *International Conference on Robotics and Automation (ICRA)*, 2007, pp. 3711–3716.
[19] OMPL-Team, *The Open Motion Planning Library (OMPL)*, http://ompl.kavrakilab.org, 2010.
[20] *OMPL ROS Interface Package*, http://www.ros.org/wiki/ompl_ros_interface.
[21] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," in *ICRA*, 1991, pp. 1008–1014.
[22] S. Gottschalk, M. Lin, and D. Manocha, "OBB-Tree: A hierarchical structure for rapid interference detection," in *Proc. SIGGRAPH*, 1996, pp. 171–180.
[23] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal of In Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.
[24] S. Ehmann and M. Lin, "Accurate and fast proximity queries between polyhedra using surface decomposition," *Computer Graphics Forum*, vol. 20, no. 3, pp. 500–510, 2001.
[25] L. G. Khachiyan, "Rounding of polytopes in the real number model of computation," *Mathematics of Operations Research*, vol. 21, no. 2, pp. 307–320, 1996.
[26] E. Welzl, "Smallest enclosing disks (balls and ellipsoids)," in *Results and New Trends in Computer Science, LNCS*, 1991, pp. 359–370.
[27] Webots, "http://www.cyberbotics.com," commercial Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com