Delineating imprecise regions via shortest-path graphs

Citation for published version (APA):

DOI:
10.1145/2093973.2094010

Document status and date:
Published: 01/01/2011

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
ABSTRACT
An imprecise region, also called a vernacular region, is a region without a precise or administrative boundary. We present a new method to delineate imprecise regions from a set of points that are likely to lie inside the region. We use shortest-path graphs based on the squared Euclidean distance which capture the shape of region boundaries well. Shortest-path graphs naturally adapt to point sets of varying density, and they are always connected. As opposed to neighborhood graphs, they use a non-local criterion to determine which points to connect. Furthermore, shortest-path graphs can easily be extended to take geographic context into account by modeling context as "soft" obstacles. We present efficient algorithms to compute shortest-path graphs with or without geographic context. We experimentally evaluate the quality of the imprecise regions computed with our method. To fairly compare our results to those obtained by the common KDE approach, we also show how to integrate context into KDE by again using soft obstacles.

Categories and Subject Descriptors
I.5.1 [Pattern Recognition]: Models—Geometric; H.2.8 [Database Management]: Database Applications—Spatial Databases and GIS; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

General Terms
Algorithms, Theory

Keywords
Vernacular regions, Shortest-path graph, Geographic context

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM SIGSPATIAL GIS ’11, November 1-4, 2011. Chicago, IL, USA

1. INTRODUCTION
In everyday language, people tend to use vernacular regions to describe places of interest. Vernacular regions are regions without a precise or administrative boundary. Other used terms are vague or imprecise regions. Examples are “downtown Chicago” and “the English Midlands”. Whereas humans are often able to interpret such descriptions intuitively, this is not straightforward for a computer: a formalization is needed to model vernacular regions such that, for example, search engines can give better results on queries like “hotels in the English Midlands”. A common approach is to use the internet as a source of information, using search engines to relate exact geographic locations to vernacular regions. The question arises how to find the (approximate) boundary of a region, given a set of points that are likely to lie inside it.

Overview and results. We start in Section 2 by listing a number of desirable properties of methods that aim to delineate vernacular regions, and by discussing previous work. The discussion shows that none of the existing methods has all desirable properties. In particular, most methods have difficulty in dealing with sample sets of varying density and/or in dealing with regions containing holes. In Section 3 we describe a new approach to compute vernacular regions, which is based on shortest-path graphs under the squared Euclidean distance. These graphs capture the shape of the regions well, even when the sampling density varies. Shortest-path graphs do not require any parametrization and are always connected. In Section 4 we explain how to compute shortest-path graphs and how to turn them into a delineation for a vernacular regions. This includes a method to detect outliers and to detect holes in the region. In some situations, there may be geographic context information available about the presence of rivers, roads, etcetera, and such information can be relevant when computing vernacular regions. We therefore describe in Section 5 how context information can be incorporated into our method, by modeling context as “soft” obstacles. In Section 6, we evaluate our method in a brief experimental study based on real data obtained from geotagged photos on Flickr (www.flickr.com). We compare our results to the common KDE method. For fair comparison of results, we also show how to integrate context into KDE by again using soft obstacles.

2. VERNACULAR REGIONS
In this section, we first discuss desirable properties of methods that aim to delineate a vernacular regions. After that, we discuss related work with respect to these properties.
2.1 Desirable properties

We discuss seven desirable properties of vernacular regions and the methods to compute them.

**Connected region.** An important consideration is whether a vernacular region is required to be connected or not. Some existing methods explicitly take this requirement into account [21], others explicitly do not use it [8]. We believe that in most cases connectedness is a desirable property and, hence, a method should be capable of guaranteeing it. Ideally, the method should be such that one can also deviate from this default, if so desired by the user.

**Support for holes.** Often a vernacular region is simply connected—that is, it does not contain holes—but this is not always the case. Think of regions like “the Irish coast” or “the Parisian suburbs”. Thus one would like to have a method that can support holes in a natural manner.

**Fuzzy boundary.** The result of a method should be able to represent the imprecision in the delineation of a vernacular region. A simple polygonal representation (a crisp boundary) is sometimes used. This has the advantage of simplicity. However, it does not capture the imprecision of the boundary. On the other hand, fuzzy boundaries are also used, a representation in which every point in the plane is assigned a certain value to indicate the certainty or degree of membership. However, storing such a representation can be more difficult. An intermediate solution divides the plane into multiple regions, where each region has a certainty or degree of membership. Often, three (not necessarily connected) regions are used: “certainly inside”, “certainly outside”, and “potentially inside”. This model is referred to as the egg-yolk model [4]. We consider such a representation connected if the yolk (“certainly inside”) is a connected region and the yolk and egg-white (“potentially inside”) together are also a connected region. Simple examples of these three representations are given in Fig. 1.

![Figure 1](image_url)

**Figure 1:** Three ways to represent an imprecise boundary. (a) Crisp boundary. (b) Fuzzy boundary. (c) Egg-yolk model.

**Positive samples.** Input is commonly obtained from the web by querying search engines for the vernacular region in combination with trigger phrases. By grounding the found references to precise places, one obtains a number of sample points which are likely to be inside or near the intended region. These samples are called positive. Some approaches also use negative samples, that are not inside the vernacular region. However, these points are often hard to obtain accurately. Therefore, a method that does not need negative samples is preferable.

**Varying density.** Depending on the source of the input data, the sample density may vary strongly within a vernacular region. For example, tourist attractions, area of high population density, or area of political power may cause a high local sample density in comparison to other places of the same vernacular region [20]. Hence, methods should be able to cope with varying density.

**Outliers.** Due to inaccurate or incorrect information on the web, data sets often have outliers, samples that claim to be inside the vernacular region, but in fact are not. It is important to filter these outliers, either explicitly (e.g. by preprocessing) or implicitly (i.e. the method automatically deals with outliers).

**Geographic context.** Vernacular regions are often affected by geographic elements, such as administrative borders, shorelines, roads, and rivers. For example, “southern Limburg” in the Netherlands is affected by the administrative boundary of Limburg; the Parisian suburbs are affected by the ring road of Paris, the Boulevard Périphérique. The inclusion of geographic context may result in partially crisp boundaries for a vague region (see Fig. 2). In some situations, this may be desirable [5]. Therefore, a method should be able to take this geographic context into account. However, methods should not rely on such information, as it may not always be available, accurate, or even suitable.

![Figure 2](image_url)

**Figure 2:** Southern Limburg has a partially crisp boundary.

We assume that geographic context is represented by a set of line segments. This is appropriate for the geographic elements mentioned above—rivers, roads, borders—but less suitable for some other context information, such as terrain elevation. When searching for “the Alps”, one will find samples mostly located in the mountains (having a high terrain elevation) and hence regions with low terrain elevation are unlikely to be part of “the Alps”. Also, the type of terrain could be included, for example urban area, agricultural area and natural area. When searching for “downtown Chicago”, one will mostly find points located in urban areas and hence points that are, for example, in an agricultural area are likely to be outliers. However, this information assigns properties to regions in the plane, distinguishing them from the linear elements, such as roads and rivers. For this reason, we do not include this type of geographic context in our method, leaving incorporation of these other types of context as future work.

2.2 Related work

There is a large collection of work available on vernacular regions. There are roughly four categories of research, though most of the work belongs to multiple categories.
The first category concerns user studies: how do people deal with imprecise regions and what do they expect? Montello et al. [19] performed a study asking people in Santa Barbara to delineate “downtown” on a map. One of their observations is that the “core” of the region is not necessarily the geometric center. The vagueness of “downtown” seems to be intuitive to most interviewed people when asked to delineate with 50% and 100% certainty. Recently, Davies et al. [5] also performed a user study, concluding that the best way to model an imprecise region depends on the type of region and the expectations of the user.

The second category discusses ways to represent a vernacular region. As mentioned in the previous subsection, common representations are fuzzy boundaries [10, 13, 20], crisp boundaries [21], and the egg-yolk model [4, 8]. Each approach has its own advantages and disadvantages.

The third category focuses on methods to obtain sample data from the web. Typically, data is obtained by ground-
ing results from search engines to geographic locations [10, 12, 13, 20]. This approach uses trigger phrases, such as “* in the English Midlands”, to find items that are likely located in the intended vernacular region. The effectiveness of this approach was studied by Jones et al. [13] and Grothe and Schaab [10]. Data obtained this way often contains numerous outliers, due to either incorrect information on the web or incorrect grounding. Purves et al. [20] identified this problem and observed that also the search engine used affects the results: English documents about Scotland are likely to be located in areas of dense population and politi-
cal influence, while French and German documents typically are tourist information and cover a larger area of Scotland.

The fourth category develops methods to derive a region boundary from a set of samples. Our method belongs to this category. A common approach applied in literature is Kernel Density Estimation (KDE) [10, 13, 20]. This method assigns a certainty to every point in the plane by calculating the influence of nearby samples on that point using a kernel function. The advantage of this approach is its simplicity. However, it does not include geographic context, nor can it guarantee a connected region. Grothe and Schaab [10] applied a Support Vector Machine and compared their approach to the KDE method, concluding that their method outperforms KDE. However, their method cannot deal with context or holes, nor does it guarantee a connected region. Alani et al. [1] applied Voronoi diagrams to delineate imprecise regions. However, this method heavily relies on negative samples and a database containing information on vernacular regions. For some countries this data may be avail-
able, but it is a severe restriction for worldwide application. The dual of a Voronoi diagram, a Delaunay triangulation, is also used to delineate vague regions [21]. However, this approach again relies on negative samples. Reinbacher et al. [21] applied a method based on a-shapes. An a-shape is used to find an initial estimate of the region and this region is adapted based on the negative samples. The main issues with this approach are that it can neither deal with context nor with strongly varying density.

There is a large collection of work available on shape recognition, a related area. Given a set of points, an approx-
imate shape is calculated. However, often an assumption is made on the density of the input points. An exam-
ple is the dot pattern as assumed by Chaudhuri et al. [3]. Such a pattern is a near-uniform distribution of points in the intended region or object to be found. As discussed, such assumptions are too strict for delineating imprecise re-
gions. Besides a density assumption, outliers are often also not considered, as is for example the case for the χ-shapes presented by Duckham et al. [7]. Several methods exist that try to capture the neighborhood of a vertex by connecting it to its neighbors. In particular, we point out the Gabriel graph [9] and the more generic β-skeleton [15]. These meth-
ods apply a local criterion to be able to deal with varying density. This locality can cause issues. Since there is a spe-
cial relation between the Gabriel graph and our method, we treat this in more detail in Section 3.3.

3. SHORTEST-PATH GRAPH

The main element of our method is the shortest-path graph using the squared Euclidean distance. We use it to compute an initial crisp delineation of a vernacular region. In this section, we describe how this graph is constructed and relate it to the well-known Gabriel graph.

We assume that the input is a set S of positive samples in R². For now, we assume that S does not contain outliers. Our basic approach is to first compute a “skeleton” of the vernacular region, which is the above-mentioned shortest-path graph on the given point set, and which captures the connectivity of the region. The idea is to choose the edges of the graph in such a way that shortest paths within the vernacular region roughly correspond to shortest paths in the graph. The question is which edges to include in the graph in order to achieve this. Adding all edges is clearly undesir-
able: this would assume that the shortest path between any two points in the vernacular region is a straight-line segment and, hence, imply that the vernacular region is convex. The next option is to include only short edges into the graph. The intuition is that if a path consists of short edges then this gives evidence that the path is in the vernacular re-
gion. This, however, would require a threshold to determine which edges are deemed short and the method would not adapt well to varying density. An elegant solution to this problem is to use the shortest-path graph under the squared Euclidean distance, as explained next.

First we construct a complete graph G_s on S, that is, a graph in which there is an edge between any pair of points. We assign each edge (p, q) a weight, which is the square of the Euclidean distance between p and q. Next we construct the shortest-path graph G∗ on the weighted graph G_s: the graph G∗ contains an edge (p, q) if and only if (p, q) is the shortest path between p and q in G_s. Note that this is equivalent to saying that we include (p, q) if and only if (p, q) is used in some shortest path in G_s. The use of the squared Euclidean distance means that a sequence of short edges can be preferable over a single long edge, even if the sum of the (unsquared) edge lengths in the sequence is more than the length of the long edge. As a result, the shortest-path graph under the squared Euclidean distance nicely captures the shape of the sample points, even in the case of varying density—see Fig. 3 and Fig. 5(a) for examples. Note that

1 instead of using |pq|^2 as the weight of the edge (p, q), we could also use |pq|^t, for some power t ≥ 1. For t = 1 the shortest-path graph is just the complete graph, while for t = ∞ the shortest-path graph reduces to the minimum spanning tree (if the points are in general position). Using t = 2 gives good results and the resulting graph has some nice properties, so this is the value we will use.
the shortest-path graph is necessarily connected. A rooted variant of shortest-path graphs (with arbitrary weights) has been used before to solve a different problem, namely that of connectivity queries in an online setting [14, 22].

In the next section, we explain how we obtain a vernacular region from the shortest-path graph. First, however, we investigate its properties. In particular, we show that the shortest-path is a subset of the Gabriel graph and, hence, of the Delaunay triangulation. This implies that the shortest-path graph is planar, and also that we can base its computation on the Delaunay triangulation, which has only a linear number of edges. This is a major improvement on the complete graph, which has a quadratic number of edges.

**Relation to Gabriel graph.** The Gabriel graph [9] is defined as a graph on a point set, where an edge between two vertices exists if and only if the open disk that has the vertices as diametrical points is empty. From the Law of Cosines, it trivially follows that

\[ d^2(p, q) \leq d^2(p, r) + d^2(r, q) \]

and only if \( \angle prq \leq \frac{\pi}{2} \). That is, the edge \((p, q)\) cannot exist in the shortest-path graph \( G' \) if there is a vertex \( r \) such that the angle \( \angle prq \) is greater than \( \frac{\pi}{2} \). From Thales’ Theorem, we know that any such vertex must lie within the circle of which \( p \) and \( q \) are diametrical points. Therefore, any edge of the shortest-path graph with squared Euclidean distance must be present in the Gabriel graph: the shortest-path graph is thus a subset of the Gabriel graph. Since the Gabriel graph is a subset of the Delaunay triangulation, the shortest-path graph is also a subset of the Delaunay triangulation.

The Gabriel graph uses a local criterion, in contrast to the non-local criterion applied by the shortest-path graph. The existence of an edge for the Gabriel graph depends only on the smallest enclosing circle of the edge. However, the criterion of the shortest-path graph does not suffer from this locality as it is based on paths. Fig. 4 illustrates this effect. This feature of the shortest-path graphs is of particular interest when dealing with regions that have large holes or indents, such as regions that follow the shoreline of a lake or bay, or the suburbs of a city enclosing the city center. A comparison between the shortest-path graph and the Gabriel graph for Lake Michigan is given in Fig. 5. Outliers were coarsely filtered beforehand. The Gabriel graph wrongfully connects the opposite sides of the lake multiple times. The shortest-path graph neatly traces the shoreline, though skipping over some bays. One can argue that removing “long” edges could alleviate the problem. However, this approach cannot easily cope with varying density and it introduces an additional parameter, one that can be avoided by using the shortest-path graph.

The \( \beta \)-skeleton, although more general than the Gabriel graph, still uses a local criterion to select edges and, hence, suffers from similar problems as the Gabriel graph [18].

### 4. COMPUTING VERNACULAR REGIONS

In this section, we describe how to compute a vernacular region using the shortest-path graph. First, we discuss how to efficiently compute the shortest-path graph of a set of samples. We also treat the special case where only the outline\(^2\) of the shortest-path graph is required, i.e. holes need not be detected. After that, we discuss how to detect outliers and holes and how to obtain an egg-yolk model from the shortest-path graph.

**Computing the shortest-path graph.** Since the shortest-path graph is a subset of the Delaunay triangulation, we first compute this triangulation on the set of samples. This takes \( O(n \log n) \) time and \( O(n) \) space [17]. The question remains how to eliminate the edges that are not part of a shortest path between two samples. This can be done by finding the shortest path distance between any two adjacent samples in the triangulation. If an edge \( e \) between two samples is not the shortest path, surely no shortest path between two other samples uses \( e \), as replacing it with the shorter path yields an even shorter path for the other samples. Cabello [2] solves the \( k \)-many distances problem for arbitrary planar graphs, finding the shortest path distances between

---

\(^2\)With “outline” we mean the outer boundary of the region. The outline is a (weakly) simple polygon, if we assume the region to be connected.
$k$ given pairs. Using the algorithm of Cabello, we can compute the shortest-path graph in $O(n^{4/3} \log n)$ time and $O(n)$ space from the Delaunay triangulation.

**Computing only the outline.** Suppose we wish to compute a vernacular region of which we know that it does not have holes. For this special case, we develop a faster algorithm, one that takes $O(n \log n)$ time and $O(n)$ space. It is based on an algorithm presented by Klein [16]. Klein’s algorithm constructs a data structure in $O(n \log n)$ time and space that allows for querying the shortest-path distance in a planar graph between a node on the outer face and another node in $O(\log n)$ time. We only briefly describe Klein’s algorithm and the required changes to find the outline of the shortest-path graph. For more details and formal definitions, refer to the work of Klein [16].

We say that a directed path $P$ is “right” of a directed path $Q$, if two conditions are met. Paths $P$ and $Q$ start at the same vertex on the outer face and at every shared vertex $v$, the edges around $v$ occur in the following clockwise order: outgoing edge of $P$, incoming edge of $P$, incoming edge of $Q$, outgoing edge of $Q$. The incoming edges and/or the outgoing edges are allowed to coincide. A right-short tree $T'$ of a weighted graph $G'$ is now defined as a tree rooted at a vertex $r$ on the outer face such that any path $P$ from $r$ to a vertex $t$ in $T'$ is strictly shorter than any other path from $r$ to $t$ in $G'$ that is “right” of $P$. A rightmost shortest-path tree is a shortest-path tree that is also a right-short tree.

Klein’s algorithm constructs such a rightmost shortest-path tree, $T$, for an arbitrary vertex on the outer face of the graph. Then the root is shifted to the next vertex in clockwise order and the tree is updated. This way, all shortest-path trees for vertices on the outer face are calculated.

The input to the modified algorithm is the Delaunay triangulation of the sample set $S$ where each edge has a weight equal to the squared Euclidean distance. It computes the outer face of the shortest-path graph. Only one modification is required to make Klein’s algorithm compute this outline. Instead of moving blindly to the next vertex on the outer face, the algorithm first checks whether the single edge is a shortest path to the next vertex. If it is, then nothing changes and the roots shifts to the next vertex. However, if it is not, then that edge is removed, causing another vertex to be the next on the outer face. This process is repeated until the edge to the next vertex is a shortest path between the two points. An edge that is removed by the modified algorithm is never part of the tree $T$ [18]. This implies that such an edge can never affect the shortest-path tree, not even before its removal. Hence, the modification does not affect any of the proofs given by Klein [16], and we obtain the following result:

**Theorem 1.** The outline of the shortest-path graph on a set $S$ of $n$ sample points can be computed in $O(n \log n)$ time and $O(n)$ space.

Outliers may prevent the algorithm from computing with the “actual” outline: every vertex that is on the outer face beforehand is on the outer face after the algorithm, while everything inside of the outline remains unchanged. Therefore, outliers should be filtered before executing this algorithm.

**Detecting outliers and holes.** The input $S$ may contain a large number of outliers. Hence, we briefly describe the method we used to filter these erroneous samples. However, other methods could be used as well as a preprocessing step.

Outliers are characterized by laying far away from (most of) the other points: the points in the intended region have relatively short distances between them, while outliers have far greater distances to other points.

Removing outliers is done by removing edges that are longer than the mean edge length multiplied by some constant factor. Let $\tau > 1$ be the constant factor and let $\mu_X$ denote the mean length over a set of edges $X$. We define the filter process with a dynamic mean. Based on a set of edges $E$, we look for a set $C$ that satisfies the following equation:

$$C = \{ e \mid e \in E \text{ and } |e| \leq \tau \cdot \mu_C \}$$

This equation can have multiple solutions as adding or removing edges affects the value of $\mu_C$. We wish to find the maximal solution, the set with highest cardinality. To obtain this solution, we start with an initial estimate of $C = E$, all available edges. While $|e| > \tau \cdot \mu_C$ holds for the longest edge $e \in C$, we remove edge $e$ from $C$ and update $\mu_C$ accordingly. Note that this procedure ends with at least one edge, as then $|e| = \mu_C$ must hold. This procedure can be executed in $O(n \log n)$ time by sorting the edges by length.

The outlier filtering as described above can lead to a disconnected graph. To avoid this, the filtering procedure can be adapted as follows. When removing an edge $e$, we check whether the result is still connected. If it is, we can simply continue. If the result is not connected, the edges are split into two components. We call a component too small if it has less samples than a certain number, say $\vartheta$. If the smaller component has less than $\vartheta$ samples, the entire component is considered an outlier and thus removed. If the smaller component has at least $\vartheta$ samples, we do not remove $e$. We still have to continue the process as one of the components may consist of edges that are all removed at some point. In that case, the removal of $e$ no longer splits the edges and thus it can be removed after all. Another step in removing outliers is to remove edges with an endpoint of degree 1, until no such edge exists. This can be done in $O(n)$ time.

After computing the shortest-path graph and filtering outliers, we have a new graph. If we do not want to have holes in the vernacular region, then we define the certain part of the vernacular region (the “yolk” of the egg-yolk model) to be the union of the bounded faces of this graph. If holes are allowed, however, we still need to decide which faces to include into the yolk and which faces to remove. We do this using a method similar to our outlier filtering, using the area of the faces (in lieu of the lengths of the edges) and a dynamic mean. Thus we compare the area of the largest face, $f$, to the mean area of the current set of faces, and remove $f$ if its area is more than $\vartheta h$ times the average (for some suitable parameter $\vartheta_h$). If the removal of $f$ splits the vernacular region into two subregions, we completely remove a subregion if has fewer than $\vartheta_h$ faces. As a last step, we add faces back until all (large) components are connected again.

Note that the outline may have multiple components inherently. Hence, we add faces only if this can in fact connect a separated component. Only the largest component is maintained at the end of the process. All removed faces are then merged, to prevent the graph from containing parts that are infinitely thin (single edges).

**Computing an egg-yolk model.** One can interpret the outline of the shortest-path graph as a polygon, where the detected holes are simply holes in the polygon. This polygon is a crisp boundary, whereas a fuzzy boundary or egg-yolk
model is desired. A straightforward way to obtain a fuzzy boundary is to inversely relate the certainty at a point \( p \) to the Euclidean distance of \( p \) to the polygon. An egg-yolk model can be found by setting a threshold distance \( \delta \). Any point with a distance greater than \( \delta \) is considered outside the region. The polygon itself is the “yolk”, the certain area. All other points form the “egg white”, the uncertain area. To compute this egg-yolk model, we compute the Minkowski sum of the polygon and a circle of radius \( \delta \). This is also known as polygon offsetting. Multiple methods exist to compute this offset explicitly, for example that of Wein [23].

5. INCLUDING CONTEXT

In this section, we show how geographic context can be integrated into our method. Recall that the idea behind the shortest-path graph is that shortest paths in the vernacular region roughly correspond to shortest paths in the graph. However, the “length” of a path can be influenced by other factors than just Euclidean distance. For example, two points on opposite sides of a river or some other obstacle can be very close together, but a path between the points would have to go around the obstacle, or perhaps cross it at some additional cost. By using geographic context that indicates the presence of obstacles (if such information is available), we may be able to get more accurate results.

Next, we show how geographic context can be integrated into our model. This is done by modifying only the distance measure between two points in the plane, based on the given geographic context. This is discussed first. After that, we discuss some properties of shortest-path graphs in combination with the new distance measure. These properties show that computing the shortest-path graph has become more difficult. Finally, the adapted algorithm is briefly described, as well as the method to compute the egg-yolk model from the result. We assume that the context is given as a collection \( O \) of line segments, which are considered obstacles. Which context-information to use is an important question, which we leave for future research.

Changing the distance measure. To incorporate geographic context, we redefine the weight of an edge between two points \( p \) and \( q \), that is, the “distance” between those points. Previously the weight was based on the Euclidean distance (more precisely, it was the square of the Euclidean distance) but now we have to modify the distance to take the obstacles into account. Thus, rather than using the Euclidean distance, we use the length of the shortest obstacle-avoiding path. However, completely avoiding obstacles may be unrealistic or even impossible. Therefore, it should be possible to pass through the obstacles, albeit at some additional cost. This means we deal with soft obstacles, obstacles we can cross, instead of hard obstacles, those we cannot cross. Formally, we define the distance between two samples, \( p \) and \( q \), as follows. Let \( P \) denote the set of all paths between \( p \) and \( q \), \( |P| \) the Euclidean length of path \( P \), and \( kP \) the number of intersections between path \( P \) and \( O \). Also, let \( \kappa \) denote the additional cost of one obstacle intersection (\( \kappa \geq 0 \)). Note that it is possible to give obstacles different crossing costs, but to simplify notation and computation, we assume that the crossing cost is the same for all obstacles. We express the distance between \( p \) and \( q \) as:

\[
d^2_c(p, q) = \min_{P \in P} \left\{ |P|^2 + \kappa \cdot kP \right\}
\]

Properties. First of all, we observe that in the presence of obstacles, the shortest-path graph need not be planar. In the worst-case scenario, it is even a complete graph with \( O(n^2) \) edge-edge intersections [18]. However, the worst-case situation is rather unrealistic and unlikely to occur when working with real vernacular regions. Usually, many samples points lie close together in a free area of the plane, while the actual paths that are hindered by the geographic context are limited. We observe that the presence of obstacles cannot decrease the distance between samples. Therefore, if a convex area is free of objects, we can use the Delaunay triangulation for the samples in this area. This is stated by the following lemma.

**Lemma 1.** Let \( A \) be a convex, obstacle-free area in \( \mathbb{R}^2 \) and let \( S \) be a set of samples. Let \( S_A \) denote \( S \cap A \) and let \( p \) and \( q \) be two points in \( S_A \). If the edge \( pq \) is not part of the Delaunay triangulation of \( S_A \), then there is no shortest path between two points in \( S \) that contains edge \( pq \).

The relatively simple proof is given by Meulemans [18]. It can be used to reduce the number of edges in the original graph as we could start by finding maximal sets whose convex hull does not contain any (part of) an obstacle. For each of these subsets, we calculate the Delaunay triangulation. We know that a linear number of edges is used in one such set. However, it remains to connect these “Delaunay components” in a proper way. Unfortunately, for a complete solution, we cannot connect only the boundaries of these components: given two components, it is possible that two samples not on the boundary may need to be connected for the shortest-path graph [18].

Without geographic context, a more efficient algorithm can be used when we are interested only in the outline of the shortest-path graph. If we wish to use a similar approach now that context is included, we need a bound on the complexity of this outline. We observe that even if two vertices are not on the outer face, their shortest, obstacle-avoiding path can be partially on the outer face. We believe that between two neighboring samples on the outer face, there can be only a low constant number of paths that add to the part of the outer face between the two samples. This would bound the number of paths on the outer face to \( O(n) \), where \( n \) is the number of samples. Furthermore, we believe that one obstacle can influence only a constant number of paths on the outer face. This leads to the following conjecture.

**Conjecture 1.** Let \( S \) be a set of samples and \( O \) a set of obstacle line segments. The number of edges on the outline of the shortest-path graph of \( S \) is bounded by \( O(|S| + |O|) \).

Computing distances. As the definition of distance between samples changed to a minimum over all paths, we need an algorithm to compute these distances. We give only a high-level description of the algorithm here; details of the method are given by Meulemans [18]. Given a set of soft obstacles \( O \), we wish to find shortest obstacle-avoiding path between samples.

We make use of a so-called layered graph. The idea is to have a graph in which every sample or vertex of the obstacles is connected to every other. Each edge then has a Euclidean length and a number indicating the number of obstacles that are crossed by that edge. This is copied into a number of layers, where every edge now goes from each layer \( i \) to layer
i + k (if it exists), where k is the number of intersections of that edge. The weight of an edge is its Euclidean length. An example of such a graph is depicted in Fig. 6. The shortest path in a layered graph from a sample in layer 0 to a sample in layer i then represents the shortest path between the two samples that has exactly i intersections. By computing the distance from a sample in layer 0 to each copy of another sample, we can find the path P between these that minimizes |P|^2 + κ · k_P. Using Dijkstra’s algorithm, the total execution time for this procedure is O(n^2 · m · k_{max} + n · m^2 · k_{max}), where n is the number of samples, m the number of obstacles, and k_{max} the maximal number of obstacle intersections of a straight line between any two samples. The memory usage is O(n^2 · m + m^2).

Outline and offsetting. Given the shortest-path graph of a region, we must find its outline and holes and also offset these to obtain an egg-yolk model. To find the outer boundary of a region, we must find its outline and holes and also offset the outline and holes to obtain an egg-yolk model. Given the shortest-path graph

\[ O \]

as before. While this approach is conceptually reasonable, we can use an algorithm to compute a face in an arrangement of line segments. De Berg et al. [6] present an O(\alpha(k) \log k) algorithm for k line segments, where \alpha(k) is the inverse of the Ackermann function. As the complexity of the shortest-path graph can be quadratic in the number of samples, the algorithm finds the outline of the shortest-path graph in O(n^2 · m + n(m \log nm)) time, where n is the number of samples and m the number of obstacles.

Detecting holes gives rise to another problem. Since the shortest-path graph is no longer necessarily planar, we cannot use the concept of face directly. However, holes can still be thought of as large empty areas when the shortest-path graph is “drawn”. Hence, we make the graph planar by inserting vertices on intersections and using the same method as before. While this approach is conceptually reasonable, the graph may become quite large. In a worst-case scenario, the shortest-path graph has O(n^3) intersections. Replacing all intersections in such a graph results in \( O(n^4) \) faces. Hence, it takes \( O(n^4 \log n) \) time to detect holes.

Offsetting the outline and holes to obtain an egg-yolk model is more complex in the presence of context. For hard obstacles, the problem of finding shortest obstacle-avoiding paths is well studied. For example, Hershberger and Suri [11] have studied the problem of computing the shortest-path map for a given point in the presence of polygonal obstacles. We can use this algorithm to find the egg-yolk, using a generalization from point source to polygonal source. In particular, we are interested in the “wavefront” (the reachable area) at distance \( \delta \).

For soft obstacles, there seems to be no algorithm readily available. It might be possible to adapt the algorithm of Hershberger and Suri [11], by handling wave-obstacle collisions differently. We observe that if a wavefront hits an obstacle, it can continue on the other side at increased distance. Locally, it is always best to cross the obstacle in a straight line. Hence, if the wavefront continues on the other side, the source remains the same. On collision, we could then duplicate the wavelet and increase the distance on the source. However, the details and verification of such an algorithm are still an open problem.

6. EXPERIMENTS

In this section, we discuss experimental results of our method and compare it to Kernel Density Estimation. We first briefly describe the KDE method we used, before discussing the results. For details on input and parametrization of the algorithms, refer to the work of Meulemans [18].

6.1 Kernel Density Estimation

We compare the results of our method to the results of Kernel Density Estimation. To this end, we use an unweighted Gaussian kernel function. The Gaussian kernel function is commonly used (for example, see Grothe and Schaab [10]). Using a radius \( r \), the certainty at a location, \( C(x) \), is expressed in the kernel function \( K(d) \) as follows:

\[
C(x) = \frac{1}{|S|} \sum_{s \in S} K\left(\frac{|x-s|}{r}\right), \quad \text{where } K(d) = \frac{1}{\sqrt{2\pi}} e^{-\frac{d^2}{2}}
\]

To obtain an egg-yolk model from Kernel Density Estimation, we cut the KDE surface using two parameters, \( \alpha \) and \( \beta \). The “yolk” is the set of all points for which \( \beta \leq C(x) \); the “egg white” is the set of all points for which \( \alpha \leq C(x) < \beta \).

To be able to compare results with context more fairly, we also integrate geographic context into the KDE method in a similar way as we did for our method. Instead of using the Euclidean distance \(|x-s|\), we use the following distance with soft obstacles:

\[
d^2_0(x, s) = \min_{P \in P} \left\{ |P| + \kappa \cdot k_P \right\}
\]

Note the difference in power of the path length (that is, the formula uses \(|P|\) and not \(|P|^2\)). By not squaring this length, we know that it gives the same results when no context is provided. As this distance is additive, it is slightly easier to compute: we do not need a layered graph to compute the distance, the augmented graph suffices (the augmented graph is the “unlayered” version of the layered graph [18]).

Giving a closed formula for the function \( C(x) \) is impractical. Therefore one usually evaluates \( C(x) \) on the vertices of a regular grid, using linear interpolation in the interior of the grid cells. This may cause some degeneracy near geographic context when it is included. For more accurate results, a constrained Delaunay triangulation of the obstacles with the regular grid can be used, sampling \( C(x) \) not only on the grid points, but on the obstacle vertices as well.

6.2 Results and discussion

We consider three different data sets: the Alps, a region that is unaffected by the provided geographic context; the Parisian suburbs, a region with a large hole that is affected by only a part of the provided context; and the Irish coast, a data set with highly varying density due to major cities and a hole. For each of the input data sets, we present figures that show the result of KDE and our method, both with and without using geographic context.
The Alps. The results of the Alps (Fig. 7) show some important aspects of our method. Whereas the result of our method without context looks decent, it actually deteriorates when providing country borders as context information, as the Alps are not influenced by these. However, it is desirable for a method to be able to cope with irrelevant context. One solution would be to use a lower crossing cost (close or equal to zero). But then the context—none of it—would be really taken into account, while in general this should be possible. There is an indication that the provided context here is not relevant: multiple faces of the subdivision (Austria, France, Italy, and Switzerland) contain a large portion of the samples. This information may be useful for automatic parametrization: in this case, a low crossing cost should be used. If most of the samples lie within one face, a high crossing cost can be used. The KDE method gives a more smooth result for the Alps. While this fits intuition for an imprecise region, the coverage of our method is better. When including context into the KDE method, it no longer gives a connected certain area (the “yolk”); it seems to retract from the boundaries that should not affect the region. Detecting which boundaries are likely to affect the region beforehand might improve results for KDE here.

Parisian suburbs. The results for the Parisian suburbs are depicted in Fig. 8. The results for KDE and our method without context are comparable. The certain area found using the KDE method covers a bit more of the city center, but the certain area of our method covers less of the area close to, but outside of the center. Our method actually detects a hole located on the city center, whereas the KDE method includes the entire area in the uncertain region. When we include context (the major roads and the Seine), our method neatly excludes the city center from the region, encircling the Boulevard Périphérique. We observe that the certain area of the region is no longer all around the center. Instead it is split at the western side. This is caused by the outlier filter due to a low sample density in that area. This result indicates that the method is somewhat capable of handling context, part of which is relevant (such as the central ring road) and part of which is irrelevant. However, this data set is manually generated and, although some outliers have been inserted, the results are an indication at best. The result of KDE with context is nearly the same as the result without context, but now the certain area does not overlap the city center. However, the uncertain area still covers it.

Irish coast. Results for the coast of Ireland are shown in Fig. 9. Without context, our method performs very well: the midlands of Ireland are filtered out during the hole detection process. When we include context, the outlier filter process creates a gap in the certain region in the southwest of Ireland. This is caused by the low sample density around the Shannon Estuary. No hole detection has been applied for this reason, as the large perceived hole in the middle of Ireland is in fact part of the infinite face. It still manages to capture the general shape of the shoreline. The KDE method seems unable to grasp the shoreline. It finds three major clusters: one near Dublin, one near Belfast and one just above the Shannon Estuary. The entire midland of Ireland is included in the uncertain area. When including context, the result improves but the certain area still does not cover the entire shoreline. It also covers more area that is further away from the shoreline. Therefore, we conclude that our method outperforms KDE in this experiment.

General observations. From the process of manually parameterizing the experiments, we learned that it requires thorough knowledge of the algorithms to be able to obtain the correct parameters. Especially for outlier filtering, the parameter settings are quite sensitive. If clusters are allowed to be too large (a high value for $\theta$) or the filter strength is too high (a low value for $\tau$), the certain region tends to quickly deteriorate to a small cluster, having less than one percent of the total number of samples. In contrast, if clusters are not allowed to be large or the filter strength is low, there seems to be a tendency to remove little to nothing. A possible solution to this problem is to have a minimal certain region size (say 50 percent of the total number of samples). This however introduces yet another parameter. Ideally, parameters would be chosen (semi-)automatically, for example by determining the variance of the samples.

The results of the Kernel Density Estimation are smooth shapes. For regions that are (partially) bounded by context, this seems undesirable. However, for shapes that are not strictly bounded, such as the Alps, a smooth shape seems more suitable than the jagged shape our method produces. Since we have a polygonal outline, we could consider applying polygon-smoothing methods to obtain a more smooth shape. To integrate this into the method, an algorithm should be used that can take the context into account. As proof-of-concept, Fig. 10 shows the result of smoothing the outline of the shortest-path graph found for the Alps.

![Figure 10: Result after applying a Bilaplacian smoother to the outline of the shortest-path graph.](image-url)
Figure 7: The experimental results for the Alps. (a) KDE without context. (b) KDE with context. (c) Our method without context. (d) Our method with context.

Figure 8: The experimental results for the Parisian suburbs using manually generated data. (a) KDE without context. (b) KDE with context. (c) Our method without context. (d) Our method with context.

Figure 9: The experimental results for the Irish coast. (a) KDE without context. (b) KDE with context. (c) Our method without context. (d) Our method with context.
rithm [2] can be used to compute the shortest-path graph in $O(n^{4/3} \log n)$ time. For vernacular regions that are known to have no holes, it suffices to compute only the outer face of the shortest-path graph. For this case, we improved the bound to $O(n \log n)$ by adapting Klein’s algorithm [16].

We modeled geographic context as a set of line segments. Each line segment was considered to be a soft obstacle. This caused the shortest-path graph to be a complete graph in a worst-case scenario and thus no longer a subset of the Delaunay triangulation. Moreover, the computation of the shortest path between two input samples became non-trivial. We defined a layered graph to be used with a shortest-path algorithm. Using this graph, we can compute the shortest-path graph with context in $O(n^2 \cdot m \cdot k_{\text{max}} + n \cdot m^2 \cdot k_{\text{max}})$ time, where $n$ is the number of input samples, $m$ is the number of obstacles and $k_{\text{max}}$ is the maximal number of obstacles intersecting the straight line between any two samples.

**Future work.** To be able to deal with outliers and detect holes, quite a few parameters are used. While this offers flexibility, it also makes obtaining a good result more difficult. Automatic parametrization may help. This is mainly of interest for the crossing cost and the parameters of the outlier filter and hole detection.

Without geographic context, our method produces a rather constrained egg-yolk model. It may be possible to adapt the method to fully exploit the potentials of an egg-yolk model. For example, the offset distance could be changed to depend on the size of the adjacent face: the bigger the face, the more uncertainty there is. Another possibility is to use the density to locally define an offset distance.

We have introduced a distance measure based on soft obstacles. As we have shown, KDE can also be adapted to include context. However, there may be more suitable ways to obtain a KDE method that takes context into account.

**Acknowledgements.** The authors would like to thank Ross Purves for providing feedback on the new method as well as the data sets used during the experimental study. The authors would also like to thank Philip Klein for providing additional details on his work.

8. REFERENCES


