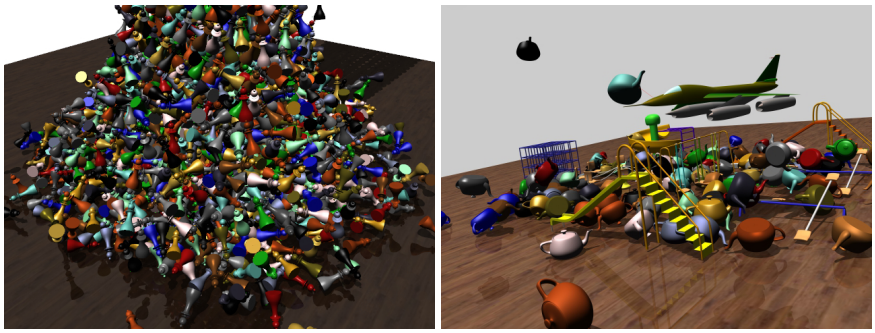
Coons BVH for Freeform Geometric Models

Yong-Joon Kim[1] Young-Taek Oh[1] Seung-Hyun Yoon[2] Myung-Soo Kim[1*] Gershon Elber[3]

[1]Seoul National University [2]Dongguk University [3]Technion



(a) (b)

Figure 1: *(a) 10000 freeform geometric models (chess pieces) falling into a pile, where the same models share a common BVH of Coons patches approximating the given freeform NURBS surfaces within an error bound 10^{-5}, where the unit length is taken as the largest side length of the minimum bounding box of the model, and (b) the minimum distance computation between a flying B58 model and a complex dynamic scene with many Utah teapots falling to the playground.*

Abstract

We present a compact representation for the bounding volume hierarchy (BVH) of freeform NURBS surfaces using Coons patches. Following the Coons construction, each subpatch can be bounded very efficiently using the bilinear surface determined by the four corners. The BVH of freeform surfaces is represented as a hierarchy of Coons patch approximation until the difference is reduced to within a given error bound. Each leaf node contains a single Coons patch, where a detailed BVH for the patch can be represented very compactly using two lists (containing curve approximation errors) of length proportional only to the height of the BVH. We demonstrate the effectiveness of our compact BVH representation using several experimental results from real-time applications in collision detection and minimum distance computation for freeform models.

Keywords: Coons patch, freeform surface, bilinear surface, NURBS, bounding volume hierarchy (BVH), tetrahedron, offset, collision detection, minimum distance computation

Links: DL PDF

1 Introduction

Hierarchical spatial data structures play an essential role in the design of efficient geometric algorithms for three-dimensional objects [Samet 2006]. Real-time algorithms for polygonal meshes employ various different types of BVHs that are built in a preprocessing stage of the geometric computation [Akenine-Möller et al. 2008]. The BVH for a polygonal mesh usually requires a much larger memory space compared to the original model itself [Yoon and Manocha 2006]. Thus it is an important subject of research to develop compact representations for BVH structures.

Freeform geometric models are more compact than polygonal meshes. The BVH structure of freeform geometry can be generated by recursively subdividing the freeform surfaces [Johnson and Cohen 1998]. Nevertheless, it is unclear, in general, where to stop the recursive subdivision and how to proceed with the geometric computation when we reach the leaf level. In this paper, we address these two important issues and propose a compact BVH construction scheme for freeform geometry that is based on the special structure of the Coons patch.

The Coons patch is one of the earliest freeform representation schemes in CAGD and was developed in the early 1960's [Coons 1964]. (For an introduction to Coons patches, see Chapter 14 of [Cohen et al. 2001] and Chapter 22 of [Farin 2002].) Compared with other freeform surfaces, such as B-spline or Bézier surfaces, Coons patches are seldom used in contemporary freeform modeling applications. Nevertheless, there are many useful properties of Coons patches that we employ in this work for the acceleration of geometric algorithms for freeform shapes. The most important property, for our purpose, is that Coons patches are uniquely determined by their boundary curves. As a direct consequence, Coons patches can be subdivided very efficiently by evaluating points only on their boundary curves.

The hierarchy of recursive Coons approximations generates a BVH, where the leaf nodes contain the Coons patches approximating the original freeform surface to within a given error bound. The interior nodes of the BVH correspond to freeform surface patches that are recursively subdivided. While it may seem there is nothing dramatically different from conventional BVH approaches, the significant difference is in the size of the surface patches that are stored in the leaf nodes, and hence the size of the entire BVH. Because the Coons patches approximate the freeform surfaces very

*Corresponding author; e-mail: mskim@snu.ac.kr

tightly, the recursive subdivision process terminates early and the Coons patches are in general much larger than typical triangles in the polygonal mesh approximation for similar tolerances.

Switching from the freeform surface to the Coons patch at the leaf level, we can proceed to the deeper levels of the hierarchy more efficiently than the upper levels. From a geometric processing point of view, Coons patches are fully prescribed by their boundary space curves. The special structure of the Coons patch essentially reduces the BVH construction to those for the boundary curves.

The BVH for a pair of opposite boundary curves is represented very compactly by storing only the maximum error ϵ of the two polylines that approximate the opposite curves simultaneously at each level of the curve subdivision hierarchy. Thus we keep only a short list (containing curve approximation errors) of length proportional to $\log n$, where n is the total number of curve segments at the final level of subdivision. Using the four boundary curves thus approximated, a Coons patch subdivided into mn subpatches can be approximated by mn bilinear surfaces within a maximum error $\epsilon_u + \epsilon_v$, the summation of the respective curve approximation errors along the u and v-directions. (See Appendix A for the proof.)

The main contribution of this work can be summarized as follows:

- We propose a compact BVH representation scheme for freeform NURBS surfaces, which requires considerably (often more than 1000 times) less memory space than conventional BVHs.
- We demonstrate the effectiveness of the proposed BVH scheme by developing real-time algorithms for collision-detection and minimum distance computation, for freeform geometric models of non-trivial complexity.

2 Related Work

Many efficient geometric algorithms for polygonal meshes are based on a hierarchy of bounding volumes constructed in a pre-processing step [Akenine-Möller et al. 2008]. The specific type of bounding volume(s) in use has a direct consequence on the performance of algorithms; thus, this need has promoted the development of many different types of bounding volumes. The most representative ones include spheres, axis-aligned bounding boxes (AABBs), oriented bounding boxes (OBBs), and discrete oriented polytopes.

For the distance computation among freeform geometric models, [Johnson and Cohen 1998] generated the BVH of freeform surfaces on the fly by recursively subdividing the B-spline or Bézier surfaces and bounding each surface patch using the convex hull of its control points. The bivariate surface subdivision is time-consuming, which has been the main bottleneck against the realization of real-time algorithms for freeform models. In this paper, we show that the special structure of the Coons patch provides an excellent solution for the resolution of this problem.

[Krishnan et al. 1998a] made an early attempt to employ a pre-built BVH for freeform geometric models using spherical shells. There is, however, no clear termination condition for the BVH construction since the leaf nodes still contain freeform surfaces indefinitely. At the leaf level one has to switch to the recursive subdivision of freeform surfaces as in [Johnson and Cohen 1998]. A tightly fitting bounding volume may alleviate this problem by providing a good initial solution to the subsequent numerical procedures. Nevertheless, there is a fundamental limitation of the conventional BVHs in this respect, due to the slow convergence rate in the approximation of arbitrary freeform surfaces using regular shapes (such as spheres, boxes, rectangles, etc.)

When dealing with freeform models, the "tightness" of fitting should be measured very carefully, i.e., in terms of the Hausdorff distance. The Coons BVH exhibits an important geometric feature that each point of the bounding volume has at least one point from the original surface in a small neighborhood. In this paper, we demonstrate how this property greatly simplifies the design of geometric algorithms and the associated data structures.

For distance-related applications, the offset volumes such as line swept sphere (LSS) and rectangle swept sphere (RSS) have many useful properties [Larsen et al. 1999]. The LSS/RSS volumes are only slightly looser than OBB in bounding objects of arbitrary shape. On the other hand, distance computation with LSS/RSS becomes considerably easier as it is reduced to a simpler problem for lines/rectangles. In this paper, we employ the offset of a tetrahedron as the bounding volume and reduce the distance computation to that of tetrahedra. Employing an efficient and robust implementation [van den Bergen 1999] of the GJK-algorithm [Gilbert et al. 1988], the distance between two tetrahedra can be computed in a speed comparable to that of two rectangles.

Collision detection algorithms and systems have a long history of development [Lin and Gottschalk 1998; Lin and Manocha 2004; Teschner et al. 2005], which has culminated at the first real-time implementation for highly complex polygonal scenes built in the OBB tree [Gottschalk et al. 1996]. Recent work on collision detection deals with more complicated cases such as deformable models [James and Pai 2004; Teschner et al. 2005; Zhang and Kim 2007], continuous collision detection (CCD) [Redon et al. 2007; Tang et al. 2008], and GPU-based algorithms including those executed on multiple CPU/GPUs [Govindaraju et al. 2003; Kim et al. 2009]. A full coverage of these topics is beyond the scope of this paper which is focused on a specific BVH data structure for freeform NURBS surfaces. Nevertheless, the freeform geometric models enhanced with our BVH structure have many useful properties that can better deal with these general topics.

3 Coons BVH

Given a freeform surface $S(u,v)$, $0 \le u,v \le 1$, a bilinearly blended Coons patch $X(u,v)$ interpolates the four boundary curves of $S(u,v)$:

$$\begin{aligned} & X(u,v) \\ = \; & (1-u)S(0,v) + uS(1,v) + (1-v)S(u,0) + vS(u,1) \\ & -[(1-u)(1-v)S(0,0) + (1-u)vS(0,1) \\ & \quad +u(1-v)S(1,0) + uvS(1,1)]. \end{aligned}$$

Note that the Coons patch $X(u,v)$ is generated by adding together two ruled surfaces (each interpolating two opposite boundary curves) only to subtract a bilinear surface that interpolates the four corners. Standard textbooks on CAGD [Cohen et al. 2001; Farin 2002] discuss more details of the Coons patch construction.

BVH for Freeform Surfaces: Given a freeform surface $S(u,v)$, we recursively subdivide the surface until the maximum error: $\max_{(u,v)} \|S(u,v) - X(u,v)\|$ becomes less than a given error bound. The maximum error can be bounded using

$$\begin{aligned} & \|S(u,v) - X(u,v)\| \\ = \; & \left\| \sum\sum (\mathbf{s}_{ab} - \mathbf{x}_{ab}) B_a(u) B_b(v) \right\| \\ \le \; & \sum\sum \|\mathbf{s}_{ab} - \mathbf{x}_{ab}\| B_a(u) B_b(v) \\ \le \; & \max \|\mathbf{s}_{ab} - \mathbf{x}_{ab}\|, \end{aligned}$$

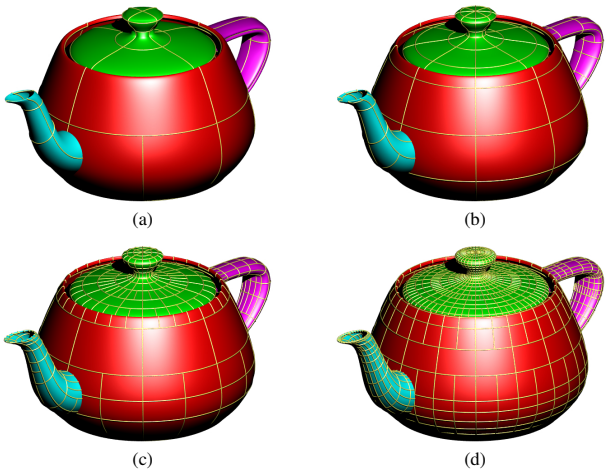


Figure 2: *Approximation of the Utah teapot using Coons patches: (a) 44 Coons patches within an error bound 10^{-2}, (b) 150 patches for 10^{-3}, (c) 492 patches for 10^{-4}, and (d) 1688 patches for 10^{-5}, where the unit length is taken as the largest side length of the minimum bounding box of the teapot.*

where $\mathbf{s}_{ab}$ and $\mathbf{x}_{ab}$ are the control points for $S(u,v)$ and $X(u,v)$, and $B_a(u)$ and $B_b(v)$ are the basis functions. (We can improve the upper bound estimation by further subdividing the difference surface at the parametric node corresponding to the control point of $\max \|\mathbf{s}_{ab} - \mathbf{x}_{ab}\|$.)

The hierarchy of recursive subdivisions is recorded as a dual BVH, where each node contains the radius of a bounding sphere of the corresponding surface patch $S_{ij}(u,v)$, for $u_i \le u \le u_{i+1}$ and $v_j \le v \le v_{j+1}$, and the maximum deviation of $S_{ij}(u,v)$ from the tetrahedron determined by the four corners. Thus each node contains only two scalar values which are computed in a preprocessing step.

The sphere center is taken as the surface mid-point, $S(\frac{u_i+u_{i+1}}{2}, \frac{v_j+v_{j+1}}{2})$. (The surface mid-points are also useful for the different applications to be discussed in the coming sections.) The surface center and corner points can be evaluated on the fly. For the sake of compactness, we store none of these points in the BVH representation. The leaf node of the BVH corresponds to a Coons patch that interpolates the surface $S(u,v)$ along the corresponding four boundary iso-curve segments. The points on the four iso-curves are also generated on the fly.

Figures 2(a)–(d) show the result of approximating the Utah teapot using Coons patches within an error bound 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5}, respectively. For the cases of low precision approximations such as those shown in Figures 2(a)-(b), the number of Coons approximation patches is even smaller than the number of Bézier surfaces that comprise the Utah teapot. In each example, we start with 8 NURBS surfaces, where each periodic surface in the original model is subdivided into two NURBS surfaces. These NURBS surfaces are then approximated with 44, 150, 492, and 1688 Coons patches, respectively, in the four examples of Figures 2(a)–(d).

Error Bound for Coons Patches: Assume that the opposite boundary curves $S(u,0)$ and $S(u,1)$ are approximated within an error bound ϵ_u by polylines sampled simultaneously at u_i, $i = 0, \cdots, m$. Similarly, the curves $S(0,v)$ and $S(1,v)$ are approximated within an error bound ϵ_v by polylines sampled at v_j, $j = 0, \cdots, n$. We can show that each iso-curve in $X(u,\hat{v})$, for a fixed $\hat{v}$, can be approximated within the error bound ϵ_u by a polyline determined by $(m+1)$ points $X(u_i,\hat{v})$. Moreover, the Coons patch $X(u,v)$ can be approximated within an error bound $\epsilon_u + \epsilon_v$ by mn bilinear surfaces $L_{ij}(u,v)$, each determined by four corners $X(u_i,v_j)$, $X(u_{i+1},v_j)$, $X(u_i,v_{j+1})$, and $X(u_{i+1},v_{j+1})$. (See Figure 3.) This claim is a direct consequence of Theorem 14.6 (The Remainder Theorem) of [Cohen et al. 2001]. The proof is based on the concept of linear operator. (See also Appendix A.)

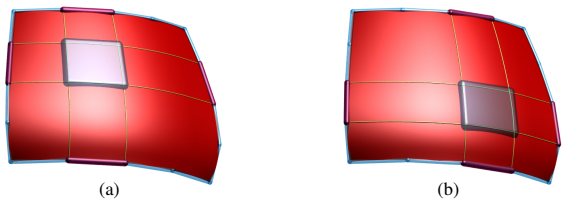


Figure 3: *Coons patch and pairs of opposite boundary curves.*

BVH for Coons Patches: The BVH for a Coons patch $X(u,v)$ can be generated using a hierarchy of offset volumes:

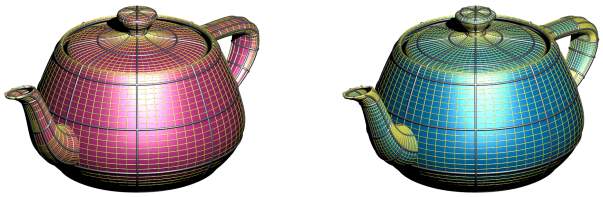
$$O_{\epsilon_u+\epsilon_v}(V_{ij}) = \{\mathbf{p} \mid d(\mathbf{p}, V_{ij}) \le \epsilon_u + \epsilon_v\},$$

which contains all points within a distance $\epsilon_u + \epsilon_v$ from the tetrahedron V_{ij} determined by the four corners of the Coons subpatch $X_{ij}(u,v)$ (defined on a parameter domain $[u_i, u_{i+1}] \times [v_j, v_{j+1}]$) that corresponds to each node of the BVH. This offset volume can be generated by sweeping a ball of radius $\epsilon_u + \epsilon_v$ over the tetrahedron V_{ij}. (The bilinear surface $L_{ij}(u,v)$ with its four corners is contained in V_{ij}, and thus the offset volume $O_{\epsilon_u+\epsilon_v}(V_{ij})$ completely bounds the Coons subpatch $X_{ij}(u,v)$ that is within distance $\epsilon_u + \epsilon_v$ from the bilinear surface $L_{ij}(u,v)$.) Nevertheless, an explicit representation of the BVH would require a storage size proportional to mn. Based on the special structure of the Coons patch, we can design a considerably more compact BVH representation scheme that is proportional to $\log m + \log n$.

When we apply a uniform binary recursive subdivision to the boundary curve $S(u,0)$, for $0 \le u \le 1$, we will end up with a total of $n = 2^h$ subsegments, where h is the maximal level of recursion. At each level of the recursive subdivision, we take the maximum error ϵ of the iso-curve $S(u,0)$ from the polyline sampled at uniform parameters $u_i = i/n$, for $0 \le i \le n$. Though this global maximum error slightly over-estimates the local error for each subsegment, this gives a reasonably good error bound. Moreover, to reduce the storage size by half, we store the maximum error for both opposite boundary curves $S(u,0)$ and $S(u,1)$. We store these error values, ϵ, in a list of length $\log n$ for each pair of opposite boundary curves of the Coons patch. The ϵ-list for the v-direction can be generated in a similar way. Therefore, the BVH structure for a Coons patch can be represented by storing a total of $\log m + \log n$ values of the curve approximation error ϵ.

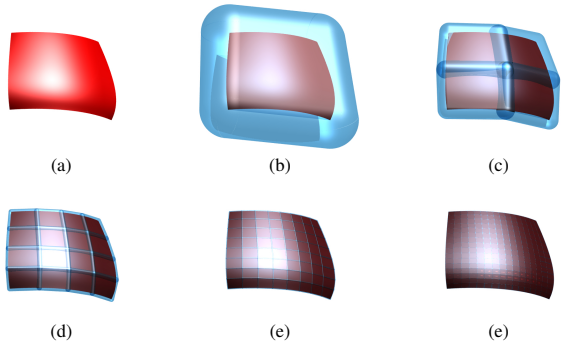
Figure 4(a) shows the Utah teapot precisely subdivided into 160 Bézier surfaces and each Bézier surface subdivided into smaller Bézier patches until each of which is within distance $1.2 * 10^{-3}$ from the bilinear surface interpolating the four corners. Figure 4(b) shows the result of subdividing the Coons patches of Figure 2(b) into smaller patches until each of which is within distance $2 * 10^{-4}$ from the bilinear surface interpolating the four corners. For the Coons patches, the maximum error from the bilinear surface is measured by adding $\epsilon_u + \epsilon_v$ as explained above.

The number of bilinear surfaces for Figure 4(a) is 5074, whereas the number for Figure 4(b) is 12864. Though the same error bound is used for both examples, the error estimation for the Coons patches



(a) (b)

Figure 4: *Approximation of Bézier surfaces and Coons patches using bilinear surfaces: (a) 160 Bézier surfaces (precisely on the Utah teapot) approximated with 5074 bilinear surfaces within an error bound* $1.2 * 10^{-3}$ *and (b) 150 Coons patches (within a distance* 10^{-3} *from the Utah teapot) approximated with 12864 bilinear surfaces within an error bound* $2 * 10^{-4}$.



(a) (b) (c)

(d) (e) (e)

Figure 5: *Bounding Coons patch with offset volumes.*

in Figure 4(b) is looser than the case for the Bézier surfaces in Figure 4(a). Nevertheless, the generation of surface points is more efficient for the Coons patches. Each point of Figure 4(a) is generated by evaluating the bivariate Bézier surfaces. On the other hand, the points of Figure 4(b) are generated as the result of combining linear interpolations of univariate boundary curve points, which is considerably more efficient to compute than the evaluation of points on a bivariate surface.

Example: Figure 5(a) shows a Coons patch on the body of the Utah teapot. The bounding volume of the Coons patch is shown in Figure 5(b) as an offset volume of the tetrahedron determined by the four corners. The result of recursive subdivisions along the u and v-directions at the mid-parameter values is shown in Figures 5(c)-(f). The maximum errors in approximating the boundary curves $S(u,0)$ and $S(u,1)$ by their respective polylines along the u-direction are 0.0562, 0.0142, 0.0037, 0.0009, and 0.0002. Moreover, the maximum errors along the v-direction are 0.0314, 0.0145, 0.0044, 0.0011, and 0.0003. Consequently, the maximum errors of the Coons patch from the bilinear surfaces in the five different levels of approximation are 0.0876, 0.0287, 0.0081, 0.0020, and 0.0005. Depending on specific applications, we may not subdivide the Coons patch in an alternating order of the u and v-directions. Moreover, the maximum level of subdivision may be different in each direction. Thus, we keep the two lists of curve approximation errors separately. For this example, we store two lists, each containing five error values, in the BVH of the Coons patch. The error values are relatively small compared with the coordinate values of other data points. Thus we store only quantized error indices. In our implementation, we have quantized the approximation error in 256 different levels and the storage of each error index takes only 1 byte. (A simple method may employ $\epsilon_0 = 1$, $\epsilon_1 = r, \ldots, \epsilon_{255} = r^{255} = 10^{-7}$, where $0 < r = 10^{(\frac{-7}{255})} < 1$.)

4 Collision Detection

For collision detection, we now employ the sphere tree representation of the BVH for freeform surfaces, where each node corresponds to a subdivided surface subpatch. The collision detection algorithm for this high-level BVH proceeds in the same way as in conventional algorithms. The only difference is that we need to evaluate the sphere center as the surface mid-point.

When we reach the leaf level, we switch to the BVH for a Coons patch, where the bounding volumes are the offsets of the tetrahedra, $O_{\epsilon_u+\epsilon_v}(V)$. For collision detection, we test the overlap between $O_{\epsilon_1}(V_1)$ and $O_{\epsilon_2}(V_2)$, for each pair of Coons patches under comparison. For an overlap test, we first compute the distance $d(V_1, V_2)$ between two tetrahedra using the efficient and robust implementation [van den Bergen 1999] of the GJK-algorithm[Gilbert et al. 1988]. The two bounding volumes $O_{\epsilon_1}(V_1)$ and $O_{\epsilon_2}(V_2)$ overlap if $d(V_1, V_2) \leq \epsilon_1 + \epsilon_2$. Otherwise, they are separated.

When the bounding volumes overlap and their sizes are relatively small, i.e., $\epsilon_1 + \epsilon_2 < 10^{-3}$, we check if the conservative distance $d(V_1, V_2) + \epsilon_1 + \epsilon_2$ is also less than 10^{-3}, the tolerance we take for the case of Coons approximation with an error bound 10^{-5}. If it is, we report a collision event. (The Hausdorff distance between V_i and the corresponding surface is less than ϵ_i; thus there are some points $\mathbf{p}_i$ from each surface with $\|\mathbf{p}_1 - \mathbf{p}_2\| = \|\mathbf{p}_1 - \mathbf{v}_1\| + \|\mathbf{v}_1 - \mathbf{v}_2\| + \|\mathbf{v}_2 - \mathbf{p}_2\| \leq d(V_1, V_2) + \epsilon_1 + \epsilon_2$, for some $\mathbf{v}_i \in V_i$.) Otherwise, we subdivide the Coons patches and continue the recursive procedure. We also report a collision event when we reach the leaf levels of the Coons BVHs under comparison. Nevertheless, we have never experienced this case in all our experiments since the resolution of our Coons BVH approximation is considerably higher than the distance 10^{-3}.

5 Minimum Distance Computation

Given two objects, A and B, their minimum distance $d(A, B)$ satisfies the following relation:

$$d(\bar{A}, \bar{B}) \leq d(A, B) = \min_{\mathbf{p} \in A} \min_{\mathbf{q} \in B} \|\mathbf{p} - \mathbf{q}\| \leq \|\mathbf{p}_A - \mathbf{q}_B\|,$$

for any $\mathbf{p}_A \in A \subset \bar{A}$ and $\mathbf{q}_B \in B \subset \bar{B}$. We take the supersets $\bar{A}$ and $\bar{B}$ as the unions of bounding volumes, and sample the points $\mathbf{p}_A$ and $\mathbf{q}_B$ on the freeform surfaces. Our algorithm recursively refines the bounding volumes so that the lower bound $d(\bar{A}, \bar{B})$ increases and the algorithm also updates the point locations $\mathbf{p}_A$ and $\mathbf{q}_B$ so that the upper bound $\|\mathbf{p}_A - \mathbf{q}_B\|$ decreases. When the difference between the two bounds drops below a certain tolerance, we report their average as the minimum distance $d(A, B)$.

For the distance computation, we employ the bounding offset volumes $O_{\epsilon_A}(V_A)$ and $O_{\epsilon_B}(V_B)$ for computing and updating the lower bound. The lower bound $\underline{d}$ is initially set to the distance between $O_{\epsilon_A}(V_A)$ and $O_{\epsilon_B}(V_B)$, which can be derived from the distance $d(V_A, V_B)$ between their tetrahedra. The GJK-algorithm for computing $d(V_A, V_B)$ also returns the minimum distance points $\mathbf{v}_A \in V_A$ and $\mathbf{v}_B \in V_B$ and their barycentric coordinates in some triangles of the tetrahedra. Using these coordinates, we can sample the corresponding surface points $\mathbf{p}_A$ and $\mathbf{q}_B$ on the original freeform surfaces (see Remark below). The upper bound $\bar{d}$ is then set to $\|\mathbf{p}_A - \mathbf{q}_B\|$.

We then refine one bounding volume to two smaller ones $O_{\epsilon_A^i}(V_A^i)$, for $i = 1, 2$, and compute the distance between $O_{\epsilon_A^i}(V_A^i)$ and $O_{\epsilon_B}(V_B)$). If the distance is larger than the upper bound $\bar{d}$, we can eliminate the corresponding pair from further consideration. Otherwise, we can compute the surface points $\mathbf{p}_A^i$ and $\mathbf{q}_B^i$ and update

the upper bound $\bar{d}$ as needed. Finally, we update the lower bound $\underline{d}$ to $\min_{i=1,2} d(O_{\epsilon_A^i}(V_A^i), O_{\epsilon_B}(V_B))$ if it is larger than the current lower bound $\underline{d}$.

Remark: Given a bilinear surface $L(u,v) = (1-u)(1-v)\mathbf{p}_{00} + (1-u)v\mathbf{p}_{01} + u(1-v)\mathbf{p}_{10} + uv\mathbf{p}_{11}$, and a boundary triangle $T(u,v) = (1-u-v)\mathbf{p}_{00} + u\mathbf{p}_{10} + v\mathbf{p}_{01}$ of the tetrahedron V,

$$L(u,v) - T(u,v) = uv(\mathbf{p}_{00} + \mathbf{p}_{11} - \mathbf{p}_{01} - \mathbf{p}_{10}).$$

The distance between $T(u,v)$ and the corresponding surface point $S(u,v)$ is thus bounded by $1.2{*}10^{-k} + uv\|\mathbf{p}_{00} + \mathbf{p}_{11} - \mathbf{p}_{01} - \mathbf{p}_{10}\|$, where a Coons patch $X(u,v)$ approximates $S(u,v)$ within 10^{-k} and $L(u,v)$ approximates $X(u,v)$ within $0.2 * 10^{-k}$, for some k. In a triangular domain $0 \le u, v, 1-u-v \le 1$, we have $0 \le uv \le 1/4$, and $uv = 1/4$ only when $u = v = 1/2$. This means that the Hausdorff distance between the bilinear surface L and its bounding tetrahedron V is exactly $\|\mathbf{p}_{00} + \mathbf{p}_{11} - \mathbf{p}_{01} - \mathbf{p}_{10}\|/4$.

6 Experimental Results

We have implemented the proposed BVH construction algorithm in C++ on an Intel Core i7-2600 3.4GHz CPU with a 16GB main memory and an NVIDIA Geforce GTX570. To demonstrate the effectiveness of the proposed BVH structure, we have also implemented real-time algorithms for collision detection and minimum distance computation for several different freeform geometric models of non-trivial complexity.

Coons BVH Construction: Table 1 reports the performance of surface approximation using the Coons patches for different levels of approximation error 10^{-k}, for $k = 2, 3, 4, 5$. Also shown are the results of approximating the boundary curves of the Coons patches using polylines within an error bound $0.1 * 10^{-k}$. This guarantees that the Coons patches are approximated by bilinear surfaces within twice the curve approximation error, and consequently the bilinear surfaces approximate the original freeform surfaces within a maximum error bound $1.2 * 10^{-k}$.

Table 1(a) shows the BVH construction result for the Utah teapot which is composed of four periodic NURBS surfaces. Each periodic surface is subdivided into two and the total number of NURBS surfaces is shown as eight in the first row. Each NURBS surface may contain a different number of piecewise polynomial/rational surfaces. Thus also shown in the first row is the total number of Bézier surfaces in the freeform model. Nevertheless, the Coons patch approximation is directly applied to the eight NURBS surfaces instead of the Bézier surfaces, which explains why the number of Coons patches can be smaller than the number of Béizer surfaces in the two low precision cases. The second row shows the size of the freeform model and the four different levels of error bound we employ for the Coons approximation. The model size is measured as the memory space for storing the NURBS control points and the knot sequences. The largest side length of the minimum bounding box for the freeform model is taken as the unit length. The items below each error bound report the total number of Coons patches at the leaf level of the surface BVH, the total number of bilinear surfaces approximating the Coons patches, and the total size of the BVH structure including the surface BVH (containing the Coons patches in the leaf nodes) as well as the Coons BVH (containing the lists of curve approximation errors). The last row shows the total BVH construction time (in seconds). Though the preprocessing algorithm is not a highly optimized one, the performance is reasonably acceptable since the BVHs can be constructed in a few seconds/minutes for all the freeform surface models we have tested in this paper.

Teapot (4KB)	#NURBS(8)		#Bézier(160)	
	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Coons	44	150	492	1,688
#Bilinear	7.8K	51.8K	636K	5.6M
Coons BVH	0.73KB	2.7KB	9.7KB	55KB
Time (sec)	0.14	0.42	1.65	8.24

(a)

B58 (12KB)	#NURBS(240)		#Bézier(266)	
	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Coons	210	276	530	1,552
#Bilinear	23.9K	75.4K	726K	10.1M
Coons BVH	2.3KB	4.1KB	9.6KB	31.9KB
Time (sec)	0.09	0.30	1.14	5.90

(b)

Eagle (102KB)	#NURBS(273)		#Bézier(9786)	
	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Coons	348	1,313	6,314	24,594
#Bilinear	463K	3.4M	39.7M	235M
Coons BVH	5.8KB	23.8KB	118KB	482KB
Time (sec)	18.2	34.1	70.9	184.7

(c)

Playground (120KB)	#NURBS(984)		#Bézier(2264)	
	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Coons	1,243	1,723	4,476	12,526
#Bilinear	284K	2.5M	10.5M	98.4M
Coons BVH	13.8KB	26.6KB	82.5KB	258KB
Time (sec)	1.5	3.8	14.1	60.0

(d)

Table 1: *Results of Coons approximation and BVH construction.*

Teapot	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Triangles	1.2K	11K	99K	938K
BVH (RSS)	320KB	2.7MB	25MB	240MB
Time (sec)	0.009	0.094	0.95	9.7

(a)

B58	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Triangles	1K	5K	45K	424K
BVH (RSS)	250KB	1.3MB	11MB	109MB
Time (sec)	0.006	0.047	0.41	4.2

(b)

Eagle	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Triangles	2K	26K	199K	1.6M
BVH (RSS)	550KB	6.5MB	51MB	421MB
Time (sec)	0.016	0.24	2	18

(c)

Playground	10^{-2}	10^{-3}	10^{-4}	10^{-5}
#Triangles	14K	112K	1.0M	9.3M
BVH (RSS)	3.6MB	29MB	259MB	2.4GB
Time (sec)	0.12	1.1	11	108

(d)

Table 2: *Comparison with conventional BVH construction.*

(a) (b)

Figure 6: *B58 and eagle models approximated with 1.6K and 25K Coons patches, respectively, within an error bound* 10^{-5}.

Table 1(b) reports the result for the B58 model shown in Figure 6(a). This model has many sharp edges as the NURBS representation has multiple knots. The initial NURBS surfaces are subdivided along these sharp edges and the periodic NURBS surfaces are also subdivided in the middle. The total number of NURBS surfaces after the subdivision is shown in the first row of Table 1(b), together with the total number of Bézier surfaces. Table 1(c) is the result for the eagle model shown in Figure 6(b). This model contains a total of 9786 bicubic Bézier surfaces. The large number is mainly due to the irregular shape of the eagle model, whose NURBS representation requires relatively long knot sequences.

Figure 7 contains several different freeform models for the structures commonly found in the playground for kids. The slide is shown twice, once with the sand model. There are also two instances of the t-swing. Each model is taken separately in the BVH construction. We take the unit length of each model as the largest side length of the individual minimum bounding box. Table 1(d) shows the result of the BVH construction for all the NURBS surfaces included in the freeform models of the playground, where the slide and the t-swing are counted twice as they appear twice.

Efficient geometric computations: Figure 1(a) shows a snapshot from animation clips we have generated to demonstrate the performance of our collision detection algorithm. Employing Open Dynamics Engine (ODE), an open source library for simulating rigid body dynamics (http://www.ode.org), we have conducted a dynamic simulation for 10000 chess pieces falling to a pile. The penetration depth was computed using the technique of [Guendelman et al. 2003], which is based on a precomputed distance field in the interior of each model. In this simulation, each frame was generated in 30 seconds on average. In addition to the computing time for collision detection, this includes the dynamic simulation time by the ODE system as well as the graphics rendering time. Figure 1(b) shows the B58 model flying in a complex dynamic scene that contains many Utah teapots falling to the playground. The minimum distance was computed between the flying B58 model and the freeform models in the dynamic scene. In this example, we have observed the performance of generating each frame in less than 30 milliseconds. In all the tests reported in this paper, we have used the Coons patch approximations of the freeform models with the error bound 10^{-5}.

Performance comparison with PQP: There is a fundamental limitation of conventional bounding volumes in representing the BVH of freeform geometric models. Since there is no clear termination condition for the leaf level, one has to deal with freeform surfaces at the end of the BVH traversal. Because of the exponential growth of the conventional BVH size, the maximum BVH level is also severely limited. (On the other hand, the Coons BVH exhibits only a linear growth of its size.) Our trial to approximate the Utah teapot using an RSS tree has not been successful when we bounded, as the termination condition, the Hausdorff distance between an RSS and the surface patch within a given tolerance.

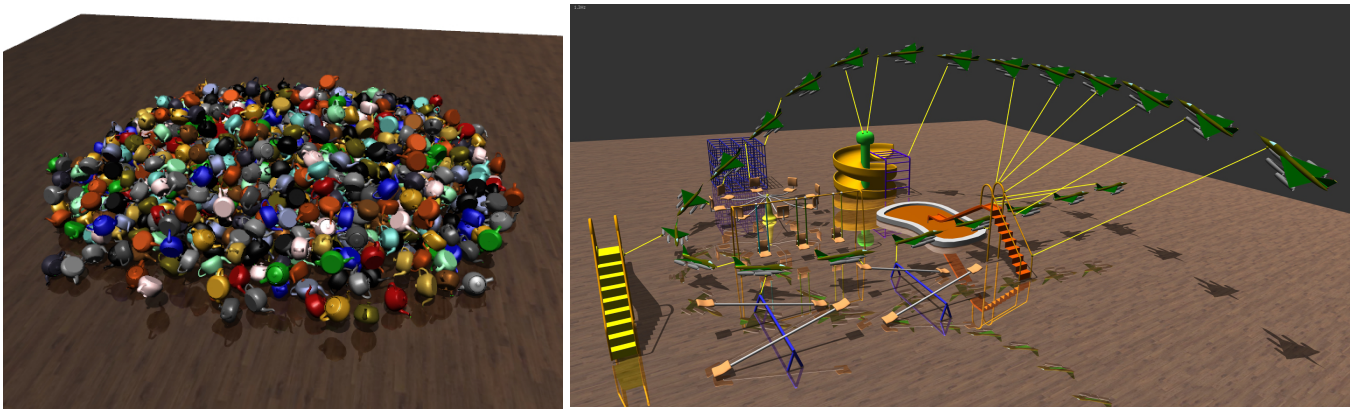
As a reasonable alternative for performance comparison with the conventional BVH schemes, we have thus employed polygonal approximations to the freeform geometric models. Using the IRIT library function `CagdSrfAdap2Polygons` [IRIT 10.0], we have generated triangular meshes approximating the test freeform surfaces within given tolerances in terms of the Hausdorff distance measure. The BVH of each polygonal model is then constructed using the PQP library (http://gamma.cs.unc.edu/SSV/) that is based on [Larsen et al. 1999]. The BVH construction result is reported in Table 2, where the BVH size includes only the BVH node data but not the input triangles. We can notice that the Coons BVH is considerably more compact than the RSS tree, which often shows more than 1000 times size reduction for the high-precision case of error bounds 10^{-k}, for $k = 4, 5$.

To measure the relative performance of our algorithms (based on the Coons BVH) against similar algorithms implemented in the PQP library (using only the RSS tree of [Larsen et al. 1999], not activating the dual OBB tree part of the PQP system), we have conducted two comparison tests as shown in Figure 7: (a) collision detection for a dynamic simulation of 1000 Utah teapots falling to a pile, and (b) the minimum distance computation for a B58 model flying over the playground. In these two tests, we have chosen the triangular meshes with approximation error 10^{-3}, whose BVHs require relatively small memory space compared with other high-precision cases. On the other hand, we employ the Coons BVHs with approximation error $1.2 * 10^{-5}$ and set the tolerance of collision detection to $2 * 10^{-3}$. Note that the PQP collision detection also guarantees the same tolerance for the minimum distance between two original surface models.

The two charts of Figure 8 report the comparison results for the relative performance of the two tests. The graphs in blue correspond to the performance of the PQP library in computing time. The graphs in green are for the algorithms based on the Coons BVH. The result in green reports the computing time, including the dynamic generation of Coons BVH nodes as well as other geometric computations. The nodes dynamically generated are immediately discarded to save the runtime memory space for the Coons BVH. Nevertheless, we keep the basis function values $B_a(u)$ and $B_b(v)$ evaluated at different parameters u and v, which are quite expensive to recompute but can be saved only in a small space. By traversing the Coons BVH in a preprocessing step, we can precompute and store the function values $B_a(u)$ and $B_b(v)$ up to a certain level. This technique greatly accelerates the runtime performance of our algorithm. Figure 9 shows the growth of the memory space for the three models in the two tests of Figure 7. Note that the initial memory size is bigger than the BVH size for each model, since this includes the NURBS control points and knot sequences and we also do some precomputing for the basis function values and other auxiliary data structures for the BVH. Nevertheless, these precomputations usually take less time than loading the mesh models.

7 Conclusions

We have presented a compact BVH representation scheme for freeform surfaces. The BVH size is often more than 1000 times smaller than those for polygonal meshes of similar complexity. Thus the proposed BVH representation has a great potential in mobile applications where compact representations are important not only for the object modeling itself but also for the associated data structures. Even though we have demonstrated the efficiency of our algorithms for freeform models, there is a large room for further improvement as our approach employs a large number of point evaluations on freeform curves and surfaces. In future work, we plan to improve the performance of our algorithms by developing new acceleration techniques, including GPU accleration.



(a) (b)

Figure 7: *Performance comparison tests against the PQP library: (a) collision detection in a dynamic simulation for 1000 Utah teapots falling to a pile, and (b) the minimum distance computation between a flying B58 model and the playground.*

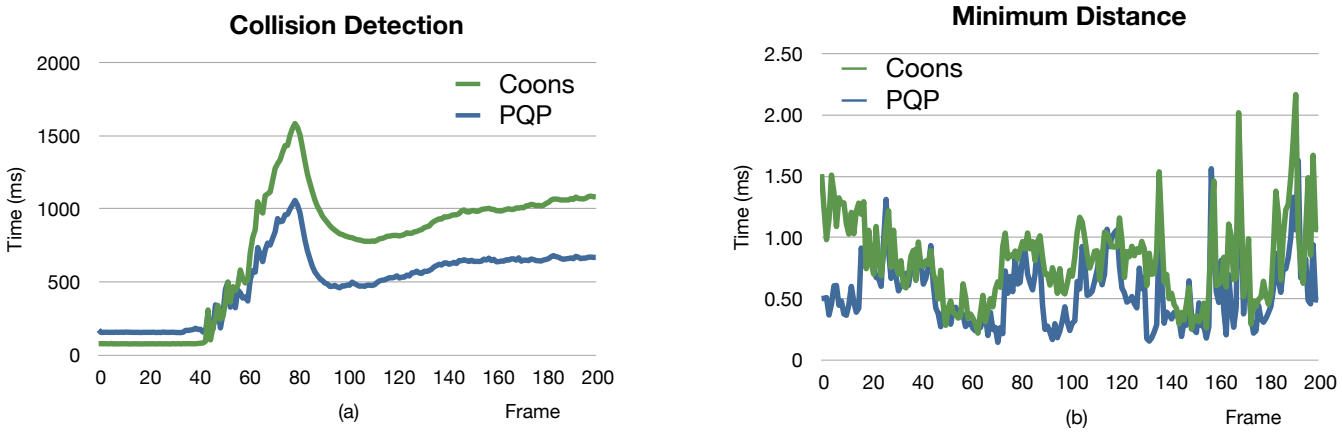


Figure 8: *Performance comparison results for the two tests: (a) collision detection and (b) minimum distance computation.*

Acknowledgments: The authors would like to thank the anonymous reviewers for their invaluable comments. This research was supported in part by the Israeli Ministry of Science Grant No. 3-8273, and in part by NRF Research Grants (Nos 2010-00787, 2011-0003474, and 2011-0018017).

References

Akenine-Möller, T., Hains, E., Hoffman, N.: *Real-Time Rendering*, A.K. Peters, Natick, MA, 3rd Ed., 2008.

Cohen, E., Riesenfeld, R., and Elber, G.: *Geometric Modeling with Splines: An Introduction*, A.K. Peters, Natick, MA, 2001.

Coons, S.: *Surfaces for Computer-Aided Design*, Technical report, MIT, 1964. Available as AD 663 504 from the National Technical Information Service, Springfield, VA, 22161.

Farin, G.: *Curves and Surfaces for CAGD*, 5th Ed., MorganKaufmann, San Francisco, CA, 2002.

Gilbert, E., Johnson, D., Keerthi, S.: A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Trans. Robot. Automat. 4*, 2, 193–203, 1988.

Gottschalk, S., Lin, M., Manocha, D.: OBB-tree: a hierarchical structure for rapid interference detection. *Computer Graphics (SIGGRAPH 1996)*, 171–180, 1996.

Govindaraju, N., Redon, S., Lin, M., Manocha, D.: Cullide: interactive collision detection between complex models in large environments using graphics hardware. *Proc. Eurographics/SIGGRAPH Graphics HardwareWorkshop*, pp. 25–32, 2003.

Guendelman, E., Bridson, R., Fedkiw, R.: Nonconvex Rigid Bodies with Stacking. *Proc. of SIGGRAPH 03*, Computer Graphics Annual Conference Series, 2003.

IRIT 10.0 User's Manual, Technion. http://www.cs.technion. ac.il/∼irit.

James, D., Pai, D.: Bd-tree: output-sensitive collision detection for reduced deformable models. *ACM Trans. on Graphics 23*, 3, 393–398, 2004.

Johnson, D., Cohen, E.: A framework for efficient minimum distance computations. *IEEE Int'l Conf. on Robotics and Automation*, 3678-3684, 1998.

Kim, D., Heo, J.-P., Huh, J., Kim, J., Yoon, S.-E.: HPCCD: Hybrid parallel continuous collision detection using CPUs and GPUs. *(Proc. of Pacific Graphics 2009)*, *Computer Graphics Forum 28*, 7, 1791–1800, 2009.

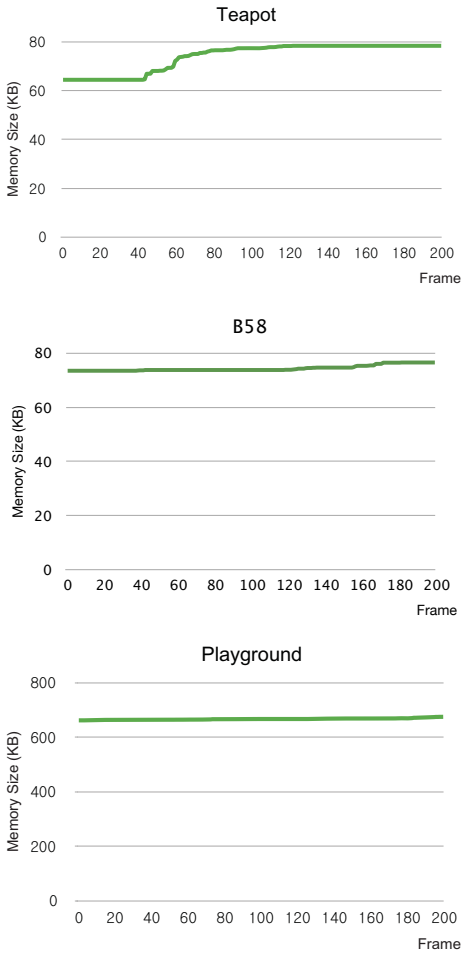


Figure 9: *Memory usage of three models in the two tests.*

Krishnan, S., Gopi, M., Lin, M., Manocha, D., Pattekar, A.: Rapid and accurate contact determination between spline models using ShellTrees. *Computer Graphics Forum 17*, 3, 315–326, 1998.

Larsen, E., Gottschalk, S., Lin, M.C., Manocha, D.: Fast proximity queries using swept sphere volumes. Technical Report TR99-018, Dept. of Computer Science, UNC, 1999.

Lin, M.C., Gottschalk, S.: Collision detection between geometric models: A survey. *Proc. of IMA Conference on Mathematics of Surfaces*, pp. 37–56, 1998.

Lin, M.C., Manocha, D.: Collision and proximity queries. *Handbook of Discrete and Computational Geometry*, 2nd Ed., J.E. Goodman and J. O'Rourke, Eds., Chapman & Hall/CRC, pp. 787–807, 2004.

Redon, S., Kim, Y., Lin, M., Manocha, D.: Fast continuous collision detection for articulated models. *Proc. ACM Symp. on Solid Modeling and Applications*, pp. 145–156, 2004.

Samet, H.: *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann, San Francisco, CA, 2006.

Tang, M., Curtis, S., Yoon, S.-E., Manocha, D.: Interactive continuous collision detection between deformable models using connectivity-based culling. *SPM '08: Proc. of ACM Symp. on Solid and Physical Modeling*, Stony Brook, New York, pp. 25–36, 2008.

Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M.-P., Faure, F., Magnenat-Thalmann, N., Strasser, W., Volino, P.: Collision detection for deformable objects. *Computer Graphics Forum 24*, 1, 61–81, 2005.

van den Bergen, G.: A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools 4*, 2, 7–25, 1999.

Yoon, S.-E., Manocha, D.: Cache-efficient layouts of bounding volume hierarchies. *Computer Graphics Forum 25*, 3, 507–516, 2006.

Zhang, X., Kim, Y.: Interactive collision detection for deformable models using streaming AABBs. *IEEE Trans. on Visualization and Computer Graphics 13*, 2, 318–329, 2007.

A Bounding the Error for Coons Patches

A Coons patch $X(u,v)$ bilinearly interpolates its four boundary curves. For a fixed $\hat{v}$, the iso-curve $X(u,\hat{v})$ is given as follows

$$\begin{aligned} X(u,\hat{v}) \;=\; & (1-u)X(0,\hat{v}) + uX(1,\hat{v}) \\ & +(1-\hat{v})X(u,0) + \hat{v}X(u,1) \\ & -[(1-u)(1-\hat{v})X(0,0) + (1-u)\hat{v}X(0,1) \\ & \quad +u(1-\hat{v})X(1,0) + u\hat{v}X(1,1)]. \end{aligned}$$

Now we consider the difference between the iso-curve $X(u,\hat{v})$ and the line segment connecting the two end points:

$$\begin{aligned} & X(u,\hat{v}) - [(1-u)X(0,\hat{v}) + uX(1,\hat{v})] \\ = \; & (1-\hat{v})X(u,0) + \hat{v}X(u,1) \\ & -[(1-u)(1-\hat{v})X(0,0) + (1-u)\hat{v}X(0,1) \\ & \quad +u(1-\hat{v})X(1,0) + u\hat{v}X(1,1)] \\ = \; & (1-\hat{v})[X(u,0) - (1-u)X(0,0) - uX(1,0)] \\ & +\hat{v}[X(u,1) - (1-u)X(0,1) - uX(1,1)]. \end{aligned}$$

Thus the maximum deviation of the iso-curve $X(u,\hat{v})$ is a convex combination of those for the two boundary curves $X(u,0)$ and $X(u,1)$ from their respective line segments.

Now we can derive the maximum deviation of the Coons patch $X(u,v)$ from the bilinear surface $L(u,v)$ interpolating the four corners as follows:

$$\begin{aligned} & X(u,v) - L(u,v) \\ = \; & (1-u)X(0,v) + uX(1,v) - L(u,v) \\ & +(1-v)X(u,0) + vX(u,1) - L(u,v) \\ = \; & (1-u)[X(0,v) - (1-v)X(0,0) - vX(0,1)] \\ & +u[X(1,v) - (1-v)X(1,0) - vX(1,1)] \\ & +(1-v)[X(u,0) - (1-u)X(0,0) - uX(1,0)] \\ & +v[X(u,1) - (1-u)X(0,1) - uX(1,1)]. \end{aligned}$$

The Coons subpatch $X(u,v)$, for $u_i \le u \le u_{i+1}$ and $v_j \le v \le v_{j+1}$, is bounded by four iso-curves $X(u_i,v)$, $X(u_{i+1},v)$, $X(u,v_j)$, and $X(u,v_{j+1})$. The maximum deviation of the Coons subpatch from the bilinear surface $L_{ij}(u,v)$ with four corners: $X(u_i,v_j)$, $X(u_i,v_{j+1})$, $X(u_{i+1},v_j)$, and $X(u_{i+1},v_{j+1})$, can be bounded by a similar combination of the maximum deviations of the iso-curve segments from the four edges of the bilinear surface. This can be proven by repeatedly subdividing the Coons patch along each of the four iso-curves and applying the two properties shown above.