CP122682

The Inventor 7 Deadly Sins of 3D Part Modelling

Paul Munford Graitec UK

Learning Objectives

- Discover bad modeling practices and how you can avoid and overcome them.
- Learn how to diagnose and fix unexpected failures.
- Learn how to create successful, stable, parametric 3D models.

  Understand how Inventor interprets your requests to build shapes.

Description

Have you ever been in a situation where your Inventor software model failed—and you didn't know why? (I know you have, I have too).

Did you know that your Inventor 3D model's complex shape geometry is calculated by ASM—the Autodesk Shape Manager kernel?

This class is aimed at intermediate and advanced Inventor users who want to increase their knowledge of Inventor software's part modeling tools by gaining a deeper understanding of how Inventor thinks.

We'll cover tips and tricks for the 3D modeling of complex shapes in Inventor. We'll explain how to avoid and overcome modeling practices that give you unexpected results. We'll learn how to diagnose unexpected failures in your Inventor model and discuss ways to fix them.

We hope that you'll come away from this class with a clearer idea of how to build stable, powerful models with Inventor. Come and learn how to improve your Inventor modeling skills and become more valuable to your company.

Speaker

Paul Munford is an Application Engineer for Graitec UK. Until recently Paul was a specialist joinery draughtsman (a "setter out") and CAD/CAM manager for a U.K. based custom furniture contractor.

Paul had 8 years of experience "on the tools" before joining the CAD department in 2005. As an Application Engineer, Paul handles licensing, deployment, and training for AutoCAD and Inventor software. Paul also uses AutoCAD and Inventor to create manufacturing "workshop" drawings for Graitec's customers.

In his spare time, Paul writes the blog entitled CAD Setter Out, and he also authored Mastering Autodesk Inventor 2016. This will be Paul's 9th trip to Autodesk University, and his 6th as a speaker.

@Cadsetterout

About the Author

Jake Fowler is a Sr. User Experience Designer at Autodesk®. He now works in Shanghai, China, but for six years, Jake worked with the Autodesk® ShapeManager (ASM) development team in Cambridge, UK, with responsibility for both customer support and software quality assurance of Inventor®'s modeling kernel.

Jake is one of Autodesk®'s leading internal experts on Inventor® part modeling, and has been responsible for training Autodesk® employees, resellers and customers on the technical aspects of Inventor®'s modeling functionality.

Jake originally has a background in consumer product design, and holds a Master of Engineering degree in Innovation & Engineering Design from the University of Bath, UK.

jake.fowler@autodesk.com

Introduction and terminology

This course is tailored for intermediate & advanced Inventor® users aiming to deepen their knowledge of Inventor®s part modeling toolset, and to become better and more effective 3D modelers.

This handout is designed as a self-contained reference manual that covers the full set of material from the presentation. However, the presentation aims to explain these concepts a lot more coherently and completely than can be done with text and diagrams alone, so its recommended to attend/watch the presentation first, then use this handout as a reference, to get the deepest understanding of the material.

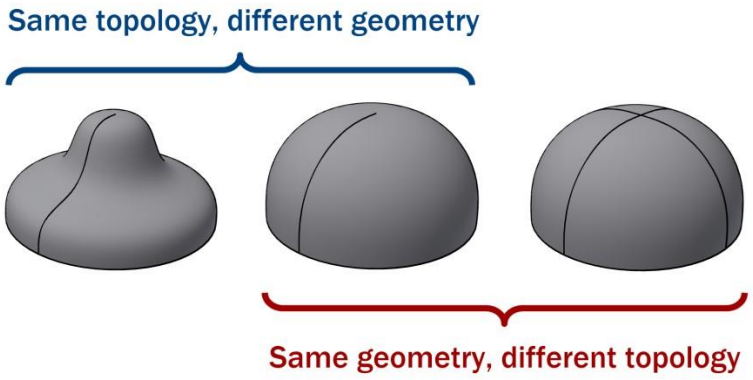
I've tried my best to describe concepts in the simplest possible terms, and steered clear of using jargon when avoidable, but since this is quite an advanced topic it's difficult to completely avoid describing some concepts using technical terminology. Therefore, its preferable that you are familiar with the following definitions before starting the course.

Geometry and topology

The distinction between the terms "geometry" and "topology" is often not well understood; probably because "geometry" is frequently used as a general term to describe any model data. However, on a technical level, there is a clear and important distinction between the two.

- Geometry refers to the mathematical formulae that describe the shapes of surfaces and curves (i.e. numbers and functions in terms of x, y and z, etc.)
- Topology refers to how many entities a model has, their types, and how they fit together (i.e. faces, edges, vertices, and the relationships between them, etc.)

For each topological entity (such as a face, edge or vertex), there needs to be a piece of geometry (e.g. surface, curve or position) that describes its actual shape in 3D space. It's possible for two models to have the same topology but different geometry, and it's also possible (but a lot rarer) for two models to have the same geometry but different topology.



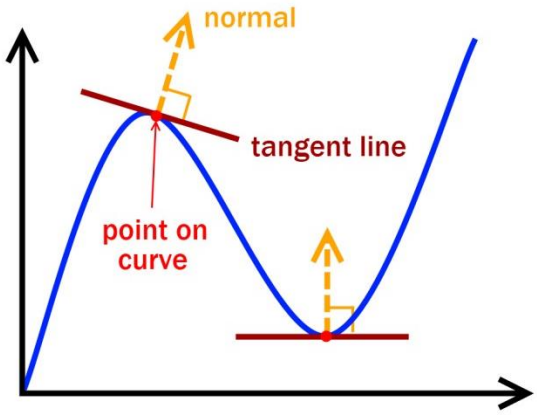
The left-hand model has the same topology as the central model (one face, two edges), but you can clearly see that it has different face and edge geometry; the right-hand model has the same geometry as the central model (same overall shape), but the right-hand model has more topological elements (i.e. More edges, and more vertices where the edges meet)

Surface normals

Any point on a surface has a normal vector (commonly shortened to “normal”): in its simplest terms, this can be thought of as a line pointing directly outwards from the surface at that point.

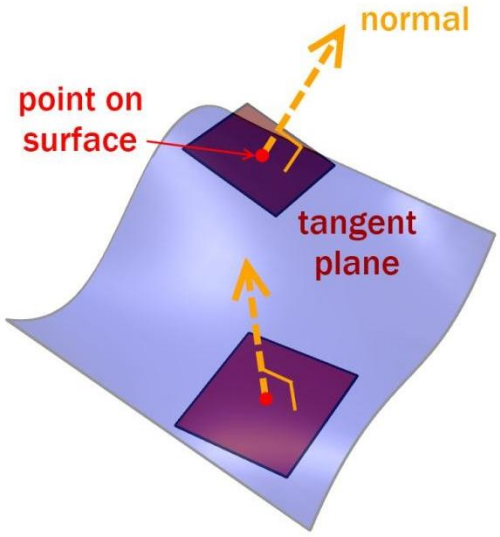
To describe it a bit more accurately: the normal at a point on a surface is the line perpendicular to the tangent plane of the surface at that point.

It’s probably easiest to first think about this concept in two dimensions. At any point on a curve, you can draw a straight line that passes through this point and lies “flat” on the curve: this is the tangent line of the curve at this point. Drawing a line perpendicular to this tangent line will give you the normal for the curve at this point.



The tangent line of a point on a curve is the line lying “flat” to the curve at that point; a vector drawn perpendicular (i.e. at 90 degrees) to the tangent line is the curve normal at that point

Taking this same idea into three dimensions: at any point on a surface, there is a plane that lies “flat” with the surface: this is the tangent plane of the surface at this point, and the line perpendicular to this plane is what we refer to as the normal of the surface at this point.



Considering the previous 2D example in 3D: any point on a surface has a tangent plane, and a vector drawn perpendicular to that plane is the surface normal at that point

(In both the 2D and 3D examples, the normal vector could be pointing “downwards” instead of “upwards”: either of these would be a valid normal, but for a solid model, the rule is that the normal always points “outwards” from the solid volume).

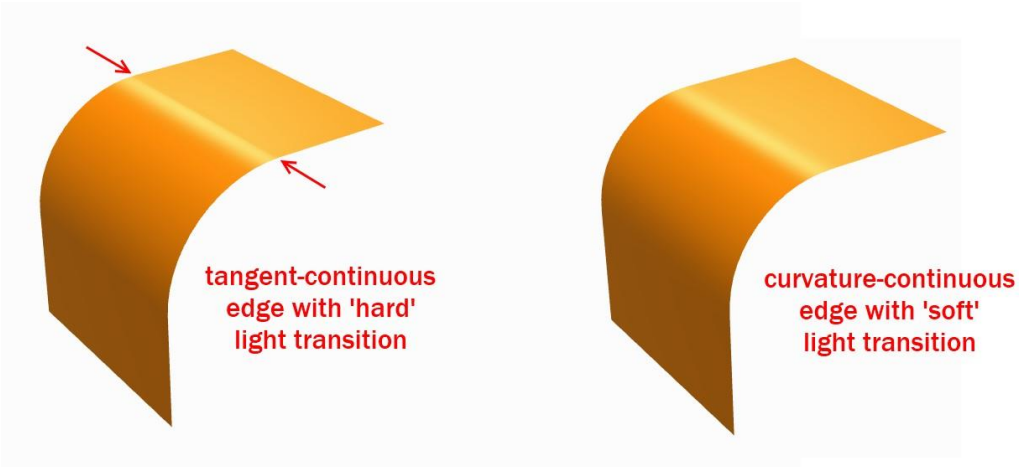
Why is this concept important? As you will see, surface normals are frequently used by Inventor® to decide how to build new geometry, so having a vague idea of which directions the normals along a surface point will often be useful for understanding why Inventor makes a certain shape for a particular operation.

Tangent- and curvature-continuity

Tangent- and curvature-continuity (sometimes referred to as G1- and G2-continuity respectively) describe the smoothness of a join between two faces or two edges (most commonly used in reference to two faces).

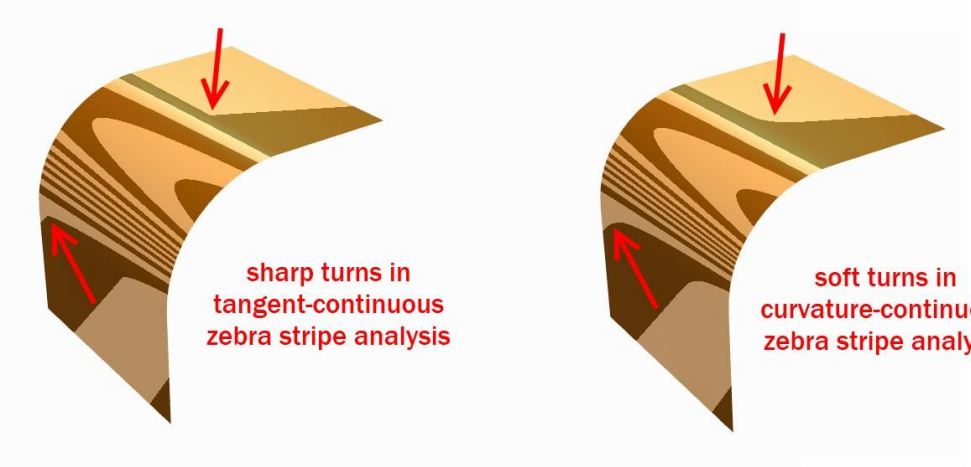
Two faces are tangent continuous if their normals are identical along the adjoining edge. In practical terms, this basically describes the condition where one face runs smoothly into another.

Two faces are curvature-continuous if their curvatures are identical along the adjoining edge. This basically ensures an “extra level” of smoothness. Typically, if a join is tangent-continuous but not curvature-continuous, light reflecting off that part of the model will change direction sharply at the join, giving the appearance of an unsmooth join. If a join is curvature continuous, reflected light should flow smoothly at the join.



An edge that is tangent-continuous but not curvature-continuous is technically smooth, but may have the appearance of a non-smooth edge when the reflected light is observed; a curvature-continuous edge, on the other hand, will have a visibly smoother appearance

Tangent- and curvature-continuity are typically evaluated using the zebra stripe analysis tool. Zebra stripes mimic the effect of light reflecting off the model. If a join is tangent-continuous, there should be no break in the zebra stripes along the join, but the stripes may bend sharply. If a join is curvature continuous, there should be no breaks *and* no sharp bends in the zebra stripes along the join.



Zebra stripe analysis is often used to evaluate surface curvature: any sharp turns in the stripes indicate a join that is tangent-continuous but not curvature-continuous

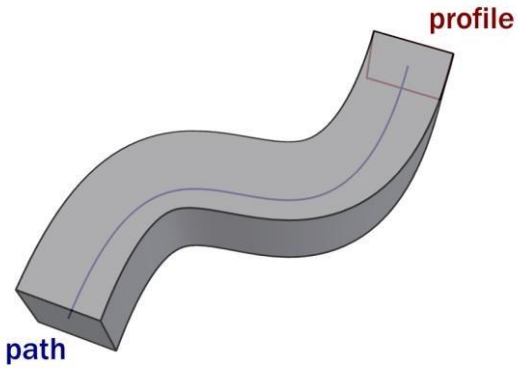
When modeling in Inventor®, generally tangent-continuous joins between faces are sufficient where aesthetics are not a concern (e.g. engineering fillets), but for any joins that need to appear pleasingly smooth to the eye (e.g. joins between shape-defining loft surfaces), curvature-continuity should be used.

Modelling tools under-the-hood

This section offers a behind-the-scenes look at how Inventor®s part modeling tools work, and answers some frequently asked questions about the functionality.

Sweep

The Sweep tool is designed to sweeps a single profile along a defined path.

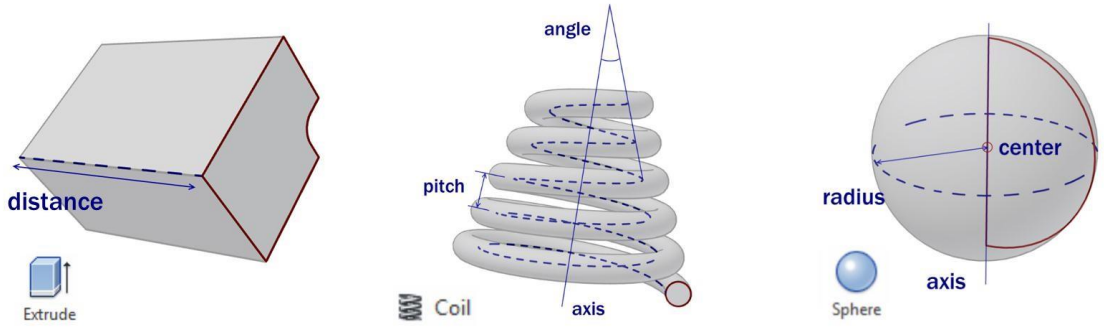


A simple Sweep operation, showing how the profile is guided along the length of the path to create the resultant body

Many Inventor® shape creation tools actually use the same underlying algorithm as Sweep for creating model geometry, so features created using any of these tools can also be considered as “sweeps”:

- Extrude
- Revolve
- Coil
- Primitives (Box, Cylinder, Sphere, Torus)

The primary difference between these tools and the regular Sweep command is the parameters supplied by the user to define the sweep path. While the Sweep tool requires that the user explicitly provides a path curve, these other tools use the parameters provided by the user to automatically ‘build’ a path for the swept body to follow.



Extrude, Coil and Sphere are three examples of other modeling operations that use Sweep technology to create their geometry: they build their sweep path internally using parameters provided by the user

A key property of these “sweep tools” is that they **accurately maintain a single cross-sectional shape** throughout the generated body (some tools allow you to twist, scale or taper the cross-sectional shape as it sweeps along the path, but it will still be based on a single defined profile). So, **for shapes where the design intent is for a controlled cross-sectional shape, one of the “sweep tools” should be used**. Other shape creation tools, such as Loft, do not make the same guarantees about cross-sectional consistency.

Is it possible to Sweep between multiple profiles?

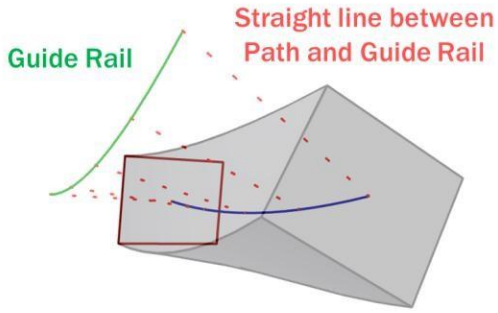
Swept bodies are a function of only one profile. Once a shape needs to be defined using multiple cross-section profiles, it should be created using Loft functionality.

The **Centerline Loft** tool is a hybrid of the Loft and Sweep algorithms: the centerline guides the shape in the same way as the path of a sweep, and you can supply any number of cross-sectional profiles to transition between. So, Centerline Loft can be thought of as a tool for sweeping between multiple profiles.

How does a Guide Rail or Guide Surface control the shape of a Sweep?

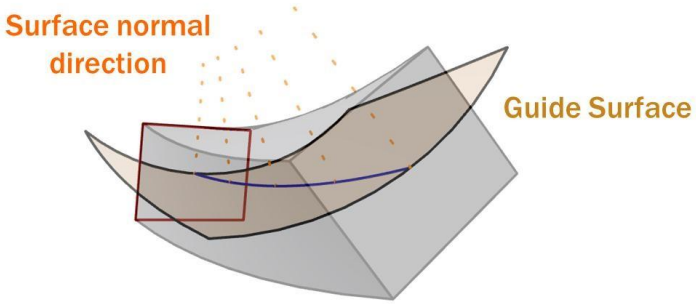
A Guide Rail or Guide Surface will act to **control the orientation (or "twist") of the profile** as it sweeps along the path.

When sweeping with a **guide rail**, this "twist control" can be pictured by imagining **a straight line drawn directly from the path curve to the guide rail**. As the path and guide rail move with respect to one another, this imaginary line will rotate, and this is the rotation that will guide the orientation of the profile.



The position of the guide rail with respect to the sweep path determines how the profile will twist along the length of the sweep

A **guide surface** will control the swept body in a similar way, but in this case the twisting will be controlled by the **surface normal of the guide surface**.

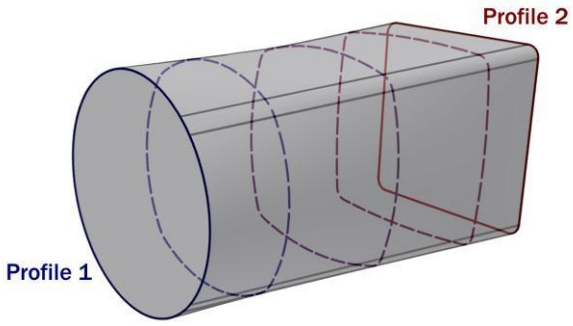


The normal direction of the guide surface is used to control the sweep twist when this is supplied by the user

If neither a guide rail nor guide surface is supplied, the Sweep algorithm to designed to **minimize the twist** of the shape as it travels along the path.

Loft

Loft is designed to create a shape that transitions smoothly between two or more defined profiles.



A simple two-profile loft: the cross-sectional shape of the loft surface transitions smoothly between the two defined profile shapes

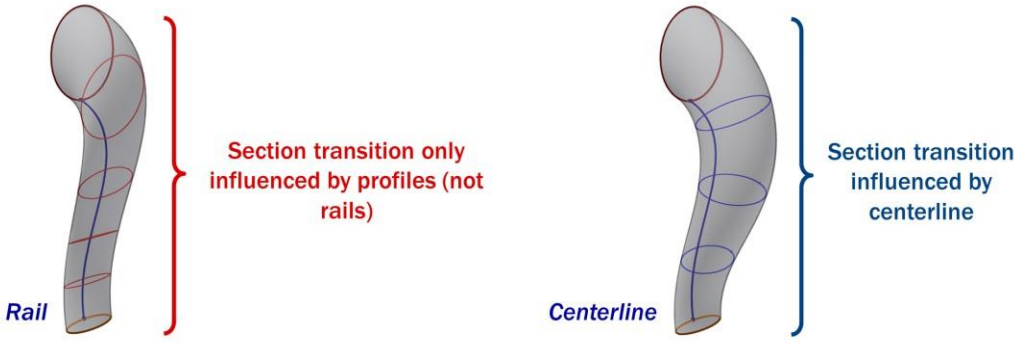
Given the wide range of options for shape control provided by the Loft tool, **Loft is also a very versatile surfacing tool that can be used for surface modeling workflows.**

What is the difference between using Rails and using a Centerline?

Rails are designed to **constrain the shape of a loft along its length**. The precise influence exerted by each rail depends on a variety of factors (number & position of rails, whether profile is open or closed, etc.), but rails can be thought of as lengthways profiles (if you turn your loft inputs by 90° - i.e. swap your profiles for rails and vice versa - the Loft result should be similar, assuming your profiles can be used as valid rails)

Rails will not influence how the loft defines its cross-sections; these will just be defined as a smooth transition between the profiles supplied by the user (as with an unconstrained loft).

As previously mentioned, a centerline will guide the loft shape in the same way as the path does for a sweep. What this means is that **the centerline will override the natural orientation that the loft uses to define cross-sections**, and cross-sections will be aligned **normally to the centerline**.

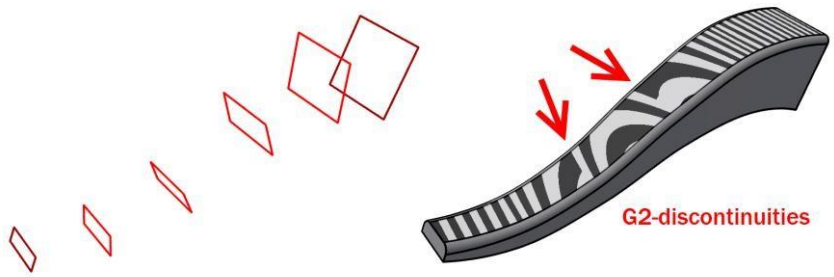


These two lofts are built from the same set of curves, but the rail from the left example was used as a centerline for the right example: this shows how a centerline will guide the cross-section transition along
the sweep, while a rail will just constrain the loft shape

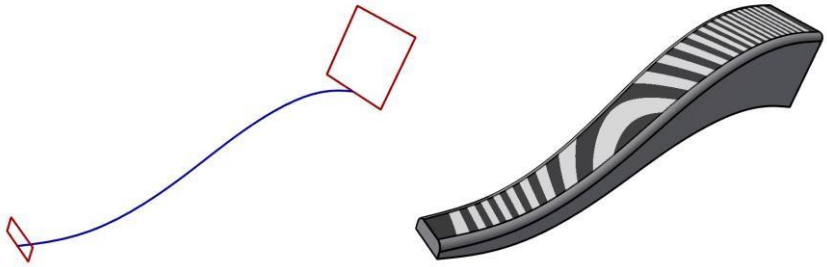
The result of this is that a centerline will tend to have a more “global” influence over the shape of the loft than a single rail. If you are providing only one influence curve, supplying it as a centerline rather than a rail tends to result in a more natural-looking loft shape, with a more uniform curvature flow.

How do I improve the quality of my loft?

An important rule of thumb for Loft (which applies to any surfacing workflows) is to **try to use the minimum number of curves possible to define your shape**. It is not uncommon to see loft inputs that consist of a long series of profiles constraining the loft at regular intervals along its length. This workflow is liable to result in surface quality issues, because the loft may need to sacrifice internal surface continuity to ensure that it passes through all of the user’s specified profiles precisely.



Controlling loft shape with a chain of profiles

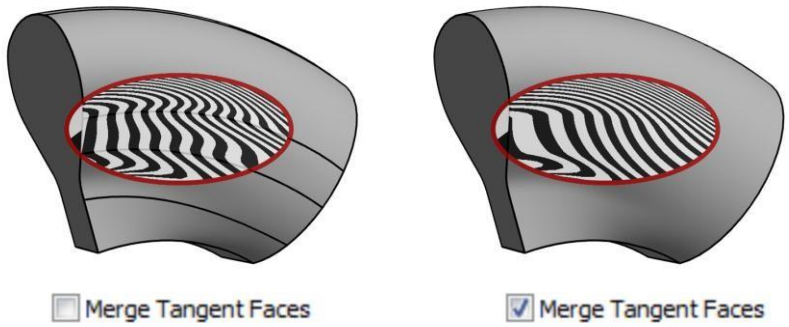


Controlling loft shape with a single rail

A comparison of two similar lofts: one built just using a series of profiles, and one build with two profiles and a single rail; with many profiles, the loft sacrifices smooth curvature to accurately pass through all defined sections, whereas a loft with minimal profiles can flow more naturally

It is also **not recommended to build fillets into your lofts** using profile sketch fillets. Although the loft will adhere to your defined shape as it passes through each profile, no guarantees are made about cross-sectional shape of the loft between defined profiles.

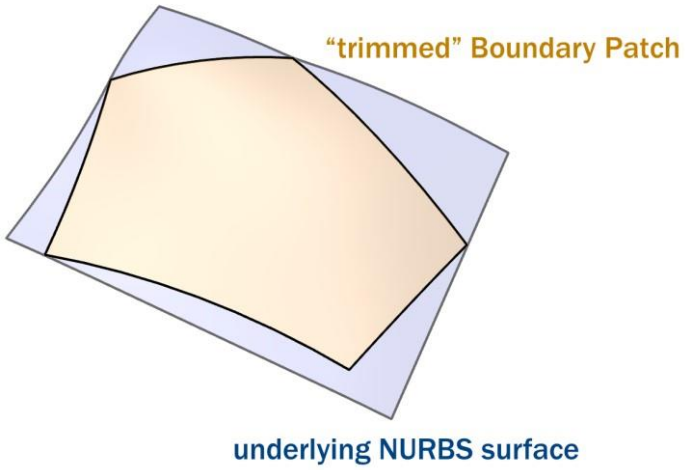
Another tip is to try checking the **Merge Tangent Faces** option in the Loft dialog. If your loft has any mergeable edges, this will reduce the topology created by your loft operation, and may have a positive effect on surface curvature and subsequent offset robustness.



The Merge Tangent Faces option will not only merge-out tangent edges running along the length of the loft, but in some cases, may also improve surface quality

Patch

The Boundary Patch tool will create a NURBS surface that fits to a set of defined input curves. The resulting surface will always be a single four-sided surface that has been trimmed to the defined edges.



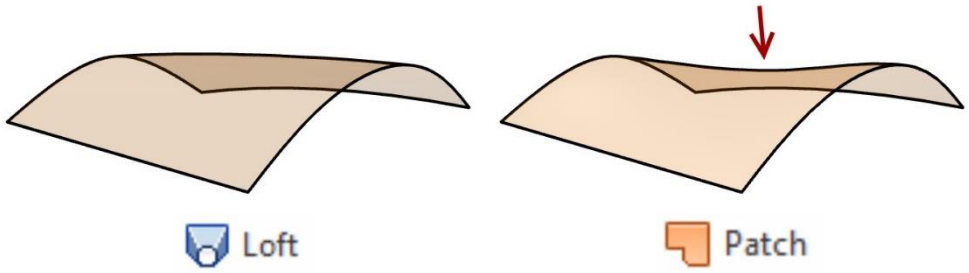
An edge chain with any number of sides can be supplied for a Boundary Patch operation; a four-sided NURBS surface will be fitted to the edges, and then trimmed to return a face with the correct edge geometry

When should I use Patch instead of a Loft Surface?

Patch is usually the preferred surfacing method when surfacing a **non-four-sided edge chain**. Because the Boundary Patch tool will make a trimmed four-sided surface, it can fit a high-quality surface to a curve chain with any number of edges, including those that would be difficult or impossible to surface neatly using Loft (e.g. using Loft to surface a five-sided edge chain results in two separate faces; using Loft to surface a three-sided edge chain results in a surface with a singularity).

When should I use a Loft Surface instead of Patch?

When **creating four-sided faces**, Loft will usually give higher-quality and more natural-looking surfaces than Patch. Loft also offers more options for controlling the shape of a surface (for example, internal rail curves, take-off directions, and tangent-/curvature-continuous constraint weights).

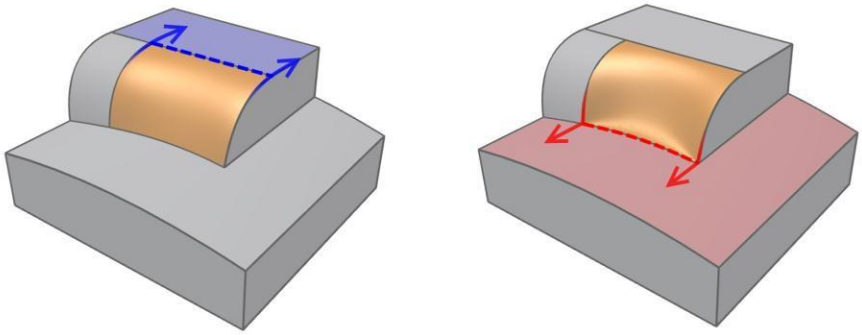


For surfacing four-sided edge chains, Loft usually produces preferable results to Patch (which in this case has created an unpleasant „dip" in the resultant surface)

Loft should also be the preferred tool for surfacing **between distant edge chains**. Patch may give a result in these circumstances, but since Patch is limited to creating a single four-sided NURBS surface, the Patch tool is likely to stretch its underlying surface into a very awkward shape, which would almost certainly encounter robustness issues when worked-with downstream.

Why does my Patch have an awkward shape after applying tangency?

When applying a tangent-continuous constraint to a Boundary Patch edge, **ensure that the adjacent model edges agree with the constraint you are applying**. This means that the edges either side of the edge at which you are applying tangency must be tangent-continuous with the face you are applying tangency to. If there is not agreement here, there will be conflicting requirements at the corner(s) of the Boundary Patch, where your new tangent edge meets adjacent edge(s) (i.e. the tangent constraint is pulling the surface in one direction, but the adjacent edge geometry is pulling the surface in a different direction).



The top edge of this patch should have a tangent condition applied, because the two adjacent edges both flow tangent-continuous into the top (blue) face; this is not so for the bottom edge, because the adjacent edges arrive at the bottom (red) face at a sharp angle

If the adjacent edges are not tangent to a face, you should leave a Free condition for the Boundary Patch at this edge for optimum surface quality.

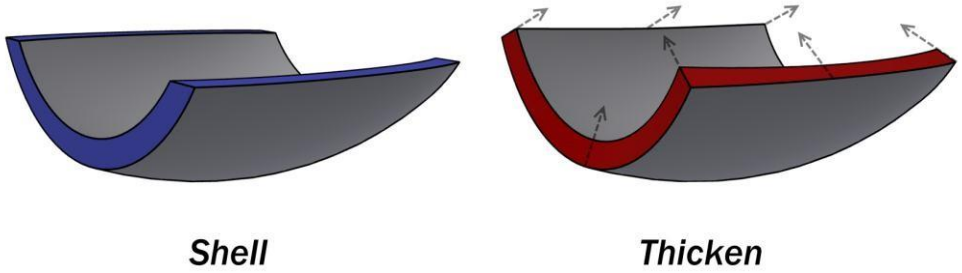
Offset

What's the difference between Shell and Thicken?

The Shell, Thicken and Offset tools all use the same algorithm for generating the offset faces, but what differs between them is **the way that the side wall faces are constructed** (these are the thin faces connecting the offset faces with the original faces).

The Shell command requires a solid body to be supplied as the input, and the user selects faces from the body to be "open faces". **The Shell command will use the geometry of the open faces** to create the side wall faces.

The Thicken command just takes a set of faces as input. Since there is no geometry to create the side wall faces from, **it builds new faces in the normal direction of the original faces** (i.e. perpendicular to the original faces).



The Shell tool uses the "open face" geometry to build its side walls (blue), whereas the Thicken tool will build new side wall geometry normal to the original input faces (red).

In practice, this tends to make **Shell the preferred tool for creating thin-walled parts** where possible, because **the geometry of the side walls can be easily predicted and controlled**. When using the Thicken command, be conscious of the side wall geometry being created, and make sure that it satisfies your design intent (if not, you can always use downstream features to tweak the geometry of these side walls).

The Offset command works similarly to Thicken, but since the output is just an offset surface, the tool will not create side wall faces. The boundary geometry of the offset surface will, like Thicken, be based on the normal direction of the original faces (you can think of an Offset as Thicken without the side walls).

Why does my Thicken only work with Automatic Blending switched on/off?

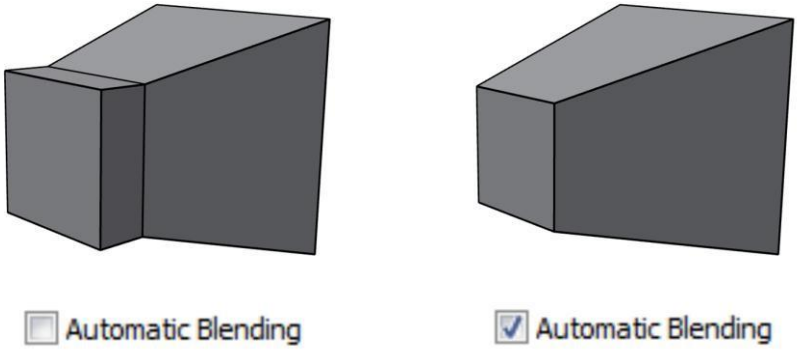
The Automatic Blending option was introduced to some modeling operations in Autodesk® Inventor® 2011; it allows fillets to be preserved when making local changes to filleted models.

For tools such as Move Face and Face Draft, this option will perform some extra processing before and after the operation to detect and reapply fillets where needed. But for Thicken, Automatic Blending does not give sensible results when applied to the traditional thickening algorithm, so **a new algorithm was introduced for operations where Automatic Blending is enabled**.

However, what this means is that some operations that do not involve fillets may also exhibit different behavior depending on whether the Automatic Blending option is checked or unchecked.

(Note that this option is only available when using Thicken to modify faces on an existing solid model; when thickening a sheet body the Automatic Blending toggle will be disabled.)

When the traditional thicken algorithm is performed on faces from a solid model, Inventor® will thicken the selected faces in isolation, and then combine this body with the existing solid model to give you the thickened result. However, when Automatic Blending is switched on, a more complex algorithm is used: the selected faces will be offset by the specified distance, and then the adjacent faces will be extended and re-intersected with the face to create the thickened result. This can create significantly different results to the traditional algorithm.



When thickening a face belonging to an existing solid model, the traditional Thicken algorithm (Automatic Blending off) will thicken the face in isolation and then join it to the existing solid, whereas with Automatic Blending on, the different algorithm will be used, which extends the surrounding face geometry to produce a "cleaner" result

In general, the traditional algorithm is less demanding and more likely to return a successful result. However, when thickening the faces of a solid body, **the Automatic Blending algorithm tends to give a more intuitive, "cleaner" result** (i.e. less new topology is introduced).

My surface model fails to Thicken/Shell, what can I do?

When a Thicken operation fails to successfully offset a set of faces, the problem is most likely isolated to one or two regions of difficult geometry or topology on the input body. A method that can be used to "debug" the problem is to **incrementally thicken** the model in stages, noting at which point the Thicken operation starts failing, and then identifying which face or combination of faces can reproduce the failure in isolation.

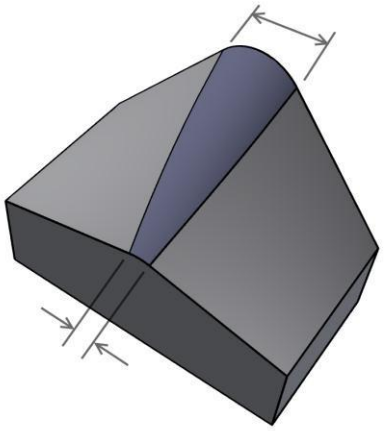
Examination of this region may reveal a difficult or erroneous condition on the input model, and simply rebuilding some of the model faces in this area may be enough to resolve the problem.

(This workflow can also be applied to Shell failures, but since using this „incremental" strategy would be very laborious if attempted with the Shell tool, it may be better to investigate these by making a surface copy of your solid body, and then use the Thicken tool to investigate the issue instead.)

Fillet

The Fillet tool is designed to round-off model edges for manufacturability purposes, but what the tool does on a technical level is to **create a chain of faces smoothly connecting sets of existing faces**.

For a regular constant radius fillet, these faces will have a circular cross-section and align tangent continuous to the faces being filleted. The method for constructing the surfaces is referred to as the **rolling-ball** algorithm, and can be thought of as rolling a ball bearing along model, with new edges being created where the ball bearing touches the two faces. Using this method results in fillet faces that are wide for sharp edges, and narrow for shallow edges.



Fillet uses the rolling-ball algorithm to build face geometry; this results in wide fillet faces when filleting sharp angles, and narrow faces when filleting shallow angles

This is different to the Chamfer tool, which uses an **edge-offset** algorithm. This ensures that the edges of the chamfer faces are a fixed distance from the original edge being chamfered. This results in faces that are wide for shallow edges, and narrow for sharp edges.

What can I do to resolve a Fillet failure?

It is difficult to define hard-and-fast rules for resolving fillet issues, since problems are often specific to the geometry & topology configuration being filleted. However, here are six tips that may give you some options to try if encountering problems when attempting a complex fillet operation:

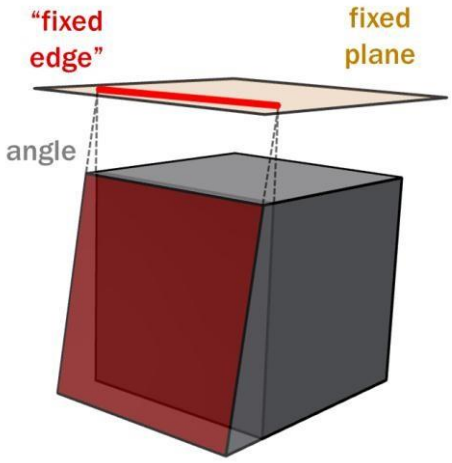
1. Try **breaking a single Fillet operation into multiple Fillet features**. This will allow you to experiment with different combinations of fillet sequences, and you may discover a successful set of sequences that the Fillet tool's own reordering mechanism was not able to identify.
2. The order in which fillets are placed can also impact success. Best results tend to be achieved when you perform **big fillets before smaller fillets**, and when you perform **concave fillets before convex fillets (rounds).**
3. If a problematic fillet operation "rolls onto" any sharp edges (i.e. a concave fillet encountering a sharp convex edge, or vice versa), **try filleting those sharp edges with a small radius first**: sometimes a fillet will find it easier to roll onto smooth edges rather than sharp edges.
4. If you identify any specific problematic edges or regions, try filleting one or more individual edges with **Automatic Edge Chain switched off**. This may create new routes for interconnecting fillet chains to travel through.
5. If you encounter problems at any specific vertex fillets (i.e. where three or more filleted edges meet at a vertex), try **enabling vertex setbacks** in the Setbacks tab of the Fillet dialog box, and experiment with different setback configurations and distances.
6. Try using **face fillet** as a substitute for a failing edge fillet chain, as this may sometimes travel through a different route in the algorithm and find a successful solution.

Draft

How does Fixed Plane Draft determine where to "pivot"?

When applying a face draft using either of the Fixed Edge or Parting Line modes, the curves about which the taper will "pivot" are explicitly defined by the fixed edge or parting line you provide. But for a Fixed Plane taper, only a plane is supplied as input, so it may not be clear where the origin of the taper's "pivot" lies.

For Fixed Plane taper, the "pivot" curves are determined **by intersecting your taper face(s) with the fixed plane**. If your fixed plane is some distance away from your taper faces, Inventor® will (in the background) extend your taper faces until they intersect the fixed plane, and then taper about this intersection curve.



The Fixed Plane Draft mode will determine its "pivot" curve by intersecting the taper faces with the specified fixed plane, and using the intersection of these two surfaces as the origin for pivoting

Why do my faces sometimes get split when I apply a draft angle to them?

When using the Fixed Edge or Parting Line mode apply a Face Draft, it is possible that a single face on the original model will be split into multiple faces on the tapered result.

This is the expected (and mathematically correct) result in cases **where the fixed edge or parting line contains vertices, or G2 discontinuities**. If this happens unexpectedly and does not seem to correspond to the fixed edge geometry, it may be the result of hidden continuity issues in the fixed edge curves.

Part 2: The Seven Deadly Sins

This section introduces the seven most common causes of failures & problems when modeling in Inventor, and describes techniques for dealing with them.

1. High curvature

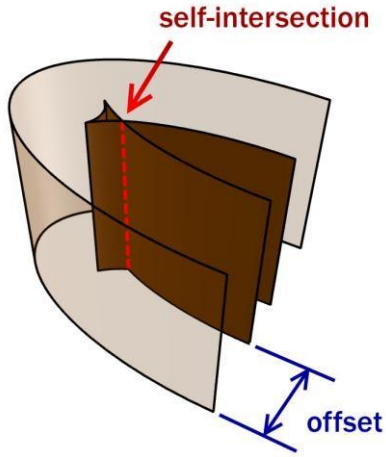
What is it?

Curvature is the measure of how "sharp" a surface (or curve) is: a planar surface has zero curvature; a surface with a shallow curve is said to have low curvature; and surface with a sharp curve is said to have **high curvature**.

Why is it a problem?

High curvature geometry is not itself a problem, and is generally only present as part of the design intent. High curvature geometry only poses a problem **if it is supplied as input geometry for a later modeling operation**.

If new geometry is based from existing high-curvature inputs, there is a chance that this new geometry may encounter a self-intersection.



If a high-curvature face is offset inwards by a large enough distance, the offset result will intersect itself

Some modeling operations have the capability to heal self-intersecting regions and create valid geometry (e.g. Offset and Fillet), but some operations may encounter failures if self-intersecting geometry is generated (e.g. Sweep).

What causes it?

High curvature is usually **introduced as part of the design intent**. In rare cases, **imported models** or **poorly-modeled surfaces** may feature an unexpected "fold" of high-curvature.

How can I diagnose it?

There are two diagnosis tools specifically design for analyzing face curvature the Surface Analysis tool (which can give a color-coded representation of the model curvature), and the Curvature Analysis tool (which draws “combs” along the surface to represent the curvature at discrete points).

How can I resolve it?

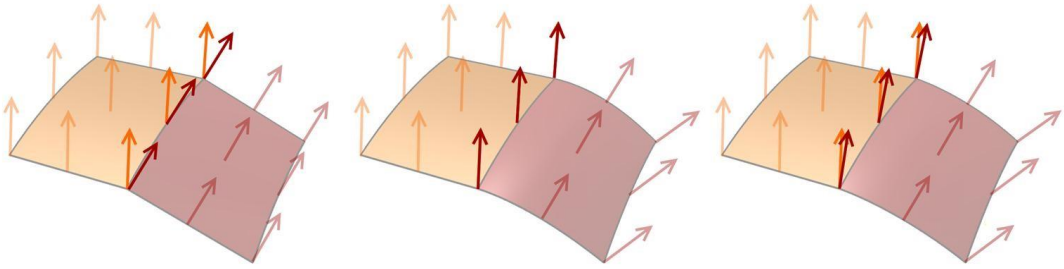
If a failure is encountered due to self-intersection in a region of high curvature, it may be possible to approximate the self-intersecting result by splitting the operation into separate chunks, and then joining these chunks together at the end (e.g. using the Combine tool).

2.Near-tangency

What is it?

Two adjacent faces are tangent-continuous with each other when they meet with a smooth join. The way to describe this mathematically is that the normal directions of both faces at their shared edge are identical.

If there is a discrepancy between the normal directions at any point along the edge, we no longer classify the edge as tangent-continuous. If the discrepancy is small, the edge may appear smooth to the naked eye, and it's possible that the design intent was for this edge to be tangent-continuous. We call this condition **near-tangency**.



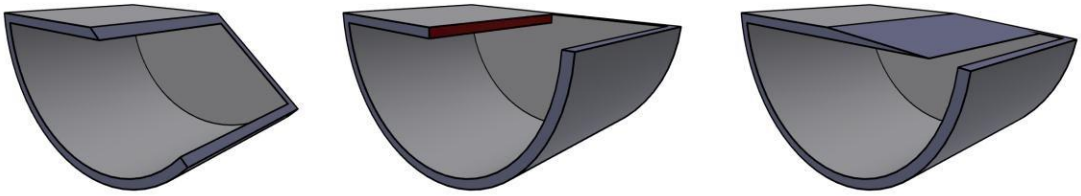
Non-tangent *Tangent* *Near-tangent*

Whether an edge is classified as tangent-continuous or not is determined by the surface normal directions of the two neighboring faces: if these are identical along the entire edge, it will be classified as tangent continuous

Why is it a problem?

Problems can arise from the fact that Inventor® often uses different algorithms depending on whether a piece of topology is classified as tangent or non-tangent.

The tangent algorithms give desirable behavior for smooth edges, and the non-tangent algorithms give desirable behavior for sharp joins. But if an edge is near-tangent, Inventor® will use the non-tangent algorithm for geometry that's better suited for the tangent algorithm. This can result in unexpected shapes, or more commonly, modeling failures.



Non-tangent *Tangent* *Near-tangent*

The Shell tool will use a special "tangent algorithm" that creates new side wall geometry at any tangent edges adjacent to an open face; for a near-tangent edge, the regular "non-tangent algorithm" will be used, resulting in an undesirable shape

What causes it?

There are two common sources of near-tangent edges. The first is from native Inventor® model features where the design intent was for a tangent join, but **a tangent condition was not explicitly set** when creating the new surfaces. The second common source is from **imported models**, where the original CAD software may have used a looser tolerance than Inventor® does to define tangent-continuity.

How can I diagnose it?

Because filleting can only be performed on non-tangent edges, you can **use the Fillet tool to "scan"** for which edges are classified as tangent and which are classified as non-tangent. Open the Fillet command, and hover your mouse over edges in the graphics window. If an edge gets highlighted as selectable, it is classified as non-tangent; so, if this happens but the edge appears to be smooth, this could be considered to be a near-tangent edge.

You can alternatively perform a similar trick outside of the Fillet command, by clicking on suspected edges. If the in-canvas Fillet/Chamfer buttons appear, this will also tell you that the edge is classified as non-tangent.

How can I resolve it?

If a near-tangent edge was created from native Inventor® features, edit the responsible feature and try to enforce a true tangent-continuity at this edge.

If the edge comes from imported model data, the best solution may be to enforce stricter tangent-continuity in the original CAD package, and re-import the model data.

If this is not possible, an alternative is to rebuild one of the faces using a native Inventor® patch or loft surface, and apply the correct tangency conditions here.

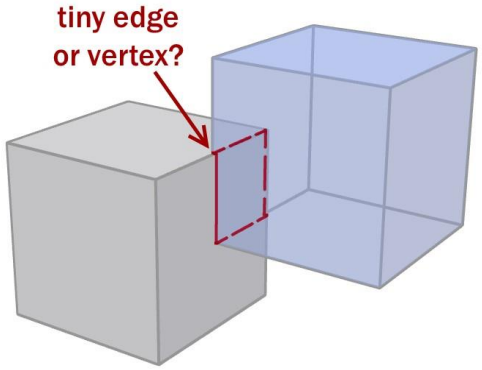
3.Near-coincidence

What is it?

Near-coincidence refers to two overlapping pieces of geometry (typically face geometry) that are **nearly the same, but not identical**.

Why is it a problem?

This condition will cause problems for any operations that perform a Boolean calculation (for example, shape creation tools that involve a Join, Cut or Intersect between the new body and the existing solid body). Boolean operations work by calculating the intersections between the bodies being interacted, and **if two pieces of geometry being intersected nearly identical but not quite, finding the precise intersections between them is very difficult**.



When trying to find the intersections between near-coincident geometry, it is likely that the calculations will encounter borderline cases that are difficult to solve

What causes it?

Near-coincidence can be introduced natively in Inventor® **if new geometry is built "similar" to existing geometry** (e.g. if a Boundary Patch is built using the edges of an existing face – the edges will be the same, but the underlying surface geometry will be different, and possibly near-coincident). It may also come from **imported models** where coincident faces are not precisely aligned.

How can I diagnose it?

Since the problem usually causes a failure at the Boolean stage, if you **reattempt a failing operation with the "New Solid" option checked**, and the operation then starts working, there is a strong possibility that near-coincidence is responsible for the operation failure.

How can I resolve it?

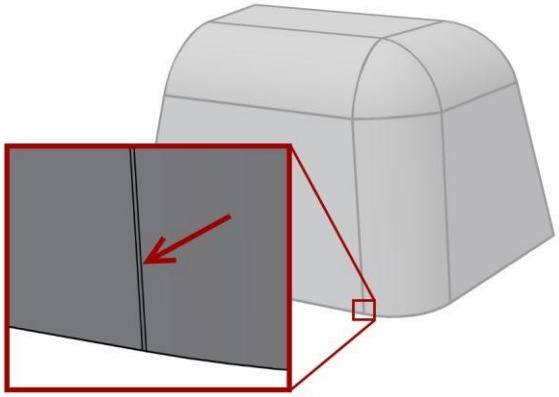
Boolean operations are Inventor's primary method for joining shapes together, but an alternative method for joining shapes that does not rely on the Boolean intersection calculations is **Stitch**. If you can delete faces so that your two shapes meet at a stitch-able gap, this is quite likely to yield success.

Another workaround is to try replacing the geometry of one of the near-coincident faces with the geometry of the other, thus turning a near-coincident join into a precisely-coincident join.

4.Sliver faces

What are they?

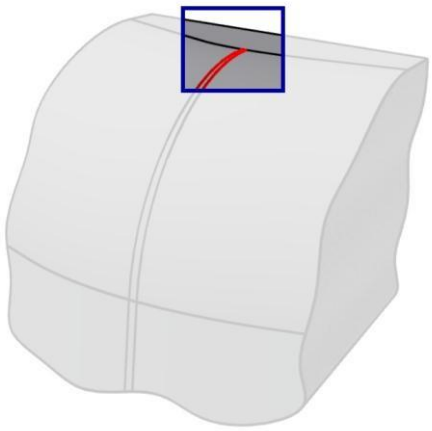
Sliver faces are technically just **extremely thin faces**. We generally classify a thin face as a sliver face when it is unexpected or not part of the design intent. Sliver faces are often so thin that they are invisible at normal zoom levels, and can only be seen when the view is zoomed extremely tightly to the affected region.



Sliver faces are often so small that they are only visible when the view is zoomed very tightly to the relevant region of the model

Why are they a problem?

These are a problem because Inventor® will treat sliver faces as regular faces, and calculate proper topology changes when these are operated upon. Having such a thin face means that **there are lots of topological entities (edges & vertices) squeezed within a very small space on the model**, and if this is operated upon with a relatively large modeling operation, the topology in this tightly-packed space may interact in unexpected ways.



Performing a modeling operation in the region of a sliver face can cause the topology around the sliver face to interact in unexpected ways, which may result in operation failures

What causes them?

Sliver faces are most commonly come directly from **imported models**. It is also possible for these to be introduced natively, generally as **the downstream result of poor modeling practice** (for example, near tangency or near-coincidence introduced from earlier features, or constraints & dimensions not being accurately applied).

How can I diagnose them?

Identifying sliver faces is a difficult task, because they are, in their nature, virtually invisible. A trick that might be helpful is to get a count of the number of model faces, and see if that number matches your expectation.

You can do this using the **Unstitch tool in the Repair environment**. Make a copy of your body using the Copy Object tool, and then use the Repair Bodies tool to open this in the Repair environment.

Activate the Unstitch tool, and highlight the entire body: this will report the number of model faces that have been selected. If the number is larger than your expectation, it's possible that there are one or more sliver faces hidden on your model.

How can I resolve them?

Once you have found a sliver face, removing it and healing the model is usually not difficult, and this will successfully resolve any related modeling failures in most cases.

The first tool to try for removing sliver faces is **Delete Face with the Heal option enabled**, which will remove the sliver face and attempt to re-intersect the surrounding geometry.

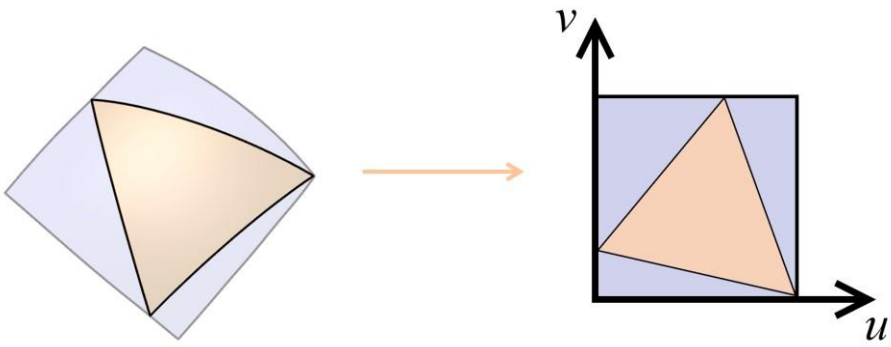
If this fails, another possibility is to use the Delete Face tool *without* Heal enabled (leaving a gap in the model where the sliver face was), and use Stitch to heal the gap (this is only recommended if you can successfully stitch the gap using a small tolerance value).

If both methods fail, the only solution may be to delete one of the adjacent faces along with sliver face itself, and then replace this with a new face that meets the adjacent topology accurately.

5.Singularities

What are they?

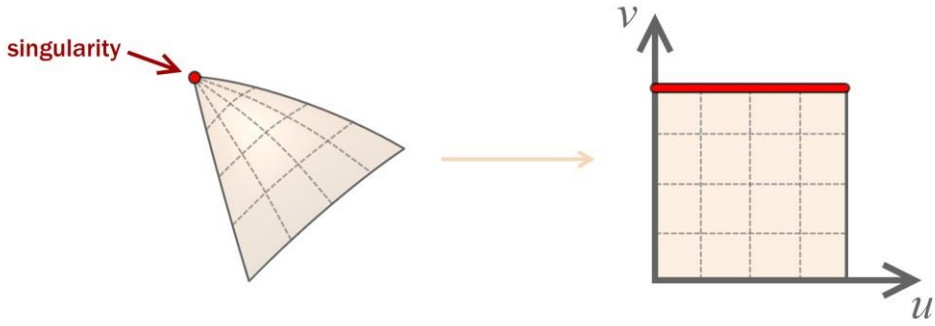
In 3D modeling operations, it is common for three-dimensional surface geometry to be mapped into a two-dimensional representation of the surface (called the “parameter space” of the surface). The Boundary Patch tool is specially designed to make underlying surfaces that are rectangular, and therefore map very cleanly into a two-dimensional representation.



The Boundary Patch tool is designed to create faces with a rectangular underlying surface, which will therefore map neatly into two-dimensional parameter space

Other shape creation tools (such as Loft and Sweep) will generally base the new underlying surface geometry on the operation’s input geometry.

If a surface created with one of these tools results in a three sided underlying surface, this will still be mapped to a rectangular shape in two-dimensional space, so effectively one point on the 3D surface is represented by an entire “edge” of the 2D rectangle. This point on a surface is called a **singularity** (this may also be referred to as a “degenerate point”).



If an underlying surface has three sides, it will still be represented as a rectangular shape in parameter space, so one whole side of this “rectangle” will represent one point in 3D space: this point is what we refer to as a singularity

Why are they a problem?

A surface singularity causes **ambiguity** when trying to map three-dimensional calculations into the two-dimensional representation. Usually a position on the 3D surface can be accurately and unambiguously mapped to a position on the two-dimensional representation, but this is not so for a singular point.

What causes them?

Singularities are most commonly introduced by **lofting a surface within a three-sided edge chain**, or **revolving a surface around a fixed point**. They may also be introduced by importing model entities that was created in a similar manner in other CAD software.

How can I diagnose them?

There are two ways to analyze the shape/structure of a faces underlying surface:

1. The **Extract Loop tool in the Repair environment** is designed to return the untrimmed underlying surface of an input face. If you suspect a model face contains a singularity, you can create a copy of it using the Copy Object tool, then take this into the Repair environment using the Repair Bodies tool. If the output of the Extract Loop command is a three-sided surface, this is probably converging to a singular point.
2. Another trick you can use for analyzing the structure of a face's underlying surface is that the **Curvature Analysis** tool will always draw its curvature combs in the direction of the surface parameter lines. If you draw curvature combs on a face using this tool, and the combs in one direction all converge to a single point on the surface, that point is a singularity.

   (When using this trick, it may be useful to minimize the height of the combs and increase the number of them, so that you can solely focus on the direction that the combs are flowing in.)

How can I resolve them?

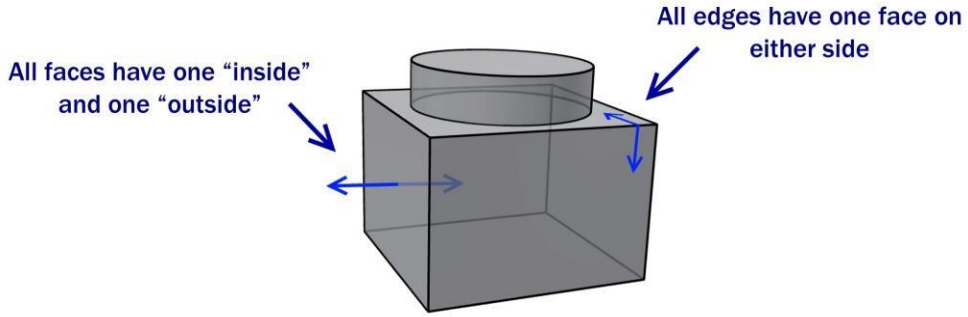
Since the Boundary Patch tool is designed to create a four-sided underlying surface, no matter how many edges the face has, deleting a face with a singularity and replacing it with a Boundary Patch should remove the problem (but make sure that the new Boundary Patch face meets your design intent; the underlying surface structure change could result in significantly different face geometry).

If the singularity is part of the design intent, you can minimize the possibility of modeling failures by making sure that the singularity is not "nearly-flat" (i.e. slight variation in the surface normals at the singular point).

6.Non-manifold topology

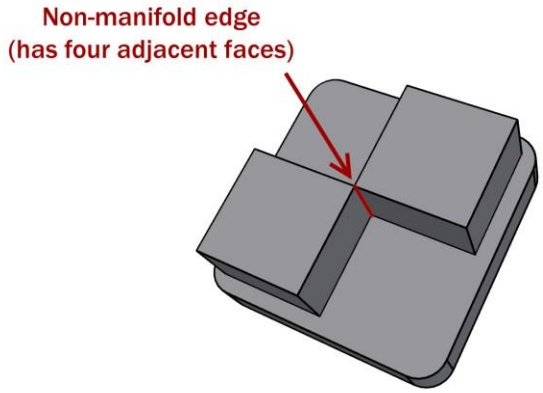
What is it?

A model is manifold if it realistically represents a solid manufacturable object. To qualify as manifold, its topology must meet certain conditions: for example, each edge can only have one face on either side, and each face must have one side facing “inside” and one side facing “outside” of the model.



A model must meet numerous criteria to qualify as a valid manifold solid (this diagram just shows two examples)

If any topology breaks these rules, we refer to it as **non-manifold topology**. A classic example of this is a shape consisting of two cubes sharing a single edge: this edge is a non-manifold edge, because it has four faces adjacent to it.



If two cubes join at a single shared edge, that edge is non-manifold, because it has four adjacent faces (two faces from each cube)

Why is it a problem?

Inventor® usually assumes that operation inputs are manifold, because **non-manifold topology can introduce ambiguities for a modeling operation**.

For example, in the case of two cubes sharing an edge: if you want to apply a fillet to that edge, there are four equally valid possible solutions.

On a manifold model, when you fillet an edge, the set of faces you want to be filleted together is unambiguous; this is the assumption generally made by Inventor® when accepting inputs for an operation.

What causes it?

Non-manifold edges are usually introduced to an Inventor® model simply by **creating features that precisely touch at a single edge**.

Non-manifold faces are generally less commonly created natively inside Inventor® (Inventor® won't recognize a model as a solid body if it contains non-manifold faces); these are more likely to come from **imported models where the originating CAD package did not enforce logical manifold joins between faces**.

How can I diagnose it?

If non-manifold edges or vertices were introduced natively in Inventor, these should be **visible on the model as "just-touching" features**.

If you **import some model data and the Stitch operation fails**, this could be a sign that the imported data does not represent a valid manifold model.

How can I resolve it?

Usually non-manifold topology, once found, can easily be avoided by **reordering features or revising the modeling workflow**. In cases where non-manifold topology is the design intent, you may need to build temporary shape features to make the model manifold if any operation failures are blocking your downstream modeling workflow.

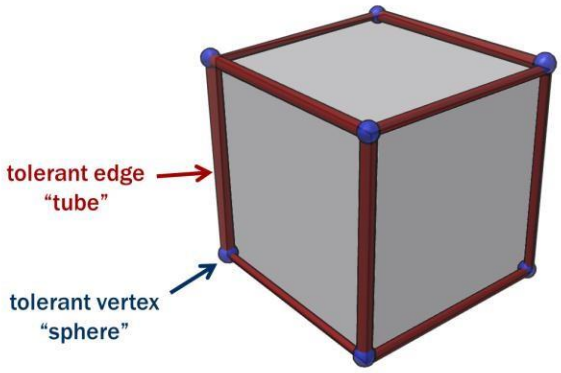
7.Loose tolerant geometry

What is it?

When you import 3rd-party model data with geometric tolerances looser than Inventors own internal tolerance, or which does not contain information about topological joins, Inventor® will need to Stitch the model to construct valid solid and surface bodies.

The Stitch command allows you to specify a tolerance within which you want neighboring edges and vertices to be considered as one topological entity.

After stitching, it appears from the graphics window that model edges have been "glued together"; but in fact, the original model geometry hasn't been adjusted: Inventor® has merely defined "tubes" (for tolerant edges) and "spheres" (for tolerant vertices) within which multiple geometric entities are considered to be part of the same topological entity.



After stitching, a model will have tolerant edges and vertices where separate geometric entities were joined to create new topological entities; these can be thought of as "tubes" and "spheres" that encapsulate all the geometric entities belonging to one topological entity

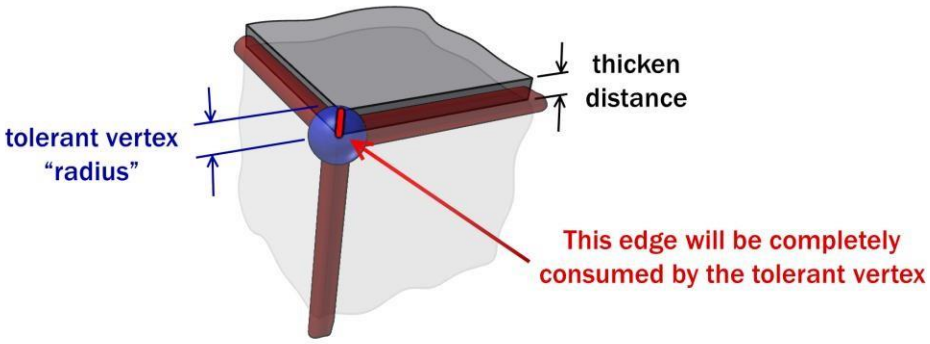
This results in model **geometry with tolerances larger than Inventors regular modeling tolerance**; but in most cases, the tolerance required to fully stitch a model will still be very small with respect to the part scale (typically fractions of a millimeter).

However, if a large tolerance value is required to Stitch a model (typically values over 0.1mm/0.004in), this results in **loose tolerant geometry** which may contribute to modeling issues downstream.

Why is it a problem?

Geometry tolerances add uncertainty to a model, but usually this uncertainty is small enough to be negligible. However, if **model tolerances are large with respect to the modeling operations being performed**, there is a possibility that the presence of this tolerance will interfere with modeling operation behavior.

For example, if we thicken a face by a small distance, but one of the vertices on that face has a tolerance larger than the thicken distance being used, the newly created edge in that region of the model will lie entirely within the "sphere" of the tolerant vertex: therefore, the tolerant vertex will "consume" the new topology in this region, which will most likely result in a modeling failure.



If a tolerant entity is large with respect to modeling operations being performed, it's possible that newly created model entities will be erroneously consumed by existing tolerant edges or vertices, resulting in invalid models or modeling failures

What causes it?

If a large Stitch tolerance is required to create valid model topology in Inventor, this is a knock-on effect of loose tolerances between the faces of the original model, which nearly always stems from when 3rd-party data was originally imported into Inventor. In these cases, it's likely that **the software used to create the original data only enforced a loose tolerance between geometric entities**.

How can I diagnose it?

Needing a large tolerance to fully stitch a model should be a warning sign of loose geometric tolerances.

If large tolerances are already present on the stitched model, you may be able to see large tolerances in the graphics window as "edge spikes" in the region of a vertex.

How can I resolve it?

The best solution for this problem is to **enforce tighter geometric tolerances in the original CAD system**. If this is not possible, the only workaround may be to rebuild some of the model entities inside Inventor® (it is preferable to do this before stitching, since healing a large tolerant vertex is not easy one the model has been stitched).

The Stitch tool provides feedback about which model entities failed to stitch together for a given tolerance, so you can use this information to identify the areas with tolerances larger than your desired maximum, and thus which regions of the model need attention.