IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 10, 2018, accepted November 19, 2018, date of publication December 7, 2018,
date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2885651

A Decision Support System for Managing the Water Space

DONALD MCMENEMY[1], GOPI VINOD AVVARI[2], DAVID SIDOTI[1], ADAM BIENKOWSKI[1], AND KRISHNA R. PATTIPATI[1], (Fellow, IEEE)
[1]Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-4157, USA
[2]Aptiv, Kokomo, IN 46902, USA

Corresponding author: Donald Mcmenemy (donald.mcmenemy@uconn.edu)

This work was supported in part by the United Technologies Corporation–Institute of Advanced Systems Engineering and in part by the U.S. Office of Naval Research and Naval Research Laboratory under Grant N00014-16-1-2036, Grant N00173-16-1-G905, Grant N00014-18-1-2838, and Grant HPCM034125HQU.

ABSTRACT A novel decision support system comprising of several interference identification algorithms for the detection of hazardous (i.e., risk to life, assets, and/or mission) voyages and water space allocations in maritime operations is proposed. Given a specified spatio-temporal region for each polygonal or ellipsoidal object, the objective is to identify interference, defined as overlap, among the assigned regions. The challenge is to fuse information on bathymetry, obstacles, and planned spatial assignments to identify conflicts among any combination of polygons or ellipses. We briefly review data structures for the representation of spatio-temporal regions computationally, and efficient algorithms for collision detection. Specifically, we propose a two-phase solution approach to interference identification using time-parameterized R-trees, linear programming, and quadratic programming. The proposed interference detection algorithms can detect overlaps among thousands of polygons and elliptic regions. The resulting decision support tool, conflict identification, is part of a larger system for water space planning.

INDEX TERMS Water space planning, collision avoidance, collision detection, interference, interference identification, bathymetry, linear programming, quadratic programming, time-parameterized R-trees, decision support system.

I. INTRODUCTION

A. MOTIVATION

The prevention of the loss of lives, assets and missions is vital to the success of maritime operations. Some of the major collisions involving U.S. Navy vessels since 1975 are listed in Table 1. For example, in 2009, the USS Hartford and the USS New Orleans collided in the strait of Hormuz, needing $120 million dollars worth of repairs and loss of 21 months of operations [1]. The USS Hartford, a submarine, was unaware of the USS New Orleans when the vessels collided as the USS Hartford ascended to the surface. These accidents highlight the importance of proactive notification of potential spatial conflicts to human operators via the fusion of information from multiple data sources, including bathymetry, obstacles and spatial assignments.

Interference is defined as an overlap among a pair of spatio-temporal regions. A well-established definition of an interference is the loss of separation, meaning the separation distance between vessels is below a user-specified threshold. Voyages and water space allocations are made safer by finding interferences with planned routes, and resolving them by altering plans or dispatching advisories. The mission of water space planning is to perform coordinated operations, and interference detection is a key component of planning. Advisories given ahead of time to submarine commanders are useful for scheduling and focusing their crews to handle complex interference scenarios.

B. RELATED RESEARCH

A common approach for determining if two regions overlap is a two-phase process [2]. The first phase, called the *broad phase*, quickly identifies region pairs that *may* overlap. Broad phase methods can result in false alarms, that is, an interference may be identified when none exists. The second part of the two phase approach is called the *narrow phase* and includes collision detection algorithms that determine whether an intersection indeed exists between any two regions.

1) CONVEX DECOMPOSITION

The polygons used to model safety regions proposed by a planner may not be convex and may contain holes.

TABLE 1. Incidents involving navy vessels since 1975.

Date	Assets Involved	Effects
21 August 2017	McCain	Loss of 10 lives, Damaged
17 July 2017	Fitzgerald	Loss of 7 lives, Estimated $250M - $500M in repairs
9 May 2017	Lake Champlain	No loss of life, Damaged
19 August 2016	Louisiana (SSBN), Eagleview	No loss of life, Damaged
20 November 2014	Amelia Earhart, Walter S. Diehl	No loss of life, Damaged
12 August 2012	Porter	No loss of life, $0.7M in repairs
20 March 2009	Hartford (SSN), New Orleans	No loss of life, $120M in repairs for Hartford
5 September 2005	Philadelphia (SSN)	No loss of life, Damaged
8 January 2005	San Francisco (SSN)	Loss of 1 life, $80M in repairs
22 July 2004	John F. Kennedy	No loss of life, Two fighter jets damaged
25 October 2003	Hartford (SSN)	No loss of life, Damaged
13 November 2002	Oklahoma	No loss of life, Damaged
9 February 2001	Greeneville (SSN)	Loss of 9 lives, $2M in repairs
9 July 2000	Denver	No loss of life, Damaged
14 June 1989	Houston	Loss of 1 life, Damaged
14 September 1976	John F. Kennedy Bordelon	No loss of life, Damaged
22 November 1975	John F. Kennedy Belkap	Loss of 8 lives, Damaged

The presence of holes is required to model obstacles and places where submarines dive or surface within another area of assignment. Our narrow phase algorithms require convex inputs, so the polygons that are not convex must be decomposed into a union of convex polygons. An ideal convex decomposition is fast and returns the partition with a minimal number of pieces.

There are many convex decomposition algorithms for polygons. Some of these triangulate the space internal to a polygon [3], [4], while others form trapezoidal shapes [5], [6] and some seek to find a set of convex polygons [7]–[9]. The triangulation methods result in the highest number of pieces, but can be modified to contain fewer pieces by removing edges of the triangulation [4]. The trapezoidal partitioning (trapezoidation) of a polygon works by extending segments horizontally from each vertex of the polygon. The intersection of a segment with the boundary of a hole or another edge of the polygon forms an end point for the segment [5]. Since the segments are parallel, the resulting partition forms trapezoids inside the polygon. Algorithms for constructing a partition with polygons, that are not trapezoidations of the space, add an edge originating from a reflex angle (or notch), which are internal angles of radian measure greater than π. The idea is to add edges such that there are no internal angles that are reflex, thereby guaranteeing a convex partition [7]. The Hertel and Mehlhorn algorithm [7] is one of the early algorithms that adds edges terminating at notches; this algorithm was enhanced using heuristics in [8] and [9]. An experimental comparison of various convex decomposition methods was performed in [9], where it was found that

a heuristic, termed *A5m*, resulted in the lowest cardinality of decomposition on average. A python implementation of *A5m*, termed Py2D, is used for convex decomposition with holes in this paper.

2) BROAD PHASE

The computational efficiency of broad phase methods stems from the use of bounding boxes, or multiple resolutions of bounding polytopes. For example, the method of axis aligned bounding boxes (AABBs) draws a rectanglular prism that completely contains a region. If the projections of the bounding boxes onto the global axis overlap, then the regions may overlap; otherwise, the regions are deemed disjoint. Oriented bounding boxes (OBB) provide a tighter fit than AABBs, since the bounding volume is oriented to minimize the total volume [10]. The AABB and OBB volumes can be embedded as hierarchical tree structures to contain assigned regions. Examples of hierarchical tree data structures include octrees [11], k-d trees [12], bucket trees [13], and R-trees [14]. Octrees recursively divide a cube into eight octants, while quadtrees [15], the 2-dimensional analogs, divide a plane into quadrants. An octree is formed with child elements that are the octants generated by recursively splitting parent elements. Each child element contains an indicator of whether a region is contained in the child element or not. The k-d tree is a generalization of the octree, where instead of binary splits, as used in generating the octree, the volume is split into k subtrees. The analysis of k-d trees applies to octrees and quadtrees. The resolution (i.e, size of the space partitions) of the k-d tree represents a trade-off between the false positive rate and the query time. A finer resolution results in fewer false positives at the expense of an increased query time. The choice of resolution is critical to the success of the k-d trees and the related space partitioning methods (octrees, quadtrees, etc).

For moving objects, the update of a child node in a k-d tree, octree, or quadtree can be cumbersome, because the child nodes that contain parts of the object need not share the same parent. An attempt to improve the update of regions in octrees is termed the bucket tree, where an array of bounds is updated under the temporal coherence assumption -- regions are of similar shape and position across time steps. Rather than exhaustively updating the index of each entry, the bounds are used to update the indices of entries. However, in the context of water space planning, the temporal coherence assumption is not valid; thus, neither the Bucket tree nor the k-d tree approaches are suitable for the interference identification problem posed in this paper.

Rather than partitioning the entire space as in k-d trees, the R-tree is formed by surrounding regions that are close together with a minimum bounding box and using it to represent the regions at the parent node level [16]. Consequently, successive levels above the leaves of an R-tree contain a single minimum bounding box that contains all the child elements. R*-tree is an improvement upon the R-tree in that it utilizes heuristics to optimize the tree by modifying the set of minimum bounding boxes to minimize the area, perimeter, or mutual overlap of the minimum bounding boxes [17]. This optimization is shown to speed up queries. In order to handle moving regions, the minimum bounding boxes may be enlarged to keep the child elements surrounded for a chosen time window. This modification results in a structure known as time-parameterized R-tree (TPR-tree) [18]. The TPR-tree is periodically re-optimized to enhance query performance on the data structure via reduction of the area and the number of overlaps of minimum bounding boxes. Due to the rapid update and query times, the R-tree variants are ideally suited for applications involving spatio-temporal data.

3) NARROW PHASE

Examples of existing narrow phase algorithms for convex polygon and polytope regions include GJK (Gilbert, Johnson, Keerthi), Voronoi Marching (Closest Feature Pair), and linear programming algorithms. The GJK algorithm iteratively minimizes the distance between two convex polygons, or more generally, polytopes [19], [20]. Vononoi Marching finds the closest feature pair, where features are Voronoi partitions of the exterior space of a convex polygon (polytope) [21]. A linear program [22] is solvable by an interior point method to detect intersections of polytopes. The GJK algorithm is a simplex-based method, and thus inferior to an interior point method for polygons (polytopes) with a large number of vertices, since the latter can handle models with a large number of vertices faster. The Voronoi Marching method has robustness issues and can exhibit cycling [23].

Exact methods for detecting intersection between a pair of ellipsoids was examined and a novel approach based on least squares minimization over a sphere was proposed in [24]. This method was shown to be computationally superior to a method that finds the signs of the roots of a 4^{th} order polynomial [25]. The approach proposed in [24] enhances the method in [26] by testing squared Mahalanobis distance metrics for a pair of ellipsoids and, to the authors' knowledge, is the fastest known algorithm for this problem.

Interference determination between a polygon (polytope) and an ellipse (ellipsoid) is performed by circumscribing the ellipse (ellipsoid) by a polygon (polytope) with a large number of vertices (> 32) and solving the resulting problem using an interior point method for linear programming [22]. Directly solving polygon-ellipse inferference problems as a constrained quadratic programming problem with linear constraints is left as a future research issue.

C. SCOPE AND ORGANIZATION OF THIS PAPER

Towards ensuring safe maritime operations, we have developed CONFIDENT, a conflict (interference) identification tool, that detects interferences between operating regions represented as polygons and/or ellipses. The polygons may be non-convex and of non-zero genus, that is, they may contain holes. These features make CONFIDENT a unique solution

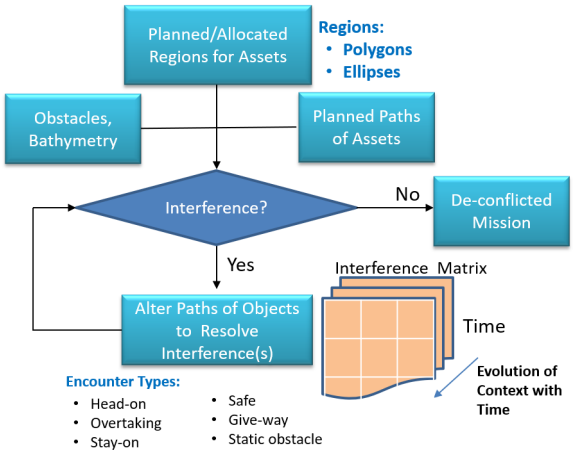


FIGURE 1. The CONFIDENT system at a high level.

to interference identification using any of the definitions of interference between static and dynamic regions.

In this paper, we present a two phase approach to interference determination between any combination of polygonal and elliptical regions for water space planning. The broad phase consists of a TPR-tree with bounding boxes for each region at the leaves. The TPR-tree is chosen due to its efficiency over other spatio-temporal indexing structures. The narrow phase for polygon/polygon encounters is an interior point method from [22]. Interference determination between a polygon and an ellipse is performed by approximating the ellipse with a circumscribing polygon, and using the broad and narrow phases for polygon/polygon encounters. Our method for interference detection between a pair of ellipsoids involves using squared Mahalanobis distance metrics and a least squares minimization over a sphere [24]. The overall system performs interference detection among any combination of thousands of polygonal and elliptical regions.

The paper is organized as follows. In Section II, the water space planning problem is described. We present our 2-dimensional solution approach in Section III and detail the broad phase TPR-tree and its extension to 3-D regions in Section IV. In Section V, we present the narrow phase algorithms. Section VI contains a discussion of the runtime and scalability of the interference detection algorithms. The paper is concluded in Section VII with a summary and suggestions for future work.

II. PROBLEM DESCRIPTION

Among the tasks of a planner are scheduling operations in reserved water space, voyages between ports, and alerting parties of scheduling changes through messages. The schedules aim to ensure that vessels avoid shallow water, rough seas, and hazards due to conflicting region assignments with other vessels. Such hazards include potential collisions, and damage due to towed cables, nets, or fire from live weapon systems. Automated detection of encounters that are too close is needed, since manually detecting interferences is tedious, complex and cumbersome. The number of regions, planned paths, and tracked vessel paths may be on the order of thousands, making manual interference identification intractable. The planning period and the corresponding amount of time available for interference identification varies depending on the situations being monitored.

TABLE 2. Notation.

N	The number of objects
$X^{\{j,t_s,t_e\}}$	A polygon
t_s	Interval starting time
t_e	Interval ending time
j	Index of an asset
i	Index of a vertex
$\underline{x}_i^j$	Position of a vertex of a polygon
$\{\underline{\mu}, \underline{r}, \theta\}^{\{j,t_s,t_e\}}$	Characterization of an ellipse
$\underline{\mu}$	Mean position vector
$\underline{r}$	Ranges (nm)
θ	Bearing angle (rad)
Σ	Covariance matrix
R	Rotation matrix
Λ	Eigenvector matrix
$P = \{W, B, \underline{t}\}^{\{j,\underline{t}\}}$	A planned path with polygonal safety region
W	Matrix of way-point vector columns
B	Matrix of buffer values
$\underline{t}$	Vector of desired arrival times corresponding to each way-point

An essential part of efficient operation on the ocean is to choose safety regions, such as the closest point of approach (CPA), that allow for ample time to react to a threat and yet do not generate spurious resolution maneuvers. For example, a safety region that is too small runs the risk of endangering the vessel, while a safety region that is too large encourages making unnecessary evasive maneuvers at far off distances. With the required separation distances, a planner can input polygons, ellipses, and paths to model reserved water spaces and moving havens for vessels. A summary of 2-D area and path types, which are static and dynamic, respectively, is presented in Table 2.

A polygon assigned to an asset or an obstacle, indexed by j, with n constant vertices in a time interval starting at t_s and ending at t_e can be represented by a $2 \times n$ matrix, $X^{\{j,t_s,t_e\}}$, with columns $\underline{x}_i^j$. The units of the vertices, $\underline{x}_i^j$, are typically in degrees longitude (long) and degrees latitude (lat), or nautical miles from some origin. An ellipse can be described by a set $\{\underline{\mu}, \underline{r}, \theta\}^{\{j,t_s,t_e\}}$, where $\underline{\mu} \in R^2$ is a position vector with units

of degrees (lat, long) or nautical miles with respect to some origin. The vector $\underline{r} \in R^2$ is a vector with ranges in units of nautical miles. The bearing angle, θ is measured in radians. The characterization of an ellipse, $\{\mu, \underline{r}, \theta\}^{\{j,t_s,t_e\}}$ is used to generate a covariance matrix $\Sigma_j = R_j^T \Lambda_j R_j$, where

$$R_j = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \quad (1)$$

$$\Lambda_j = \begin{bmatrix} {r_1}^2 & 0 \\ 0 & {r_2}^2 \end{bmatrix}. \quad (2)$$

The level curve of points, $\underline{p}$, such that $(\underline{p} - \underline{\mu})^T \Sigma_j^{-1} (\underline{p} - \underline{\mu}) = 1$ define the boundary of the elliptical safety region. A constant other than 1 in the right hand side can be accommodated by scaling the covariance matrix or eigenvalues.

In the case of moving vessels, a path is indicated by a series of vertices and a description of the safety region on each leg of the path between vertices. A planned path with n way-points and a rectangular safety region can be represented by a set $\{W, B, \underline{t}\}^{\{j,t_s,t_e\}}$, where $W \in R^{2\times n}$ is a matrix with columns $\underline{w}_i \in R^2$ as vertices, $B \in R^{2\times n}$ is a matrix with columns $\underline{b}_i$ whose elements are the buffer ahead/astern b_{aa}, and buffer port/starboard, b_{ps} distances in a rectangular region. The buffers for the i^{th} leg given in $\underline{b}_i$, or the i^{th} column of B, correspond to the path between way-points of $\underline{w}_i$ and $\underline{w}_{i+1}$, $i \in [1, n]$. The vector $\underline{t} \in R^n$ contains the desired arrival time at each way-point in W. The element t_i of $\underline{t}$ must satisfy $t_i \in [t_s, t_e] \ \ \forall i \in [1, n]$.

Figure 2 depicts a scenario with a mixture of elliptical, polygonal, and path assignments. Given a set of inputs as regions and paths, the objective is to determine if the intersection of each pair of spatio-temporal regions is non-empty. If the intersection is non-empty, there exists at least a common border between the regions. If a common border is not allowed between a pair of spatio-temporal regions, then there exists an interference.

III. SOLUTION APPROACH

A two-phase interference identification method, embedded in CONFIDENT, is presented as Algorithm 1. It takes as input a set of areas A, paths P and a TPR-Tree. Optionally,

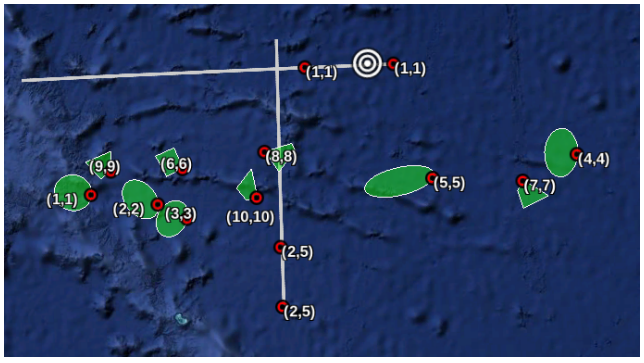


FIGURE 2. Ten static assignments and two paths are shown. The first index is that of the vessel in the assignment and path. The second index is the index of the geometric region.

Algorithm 1 CONFIDENT Called by WaSP

▷ The TPR-Tree *TPR*
▷ Areas A, and paths P to add to the TPR-Tree
▷ Areas *rm_A*, and paths *rm_P* to delete from the TPR-Tree
function *CONFIDENT*(P, A, *rm_P*, *rm_A*, *TPR*)
  ▷ Pre-processing
    TPR.delete(*rm_P*, *rm_A*)
    AABB_A = *make_AABB*(*convex_decomp*(A))
    AABB_P = *make_AABB*(P)
    TPR.insert([*AABB_A*, *AABB_P*])
  ▷ Broad Phase
    for x in [*AABB_A*, *AABB_P*]
      bp_conflict[x] = *TPR.intersection*(x)
    end for
  ▷ Narrow Phase
    jobs = *split*(*bp_conflicts*, *N_nodes*)
      ▷ Parallel processing of interference checks
    conflicts_sbs = *map*(*narrow_phase*(), *jobs*)
return *TPR*, *conflicts_sbs*
end function

a set of areas and paths to delete from the TPR-Tree may be specified as *rm_A* and *rm_P* respectively. The steps can be categorized as pre-processing, broad phase and narrow phase. Pre-processing involves the computation of the convex decomposition of non-convex polygons, the approximation of ellipses by a polytope, the computation of bounding boxes and the creatation or update of the TPR-tree. Note that the convex decomposition and approximation of an ellipse by a polygon may not be required if the water space allocations are either convex or elliptic, or if the broad phase indicates that there is no overlap. The convex decomposition and the approximation of an ellipse by a polygon can be performed offline, preferably at the time each region is created. Doing so reduces the total processing time of the narrow phase. The convex decompostition, *convex_decomp*(), is performed using the *A5m* algorithm in [9]. Approximating an elliptic region with a polygon or polytope is presented in Section IV. The generation of bounding boxes, via *make_AABB*, is presented in Algorithm 2 for area and path assignments, which are static and dynamic, respectively. The updated TPR-tree, and lists of axis-aligned bounding boxes, *AABB_A, AABB_P*, are passed to the broad phase.

Section IV describes the broad phase, where all regions have a bounding box representation in a TPR-tree as made in the preprocessing step. The TPR-Tree is queried for intersection with the AABB for each area [18]. The resulting interferences, *bp_conflict*, are passed onto the narrow phase for validation. Algorithm 5 shows the narrow phase of interference identification between a pair of regions for one time interval. The narrow phase algorithm used depends on the type of regions. Two polygons may be checked for interference by formulating a linear programming problem and solving the resulting optimization problem using a primal-dual path

Algorithm 2 Make ABBB for Areas and Paths

function *make_BB*(*S*)
  ▷ S is a set of area assignments and paths
  for *X* in *S* **do** ▷ X is an area
    $BB(X,1) = \begin{bmatrix} x_1^{\dashv}, x_1^{\dashv}, x_1^{\vdash}, x_1^{\vdash} \\ x_2^{\dashv}, x_2^{\vdash}, x_2^{\vdash}, x_2^{\dashv} \end{bmatrix}$
    $V(X,1) = \begin{bmatrix} 0, 0, 0, 0 \end{bmatrix}$
  end for
  for *P* in *S* **do** ▷ P is a path
    for $i = 1 : length(W) - 1$ **do**
      $\underline{\psi}(i) = disbear\_rl(\underline{w}_i, \underline{w}_{i+1})$
      $\overline{BB}(P,i) = path\_BB(\underline{w}_i, \underline{\psi}(i), \underline{b}_i)$
    end for
    for $i = 1 : len(W) - 2$ **do**
      $V(P,i) = edge\_V(BB(P,i),$
      $BB(P,i+1), t_i, t_{i+1})$
    end for
    $BB\_end = path\_BB(\underline{w}_{i+1}, \underline{\psi}(i), \underline{b}_i)$
    $V(P, len(W) - 1) = edge\_V(BB(P,i),$
    $BB\_end, t_i, t_{i+1})$
  end forreturn *BB*, *V*
end function

following algorithm [27]. When two ellipses are interfering at the broad phase level, a series of squared Mahalanobis distance-based tests and a quadratic program is used for interference determination. Typically, the Mahalanobis distance based tests obviate the need for solving the quadratic programming problem in more than 90% of the cases. The polygon approximation of the elliptic region is used when an ellipse and a polygon are checked for interference. A dictionary containing all interferences, *conflicts_sbs*, and the TPR-tree are returned as outputs.

IV. BROAD PHASE ALGORITHMS

Querying the TPR-tree involves specifying an initial AABB, start time, end time, and velocities for each of the bounds for the AABB [29]. The query is performed for each AABB for all areas and paths. The return from each query is the list of AABBs that intersect the query AABB during the specified time interval. This is a rapid query-response process, since it avoids performing checks in a pairwise manner.

The generation of the AABB and computation of the velocities of bounds of the AABB for paths are done for all area assignments and paths. Area assignments are static and have a constant AABB over the active interval $[t_s, t_e]$. Paths have an AABB that grows over the duration of each leg. The TPR-tree is updated with the constant AABB for areas, and an initial AABB and velocity for each leg of a path. The pseudocode for this pre-processing is shown as Algorithm 2. We drop the index of the object, *j* to simplify the notation.

Bounding box representations for area assignments (static polygons or ellipses) are computed by finding the maximum and minimum values in each dimension. For a 2-dimensional

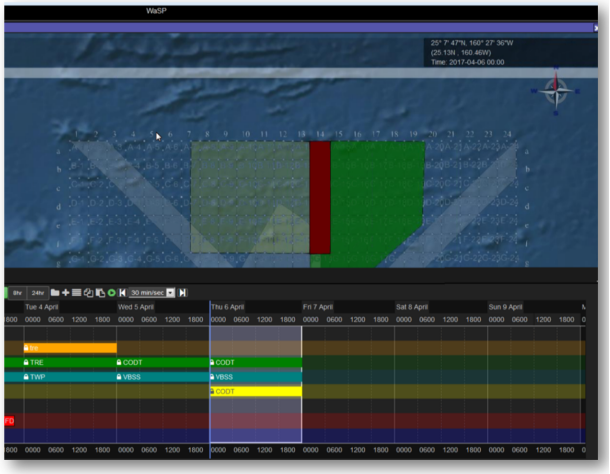


FIGURE 3. A representation of the temporal distribution of assignment regions and dynamic paths. The active time, regions and paths are shaded in grey.

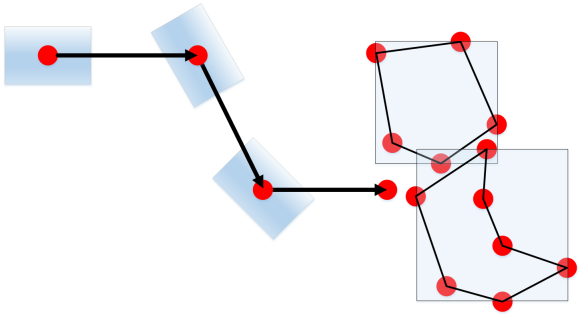


FIGURE 4. The axis aligned bounding boxes for two polygons are shown bounding each polygon. The path to the right has 4 way-points. There are 3 moving havens; one for each leg.

object ($k = 1, 2$),

$$x_k^{\dashv} = \max\left[X^{\{j,t_s,t_e\}}(k,:)\right] \tag{3}$$

$$x_k^{\vdash} = \min\left[X^{\{j,t_s,t_e\}}(k,:)\right] \tag{4}$$

the minimum and maximum values are computed, respectively. A MATLAB-like notation is used here to indicate that each row of the matrix $X^{\{j,t_s,t_e\}}$ corresponds to a dimension of the polygon. The bounds $\underline{x}^{\vdash}$ and $\underline{x}^{\dashv}$ typically have units of degrees of latitude and longitude, respectively. The bounding box for an ellipse is computed via the approximation for an ellipse, and then the algorithm for computing the bounding box for a polygon.

The bounding box representation for static objects are stored in the TPR-tree with the associated start and end times. Similarly, the tracks are parameterized by time with a computed AABB and velocity of each edge for the bounding box over each leg [18] using Algorithms 3 and 4, respectively. An example path is shown in Figure 4, with 3 moving havens defined on the 3 legs of the path. The edge velocities for the AABB surrounding the moving haven are simply the differences between the starting AABB and ending AABB edges divided by the nominal transit time. The velocities of the edges are computed, in Algorithm 4. It is seen in

Algorithm 2 that the velocity of the edges of the final leg are computed using the last AABB and an AABB positioned at the final way-point. Finally, the area assignments and paths are inserted into the TPR-Tree as in Algorithm 1, and are then queried for intersection. The resulting overlaps are logged, and passed onto the narrow phase portion for exact overlap identification.

Algorithm 3 Make ABBB for a Single Leg of a Path

function *path_BB*($\underline{w}$, *bearing*, $\underline{b}$)
▷ b_i contains the values for buffer front, right, left, back
Compute the vertices of the bounding box at $\underline{w}$
$[front, right, left, back] = \underline{b}$
$p_0, p_1 = lonlat\_rl(w[0], w[1], front, bearing)$
$lon, lat = lonlat\_rl(p_0, p_1, left, bearing - 90)$
$H.append([lon, lat])$
$lon, lat = lonlat\_rl(p_0, p_1, right, bearing + 90)$
$H.append([lon, lat])$
$p_0, p_1 = lonlat\_rl(w[0], w[1], back, bearing + 180)$
$lon, lat = lonlat\_rl(p_0, p_1, right, bearing + 90)$
$H.append([lon, lat])$
$lon, lat = lonlat\_rl(p_0, p_1, left, bearing + 270)$
$H.append([lon, lat])$
$BB = \begin{bmatrix} h_1^{\dashv}, h_1^{\dashv}, h_1^{\vdash}, h_1^{\vdash} \\ h_2^{\dashv}, h_2^{\vdash}, h_2^{\vdash}, h_2^{\dashv} \end{bmatrix}$
return *BB*
end function

Algorithm 4 Compute Edge Velocities for ABBB for a Single Leg of a Path

function *edge_Velocities*($BB(P, i)$, $BB(P, i+1)$, t_i, t_{i+1})
$[l_e, b_e, r_e, t_e] = BB(P, i+1)$
$[l_s, b_s, r_s, t_s] = BB(P, i)$
$left\_right\_v_1 = (l_e - l_s)/(t_{i+1} - t_i)$
$top\_bottom\_v_1 = (b_e - b_s)/(t_{i+1} - t_i)$
$left\_right\_v_2 = (r_e - r_s/(t_{i+1} - t_i)$
$top\_bottom\_v_2 = (t_e - t_s)/(t_{i+1} - t_i)$
$leftv = min(left\_right\_v_1, left\_right\_v_2)$
$rightv = max(left\_right\_v_1, left\_right\_v_2)$
$bottomv = min(top\_bottom\_v_1, top\_bottom\_v_2)$
$topv = max(top\_bottom\_v_1, top\_bottom\_v_2)$
$velocities = (leftv, bottomv, rightv, topv)$
return *velocities*
end function

A. EXTENSION TO 3-D

Current operations require 2-D elements at various depth levels. However, the TPR-tree, and the narrow phase algorithms also generalize to any interference detection in any number of dimensions. The narrow phase algorithms require no modification, since they are dimension-independent. The TPR-tree implementation is also dimension-indepenedent [29]. The only modifications required are in pre-processing, where ellipsoids are approximated by circumscribing polytopes, convexification of 3-D non-convex polytopes, and computation of the bounding volumes and velocities in three or more dimensions. Convexification of 3-D non-convex polytopes is left for future work.

V. NARROW PHASE ALGORITHMS

The narrow phase algorithms presented here provide a new approach to detect conflicts between any combination of ellipses and polygons. The resulting narrow phase for any combination of ellipses, and convex polygons is presented.

Algorithm 5 Narrow Phase Pseudocode

function *Narrow*(*Pair*)
if both regions in *Pair* are ellipsoids **then**
▷ Use Algorithm 10 in [24]
else
if Any region is elliptic **then**
▷ Look-up the approximation via Algorithm 12
$polyVerts = circumscribe(R, \Lambda, \underline{p})$.
end if
▷ Solve LP [22] via Algorithm 6
end if return *Encounter_Type*, *Pair*
end function

A. INTERIOR POINT METHOD FOR A PAIR OF POLYGONS

A primal-dual path following algorithm [27] is used to solve the linear program (LP) for an indication of overlap between a pair of polygons [22]. Objects represented as two convex polygons $X^j \in R^{2\times n_j}$ and $X^q \in R^{2\times n_q}$ $(j \neq q)$, are defined by vertices $\underline{x}_i^j$ and $\underline{x}_k^q$ where $i \in [1, 2, \ldots n_j]$ and $k \in [1, 2, \ldots n_q]$. We drop the time superscripts since both of the polygons exist at a common time. Let $\underline{u} \in R^2$ represent the coordinates of the distance between two points that are guaranteed to belong to X^j and X^q, respectively, such that

$$\underline{u} = \left(\frac{1}{n_q}\sum_{k=1}^{n_q} \underline{x}_k^q\right) - \left(\frac{1}{n_j}\sum_{i=1}^{n_j} \underline{x}_i^j\right) \tag{5}$$

A convex hull for these three vector sets is defined as

$$\frac{1}{n_j}\sum_{i=1}^{n_j} \underline{x}_i^j \alpha_i - \frac{1}{n_q}\sum_{k=1}^{n_q} \underline{x}_k^q \beta_k + \underline{u}\phi = 0 \tag{6}$$

where $\underline{\alpha} = \{\alpha_i\}_{i=1}^{n_j}$, $\underline{\beta} = \{\beta_i\}_{i=1}^{n_q}$, and ϕ are the non-negative coefficients ensuring convexity. To reformulate the problem in terms of these non-negative coefficients, we define

$$\rho = ||\underline{u}||_\infty \tag{7}$$

and let the primal model, whose solution can indicate if X^j and X^q have mutual interference, be

$$\min \ \rho\phi \tag{8}$$

$$s.t. \ \sum_{i=1}^{n_j} \underline{x}_i^j \alpha_i - \sum_{k=1}^{n_q} \underline{x}_k^q \beta_k + \underline{u}\phi = \underline{0} \tag{9}$$

$$\sum_{i=1}^{n_j} \alpha_i = 1 \tag{10}$$

$$\sum_{k=1}^{n_q} \beta_k = 1 \tag{11}$$

$$\phi \geq 0; \quad \alpha_i \geq 0;\ \beta_k \geq 0;\ \forall\, i, k \tag{12}$$

where $\underline{\alpha}$, $\underline{\beta}$ and ϕ are the primal decision variables. The primal objective is a measure of separation between the two polygons. The primal problem has $n_j + n_q + 2$ decision variables, 4 equality constraints and non-negativity constraints on the decision variables. The optimal solution to the primal problem is zero when the two polygons overlap. Otherwise, the optimal solution has some positive value. The dual problem is

$$\max\ \pi_3 + \pi_4 \tag{13}$$

$$s.t. \begin{bmatrix} X^{jT}, & K_1^T \\ X^{qT}, & K_2^T \\ \underline{u}^T, & \underline{0}^T \end{bmatrix} \underline{\pi} + \underline{s} = \underline{c} \tag{14}$$

$$\underline{s} \geq \underline{0} \tag{15}$$

where $\underline{0}$ is a 2×1 vector of all zeros,

$$K_1 = \begin{bmatrix} 1, 1, \ldots, 1 \\ 0, 0 \ldots, 0 \end{bmatrix} \in R^{2\times(n_j+n_q)}, \tag{16}$$

$$K_2 = \begin{bmatrix} 0, 0 \ldots, 0 \\ 1, 1, \ldots, 1 \end{bmatrix} \in R^{2\times(n_j+n_q)}, \tag{17}$$

$$\underline{c} = \begin{bmatrix} \underline{0}^T, & \rho \end{bmatrix}^T \in R^{(n_q+n_j+1)}, \tag{18}$$

$$\underline{\pi} = \begin{bmatrix} \pi_1, \pi_2, \pi_3, \pi_4 \end{bmatrix} \in R^4, \tag{19}$$

are dual decision variables, and $\underline{s} \in R^{(n_q+n_j+1)}$ are dual slack variables. The primal-dual path following algorithm used to solve (8-19) is shown in Algorithm 6. The function takes a pair of polytopes or polygons and determines whether they interfere or not. The initialization of the primal decision variables, $\underline{x}$, dual slack variables, $\underline{s}$, and dual decision variables, $\underline{\pi}$ are done in Algorithm 7. The function *diag*() represents the function that creates a diagonal matrix given a vector containing the elements of the diagonal. The choice of how to update the barrier scaling parameter, χ, is given in [27]. It uses the duality gap which monotonically decreases, and so eventually the barrier scaling parameter decreases. Algorithm 8 computes the Newton directions and step sizes for the primal and dual variables. Then, the duality gap, constraint matrix A, cost coefficients ϕ and ρ are updated as per [22]. Termination checks are performed to end the algorithm. A maximum number of iterations is set to ensure termination.

The algorithm requires the polygons to be convex. Therefore, any non-convex polygons were previously decomposed into a set of convex polygons. The *A5m* algorithm in [9] is used for convex decomposition. After decomposing a non-convex polygon, the LP problem is solved for all combinations of convex polygons as in Algorithm 5.

Algorithm 6 (LP) Primal-dual path following algorithm solving [22]

function *LP_narrow_phase*(X^j, X^q)
▷ X^j is an 2 by n_j matrix
▷ X^q is an 2 by n_q matrix
$\chi, \gamma, \epsilon, \underline{\alpha}, \underline{\beta}, \underline{x}, \eta, \underline{\pi}, X, \underline{s}, S, \underline{u}, \rho, A, \underline{b}, \underline{c}$ =
LP_initialize(n_j, n_q, X^j, X^q)
$gap = \underline{s}^T \cdot \underline{x}$
$counter = 0$
while $gap \geq \epsilon$ **do**
$counter+ = 1$
$\chi = 0.8\frac{gap}{4}$
$\underline{x}, \underline{s}, \underline{\pi}, X, S$ =
LP_Newton_dir_steps($A, S, X, \underline{\pi}, \underline{x}, \underline{s}, \underline{b}, \underline{c}, \chi, \gamma$)
$gap = \underline{s}^T \cdot \underline{x}$
$\underline{\alpha}, \underline{\beta}, \phi, \underline{u}, \rho, \underline{c}, A$ =
LP_primal($\underline{x}, X^q, X^j, \underline{\alpha}, \underline{\beta}, \underline{u}, 2, n_j, n_q, \rho$)
if $\rho \cdot \phi \leq 10^{-5}$ **then**
tf_interference = *true*
tf_termination = *true*
break
end if
if $\pi_3 + \pi_4 > 0$ **then**
tf_interference = *false*
tf_termination = *true*
break
end if
if *counter* > 100 **then**
tf_interference = *false*
tf_termination = *false*
break
end if
end while
return *tf_interference*, *tf_termination*
end function

B. SQUARED MAHALANOBIS DISTANCE TESTS AND QP FOR A PAIR OF ELLIPSES

The interference detection between ellipses is handled via squared Mahalanobis distance test and QP, Algorithm 10 in [24]. It first checks the squared Mahalanobis distance between the ellipses, then, if necessary, solves a quadratic program termed (LSMOS) [30]. Both of the ellipses are defined with the boundary of the safety region to coincide with a squared Mahalanobis distance of 1. This is achieved by specifying the eigenvalues of the covariance matrix in Equation 2. Given means $\underline{\mu}_i$, $\underline{\mu}_j$ and covariances Σ_i^{-1}, Σ_j^{-1} the squared Mahalanobis distance

$$d_{ij} \leftarrow (\underline{\mu}_i - \underline{\mu}_j)^T (\Sigma_i + \Sigma_j)^{-1} (\underline{\mu}_i - \underline{\mu}_j) \tag{20}$$

is compared to a threshold of 2. It is proven in [24] that $d_{ij} > 2$ corresponds to separated ellipses as defined in Section II. Rapid indication of overlap is found by finding the squared Mahalanobis distance of a query point $\underline{\mu}_{ij}$, which

Algorithm 7 Initialize Primal and Dual Variables

function *LP_initialize*(n_j, n_q, X^j, X^q)
$\chi = 1$ ▷ Duality gap parameter
$\gamma = 0.995$ ▷ Step factor scaling parameter
$\epsilon = 10^{-8}$ ▷ Duality gap tolerance parameter
$\underline{\alpha} = \frac{1}{n_j} \cdot \underline{e}$ ▷ $\underline{e}$ is a n_j vector of ones.
$\underline{\beta} = \frac{1}{n_q} \cdot \underline{e}$ ▷ $\underline{e}$ is a n_q vector of ones.
$\phi = 1$
$\underline{x} = [\underline{\alpha}^T, \underline{\beta}^T, \phi]^T$
$\eta = \max\left(\max(abs(X^j)), \max(abs(X^q))\right)$
$\underline{\pi} = [\underline{0}^T, \eta, \eta]^T$
$X = diag(\underline{x})$
$\underline{s} = \chi \cdot X^{-1}\underline{e}$ ▷ $\underline{e}$ is $n_j + n_q + 1 \times 1$
$S = diag(\underline{s})$
$\underline{u} = -\left(\sum_{i=1}^{n_j} X^j(i,:) \cdot \alpha_i - \sum_{k=1}^{n_q} X^q(k,:) \cdot \beta_k\right)$
$\rho = ||\underline{u}||_\infty$
$K_1 = \begin{bmatrix} 1 \cdots 1 \\ 0 \cdots 0 \end{bmatrix}; K_2 = \begin{bmatrix} 0 \cdots 0 \\ 1 \cdots 1 \end{bmatrix}$
$A = \begin{bmatrix} X^j & -X^q & \underline{u} \\ K_1 & K_2 & \underline{0} \end{bmatrix}$ ▷ A is a $(4 \times (n_j + n_q + 1)$ matrix.
$\underline{b} = \begin{bmatrix} \underline{0} \\ \underline{e} \end{bmatrix}; \underline{c} = \begin{bmatrix} \underline{0} \\ \rho \end{bmatrix}$
return $\chi, \gamma, \epsilon, \underline{\alpha}, \underline{\beta}, \underline{x}, \eta, \underline{\pi}, X, \underline{s}, S, \underline{u}, \rho, A, \underline{b}, \underline{c}$
end function

Algorithm 8 Compute Newton Directions and Step Sizes

function *LP_Newton_dir_steps*($A, S, X, \underline{\pi}, \underline{s}, \underline{b}, \underline{c}, \chi, \gamma$))
$LL = A \cdot S^{-1} \cdot X \cdot A^T$
$\underline{\delta}_D = A^T \cdot \underline{\pi} + \underline{s} - \underline{c}$
$\underline{d}_\pi = LL(\underline{b} - \chi \cdot A \cdot S^{-1}\underline{e} - A \cdot S^{-1} \cdot X\underline{\delta}_D)$
$\underline{d}_s = -\underline{\delta}_D - A^T\underline{d}_\pi$
$\underline{d}_x = S^{-1}(-X\underline{d}_s - X \cdot S\underline{e} + \chi\underline{e})$
$\beta_x = \gamma \cdot \min_k \left\{ \frac{-\underline{x}(k)}{\underline{d}_x(k)} : \underline{d}_x(k) < 0 \wedge \underline{d}_x(k) \text{ is finite} \right\}$
$\beta_\pi = \gamma \cdot \min_k \left\{ \frac{-\underline{s}(k)}{\underline{d}_s(k)} : \underline{d}_s(k) < 0 \wedge \underline{d}_s(j) \text{ is finite} \right\}$
$\underline{x} = \underline{x} + \beta_x\underline{d}_x$
$\underline{s} = \underline{s} + \beta_\pi\underline{d}_s$
$\underline{\pi} = \underline{\pi} + \beta_\pi\underline{d}_\pi$
$X = diag(\underline{x})$
$S = diag(\underline{s})$
return $\underline{x}, \underline{s}, \underline{\pi}, X, S$
end function

is the point of minimum total squared Mahalanobis distance between the two ellipses to each of the centers of the ellipses. By the definition of ellipses, overlap exists when both squared Mahalanobis distances to this point are less than 1. Formally, the query point $\underline{\mu}_{ij}$ is given by

$$\underline{\mu}_{ij} = \Sigma(\Sigma_i^{-1}\underline{\mu}_i + \Sigma_j^{-1}\underline{\mu}_j) \tag{21}$$

$$\Sigma = (\Sigma_i^{-1} + \Sigma_j^{-1})^{-1}. \tag{22}$$

These squared Mahalanobis distance tests filter approximately 92% of cases; for the remaining cases, the least squares minimization over a sphere algorithm (LSMOS) is employed [24]. The squared Mahalanobis distance tests wrap the LSMOS, which, in 2-D, is a quadratic programming problem to find the closest point on the perimeter of a circle to an ellipse.

Algorithm 9 Update Primal Variables

function *LP_primal*($\underline{x}, X_q, X_j, \underline{\alpha}, \underline{\beta}, \underline{u}, m, n_j, n_q, \rho$)
$\underline{\alpha} = \underline{x}[0 : n_j]$
$\underline{\beta} = \underline{x}[n_j : n_j + n_q]$
$\underline{\phi} = \underline{x}[n_j + n_q]$
$\underline{u} = X^q \cdot \underline{\beta} - X^j \cdot \underline{\alpha}$
$\rho = ||\underline{u}||_\infty$
$\underline{c} = \begin{bmatrix} \underline{0} \\ \rho \end{bmatrix}$
$A[1 : m, n_j + n_q + 1] = \underline{u}$
return $\underline{\alpha}, \underline{\beta}, \phi, \underline{u}, \rho, \underline{c}, A$
end function

Algorithm 10 QP When Means $\underline{\mu}_i, \underline{\mu}_j$ and Covariances Σ_i^{-1}, Σ_j^{-1} Are Given

function check4interference($\{\underline{\mu}_i, \Sigma_i^{-1}\}, \{\underline{\mu}_j, \Sigma_j^{-1}\}$)
$d_{ij} \leftarrow (\underline{\mu}_i - \underline{\mu}_j)^T(\Sigma_i + \Sigma_j)^{-1}(\underline{\mu}_i - \underline{\mu}_j)$
if $d_{ij} \leq 2$ **then**
$d_i \leftarrow (\underline{\mu}_i - \underline{\mu}_{ij})^T(\Sigma_i)^{-1}(\underline{\mu}_i - \underline{\mu}_{ij})$
if $d_i < 1 \wedge (d_{ij} - d_i) < 1$ **then**
$verdict \leftarrow 'Overlap'$
else
$verdict, \underline{x} \leftarrow$ LSMOS$\left(\{\underline{\mu}_i, \Sigma_i^{-1}\}, \{\underline{\mu}_j, \Sigma_j^{-1}\}\right)$
end if
else
$verdict \leftarrow 'Separate'$
end if
return $verdict, \underline{x}$
end function

The LSMOS algorithm is formulated by transformation of one ellipse by subtracting the mean of the other ellipse and then multiplying by the square root of the inverse covariance matrix of the other ellipse. This transformation establishes a new ellipse, and a unit circle at the origin. The objective is to find the minimum point on the unit circle to the ellipse.

The LSMOS is done using a Newton-Raphson algorithm. Algorithm 11 is a modification of the least squares minimization over a sphere from [30].

C. ELLIPSE AND POLYGON NARROW PHASE APPROACH

The case when an ellipse and a polygon are checked at the narrow phase level employs an approximation of the ellipse by a circumscribing polygon. The resulting polygons are processed by the primal-dual path following Algorithm 6. The ellipse is approximated by transforming n points around

Algorithm 11 Least Squares Minimization Over a Sphere (LSMOS)

function LSMOS($\{\underline{\mu}_i, \Sigma_i^{-1}\}, \{\underline{\mu}_j, \Sigma_j^{-1}\}$)
  $G_1 \leftarrow chol(\Sigma_1^{-1})$ ▷ Cholesky Factor of Σ_1^{-1}
  $G_2 \leftarrow chol(\Sigma_2^{-1})$ ▷ Cholesky Factor of Σ_2^{-1}

  ▷ Compute Singular Value Decomposition of $G_2^T G_1^{-T}$
  $[U, D, V] \leftarrow svd(G_2^T G_1^{-T})$

  $scaleFactor = min(diag(D))$
  $D = D/scaleFactor$
  $\tilde{\underline{b}} \leftarrow U^T G_2^T (G_1^{-T})(\underline{\mu}_2 - \underline{\mu}_1)/scaleFactor$
  $temp = \sum_i \frac{1}{D(i,i)} \tilde{\underline{b}}(i)$
  if $temp > 1$ **then**
    $\underline{z} \leftarrow$ NewtonRaphson($D, V, \tilde{\underline{b}}$) ▷ [30]
  else
    for i = 1,2,3 **do**
      $\underline{z} \leftarrow \underline{z} + \frac{1}{D(i,i)} \tilde{\underline{b}}(i) \underline{v}_i$ ▷ $V = [\underline{v}_1, \underline{v}_2, \underline{v}_3]$
    end for
  end if
  $\underline{x} \leftarrow scaleFactor \cdot (G_1^{-T})\underline{z}$
  if $\left(\underline{x} - \underline{\mu}_2 + \underline{\mu}_1\right)^T (\Sigma_2)^{-1} (\underline{x} - \underline{\mu}_2 + \underline{\mu}_1) \leq 1$ **then**
    $verdict \leftarrow 'Overlap'$
  else
    $verdict \leftarrow 'Separate'$
  end if
return $verdict, \underline{x}$
end function

the unit circle using the square root factor of the covariance matrix for the ellipse. A method to bisect facets and scale the vertices is proposed for finding a circumscribing polytope around the unit sphere. These vertices are then transformed using the square root factor of the covariance matrix of the ellipse. The pseudocode given below details the process for generating these circumscribing polygons and polytopes to ellipses and ellipsoids, respectively. The MATLAB like notation is used along with *len*(.) to indicate the number of elements in a list, and *mean*(.) which is the mean computed for each dimension. The *scaleFactor* is computed as the magnitude of the projection of any vertex of a facet onto the unit vector passing through the dual point of the facet. For the initial octahedron, the magnitude of this projection is 0.5774.

VI. RESULTS

A. BROAD PHASE: TPR-TREE

The average time for building a TPR-tree and intersection query times are shown in Table 4. The statistics are averages over 100 Monte Carlo runs. The algorithms were run on an i7 Processor with 8GB RAM running Ubuntu 16.04 LTS. The build times are negligible at less than 60s for problem sizes of 300,000 objects. For example, the tree build time takes less than 1s for 10,000 objects or less. The broad phase queries for 10,000 objects have responses on average

Algorithm 12 Generates a Polygon or Polytope to Circumscribe an Ellipse or Ellipsoid

R is the matrix of right singular vectors
Λ is the matrix of singular values
p is the position of the elliptic region
function *circumscribe*(R, Λ, p)
  Determine if the input elliptical region is 2D or 3D.
  if the input is 2D **then**
    $numVerts = 32$
    $verts = circumscribe\_circle(numVerts)$
  else
    $numBisections = 1$
    $numVerts = 8 \cdot 4^{numBisections}$
    $verts = circumscribe\_sphere(numBisections)$
  end if
  $G = R^T \Lambda^{-\frac{1}{2}} R$
  for $i = 0 : 1 : (numVerts - 1)$ **do**
    $transformedVerts(i) = G \cdot verts(i) + \underline{p}$
  end for
return *transformedVerts*
end function

Algorithm 13 Circumscribing a Circle

function *circumscribe_circle*(*numVerts*)
  $\theta = \pi / numVerts$ ▷ π *radians*
  $r = 1/\cos(\theta)$
  $angles = \theta : 2\theta : (2\pi - \theta)$
  for $i = 0 : 1 : (numVerts - 1)$ **do**
    $verts(i) = [r \cdot \cos(angles(i)), r \cdot \sin(angles(i))]$
  end for
return *verts*
end function

Algorithm 14 Circumscribing a Sphere

function *circumscribe_sphere*(*nBisections*)
  ▷ Start with a list of all the facets defined by the octahedron with vertices given in Table 3
  $scaleFactor = 0.5774$
  for $k = 1 : 1 : nBisections$ **do**
    $facets, scaleFactor = bisect\_facets(facets)$
  end for
  ▷ Compute the dual points to reduce the number of vertices
  for $0 : 1 : (len(facets) - 1)$ **do**
    $dualVerts(i) = \frac{mean(facets(i).verts)}{scaleFactor \cdot ||mean(facets(i).verts)||_2}$
  end for return *dualVerts*
end function

in less than 6.5s. Computing intersection queries via the TPR-Tree scales well beyond the application demands as seen by solving a 300,000 object intersection problem in less than 1.5 hours. A scenario requiring 5000 intersection queries can be performed in less than 2s. Since the *CON-*

TABLE 3. Vertices of an octahedron which circumscribes the unit sphere.

x	y	z
0	0	1
0	0	-1
$-\sqrt{2}/2$	$-\sqrt{2}/2$	0
$\sqrt{2}/2$	$-\sqrt{2}/2$	0
$\sqrt{2}/2$	$\sqrt{2}/2$	0
$-\sqrt{2}/2$	$-\sqrt{2}/2$	0

Algorithm 15 Bisecting Facets

function *bisect_facets*(*facets*)
  scaleFactor = 100
  for each *facet* in *facets* **do**
    for $i = 0 : 1 : 2$ **do**
      $v(i) = (facet.verts(i-1) + facet.verts(i))/2$
    end for
    $newFacets.append(facet.verts(0), v(0), v(1))$
    $newFacets.append(facet.verts(1), v(1), v(2))$
    $newFacets.append(facet.verts(2), v(2), v(0))$
    $newFacets.append(v(0), v(1), v(2))$
  end for
  $p(i) = \frac{mean(facets(i).verts)}{||mean(facets(i).verts)||_2}$
  $scaleFactor = min_{newFacets\ i}\{min_{vertex\ j}\{v_j \cdot p(i)\}\}$
return *newFacets*, *scaleFactor*
end function

TABLE 4. TPR-tree Tests.

Vessels	500	1k	5k	10k	100k	200k	300k
Average Tree Build Time (Seconds)	0.02	0.05	0.36	0.88	11.6	26.6	46.4
Intersection Query Runtime (Seconds)	0.07	0.17	1.92	6.44	517.7	2100	5030

FIDENT algorithm should process the inputs at worst on the order of minutes, the performance of the broad phase is satisfactory.

B. NARROW PHASE

The narrow phase algorithms include primal-dual path following algorithm to solve the linear program (LP) [22], and the QP approach developed in [24] for ellipses. To check for interference between an ellipse and a polygon, the ellipse must be approximated as a polygon. Table 5 shows the average run times for 100 scenarios consisting of randomly generated elliptic safety regions (one per vessel). Here, the broad phase is bypassed to evaluate the average runtimes of the narrow phase portions of CONFIDENT.

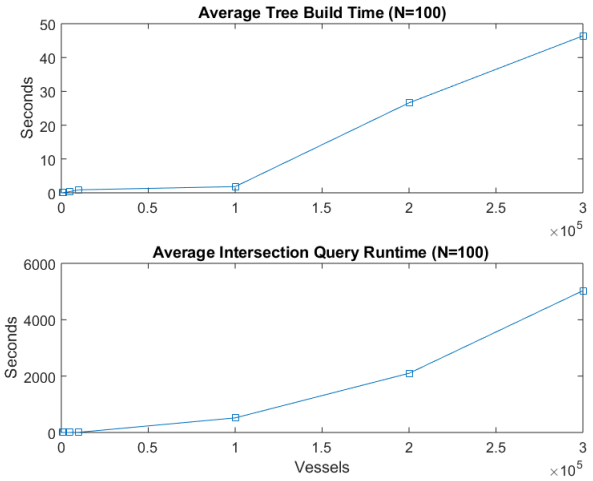


FIGURE 5. TPR-tree build and query times for simple convex polygons with 4 vertices.

TABLE 5. Narrow phase testing.

Vessels	2	10	15	32	45
Checks	1	45	105	496	990
Average Total Approximation Run-time (ms)	8.3	11.6	13.4	34.8	49.3
Average Total LP Run-time (ms)	26.4	1223	3230	10,300	20,400
Average Total QP Run-time (ms)	0.15	3.9	9.98	25.6	44.1

The ellipses are approximated as polygons with 32 vertices in order to compare the LP and QP run times on the scenarios. The average time to approximate 45 ellipses is less than 50 ms. The average total run time for checking 990 combinations of polygons is 20.4 s. A single pair of polygons is checked in 26.4 ms. Operationally, this performance is satisfactory.

The average run times for determining interferences between ellipses using the QP are shown in Table 5. The eigenvalues and eigenvectors of the covariance matrix defining the ellipses are generated from the course angles and safety region extents. The time required to compute the covariance matrices are included in the average total run times reported in Table 5. The QP can, on average, determine interference between 990 pairs of ellipses in less than 45 ms. A single pair of ellipses are checked in 150 μs. When processing mutual interferences between two ellipses, approximately 92% of cases are solved using the squared Mahalanobis

TABLE 6. CONFIDENT: ellipses only (8 CPUs).

Number of Objects	5000	10000	10000	10000	10000
Number of Collisions Pre-Narrow Phase	6894	3656	5078	7562	12720
Number of Collisions Post-Narrow Phase	2173	1164	1569	2419	4046
Time taken for Broad Phase (Seconds)	1.99	6.00	6.29	6.08	6.33
Time taken for Narrow Phase (Seconds)	0.93	0.57	0.75	1.05	1.61

distance checks, avoiding invocation of the LSMOS method. The ellipse/ellipse interference detection is very satisfactory in terms of processing time. Also, the benefit of having a rapid algorithm for ellipse/ellipse interference detection is the alternative application of tracking utilizing estimates of location described through mean and covariance.

C. CONFIDENT SYSTEM

The run times for CONFIDENT for various input types and scenarios are shown in Tables 6 - 8. These scenarios are generated using a constant number of objects positioned randomly inside a box initialized to $[-155, -145; 35, 45]$. The area of the box is enlarged by 8 degrees longitude and 2 degrees latitude at each iteration. CONFIDENT processes ellipses rapidly, as shown in Table 6. For example, a scenario with 5000 objects, and 6894 interferences identified by the broad phase had a narrow phase processing time of less than 1s. The average time per object in the broad phase is 0.13 μs. The speed of the TPR-tree is due to its ability to collect all of the members that intersect a query region in one step; this results in $O(N)$ complexity. The average time per object in the narrow phase is 0.14 ms. The complexity of CONFIDENT is $O(N^2)$ due to the need for invoking narrow phase pair-wise.

Table 7 lists the run times for CONFIDENT on convex polygons with a random number of vertices between 4 and 16. The LP narrow phase requires more time to process per conflict identified at the broad phase level. For a scenario with 932 conflicts identified by the broad phase, the processing time is 318.16s.

Table 8 shows the run times for CONFIDENT on scenarios containing only non-convex polygons with a random number of vertices between 4 and 16. A python implementation of *A5m*, termed Py2D, is used for convex decomposition with holes. The non-convex polygons contain a single hole that have 4 random vertices. The requirement to convexify the polygon, for LP narrow phase processing, increases the number of broad-phase conflicts as seen by comparing Tables 7 and 8. In Table 8, the number of collisions pre-narrow phase corresponds to the number of convex pieces across area assignments that are in conflict. The number of collisions post-narrow phase is the number of area assignments in conflict. The larger number of pre-narrow phase conflicts results in the increased run time for narrow phase processing of non-convex inputs as compared to scenarios with only convex inputs.

TABLE 7. CONFIDENT: convex polygons only (8 CPUs).

Number of Objects	5000	10000	10000	10000	10000
Number of Collisions Pre-Narrow Phase	1134	932	1316	1676	2320
Number of Collisions Post-Narrow Phase	439	360	487	635	899
Time taken for Broad Phase (Seconds)	1.85	7.66	6.32	7.48	7.75
Time taken for Narrow Phase (Seconds)	402.72	318.16	438.85	580.04	790.43

TABLE 8. CONFIDENT: non-convex polygons with 1 hole only (8 CPUs).

Number of Objects	5000	10000	10000	10000	10000
Number of Collisions Pre-Narrow Phase	7374	7198	8414	12240	18140
Number of Collisions Post-Narrow Phase	391	365	457	664	988
Time taken for Broad Phase (Seconds)	2.65	6.81	7.05	7.91	9.57
Time taken for Narrow Phase (Seconds)	892.17	879.62	1018.24	1534.61	2233.50

VII. CONCLUSION AND FUTURE WORK

CONFIDENT is a novel system that determines mutual interferences between any combination of ellipses and polygons of any genus (number of holes). This system provides planners the ability to define any 2-D model of obstacles and safety regions in order to accurately define realistic descriptions of interference at sea. Specifically, CONFIDENT solves an essential problem for uncertain asset positions introduced by submarine operations. The regions where a submarine operates must be modeled by 2-D shapes due to the uncertain position of a submerged submarine. Thus, planning for position uncertainty, and modeling interferences, via CONFIDENT, allows planners to generate safer operations. The planning cycle using CONFIDENT is reduced to an estimated 2 to 4 man hours, which is a substantial savings when compared to the estimated 120 man hours when done manually. The algorithm scales to solve problems of practical scale, viz., thousands of objects, on the order of seconds.

CONFIDENT can be improved as follows. The 2-D convex decomposition algorithm can be extended to moving

objects. Incorporating 3-D convex decomposition for static and dynamic regions would complete the functionality for processing any 2-D or 3-D interferences. A comparison with other libraries that process polygon/polygon interference queries is of interest to enhance the speed of processing.

ACKNOWLEDGMENT

G. V. Avvari was with the University of Connecticut, Storrs, CT 06269-4157, USA.

REFERENCES

[1] J. McDermott. (Jul. 2011). Electric boat gets USS hartford back to sea. The New London Day. [Online]. Available: http://www.theday.com/article/20110717/NWS09/307179942/1018

[2] B. Mirtich, "Efficient algorithms for two-phase collision detection," Mitsubishi Electr. Res. Lab., Cambridge, MA, USA, Tech. Rep., Dec. 1997. [Online]. Available: http://www.merl.com/publications/docs/TR97-23.pdf

[3] J. O'Rourke, *Computational Geometry in C*. Cambridge, U.K.: Cambridge Univ. Press, 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=521378

[4] M. Held, "FIST: Fast industrial-strength triangulation of polygons," *Algorithmica*, vol. 30, no. 4, pp. 563–596, 2001. [Online]. Available: https://link.springer.com/article/10.1007/s00453-001-0028-4

[5] R. Seidel, "A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons," *Comput. Geometry Theory Appl.*, vol. 1, pp. 51–64, Mar. 1989. [Online]. Available: http://dl.acm.org/citation.cfm?id=117043

[6] A. Narkhede and D. Manocha, "Fast polygon triangulation based on Seidel's algorithm," in *Graphics Gems V*. San Diego, CA, USA: Elsevier, 1995. [Online]. Available: http://gamma.cs.unc.edu/SEIDEL/

[7] S. Hertel and K. Mehlhorn, "Fast triangulation of simple polygons," in *Foundations of Computation Theory*. Berlin, Germany: Springer-Verlag, 1983. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-12689-9_105

[8] J. Fernández, L. Cánovas, and B. Pelegrín, "Algorithms for the decomposition of a polygon into convex polygons," *Eur. J. Oper. Res.*, vol. 121, no. 2, pp. 330–342, 2000. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221799000338

[9] J. Fernández, B. Toth, L. Cánovas, and B. Pelegrín, "A practical algorithm for decomposing polygonal domains into convex polygons by diagonals," *TOP*, vol. 16, no. 2, pp. 367–387, 2008. [Online]. Available: https://link.springer.com/article/10.1007/s11750-008-0055-2

[10] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," in *Proc. ACM SIGGRAPH*, 1996, pp. 171–180. [Online]. Available: http://gamma.cs.unc.edu/SSV/obb.pdf

[11] C. L. Jackins and S. L. Tanimoto, "Oct-trees and their use in representing three-dimensional objects," *Comput. Graph. Image Process.*, vol. 14, no. 9, pp. 249–270, 1980. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0146664X80900556

[12] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975. [Online]. Available: http://dl.acm.org/citation.cfm?id=361007

[13] F. ganovelli, J. Dingliana, and C. O'Sullivan, "BucketTree: Improving Collision Detection Between Deformable Objects," in *Proc. Spring Conf. Comput. Graph.*, 2000, p. 19. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.3994

[14] Y. Manolopoulos, A. Nanopoulos, A. N. Papadopoulos, and Y. Theodoridis, *R-Trees: Theory and Applications*. New York, NY, USA: Springer-Verlag, 2006.

[15] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Inf.*, vol. 4, no. 1, pp. 1–9, Apr. 1974. [Online]. Available: https://link.springer.com/article/10.10072FBF00288933

[16] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57. [Online]. Available: http://www.cs.uml.edu/~cchen/580-S06/reading/Gut84.pdf

[17] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in *Proc. ACM SIGMOD Rec.*, 1990, pp. 322–331. [Online]. Available: https://www.cs.umd.edu/class/fall2002/cmsc818s/Readings/rstar-tree.pdf

[18] Y. Tao and D. Papadia, "Time-parameterized queries in spatio-temporal databases," in *Proc. ACM SIGMOD*, Jun. 2002, pp. 334–345. [Online]. Available: https://pdfs.semanticscholar.org/5625db5a3c33cfda7f0f135223400c16ccdd4c23.pdf

[19] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE J. Robot. Autom.*, vol. 4, no. 2, pp. 193–203, Apr. 1988. [Online]. Available: http://ieeexplore.ieee.org/document/2083/

[20] G. van den Bergen, "A fast and robust GJK implementation for collision detection of convex objects," *J. Graph. Tools*, vol. 4, no. 2, pp. 7–25, 1999. [Online]. Available: http://dl.acm.org/citation.cfm?id=334711

[21] S. A. Ehmann and M. C. Lin, "Accelerated proximity queries between convex polyhedra by multi-level Voronoi marching," in *Proc. Int. Conf. Intell. Robots Syst.*, Oct./Nov. 2000, pp. 2101–2106. [Online]. Available: http://ieeexplore.ieee.org/document/895281/

[22] A. Akgunduz, P. Banerjee, and S. Mehrotra, "A linear programming solution for exact collision detection," *Trans. ASME*, vol. 5, no. 1, pp. 48–55, 2005. [Online]. Available: http://computingengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1400139

[23] B. Mirtich, "V-Clip: Fast and robust polyhedral collision detection," Mitsubishi Electr. Res. Lab., Cambridge, MA, USA, Tech. Rep. 1, Dec. 1997. [Online]. Available: http://dl.acm.org/citation.cfm?id=285860

[24] D. McMenemy, D. Sidoti, K. R. Pattipati, and F. A. N. Palmieri, "A conflict detection method for ellipsoidal safety regions," *IEEE Trans. Aerosp. Electron. Syst.*, to be published. [Online]. Available: https://ieeexplore.ieee.org/document/8531769

[25] W. Wang, J. Wang, and M.-S. Kim, "An algebraic condition for the separation of two ellipsoids," *Comput. Aided Geometric Des.*, vol. 18, no. 6, pp. 531–539, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=510881

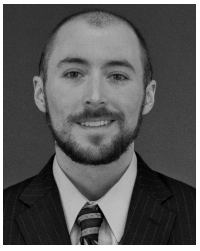
[26] S. B. Pope, "Algorithms for ellipsoids," Cornell Univ., Ithaca, NY, USA, Tech. Rep. FDA-08-01, Feb. 2008. [Online]. Available: https://tcg.mae.cornell.edu/pubs/Pope_FDA_08.pdf

[27] P. A. Jensen and J. F. Bard, "Linear programming," in *Operations Research Models & Methods*, vol. 1, 1st ed. Hoboken, NJ, USA: Wiley, 2003. [Online]. Available: http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471380040.html

[28] International Maritime Organization. (Sep. 22, 2009). *Colregs: International Regulations for Preventing Collisions at Sea*. Accessed: Jun. 18, 2018. [Online]. Available: http://www.jag.navy.mil/distrib/instructions/COLREG-1972.pdf

[29] M. Hadjieleftheriou. (2014). *Libspatialindex*. [Online]. Available: https://libspatialindex.github.io/

[30] G. H. Golub and C. F. van Loan, *Matrix Computations*, 4th ed. Baltimore, MD, USA: The Johns Hopkins Univ. Press, 2013, p. 409.



DONALD MCMENEMY received the B.S. degree in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA, in 2014, where he is currently pursuing the Ph.D. degree under the supervision of Dr. K. R. Pattipati. He was with Belcan, LLC, Windsor, CT, USA, from 2014 to 2016. His research interests include algorithms for path planning.

GOPI VINOD AVVARI received the B.Tech. degree in electronics and communication engineering from Acharya Nagarjuna University, Guntur, India, in 2011, and the M.S and Ph.D. degrees in electrical engineering from the University of Connecticut, in 2016 and 2018, respectively. He is currently the Lead of the Data Science Team, Aptiv, Kokomo, IN, USA. The primary focus of his research interests includes developing algorithms for the advanced driver assisting systems and autonomous driving.

DAVID SIDOTI received the B.S., M.S., and Ph.D. degrees in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA, in 2011, 2016, and 2018, respectively, under the supervision of Dr. K. R. Pattipati. His current research interests include multi-objective algorithms for dynamic scheduling, resource management in weather-impacted environments, information valuation, combining optimization techniques with deep learning, and sailing vessel routing. He was a co-recipient of the Tammy Blair Award for Best Student Paper at FUSION 2016.

ADAM BIENKOWSKI received the B.S. and M.Eng. degrees in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA, in 2013 and 2017, respectively, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering Department, under the supervision of Dr. K. R. Pattipati. He was an Electrical Engineer with General Dynamics Electric Boat, Groton, CT, USA, from 2013 to 2017. His current research interests include modeling dynamic and uncertain environments for asset allocation and path planning, context aware decision support systems, and optimization and machine learning-based techniques for mission planning and coordination.

KRISHNA R. PATTIPATI (S'77–M'80–SM'91–F'95) received the B.Tech. degree (Hons.) in electrical engineering from IIT Kharagpur, Kharagpur, in 1975, and the M.S. and Ph.D. degrees in control and communication systems from the University of Connecticut (UConn), Storrs, in 1977 and 1980, respectively. He was with AlphaTech, Inc., Burlington, MA, USA, from 1980 to 1986. He has been with the Department of Electrical and Computer Engineering, UConn, where he is currently the Board of Trustees Distinguished Professor and the UTC Chair Professor in systems engineering. He is also a Co-Founder of Qualtech Systems, Inc., a firm specializing in advanced integrated diagnostics software tools (TEAMS, TEAMS-RT, TEAMS-RDS, TEAMATE, and PackNGo), and serves on the board of Aptima, Inc. His research activities are in the areas of proactive decision support, uncertainty quantification, smart manufacturing, autonomy, knowledge representation, and optimization-based learning and inference. A common theme among these applications is that they are characterized by a great deal of uncertainty, complexity, and computational intractability.

Dr. Pattipati is an Elected Fellow of the IEEE and the Connecticut Academy of Science and Engineering. He was selected by the IEEE Systems, Man, and Cybernetics (SMC) Society as the Outstanding Young Engineer of 1984, and received the Centennial Key to the Future Award. He was a co-recipient of the Andrew P. Sage Award for the Best SMC Transactions Paper for 1999, the Barry Carlton Award for the Best AES Transactions Paper for 2000, the 2002 and 2008 NASA Space Act Awards for A Comprehensive Toolset for Model-Based Health Monitoring and Diagnosis, and Real-time Update of Fault-Test Dependencies of Dynamic Systems: A Comprehensive Toolset for Model-Based Health Monitoring and Diagnostics, and the 2003 AAUP Research Excellence Award at UConn. He has served as the Editor-in-Chief of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS–Part B, from 1998 to 2001.