Gap-based navigation reinterpreted in spaces of half-tangents

Chengyuan Qian and Dylan A. Shell

Dept. of Computer Science and Engineering, Texas A&M University.
{cyqian,dshell}@tamu.edu

Abstract. Efficient navigation in planar environments is possible with severely limited data: cues describing depth discontinuities, as reported by so-called 'gap' sensors, provide sufficient information to generate distance-optimal motions. Gap sensors are suggestive of general lessons about well-chosen robot abstractions: they bridge geometric and combinatorial realms, enabling computational decision making that is both parsimonious and informed by perceptible features of the environment. Yet, the elegance of the gap navigation tree data structure (and allied algorithms) obscures much of how this bridging is actually achieved. The present paper describes new representations obtained via a theoretical investigation that aims to elucidate how and why gap sensing works. We study a space of visible half-tangents, revealing essential aspects of gaps and their dynamics as the robot moves. To examine the combinatorial structure of this evolution, we then connect the continuous representation to a discrete one, analyzing the orders in which events may be experienced. This involves a temporal relaxation, with some aspects of the order's "lack of totalness" reflecting inherent ambiguities. We explore how arguments of correctness for the gap navigation algorithms are actually made on this relaxed (i.e., larger set) of robot experiences.

1 Introduction

An influential body of research asks not merely how to build and program robots to accomplish tasks, but how to do so with constrained resources, usually in terms of limited sensors, actuators, memory, etc. While being frugal certainly has practical implications —e.g, in manufacturing cheaper robots, or limiting knowledge of private/sensitive data— the study of minimal robots helps establish a more fundamental understanding of what is necessary to perform some class of activity [1–3].

The Gap Navigation Tree (GNT) algorithm and data structure [11] (see also [4, §12.3.4]) achieves a feat which, at first blush, does not seem possible: it enables a mobile robot to reach goals via distance-optimal paths, in an initially unknown planar environment, despite the robot never having access to metric data for positions, distances, or angles. The GNT (along with its follow-ups [5–7] and extensions to different tasks [10]) make use of qualitative changes of visibility that manifest as changes to sensed gaps, triggered through motions based on the chasing of gaps. In its essence, the GNT data structure maintains an ego-centric expression of combinatorial aspects of the robot's experience of its world. The representation is concise and that conciseness stems from how it exploits the interplay of sensing with action, mediated by the environment imposing structure on the evolution of regions of visibility. It is extraordinarily

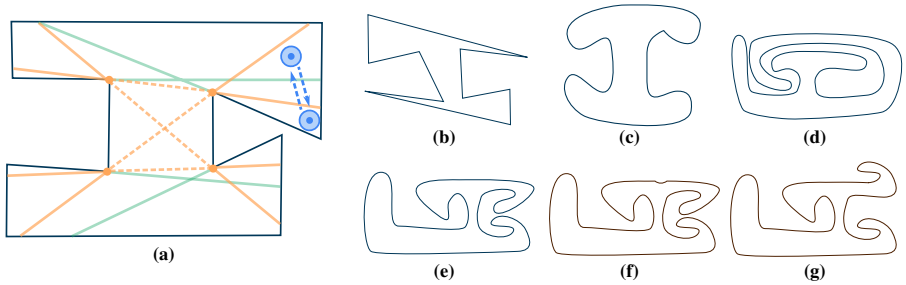

Fig. 1: A family of five different environments (a)–(e) that produce identical GNTs (based on [4, Fig. 12.37, pg. 681], with minor corrections). These environments illustrate how polygonal vertices may be rounded or regions stretched/warped without altering navigational decisions in terms of gap chasing. Yet, small changes, exemplified by just altering (e)→(f), can violate this equivalence. There are also spiral-like stretching and winding transformations for which sidedness is significant, so (c)→(e) maintains GNT-invariance, while (c)→(g) breaks it. Additionally, in (a), event lines are plotted: bitangent lines (orange dashed), bitangent complement lines (orange solid), and inflection lines (green solid). A robot (blue disk) moving along the dashed arrows observes all four gap event types: GAPSPLIT, GAPDISAPPEAR, GAPAPPEAR, and GAPMERGE.

elegant. The present paper develops a new standpoint from which to study existing gap-based navigation strategies, and offers new insights on how and why they work: several aspects that were previously only tacit are seen anew and obtain a fresh appreciation.

Recently, the present authors [9] studied an extended family of gap sensors, carefully scrutinizing the specific model underlying the GNT. Doing so highlighted some of the assumptions and cleverness involved in the GNT; it also shows that, common with many elegant things, it is also rather delicate. Part of the approach in that work was to rigorously examine how gap information is tracked across time. Indeed, it illustrates what *must* be tracked and explicates how cyclic orderings (as used in [11]) suffice to establish this, provided certain conditions are met. The fundamental issue can be said to be one of data association: recognizing when two gaps sensed at different times ought or ought not to be treated as distinct. One wishes to say, in plain language, the sensor has detected the "same" gap. In order to argue that a method (either that of the original GNT [11] using cyclic orderings, or the alternative in [9], for instance) succeeds, one might appeal to a diagram and make a geometrical argument that under certain motions the association will be correctly recovered. This appeal to geometry feels unsatisfactory because it violates the spirit of the thing — the GNT and its ilk operate on data that are not metrical. Indeed, one key feature is that a wide range of vastly different environments distill down to equivalent GNTs (see Fig. 1); big differences in appearance can turn out to be superficial differences when considered in terms of navigational behavior.

This paper introduces *Visible Gap Manifolds* (VGMs) to serve as an appropriate basis for grounding the data association relations that underpin gap navigation. The VGM is a non-metrical intermediate representation somewhere between a full geometrical world and the sorts of topological-combinatorial descriptions that a gap-sensing robot maintains. Each VGM discards some spatial aspects, like irrelevant distances, but retains parts that are crucial to the execution of a gap sensing robot, namely, spells of continuous experience, punctuated by discontinuous events.

As described in [11], the GNT algorithm operates in continuous spatial regions and continuous time. Continuity is important to the whole underlying model. A gap is defined as a depth discontinuity, i.e., a violation of spatial continuity. And gap chasing, the core motion primitive, occurs in continuous time. The key to the data association mentioned above is a relation that represents spatial continuity across time (dubbed a coherence relation in [9]). Repetition of the concept of continuity, but also the fact some deformations of environments (as seen in Fig. 1a–1e) are indistinguishable to a gap-sensing robot, hint toward the importance of topological ideas. Perhaps what is most interesting, however, is what is *not* captured topologically, including: aspects of visibility (cf. Fig. 1e *vs* 1f or 1g), a certain directed-ness tied to motions, and the interplay of multiple concurrent points on the structure. We explore these aspects by introducing a graph abstraction of the VGMs and their interrelations, revealing additional facts about the temporal experience of a gap sensing robot.

Overview & Contributions This paper assembles scaffolding to help formalize arguments about the correctness of gap-based navigation algorithms. It employs multiple descriptions of the same phenomena, but at differing levels of detail and mediates their interrelations carefully. The core conceptual contribution is to model perceptual experience, rather than state of the system, by focusing on individual gaps as they exist and persist across time. This scheme is appropriate for the class of algorithms studied because they act upon a single primary gap, and reason (at specific junctures) about relationships between that gap and others. The paper's methodological contribution is in the gradual, systematic shift from an objective or extrinsic, bird's-eye view to that of the robot's direct experience by discarding information that is inaccessible to the robot.

2 Preliminaries

We denote the unit circle by $\mathbb{S}^1$ and will use coordinates $[0, 2\pi)$. A change is said to be in the increasing direction when the parameter increases, and in the decreasing direction in reverse; the identification $2\pi \sim 0$ is handled in the obvious way. We will write $\lim_{\eta \uparrow \theta}$ for the limit taken in $\mathbb{S}^1$ approaching θ in the increasing direction; and analogously $\lim_{\eta \downarrow \theta}$ for decreasing. For $\theta_1, \theta_2 \in \mathbb{S}^1$, we will write $(\!|\theta_1, \theta_2|\!)$ for the interval from θ_1 to θ_2 along the increasing direction; bracket '(|' indicates that the left endpoint of an interval that can be either '(' open or '[' closed, and '|)' is analogous for the right endpoint.

The robot navigates in a planar environment $\mathcal{E} \subset \mathbb{R}^2$. We require $\mathcal{E}$ to be bounded, closed, and path-connected. Given a vector in $\mathbb{R}^2$, we will talk of the left- *vs* right-hand side of the vector, as picking any exterior standpoint establishes the convention.

2.1 Environment and Boundary Curves

The boundary of the environment $\partial\mathcal{E}$ comprises finitely many curves that are closed, simple, piecewise differentiable, and do not intersect with each other. These boundary curves each split the plane into two connected regions, one being accessible to the robot, while the other representing an obstacle. The union of accessible regions of all curves forms $\mathrm{int}(\mathcal{E})$, the interior of the environment. A closed curve c can be parameterized by a continuous function $\varphi_c : \mathbb{S}^1 \to c$ such that as $t \in \mathbb{S}^1$ scans from 0 to 2π, $\varphi_c(t)$ traverses c with the obstacle area on the left-hand side of the traversing direction. We refer to such

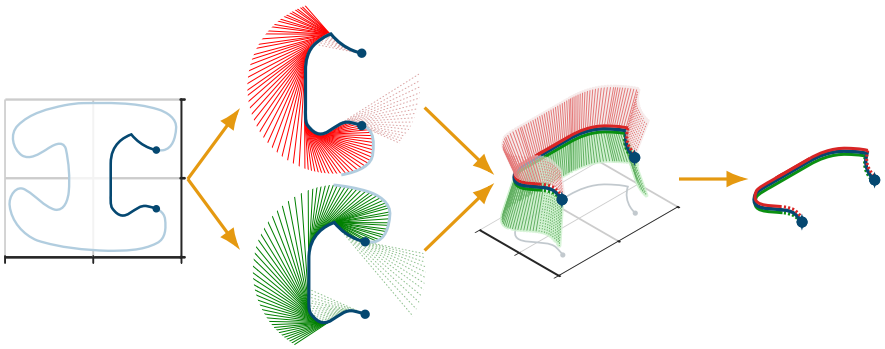

Fig. 2: The environment's boundary consists of curves; an observer on a half-tangent ray senses these as range discontinuities, or gaps. Using a snippet of the boundary of Fig. 1c shown in blue, we illustrate this construction of the topological space formed by these rays. Two distinct spaces arise, and they remain separate after collapsing the distance along each ray — distance being imperceptible to the robot. (Under an anti-clockwise parameterization, red denotes right rays, since the obstacle lies to the right when viewed along the ray, and green denotes left rays.)

a φ_c as a *left-hand parameterization* of curve c. (Note that the outer boundary and inner obstacles, if any, 'curl' in opposite directions.) Further, the following describe (left *vs* right) directional derivatives of the curve:

$$\hat{\nabla}\varphi_c(t) := \lim_{\tau \downarrow t} \frac{\varphi_c(\tau)-\varphi_c(t)}{\tau - t},\ \hat{\hat{\nabla}}\varphi_c(t) := \frac{\hat{\nabla}\varphi_c(t)}{\left\|\hat{\nabla}\varphi_c(t)\right\|_2};\quad \hat{\nabla}\varphi_c(t) := \lim_{\tau \uparrow t} \frac{\varphi_c(\tau)-\varphi_c(t)}{\tau - t},\ \hat{\hat{\nabla}}\varphi_c(t) := \frac{\hat{\nabla}\varphi_c(t)}{\left\|\hat{\nabla}\varphi_c(t)\right\|_2}. \quad (1)$$

Because $\hat{\hat{\nabla}}\varphi_c(t)$ and $\hat{\hat{\nabla}}\varphi_c(t)$ are unit vectors, by considering their bearing angle from some fixed direction, we can view them as elements of $\mathbb{S}^1$.

Before giving two technical regularity conditions on the environment's boundary, we first define some nomenclature. Let c be a boundary curve, φ_c be a left-hand parameterization of c, and $\varphi_c(t)$ be a differentiable point. We refer to $\varphi_c(t)$ as a differentiable inflection point if $\forall \varepsilon > 0$, $\exists \delta_1, \delta_2 \in (0, \varepsilon)$ such that $\hat{\hat{\nabla}}\varphi_c(t+\delta_1)$ and $\hat{\hat{\nabla}}\varphi_c(t-\delta_2)$ point to the same side of $\hat{\hat{\nabla}}\varphi_c(t)$. The end-points of straight line segments form non-differentiable inflection points if prolonging the line segment will separate a neighborhood of both adjoining curves. By just inflection point we mean either of these cases.

Assumption 1. *We require that for each inflection point $\varphi_c(t)$ on $\partial\mathcal{E}$, there exists an $\eta > 0$ such that $\varphi_c(\tau)$ is not an inflection point for all $\tau \in (t-\eta, t+\eta) \setminus \{t\}$.*

Assumption 2 (General position)**.** *No three points on the boundary $\partial\mathcal{E}$ are tangent to the same line ℓ, excluding when the three share a polygonal edge.*

2.2 Background: Recap on Gap sensing and Gap Events

Gap sensing is a minimal-information approach where a robot detects only depth discontinuities in the environment boundary, without measuring distances or angles. Gaps admit a visibility-based interpretation: each gap indicates the boundary of an invisible region. As the robot moves, the number of gaps changes through occurrences that we call *gap events*. Each gap event represents a fundamental transition in the visibility structure and thus conveys critical information about the geometry of the environment.

Gap events fall into four distinct types: GAPAPPEAR, GAPDISAPPEAR, GAPSPLIT, and GAPMERGE, as depicted in Fig. 1a. The dashed orange line represents a bitangent with tangent points marked by orange dots, while the solid orange lines show the corresponding bitangent complement lines. When the robot (shown as a blue disk) crosses a bitangent complement line upward, a GAPSPLIT occurs: the invisible region on the left divides in two. Crossing back in the opposite direction causes a GAPMERGE, where the two regions coalesce. The solid green line marks an inflection line. After the split, if the robot continues upward and crosses this inflection line, a GAPDISAPPEAR occurs: the more distant invisible region disappears. Crossing back produces a GAPAPPEAR, where the vanished invisible region reappears.

3 Topological Space of Gaps

We begin by constructing an abstraction of the spaces wherein perceived gaps evolve.

3.1 Definition and Basic Topology

Definition 1 (Gaps and Rays). *Let $c \subseteq \partial\mathcal{E}$ be a boundary curve, and φ_c be a left-hand parameterization of c. We define a* left ray *as the tangent ray originating from a boundary point $\varphi_c(t)$ and extending in a direction from $[\hat{\nabla}\varphi_c(t), -\hat{\nabla}\varphi_c(t)]$, and a* right ray *is a tangent ray extending in a direction from $[-\hat{\nabla}\varphi_c(t), \hat{\nabla}\varphi_c(t)]$. A* gap *refers to either a left or right ray.*

More intuitively, a gap originates from a boundary point and extends in the gradient direction at differentiable points, or in a subgradient direction at non-differentiable points. Gaps are closely related to tangency, representing half-tangent lines to the boundary; we call the point on the curve where the ray emanates, its origin. Fig. 2 shows some examples of the idea, explaining how the left *vs* right label depends on vantage point.

We collect the left and right rays into mutually disconnected spaces, which is easiest to visualize by parametrizing these as tuples $(p, \theta) \in \mathbb{R}^2 \times \mathbb{S}^1$, where $p = \varphi_c(t)$ is the origin and θ is the direction of the ray. Note the absence of the distance along the ray, which would be required to make a spatial correspondence, but that a robot equipped with only a gap sensor is unable to sense. For each closed curve making up the environment boundary, we obtain left and right submanifolds in $\mathbb{R}^2 \times \mathbb{S}^1$, with the collection of all these denoted as $\mathcal{M}(\mathcal{E})$. Fig. 3 illustrates an obstacle and the curve corresponding to all its right rays; the curve for its left rays is similar but shifted by π along the vertical axis representing $\mathbb{S}^1$.

Under the parameterization described, polygonal edges have fixed angles and appear as orange horizontal segments in Fig. 3. Left rays with origins on the same polygonal edge all point one way; the right rays point in the opposite direction. A gap-sensing robot on a left ray would be on the whole set of left rays simultaneously. The robot can not discriminate any pair of left rays emanating from the same polygonal edge as it only perceives spatial discontinuities. These rays (and, of course similarly for the right rays) should be treated identically, thus, we define equivalence relation $\sim_{\mathrm{p}}$ as: $(p_1, \theta) \sim_{\mathrm{p}} (p_2, \theta) \iff p_1$ and p_2 are on same polygonal edge. In what follows, we will quotient away the polygonal edges. Less informally: the construction gives a topological space by applying that operation to the subspace topology of the product topology of $\mathbb{R}^2 \times \mathbb{S}^1$.

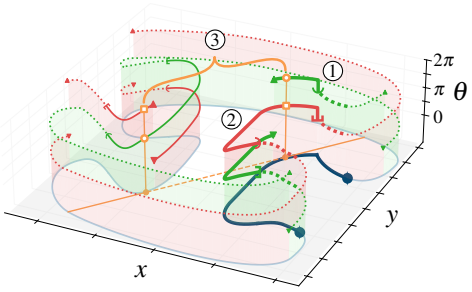

Fig. 3: The (p,θ) curves of tangent rays in the environment from Fig. 1c. The environmental boundary lies in the bottom X–Y plane. All right and left gap rays form two loops, plotted in red and green respectively. The curtain illustrates how the gap ray manifolds correspond to the boundary curve. The curves drawn more thickly above the dark blue boundary correspond to those gap rays that were illustrated Fig. 2. Triangles mark the angle identification at 0 and 2π, and mean that the curves connect at those points. On these sharing the same the same non-differentiable origin p form vertical lines (segments near ①), while rays whose origins lie on a common polygonal edge form horizontal lines (segments near ②) and are identified under $\sim_{\mathrm{p}}$. The orange dots under ③ mark two pairs of bitangent gaps, corresponding to the bitangent line (orange dashed) and the two bitangent complement lines (orange solid) in the environment. Each pair shares the same θ, with the circle representing the proximal gap and the square representing the distal gap.

Some later discussion will require that we distinguish between a gap parameterization as a point in $\mathbb{R}^2 \times \mathbb{S}^1$ and a ray as a geometric subset of $\mathbb{R}^2$. When the distinction is required, we denote the parameter by (p,θ) and the geometric ray by $\overrightarrow{(p,\theta)}$.

3.2 Visible Gaps

The gaps in Definition 1, derived from boundary points and tangents, do not yet capture gaps corresponding to gap sensing. Perceived depth discontinuities require that we account for visibility. Obstacles in $\mathcal{E}$ not only restrict robot motion but also occlude line of sight: for a robot at some position $q \in \overrightarrow{(p,\theta)}$, if an obstacle obstructs the line segment between q and p, the gap sensor cannot report that gap. To incorporate visibility constraints, we define visible gap rays as follows

Definition 2 (Visible gap). *Let $\overline{(a,b)}$ be the open line segment between points a and b, and define* $\textsc{Trunc}[\overrightarrow{(p,\theta)}] = \left\{ s \in \overrightarrow{(p,\theta)} \middle| \overline{(p,s)} \subset \mathrm{int}(\mathcal{E}) \right\}$*, namely a truncation operator. A gap (p,θ) is a* visible gap *if* $\textsc{Trunc}[\overrightarrow{(p,\theta)}] \neq \varnothing$.

For a visible gap (p,θ), since its gap ray enters $\mathrm{int}(\mathcal{E})$ immediately after point p, there must be a small area adjacent to p in the obstacle area that forms a convex set lying entirely on one of the two sides of the gap ray $\overrightarrow{(p,\theta)}$. For a left gap, from the robot's point of view, the obstacle area near p will be on the left-hand side of the gap ray. A right gap is named for a similar reason. The set of all visible gaps in environment $\mathcal{E}$, denoted as $\mathcal{M}_v(\mathcal{E})$, forms a subset of $\mathcal{M}(\mathcal{E})/\sim_{\mathrm{p}}$. Returning to the visuals in Fig. 3, imagine the orange lines contracted to a point, and dashed blue curve deleted.

As $\mathcal{M}_v(\mathcal{E})$ excludes from $\mathcal{M}(\mathcal{E})/\sim_{\mathrm{p}}$ those gaps which are not visible, the loops formed by gaps (p,θ) with p on the same boundary curve c may break into multiple pieces. We term each connected component a Visible Gap Manifold (VGM); the collection of VGMs partition $\mathcal{M}_v(\mathcal{E})$. Each component consists of either all left or all right visible gaps, exclusively. When treating an instance of the former and needing to

emphasize that fact, we will refer to it as a Left Visible Gap Manifold (L-VGM). And analogously for a Right Visible Gap Manifold (R-VGM).

Each VGM, being a portion of a curve, is a 1-manifold that can be traversed in two different directions:

Lemma 1. *Let* $\mathrm{M} \subseteq \mathcal{M}_v(\mathcal{E})$ *be a VGM. Tracing a curve on* M*, the* θ *parameter of the gap* (p,θ) *varies continuously. Traversing without reversing direction means* θ *varies strictly monotonically (strictly increasing or strictly decreasing).*

The proof sketch is deferred to Appendix A. This lemma allows us to make the following definition:

Definition 3 (Revealing and concealing direction). *Let* $\mathrm{M} \subseteq \mathcal{M}_v(\mathcal{E})$ *be a VGM; when a point* $(p,\theta) \in \mathrm{M}$ *is traversing* M*, we say it moves in the revealing direction if* M *is an L-VGM and* θ *is increasing, or if* M *is an R-VGM and* θ *is decreasing. The opposite direction on each manifold is referred to the concealing direction.*

These names are not merely added variety over-and-above left/right directions, but they are an intrinsic (versus extrinsic) description of direction: indeed, Definition 3 holds, for example, even if you pick a different clockwise/anti-clockwise convention. Geometrically, as a gap traverses M in the revealing direction, the gap ray rotates anti-clockwise if the gap is a left gap, and clockwise if it is a right gap—using the usual convention, of course. In both cases, this motion tends to reduce the invisible area behind the gap, hence referring to it as revealing; while the reverse operation tends to enlarge the invisible area. It is also not a coincidence that these names move us closer to semantically significant nouns. These are directions that connect with the experience (and the, fundamentally, local operations) of a robot navigating via gap chasing.

Using this, we can now describe a property of all VGMs.

Lemma 2. *Let* $\mathrm{M} \subseteq \mathcal{M}_v(\mathcal{E})$ *be a VGM with two endpoints, the endpoint approached by traversing* M *in the revealing direction is always closed, while the other endpoint is open and, moreover, it does not appear in* $\mathcal{M}_v(\mathcal{E})$*.*

The proof sketch is deferred to Appendix B.

3.3 Events and VGM intervals

Within $\mathcal{M}_v(\mathcal{E})$, we give prominence to those specific pairs of gaps whose gap rays overlap. If (p_1,θ) and (p_2,θ) are such a pair, the line on which both $\overrightarrow{(p_1,\theta)}$ and $\overrightarrow{(p_2,\theta)}$ lie is a bitangent line of the environment $\mathcal{E}$, tangent to the boundary points p_1 and p_2. We formally define these pairs as follows:

Definition 4 (Bitangent gap pairs). *Let* $(p_1,\theta),(p_2,\theta) \in \mathcal{M}_v(\mathcal{E})$ *be two distinct gaps; then,* $((p_1,\theta),(p_2,\theta))$ *is a* bitangent gap pair *if* $\overrightarrow{(p_1,\theta)} \subset \overrightarrow{(p_2,\theta)}$.

The definition is not symmetric: we name the first gap in the bitangent pair the *proximal gap* and the second the *distal gap*. In Fig. 3, the proximal gap is marked with a solid red star and the distal gap as the empty red star. The geometry shows that there is a bijective correspondence between points on the interior of the L-VGM and the interior of the R-VGM associated with each curve c. The proximal gap on the L-VGM corresponds with the distal gap on the R-VGM, and vice versa.

Definition 5 (Event points, lines). *A gap* $(p,\theta) \in \mathcal{M}_v(\mathcal{E})$ *is an* event point *if either:*

1. *Gap* (p,θ) *is a closed endpoint of a VGM* $\mathrm{M} \in \mathcal{M}_v(\mathcal{E})$*. Then* $\textsc{Trunc}[\overrightarrow{(p,\theta)}]$ *is its inflection line.*
2. *Gap* (p,θ) *is a proximal gap of a bitangent gap pair. Then* $\textsc{Trunc}[\overrightarrow{(p,\theta)}]$ *is a bitangent complement line.*

The associated truncated gap rays, $\textsc{Trunc}[\overrightarrow{(p,\theta)}]$*, are referred to as* event lines*.*

This definition allows another interpretation of Lemma 2. Event lines divide the space into a closed set and an open one, only the former including the line itself. When a gap disappears after physical motions (e.g., involving gap-chasing or revealing), the event line is a boundary the robot must cross (not merely reach). Or, said differently: the experience of some events involve stopping times while others are only optional times—but a model that constructs a filtration is outside the scope of the current treatment.

The intuition just given, once released from its geometric trappings, has a corresponding pattern for the manifolds. Since a VGM is a 1-manifold, the event points on a VGM slice it into intervals.

Definition 6 (VGM intervals). *Let* x *be an event point on a VGM* $\mathrm{M} \in \mathcal{M}_v(\mathcal{E})$*. The* VGM interval of M abutting x *is the interval starting at* x *and extending maximally in the concealing direction without including another event point.*

Whenever there is an event point $x' \neq x$ in the concealing direction of x, the interval is in the concealing direction of x and the revealing direction of x'; moreover, one endpoint (namely, x) is closed and the other (x') open. Should there be no event point in the concealing direction from x, then a VGM interval is defined between x and M's open endpoint. Note that if M forms a loop and, further, contains a single event point x, then $x = x'$ and the interval is the entire VGM, cut at x, with the closed endpoint reached by traversing the interval in the revealing direction.

Let x be a proximal gap of a bitangent gap pair, then there exist two abutting intervals on M connected at x: one interval contains x as a closed endpoint and the other is open at x. If M has an endpoint, the intervals are connected sequentially starting from the open endpoint of M and proceeding until the interval containing the closed endpoint of M. Otherwise, if M forms a loop, the intervals of M are connected cyclically.

3.4 Trajectories on VGMs

The preceding has given us a collection of 1-manifolds (and shown that, in some cases, 1-manifolds with boundary) in a way that allows one to discuss sensed gaps and their evolution. Whenever the robot observes multiple gaps in its environment, its physical position q is on the intersection of multiple visible gap rays. These gaps are points on the VGMs, each referred to as a *visible gap state* (VG-state). The set of all VG-states gives a low-fidelity/lossy description, representing the robot's state entirely via VGMs. As the robot moves continuously in the environment, the VG-states also trace trajectories in $\mathcal{M}_v(\mathcal{E})$, and each VG-state also moves continuously within a VGM. For instance, should a gap be observed on a differentiable boundary then, as the robot's physical motion is continuous, the origin and the direction of the gap ray will evolve continuously, corresponding to the VG-state moving continuously on its VGM because

$\mathcal{M}_v(\mathcal{E})$'s topology is inherited from the parameterization space $\mathbb{R}^2 \times \mathbb{S}^1$. Whenever the observed gap is on a non-differentiable point, as the robot moves, while the gap ray's origin remains fixed, the direction evolves on its VGM, and the VG-state will also move continuously. As the origin of the gap ray leaves the non-differentiable point, there are two cases. If it moves in a continuous manner to the curve after the non-differentiable point, the VG-state will naturally also move continuously; however, if the non-differentiable point is followed by a polygonal edge, the gap's origin jumps discontinuously. But, as the gaps along the polygonal edge are identified under $\sim_{\mathrm{p}}$, this still results in a continuous motion of the corresponding VG-state in the VGM.

The different robot experiences correspond to different behaviors of the VG-state on the VGMs, and the demarcation into VGM intervals helps in understanding this. As a robot moves, if no gap event occurs, then all VG-states move within their VGM intervals. Recalling Section 2.2, if the robot crosses an inflection line, a GAPAPPEAR or a GAPDISAPPEAR will occur, depending on the direction of crossing. While on the VGMs, a new VG-state will appear from the closed endpoint of a VGM, or an existing VG-state will reach the closed endpoint of a VGM and disappear from there. If the robot crosses a bitangent complement line, a GAPSPLIT and a GAPMERGE will occur depending on the direction of the robot's motion. In a GAPSPLIT, an existing VG-state will move across a proximal gap in the revealing direction and enter the next interval from the open endpoint. At the same time, another VG-state will appear at the distal gap in the same bitangent gap pair. In a GAPMERGE, an existing VG-state will move across a proximal gap in the concealing direction and it will enter the next interval from the closed endpoint, and another VG-state will disappear at the distal gap. Note that a VG-state crossing a distal gap of a bitangent gap pair does not cause any gap event, as the truncated gap ray of the distal gap is the bitangent line itself, not the bitangent complement—a gap event only occurs when the robot crosses a bitangent complement. For this reason, we do not count a distal gap as an event point.

Note that gap events occur as the event point is attained or passed over, and that the type of gap event depends on the direction of travel. When a gap arrives at or crosses the event point in the revealing direction, a *revealing event* is said to have occurred; a *concealing event* occurs when the same takes place while the gap moves in the concealing direction. As these two different events are associated with the same event point, we will refer to them as *dual events*.

4 Combinatorial Structure of the VGMs

The preceding discussion has begun contemplating multiple VG-states, for instance corresponding to multiple gaps observed simultaneously. It has also touched upon events relating different VGM points—shortly, we shall explore how their interrelationships encode constraints on possible observation sequences produced as a robot undergoes motion. While manifolds are especially useful objects for thinking about continuous aspects of behavior, the occurrence of an event is a discrete element of the robot's experience. Consequently, for sequences of events, we now introduce a graph representation.

We desire to associate bitangent gap pairs across VGMs. Specifically, for each interval with the closed endpoint at a proximal gap, we relate it with the interval that contains the distal gap in the same pair. The following helps express this:

Definition 7 (VGM Graph). *The* VGM graph *of $\mathcal{M}_v(\mathcal{E})$, which will be denoted $G_v(\mathcal{E})$, is the hypergraph $(\mathbb{V},\mathbb{E})$ where*

1. *the vertex set $\mathbb{V}$ contains a unique vertex ι for each VGM interval in $\mathcal{M}_v(\mathcal{E})$ and a special terminal vertex $\overline{\top}$;*
2. *the edge set $\mathbb{E} = \mathbb{E}_r \cup \mathbb{E}_c$, where $\mathbb{E}_r$ is referred to as the revealing edge set and $\mathbb{E}_c$ as the concealing edge set;*
3. *for interval vertex ι, let x be the closed endpoint of the VGM interval,*
   (a) *if x is a proximal gap, then $(\iota,(\iota',\iota''))\in\mathbb{E}_r$ and $((\iota',\iota''),\iota)\in\mathbb{E}_c$, with ι' being the vertex of the interval that is open at x, and ι'' being the vertex of the interval that contains the distal gap in the same bitangent gap pair as x.*
   (b) *if x is the closed endpoint of a VGM* M*, then $(\iota,\overline{\top})\in\mathbb{E}_r$ and $(\overline{\top},\iota)\in\mathbb{E}_c$.*

The graph is "directed" in a generalized form: For edges $(\iota,\overline{\top})$ and $(\overline{\top},\iota)$, the notion of starting and ending vertex is standard. For a hyperedge $(\iota,(\iota',\iota''))$, we refer to ι as the starting vertex, and both ι',ι'' as ending vertices. Analogously, for $((\iota',\iota''),\iota)$, ι' and ι'' are both starting vertices and ι is the ending vertex. For two edges $e,e'\in E$, we say e' *follows e at ι* if an ending vertex ι of e, with $\iota\neq\overline{\top}$, is a starting vertex of e'.

4.1 Permissible Trajectories

Section 3.4 provided a detailed description establishing the correspondence between gaps a robot experiences as it undergoes motion and trajectories in the VGM. These trajectories can be represented as paths on the VGM-graph, where VG-states traverse vertices and edges according to gap events. The formal structure is captured through the notion of permissible trajectories.

Definition 8 (Permissible trajectory). *A sequence $\lambda=(I_1,e_1,I_2,\ldots,e_n,I_{n+1})$ is a* permissible trajectory *on a VGM-graph $G_v(\mathcal{E})$ if each $I_i\subseteq\mathbb{V}(G_v(\mathcal{E}))\setminus\{\overline{\top}\}$, each $e_i\in\mathbb{E}(G_v(\mathcal{E}))$, and*

1. *if $e_i=(\iota,\overline{\top})$, then $\iota\in I_i$, $I_{i+1}=I_i\setminus\{\iota\}$;*
2. *if $e_i=(\overline{\top},\iota)$, then $\iota\notin I_i$, $I_{i+1}=I_i\cup\{\iota\}$;*
3. *if $e_i=(\iota,(\iota',\iota''))$, then $\iota\in I_i$, $\iota',\iota''\notin I_i\setminus\{\iota\}$, and $I_{i+1}=(I_i\setminus\{\iota\})\cup\{\iota',\iota''\}$;*
4. *if $e_i=((\iota',\iota''),\iota)$, then $\iota',\iota''\in I_i$, $\iota\notin I_i\setminus\{\iota',\iota''\}$, and $I_{i+1}=(I_i\setminus\{\iota',\iota''\})\cup\{\iota\}$.*

In this definition, the configuration I_{i+1} is fully determined by I_i and e_i, so λ will be abbreviated as $(I_1,e_1,e_2,\ldots,e_n)$.

Given a VGM-graph $G_v(\mathcal{E})$, the set of all permissible trajectories is denoted by $\Lambda(G_v(\mathcal{E}))$. A permissible trajectory $\lambda\in\Lambda(G_v(\mathcal{E}))$ provides a temporal and spatial discretizations of VG-state dynamics on $\mathcal{M}_v(\mathcal{E})$: interval vertices in I_i represent the intervals occupied by VG-states between gap events,[1] while each edge e_i represents the occurrence of a gap event. The difference between consecutive configurations I_i and

[1] This relies on the fact that no two VG-states occupy the same interval simultaneously. If two VG-states occupied the same interval, their gap rays would originate from a shared boundary curve, with the robot lying on both rays. Since both gaps are either left or right gaps, part of the curve must lie inside its own convex hull, implying a bitangent line with one tangent on this curve and another on a different curve or elsewhere on the same curve. This would introduce an event point within the interval, contradicting the single-interval assumption.

I_{i+1} captures the change of VG-state across the ith event. The four requirements ensure that if a starting vertex of e_i is an interval vertex, then the interval must be occupied before the gap event e_i, and the intervals are never doubly occupied after an event.

The VG-state dynamics on the VGM-graph share structural similarities with Petri nets [8] in modeling state transitions with varying cardinalities: interval vertices correspond to places, the $\overline{\top}$ vertex to a source/sink place, hyperedges to transitions with fork-join patterns, and VG-states to tokens whose total count varies as the robot moves. However, unlike standard Petri nets where tokens are interchangeable, VG-states are distinguishable as they correspond to distinct gap observations.

Note that each e_i in a permissible trajectory λ has two referents: an edge in the VGM-graph, and the gap event represented by this edge. We use e_i to denote the edge, and use "gap event e_i" to emphasize the specific event it corresponds to.

4.2 Event Ordering

The VGM-graph expresses temporal obligations about gap events in λ:

Definition 9 (Temporally precede). *Let $\lambda = (I_1, e_1, e_2, \ldots, e_n)$ be a permissible trajectory in $\Lambda(G_v(\mathcal{E}))$. For $e_i, e_j \in \lambda$, gap event e_i* temporally precedes *e_j in λ, denoted as $e_i \prec e_j$, if there exists a non-terminal ending vertex ι of e_i, such that e_j is the edge with the smallest index among edges starting at ι with an index greater than i.*.

An edge $e_i = (\iota, (\iota', \iota''))$ can temporally precede up to two edges, and $((\iota', \iota''), \iota)$ can be preceded by two. The $\prec$ relation captures the basic property of the VG-state's motion on the VGM-graph: an event that causes a VG-state to enter a certain interval must precede the next event in which the same VG-state leaves that interval. Note that while events are temporally ordered in a permissible trajectory, the $\prec$ does not prescribe a total order. This partial freedom can reflect reality: Fig. 4a provides an example where, depending on the path the robot takes, any order between events not violating the $\prec$ can be experienced.

In this example, a robot at position q_1 will navigate to q_2. A black boundary curve between q_1 and q_2 defines an L-VGM and an R-VGM. The L-VGM contains two event points: a closed endpoint with the event line drawn in green and a proximal gap with the event line drawn in orange. The distal gap of the proximal gap is outside this figure and is not depicted. The R-VGM also has a closed endpoint and a proximal gap, with the event lines shown in orange and green respectively. The robot at q_1 currently observes one gap in the R-VGM. Before arriving at q_2, the robot must experience the two revealing events at the event points in the R-VGM, which are a GAPSPLIT and a GAPDISAPPEAR labeled e_1^r and e_2^r respectively, and concealing events in the L-VGM—a GAPMERGE and a GAPAPPEAR labeled e_1^l and e_2^l. The VGMs graphs of the two VGMs are illustrated in Fig. 4c, with the blue dot marking the interval containing the VG-state at the initial position q_1. The graph encodes the temporal precedence between gap events: $e_1^r \prec e_2^r$ and $e_2^l \prec e_1^l$, with all other event pairs remaining free. Meanwhile, these event lines split the planar environment into nine regions, labeled ①–⑨. The arrangement of these regions forces e_1^r to occur before e_2^r and e_2^l to occur before e_1^l, while other event orders can all be realized depending on the robot's path. For example, the robot can reach q_2 by traversing regions ①②④⑦⑨ sequentially. Along this trajectory,

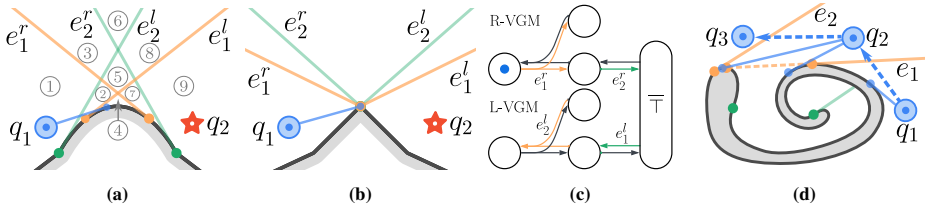

Fig. 4: Examples of the partial order of gap events. (a) An environment with two VGMs for the convex part of the boundary curve between robot positions q_1 and q_2. The L-VGM contains a closed endpoint (green event line) and a proximal gap (orange event line), while the R-VGM contains a proximal gap (orange event line) and a closed endpoint (green event line). Event lines partition the environment into nine regions (labeled ① to ⑨), clarifying how path selection determines the order of gap events. The robot must experience events e_1^r before e_2^r and e_2^l before e_1^l, while other event orderings depend on the chosen trajectory. **(b)** A polygonal environment where multiple gap events occur simultaneously as the robot traverses from q_1 to q_2 by crossing a polygonal vertex. The VGM-graph structure matches Fig. 4a, with four event points having gap rays originating from the same boundary point. **(c)** VGM-graph representation of the two VGMs from Fig. 4a. The blue dot marks the interval of the current VG-state corresponding to the robot's position at q_1. The orange and green edges correspond to the events the robot will experience in Fig. 4a and 4b. **(d)** An environment showing how two VG-states may exist on the same VGM simultaneously.

the event points will be encountered in order $e_2^l, e_1^l, e_1^r, e_2^r$; while along a path traversing regions ①③⑥⑧⑨ the event points will be encountered in order $e_1^r, e_2^r, e_2^l, e_1^l$.

The path traced in reality depends on the physical extent of the robot, the different actions the robot chooses to execute, and it being subject to motion noise. For example, the GNT algorithm has a chasing motion so that a point robot on a boundary curve like that of Fig. 4c must continue strictly on the boundary, and in the above example will experience event order $e_2^l, e_1^l, e_1^r, e_2^r$, whereas a disk robot that cannot enter region ④ will never encounter such an order.

Order Violation? In the previous discussion (and also the example) it appears that the events *on the same VGM* are totally ordered: A VGM is a 1-dimensional manifold and VGM intervals form a chain. A VG-state moving continuously on a VGM cannot skip intervals; if it moves in the revealing direction towards the closed endpoint of the VGM, then the GAPSPLIT it has already experienced should not happen again. These facts are true, but the conclusion is not. In the example in Fig. 4d, this seeming total order of events within the same VGM is violated. A spiral-shaped obstacle is illustrated, whose boundary consists of a convex curve and a concave curve divided by the two inflection points (green solid dots). An L-VGM is defined on the convex curve. The event lines in the L-VGM are shown as solid lines, with green for inflections and orange for bitangent complement lines. The dashed orange line represents a bitangent line. As the robot moves from q_1 to q_2 then to q_3, two GAPSPLITs e_1 and e_2 will occur in this L-VGM as the robot crosses the two bitangent complement lines. Temporally, e_1 occurs before e_2. However, on the VGM-graph, e_1 follows e_2, as e_1 occurs on the bitangent complement line that is closer to the closed endpoint of the VGM than e_2.

The reason of this conflict in ordering is that, after GAPSPLIT e_1 occurs, a new VG-state appears as a distal gap. That new VG-state is on the *same* VGM as the proximal

gap. The new VG-state then experiences e_2. Each individual VG-state follows the event order on a VGM, but they are decoupled and their joint experience need not necessarily follow the same order. Casting one's eye back across the earlier definitions shows that there was no claim that, at any point in time, at most one VG-state could appear on a VGM. In fact, at most two are possible and the spiral exemplifies such cases.

4.3 Simultaneity in Gap Events

Even with Assumption 2, gap events can still occur simultaneously. This situation can be inevitable for the GNT algorithm, with motion continuing strictly on the environment boundary. In the polygonal environment shown in Fig. 4b, the VGM-graph structure is identical to that of Fig. 4a, while multiple event points have their gap rays originating from a single boundary point. A robot at q_1 chasing the gap it observes will reach q_2. As it crosses the polygonal vertex, gap events on all four event points will be triggered. This situation is thorny, because we cannot simply combine all four events into one operation without losing information about each individual event. One needs to assume that individual events can still be identified. But this raises the question: in which order these events should occur? The VGM-graph provides the answer: the order must be one following the $\prec$ encoded by the VGM-graph. This ensures that the trajectory is permissible. A starting vertex of e_i must be in I_i, and no interval is doubly occupied.

5 Navigation with Permissible Trajectories

The preceding sections constructed the VGM-graph $G_v(\mathcal{E})$ as a combinatorial abstraction of gap dynamics, and characterized permissible trajectories $\Lambda(G_v(\mathcal{E}))$ as sequences of gap events respecting the graph's structure. We now turn to a key question: how do gap navigation algorithms operate on this abstraction?

The important insight is that algorithms like the GNT cannot distinguish between physically realizable trajectories and the broader class of permissible ones. Given only gap event data, a robot cannot infer whether its motion corresponds to a continuous path in the environment or to some other sequence of gap events that is consistent with the VGM-graph but physically unrealizable. Consequently, any correct navigation algorithm must handle *all* permissible trajectories, not merely those corresponding to physical paths. This observation transforms what might seem like a limitation into a strength: by proving correctness over the larger class $\Lambda(G_v(\mathcal{E}))$, we obtain algorithms robust to sensing ambiguities and motion uncertainties.

5.1 Physical Realizability *vs* Permissibility

Not every permissible trajectory corresponds to a physical robot motion. The definition of permissible trajectories (Definition 8) enforces only local consistency—that edges connect properly and intervals are not doubly occupied—but imposes no global geometric constraints. We make this distinction precise.

Definition 10 (Physically realizable trajectory). *A permissible trajectory* $\lambda = (I_1, e_1, \ldots, e_n) \in \Lambda(G_v(\mathcal{E}))$ *is* physically realizable *if there exists a continuous path* $\gamma\colon [0,T] \to \mathcal{E}$ *inducing exactly the gap events* $e_1, \ldots, e_n$ *in order, with initial gap observation corresponding to* I_1.

Physically realizable trajectories form a proper subset of $\Lambda(G_v(\mathcal{E}))$. The discrepancy arises from three sources: (1) *impossible simultaneity*—two intervals may be jointly occupied in some I_i even if no position in $\mathcal{E}$ lies on both gap rays; (2) *forbidden orderings*—geometric arrangement of event lines may force certain crossings to precede others, constraints not encoded in the VGM-graph; and (3) *motion constraints*—kinematic limitations such as robot's physical footprint or bounded turning radius can render certain event sequences unreachable.

Despite these discrepancies, permissible trajectories are useful for analysis. An algorithm's correctness holds in reality if established over the larger class $\Lambda(G_v(\mathcal{E}))$.

5.2 Gap Navigation and Dual Event Identification

The GNT algorithm [11] maintains a tree data structure as a roadmap for navigation. Each child of the root represents a currently observed gap, with siblings ordered cyclically according to the gaps' angular positions. The tree evolves with gap events as follows: At a GAPAPPEAR, a new node is added as a child of the root, inserted into the cyclic order according to the gap's direction. At a GAPDISAPPEAR, the leaf node corresponding to the vanishing gap is removed. At a GAPMERGE where g_1 and g_2 merge into g, nodes g_1 and g_2 become children of a new node g, which is added to the root; the cyclic order between g_1 and g_2 is stored. At a GAPSPLIT where g splits into g_1 and g_2, if g is a leaf, it is replaced by two new leaf nodes g_1 and g_2; if g has children, node g is removed and its children are re-associated with g_1 and g_2 by matching the stored cyclic order against current observations.

Navigation proceeds by chasing gaps to trigger GAPSPLITs that undo previous merges. When the target lies in a region hidden by some gap g in a GAPMERGE, the algorithm locates the corresponding node and chases gaps to remove g's predecessors in the tree. Once g is attached directly to the root, chasing g triggers a GAPSPLIT that reveals the target.

The algorithm's correctness hinges on *dual event identification*: the GAPSPLIT where node g is removed must be the dual of the GAPMERGE that created g, and the children of g correctly re-associated. The original GNT uses cyclic ordering for this re-association, but as shown in [9], this can fail when strict gap-chasing motion is violated.

The event-augmented coherence sensor [9] resolves this limitation by distinguishing proximal and distal gaps through coherence relations $\overset{\text{S}}{\rightsquigarrow}$ and $\underset{\text{S}}{\rightsquigarrow}$ in a GAPSPLIT and $\overset{\text{M}}{\rightsquigarrow}$ and $\underset{\text{M}}{\rightsquigarrow}$ in a GAPMERGE. The *modified GNT algorithm* replaces cyclic order matching with edge-type matching: at a GAPMERGE, children are attached via a *proximal edge* or *distal edge* according to which gap is proximal and which is distal; at a GAPSPLIT where g has children, they are re-associated with the new gaps by matching proximal/distal edge types between the merge and split events.

The temporal precedence relation $\prec$ of Definition 9 provides the theoretical foundation for dual event identification:

Lemma 3. *Let $e_{i_1}, e_{i_2}, \ldots, e_{i_{2m}} \in \lambda$ be a sequence of* GAPSPLIT*s and* GAPMERGE*s with e_1 being a* GAPSPLIT*, e_{2m} being a* GAPMERGE*, and $e_{i_j} \prec e_{i_{j+1}}$ for all j, and each event involving a proximal gap in the same VGM. If the sequence contains exactly m* GAPSPLIT*s and m* GAPMERGE*s, then e_{i_1} and $e_{i_{2m}}$ are dual events.*

The proof sketch for this lemma is deferred to Appendix C.

This counting principle underlies the GNT's tree structure: the depth of a node encodes the number of "unresolved" GAPMERGE events along the VG-state's path through intervals. Each GAPMERGE increases depth by attaching nodes as children; each GAPSPLIT of a non-leaf node decreases it by promoting children back toward the root. However, the counting must be tracked carefully across GAPMERGE and GAPSPLIT events. At a GAPMERGE where g_1 and g_2 merge into g, the proximal gap (g_1) continues the VG-state's trajectory from the previous interval, inheriting the ongoing count, while the distal gap (g_2) pauses its own counting. Symmetrically, at a GAPSPLIT where g splits into g_1 and g_2, the proximal gap (g_1) continues the count, and the distal gap (g_2) restarts its counting independently. By distinguishing proximal and distal edges at each event, the modified algorithm correctly maintains separate counts for each VG-state's path. When the count for a particular path returns to zero—equal numbers of concealing and revealing events involving proximal gaps—the algorithm identifies the dual event and restructures the tree accordingly.

5.3 Correctness on Permissible Trajectories

We now establish that the modified GNT algorithm correctly maintains its data structure across all permissible trajectories and satisfies a geometric property which characterizes observations in simply-connected environments.

Definition 11 (Revealing descendant). *Let v and v' be vertices in $G_v(\mathcal{E})$. We say v' is a* revealing descendant *of v if there exists a sequence of edges $e_1, \ldots, e_n \in \mathbb{E}_r$ such that v is a starting vertex of e_1, v' is an ending vertex of e_n, and e_{j+1} follows e_j for all j. Further, v' is a* strict revealing descendant *of v if v is not a revealing descendant of v'.*

Intuitively, v' being a strict revealing descendant of v means that the interval v' can only be reached from v through a sequence of gap motions in the revealing direction, connected by GAPSPLITs. This captures a notion of "depth" in the GNT data structure.

Definition 12 (Star configuration). *A set $I \subseteq \mathbb{V}(G_v(\mathcal{E})) \setminus \{\top\}$ is a* star configuration *if no pair $\iota_1, \iota_2 \in I$ exist such that ι_1 is a strict revealing descendant of ι_2.*

Star configurations characterize physically achievable observations within simply-connected environments: if two gaps were observed simultaneously with one being a strict revealing descendant of the other, chasing both would lead to the same boundary point via two shortest paths—an impossibility when the environment has no holes. The following theorem shows that the modified GNT correctly maps the VGM-graph structure under this assumption.

Additionally, in a simply-connected environment, there do not exist non-strict revealing descendants for a similar reason. For two intervals in $\mathbb{V}(G_v(\mathcal{E}))$, if neither of them is a strict revealing descendant of the other, then they are not connected by only revealing or concealing edges in $\mathbb{E}(G_v(\mathcal{E}))$. This fact is used in the proof of the theorem.

Theorem 1. *Let $G_v(\mathcal{E})$ be the VGM-graph of a simply-connected environment $\mathcal{E}$ and $\lambda = (I_1, e_1, I_2, \ldots, e_n, I_{n+1}) \in \Lambda(G_v(\mathcal{E}))$ be a permissible trajectory where each I_i is a star configuration. Then the GNT data structure G maintained by the modified GNT algorithm at the end of λ satisfies:*

1. *There exists an injective function f mapping each non-root node $g \in \mathbb{V}(G)$ to an interval vertex $f(g) \in \mathbb{V}(G_v(\mathcal{E}))$, such that g is a child of g' in G if and only if edge $(f(g'), (f(g), \iota))$ or $(f(g'), (\iota, f(g)))$ exists in $\mathbb{E}(G_v(\mathcal{E}))$ for some ι.*
2. *For any interval ι visited by λ, if $\exists g \in \mathbb{V}(G)$ such that ι is a revealing descendant of $f(g)$, then $f^{-1}(\iota) \in \mathbb{V}(G)$.*

Property (1) states that the tree structure of G mirrors the edge structure of the VGM-graph: parent-child relationships in G correspond to GAPMERGE edges in $\mathbb{E}(G_v(\mathcal{E}))$. Property (2) ensures completeness: all visited intervals that are relevant for future dual event identification are represented in G.

The proof of the theorem appears in Appendix D.

The star configuration requirement matches the geometry of simply-connected environments. In multiply-connected environments, a gap may "wrap around" a hole, causing a VG-state to be a revealing descendant of itself. The GNT handles such cases by mapping to a covering space where loops unfold into infinite spirals; we defer this extension to future work.

6 Concluding discussion

This paper builds up formal representations incrementally: it abstracts the environment $\mathcal{E}$ into the topological space $\mathcal{M}_v(\mathcal{E})$ and then further into the VGM-graph $G_v(\mathcal{E})$. Metric information was deliberately discarded along the way—but even the most cursory look at the description of a gap sensor indicates that such information must be beyond the robot's semantic reach, so that was an obvious step. But glancing at the collection illustrated in Fig. 1, it is less obvious what other aspects are at play in the invariance. One consequence of Theorem 1 is that Fig. 1(a)–(e) have the same GNTs because they have identical $\mathcal{M}_v(\mathcal{E})$s and hence isomorphic $G_v(\mathcal{E})$s too; Fig. 1(f) differs precisely because the little intrusion introduces an extra pair of VGMs, so there are extra VG-state intervals, and hence more vertices in its VGM-graph.

One of our modeling principles has been to respect that some facts are unknowable to the robot. In fact, certain topological information was also excluded from our construction owing to it also being fundamentally inaccessible to a robot equipped with only a gap sensor—for instance we know each boundary curve induces two VGMs: a left one (L-VGM) and a right one (R-VGM), but a robot observing two gaps cannot determine whether they originate from a pair of VGMs on the same boundary curve or not. Fig. 1(g) emphasizes that gaps on the left form a distinct class from the gaps on the right, even though a robot can't tell whether it is dealing with a left or right gap. The gap navigation algorithms succeed not by overcoming these sources of inevitable ignorance but by showing that they are irrelevant to navigation.

Another underlying modeling principle has been to represent the robot's perceptual signals —rather than its state. Gaps are curious because their persistence gives them a sort of existence all their own, while also (like all sensing signals) being fundamentally relational in terms of depending on both the observer and that which is being observed.

References

1. Blum, M., Kozen, D.: On the power of the compass (or, why mazes are easier to search than graphs). In: IEEE FOCS. pp. 132–142 (1978)
2. Connell, J.H.: Minimalist Mobile Robotics. A Colony-Style Architecture for an Artificial Creature, Perspectives in AI, vol. 5. Academic Press, Inc. (1990)
3. Erdmann, M.: Understanding Action and Sensing by Designing Action-Based Sensors. IJRR **14**(5), 483–509 (1995). https://doi.org/10.1177/027836499501400506
4. LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge, U.K. (2006), available at http://planning.cs.uiuc.edu/
5. Lopez-Padilla, R., Murrieta-Cid, R., Lavalle, S.M.: Optimal gap navigation for a disc robot. In: WAFR. vol. 86, pp. 123–138. Springer (2013). https://doi.org/10.1007/978-3-642-36279-8_8
6. Murphy, L., Newman, P.: Using incomplete online metric maps for topological exploration with the gap navigation tree. In: IEEE ICRA. pp. 2792–2797 (2008). https://doi.org/10.1109/ROBOT.2008.4543633
7. Nasir, R., Elnagar, A.: Gap Navigation Trees for Discovering Unknown Environments. Intelligent Control and Automation **6**(4), 229–240 (2015). https://doi.org/10.4236/ICA.2015.64022
8. Petri, C.A., Reisig, W.: Petri net. Scholarpedia **3**(4), 6477 (2008)
9. Qian, C., Shell, D.A.: Extracting synthetic sensors: A study in gap navigation. In: WAFR. Oulu, Finland (Jun 2026)
10. Sachs, S., LaValle, S.M., Rajko, S.: Visibility-based pursuit-evasion in an unknown planar environment. IJRR **23**(1), 3–26 (2004). https://doi.org/10.1177/0278364904039610
11. Tovar, B., Murrieta-Cid, R., LaValle, S.M.: Distance-optimal navigation in an unknown environment without sensing distances. IEEE TRO **23**, 506–518 (2007). https://doi.org/10.1109/TRO.2007.898962

A Proof Sketch for Lemma 1

In this section, we prove the omitted proof sketch of Lemma 1:

Proof sketch. The continuity is implied from the discussion thus far, where the original continuity of the boundary curve has been preserved. As to strict monotonicity, we proceed in two steps. First, the θ parameter must vary while traversing a VGM. This is simple as $\sim_{\mathrm{p}}$ identifies all gaps originating from the same polygonal edge with the same direction. After the quotient operation, the remaining gap rays all have their origins on a cusp or a point with curvature. Thus, the θ parameter of neighboring rays is never constant. Second, the change of θ is monotonic. For a (p,θ) that is not an endpoint of M, the (p,θ) bisect its neighborhood into two subintervals: it can only violate monotonicity if the both neighborhoods have angle parameter less than θ, or both neighborhoods have angle parameter greater than θ. Suppose either of these cases occur, and (p,θ) is a differentiable point. Then it meets the definition of a differentiable inflection point, and hence Assumption 1 implies we can find, on either side, a neighborhood that is entirely concave or convex. But the inflection requires that one be concave and other be convex. Then the concave neighborhood violates visibility, so must be outside of M as $\mathrm{M} \subseteq \mathcal{M}_v(\mathcal{E})$. This, thus, contradicts the interiority of (p,θ). The situation when (p,θ) is a non-differentiable point is similar: it meets the definition of a non-differentiable inflection point, and the argument is analogous, albeit with additional care to consider angles that are extremal. For monotonicity to be violated by an endpoint, it would need to break continuity. [End of sketch]

B Proof Sketch for Lemma 2

In this section, we prove the omitted proof sketch of Lemma 2:

Proof sketch. We first characterize VGM endpoints. A point (p,θ) is an endpoint of M only if both neighboring gaps have angle parameters either all less than θ or all greater than θ. If instead the neighbors had angles on opposite sides of θ, then similar to the argument in the previous proof, such (p,θ) can not be the endpoint of a VGM, as the neighboring gap rays are either all visible or all non-visible on both sides of (p,θ).

We now establish the directionality. Let (p,θ) be an endpoint of M with $p = \varphi_c(t)$ on boundary curve c. If p is differentiable, the neighborhood of p on $\partial\mathcal{E}$ is strictly concave on one side and strictly convex on the other. Whether $(p,\theta) \in \mathrm{M}$ depends on which side is convex and whether M is an L-VGM or R-VGM. Suppose M is an R-VGM. If the convex side corresponds to $\tau > t$, then $(p,\theta) \in \mathrm{M}$, forming a closed endpoint; the angle decreases as a gap approaches (p,θ), meaning the endpoint is reached by moving in the revealing direction. Symmetrically, if M is an L-VGM and the convex side has $\tau < t$, then (p,θ) is also a closed endpoint of M reached in the revealing direction. The other endpoint has the convex neighborhood on the opposite side, forming an open endpoint approached in the concealing direction.

If M ends on a non-differentiable point, a similar argument applies by taking into consideration the gaps in the neighborhood of (p,θ) that emanate from p.

[End of sketch]

C The Proof Sketch of Lemma 3

Proof sketch. Relation $\prec$ ensures consecutive events involve the same VG-state traversing intervals that are adjacent. Each revealing event advances the VG-state one interval in the revealing direction; each concealing event reverses this. With equal counts, the VG-state returns to its original interval whence it departed in event e_1. As e_1 is a GAPSPLIT and e_{2m} is a GAPMERGE, they are dual events. [End of sketch]

D The Proof of Theorem 1

In this section, we prove the omitted proof for Theorem 1:

Proof. We proceed by induction on the trajectory length.

Base case: At I_1, the algorithm creates a child of the root for each observed gap. Since I_1 is a star configuration, no interval is a strict revealing descendant of another, so in a simply-connected environment, the interval nodes in I_1 are not connected by concealing edges in $\mathbb{E}(G_v(\mathcal{E}))$. This matches the GNT structure: these nodes are not connected to each other. Setting f to map each node to its corresponding interval satisfies both properties.

Inductive step: Assume the theorem holds at configuration I_i. We show it holds at I_{i+1} after edge e_i.

Case $e_i = (\iota, \overline{\top})$ (GAPDISAPPEAR): The algorithm removes the leaf node g with $f(g) = \iota$. Removing g preserves both properties.

Case $e_i = (\overline{\top}, \iota)$ (GAPAPPEAR): A new node g is added to the root with $f(g) = \iota$. Since I_{i+1} is a star configuration, ι is not a strict revealing descendant of any other interval in I_{i+1}, so ι is not connected to any existing $f(g')$ in $\mathbb{V}(G_v(\mathcal{E}))$. Thus g is correctly placed as a child of the root, preserving property (1). Property (2) holds since ι has no revealing descendants.

Case $e_i = (\iota, (\iota', \iota''))$ (GAPSPLIT): Let g be the node with $f(g) = \iota$. If g is a leaf, it is replaced by nodes g' and g'' with $f(g') = \iota'$ and $f(g'') = \iota''$; property (1) holds as ι' and ι'' are not connected by concealing edges in $\mathbb{E}(G_v(\mathcal{E}))$. Since ι has no visited revealing descendants, neither do ι' and ι'', and property (2) holds. If g has children, they were added at a previous GAPMERGE $((\iota', \iota''), \iota)$. By Lemma 3, this GAPSPLIT is the dual of that GAPMERGE. The algorithm removes g and re-associates its children with g' and g'' by matching proximal/distal edge types. Since the edge types are fixed, the re-association is correct: children previously connected via edges involving ι' attach to g', and similarly for ι''. The function f remains valid, and property (1) is preserved. Moreover, the visited revealing descendants of ι' and ι'' are contained in those of ι, which already exist in G, hence property (2) is preserved.

Case $e_i = ((\iota', \iota''), \iota)$ (GAPMERGE): Nodes g' and g'' with $f(g') = \iota'$ and $f(g'') = \iota''$ become children of a new node g with $f(g) = \iota$, connected by proximal and distal edges matching the event. Since ι' and ι'' are strict revealing descendants of ι, and they and their strict revealing descendants are exactly the strict revealing descendants of ι, and property (2) holds. Property (1) holds by construction. □