# Approximate Distance Oracle for Fault-Tolerant Geometric Spanners

**Kyungjin Cho, Jihun Shin, Eunjin Oh**

Pohang University of Science and Technology, South Korea.
{kyungjincho, tmtingsamuel, eunjin.oh}@postech.ac.kr

## Abstract

In this paper, we present approximate distance and shortest-path oracles for fault-tolerant Euclidean spanners motivated by the routing problem in real-world road networks. A fault-tolerant Euclidean spanner for a set of points in Euclidean space is a graph in which, despite the deletion of small number of any points, the distance between any two points in the damaged graph is an approximation of their Euclidean distance. Given a fault-tolerant Euclidean spanner and a small approximation factor, our data structure allows us to compute an approximate distance between two points in the damaged spanner in constant time when a query involves any two points and a small set of failed points. Additionally, by incorporating additional data structures, we can return a path itself in time almost linear in the length of the returned path. Both data structures require near-linear space.

## Introduction

Computing the shortest path in a graph is a fundamental problem motivated by potential applications such as GPS navigation, route planning services and POI recommendation for real-world road networks. Although the shortest path can be computed by Dijkstra's algorithm, it is not sufficiently efficient if the given graph is large. This requires us to preprocess a given graph so that for two query vertices, their shortest path can be computed more efficiently. A data structure for this task is called a *shortest-path (or distance) oracle*.

From the theoretical viewpoint, this problem is not an easy task. More specifically, any data structure for answering $(2k+1)$-approximate distance queries in $O(1)$ time for $n$-vertices graphs must use $\Omega(n^{1+1/k})$ space assuming the 1963 girth conjecture of Erdős (Thorup and Zwick 2005). On the other hand, there are algorithms for this task that work efficiently for real-world road networks in practice such as contraction hierarchies (Kuhn et al. 2005), transit nodes (Bast et al. 2007), and hub labels (Abraham et al. 2011). Although these algorithms work well in practice, there is still a lack of theoretical explanation for this. Bridging this theory-practice gap is one of interesting topics in computer science. Indeed, there are lots of works on bridging the theory-practice gap in the routing problems such as (Blum, Funke, and Storandt 2018; Kosowski and Viennot 2017; Abraham et al. 2016).



**Motivation.** In real-life situations, networks might be vulnerable to unexpected accidents: edges or vertices might be failed, but these failures are transient due to a repair process. As the network is large, we cannot afford to construct the entire data structure from scratch. This motivates the study of *fault-tolerant* distance and shortest-path oracles on *vulnerable* networks: preprocess a graph $G = (V, E)$ so that for any set $F$ of $f$ *failed* vertices (or edges) of $G$ and two query vertices $s$ and $s'$, we can compute a shortest path in $G - F$ between $s$ and $s'$ efficiently. A data structure that can handle failed vertices (or edges) is said to be *fault-tolerant*. The theoretical performance of a fault-tolerant data structure is measured by a function of the number $n$ of vertices of an input graph and the maximum number $f$ of failed vertices.

Although this problem is natural, little is known for vertex failures while there are lots of work on edge failures. To the best of our knowledge, there is the only published result on (approximate) distance oracles for general graphs in the presence of vertex failures: an $O(n^2)$-sized approximate distance oracle answering queries in $\mathrm{polylog}(n)$ time in the presence of a constant number of vertex failures (Duan, Gu, and Ren 2021). On the other hand, there are lots of theoretical results on (approximate) distance oracles for general graphs in the presence of edge failures (Bodwin et al. 2018; Chechik et al. 2017; Ren 2022). Dynamic graphs whose edge weights change over time also have been studied from a more practical point of view (Ouyang et al. 2020; Zhang et al. 2021; Zhang, Li, and Zhou 2021). In this case, vertex updates (and vertex failures) are not allowed.

This raises an intriguing question: Can we design an efficient oracle for handling vertex failures? The size of the best-known oracle is $O(n^2)$, which is still large for practical purposes. In real-life situations, vertices as well as edges are also prone to failures. A bus map can be considered as a graph where its vertices correspond to the bus stops. A bus stop can be closed due to unexpected events, and then buses make detours. This changes the bus map temporarily. To address this scenario, we need vertex-fault tolerant oracles. In this paper, we design a new efficient oracle for answering approximate distance and shortest-path queries for real-world road networks from a theoretical point of view.

Similar to the static case, there is a huge gap between theory and practice in the dynamic settings: theoretical solutions are not efficient in general while practical solutions do not

| | | Kernel oracle | Distance oracle | Path-Pres. Kernel oracle | Shortest-Path oracle |
|---|---|---|---|---|---|
| **Space** | | $h(t,f,\varepsilon)(n\log n)$ | | | $h(t,f,\varepsilon)(n\log^2 n\log\log n)$ |
| **Construction time** | | $h(t,f,\varepsilon)(n\log^2 n)$ | | | $h(t,f,\varepsilon)(n^2\log^2 n)$ |
| **Query time** | **Moderately Far** | $O(t^8f^4)$ | | $O(t^8f^4\log^3(tf/\varepsilon)\log^2 n)$ | $O(f^4\log^2 n\log\log n+\mathsf{sol})$ |
| | **General** | - | $O(t^8f^4)$ | - | $O(f^4\log^2 n\log\log n+f\cdot\mathsf{sol})$ |

Table 1: Performance of our oracles. $h(t,f,\varepsilon)=\exp(O(f\log(tf/\varepsilon)))$ and $\mathsf{sol}$ is the number of vertices of the reported path.

give theoretical explanation why they work efficiently. Researchers also tried to bridge the theory-practice gap. For instance, (Ouyang et al. 2020) gave a shortest-path oracle for dynamic road networks with theoretical guarantees together with experimental evaluations. Their theoretical guarantees partially explain why their result works efficiently. In particular, they analyzed the performance of their oracle in terms of the size of an input graph and some parameter depending on their construction algorithms. Then they showed that this parameter is small for real-world road networks. This still does not tell us why this parameter is small in practice, and the definition of the parameter does not look intuitive as it depends on their algorithms. Instead, we choose a different approach to bridge the gap between theory and practice from a more theoretical point of view in a more classical method: first define a theoretical model for real-world road networks, and then design an oracle on this theoretical model.

Due to lack of space, some proofs and details are omitted in the conference version of this paper. All missing proofs and details can be found in the full version of this paper.

**Theoretical model.** A *geometric graph* is a graph where the vertices correspond to points in $\mathbb{R}^d$ and the weight of each edge is the Euclidean distances between the endpoints of the edge. Let $V$ be a set of $n$ points in $\mathbb{R}^d$ for a constant $d\geq 1$. A geometric graph $G=(V,E)$ with $|E|=O(|V|)$ is called a *Euclidean $t$-spanner* of $V$ if the distance in $G$ between any two vertices is at most $t$ times the Euclidean distance of their corresponding points. More generally, a geometric graph $G=(V,E)$ with $|E|=f^{O(1)}|V|$ is an *$f$-fault-tolerant Euclidean $t$-spanner* if the distance in $V-F$ between two vertices $u$ and $v$ is at most $t$ times the Euclidean distance between $u$ and $v$ for any set $F$ of at most $f$ vertices of $G$. Here, we call the vertices of $F$ the *failed vertices*.

Lots of road networks can be represented as Euclidean $t$-spanners for a small constant $t$. For instance, a southern Scandinavian railroad network is a 1.85-spanner (Narasimhan and Smid 2007). Thus it is reasonable to use a fault-tolerant Euclidean spanner as a theoretical model for our purposes (Sommer 2014). Apart from this, Euclidean spanners have various applications such as pattern recognition, function approximation and broadcasting systems in communication networks (Narasimhan and Smid 2007). Due to its wide range of applications, many variations of fault-tolerant Euclidean spanners have been studied extensively (Abam et al. 2009; Levcopoulos, Narasimhan, and Smid 1998, 2002; Bose et al. 2013; Buchin, Har-Peled, and Oláh 2020, 2022; Filtser and Le 2022). For static Euclidean spanners without vertex/edges failures, (Gudmundsson et al. 2008; Oh 2020) showed that a Euclidean spanner admits an efficient approximate distance oracle. Their oracle has size $O(n\log n)$ and answers approximate distance queries in $O(1)$ time.

**Our result.** In this paper, we present the first near-linear-sized approximate distance and shortest-path oracle specialized for fault-tolerant Euclidean spanners. More specifically, given a fault-tolerant Euclidean $t$-spanner with constant $t$ and a value $\varepsilon$, we present a near-linear-sized data structure so that given two vertices $s$, $s'$ and a set $F$ of at most $f$ failed vertices, an $(1+\varepsilon)$-approximate distance between $s$ and $s'$ in $G-F$ can be computed in $\mathrm{poly}\{f,t\}$ time. Moreover, we can report an approximate shortest path $\pi$ in time almost linear in the complexity of $\pi$. The explicit bounds are stated in Table 1. Our oracle is significantly more compact compared to the quadratic-sized distance oracle (Duan, Gu, and Ren 2021) for general graphs.

We only consider spanners constructed in a two-dimensional Euclidean space. However, our ideas can be extended to a general $d$-dimensional Euclidean space without increasing the dependency on $n$ in the performance guarantees. We provide a sketch of this extension in the appendices.

**Related work.** Although nothing is known for (approximate) distance oracle specialized for fault-tolerant Euclidean spanners, designing fault-tolerant structures is a popular topic in the field of algorithms and data structures. Fault-tolerant structures have received a lot of interests over the past few decades. In general, there are two types of problems in the research of fault-tolerant structures. For the first type of problems, the goal is to process a given graph $G=(V,E)$ which can have failed vertices (or edges) so that for a set $F$ of failed vertices (or edges) given as query, it can efficiently respond to several queries on the subgraph of $G$ induced by $V-F$ (or $E-F$). Various types of queries have been studied, for instance, reachability queries (van den Brand and Saranurak 2019), shortest path queries (Bodwin et al. 2018; Charalampopoulos, Mozes, and Tebeka 2019; Duan, Gu, and Ren 2021; Ren 2022; van den Brand and Saranurak 2019), diameter queries (Bilò et al. 2021), and $k$-paths and vertex cover queries (Braverman 2022). The problem we consider in this paper also belongs to this type of problems. For the second type of problems, the goal is to compute a sparse subgraph $H$ of a given graph $G$ so that for any set $F$ of failed vertices (or edges), $H-F$ satisfies certain properties. For instance, the problems of computing sparse fault-tolerant spanners (Bilò et al. 2015; Chechik et al. 2009; Czumaj and Zhao 2004; Dinitz and Krauthgamer 2011; Dinitz and Robelle 2020; Parter 2022) and fault-tolerant distance preservers (Bodwin

et al. 2020) have been widely investigated.

## Preliminaries

For a graph $G = (V, E)$ and two vertices $u$ and $v$ of $G$, let $\pi_G(u, v)$ be a shortest path between $u$ and $v$ within $G$, and $d_G(u, v)$ be the distance between $u$ and $v$ in $G$. An $(1 + \varepsilon)$-*approximate distance* between $u$ and $v$ of $G$ is defined as any value $\ell$ lying between $d_G(u, v)$ and $(1 + \varepsilon)d_G(u, v)$. Here, there does not necessarily exist a path in $G$ of length exactly $\ell$, but it can be considered as a good estimate of the distance between $u$ and $v$. Analogously, an $(1+\varepsilon)$-*approximate shortest path* between $u$ and $v$ in $G$ is a path in $G$ of length at most $(1 + \varepsilon)d_G(u, v)$.

For two points $p$ and $q$ in a Euclidean space, let $|pq|$ be the Euclidean distance between $p$ and $q$. With a slight abuse of notation, we use $|\gamma|$ to denote the length of a path $\gamma$ of $G$ (the sum of the weights of the edges of $\gamma$). Note that a path in a graph is a sequence of adjacent edges, and equivalently, it can be represented by a sequence of incident vertices. For two numbers $a, b \in \mathbb{R}$, we let $[a, b]$ be the closed interval between $a$ and $b$. Also, let $(a, b]$ and $[a, b)$ be half-closed intervals excluding $a$ and $b$, respectively. In addition, let $(a, b)$ be the open interval between $a$ and $b$.

A geometric graph $G$ is called an *$L$-partial $f$-fault-tolerant Euclidean $t$-spanner* if $d_{G-F}(u, v) \leq t|uv|$ for any two vertices $u$ and $v$ with $|uv| \leq L$ and any set $F$ of at most $f$ failed vertices. We say two points $s$ and $s'$ are *moderately far* in $G$ if $|ss'| \in [\frac{L}{m^2}, \frac{L}{t})$, where $m$ is the number of edges in $G$. For two paths $\gamma$ and $\gamma'$, we say $\gamma$ can be *extended* to $\gamma'$ if the vertex sequence of $\gamma$ is a subsequence of the vertex sequence of $\gamma'$. Even if $\gamma$ and $\gamma'$ are from different graphs $H$ and $H'$, respectively, we can define the extension relation when $V(H) \subseteq V(H')$.

In the rest of this section, we introduce four main tools and concepts used in the design of our data structures.

**Generalization Lemmas.** The following *generalization lemmas* states that once we have oracles on $L$-partial $f$-fault tolerant spanners for moderately far vertices, we can use them as black boxes to handle a query consisting of any two (not necessarily moderately far) vertices.

**Lemma 1.** *Assume that for any $\varepsilon' > 0$, we can construct an oracle of size $h_s(n)$ in $h_c(n)$ time on an $n$-vertices partial fault-tolerant Euclidean spanner for supporting $(1 + \varepsilon')$-approximate distance queries for moderately far vertices and $f$ failed vertices in $T$ time. Then we can construct an oracle of size $O(h_s(f^2n) + f^2n)$ in $O(h_c(f^2n) + f^2n \log n)$ time for answering $(1 + \varepsilon)$-approximate distance queries for any $\varepsilon > 0$, two vertices, and $f$ failed vertices in $O(f + T)$ time.*

**Lemma 2.** *Assume that for any $\varepsilon' > 0$, we can construct an oracle of size $h_s(n)$ in $h_c(n)$ time on an $n$-vertices partial fault-tolerant Euclidean spanner for supporting $(1 + \varepsilon')$-approximate shortest-path queries for moderately far vertices and $f$ failed vertices in $T$ time. Then we can construct an oracle of size $O(h_s(f^2n) + f^2n \log^2 n \log\log n)$ in $O(h_c(f^2n) + f^2n^2 \log^2 n)$ time for answering $(1 + \varepsilon)$-approximate shortest-path queries for any $\varepsilon > 0$, two vertices, and $f$ failed vertices in $O(f^4 \log^2 n \log\log n + T + f \cdot \mathsf{sol})$ time, where $\mathsf{sol}$ is the number of vertices in the returned path.*

Note that the parameter $L$ does not appear in the performance guarantees. This is because $L$ determines if two vertices are moderately far. In particular, for $L = 0$, all geometric graphs are $L$-partial fault-tolerant Euclidean spanners, but no two vertices are moderately far. In the following, we let $G$ be an $L$-partial $f$-fault-tolerant Euclidean spanner, and $(s, s')$ be a pair of moderately far vertices unless otherwise stated.

**Kernel.** We call an edge-weighted graph $H = (V_H, E_H)$ an $(s, s', F; \varepsilon)$-*kernel* of $G$ if the following hold.

- $s, s' \in V_H \subseteq V(G)$,
- $d_{G-F}(s, s') \leq d_H(s, s') \leq (1 + \varepsilon)d_{G-F}(s, s')$, and
- $\pi_H(s, s')$ can be extended to an $(1+\varepsilon)$-approximate shortest path between $s$ and $s'$ in $G - F$.

We define the *size* of a kernel as the number of vertices and edges of the kernel. For an edge $uv$ of $H$, its weight is denoted by $w_H(uv)$.

Our main strategy is to construct a data structure on a partial fault-tolerant Euclidean spanner $G$ and a value $\varepsilon$ for computing an $(s, s', F; \varepsilon)$-kernel of small complexity for two vertices $s$ and $s'$ and a set $F$ of failed vertices given as a query. We call this data structure a *kernel oracle*. By the definition of kernels, once we have an $(s, s', F; \varepsilon)$-kernel of $G$, we can compute an $(1+\varepsilon)$-approximate distance between $s$ and $s'$ in $G - F$ in time near linear in the complexity of the kernel by applying Dijkstra's algorithm to the kernel. For distance oracles, it is sufficient to construct a kernel oracle for computing a kernel of small complexity.

However, it is not sufficient for approximate shortest-path oracles. To retrieve an $(1 + \varepsilon)$-approximate shortest path of $G - F$ from a shortest path $\pi$ of a kernel $H$, we are required to efficiently compute a path in $G - F$ between $u$ and $v$ of length $w_H(uv)$ for every edge $uv$ of $H$. Then we can replace every edge of $\pi$ with its corresponding path such that the resulting path becomes an $(1 + \varepsilon)$-approximate shortest path between $s$ and $s'$ in $G - F$. From this motivation, we define the notion of *path-preserving kernels* as follows. We say a $(\cdot, \cdot, F; \cdot)$-kernel $H$ of $G$ is *path-preserving* if for every edge $uv$ with $w_H(uv) \leq d_H(u, v)$, at least one of the following holds:

- $d_G(u, v) \leq tL/m^6$, or
- we can efficiently compute a path in $G - F$ between $u$ and $v$ of length $w_H(uv)$.

Note that an edge $u'v'$ with $w_H(u'v') > d_H(u', v')$ does not appear in any shortest path in $H$. If $d_G(u, v) \leq tL/m^6$, we will see that it is sufficient to replace $uv$ with an arbitrary path between $u$ and $v$ consisting of edge of length at most $tL/m^6$ to obtain an approximate shortest path in $G - F$.

**Net Vertices.** For an $r > 0$, a set $\mathcal{N}$ of vertices of $G$ is called an *$r$-net* if

- $d_G(u, v) \geq r$ for any two net vertices $u, v \in \mathcal{N}$, and
- $\min_{v \in \mathcal{N}} d_G(x, v) \leq r$ for any vertex $x \in V(G)$.

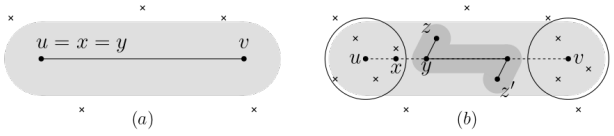


Figure 1: (a) Cross marks failed vertices. The path between $u$ and $v$ is *safe* since no failed vertex lies in the gray area. (b) Illustration of Lemma 3. The path between $u$ and $v$ is *weakly safe*, and there is a *safe* path between net vertices $z$ and $z'$.

If it is clear from the context, we simply call it a *net* and call a vertex in a net a *net vertex*. We can compute an $r$-net of $G$ in $O(m + n \log n)$ time. For a vertex $v \in V(G)$ and a value $c > 0$, the number of net vertices $u$ with $d_G(u, v) \leq cr$ is at most $4(c + 1)^2$.

**Safe Paths and Weakly Safe Paths.** In the design of kernel oracles, a key idea is to decompose a shortest path between $s$ and $s'$ in $G - F$ into subpaths each of which is *safe*, *weakly safe*, or sufficiently short. We define safe paths and weakly safe paths as follows. With a slight abuse of notation, for a path $\gamma$ in $G$, we define $d_G(\gamma, v)$ as the minimum distance in $G$ between vertex $v$ and any vertex on $\gamma$. Let $F$ be a set of failed vertices and $(t, r)$ be a pair of positive parameters.

- $\gamma$ is $(t, r)$-*safe* if $d_G(x_f, \gamma) \geq tr$ for any $x_f \in F$, and
- $\gamma$ is $(t, r)$-*weakly safe* if $\min\{d_G(u, x_f), d_G(v, x_f)\}$ is at most $(2t^2 + 3t + 1)r$ for any $x_f \in F$ such that $d_G(x_f, \gamma)$ is at most $(1 + t)r$, where $u$, $v$ are two end vertices of $\gamma$.

For illustration, see Figure 1. If it is clear from the context, we omit the parameter $(t, r)$. Note that a safe path $\gamma$ is not necessarily weakly safe. There might be a failed vertex $x_f$ such that $d_G(x_f, \gamma)$ lies in $[tr, (1 + t)r)$, but $d_G(u, x_f)$ and $d_G(v, x_f)$ are both at least $(2t^2 + 3t + 1)r$.

Let $\mathcal{N}$ be an $r$-net of $G$. Then the following lemmas hold.

**Lemma 3.** *Assume that all edges of $G$ have length at most $r$. Let $\gamma$ be a $(t, r)$-weakly safe path between two vertices $u$ and $v$ in $G - F$ with $|\gamma| \geq (4t^2 + 8t + 4)r$. Then there is a $(t, r)$-safe path $\gamma'$ between two net vertices $z, z' \in \mathcal{N}$ with $d_G(u, z), d_G(v, z') \leq (2t^2 + 4t + 4)r$ and $|\gamma'| \leq |\gamma|$.*

**Lemma 4.** *Assume that all edges of $G$ have length at most $r$. Let $u$ and $v$ be two vertices of $G - F$ such that $\pi_{G-F}(u, v)$ is neither $(t, r)$-safe nor $(t, r)$-weakly safe. Then there are a vertex $y$ of $\pi_{G-F}(u, v)$ and a net vertex $z \in \mathcal{N}$ such that*

- *$\pi_{G-F}(u, y) \cdot \pi_{G-F}(y, z)$ is $(t, r)$-weakly safe,*
- *$d_{G-F}(y, z) \leq (t^2 + 2t)r$, and*
- *$d_{G-F}(z, v) \leq d_{G-F}(u, v) - t^2 r$.*

Here, $\gamma \cdot \gamma'$ denotes the concatenation of two paths $\gamma$ and $\gamma'$ having a common endpoint. Notice that $\pi_{G-F}(u, y)$ is a subpath of $\pi_{G-F}(u, v)$, which will be used in the proof of Lemma 9.

**Organization of this paper.** Our main ideas lie in the design of kernel oracles. Once a kernel oracle is given, we can answer approximate distance queries immediately. Also, with a path-preserving kernel oracle and an additional data structure, we can answer approximate shortest-path queries efficiently. A kernel oracle consists of substructures called the *FT-structures* with different parameters. If two net vertices $u$ and $v$ are connected by a safe path $\gamma$ of length at most $2t|uv|$, we can find a path of length at most $|\gamma|$ between them using a FT-structure.

In the following, we first describe FT-structures, and then show how to use it to construct a kernel oracle. Finally, we describe an approximate distance oracle and an approximate shortest-path oracle. Recall that $G$ is an $L$-partial $f$-fault tolerant Euclidean $t$-spanner.

## FT-Data Structure

The FT-structure is defined with respect to a pair $(u, v)$ of net vertices and a parameter $W \leq L$. We denote this data structure by $\mathsf{FT}(u, v; W)$. If $W$ is clear in the context, we use $\mathsf{FT}(u, v)$ simply to denote it. For a set $F$ of at most $f$ failed vertices in $G$ with $u, v \notin F$, it allows us to compute a path in $G - F$ between $u$ and $v$ of length at most $|\gamma|$ efficiently, where $\gamma$ is a $(t, W)$-safe path between $u$ and $v$ in $G - F$ if it exists. This structure is a modification of the one introduced in (Chechik et al. 2017). While the work in (Chechik et al. 2017) deals with *failed edges*, we handle *failed vertices*. Since the degree of a vertex can be large, the modification is not straightforward. Moreover, to reduce the space complexity of (Chechik et al. 2017) near linearly, we apply two tricks. While (Chechik et al. 2017) constructs $\mathsf{FT}(u, v)$ for every pair $(u, v)$ of vertices of $G$, we construct $\mathsf{FT}(u, v)$ for every pair $(u, v)$ of net vertices. In addition to this, we construct the data structure on the subgraph $\hat{G}(u, v)$ of $G$ induced by the vertices $p$ with $\max\{|pu|, |pv|\} \leq 2t|uv|$. We will see that this is sufficient for our purpose; this is one of main technical contributions of our paper.

### Construction of $\mathsf{FT}(u, v; W)$

The FT-structure for $(u, v; W)$ is a tree such that each node $\alpha$ corresponds to a subgraph $G_\alpha$ of $\hat{G}(u, v)$ and stores the shortest path $\pi_\alpha$ between $u$ and $v$ of $G_\alpha$. Initially, we let $G_r = \hat{G}(u, v)$ for the root node $r$. In each iteration, we pick a node $\alpha$ whose children are not yet constructed. We decompose $\pi_\alpha$ into *segments* with respect to vertices $u_1, \ldots, u_k$ of $\pi_\alpha$ such that $u_i$ is the farthest vertex from $u$ along $\pi_\alpha$ with $|\pi_\alpha[u, u_i]| \leq i \cdot t \cdot \frac{W}{4}$, where $\pi_\alpha[u, u_i]$ is the subpath in $\pi_\alpha$ between $u$ and $u_i$.

Then we construct the children of $\alpha$ corresponding to the *segments* of $\pi_\alpha$. Let $\eta_{\alpha'}$ be the segment of $\pi_\alpha$ corresponding to a child $\alpha'$ of $\alpha$. For illustration, see Figure 2. To construct $G_{\alpha'}$, we first remove all edges and vertices in $\eta_{\alpha'}$ except $u$ and $v$ from $G_\alpha$. Also, we additionally remove all vertices $p$ with $d_G(u_{\alpha'}, p) \leq t \cdot \frac{W}{4}$ for an arbitrary internal vertex $u_{\alpha'}$ of $\eta_{\alpha'}$. In this way, we can obtain $G_{\alpha'}$, and define $\pi_{\alpha'}$ as a shortest path between $u$ and $v$ in $G_{\alpha'}$. If $\alpha'$ has level $(f + 1)$ in the tree, $u$ and $v$ are not connected in $G_{\alpha'}$, or $\pi_{\alpha'}$ is longer than $2t|uv|$, we set $\alpha'$ as a leaf node of $\mathsf{FT}(u, v; W)$.

In the full version of the paper, we show that each node of $\mathsf{FT}(u, v; W)$ has at most $8t|uv|/W$ children. By construction, observe that the depth of $\mathsf{FT}(u, v; W)$ is at most $(f+1)$. Thus, $\mathsf{FT}(u, v; W)$ has $(8t|uv|/W)^{(f+1)}$ nodes. Note that

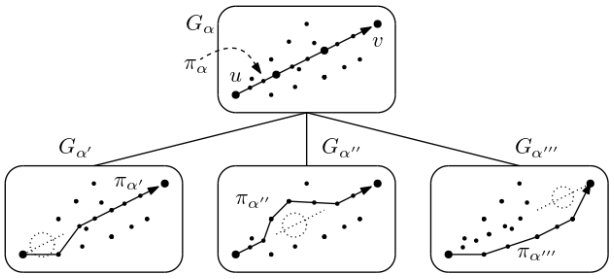


Figure 2: A node $\alpha$ of $\mathsf{FT}(u, v; W)$ has three children, where each child corresponds to a segment of $\pi_\alpha$. Those segments are drawn as dotted line segments. The graph $G_{\alpha'}$ of each child $\alpha'$ is obtained from $G_\alpha$ by removing several vertices around the corresponding segment (vertices contained in the dotted disk). Each node stores the shortest path between $u$ and $v$ on its corresponding graph.

each non-leaf node of $\mathsf{FT}(u, v; W)$ may have more than two children. To traverse the tree efficiently, we use a two-dimensional array so that given a vertex $p$ in $\hat{G}(u, v)$ and a node $\alpha$ in $\mathsf{FT}(u, v; W)$, the child node $\alpha'$ of $\alpha$ with $p \in \eta_{\alpha'}$ can be computed in constant time. We call this array the *assistant array* of $\mathsf{FT}(u, v; W)$.

## Computation of the FT-Path

Given a query of a set $F$ of at most $f$ failed vertices, we can find a node $\alpha^*$ of $\mathsf{FT}(u, v; W)$ such that $\pi_{\alpha^*}$ does not contain any vertex of $F$ as follows. We traverse $\mathsf{FT}(u, v; W)$ starting from the root node $r$. Let $\alpha$ be the current node. If $\pi_\alpha$ contains at least one failed vertex, we visit one of its children $\alpha'$ such that $\eta_{\alpha'}$ contains a failed vertex using the assistant array of $\mathsf{FT}(u, v; W)$. We repeat this process until we reach a node $\alpha^*$ such that either $\pi_{\alpha^*}$ contains no failed vertices, or $\alpha^*$ is a leaf node. If $\pi_{\alpha^*}$ contains no failed vertices, then we return $\pi_{\alpha^*}$ as output. Clearly, the returned path is in $G - F$. We call the returned path the $\mathsf{FT}$-*path* of $G - F$. Otherwise, we reach a leaf node, then we do not return any path.

**Lemma 5.** *Given a set $F$ of at most $f$ failed vertices, we can compute the node of $\mathsf{FT}(u, v; W)$ storing the $\mathsf{FT}$-path of $G - F$ in $O(f^2)$ time, if it exists.*

**Lemma 6.** *Let $F$ be a set of at most $f$ failed vertices and $W$ be a parameter in $[4D, L]$, where $D$ is the longest edge length in $\hat{G}(u, v)$. For two vertices $u, v$ of $V - F$, the $\mathsf{FT}$-path obtained from $\mathsf{FT}(u, v; W)$ with respect to $F$ exists if there is a $(t, W)$-safe path $\gamma$ between $u$ and $v$ in $G - F$ with $|\gamma| \leq 2t|uv|$. Moreover, the $\mathsf{FT}$-path has length at most $|\gamma|$.*

# Kernel Oracle

For a value $\varepsilon > 0$, we construct a kernel oracle that allows us to compute an $(s, s', F; \varepsilon)$-kernel of small complexity for moderately far vertices $s$ and $s'$ and a set $F$ of failed vertices of $G$. In particular, we can compute a kernel of size $O(t^8 f^2)$ in $O(t^8 f^4)$ time, and a path-preserving kernel of size $O(t^8 f^2 \log(tf/\varepsilon') \log^2 n)$ in $O(t^8 f^4 \log(tf/\varepsilon) \log^2 n)$time. Recall that $m = |E|$ and $n = |V|$, and $m \in O(n)$. For an overview of the structure of a kernel oracle, see Figure 3.

The kernel oracle consists of several $\mathsf{FT}$-structures with different parameters $(u, v, W)$. For this purpose, we first choose several values for $W$ such that for any two moderately far vertices $s$ and $s'$, there are at least one value $W'$ with $|ss'| \in [W'/2, W)$. Recall that for two moderately far vertices $s, s'$ in $G$, their Euclidean distance lies in $[L/m^2, L/t) \subseteq [L/(2m^6), L)$. We first decompose the interval $[\frac{L}{2m^6}, L)$ into $(6 \log m + 1)$ intervals $[W_{i-1}, W_i)$, where $W_i = (2^i \cdot W_0)$ for $i \in [1, 6 \log m + 1]$ with $W_0 = L/(2m^6)$. We say two vertices $s$ and $s'$ are *well separated* with respect to $W$ if $|ss'| \in [W/2, W)$. Note that a moderately far vertices are well separated with respect to $W_i$ for at least one index $i \in [1, 6 \log m + 1]$.

## Data Structure

Lemma 3 and Lemma 4 hold only when all edges of $G$ have length at most $r$. To satisfy this condition, we modify $G$ by splitting long edges as preprocessing before constructing the $\mathsf{FT}$-structures. First, we delete the edges in $G$ of length at least $2L$. Since we want to find an $(1 + \varepsilon)$-approximate shortest path or distance between two moderately far vertices, those long edges never participate in a path that we have desired. Next, for each edge $e$ in $G$ of length larger than $(\varepsilon' L)/(4m^6)$ with $\varepsilon' = \frac{\varepsilon}{500t^3(f+1)}$, we split $e$ into subedges of length at most $\frac{\varepsilon'}{4} W_j$, where $W_j$ is the smallest value such that $|e| \in [\frac{\varepsilon'}{4} W_j, 4tW_j]$. This process increases the number of edges by a factor of $O(t/\varepsilon')$. In the following, to avoid confusion, we use $G_0$ to denote the original given graph. Furthermore, we denote the number of vertices and edges in $G$ by $n$ and $m$, respectively.

By construction, it is sufficient to deal with queries of two vertices $s$, $s'$ and a set $F$ of failed vertices in $G$ such that $\{s, s'\} \cup F$ is a subset of $V(G_0)$ and $|ss'| \in [L/m^2, L)$. That is because $[L/m_0^2, L) \subseteq [L/m^2, L)$, where $m_0$ is the number of edges in $G_0$. Notice that $G$ is not always a Euclidean $t$-spanner because of new vertices. However, it has a weaker property stated as follows. It is not difficult to see that this property is sufficient for obtaining Lemma 3 and Lemma 4. Let $\mathcal{V}(e)$ be the set of new vertices of $G$ obtained from splitting an edge $e$ of $G_0$.

**Lemma 7.** *Let $u$ and $v$ be two vertices of $G - F$ with $d_G(u, v) \leq L$ neither of which is contained in the union of $\mathcal{V}(e)$ for all edges $e$ adjacent to the vertices of $F$ in $G_0$. Then $d_{G-F}(u, v) \leq t \cdot d_G(u, v)$.*

We construct $\mathsf{FT}(u, v; \varepsilon' W_j)$ for all indices $j$ with $j \in [1, 6 \log m + 1]$ and all net vertex pairs $(u, v)$ of an $(\varepsilon' W_j)$-net $\mathcal{N}_j$ with $|uv| \leq (1+\varepsilon)tW_j$. Here, the nets $\mathcal{N}_j$ we use must be aligned, that is, $\mathcal{N}_j \subseteq \mathcal{N}_{j'}$ for any two indices $j$ and $j'$ with $j \geq j'$. While the work in (Chechik et al. 2017) construct the $\mathsf{FT}$-structure for all pairs of vertices, we construct the $\mathsf{FT}$-structure only for pairs $(u, v)$ of net vertices. In this way, we can improve the space complexity near-linearly. However, it requires us to design a new algorithm to handle query vertices that are not net vertices.

**Lemma 8.** *The space complexity of all $\mathsf{FT}$'s and their assistant arrays is $(t/\varepsilon')^{O(f)} \cdot n \log n$. Furthermore, we can compute all of them in $(t/\varepsilon')^{O(f)} \cdot n \log^2 n$ time.*

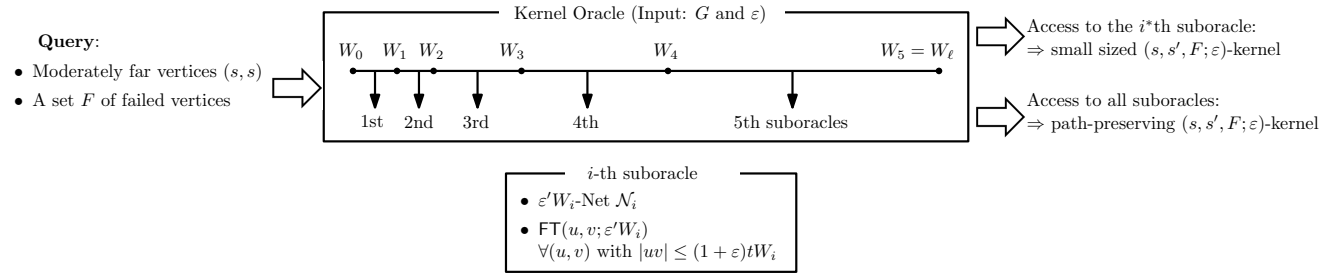


Figure 3: Overview of a kernel oracle for moderately far vertices.

## Kernel Query

Now we describe how to compute an $(s, s', F; \varepsilon)$-kernel $H$ of size $O(t^8 f^2)$ for a pair $(s, s')$ of moderately far vertices and a set $F$ of at most $f$ failed vertices given as a query. Let $i^*$ be the index such that $s$ and $s'$ are well separated with respect to $W_{i^*}$. Here, we look at the $\mathsf{FT}$-structures constructed with respect to $W_{i^*}$ only. For a point $x$ in the plane, let $\mathrm{N}(x)$ be the net vertex in $\mathcal{N}_{i^*}$ closest to $x$.

The kernel $H$ is constructed as follows. The vertices of $H$ are $s$, $s'$, $\mathrm{N}(s)$, $\mathrm{N}(s')$, and all net vertices $u$ in $\mathcal{N}_{i^*}$ with $d_G(u, F) \leq (4t^2 + 8t + 5)\varepsilon' W_{i^*}$, where $d_G(u, F)$ is the minimum of $d_G(u, x_f)$ for all $x_f \in F$. For two vertices $u$ and $v$ in $V(H)$, we add $uv$ as an edge of $H$ if the $\mathsf{FT}$-path $\pi$ exists for $\mathsf{FT}(u, v'; \varepsilon' W_{i^*})$ with respect to $F$, or $d_G(u, v) \leq (4t^2 + 8t + 5)\varepsilon' W_{i^*}$. For the former case, we set $w_H(uv) = |\pi|$. For the latter case, we set $w_H(uv) = 40t^3 \varepsilon' W_{i^*}$.

**Lemma 9.** *A $(s, s', F; \varepsilon)$-kernel of size $O(t^8 f^2)$ can be computed in $O(t^8 f^4)$ time for two well separated vertices $s$, $s'$ with respect to $W_i$ for $i \in [1, 6 \log m + 1]$ and a set $F$ of at most $f$ failed vertices given as a query.*

*Sketch of the proof.* For easy understanding, we assume that $s$ and $s'$ are net vertices in $\mathcal{N}_{i^*}$, and thus $\mathrm{N}(s) = s$ and $\mathrm{N}(s') = s'$. Since an analysis of the query time and the size of the kernel is straightforward, we only show that $H$ is indeed a $(s, s', F; \varepsilon)$-kernel. For clarity, we omit the parameters for safe and weakly safe paths: safe and weakly safe paths mean $(t, \varepsilon' W_{i^*})$-safe and $(t, \varepsilon' W_{i^*})$-weakly safe paths, respectively. We have $W_{i^*} \in (|ss'|, 2|ss'|]$ since $s$ and $s'$ are well separated with respect to $W_{i^*}$.

We first show that if a short safe or weakly safe path between two vertices $u$ and $v$ in $G - F$ exists, then their distance in $H$ approximates the length of a shortest safe or weakly safe path in $G - F$ with a small additive error. In this case, we call a shortest path in $G - F$ between $u$ and $v$ a *base path*. For each edge of $H$, either it corresponds to an $\mathsf{FT}$-path, or it has small weight. By Lemma 6, for any $u$ and $v$ in $V(H)$ and a safe path $\gamma$ in $G - F$ between them, we have $d_H(u, v) \leq |\gamma|$. Moreover, Lemma 3 says that if a weakly safe path $\gamma$ exists in $G - F$ between $u$ and $v$, then there are three edges $uz$, $zz'$, and $z'v$ in $H$ such that $uz$ and $z'v$ have small weights, and there exists a safe path between $zz'$. Refer to Figure 1. This implies that $d_H(u, v)$ is at most the sum of $|\gamma|$ and the two small weights $w_H(uz)$ and $w_H(z'v)$. This implies the claim.

Then we show that there is a path in $G - F$ between $s$ and $s'$ of length at most $(1 + \varepsilon) d_{G-F}(s, s')$ consisting of base paths. This implies that $d_H(s, s') \leq (1 + \varepsilon) d_{G-F}(s, s')$ as each base path corresponds to a path in $H$. Let $\pi_{G-F}(s, s')$ be a shortest path in $G - F$ between $s$ and $s'$. If this shortest path is safe or weakly safe, then it is already a base path, and thus we are done. Otherwise, we apply Lemma 4 to $\pi_{G-F}(s, s')$. Then we have two vertices $y$ and $z$, and consider the concatenation of $\pi_{G-F}(s, y)$, $\pi_{G-F}(y, z)$, and $\pi_{G-F}(z, s')$. Note that both $y$ and $z$ are vertices of $G$ since $V(H)$ contains the net vertices in $\mathcal{N}_{i^*}$ lying close to $s, s' \cup F$. The length of the concatenation is at least the length of $\pi_{G-F}(s, s')$. The difference (additive error) is at most $2t \cdot \varepsilon' W_{i^*}$ by Lemma 4. Since $\pi_{G-F}(s, y) \cdot \pi_{G-F}(y, z)$ is already a base path, it suffices to consider $\pi_{G-F}(z, s')$ recursively. We can show that the recursive step occurs at most $f$ times as the vertices $z$ we obtained for the recursion steps have distinct closest failed vertices. Therefore, the additive error is at most $2t\varepsilon' W_{i^*} f$, and thus the resulting path has length at most $(1+\varepsilon) d_{G-F}(s, s')$ since $W_{i^*} \in (|ss'|, 2|ss'|]$. □

## Path-Preserving Kernel Query

Now we describe how to compute a path-preserving $(s, s', F; \varepsilon)$-kernel $H$ of size for a pair $(s, s')$ of moderately far vertices and a set $F$ of at most $f$ failed vertices given as a query. Let $i^*$ be the index with $|ss'| \in [W_{i^*}/2, W_{i^*})$. The kernel $H_0$ we constructed before is not necessarily path-preserving. For an edge $uv$ added to $H_0$ due to its corresponding $\mathsf{FT}$-path, we can compute the $\mathsf{FT}$-path in time linear in its complexity using $\mathsf{FT}(u, v; \varepsilon' W_{i^*})$. On the other hand, an edge $uv$ added to $H_0$ because of its small length (i.e., $d_G(u, v) \leq (4t^2 + 8t + 5)\varepsilon' W_{i^*}$) can violate the path-preserving property of $H_0$. For such an edge, by the spanner property of $G$, $d_{G-F}(u, v)$ is at most $40t^3 \varepsilon' W_{i^*}$, and this is why we set the weight of $uv$ as $40t^3 \varepsilon' W_{i^*}$. However, although there is a path of $G - F$ of length at most $40t^3 \varepsilon' W_{i^*}$, we do not know how to compute it efficiently.

To obtain a path-preserving kernel, we first compute a $(u, v, F; \varepsilon)$-kernel for the violating edges $uv$ of $H_0$, and take the union of them together with $H_0$. Since $d_G(u, v) \leq (4t^2 + 8t + 5)\varepsilon' W_{i^*}$, $|uv|$ is significantly smaller than $|ss'|$, and thus the value $W_i$ with $|uv| \in [W_i/2, W_i]$ is smaller. Although there might still exist violating edges $u'v'$, the distance in $G$ between $u'$ and $v'$ becomes smaller. Then we repeat this procedure until for any violating edge $uv$, their distance in $G$ becomes at most $tL/m^6$. In this case, $uv$ is not violating the path-preserving kernel anymore due to the second condition for path-preserving kernels. Although this

recursive description conveys our intuition effectively, it is more convenient to describe it in an integrated way as follows for formal proofs.

The construction of $H$ works as follows. The vertices of $H$ are $s$, $s'$, and all net vertices $u$ from $\mathcal{N}_j$ with $d_G(u, F \cup \{s, s'\}) \leq (4t^2 + 8t + 5)\varepsilon' W_j$ for some index $j \in [1, i^*]$. The edge set of $H$ comprises the edges from the $(p, q, F; \varepsilon)$-kernels constructed by the previous query algorithm for all indices $j \in [1, i^*]$ and two well separated net vertices $p, q \in \mathcal{N}_j \cap V(H)$ with respect to $W_j$.

**Lemma 10.** *If an edge $uv$ lies in a shortest path in $H$, either it corresponds to an FT-path, or $d_{G-F}(u, v) \leq tL/m^6$.*

Lemma 10 together with the fact that $H$ is a supergraph of the $(s, s', F; \varepsilon)$-kernel $H_0$ we constructed before implies that $H$ is a path-preserving $(s, s', F; \varepsilon)$-kernel. The query time and the size of $H$ are analyzed in the full version.

**Lemma 11.** *For a query of two moderately far vertices $s$, $s'$ and a set $F$ of at most $f$ failed vertices, a path-preserving $(s, s', F; \varepsilon)$-kernel of size $O(t^8 f^2 \log(t/\varepsilon') \log^2 n)$ can be computed in $O(t^8 f^4 \log(t/\varepsilon') \log^2 n)$ time.*

In the preprocessing step, We split long edges. This increases the number of its vertices and edges by a factor of $O(t/\varepsilon') \subseteq \mathrm{poly}\{t, f, 1/\varepsilon\}$. So far, we use $n$ and $m$ to denote the numbers of vertices and edges, respectively, of the resulting graph. To get the final results in Table 1 (for kernel and path-preserving kernel oracles) with respect to the complexity of the original input graph, we simply replace both $n$ and $m$ with $(t/\varepsilon') \cdot m \in (tf/\varepsilon)^{O(1)} \cdot n$.

## Distance / Shortest Path Oracles

In this section, we construct approximate distance oracle and shortest path oracle for moderately far vertex queries. Let $G$ be an $L$-partial $f$-fault-tolerant Euclidean $t$-spanner. Here, $G$ might have long edges as the preprocessing step mentioned before only spans the previous section. Let $n$ and $m$ denote the numbers of vertices and edges of $G$, respectively.

For an approximate distance oracle, it suffices to construct a kernel oracle. Recall that the approximation factor $\varepsilon$ must be give in the construction of the oracles. Given two moderately far vertices $s$ and $s'$ and a set $F$ of failed vertices, we simply compute an $(s, s', F; \varepsilon)$-kernel $H$, and then compute the distance between $s$ and $s'$ in $H$ using Dijkstra's algorithm. This value is an approximate distance between $s$ and $s'$ by the definition of kernels. Therefore, the performance of our distance oracle is the same as the one for a kernel oracle as stated in Table 1.

For an approximate shortest-path oracle, we construct a path-preserving kernel oracle. Given two moderately far vertices $s$ and $s'$, and a set $F$ of failed vertices, we simply compute a path-preserving $(s, s', F; \varepsilon)$-kernel $H$, and then compute a shortest path $\pi$ between $s$ and $s'$ in $H$ using Dijkstra's algorithm. Since $\pi$ might contain an edge not in $G - F$, we replace each edge of $\pi$ with their corresponding path in $G - F$. More specifically, for an edge $uv$ of $\pi$, either its length is at most $tL/m^6$, or there is an FT-path between $u$ and $v$ of length $w_H(uv)$. For the former case, we replace $uv$ with an arbitrary path of $G - F$ consisting of edges of length at most $tL/m^6$. By the spanner property, $u$ and $v$ are connected in $G - F$, and moreover, a shortest path between them consists of edges of length at most $tL/m^6$. For the latter case, we simply replace $uv$ with the FT-path.

We have two issues here. First, we have to show how to compute an arbitrary path of $G - F$ consisting of edges of length at most $tL/m^6$. Second, this replacement increases the length of $\pi$ by $tL/m^5$. We have to argue that this value is negligible for our purpose.

We handle the first issue by constructing a fault-tolerant connectivity oracle introduced in (Duan and Pettie 2017). Given a set of failed vertices and two query vertices, it allows us to check if the two query vertices are connected in the graph in the presence of the failed vertices. By slightly modifying this data structure, we can report an arbitrary path as well. More specifically, we construct this data structure on the subgraph of $G$ induced by edges of length at most $tL/m^6$. The size of this data structure is $O(fm \log n \log\log n)$. It can be computed in $O(mn \log n)$ time, and its query time is $O(f^4 \log^2 n \log\log n)$ plus the number of returned edges.

The second issue can be handled using the fact that $s$ and $s'$ are moderately far. Even if the replacement increases the length of $\pi$, the returned path has length at most $(1 + 2\varepsilon) d_{G-F}(s, s')$ as proved below.

**Lemma 12.** *The returned path has length at most $(1 + 2\varepsilon) d_{G-F}(s, s')$.*

*Proof.* Since $s$ and $s'$ are moderately far, $|ss'| \in [L/m^2, L/t)$. The total weight of all edges in $G - F$ of length at most $tL/m^6$ is at most $tL/m^5$. For sufficiently large $m$ with $m \in \Omega(t/\varepsilon)$, the following holds.

$$L/m^5 \leq d_{G-F}(s, s')/m^3 \leq \frac{\varepsilon}{t} d_{G-F}(s, s'). \qquad (1)$$

The length of the returned path is at most $d_H(s, s') + tL/m^5$, and the distance $d_H(s, s')$ is at most $(1+\varepsilon) d_{G-F}(s, s')$ since $H$ is a $(s, s', F; \varepsilon)$-kernel of $G$. Thus, the lemma holds by Inequality 1. □

In this way, we can use the path-preserving kernel oracle to compute an approximate shortest path correctly.

## Conclusion

In this paper, we presented efficient approximate distance and shortest-path oracles for an $f$-fault-tolerant Euclidean $t$-spanner and a value $\varepsilon > 0$. Although this is the first near-linear-sized approximate shortest-path oracle for graphs with vertex failures, one might think that it is still not practical because of large hidden constants in the performance guarantees. Although it seems hard to avoid the exponential dependency on $t$ and $f$ in the oracle sizes theoretically, we believe that it can be made more efficient in practice by applying several optimization tricks. This is indeed one of interesting directions for future work; our work is just a starting point. We hope that our work would be a stepping stone towards bridging the gap between theory and practice in the routing problem for dynamic networks.

## Acknowledgements

We appreciate to the anonymous reviewers for valuable comments and suggestions. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.RS-2023-00209069)

## References

Abam, M. A.; de Berg, M.; Farshi, M.; and Gudmundsson, J. 2009. Region-fault tolerant geometric spanners. *Discrete & Computational Geometry*, 41(4): 556–582.

Abraham, I.; Delling, D.; Fiat, A.; Goldberg, A. V.; and Werneck, R. F. 2016. Highway dimension and provably efficient shortest path algorithms. *Journal of the ACM (JACM)*, 63(5): 1–26.

Abraham, I.; Delling, D.; Goldberg, A. V.; and Werneck, R. F. 2011. A hub-based labeling algorithm for shortest paths in road networks. In *Experimental Algorithms: 10th International Symposium, SEA 2011, Kolimpari, Chania, Crete, Greece, May 5-7, 2011. Proceedings 10*, 230–241. Springer.

Bast, H.; Funke, S.; Sanders, P.; and Schultes, D. 2007. Fast routing in road networks with transit nodes. *Science*, 316(5824): 566–566.

Bilò, D.; Cohen, S.; Friedrich, T.; and Schirneck, M. 2021. Space-Efficient Fault-Tolerant Diameter Oracles. In *Proceedings of the 46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*.

Bilò, D.; Grandoni, F.; Gualà, L.; Leucci, S.; and Proietti, G. 2015. Improved purely additive fault-tolerant spanners. In *Proceedings of the 23th Annual European Symposium on Algorithms (ESA 2015)*, 167–178. Springer.

Blum, J.; Funke, S.; and Storandt, S. 2018. Sublinear Search Spaces for Shortest Path Planning in Grid and Road Networks. In McIlraith, S. A.; and Weinberger, K. Q., eds., *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 6119–6126.

Bodwin, G.; Choudhary, K.; Parter, M.; and Shahar, N. 2020. New Fault Tolerant Subset Preservers. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*.

Bodwin, G.; Dinitz, M.; Parter, M.; and Williams, V. V. 2018. Optimal vertex fault tolerant spanners (for fixed stretch). In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, 1884–1900.

Bose, P.; Dujmovic, V.; Morin, P.; and Smid, M. 2013. Robust Geometric Spanners. *SIAM Journal on Computing*, 42(4): 1720–1736.

Braverman, M. 2022. Fixed-Parameter Sensitivity Oracles. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ITCS 2022)*, volume 215, 23.

Buchin, K.; Har-Peled, S.; and Oláh, D. 2020. A spanner for the day after. *Discrete & Computational Geometry*, 64(4): 1167–1191.

Buchin, K.; Har-Peled, S.; and Oláh, D. 2022. Sometimes reliable spanners of almost linear size. *Journal of Computational Geometry*, 13(1): 178–196.

Charalampopoulos, P.; Mozes, S.; and Tebeka, B. 2019. Exact distance oracles for planar graphs with failing vertices. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2019)*, 2110–2123.

Chechik, S.; Cohen, S.; Fiat, A.; and Kaplan, H. 2017. $(1+\epsilon)$-Approximate $f$-Sensitive Distance Oracles. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, 1479–1496. SIAM.

Chechik, S.; Langberg, M.; Peleg, D.; and Roditty, L. 2009. Fault-tolerant spanners for general graphs. In *Proceedings of the forty-first annual ACM symposium on Theory of computing (STOC 2009)*, 435–444.

Czumaj, A.; and Zhao, H. 2004. Fault-tolerant geometric spanners. *Discrete & Computational Geometry*, 32(2): 207–230.

Dinitz, M.; and Krauthgamer, R. 2011. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing (PODC 2011)*, 169–178.

Dinitz, M.; and Robelle, C. 2020. Efficient and simple algorithms for fault-tolerant spanners. In *Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC 2020)*, 493–500.

Duan, R.; Gu, Y.; and Ren, H. 2021. Approximate distance oracles subject to multiple vertex failures. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA 2021)*, 2497–2516. SIAM.

Duan, R.; and Pettie, S. 2017. Connectivity Oracles for Graphs Subject to Vertex Failures. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, 490–509.

Filtser, A.; and Le, H. 2022. Locality-sensitive orderings and applications to reliable spanners. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*, 1066–1079.

Gudmundsson, J.; Levcopoulos, C.; Narasimhan, G.; and Smid, M. 2008. Approximate distance oracles for geometric spanners. *ACM Transactions on Algorithms (TALG)*, 4(1): 1–34.

Kosowski, A.; and Viennot, L. 2017. Beyond highway dimension: small distance labels using tree skeletons. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, 1462–1478. SIAM.

Kuhn, F.; Moscibroda, T.; Nieberg, T.; and Wattenhofer, R. 2005. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In *Proceedings of the Distributed Computing: 19th International Conference (DISC 2005)*, 273–287. Springer.

Levcopoulos, C.; Narasimhan, G.; and Smid, M. 1998. Efficient algorithms for constructing fault-tolerant geometric spanners. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing (STOC 1998)*, 186–195.

Levcopoulos, C.; Narasimhan, G.; and Smid, M. 2002. Improved algorithms for constructing fault-tolerant spanners. *Algorithmica*, 32(1): 144–156.

Narasimhan, G.; and Smid, M. 2007. *Geometric Spanner Networks*. Cambridge University Press.

Oh, E. 2020. Shortest-Path Queries in Geometric Networks. In *Proceedings of the 31st International Symposium on Algorithms and Computation (ISAAC 2020)*.

Ouyang, D.; Yuan, L.; Qin, L.; Chang, L.; Zhang, Y.; and Lin, X. 2020. Efficient shortest path index maintenance on dynamic road networks with theoretical guarantees. *Proceedings of the VLDB Endowment*, 13(5): 602–615.

Parter, M. 2022. Nearly optimal vertex fault-tolerant spanners in optimal time: sequential, distributed, and parallel. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2022)*, 1080–1092.

Ren, H. 2022. Improved distance sensitivity oracles with subcubic preprocessing time. *Journal of Computer and System Sciences*, 123: 159–170.

Sommer, C. 2014. Shortest-Path Queries in Static Networks. *ACM Comput. Surv.*, 46(4).

Thorup, M.; and Zwick, U. 2005. Approximate Distance Oracles. *J. ACM*, 52(1): 1–24.

van den Brand, J.; and Saranurak, T. 2019. Sensitive distance and reachability oracles for large batch updates. In *IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS 2019)*, 424–435. IEEE.

Zhang, M.; Li, L.; Hua, W.; Mao, R.; Chao, P.; and Zhou, X. 2021. Dynamic hub labeling for road networks. In *Proceedings of the IEEE 37th International Conference on Data Engineering (ICDE 2021)*, 336–347. IEEE.

Zhang, M.; Li, L.; and Zhou, X. 2021. An experimental evaluation and guideline for path finding in weighted dynamic network. *Proceedings of the VLDB Endowment*, 14(11): 2127–2140.