Geometric Kernel Architecture for Panel Bender CAM: Bridging the Semantic Gap in Raw B-Rep Topology

1. Introduction: The Semantic Gap in Automated CAM

The development of Computer-Aided Manufacturing (CAM) software for Panel Bending machinery represents one of the most demanding challenges in computational geometry. Unlike traditional press brakes, which operate on a relatively simple linear kinematic model (a punch forcing material into a die), panel benders utilize a complex system of blankholders, counterblades, and interpolated bending blades to manipulate the sheet metal in both positive (up) and negative (down) directions.[1] This mechanical sophistication enables the rapid production of complex geometries, such as boxes, envelopes, and intricate profiles, but it imposes a heavy burden on the driving software: the absolute necessity for a rigorous, semantic understanding of the 3D model.

The input for such systems is typically a raw 3D STEP file (ISO 10303). While STEP is the industry standard for interoperability, it transmits only the Boundary Representation (B-Rep) of the object—a collection of faces, edges, and vertices. It does not explicitly convey "design intent." The file contains TopoDS_Face entities, but it does not label them as "Flanges" or "Bends." It contains TopoDS_Edge entities, but it does not distinguish between a "Bend Line" and a "Cut Edge." Most critically, standard STEP translation often decouples the geometric definition of a surface from the topological orientation of the material, creating the "Orientation Consistency" problem.[3]

For a Panel Bender, misinterpreting the orientation of a face is not merely a rendering error; it is a safety hazard. If the geometric kernel perceives a face's normal pointing inward rather than outward, the collision detection algorithms will invert the definition of "solid" and "air." The machine might attempt to drive a steel bending blade through a region it believes is empty space, which in reality is occupied by the workpiece, leading to catastrophic tool crashes and machine damage.

This report details the architecture of a robust Geometric Kernel built on OpenCASCADE Technology (OCCT) specifically for this application. We will deconstruct the mathematical and topological requirements for converting "dirty" raw B-Rep data into a semantic "Face-Adjacency Graph" (FAG). We will provide a rigorous mathematical derivation and C++ implementation for the critical helper function GetMaterialOutwardNormal(TopoDS_Face), and demonstrate how this vector serves as the fundamental axiom for distinguishing 'Bend Up' from 'Bend Down' operations in the context of machine kinematics.

2. Theoretical Foundations: The OpenCASCADE B-Rep Model

To solve the orientation problem, one must first master the underlying data structure of OpenCASCADE. OCCT utilizes a hierarchy of topological shapes defined in the TopoDS package, which separates the definition of the object's geometry from its topology.

2.1 Geometry vs. Topology

In OCCT, **Geometry** refers to the mathematical description of shapes in Cartesian space. Classes inheriting from Geom_Geometry (such as Geom_Plane, Geom_CylindricalSurface, Geom_Line) are purely mathematical. An infinite plane defined by the equation $Ax + By + Cz + D = 0$ has a specific normal vector, but it has no boundaries and, crucially, no concept of "inside" or "outside."

Topology defines the boundaries and connectivity. Classes inheriting from TopoDS_Shape (such as TopoDS_Face, TopoDS_Edge, TopoDS_Vertex) restrict the infinite geometry to finite regions. A TopoDS_Face uses a TopoDS_Wire to bound a Geom_Surface.

The critical link between geometry and physical reality is the **Orientation Flag** (TopAbs_Orientation). Every topological object carries this flag, which can be FORWARD, REVERSED, INTERNAL, or EXTERNAL.

- **FORWARD:** The topological object traverses the underlying geometry in its natural direction. For a face, this means the topological normal aligns with the geometric normal (intrinsic surface normal).
- **REVERSED:** The topological object traverses the geometry in the opposite direction. For a face, the topological normal opposes the geometric normal.

2.2 The Orientation Consistency Problem defined

The "Orientation Consistency" problem arises because the "Material Side" is defined by convention. In OCCT B-Rep, the material of a solid is located on the **negative** side of the face normal.[4] Therefore, the "Outward Normal" (pointing toward the air/tooling) is the face normal.

However, raw STEP files are notoriously "dirty."

1. **Inverted Solids:** The entire solid may be defined such that all face normals point *into* the material. In this state, the solid effectively has negative volume.[7]
2. **Inconsistent Shells:** In a shell of connected faces, Face A might be oriented FORWARD while its neighbor Face B is oriented REVERSED relative to the same logical surface side. This creates a "Moebius strip" effect where the "up" direction flips arbitrarily across an edge.[9]
3. **Disconnected Topology:** Visually connected faces may share no topological edges due

to translation tolerances, resulting in a compound of disjoint surfaces rather than a coherent solid.[11]

A robust geometric kernel must sanitize this input before any CAM logic can be applied. We cannot trust the raw TopoDS_Face::Orientation() until we have globally validated the solid.

3. Geometric Healing and Sanitation Strategy

Before building the Face-Adjacency Graph, the kernel must perform a multi-stage sanitation pass. This transforms the "dirty" input into a "watertight," valid manifold solid.

3.1 Stage 1: Topological Sewing

Raw STEP files often contain faces that are geometrically coincident at their boundaries but topologically disconnected. If Face A and Face B meet at a bend line, they should share a single TopoDS_Edge object. In dirty geometry, they often reference two distinct edges that happen to have the same coordinates.

We utilize BRepBuilderAPI_Sewing to resolve this.[13]

- **Mechanism:** The sewing algorithm spatializes edges and vertices into a grid. It identifies "free edges" (edges connected to only one face). It then searches for other free edges within a specified tolerance (typically 1.0×10^{-6} meters for precision mechanics).
- **Manifold Constraint:** For sheet metal, we must enforce a manifold constraint. An edge should be shared by exactly two faces. If the sewing algorithm detects a cluster of three or more faces meeting at an edge (non-manifold), this often indicates a T-junction error or a "zero-radius hem" modeled incorrectly.[13]

Insight: While BRepBuilderAPI_Sewing merges edges, it does *not* guarantee consistent orientation. It typically adopts the orientation of the first face added to the sewing cluster. Thus, the resulting shell may still have flipped normals.

3.2 Stage 2: Local Orientation Correction

Once the faces are sewn into a connected TopoDS_Shell, we must ensure local consistency. This means that if we traverse the shell, the "up" vector does not flip as we cross an edge. The ShapeFix_Shell class is the standard tool for this.[8]

- **Function:** ShapeFix_Shell::FixFaceOrientation(shell).
- **Logic:** It iterates through the shell, picking a seed face. It checks adjacent faces across shared edges. If the winding order of the shared edge is consistent (e.g., the edge is FORWARD in Face A and FORWARD in Face B), it implies an orientation conflict (since a shared edge should be traversed in opposite directions by valid neighbors). The algorithm flips the neighbor to match.

Critical Limitation: ShapeFix_Shell ensures that all faces agree with the *seed face*. If the seed face happens to be inverted (pointing into the material), the *entire shell* will be consistently inverted.

3.3 Stage 3: Global Orientation Validation

This is the most crucial step for CAM safety. We must determine if the coherent shell points "out" (positive volume) or "in" (negative volume). Relying on BRepGProp::VolumeProperties to check for negative mass is common but can be computationally expensive and ambiguous for open shells.[7]

A more robust method uses **Infinite Point Classification** via BRepClass3d_SolidClassifier.

1. We define a point at infinity (virtually).
2. We classify this point relative to the solid using BRepClass3d_SolidClassifier::PerformInfinitePoint.[16]
3. **Logic:** A valid, bounded solid must have infinity on the TopAbs_OUT side.
   - If the classifier returns TopAbs_IN: The solid considers "infinity" to be its interior. This defines the universe as the solid and the part as a void. The solid is **Inside-Out**.
4. **Correction:** If TopAbs_IN is detected, we invoke TopoDS_Shape::Reverse(). This negates the orientation flag of the root solid, which cascades down to effectively flip every face in the model, correcting the global orientation.[7]

Only after these three stages—Sewing, Local Fixing, and Global Validation—can we proceed to build the semantic graph.

4. The "Material Outward Normal": Algorithmic Implementation

With a sanitized solid, we can now implement the core helper function. The goal of GetMaterialOutwardNormal is to return a gp_Dir vector that strictly points away from the material, regardless of how the underlying surface is parameterized.

4.1 The Mathematics of Surface Normals in OCCT

Any surface in OCCT is parameterized by (u, v). The geometric normal N_{geom} at a point $P(u, v)$ is derived from the cross product of the first derivatives (tangents):

$$N_{geom}(u, v) = \frac{\partial S}{\partial u} \times \frac{\partial S}{\partial v}$$

This vector is intrinsic to the geometry. For example, for a Geom_CylindricalSurface, the normal always points radially outward from the axis (if the coordinate system is right-handed).

The **Material Outward Normal (N_{out})** depends on the Face Orientation (O_f):

$$N_{out} = \begin{cases} N_{geom} & \text{if } O_f = \mathrm{TopAbs\_FORWARD} \\ -N_{geom} & \text{if } O_f = \mathrm{TopAbs\_REVERSED} \end{cases}$$

4.2 Handling Singularities and Edge Cases

Evaluating the normal at an arbitrary point can be dangerous.

- **Poles/Apexes:** At the pole of a sphere or the apex of a cone, the derivatives may vanish or become collinear, making the cross product zero or undefined.[18]
- **Trimmed Surfaces:** A TopoDS_Face might utilize only a small patch of a larger surface. Evaluating at $(0,0)$ of the underlying plane might check a point kilometers away from the actual part.

Best Practice: We utilize BRepTools::UVBounds to determine the active parametric range $[U_{min}, U_{max}] \times [V_{min}, V_{max}]$ of the face. We then evaluate the normal at the **UV center**:

$$U_{mid} = \frac{U_{min} + U_{max}}{2}, \quad V_{mid} = \frac{V_{min} + V_{max}}{2}$$

This ensures we are sampling a point well inside the topological boundaries, avoiding edge singularities.[19]

4.3 C++ Implementation

The following implementation integrates these concepts. It uses GeomLProp_SLProps (Surface Local Properties) for differential geometry calculations, as it is lower-level and more explicit than BRepGProp.

C++

```
#include <TopoDS.hxx>
#include <TopoDS_Face.hxx>
#include <BRep_Tool.hxx>
#include <BRepTools.hxx>
#include <Geom_Surface.hxx>
#include <GeomLProp_SLProps.hxx>
```

```cpp
#include <gp_Dir.hxx>
#include <gp_Pnt.hxx>
#include <Precision.hxx>
#include <TopAbs.hxx>
#include <Standard_Failure.hxx>

/**
 * @brief Computes the material outward normal for a given topological face.
 *
 * This function resolves the "Orientation Consistency" problem by:
 * 1. Retrieving the underlying parametric surface geometry.
 * 2. calculating the geometric normal at the UV parametric center (to avoid edge singularities).
 * 3. Applying the topological orientation flag (FORWARD/REVERSED) to ensure the
 *    resulting vector points AWAY from the solid material.
 *
 * @param theFace The topological face to analyze.
 * @return gp_Dir The unit vector pointing away from the solid material (Safe Air).
 * @throws Standard_Failure if the surface is null or normal is undefined (singularity).
 */
gp_Dir GetMaterialOutwardNormal(const TopoDS_Face& theFace)
{
    // 1. Retrieve Geometric Surface and UV Bounds from the Topology
    // BRep_Tool::Surface unwraps the topology to get the math geometry handle.
    Standard_Real uMin, uMax, vMin, vMax;
    BRepTools::UVBounds(theFace, uMin, uMax, vMin, vMax);

    Handle(Geom_Surface) aSurface = BRep_Tool::Surface(theFace);

    // Robustness Check: Ensure the face actually has geometry
    if (aSurface.IsNull()) {
        throw Standard_Failure("Critical Error: Face has no underlying surface geometry.");
    }

    // 2. Compute Midpoint Parameters
    // We sample the center of the UV domain to avoid singularities often found
    // at boundaries (e.g., seams of periodic surfaces or poles of cones).
    Standard_Real uMid = (uMin + uMax) * 0.5;
    Standard_Real vMid = (vMin + vMax) * 0.5;

    // 3. Compute Surface Local Properties
    // We request 1st order derivatives (for normal).
    // 1e-7 is the derivation tolerance (linear precision).
    GeomLProp_SLProps aProps(aSurface, uMid, vMid, 1, 1e-7);

    // 4. Check if Normal is defined at this point
    // This handles cases where the UV center might coincide with a singularity
    // (though rare for valid sheet metal faces).
    if (!aProps.IsNormalDefined()) {
        throw Standard_Failure("Singularity Error: Surface normal undefined at UV center.");
    }

    // 5. Get the Intrinsic Geometric Normal
    // This vector (N_geom) is defined purely by the surface derivatives (du x dv).
    // It ignores whether the material is "inside" or "outside".
    gp_Dir aGeoNormal = aProps.Normal();

    // 6. Apply Topological Orientation
    // The TopoDS_Face orientation flag determines the material side.
    // In OCCT, material is on the negative side of the normal.
    // If Orientation == FORWARD: Face Normal == Surface Normal.
    // If Orientation == REVERSED: Face Normal == -Surface Normal.
    // Therefore, the vector pointing OUT of the material is the aligned face normal.

    TopAbs_Orientation anOrient = theFace.Orientation();

    if (anOrient == TopAbs_REVERSED) {
        aGeoNormal.Reverse();
    }

    return aGeoNormal;
}
```

5. Building the Semantic Face-Adjacency Graph (FAG)

With the ability to compute reliable normals, we construct the **Face-Adjacency Graph (FAG)**. This data structure translates the B-Rep "soup" into a structured map of Flanges and Bends, which is the input required for toolpath generation.

5.1 Graph Definition

The FAG is defined as a graph $G = (V, E)$ where:

- **Nodes (V):** Represent TopoDS_Face entities. Each node carries semantic attributes:
    - **Type:** Flange (Planar) or Bend (Cylindrical/Conical).
    - **Normal:** The Material Outward Normal (N_{out}).
    - **Geometry:** Radius (for bends), Area, Centroid.

- **Edges (E):** Represent the connectivity between faces (Physical Folds). Attributes include:
    - **Bend Line:** The TopoDS_Edge geometry.
    - **Bend Angle:** The signed angle of the fold.
    - **Convexity:** Convex (Mountain) or Concave (Valley).

5.2 Mapping Shapes to Ancestors

The most efficient algorithm to build this graph in OCCT is TopExp::MapShapesAndAncestors.[21] A naive approach of iterating every face pair to check for shared edges is $O(N^2)$. MapShapesAndAncestors creates an indexed map in near-linear time.

Algorithm:

1. Map TopAbs_EDGE $\rightarrow$ List of TopAbs_FACE Ancestors.
2. Iterate through the map. For each edge:
    - **List Size = 1:** The edge is a **Boundary Edge** (outer perimeter of the part).
    - **List Size = 2:** The edge is a **Connection Edge** (Bend Line). It connects the two faces in the list. Create a Graph Edge between these two Face Nodes.
    - **List Size > 2:** The edge is **Non-Manifold**. In sheet metal, this usually indicates a T-junction error (e.g., an internal stiffener modeled as zero-thickness). The kernel should flag this as a "Complex Feature" requiring manual intervention or specialized handling.[13]

5.3 Semantic Classification: Flange vs. Bend

We iterate through the Face Nodes and classify them using BRepAdaptor_Surface:

- **Flange:** surface.GetType() == GeomAbs_Plane.
- **Simple Bend:** surface.GetType() == GeomAbs_Cylinder.
- **Lofted/Conical Bend:** surface.GetType() == GeomAbs_Cone.
- **Complex Form:** Other types (BSpline, Torus). These are flagged as non-standard forming features (e.g., louvers, embossments).[25]

For Cylindrical Bends, we extract the **Bend Radius** directly from the underlying geometry (cylinder.Radius()). This value is critical for selecting the correct tool radius in the panel bender.[26]

6. Kinematic Logic: 'Bend Up' vs. 'Bend Down'

This section addresses the core user query: explaining *why* GetMaterialOutwardNormal is critical for the 'Bend Up' vs 'Bend Down' decision.

6.1 Panel Bender Kinematics

Unlike a press brake where the operator flips the part, a panel bender keeps the sheet horizontal and clamped. The machine has two independent bending blades:

- **Upper Blade:** Moves downward to perform a **Negative Bend** (Bend Down).
- **Lower Blade:** Moves upward to perform a **Positive Bend** (Bend Up).[1]

The CAM software must translate the geometric "fold" in the CAD model into one of these two machine operations. This translation depends entirely on the **Convexity** of the bend relative to the machine's reference plane.

6.2 The Vector Calculus of Bend Direction

To determine the direction, we analyze the transition from a **Reference Face** (the face currently held by the blankholder) to the **Target Face** (the flange being bent).

Let:

- N_{ref} be the Outward Normal of the Reference Face (pointing to $+Z$ in machine coordinates).
- N_{target} be the Outward Normal of the Target Face.
- E_{axis} be the direction vector of the bend axis.

We calculate the **Signed Angle** or **Dihedral Angle**. A robust method involves the cross product:

$$V_{cross} = N_{ref} \times N_{target}$$

The alignment of V_{cross} with the bend axis E_{axis} determines the convexity.

- **Convex Fold ("Mountain"):** The material wraps *around* the bend axis. The angle between normals is $>$ (measured internally) or the normals diverge.
  - In a standard Panel Bender setup (sheet flat on table), a Convex fold relative to the top face means the flange goes **DOWN** (Negative Bend).
- **Concave Fold ("Valley"):** The material folds *into* itself. The normals converge.
  - A Concave fold relative to the top face means the flange goes **UP** (Positive Bend).

Table 1: Geometric to Kinematic Mapping

Feature Geometry (Relative to Top Face)	**Dihedral Angle (Air Side)**	**Kinematic Operation**	**Blade Used**
Convex (Mountain)	$>$	**Bend Down (Negative)**	Upper Blade
Concave (Valley)	$<$	**Bend Up (Positive)**	Lower Blade

6.3 The Criticality of Orientation Consistency

This mapping reveals why the "Dirty Geometry" problem is fatal.

If GetMaterialOutwardNormal is not robust—if it returns the geometric normal without checking the orientation flag—the kernel might calculate N_{target} pointing *inward*.

- This creates a "Ghost Normal" that opposes reality.
- The Convex/Concave calculation flips.
- A physical "Bend Down" feature is mathematically interpreted as "Bend Up."
- **Result:** The machine commands the Lower Blade to move Up. The blade strikes the sheet from below, attempting to fold it *through* the clamp, causing a collision that can cost tens of thousands of dollars in damage.[29]

By enforcing the GetMaterialOutwardNormal check, we guarantee that the normals always represent the "Air Side," ensuring the Convex/Concave logic maps correctly to the "Safe/Collision" zones of the machine.

7. Parameter Extraction: K-Factor and Unfolding

While the FAG defines the topology, accurate manufacturing requires unfolding the 3D model into a 2D flat pattern. This requires accounting for material deformation, represented by the **K-Factor**.

7.1 STEP AP242 and Metadata

Modern STEP files (AP242) can embed Product Manufacturing Information (PMI), including sheet thickness and sometimes bend tables. However, relying on this is risky as many exporters strip this data. The kernel should attempt to read XDE (Extended Data Exchange) attributes but fallback to geometric extraction.[32]

7.2 Geometric Calculation of Neutral Axis

If the K-Factor is not provided, the kernel uses the standard Bend Allowance formulas derived from the FAG geometry.[34]

The **Bend Allowance (BA)** (length of the neutral axis arc) is calculated as:

$$BA = \frac{\pi \cdot A}{180}(R + K \cdot T)$$

Where:

- A: Bend Angle (degrees, extracted from FAG).
- R: Inside Radius (extracted from Cylindrical Surface geometry in FAG).
- T: Material Thickness.
- K: K-Factor (Material constant, e.g., 0.33 for Air Bending, 0.5 for Coining).

Thickness Extraction: The kernel calculates T by ray-casting from a face center along the *inward* normal ($-N_{out}$) to find the opposite face. This distance is the local thickness. If thickness varies across the part, the part is not valid sheet metal.[34]

7.3 Bend Deduction

For the machine operator, **Bend Deduction (BD)** is often more useful. It represents the material "lost" to the bend stretch.

$$BD = 2 \cdot OSSB - BA$$

Where $OSSB$ (Outside Setback) is:

$$OSSB = \tan\left(\frac{A}{2}\right)(R + T)$$

The kernel computes these values for every Bend Node in the FAG, outputting a "Cut List" for the laser cutter and a "Bend Sequence" for the panel bender.[36]

8. Advanced Edge Cases and Robustness

A production-grade kernel must handle cases that deviate from the ideal "Plane-Cylinder-Plane" topology.

8.1 Zero-Radius Bends (Hems)

Some designers model hems or sharp folds without a fillet radius, creating a sharp edge where two planes meet.

- **Detection:** The FAG edge connection has no intermediate "Cylinder Node."
- **Handling:** The kernel detects adjacent Planar Nodes with non-parallel normals. It inserts a "Virtual Bend Node" into the graph.
- **Kinematics:** This instructs the machine to perform a "Hemming" or "Flattening" cycle rather than a standard radial bend.[28]

8.2 Lofted Bends (Conical Surfaces)

A "Bump Bending" or "Step Bending" process is required for conical faces. The panel bender approximates this by a series of small hits.

- **Detection:** Surface type is GeomAbs_Cone or GeomAbs_BSplineSurface.
- **Processing:** The kernel discretizes the conical face into a series of strip planes and cylindrical bends in the FAG, generating a macro program for the machine.[13]

8.3 Handling Non-Manifold Errors

If the sanitation pass (Section 3) fails to produce a clean solid, the kernel must degrade gracefully.

- **Strategy:** Identify the largest connected component (Shell). Treat smaller disconnected shells as "floating artifacts" (noise) and warn the user.
- **Gap Closing:** If two edges are close (within 10^{-3} mm) but not sewn, the kernel can infer a "Virtual Connection" in the FAG, assuming the gap is a modeling error rather than a physical slit.

9. Conclusion

The transition from a raw STEP file to a functional Panel Bender program is a process of imposing semantic order on ambiguous topological data. The "Orientation Consistency" problem is the central hurdle in this workflow. By implementing a multi-stage geometric kernel that sanitizes input via Sewing and ShapeFix, and by rigorously enforcing the Material Outward Normal using the GetMaterialOutwardNormal algorithm, we establish a reliable "Truth" for the model.

This "Truth"—that every normal vector points to safe air—allows us to construct a high-level Face-Adjacency Graph. Within this graph, the mathematical abstraction of the Dihedral Angle (cross product of normals) translates directly and safely into the physical kinematics of the machine (Bend Up vs. Bend Down). This architecture ensures that the CAM software acts as a fail-safe bridge between the virtual design and the physical reality of sheet metal manufacturing, preventing costly errors and enabling true automation in the Industry 4.0 landscape.

The integration of specific K-factor calculations and the handling of advanced features like hems and lofted bends further elevates the kernel from a simple viewer to a manufacturing-grade engine. By rigorously validating geometry against topology using OpenCASCADE's powerful tools (BRepClass3d, GeomLProp, BRepAdaptor), developers can build a system that is robust against the "dirty" reality of industrial CAD data.

Works cited

1. Automatic Sheet Metal Bending Machine - Salvagnini P4, accessed February 2, 2026, https://www.salvagninigroup.com/en-US/products/panel-benders/P4
2. Salvagnini panel bending: P2 panel bender has almost no limits - YouTube, accessed February 2, 2026, https://www.youtube.com/watch?v=utXQHC9yT9k
3. TopoDS_Face Class Reference - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/refman/html/class_topo_d_s___face.html
4. Topology and Geometry in Open CASCADE. Part 4, accessed February 2, 2026, https://opencascade.blogspot.com/2009/02/continued.html
5. Modeling Data - Open CASCADE Technology Documentation, accessed February 2, 2026, https://dev.opencascade.org/doc/occt-6.7.0/overview/html/user_guides__modeling_data.html
6. TopoDS_Face with REVERSED orientation - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/topodsface-reversed-orientation
7. BRepGProp returns negative volume - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/brepgprop-returns-negative-volume
8. Problem with surface normal - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/problem-surface-normal-0
9. Normals to Outside - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/normals-outside
10. Correct face normal calculation - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/correct-face-normal-calculation
11. Creating a TopoDS_Face from non-planar wires or edges - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/content/creating-topodsface-non-planar-wires-or-edges
12. BRepBuilderAPI_Sewing and Standard_TypeMismatch - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/brepbuilderapisewing-and-standardtypemi

smatch
13. Modeling Algorithms - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/overview/html/occt_user_guides__modeling_algos.html
14. ShapeFix_Shell Class Reference - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/refman/html/class_shape_fix___shell.html
15. ShapeFix_Shell Class Reference - Open CASCADE Technology, accessed February 2, 2026, https://old.opencascade.com/doc/occt-7.5.0/refman/html/class_shape_fix___shell.html
16. BRepClass3d_SolidClassifier Class Reference - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/refman/html/class_b_rep_class3d___solid_classifier.html
17. [BUG] BRepClass3d_SolidClassifier::PerformInfinitePoint fails - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/bug-brepclass3dsolidclassifierperforminfinitepoint-fails
18. ShapeAnalysis_Surface Class Reference - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/refman/html/class_shape_analysis___surface.html
19. normal to topoDS_Face - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/normal-topodsface
20. How to compute surface normal in OpenCASCADE - TechOverflow, accessed February 2, 2026, https://techoverflow.net/2019/06/26/how-to-compute-surface-normal-in-opencascade/
21. TopExp Class Reference - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/refman/html/class_top_exp.html
22. Problem with MapShapesAndAncestors - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/problem-mapshapesandancestors
23. Finding connected faces of a CAD model - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/finding-connected-faces-cad-model
24. BRepCheck_Analyzer Class Reference - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/refman/html/class_b_rep_check___analyzer.html
25. Creating lines, arcs, curves from Edges - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/creating-lines-arcs-curves-edges
26. BRepAdaptor_Curve Class Reference - Open CASCADE Technology, accessed February 2, 2026,

https://dev.opencascade.org/doc/refman/html/class_b_rep_adaptor___curve.html
27. Getting radius of circle - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/getting-radius-circle
28. The flexible bending solution. - Sp-Tech, s.r.o., accessed February 2, 2026, https://www.sp-tech.cz/static/soubory/stranka-71/prospekt-salvagnini-p2-152.pdf
29. Calculating the attribute of a edge - Forum Open Cascade Technology, accessed February 2, 2026, https://dev.opencascade.org/content/calculating-attribute-edge
30. Finding/Calculating the bend "direction" of a Sheet metal CAD model - Stack Overflow, accessed February 2, 2026, https://stackoverflow.com/questions/60644946/finding-calculating-the-bend-direction-of-a-sheet-metal-cad-model
31. Explain signed angle calculation between 2 face normals [closed] - Math Stack Exchange, accessed February 2, 2026, https://math.stackexchange.com/questions/4687302/explain-signed-angle-calculation-between-2-face-normals
32. Sheet Metal Operations Component - Open Cascade, accessed February 2, 2026, https://www.opencascade.com/components/sheet-metal-operations/
33. STEP Translator - Open CASCADE Technology, accessed February 2, 2026, https://dev.opencascade.org/doc/overview/html/occt_user_guides__step.html
34. Bend Deduction Calculation: A Step-by-Step Guide - ADH Machine Tool, accessed February 2, 2026, https://www.adhmt.com/bend-deduction-calculation/?fbclid=iwzxh0bgnhzw0cmteaar3m1ehzt5ur2bwtkmtqibexxmbrlg9epnunuy81ksa7mllqss7dyrxnz7m_aem_5q650v2afusqwtcq28j8ia
35. Calculating Bend Allowance & Bend Deduction (+ Formulas) - SendCutSend, accessed February 2, 2026, https://sendcutsend.com/blog/guide-to-calculating-bend-allowance-and-bend-deduction/
36. Sheet metal bending calculation basics - The Fabricator, accessed February 2, 2026, https://www.thefabricator.com/thefabricator/article/bending/sheet-metal-bending-calculation-basics
37. Top 5 Sheet Metal Design Formulas for Bending (2026), accessed February 2, 2026, https://www.approvedsheetmetal.com/blog/5-sheet-metal-design-formulas
38. Guide to CNC Bending & Panel Bending - Custom Metal Components & Assemblies - High Point, North Carolina, accessed February 2, 2026, https://www.metalworkshp.com/guide-cnc-panel-bending/