Meschers: Geometry Processing of Impossible Objects

ANA DODIK, MIT CSAIL, USA
ISABELLA YU, MIT CSAIL, USA
KARTIK CHANDRA, MIT CSAIL, USA
JONATHAN RAGAN-KELLEY, MIT CSAIL, USA
JOSHUA TENENBAUM, MIT, USA
VINCENT SITZMANN, MIT CSAIL, USA
JUSTIN SOLOMON, MIT CSAIL, USA

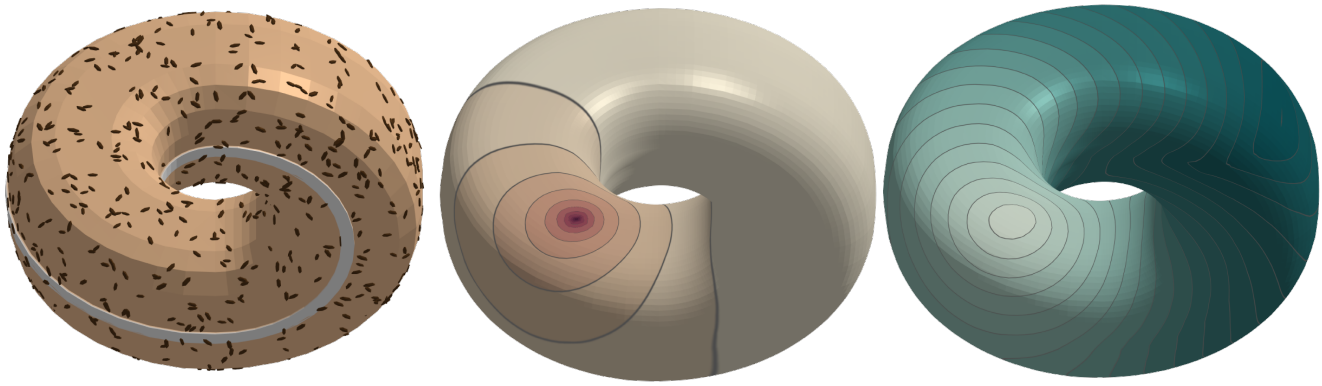


Fig. 1. The Impossibagel. The *mescher* is a geometry representation that allows rendering and relighting impossible objects (left), as well as performing intrinsic geometry processing operations like heat diffusion (center) and geodesic distance queries (right).

Impossible objects, geometric constructions that humans can perceive but that cannot exist in real life, have been a topic of intrigue in visual arts, perception, and graphics, yet no satisfying computer representation of such objects exists. Previous work embeds impossible objects in 3D, cutting them or twisting/bending them in the depth axis. Cutting an impossible object changes its local geometry at the cut, which can hamper downstream graphics applications, such as smoothing, while bending makes it difficult to relight the object. Both of these can invalidate geometry operations, such as distance computation. As an alternative, we introduce Meschers, meshes capable of representing impossible constructions akin to those found in M.C. Escher's woodcuts. Our representation has a theoretical foundation in discrete exterior calculus and supports the use-cases above, as we demonstrate in a number of example applications. Moreover, because we can do discrete geometry processing on our representation, we can inverse-render impossible objects. We also compare our representation to cut and bend representations of impossible objects.

Authors' Contact Information: Ana Dodik, anadodik@mit.edu, MIT CSAIL, USA; Isabella Yu, MIT CSAIL, USA; Kartik Chandra, MIT CSAIL, USA; Jonathan Ragan-Kelley, MIT CSAIL, USA; Joshua Tenenbaum, MIT, USA; Vincent Sitzmann, MIT CSAIL, USA; Justin Solomon, MIT CSAIL, USA.

ACM 1557-7368/2025/8-ART70
https://doi.org/10.1145/3731422

CCS Concepts: • **Computing methodologies** → **Mesh models**; **Mesh geometry models**; • **Applied computing** → *Arts and humanities*; • **Mathematics of computing** → *Mathematical analysis*.

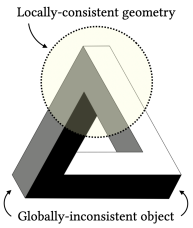
Additional Key Words and Phrases: Impossible objects, mesh representations, perception, vision science.

ACM Reference Format:
Ana Dodik, Isabella Yu, Kartik Chandra, Jonathan Ragan-Kelley, Joshua Tenenbaum, Vincent Sitzmann, and Justin Solomon. 2025. Meschers: Geometry Processing of Impossible Objects. *ACM Trans. Graph.* 44, 4, Article 70 (August 2025), 10 pages. https://doi.org/10.1145/3731422

1 Introduction

Recent work in vision science notes that visual art often aligns less with the laws of physics and instead makes sense only in the context of human perception [Hertzmann 2024]. Extreme examples of this phenomenon include *impossible objects* [Penrose and Penrose 1958], such as those shown in Figure 1. Impossible objects often appear in visual art; possibly the most famous examples are the woodcuts of M.C. Escher [1961], whereas more recent examples include popular media such as the film *Inception* and videogames *Hocus* and *Monument Valley*. Computer graphics is the computational language of visual arts, begging the natural question of how to represent constructions like those found in Escher's work computationally.

At first, it might seem counterintuitive to consider the geometry of impossible constructions, but the impossibility only arises when we attempt to embed impossible objects in 3D. Focusing



on only one corner of an impossible object, we are able to consistently perceive its *local* geometry. Its shading offers locally consistent depth and connectivity cues, e.g., a uniform color in a region implies that the region is flat and connected. It is only when we attempt to assemble—i.e., integrate—local information into a *global* geometry that the impossibility arises [Crane et al. 2013; Freud et al. 2013, 2015; Heinke et al. 2021]. What makes the object impossible is that one cannot assign an absolute depth to every point on its surface without cutting or bending the geometry, which would violate the local depth cues. In fact, all previous representations of impossible objects resort to either cutting or bending (see Section 2.1). This often breaks the geometry processing pipeline (Figure 8).

In contrast, we set out to computationally represent impossible objects in a way that still allows us to perform common graphics and geometry processing tasks. To this end, we introduce the **mescher**, a perceptually-inspired mesh data structure for Escheresque constructions. Unlike previous work, our method never integrates the impossible object into 3D coordinates, side-stepping the need to cut or bend the object (see Section 2.1). This means that, unlike in previous work, common graphics operations, such as relighting and intrinsic distance queries, behave consistently with human intuition (Figure 8).

Discrete exterior calculus (DEC) provides a natural language for this task and enables us to construct common discrete differential operators on impossible objects. In particular, we represent the relative depths of the surface of a mescher using the sum of exact and harmonic primal 1-forms in DEC. As a consequence, meschers allow artists to create visuals containing complex impossible objects by relying on graphics tools they are already used to, such as Laplacian smoothing and mesh subdivision. Our representation also allows us to answer intrinsic shortest path queries, with potential applications in vision science research or in path planning for video games. Lastly, we demonstrate how the differential operators produced by our representation enable inverse rendering of impossible objects.

In summary, our contributions are:

- A geometric representation of impossible objects built on DEC.
- The rendering and relighting of impossible objects.
- Common modeling operations of smoothing and subdivision.
- Answering distance queries on impossible objects.
- Inverse rendering of impossible objects.
- A thorough qualitative analysis, as well as comparisons with previous methods.

2 Related Work

For decades, impossible objects have captured the attention of scientists [Penrose and Penrose 1958] and artists alike [Escher 1961; Hogarth 1754; Reutersvärd 1934]. Over the years, many attempts were made to mathematically formalize these shapes. Simon [1967] introduced a model of human perception that involves "scanning" a shape to determine whether or not locally plausible-looking regions can be assembled into a globally possible whole. Similarly, Gregory [1970] hypothesized that humans detect impossibilities through conflicting depth hypotheses, whereas Draper [1978] conjectured that the problem involves a violation of the assumption that the object be spatially connected. In our paper, we phrase all of these observations using the mathematical formalism of *global integrability* (§3.1). We pose our notion of integrability in terms of differential forms, which comprise elements of the de Rham cohomology; note connections between cohomology groups and impossible objects were also explored by Penrose [1993]. Lastly, similar to our treatment (§3.1, S3.4), Térouanne [1984] attempt to explain perception of impossible objects in terms of a 2D tesselation of the image and a graph ordering of the image segments.

2.1 Rendering impossible objects

Over the past two decades, the computer graphics community has developed many methods for rendering impossible objects. These methods naturally divide into two categories. Methods in the first category (which we call **"cut representations"**) approach the problem by dissecting an impossible object into a collection of distinct parts, each of which is locally-consistent, but which are assembled into a globally-inconsistent whole [Khoh and Kovesi 1999; Lai et al. 2015; Li et al. 2024; Owada and Fujiki 2008; Savransky et al. 1999]. Taylor [2020] extends this approach to compute plausible-looking shadows for impossible objects. The mescher can be seen as the limit of this part-based approach, where each triangle is considered its own "part." This decomposition allows us to perform a variety of powerful intrinsic geometry processing operations on impossible objects.

The second category of methods (which we call **"bent representations"**) generates view-dependent deformations of a possible object's underlying geometry, such that from a particular non-generic viewpoint the object appears impossible [Elber 2011; Sánchez-Reyes and Chacón 2020; Wu et al. 2010]. These methods differ from ours in that the geometry they *represent* is different from what people *perceive*. For example, while people intuitively see all faces of the Penrose triangle as flat, these methods curve those faces to make them match up correctly in 3D space. Intrinsic geometry processing operations on such representations (e.g. computing Gaussian curvature) do not produce intuitive results, as they do on meschers.

In Section 4.6, we show how we can recover both of these types of representations from meschers.

2.2 Modeling and reconstructing impossible objects

How can we produce new impossible objects to render? One approach is to create user interfaces to model impossible objects. Sugihara [1997] designed a user interface for modeling view-dependent impossible objects, which can then be folded from paper. Owada and Fujiki [2008], Inglis [2014] and Taylor [2020] offer user interfaces for interactively constructing and rendering complex impossible objects. Li et al. [2024] procedurally design impossible structures that satisfy simple user-provided geometric constraints. Following this line of work, we allow users to create and edit meschers using operations like subdivision and smoothing.

Another approach to producing impossible objects is to automatically reconstruct their geometry from 2D images. Karpenko

and Hughes [2006] attempt to reconstruct a (possible) mesh from a line drawing of an impossible object, but find that the resulting mesh appears "flattened" and does not produce the experience of impossibility. As we will show in Section 4.5, the mescher enables direct inverse rendering of impossible objects. Weber* et al. [2024] reconstruct 3D scenes from geometrically-inconsistent cartoon animations by deforming the input images to maximize consistency. In contrast, the mescher enables recovering inconsistent geometry without warping the input.

2.3 Perception of impossible objects

Psychologists and vision scientists have long studied how people perceive impossible objects. The subjective experience of viewing an impossible object is driven by the spatial locality of vision. Because of foveation in the human retina, we perceive the world through small, spatially-localized fixations that we assemble into an abstract global percept [Biederman 1985]. Early visual processing extracts local cues like shading and occlusion, which provides the visual system with some information about geometry and depth. This information can be assembled into what Marr [1982] calls a "2.5-D sketch": a precise representation of local surface orientation, combined with a coarse representation of global depth. Impossible objects surprise us by providing valid local depth cues that fail to integrate into a consistent global depth embedding [Freud et al. 2013, 2015; Heinke et al. 2021]. Hence, the mescher represents local geometric properties (e.g. orientation and curvature) without committing to global depths.

Importantly, our visual system is robust enough to make partial local sense of such objects [Koenderink 1998; Linton et al. 2023]—when faced with an impossible object, we only detect the "impossibility" when an attempt to integrate multiple fixations leads to an obvious contradiction [Schuster 1964]. This feature of human visual perception has important consequences for graphics more broadly [Hertzmann 2024]. For example, classic methods for creating multi-perspective images and panoramas [Agarwala et al. 2006; Agrawala et al. 2000; Roman et al. 2004] rely on people's ability to integrate multiple local perspectives into a coherent global perspective, even when no single global perspective could explain the given image. In this sense, the mescher is a first step towards developing flexible geometric data structures that can represent the wide variety of perspectively-inconsistent scenes that artists wish to depict.

2.4 Beyond impossible objects

Beyond impossible objects, the graphics community has long developed methods of automatically generating a wide variety of visual illusions [Chi et al. 2008; Chu et al. 2010; Freeman et al. 1991; Ma et al. 2013; Okano et al. 2010; Oliva et al. 2006] and perceptually-ambiguous images [Burgert et al. 2023; Chandra et al. 2022; Geng et al. 2023, 2024; Wang et al. 2020]. These methods are valuable not only for creating interesting and entertaining images, but also for helping understand human visual perception through techniques in visual art [Cavanagh 2005; Livingstone 2022].

2.5 Differential mesh representations

Meschers are closely related to differential representations of meshes [Sorkine 2006], which have applications in editing and deformation [Alexa 2003; Lipman et al. 2004]. Like meschers, differential representations store a vertex's relationship with its neighbors, rather than its absolute position. A key difference, however, is that meschers only require *local* integrability, not *global*. This is what allows meschers to represent impossible objects.

Our representation is built on machinery of DEC [Desbrun et al. 2006; Hirani 2003], which offers a natural way to talk about differential quantities on meshes; see the survey by Crane et al. [2013] for more information. Interestingly, this survey in fact alludes to a connection between DEC and impossible objects, without pursuing the topic in more detail [Crane et al. 2013, pg 103].

DEC and tools from vector field design have been used in other scenarios in need of a local notion of integrability. For example, branched covers of surfaces, which locally integrate to injective parameterizations away from singular points, appear in algorithms for quadrilateral remeshing [Bommes et al. 2023; Kälberer et al. 2007] and stripe pattern design [Knöppel et al. 2015].

3 The Mescher

What makes impossible objects impossible is that they cannot be integrated into 3D, i.e., we cannot reconstruct *absolute* depth values of the vertices in a way consistent with our depth perception. Therefore, to model impossible objects in a way that corresponds to perception, we must design a geometry representation that stores topology, screen-space vertex positions, and encodes *relative* depths.

In general, impossible objects assume the existence of an observer coordinate frame and can only be visualized when viewed head-on. Therefore, we assume an orthographic camera and operate in its coordinate system. From the perspective of our camera, the x axis is left to right, y down to up, and z backward to forward.

A mescher has the same topology as a typical manifold oriented triangle mesh with (or without) boundary, with sets of triangle faces $\mathbb{F}$, edges $\mathbb{E}$, and vertices $\mathbb{V}$; we assign each edge in $\mathbb{E}$ and face in $\mathbb{F}$ an orientation. To represent a mescher's geometry, we store *per-vertex* screen-space coordinates, $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{|\mathbb{V}|}$. For depths, we store a value for relative depth *per-edge*, $\boldsymbol{\zeta} \in \mathbb{R}^{|\mathbb{E}|}$. Intuitively this means that, if an edge $i \in \mathbb{E}$ connects vertex $p \in \mathbb{V}$ to vertex $q \in \mathbb{V}$ then ζ_i represents the *signed* change in depth from p to q. The word "signed" here signifies that the change from p to q is equal to the negative change from q to p.

Looking at this construction from the perspective of discrete exterior calculus (DEC) [Crane et al. 2013; Desbrun et al. 2006; Hirani 2003], $\boldsymbol{x}$ and $\boldsymbol{y}$ are primal 0-forms (one scalar per vertex), and $\boldsymbol{\zeta}$ are primal 1-forms (one value per edge). Primal 1-forms represent the sum total change in a value across an edge; in our case the change in depth. Leveraging this relationship, we can also build up DEC machinery on a mescher. In particular, as the discrete exterior derivatives only depend on the topology, their construction follows in the same way as it does for regular meshes. This gives us access to discrete exterior derivatives $\mathrm{d}_{01} \in \{-1, 0, 1\}^{|\mathbb{E}| \times |\mathbb{V}|}$ and $\mathrm{d}_{12} \in \{-1, 0, 1\}^{|\mathbb{F}| \times |\mathbb{E}|}$ in the primal domain, which map k-forms to

$(k+1)$-forms. The transposes of these matrices $\mathrm{d}_{12}^\top$ and $\mathrm{d}_{01}^\top$ provide exterior derivatives on the dual mesh. The remaining operators in the discrete de Rham cohomology, $\star_0$, $\star_1$, and $\star_2$, require some more work to construct from a mescher, as will be discussed in the following section.

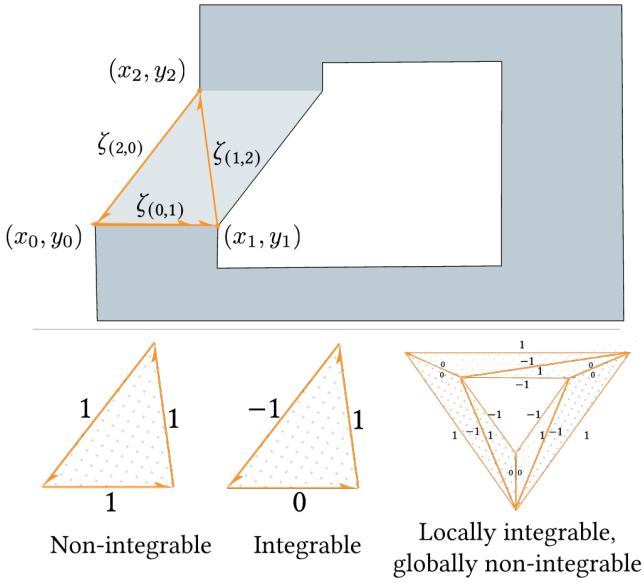


Fig. 2. A mescher is defined by a triangle mesh where vertices are stored as a pair of 2D screen-space coordinates (x, y) and edges include a directed relative depth offset ζ. The ζ's must integrate to zero around individual triangles (orange), but not necessarily around other loops, as shown in the bottom row.

3.1 Local Integrability and Geometry

We now articulate the precise space of meschers, which is smaller than the space of all possible discrete 1-forms. In particular, consider the example in Figure 2. Interpreting the 1-form $\boldsymbol{\zeta}$ as a difference in depths per edges of a triangle, the left example is *not integrable*, meaning that the depths do not sum to zero around the cycle of edges in the triangle; the right example is *locally integrable*, meaning they sum to zero. Local integrability implies that we can locally embed the triangle, i.e., extract a consistent set of depths for the three vertices given the depth of one—a condition that is needed for our mescher to have normal vectors, triangle areas/interior angles, and consequently a cotangent Laplacian operator. Local integrability does *not* imply global integrability, which is not necessary for a mescher, as shown in the third example; this property distinguishes meschers from globally-embeddable shapes.

In the language of DEC and recalling that d_{12} computes oriented sums of 1-forms around triangles, the local integrability condition can be expressed as a linear constraint:

$$\mathrm{d}_{12}\boldsymbol{\zeta} = \mathbf{0}. \tag{1}$$

This condition is connected to the discrete Poincaré lemma [Desbrun et al. 2005]. We will discuss in Section 3.3 how we obtain a feasible $\boldsymbol{\zeta}$ to initialize our editing system.

Once we have a $\boldsymbol{\zeta}$ satisfying the constraint above, we can obtain discrete Hodge star matrices to complete our DEC formulation. For each triangle on the mesh, we locally integrate $\boldsymbol{\zeta}$ to obtain depths for the three vertices (relative to one of the vertices); the screen-space coordinates of the vertices are stored in $\boldsymbol{x}$ and $\boldsymbol{y}$. Together, these coordinates allow us to compute triangle areas and half-edge cotangent weights, which we store in diagonal matrices $\star_0 \in \mathbb{R}^{|\mathbb{V}|\times|\mathbb{V}|}$ (one-ring barycentric areas), $\star_1 \in \mathbb{R}^{|\mathbb{E}|\times|\mathbb{E}|}$ (cotangent weights), and $\star_2 \in \mathbb{R}^{|\mathbb{F}|\times|\mathbb{F}|}$ (inverse triangle areas).

With the full set of DEC operators in place, we can use the discrete Hodge decomposition [Crane et al. 2013; Desbrun et al. 2006] to characterize the space of mescher 1-forms. In particular, any primal 1-form on a triangle mesh can be written as the sum of a curl-free component, a divergence-free component, and a harmonic component. The constraint in (1) implies that $\boldsymbol{\zeta}$ has no divergence-free component, leaving the factorization:

$$\boldsymbol{\zeta} = d_{01}\boldsymbol{z} + \boldsymbol{\omega}, \tag{2}$$

where $\boldsymbol{z} \in \mathbb{R}^{|\mathbb{V}|}$ and $\boldsymbol{\omega}$ is the harmonic component, satisfying $\mathrm{d}_{12}\boldsymbol{\omega} = \mathbf{0}$ and $\star_0^{-1}\,\mathrm{d}_{01}^\top\,\star_1\,\boldsymbol{\omega} = \mathbf{0}$.

Notice that when there is no harmonic component, i.e. when $\boldsymbol{\omega} = \mathbf{0}$, the mescher can be embedded in 3D space, Hence, meschers can represent both possible and impossible objects, and a mescher can be classified as possible or impossible by testing whether $\boldsymbol{\omega} = \mathbf{0}$.

In contrast to previous work which represents impossible objects through cutting or bending (see Section 2.1), our representation is compatible with many downstream geometry processing operations (see Section 4). The two discrete differential operators we make use of are the standard 0-form Laplacian, $\Delta_0 \coloneqq \star_0^{-1}\,\mathrm{d}_{01}^\top\star_1\,\mathrm{d}_{01}$, which use to answer intrinsic distance queries and perform inverse rendering in Sections 4.3 and 4.5, and the 1-form Hodge Laplacian, $\Delta_1 \coloneqq \star_1^{-1}\,\mathrm{d}_{12}^\top\star_2\,\mathrm{d}_{12} + \mathrm{d}_{01}\star_0^{-1}\,\mathrm{d}_{01}^\top\star_1$, which can be used for smoothing 1-forms as in Section 4.4.

Remark (Intrinsic geometry processing). *It is also possible to explain meschers in terms of edge lengths. The operators above involve only triangle areas and interior angles, all of which are recoverable from the discrete metric.*

3.2 Finite-element Operators

While meschers cannot be embedded globally, we can construct a *local* coordinate frame for each face. Differential quantities, such as gradients, only depend on the local tangent spaces, not on the global embedding. This means that, in addition to DEC, we can also construct standard 1st-order finite-element operators. This becomes useful in Section 4.3, where we need access to per-face gradients of a function. In our work, we use the gradients, $G_x, G_y, G_z \in \mathbb{R}^{|\mathbb{F}|\times|\mathbb{V}|}$, and the diagonal area matrix, $A \in \mathbb{R}^{|\mathbb{F}|\times|\mathbb{F}|}$. As expected, we can verify that we can construct a discrete Laplacian using finite elements, and that $G^\top AG = \mathrm{d}_{01}^\top\,\star_1\,\mathrm{d}_{01}$ and $A = \star_2^{-1}$.

3.3 Obtaining a Mescher

Having explained the inner workings of a mescher, we now briefly discuss how we can obtain one. Possibly the simplest way is to

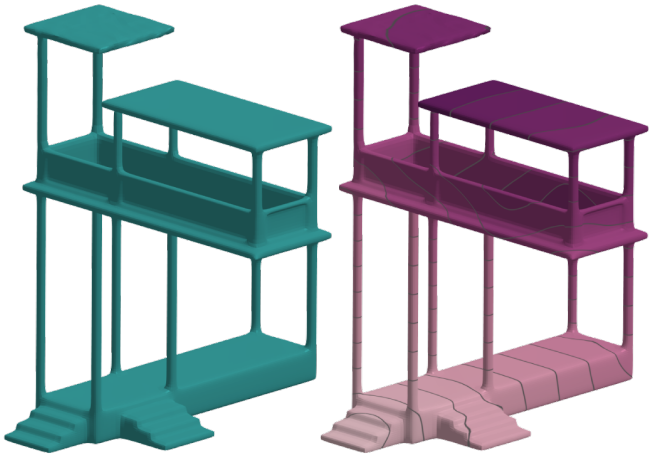


Fig. 3. Beautiful View. Using the heat method, we can compute geodesic distances on meschers (Section 4.3). Here, we show the distance measured from the front steps.

convert one from a "cut" 3D embedding of the shape, commonly used by previous work (see Section 2.1).

While impossible surfaces are difficult to reason about, creating a cut representation in 3D is an artistic practice like any other, with a vibrant community of artists and a plethora of video and textual guides [Elber 2011; Gárate 2020, 2023], books [Ernst 1986], as well as online libraries containing hundreds of different impossible objects [Alexeev 2001; Elber 2012]. Other approaches for obtaining impossible objects include direct modeling tools [Wu et al. 2010], procedural generation [Li et al. 2024], or inverse rendering (see §4.5, Figure 7).

This representation can be modeled using standard software; converting it to a mescher then involves extracting ζ from the 3D mesh and merging duplicate vertices at the cut as specified by the user. If the cut 3D model is imprecise, merging two pairs of vertices that are connected by an edge might lead to a conflict between two different values for ζ along the merged edge. Simple per-edge merging operations like averaging, however, may yield a 1-form $\vec{\zeta}$ that does not satisfy Equation 1. This and other similar situations necessitate the need for a means to *project* ζ back onto the feasible set.

Given an unconstrained ζ', we can obtain a ζ consistent with Eq. 1 by solving the following least-squares problem:

$$\begin{aligned} \min_{\zeta} \quad & \frac{1}{2}\|\zeta - \zeta'\|^2 \\ \text{s.t.} \quad & \mathrm{d}_{12}\zeta = \mathbf{0}. \end{aligned} \tag{3}$$

Introducing a Lagrange multiplier $\boldsymbol{\lambda}$, we obtain the optimal ζ by solving a linear system:

$$\begin{bmatrix} I_{|\mathbb{E}|} & \mathrm{d}_{12}^\top \\ \mathrm{d}_{12} & 0 \end{bmatrix} \begin{bmatrix} \zeta \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \zeta' \\ \mathbf{0} \end{bmatrix}. \tag{4}$$

In scenarios where there are multiple conflicting values for each edge's ζ, we minimize its averaged squared difference to both; a straightforward calculation shows that this is identical to solving (3) where the corresponding element of ζ' contains the average of the conflicting values.

Remark. *Ideally, in* (3) *the norm* $\|\zeta - \zeta'\|^2$*, would be weighted by* $\star_1$*, i.e.,* $\|\zeta - \zeta'\|^2_\star = (\zeta - \zeta')^\top \star_1 (\zeta - \zeta')$*. However, there is a "chicken-and-egg problem:" In the absence of a consistent* ζ*, we cannot compute a geometric* $\star_1$*. Therefore, instead of resorting to a complex nonlinear optimization problem, for this projection operator we opt for the simpler solution of using the purely topological* $\star_1 \coloneqq \mathbf{1}$*. This is equivalent to performing a discrete Hodge decomposition of* ζ' *using unit weights for* $\star_1$ *and setting the divergence-free component to zero.*

Once we have a locally consistent ζ, we can render the mescher and compute differential operators, opening doors to various graphics applications (§4).

3.4 Depth Ordering

A strong visual cue for depth and connectivity are so-called *T-junctions*. An example of this is visible in Figure 4, where our percept is that the middle part of the horizontal bar is in front of the middle part of the vertical bar, despite the object looking flat otherwise. So far, we have been dealing with representing the local relative depth ordering of connected geometry, meaning that we require one additional component to our model that would allow us to represent the global depth ordering of disconnected local patches.

We thus represent a *partial* depth ordering to the faces of a mescher via a directed acyclic graph, where an edge between faces i and j represents that i is perceived to be behind j. This is a separate data structure to the actual mesh, since a graph edge does not imply connectivity or geometry in the same way a mesh edge does. As the ordering is partial, most faces do not have any depth relationship to each other, which avoids having to assign an absolute depth order to local patches of an impossible object.

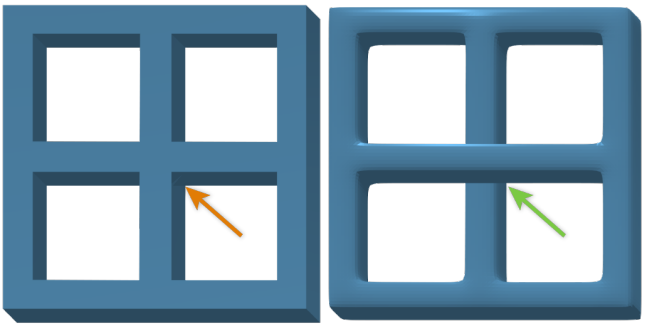


Fig. 4. Window. This mescher is globally integrable: naïvely rendering it makes the bars intersect at the same depth (left). To create an impossibility, an additional form of depth ordering is necessary (Section 3.4). After introducing depth ordering, we can render Window as an impossible object. Here, we further subdivide the mescher and apply our smoothing operator to create specular highlights that emphasize the impossible depth offset.

When converting a cut representation into a mescher, we compute intersections between all pairs of triangles and only enforce the

ordering for those that do intersect. To modify the initial ordering, a user can choose two groups of faces and assign one of them to be in front of the other. We warn the user in case of graph cycles.

It is possible no valid ordering is possible for a given construction. For example, if the Window mescher in Figure 4 were represented with few long triangles, a cycle would occur. This problem disappears under refinement, as the ordering is only specified for local patches. We cannot exclude—but have not observed—pathological examples where refinement might not help.

4 Operations on Meschers

Having built up the mescher data structure, we can move on to some practical example applications. We start of with the foundational graphics operations of rendering and lighting (§4.1), and mescher subdivision (§4.2), before moving on to more complex operations which make use of our differential operators in (§4.3-§4.5).

4.1 Rendering

To render a mescher, we flatten its triangles by setting their absolute depths to zero and use per-face or per-vertex normals derived from $\boldsymbol{\zeta}$ during shading. Impossible objects are only compatible with directional and environment map illumination. This is due to the fact that, if the incoming radiance depends only on the direction from which the light is coming, the shading becomes invariant to translations in $\mathbb{R}^3$. More complex emitters such as area lights require knowledge about the absolute 3D positions of objects and would therefore naïvely be incompatible with meschers. Meschers are, however, compatible with various local shading models; see Figure 9 for a simple example.

For a rendering to respects the depth ordering from Section 3.4, we construct a linear extension from the ordering graph, i.e., a total order that retains the ordering relationships of the partial order. In practice, this is accomplished by topologically sorting the depth ordering graph. Once we have a total ordering, we can overlay triangle renders in the order they appear in the topological sort.

Remark. *While our rendering procedure is in essence a normal map, a mescher is not the same as a mesh with a normal map. One could attempt to represent impossible objects using triangles with arbitrary normals. However, this would remove the guarantee that neighboring triangles would have the same value of ζ for their shared edge. If one were to formulate a least squares problem similar to in Equation 3 to recover consistent ζ from arbitrary normals, this would in effect bring us back to the mescher.*

4.2 Subdivision

Modeled geometry comes in varying levels of coarseness and further subdivision is often necessary to make them usable with downstream geometry processing tools. While subdividing 1-forms on general subdivision surfaces is possible [de Goes et al. 2016], our reason for subdividing is to make finite elements simulation work and not visual. Therefore, we opt for the comparably simpler Loop subdivision [Loop 1987] and leave other subdivision schemes for future work.

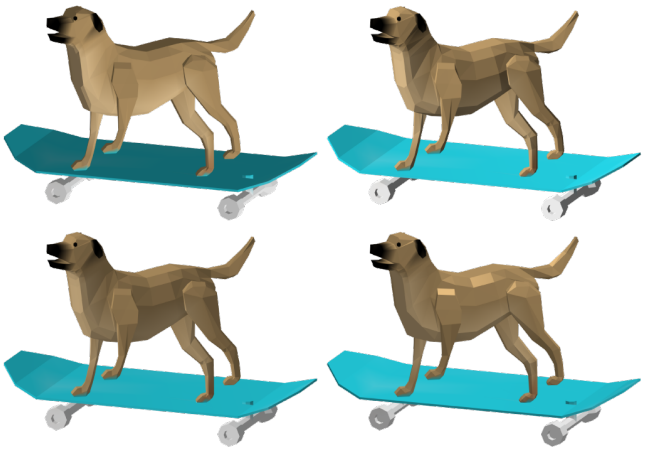


Fig. 5. Impawssible Dog. Meschers support relighting. Here, we show the same mescher rendered with four different lighting conditions. This shows that some lighting conditions create a stronger illusory percept than others.

As in Loop subdivision on standard triangle meshes, we begin by introducing a new vertex at the center-point of a triangle's edge, and reconnect the topology to create a total of 4 new triangles. The x and y positions of the vertices are assigned as expected. Subdividing the relative depth $\boldsymbol{\zeta}$ is possible as we know that the depth increases linearly between two vertices due to our orthographic camera assumption. Since the 4 new triangles are similar to the original triangle with a scaling factor of one half, a subdivided edge's $\boldsymbol{\zeta}$ will equal to the one half of the $\boldsymbol{\zeta}$ of the edge to which it is similar. Subdivision was used to generate virtually all of our results, see Figures 1 4, 6, 3, and 8.

4.3 Heat Diffusion and Geodesics

A mescher has a discrete metric, enabling intrinsic geometry processing despite not being isometrically embeddable in $\mathbb{R}^3$. A classical intrinsic problem is solving the heat equation on the surface of a shape. Starting from some initial condition $\boldsymbol{u}_0 \in \mathbb{R}^{|V|}$, we solve for $\boldsymbol{u}$ the linear system given by

$$(I + t\Delta)\boldsymbol{u} = \boldsymbol{u}_0, \tag{5}$$

where t is the heat diffusion parameter, and Δ the 0-form Laplacian operator from Section 3.1. See Figure 1 for an example of heat diffusion on an impossible object.

Heat diffusion and operations involving the FEM operators in Section 3.2 are sufficient to compute for shortest paths on a mescher using the heat method for distance computation [Crane et al. 2017]. For a short recap of the method, we first initialize $\boldsymbol{u}_0$ to be one at the source vertex and zero elsewhere and compute $\boldsymbol{u}$ for a small t. Next, we compute and then normalize the gradient of the diffused function using the FEM gradient operator: $\boldsymbol{g} = \frac{G\boldsymbol{u}}{\|G\boldsymbol{u}\|_2}$. To finally obtain the distance $\boldsymbol{d}$, we re-integrate the normalized gradient by solving a

Poisson equation, $G^T A G \boldsymbol{d} = -G^T A \boldsymbol{g}$. We demonstrate intrinsic distance computation on impossible objects in Figures 1, 3, 8.

4.4 Smoothing

A common task during modeling is to smooth out the rough edges of a modeled mesh. This can be accomplished via Laplacian smoothing [Desbrun et al. 2023]. On embeddable meshes, Laplacian smoothing applies the heat diffusion operator in (5) to the 3D coordinates of the mesh. In our case, we can apply this strategy to $\boldsymbol{x}$ and $\boldsymbol{y}$, but it is not obvious how to smooth the 1-form $\boldsymbol{\zeta}$.

To complete our smoothing operator, it is helpful to leverage the structure of $\boldsymbol{\zeta}$. Recall from Equation 2 and Section 3.3 that $\boldsymbol{\zeta}$ consists of a harmonic component $\boldsymbol{\omega}$, and a curl-free component, $\mathrm{d}_{01}\boldsymbol{z}$. We know that the harmonic component cannot be further smoothed in this manner, as it is in the null space of the k-form Laplacian. This conveniently leaves only the curl-free component, $\mathrm{d}_{01}\boldsymbol{z}$, which can be obtained via a Hodge decomposition of ζ. Since $\boldsymbol{z}$ is a primal 0-form, we can apply standard Laplacian smoothing to it and then combine it back with ω. In summary, our smoothing operator for $\boldsymbol{\zeta}$ is,

$$\boldsymbol{\zeta}_1 = \boldsymbol{\omega}_0 + \mathrm{d}_{0,1}\left((I + t\Delta)^{-1}\boldsymbol{z} - \boldsymbol{z}\right). \tag{6}$$

Since meschers lack a divergence-free component, this species of smoothing is equivalent to smoothing using the 1-form Hodge Laplacian in Section 3.1. We demonstrate the effect of our smoothing operator with varying degrees of smoothing in Figure 6.

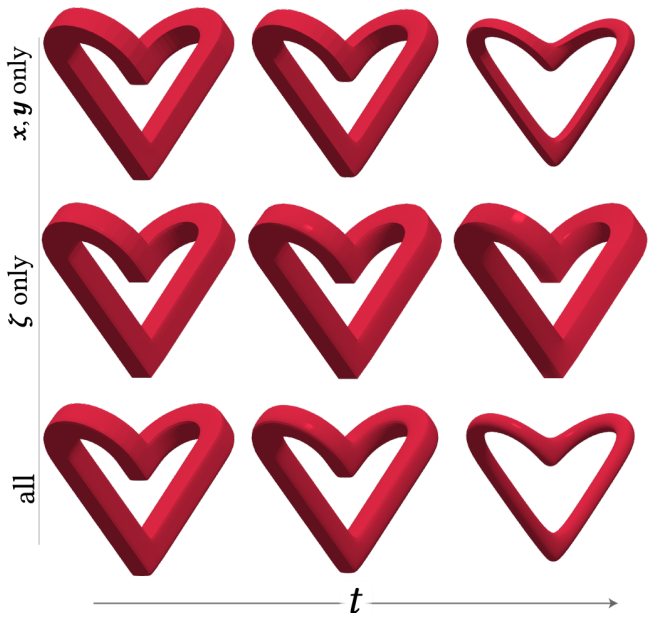


Fig. 6. Heart. We can apply Laplacian smoothing to meschers (Section 4.4). Smoothing can be used for stylization, or to smooth out modeling imperfections. Here, we subdivide and smooth the mescher Heart using various values of the heat diffusion parameter t. We can independently smooth the screen-space components (top row) and depth component (middle row), or smooth both jointly (bottom row).

4.5 Inverse Rendering

Robustly inverse-rendering even standard (embeddable) triangle meshes is an active area of research: the key challenge is that gradients are sparse, only defined on the silhouette of an object [Li et al. 2018]. One way to make inverse rendering more robust is to use Sobolev gradient descent [Nicolet et al. 2021], which effectively smooths out the sparse gradients over the surface. This algorithm only requires a Laplacian—hence, it can be applied to meschers.

Given a ground truth RGB image, we optimize a mescher's geometry so that it matches the target image when passed through the SoftRas differentiable rasterizer [Liu et al. 2019] with fixed lighting. Specifically, we optimize the mean squared error between the rendered image and the target, and use separate σ parameters in SoftRas for optimizing the screen-space (x, y) coordinates and the relative depth ζ. In practice, this means that alternate between descending on $\boldsymbol{x}, \boldsymbol{y}$, and on $\boldsymbol{\zeta}$, rendering images using different σ for each. Following Nicolet et al. [2021], we optimize this energy using Sobolev gradients. Assuming $\boldsymbol{g}$ is the gradient with respect to $\boldsymbol{x}$ and $\boldsymbol{y}$ of our energy, we compute the Sobolev gradient $\hat{\boldsymbol{g}}$ as:

$$\hat{\boldsymbol{g}} = (I + \lambda\Delta)^{-1}\boldsymbol{g}. \tag{7}$$

We use the same diffusion parameter λ as Nicolet et al. [2021].

During inverse rendering, we discard the partial ordering graph data structure and rely on a differentiable global ordering by assigning a scalar depth offset to each triangle. A partial ordering can still be extracted after the optimization by searching for overlaps and comparing triangle scalar depth values.

Figure 7 demonstrates a proof-of-concept of our inverse rendering pipeline. We initialize our shape to a standard (i.e., possible) torus which is then optimized to look more like the impossible triangle. Finally, we can check that we are indeed recovering an impossible object by examining the mescher's harmonic component.

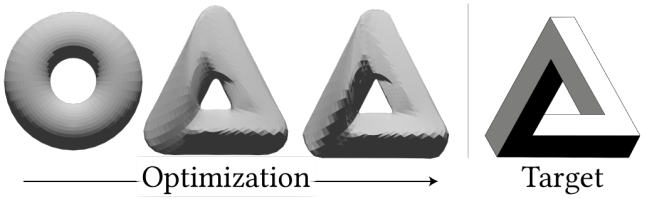


Fig. 7. Penrose triangle. We can use gradient-based inverse rendering to recover the Penrose triangle from a source image (Section 4.5). The resulting mescher really does represent impossible geometry: that is, we can verify that it has a nonzero harmonic component $\boldsymbol{\omega}$.

4.6 Embedding Meschers

We can easily convert meschers into the two existing representations of impossible objects discussed in Section 2.1.

- To recover a *cut* embedding, we can allow a user to specify a cut on the mescher. Then, we integrate the surface around that cut using a simple breadth-first traversal on its edges.
- To recover a *bent* embedding of a mescher, we have to remove the harmonic component from $\boldsymbol{\zeta}$ and then globally integrate the

resulting field. Luckily, that integral is already given to us as part of the Hodge decomposition in the form of z from Equation 2. Therefore, z is a valid choice for the depth component of a mescher.

In this way, meschers are a natural generalization of both of these approaches. In fact, they capture the "best of all worlds": as we show in Figure 8, meschers are the only representation that can support the full set of rendering and geometry processing operations we consider. For example, while all three representations support rendering, the bent representation leads to artifacts when relighting, and both cut and bent representations lead to artifacts when smoothing and when computing geodesic distances.

5 Implementation Details

We implement Meschers primarily using PyTorch [Paszke et al. 2019] and relied on the NetworkX [Hagberg et al. 2008] graph processing library for the depth ordering. For all of our experiments, we used a computer with an Intel i9-13900 CPU, 32 GB of memory, and an Nvidia GeForce RTX 4090. To implement a user interface and rendering capabilities, we relied on the ModernGL [Dombi 2020] and Dear ImGUI [Cornut 2014] libraries. We use PyTorch3D [Ravi et al. 2020] for its implementation of Softras [Liu et al. 2019].

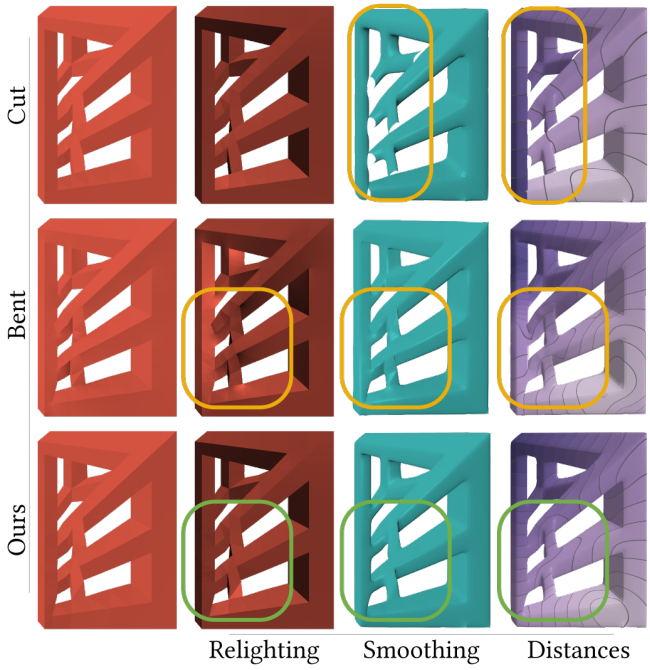
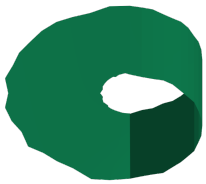


Fig. 8. Mission: Impossible, Mission: Accomplished. Here, we compare our method to the two classes of existing methods for rendering impossible geometry. While all three methods support rendering, only our method supports a full set of rendering and geometry processing operations. The *bent* representation leads to artifacts when relighting, due to the perturbed normals. Both representations lead to artifacts when smoothing and when computing geodesic distances.

6 Discussion, Limitations, and future work

In this paper, we presented a new geometric representation for impossible objects. Our representation, the mescher, is designed around the insight that human visual perception is local rather than global. The mescher thus represents locally-consistent geometry while relaxing the requirement that the geometry be globally-consistent. Conveniently, local consistency is all that is needed for a wide variety of classic geometry processing algorithms. Hence, the mescher allows for not only rendering and relighting, but also operations like subdivision, Laplacian smoothing, heat diffusion, geodesic distance queries, and inverse rendering, in a way that aligns with our perceptual intuitions.



To achieve this, we relied on DEC, which necessitates that its inputs are orientable and manifold. If we wished to represent a non-orientable impossible object, e.g., if we wished to insert the Impossible Lettuce (see inset) into the bagel in Figure 1, we would have to consider alternative formulations.

All of our meschers were modeled in an external tool, and then made impossible using our software (Section 3.3). It remains an open question of what a usable interface for directly modeling meschers is. An alternative to modeling meschers is to inverse render them, as discussed in Section 4.5. While our experiment serves as the, to our knowledge, first proof of concept of rendering a mescher, we trust that there is still room for improvement on this front. An interesting avenue for future work would be to create a mescher modeling interface that relies on inverse rendering, by, e.g., allowing users to paint a harmonic component onto a possible object.

There are many natural steps for improving meschers' renderings. For example, have not experimented with rotating meschers as when we apply a rotation to the differential coordinates, the harmonic component becomes "mixed in" into $\boldsymbol{x}$ and $\boldsymbol{y}$, which would require reprojecting them back onto the space of exact forms. Furthermore, one can imagine combining meschers with previous work to render them with shadows or transparency.

Beyond rendering, our work has only scratched the surface of geometry processing applications enabled by the mescher representation. While we compute gradients on a mescher in Section 4.3, there might be other extrinsic operations one might want to perform, such as, e.g., geometric flows. Similarly, DEC is a first-order discretization of a surface and there might be applications, such as computing occluding contours of a mescher [Capouellez et al. 2023], where a higher order discretization might be useful.

In computer vision, implicit geometry representations (e.g. signed distance functions) have seen great success due to their ability to change topology during optimization, posing the natural question of if it is possible to develop a mescher-like implicit representation.

Lastly, since our representation is inspired by vision science, we can conversely ask whether it would be in turn useful to vision science research. For example, an exciting question one might ask is whether shortest paths computed on a mescher match those that a human would manually trace out.

Perceptually-informed and human-centered models can aid the understanding human vision and the design of computer vision systems, and are central in creativity and art. Through our study of impossible objects, we hope to encourage work not only on realistic graphics models, but also those that become possible in the context of human perception.

Fig. 9. Impossible Geometric Data Processing. Just like in ordinary meshes, meschers can be rendered out with a variety of shading techniques, including both flat (top) and smooth (bottom) shading.

Acknowledgments

We would like to thank Tzu-Mao Li, Jeffrey Price and Seth Riskin for thoughtful conversations during this project.

Ana Dodik acknowledges the generous support of the MIT Presidential Fellowship and the Mathworks Fellowship. Kartik Chandra acknowledges the Hertz Foundation and the NSF GRFP for funding.

Jonathan Ragan-Kelley acknowledges the support of NSF award OAC-2403239.

Joshua Tenenbaum acknowledges the support of Schmidt Sciences AI2050 fellowship and Siegel Family Quest for Intelligence.

The MIT Geometric Data Processing group acknowledges the generous support of Army Research Office grants W911NF2010168 and W911NF2110293, of Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grant CHS-1955697, from the CSAIL Systems that Learn program, from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, from a gift from Adobe Systems, and from a Google Research Scholar award.

The MIT Scene Representation Group was supported by the National Science Foundation under Grant No. 2211259, by the Singapore DSTA under DST00OECI20300823 (New Representations for Vision, 3D Self-Supervised Learning for Label-Efficient Vision), by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) under 140D0423C0075, by the Amazon Science Hub, and by the MIT-Google Program for Computing Innovation.

References

Aseem Agarwala, Maneesh Agrawala, Michael Cohen, David Salesin, and Richard Szeliski. 2006. Photographing long scenes with multi-viewpoint panoramas. In *ACM SIGGRAPH 2006 Papers*. 853–861.

Maneesh Agrawala, Denis Zorin, and Tamara Munzner. 2000. Artistic multiprojection rendering. In *Rendering Techniques 2000: Proceedings of the Eurographics Workshop in Brno, Czech Republic, June 26–28, 2000 11*. Springer, 125–136.

Marc Alexa. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer* 19, 2 (2003), 105–114.

Vlad Alexeev. 2001. *Impossible world — im-possible.info.* [Accessed 16-04-2025].

Irving Biederman. 1985. Human image understanding: Recent research and a theory. *Computer vision, graphics, and image processing* 32, 1 (1985), 29–73.

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2023. *Mixed-Integer Quadrangulation* (1 ed.). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3596711.3596740

Ryan Burgert, Xiang Li, Abe Leite, Kanchana Ranasinghe, and Michael S Ryoo. 2023. Diffusion illusions: Hiding images in plain sight. *arXiv preprint arXiv:2312.03817* (2023).

Ryan Capouellez, Jiacheng Dai, Aaron Hertzmann, and Denis Zorin. 2023. Algebraic Smooth Occluding Contours. In *ACM SIGGRAPH 2023 Conference Proceedings* (Los Angeles, CA, USA) *(Siggraph '23)*. Association for Computing Machinery, New York, NY, USA, Article 39, 10 pages. doi:10.1145/3588432.3591547

Patrick Cavanagh. 2005. The artist as neuroscientist. *Nature* 434, 7031 (2005), 301–307.

Kartik Chandra, Tzu-Mao Li, Joshua Tenenbaum, and Jonathan Ragan-Kelley. 2022. Designing perceptual puzzles by differentiating probabilistic programs. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.

Ming-Te Chi, Tong-Yee Lee, Yingge Qu, and Tien-Tsin Wong. 2008. Self-Animating Images: Illusory Motion Using Repeated Asymmetric Patterns. *ACM Trans. Graph.* 27, 3 (aug 2008), 1–8. doi:10.1145/1360612.1360661

Hung-Kuo Chu, Wei-Hsin Hsu, Niloy J Mitra, Daniel Cohen-Or, Tien-Tsin Wong, and Tong-Yee Lee. 2010. Camouflage images. *ACM Trans. Graph.* 29, 4 (2010), 51–1. https://dl.acm.org/doi/abs/10.1145/1833349.1778788

Omar Cornut. 2014. https://github.com/ocornut/imgui.

Keenan Crane, Fernando de Goes, Mathieu Desbrun, and Peter Schröder. 2013. Digital geometry processing with discrete exterior calculus. In *ACM SIGGRAPH 2013 Courses* (Anaheim, California) *(Siggraph '13)*. Association for Computing Machinery, New York, NY, USA, Article 7, 126 pages. doi:10.1145/2504435.2504442

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2017. The Heat Method for Distance Computation. *Commun. ACM* 60, 11 (Oct. 2017), 90–99. doi:10.1145/3131280

Fernando de Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. 2016. Subdivision exterior calculus for geometry processing. *ACM Trans. Graph.* 35, 4, Article 133 (July 2016), 11 pages. doi:10.1145/2897824.2925880

Mathieu Desbrun, Eva Kanso, and Yiying Tong. 2006. Discrete differential forms for computational modeling. In *ACM SIGGRAPH 2006 Courses* (Boston, Massachusetts) *(Siggraph '06)*. Association for Computing Machinery, New York, NY, USA, 39–54. doi:10.1145/1185657.1185665

Mathieu Desbrun, Melvin Leok, and Jerrold E. Marsden. 2005. Discrete Poincaré Lemma. *Applied Numerical Mathematics* 53, 2-4 (May 2005), 231–248. doi:10.1016/j.apnum.2004.09.035

Mathieu Desbrun, Mark Meyer, Peter Schroder, and Alan H. Barr. 2023. *Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow* (1 ed.). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3596711.3596729

Szabolcs Dombi. 2020. ModernGL, high performance python bindings for OpenGL 3.3+. https://github.com/moderngl/moderngl.

Stephen W Draper. 1978. The Penrose Triangle and a Family of Related Figures. *Perception* 7, 3 (1978), 283–296. doi:10.1068/p070283 arXiv:https://doi.org/10.1068/p070283 Pmid: 693228.

Gershon Elber. 2011. Modeling (seemingly) impossible models. *Computers & Graphics* 35, 3 (2011), 632–638.

Gershon Elber. 2012. *Escher for Real — gershonelber.org.* [Accessed 16-04-2025].

B. Ernst. 1986. *Adventures with Impossible Figures.* Tarquin. https://books.google.com/books?id=HU_vAAAAMAAJ

M.C. Escher. 1961. *The Graphic Work of M.C. Escher.* Oldbourne. https://books.google.com/books?id=1T5gwgEACAAJ

William T Freeman, Edward H Adelson, and David J Heeger. 1991. Motion without movement. *ACM Siggraph Computer Graphics* 25, 4 (1991), 27–30.

Erez Freud, Tzvi Ganel, and Galia Avidan. 2013. Representation of possible and impossible objects in the human visual cortex: Evidence from fMRI adaptation. *NeuroImage* 64 (2013), 685–692. doi:10.1016/j.neuroimage.2012.08.070

Erez Freud, Bat-Sheva Hadad, Galia Avidan, and Tzvi Ganel. 2015. Evidence for similar early but not late representation of possible and impossible objects. *Frontiers in*

Psychology 6 (2015). doi:10.3389/fpsyg.2015.00094

Matías Gárate. 2020. *Art Spotlight: Impossible Penrose Snake — sketchfab.com.* [Accessed 16-04-2025].

Matías Gárate. 2023. *The ins and outs of modelling impossible figures. — Blender Conference 2023 — conference.blender.org.* [Accessed 16-04-2025].

Daniel Geng, Inbum Park, and Andrew Owens. 2023. Visual Anagrams: Generating Multi-View Optical Illusions with Diffusion Models. *arXiv preprint arXiv:2311.17919* (2023).

Daniel Geng, Inbum Park, and Andrew Owens. 2024. Factorized Diffusion: Perceptual Illusions by Noise Decomposition. *arXiv preprint arXiv:2404.11615* (2024).

R. L. (Richard Langton) Gregory. 1970. *The intelligent eye, by R. L. Gregory.* Weidenfeld & Nicolson, London.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11–15. http://conference.scipy.org/proceedings/SciPy2008/paper_2/

Dietmar Heinke, Peter Wachman, Wieske van Zoest, and E. Charles Leek. 2021. A failure to learn object shape geometry: Implications for convolutional neural networks as plausible models of biological vision. *Vision Research* 189 (2021), 81–92. doi:10.1016/j.visres.2021.09.004

Aaron Hertzmann. 2024. Toward a theory of perspective perception in pictures. *Journal of Vision* 24, 4 (04 2024), 23–23. doi:10.1167/jov.24.4.23 arXiv:https://arvojournals.org/arvo/content_public/journal/jov/938670/i1534-7362-24-4-23_1713961163.65834.pdf

Anil N. Hirani. 2003. *Discrete Exterior Calculus.* Ph. D. Dissertation. California Institute of Technology. http://resolver.caltech.edu/CaltechETD:etd-05202003-095403

William Hogarth. 1754. Satire on False Perspective. https://publicdomainreview.org/collection/william-hogarth-satire-on-false-perspective/ Frontispiece to Joshua Kirby's "Dr. Brook Taylor's Method of Perspective Made Easy".

Tiffany Inglis. 2014. Constructing Drawings of Impossible Figures with Axonometric Blocks and Pseudo-3D Manipulations. In *Proceedings of Bridges 2014: Mathematics, Music, Art, Architecture, Culture*, Gary Greenfield, George Hart, and Reza Sarhangi (Eds.). Tessellations Publishing, Phoenix, Arizona, 159–166. http://archive.bridgesmathart.org/2014/bridges2014-159.html

Olga A Karpenko and John F Hughes. 2006. Smoothsketch: 3d free-form shapes from complex sketches. In *ACM SIGGRAPH 2006 Papers*. 589–598.

Chih W Khoh and Peter Kovesi. 1999. Animating impossible objects.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2015. Stripe Patterns on Surfaces. *ACM Trans. Graph.* 34 (2015). Issue 4.

Jan J Koenderink. 1998. Pictorial relief. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 356, 1740 (1998), 1071–1086.

Felix Kälberer, Matthias Nieser, and Konrad Polthier. 2007. QuadCover - Surface Parameterization using Branched Coverings. *Computer Graphics Forum* 26, 3 (2007), 375–384. doi:10.1111/j.1467-8659.2007.01060.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2007.01060.x

Chi-Fu William Lai, Sai-Kit Yeung, Xiaoqi Yan, Chi-Wing Fu, and Chi-Keung Tang. 2015. 3D navigation on impossible figures via dynamically reconfigurable maze. *IEEE transactions on visualization and computer graphics* 22, 10 (2015), 2275–2288.

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–11.

Yuanbo Li, Tianyi Ma, Zaineb Aljumayaat, and Daniel Ritchie. 2024. PossibleImpossibles: Exploratory Procedural Design of Impossible Structures. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15052.

Paul Linton, Michael J Morgan, Jenny CA Read, Dhanraj Vishwanath, Sarah H Creem-Regehr, and Fulvio Domini. 2023. New approaches to 3D vision. 20210443 pages.

Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossi, and Hans-Peter Seidel. 2004. Differential coordinates for interactive mesh editing. In *Proceedings Shape Modeling Applications, 2004.* Ieee, 181–190.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2019).

Margaret S Livingstone. 2022. *Vision and art (updated and expanded edition).* Abrams.

Charles T. Loop. 1987. Smooth Subdivision Surfaces Based on Triangles. https://api.semanticscholar.org/CorpusID:116150707

Li-Qian Ma, Kun Xu, Tien-Tsin Wong, Bi-Ye Jiang, and Shi-Min Hu. 2013. Change blindness images. *IEEE transactions on visualization and computer graphics* 19, 11 (2013), 1808–1819. https://cg.cs.tsinghua.edu.cn/papers/TVCG-2013-changeblindness.pdf

David Marr. 1982. *Vision: A computational investigation into the human representation and processing of visual information.* MIT press.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large steps in inverse rendering of geometry. *ACM Transactions on Graphics (TOG)* 40, 6 (2021), 1–13.

Yu Okano, Shogo Fukushima, Masahiro Furukawa, and Hiroyuki Kajimoto. 2010. Embedded Motion: Generating the Perception of Motion in Peripheral Vision. In *ACM SIGGRAPH ASIA 2010 Posters* (Seoul, Republic of Korea) *(Sa '10)*. Association for Computing Machinery, New York, NY, USA, Article 41, 1 pages. doi:10.1145/1900354.1900400

Aude Oliva, Antonio Torralba, and Philippe G Schyns. 2006. Hybrid images. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 527–532. https://stanford.edu/class/ee367/reading/OlivaTorralb_Hybrid_Siggraph06.pdf

Shigeru Owada and Jun Fujiki. 2008. Dynafusion: A modeling system for interactive impossible objects. In *Proceedings of the 6th international symposium on Non-photorealistic animation and rendering*. 65–68.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 8024–8035. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Lionel S Penrose and Roger Penrose. 1958. Impossible objects: a special type of visual illusion. *British Journal of Psychology* (1958).

Roger Penrose. 1993. On the Cohomology of Impossible Figures. *Leonardo* 25 (1993), 245–247. https://api.semanticscholar.org/CorpusID:125905129

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. 2020. Accelerating 3D Deep Learning with PyTorch3D. *arXiv:2007.08501* (2020).

Oscar Reutersvärd. 1934. First Impossible Triangle. https://sis.modernamuseet.se/objects/28602/ Earliest known impossible figure, later reproduced as lithographs.

Augusto Roman, Gaurav Garg, and Marc Levoy. 2004. Interactive design of multi-perspective images for visualizing urban landscapes. In *IEEE visualization 2004.* Ieee, 537–544.

Javier Sánchez-Reyes and Jesús M Chacón. 2020. How to make impossible objects possible: Anamorphic deformation of textured NURBS. *Computer Aided Geometric Design* 78 (2020), 101826.

Guillermo Savransky, Dan Dimerman, and Craig Gotsman. 1999. Modeling and Rendering Escher-Like Impossible Scenes. In *Computer Graphics Forum*, Vol. 18. Wiley Online Library, 173–179.

Donald H Schuster. 1964. A new ambiguous figure: A threestick clevis. *The American Journal of Psychology* (1964).

Herbert A. Simon. 1967. An Information-processing Explanation Of Some Perceptual Phenomena. *British Journal of Psychology* 58, 1-2 (1967), 1–12. doi:10.1111/j.2044-8295.1967.tb01051.x arXiv:https://bpspsychub.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2044-8295.1967.tb01051.x

Olga Sorkine. 2006. Differential representations for mesh processing. In *Computer Graphics Forum*, Vol. 25. Wiley Online Library, 789–807.

Kokichi Sugihara. 1997. Three-dimensional realization of anomalous pictures—An application of picture interpretation theory to toy design. *Pattern Recognition* 30, 7 (1997), 1061–1067.

Benjamin Adam Taylor. 2020. *Modeling and Rendering Three-Dimensional Impossible Objects.* Bangor University (United Kingdom).

E. Térouanne. 1984. Impossible Objects and Inconsistent Interpretations. In *Trends in Mathematical Psychology*, E. Degreef and J. Van Buggenhaut (Eds.). Advances in Psychology, Vol. 20. North-Holland, 105–120. doi:10.1016/S0166-4115(08)62082-8

Xi Wang, Zoya Bylinskii, Aaron Hertzmann, and Robert Pepperell. 2020. Toward quantifying ambiguities in artistic images. *ACM Transactions on Applied Perception (TAP)* 17, 4 (2020), 1–10.

Ethan Weber*, Riley Peterlinz*, Rohan Mathur, Frederik Warburg, Alexei A. Efros, and Angjoo Kanazawa. 2024. Toon3D: Seeing Cartoons from a New Perspective. In *arXiv*.

Tai-Pang Wu, Chi-Wing Fu, Sai-Kit Yeung, Jiaya Jia, and Chi-Keung Tang. 2010. Modeling and rendering of impossible figures. *ACM Transactions on Graphics (ToG)* 29, 2 (2010), 1–15.