Architectural Specification: High-Fidelity Swept Volume Collision Detection for Salvagnini P4 Panel Bending Simulation

1. The Physics Gatekeeper: Architectural Mandate and Operational Context

In the domain of automated industrial robotics, specifically within the high-precision environment of the Salvagnini P4 Panel Bender, the transition from kinematic sequencing to physical execution represents a critical juncture where theoretical possibility meets physical reality. Phase 4 of the simulation pipeline, designated as the "Physics Gatekeeper," operates with a singular, absolute mandate: to validate the rigorous physical feasibility of every candidate "Bend Sequence" generated by the Phase 3 Sequencer. The Sequencer, operating on heuristics and combinatorial logic, produces candidate branches—hypothetical futures where a sheet of metal transforms into a finished panel. However, the Sequencer lacks the granular physical awareness to detect dynamic interference. It is the responsibility of the Physics Gatekeeper to simulate these futures with high fidelity, detecting catastrophic machine interferences before they manifest on the factory floor.

The simulation of the Salvagnini P4 is not a trivial static interference problem. The machine architecture involves complex, interpolated movements where the manipulator, holding the sheet via suction cups, synchronizes with the universal bending tools—the upper and lower blades and the segmented Automatic Blankholder Adjustment (ABA) mechanism.[1] A static collision check, which merely verifies that the part does not intersect the machine at the start (t_0) and end (t_{target}) of a step, is fundamentally insufficient. The flange of the sheet metal travels through a continuous kinematic arc during the bending process. This transient motion creates a "Swept Volume"—a temporal solid representing the total space occupied by the flange during the operation.[3] If this swept volume intersects with the segmented ABA tool body (in a "Bend Up" operation) or the machine table (in a "Bend Down" operation), the sequence is invalid.

The architectural challenge is further compounded by the real-time constraints of the simulation. The Phase 3 Sequencer generates thousands of potential branches to find an optimal path. Consequently, the Phase 4 Validation engine must operate with sub-millisecond latency per check (< 1 ms), rejecting invalid states almost instantaneously. This necessitates a move away from brute-force boolean operations on B-Rep (Boundary Representation)

solids toward a sophisticated, hierarchical collision detection strategy. We must architecture a Computational Physics Engine in C++ using the OpenCASCADE Technology (OCCT) kernel, leveraging Hierarchical Bounding Volume (BVH) trees, kinematic swept solid generation via BRepPrimAPI, and lazy-evaluation strategies to manage the dynamic state of the ABA tooling.[4]

This report details the comprehensive architecture for this engine, defining the mathematical models for swept volume generation, the data structures for spatial partitioning, and the algorithmic logic for providing actionable feedback—such as "Repo" (repositioning) directives—back to the Sequencer upon validation failure.

2. Geometric and Kinematic Modeling of the Salvagnini P4

To construct a valid simulation, one must first establish a rigorous mathematical definition of the collision environment. The Salvagnini P4 is a dynamic system where both the workpiece and the machine topology mutate over time.

2.1 The Manipulator and Dynamic Part State

The workpiece is not a static mesh; it is a dynamic TopoDS_Shape that evolves with every bend. The simulation tracks the "MachineState," which includes the manipulator's gripping coordinates and rotation angle. The manipulator itself—comprising the gripper frame, rotator, and suction cups—is a rigid body that translates and rotates in the XY plane.[7]

The manipulator's primary collision risk arises during the "Rotate" action. When the manipulator rotates the sheet to align a new side with the bend line, the simulation must validate that the swinging sheet (and the manipulator structure) does not strike the ABA blankholder towers or the side references. This rotation defines a swept volume distinct from the bending sweep—a "Swept Prism" or cylindrical sector generated by the rotation of the entire sheet assembly around the manipulator axis (Z). However, the focus of this specific architectural phase is the "Bend" action, where the risk is highest due to the close proximity of the deforming flange to the clamping tools.

2.2 The ABA Blankholder: A Dynamic, Segmented Obstacle

The Automatic Blankholder Adjustment (ABA) is the most complex obstacle in the simulation environment. Unlike traditional press brake tooling, which is often static, the ABA is composed of modular steel segments that engage or disengage to match the length of the bend.[8] The tool width is a variable in the MachineState input.

The segmentation of the ABA presents a unique challenge for collision detection. It cannot be modeled as a single bounding box because the "gaps" between segments are often used as

safe zones. For instance, a part with a pre-bent internal tab might maximize the ABA width such that the tab slides exactly into a gap between two segments.[1] Therefore, the collision engine must model the ABA not as a monolithic solid, but as a BVH_ObjectSet of discrete segments.

Table 1 illustrates the operational constraints of the Salvagnini P4 tooling which must be encoded into the collision geometry.[9]

Parameter	Value / Range	Implication for Collision Engine
Tooling Type	Universal Bending Tools	Dynamic adaptability requires real-time BVH refitting.
Material Thickness	0.4mm – 3.2mm	Collision tolerances must account for thickness + deflection.
Bend Angle Range	$\pm 90^{\circ}$ to $\pm 135^{\circ}$	Swept volumes can exceed 90° sectors; validation needed for re-entrant bends.
ABA Segmentation	5mm increments	High-resolution collision grid required; cannot use coarse voxelization.
Min. Clearance	Variable (approx. 0.1mm)	Mesh deflection in simulation must be $<$ to prevent false positives.

The "MachineState" input provided by the Sequencer defines the active configuration of these segments. The Physics Engine must interpret this state to "mask" or "unmask" segments in the collision world before performing any overlap tests.

2.3 The Bending Blades and Interpolated Motion

The bending blades (Upper and Lower) are the active agents of deformation. In a "Bend Up" cycle, the lower blade moves upward, contacting the flange and forcing it to rotate around the

radius of the ABA nose.[11] In a "Bend Down" cycle, the upper blade descends, forcing the flange around the counterblade.

While the blades themselves are collision objects, the *primary* geometric concern is the path they force the flange to take. The Physics Engine simplifies the complex interpolated motion of the blade into the resultant kinematic path of the flange. For standard air bending on the P4, this path is sufficiently approximated as a pure rotation around the bend line (the edge of the ABA or Counterblade). However, the engine must support the definition of the "Swept Solid" as the volume traversed by the flange face during this rotation.[4]

3. Architecting the Swept Volume: The "Swept Solid"

The core algorithmic challenge of Phase 4 is the generation of the "Swept Solid." This 3D volume represents the continuum of space occupied by the flange as it moves from its initial flat state ($\theta = 0$) to its final folded state ($\theta = \alpha_{target}$). This is a classic "Continuous Collision Detection" (CCD) problem, typically solved in robotics by analyzing the swept volume.[3]

3.1 Kinematic Sweep Strategies: Linear vs. Rotational

The OpenCASCADE kernel offers multiple mechanisms for generating swept topologies. The choice of algorithm dictates both the accuracy of the simulation and its computational cost.

3.1.1 The "Prism" Approach (Linear Sweep)

The BRepPrimAPI_MakePrism class creates a linear sweep (extrusion) of a shape along a vector.[4]

- **Application:** This is appropriate for validating the "Approach" and "Retract" phases of the machine cycle, where the manipulator moves the part linearly along the X or Y axis to position it against the reference stops.
- **Limitation:** It cannot represent the bending action. A flange bending up traces a circular arc, not a straight line. Using a linear prism to approximate a bend would result in massive inaccuracies—either underestimating the collision volume at the arc's apex or overestimating it at the chords.

3.1.2 The "Revol" Approach (Rotational Sweep)

The BRepPrimAPI_MakeRevol class creates a rotational sweep of a shape around an axis.[4]

- **Application:** This is the mathematically correct model for the P4 bending process. The flange rotates around the bend axis defined by the ABA tool edge.
- **Implementation:** The algorithm takes the TopoDS_Face of the flange and a gp_Ax1 representing the bend line. It generates a TopoDS_Solid bounded by the original face, the final face, and the lateral surfaces generated by the revolution of the flange's edges.

- **Why it is Essential:** Unlike static interference checks, MakeRevol captures the intermediate states. A long flange might clear the machine table at the start (0°) and the end (-90°), but collide with the edge of the table at -45°. Only the rotational sweep guarantees detection of this transient crash.

3.1.3 The "Pipe" Approach (General Sweep)

The BRepOffsetAPI_MakePipe class sweeps a profile along an arbitrary wire spine.[14]

- **Application:** This would be necessary if the P4 executed complex "Bump Bending" or variable-radius bending where the center of rotation shifts dynamically.
- **Decision:** For standard bending sequences, the motion is strictly rotational. The overhead of MakePipe (which involves complex frame transport calculus) is unjustified compared to the optimized MakeRevol.

3.2 Constructing the Kinematic Sweep in C++

The implementation of the swept volume generation must handle the extraction of the relevant geometry from the part. We cannot sweep the *entire* part; we must identify the "Flange" sub-shape.

C++

```
// Architectural Pseudocode for Swept Volume Generation
TopoDS_Shape PhysicsGatekeeper::GenerateSweptVolume(const TopoDS_Shape& partShape,
                                    const MachineState& state,
                                    const BendStep& step)
{
    // Step 1: Isolate the Flange Geometry
    // The Flange is the set of faces on the 'active' side of the bend line.
    TopoDS_Shape flangeFaces = GeometryUtils::ExtractFlange(partShape, step.BendLine);

    // Step 2: Define the Axis of Revolution
    // The axis is aligned with the bend line, anchored at the ABA tool tip.
    gp_Ax1 bendAxis(step.BendLine.Location(), step.BendLine.Direction());

    // Step 3: Determine the Sweep Angle
    // This includes over-bending for springback if necessary.
    Standard_Real sweepAngle = step.TargetAngle * (Math::PI / 180.0);

    // Step 4: Generate the Swept Solid (The "Revol")
```

```
    // We use BRepPrimAPI_MakeRevol to create the collision volume.
    BRepPrimAPI_MakeRevol sweeper(flangeFaces, bendAxis, sweepAngle);

    // Critical Optimization:
    // We strictly do not need the internal topology (solids within solids).
    // We only need the outer shell for collision.
    TopoDS_Shape sweptVolume = sweeper.Shape();

    // Step 5: Tessellation for Collision
    // Raw B-Rep collision is slow. We generate a mesh proxy immediately.
    // 0.5mm deflection is sufficient for Broad Phase; 0.05mm for Narrow Phase.
    BRepMesh_IncrementalMesh meshGen(sweptVolume, 0.1);

    return sweptVolume;
}
```

3.3 The Convex Hull Optimization (Implicit Sweep)

Generating a precise B-Rep swept solid via MakeRevol is computationally expensive (approx. 5-50ms depending on complexity). For a sequencer requiring <1ms performance, this is a bottleneck. We introduce a "Level-of-Detail" (LOD) strategy using Convex Hulls.[16]

Instead of a full rotational sweep, we compute the **Swept Convex Hull**.

1. Compute the Convex Hull of the flange at the Start Position (H_{start}).
2. Compute the Convex Hull of the flange at the End Position (H_{end}).
3. Compute an intermediate Convex Hull at the arc midpoint (H_{mid}) to capture the curvature (preventing the hull from "short-cutting" through the obstacle).
4. Compute the overarching Convex Hull of $H_{start} \cup H_{mid} \cup H_{end}$.

This "Convex Sweep" is a watertight proxy. If the Convex Sweep does not collide with the ABA, the actual flange definitely does not. If it *does* collide, we escalate to the expensive MakeRevol generation for a precise check. This hierarchical approach filters out 90% of safe cases in microseconds.

4. The Collision Environment: Hierarchical Bounding Volume (BVH) Architecture

To achieve the <1ms rejection target, the collision detection cannot be $O(N)$ (linear check against all machine parts). It must be $O(\log N$. This requires a Spatial Partitioning strategy

utilizing Hierarchical Bounding Volumes (BVH).

4.1 Comparison of Bounding Volumes: AABB vs. OBB

The choice of bounding volume is critical for the efficiency of the traversal.

- **Axis-Aligned Bounding Box (AABB):** Defined by min/max coordinates $(x_{min}, y_{min}, z_{min})$ and $(x_{max}, y_{max}, z_{max})$.
  - *Pros:* Extremely fast to build and intersect. Perfect for the **Static Machine Table** and the **ABA Segments** (which are aligned to the machine axes).[6]
  - *Cons:* Poor fit for rotated objects. A flange bent at 45° has a massive AABB with large "dead space," leading to high false-positive rates in the broad phase.[19]
- **Oriented Bounding Box (OBB):** Defined by a center, extents, and a rotation basis.
  - *Pros:* Tighter fit for the moving flange.
  - *Cons:* More expensive intersection math.

Architectural Decision: We implement a **Hybrid BVH Strategy**.

- **The Environment (ABA, Table):** Uses an **AABB Tree** (BVH_Tree<Standard_Real, 3>). The machine components are axis-aligned.
- **The Swept Volume:** Is wrapped in an **OBB** for the broad phase query.
- **The Query:** We perform an OBB-AABB overlap test to descend the tree.

4.2 Dynamic BVH: Handling ABA Resizing

The ABA blankholder is dynamic. In a simulation sequence, Step 1 might use a 500mm tool, and Step 2 might resize it to 300mm. Rebuilding the BVH tree from scratch (Build()) for every step is inefficient.

We utilize the **BVH Refitting** strategy available in OpenCASCADE.[20]

- **Initialization:** At the start of the simulation, we build a BVH containing *all* potential ABA segments (e.g., all 50 segments).
- **State Update:** When the MachineState changes (ABA Resize), we do not delete nodes. We simply:
  1. Update the coordinates of the active segments to their new positions.
  2. Move the "Inactive" segments to a "Holding Bin" (a coordinate far outside the collision volume, e.g., $Z = -9999$).
  3. Call m_abaBVH->Refit(). This function recalculates the bounding boxes bottom-up without restructuring the tree topology. This operation is orders of magnitude faster than Build().[21]

4.3 The "Lazy Evaluation" Pipeline

The Physics Gatekeeper employs a lazy evaluation pipeline to minimize computation:

1. **Cache Lookup:** Check if this specific (PartID, FlangeID, Angle, MachineStateID) has been validated previously (Memoization).
2. **Global AABB Check:** Check the AABB of the *entire* swept path against the AABB of the *entire* machine. If no overlap, return Valid.
3. **Broad Phase (BVH):** Traverse the Machine BVH with the Swept OBB. Collect a list of candidate collision pairs (e.g., "Segment #4 vs. Flange").
4. **Narrow Phase (Mesh):** Perform Triangle-Triangle intersection between the tessellated sweep and the candidate segments.[23]
5. **Exact Phase (B-Rep):** (Optional) Only if the Mesh phase reports a collision within the tolerance window ϵ, perform BRepExtrema_DistShapeShape for confirmation and depth calculation.

5. Critical Collision Pairs: Analysis and Algorithms

The collision engine must be tuned to detect specific failure modes inherent to the P4 panel bending process.

5.1 Scenario A: "Bend Up" (Swept Flange vs. ABA Tool Body)

This is the most frequent crash scenario in automated panel bending.

- **The Physics:** During a positive bend (Bend Up), the flange curls upward around the ABA nose. If the flange is longer than the clearance height of the ABA tool, or if the part has lateral features (side tabs), the flange will strike the face of the blankholder.
- **Geometric Nuance:** The collision often happens with the *side* of the ABA segments if the tool width is not perfectly matched to the part width.
- **Algorithm:**
    - Construct the Swept_Solid for the range $[0, \alpha_{target}]$.
    - Query the ABA_BVH.
    - **Specific Check:** Analyze the intersection normal. If the normal is opposing the bend direction, it is a hard crash.
    - **Implication:** A crash here implies the Sequencer must select a narrower tool width (if possible) or perform a "Repo" to shift the part laterally to align the flange with a gap in the segmented tool.[10]

5.2 Scenario B: "Bend Down" (Swept Flange vs. Machine Table)

- **The Physics:** During a negative bend (Bend Down), the flange swings below the bending plane.
- **The Obstacle:** The Machine Table and the Lower Blade Holder form a static floor plane ($Z = Z_{table}$).

- **Algorithm:**
  - This check is simplified. We do not need complex BVH traversal.
  - Check the global Z_{min} of the Swept_Solid.
  - If $Z_{min} < Z_{table} + \text{SafetyMargin}$, the bend is physically impossible.
  - **Implication:** This is a hard constraint. The Sequencer must likely flip the part (turn it over) earlier in the sequence to perform this bend in a positive direction, or use a "P-Tool" (Auxiliary Tool) if available.[7]

5.3 Scenario C: Manipulator Rotation vs. ABA Towers

- **The Physics:** Between bends, the manipulator rotates the part.
- **The Obstacle:** The tall "Towers" of the ABA blankholder structure.
- **Algorithm:**
  - Generate a BRepPrimAPI_MakePrism (or Cylindrical Sector) representing the rotation of the part's bounding box.
  - Check against the static "Machine Frame" BVH.
  - **Implication:** If collision occurs, the Sequencer must retract the manipulator (move along $-Y$) before rotating, then re-approach.

6. Algorithmic Implementation: The C++ Architecture

The implementation leverages the TKMath, TKBRep, and TKGeomAlgo toolkits of OpenCASCADE.

6.1 The Broad Phase: BVH_Traverse

We subclass BVH_Traverse to implement the overlap logic between our Swept OBB and the Machine AABB Tree.

C++

```
// Custom Traversal Class for Collision Detection
class CollisionTraverse : public BVH_Traverse<Standard_Real, 3> {
public:
  CollisionTraverse(const BVH_Box<Standard_Real, 3>& sweptBox)
    : m_sweptBox(sweptBox) {}

  // Rejection Metric: Is the node's box overlapping with our swept box?
  virtual Standard_Boolean RejectNode(const BVH_Vec3d& cMin,
```

```
                          const BVH_Vec3d& cMax,
                          Standard_Real& metric) const override
  {
    BVH_Box<Standard_Real, 3> nodeBox(cMin, cMax);
    return!m_sweptBox.HasIntersection(nodeBox);
  }

  // Leaf Acceptance: We found a potential collision candidate
  virtual Standard_Boolean Accept(const Standard_Integer index,
                          const Standard_Real& metric) override
  {
    m_candidates.push_back(index);
    return Standard_True; // Continue traversal to find all collisions
  }

  const std::vector<int>& GetCandidates() const { return m_candidates; }

private:
  BVH_Box<Standard_Real, 3> m_sweptBox;
  std::vector<int> m_candidates;
};
```

6.2 The Narrow Phase: BRepExtrema_DistShapeShape Optimization

As noted in research [24], BRepExtrema_DistShapeShape can be a performance bottleneck if used naively on B-Spline surfaces. It attempts to find the exact mathematical minimum distance, which involves solving high-degree polynomials.

Optimization Strategy:

1. **Poly Reference:** Instead of passing the TopoDS_Shape, we pass the Poly_Triangulation extracted via BRep_Tool::Triangulation.
2. **Early Exit:** We configure the distance tool with a Deflection parameter. If the algorithm finds *any* point pair with distance $< \text{Tolerance}$, it aborts and returns IsDone = True with distance 0. We do not need to find the *global* minimum if we have already found a crash.
3. **Mesh Quality:** We use BRepMesh_IncrementalMesh with a deflection of 0.1mm. This ensures the mesh is tight enough to the surface to prevent false collisions while remaining sparse enough for fast checking.[26]

7. Feedback Loop: Correction and "Repo" Logic

The Physics Gatekeeper is not merely a validator; it is a guidance system. When a collision is detected, the engine must calculate a "Correction Vector" to inform the Phase 3 Sequencer's

recovery logic.

7.1 Analyzing the Collision

When ValidateStep returns failure, we perform a deeper analysis:

- **Penetration Depth (d):** Calculated using BRepExtrema_DistShapeShape (or GJK on convex sub-parts).
- **Impact Normal (n):** The direction of the surface at the collision point.

7.2 Generating the Repo (Repositioning) Directive

If the collision is "Side Flange vs. ABA Segment," the engine calculates the lateral shift required to clear the obstacle.

- **Logic:**
  1. Identify the colliding ABA segment (S_{id}).
  2. Check the adjacent gaps in the ABA configuration.
  3. Calculate vector $v = \text{GapCenter} - \text{FlangeCenter}$.
  4. **Feedback:** Recommendation: REPO(X + v.x).

The Sequencer receives this recommendation and attempts to inject a "Repo" step into the sequence *before* the failing bend. This closes the loop between simulation and planning, allowing the system to autonomously solve complex interference problems without human intervention.

8. Conclusion

The architecture of the Swept Volume Collision Detection system for the Salvagnini P4 represents a significant advancement over static interference checking. By explicitly modeling the kinematic sweep using BRepPrimAPI_MakeRevol, the system captures the transient collision states inherent in high-speed panel bending. The integration of a Hierarchical Bounding Volume (BVH) strategy with dynamic refitting allows the engine to accommodate the mutable state of the segmented ABA tooling while maintaining the <1ms performance requirement dictated by the Sequencer.

Furthermore, the "Physics Gatekeeper" transcends simple validation by providing geometric feedback vectors that drive the "Repo" logic, actively contributing to the generation of successful robot programs. This architectural specification ensures that the simulation is not just a passive verification step, but an active participant in the path planning process, guaranteeing that the generated bend sequences are not only theoretically optimal but physically executable on the shop floor.

Table 2: Algorithmic Complexity and Performance Targets

Simulation Phase	Algorithm Used	Complexity	Target Latency
Broad Phase	AABB Tree Traversal (BVH_Traverse)	$O(\log N$	$<$ ms
Mid Phase	Convex Hull Sweep Overlap	$O(1$	$<$ ms
Narrow Phase	Mesh-Mesh Intersection (OverlapTool)	$O(M \cdot$	$<$ ms
Exact Phase	B-Rep Distance (DistShapeShape)	$O(P^2$	$<$ ms (Rare)

Citations: [4] OpenCASCADE BRepPrimAPI Documentation. [24] BRepExtrema Performance Analysis. [5] OpenCASCADE BVH and Refitting Mechanics. [3] Swept Volume Collision Detection Theory. [1] Salvagnini P4 Machine Topology and ABA Mechanics. [16] Convex Hull Swept Volume Optimization. [18] AABB vs. OBB for Rotational Bodies.

Works cited

1. p4-benders.pdf - Empire Machinery, accessed February 3, 2026, https://www.empire-machinery.com/wp-content/uploads/2021/06/p4-benders.pdf
2. Salvagnini panel bending: P4-2520 bends 4 different parts - YouTube, accessed February 3, 2026, https://www.youtube.com/watch?v=gOMHcSt2yQ0
3. [2509.00499] NeuralSVCD for Efficient Swept Volume Collision Detection - arXiv, accessed February 3, 2026, https://arxiv.org/abs/2509.00499
4. Open CASCADE Technology: Modeling Algorithms, accessed February 3, 2026, https://dev.opencascade.org/doc/overview/html/occt_user_guides__modeling_algos.html
5. Foundation Classes - Open CASCADE Technology, accessed February 3, 2026, https://dev.opencascade.org/doc/overview/html/occt_user_guides__foundation_classes.html
6. A literature review of bounding volumes hierarchy focused on collision detection, accessed February 3, 2026, http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0123-303320150001

00005
7. Panel Bender -P4 - MachineryHost, accessed February 3, 2026, https://f.machineryhost.com/71b2fab0316bfa7fa24ce0a0e5460bad/Brochure%20-%20Panel%20Bender%20-P4.pdf
8. Automatic Sheet Metal Bending Machine - Salvagnini P4, accessed February 3, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P4
9. Bending? Easy, with Salvagnini!, accessed February 3, 2026, https://www.salvagninigroup.com/en-INT/campaigns/panel-benders-range
10. The automatic hybrid adaptive Panel Bender. - Sanson Machinery Group, accessed February 3, 2026, http://www.sansonmachinery.com/wp-content/uploads/2013/12/P4Xe-UK-3962020312.pdf
11. Universal Sheet Metal Panel Bending Machine - Salvagnini, accessed February 3, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders
12. BRepPrimAPI_MakeRevol Class Reference - Open CASCADE Technology, accessed February 3, 2026, https://dev.opencascade.org/doc/refman/html/class_b_rep_prim_a_p_i___make_revol.html
13. Fast Continuous Collision Detection for Articulated Models - UNC Computer Science, accessed February 3, 2026, https://www.cs.unc.edu/techreports/03-038.pdf
14. BRepPrimAPI_MakeSweep Class Reference - Open CASCADE Technology, accessed February 3, 2026, https://dev.opencascade.org/doc/refman/html/class_b_rep_prim_a_p_i___make_sweep.html
15. How to sweep an rigid body? - Forum Open Cascade Technology, accessed February 3, 2026, https://dev.opencascade.org/content/how-sweep-rigid-body
16. Robot Cell Modeling via Exploratory Robot Motions - arXiv, accessed February 3, 2026, https://arxiv.org/html/2502.01484v2
17. NeuralSVCD for Efficient Swept Volume Collision Detection - arXiv, accessed February 3, 2026, https://arxiv.org/html/2509.00499v1
18. Axis-Aligned Bounding Box Calculation (AABB) for different orientations of 3D object, accessed February 3, 2026, https://stackoverflow.com/questions/74358413/axis-aligned-bounding-box-calculation-aabb-for-different-orientations-of-3d-ob
19. 3D collision detection - Game development - MDN Web Docs, accessed February 3, 2026, https://developer.mozilla.org/en-US/docs/Games/Techniques/3D_collision_detection
20. BVH_Tree< T, N > Class Template Reference - Open CASCADE Technology, accessed February 3, 2026, https://dev.opencascade.org/doc/occt-6.9.1/refman/html/class_b_v_h___tree.html
21. Example of dynamic BVH building and BVH update? - OptiX - NVIDIA Developer Forums, accessed February 3, 2026, https://forums.developer.nvidia.com/t/example-of-dynamic-bvh-building-and-bv

h-update/202276
22. How to build a BVH – part 9b: Massive - Jacco's Blog, accessed February 3, 2026, https://jacco.ompf2.com/2022/06/15/how-to-build-a-bvh-part-9b-massive/
23. Data Structures - Open CASCADE Technology, accessed February 3, 2026, https://dev.opencascade.org/doc/refman/html/annotated.html
24. Performance Issues with BRepExtrema_DistShapeShape - Open CASCADE Technology, accessed February 3, 2026, https://dev.opencascade.org/content/performance-issues-brepextremadistshapeshape
25. BRepExtrema_DistShapeShape wire intesections and face cut performance?, accessed February 3, 2026, https://dev.opencascade.org/content/brepextremadistshapeshape-wire-intesections-and-face-cut-performance
26. Upgrade from older OCCT versions - Open CASCADE Technology, accessed February 3, 2026, https://dev.opencascade.org/doc/overview/html/occt__upgrade.html
27. BRepExtrema_DistShapeShape Class Reference - Open CASCADE Technology, accessed February 3, 2026, https://old.opencascade.com/doc/occt-7.2.0/refman/html/class_b_rep_extrema__dist_shape_shape.html