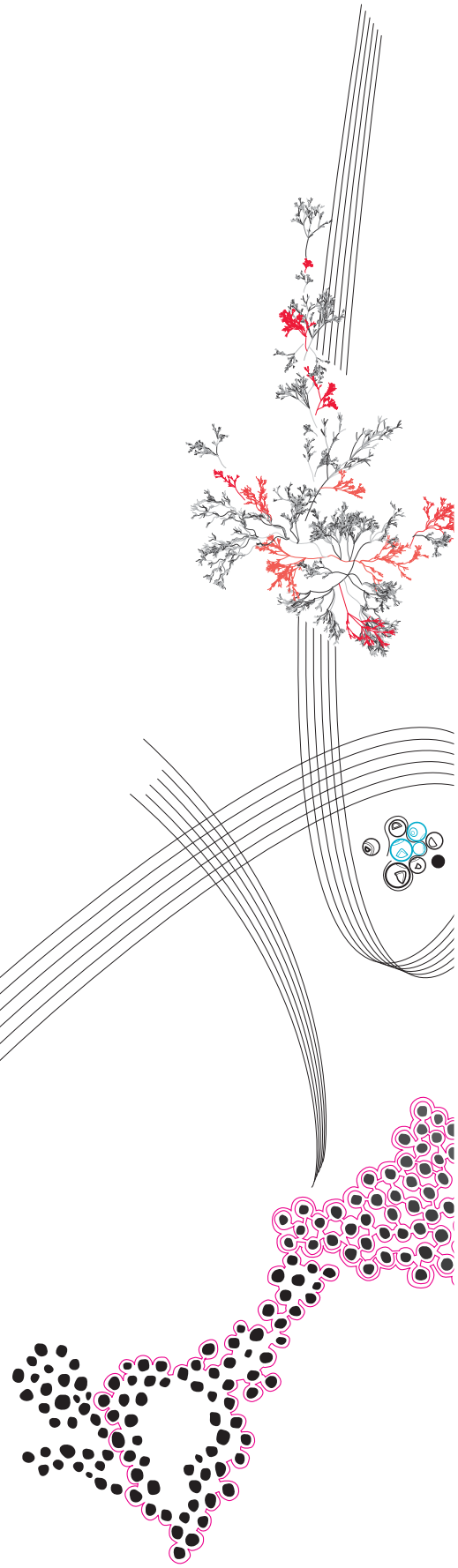


MSc Computer Science
Final Project

Computing Electro-Optical and Radar coverage in near real-time

Andrey Jaroslaw Antonowycz

Supervisor: dr. Tom van Dijk (UT) & Ir. Harm Jan Spier (Thales)

March, 2024

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Contents

Abstract

In a model of the real world, determining whether an object is in *Line of Sight* of an observer requires performing a large series of simple calculations to verify there is no terrain or object obstructing the sight-line. Upon alteration of the simulation state due to a moving observer or target, the aforementioned calculations are no longer valid and will need to be re-executed. When a time constraint is applied, the concept of a large number of small calculations yields an interesting challenge; how can one structure these calculations such that they can be reliably performed within the required time interval. Furthermore, approximation techniques can account for additional latency improvements at the cost of precision or fidelity. In this paper, different approaches to solve this problem for electro-optical and radar sensors are compared to determine a configuration which provides the best results given a set of (latency and resource) limitations.

Keywords: line of sight, digital elevation model, sensor, electro-optical, radar

Chapter 1

Introduction

A radar system uses the reflection of emitted radio waves to determine the distance, elevation and azimuth between itself and a target. An electro-optical sensor, in its simplest form, is a piece of equipment responding to changes in light using the shortest optical distance between an observer and a target. In the real world, both sensors adhere to the laws of physics that govern electro-magnetic wave propagation. By definition, a model is a simplification of the real world. As such, in a simulated environment, physical behavior is mirrored as closely as possible but never perfectly. Given an environment, determining what is visible from a certain viewpoint is a solved problem; given the positions of all objects in the area one can 'brute-force' to check for collisions between the sight-line, terrain and objects. However, radar and electro-optical sensors are often used on non-stationary objects like ships or airplanes, and their position is therefore subject to change. Changing the position of the observer invalidates previous *Line of Sight*-calculations due to changes in perspective. For example, a small change in perspective can result in a target disappearing behind a mountain or vice versa. To handle information processing, there is an interest in achieving accurate, fast and efficient calculations of radar and electro-optical sensors coverage. In particular, in a dynamic environment with the potential for accounting for a large amount of moving objects (or alternatively; entities). In particular, how should these computations be executed such that they can be achieved in near real-time, and which approximations are viable for real world use-cases.

In this paper, a set of possible approximation techniques for modeling electro-optical and radar sensor behavior are studied. Using principles explained in chapter 2, a *Line of Sight*-service has been developed in cooperation with Thales Netherlands B.V., situated in Hengelo, Overijssel. As a leading manufacturer of marine radar systems, Thales is primarily interested in the naval domain. As such, Thales will be able to establish a set of system requirements based on the approaches and approximations studied in this paper along with their applicability and implications in a naval environment. The goal of the aforementioned *LoS*-service is to serve as a tool in order to answer the research questions stipulated in chapter 3 using the scientific method, whilst providing Thales with a solid framework to productize potential spin-offs.

Chapter 2

Background

2.1 Electro-optical sensor visibility in a vacuum

The most simplistic representation of the *Line of Sight*-problem is geometric *Line of Sight*. In this model, light rays are modeled as geometric lines with an origin and 3-dimensional direction vector. In three-dimensional space, it is advantageous to use the Earth-Centered Earth-Fixed (ECEF) reference system. In ECEF, the origin is the centre of the Earth, and the z-axis crosses the north- and south pole. The use of geometry as a reduction in complexity works well in a vacuum, given the absence of an atmosphere to interact with. In a vacuum, a photon (or light ray) travels in a straight line when negating the effects of gravity. Atmospheric effects can greatly impact sensor coverage and can, for example, change the maximal view distance through refraction [13]. Approaching optical *LoS* as propagating in a vacuum allows the use of geometric approximations to determine the angle between an observer and a target, as there is no need to account for gradients in temperature or humidity as a function of height. Therefore, these calculations are far less resource-intensive. Furthermore, when assuming a flat Earth as an approximation for small distances, determining the angle between an observer and a target is trivial. Figure 2.1 shows an illustration of this example, and equation 2.1 demonstrates the simplistic nature of the calculation.

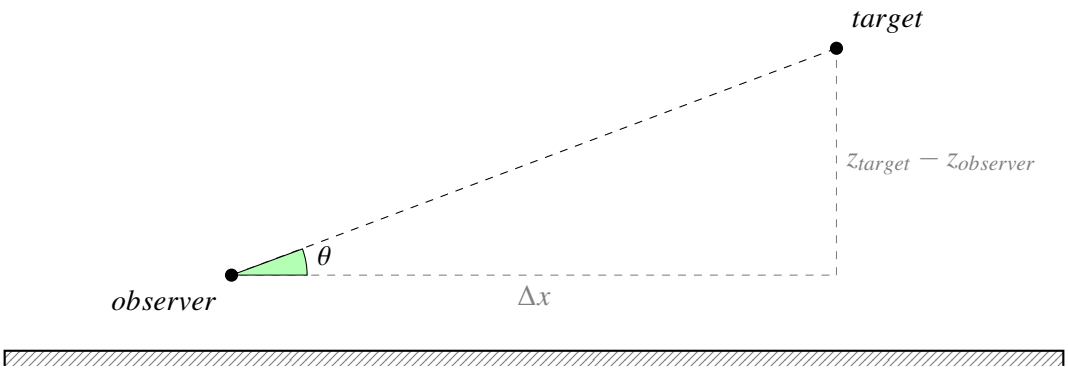


FIGURE 2.1: Determining the angle between an observer and a target on a flat Earth approximation only requires knowledge of the difference in height and the horizontal distance between the observer and target.

$$\theta_{observer \to target} = \arctan\left(\frac{z_{target} - z_{observer}}{\Delta x}\right) \tag{2.1}$$

Here z_{target} and $z_{observer}$ are the elevation of the observer and target respectively, relative to a common reference frame such as the surface of the Earth. Δx is the horizontal distance between the observer and target.

Assuming a flat Earth, the question of determining *LoS* is reduced to comparing angles; by sampling the terrain and determining the angle between the observer and the sample using formula 2.1, and comparing it against the angle between the observer and the target. If any intermediate terrain sample angle $\theta_{observer \to terrain} > \theta_{observer \to target}$, the sight-line between observer and target is obstructed. If the contrary holds, there is *Line of Sight*; the target is visible from the perspective of the observer (equation 2.2).

$$LoS = \begin{cases} true & \text{if } \forall (x,z) \in S : \arctan\left(\frac{z-z_0}{x-x_0}\right) < \arctan\left(\frac{z_{target}-z_0}{x_{target}-x_0}\right) \\ false & \text{otherwise} \end{cases} \tag{2.2}$$

2.1.1 Collision detection

In a simulated environment, for a given sight-line, there is a set of candidate objects that are near and can possibly obstruct the view. These objects must each be checked for intersection using a mathematical expression dependent on the shape of the object. An example of this is shown in figure 2.2. In this simplified 2-dimensional visualization, checking whether p, q or r intersect with sight-line $a \to b$ requires querying the position and radius of p, q and r and performing a calculation to check for an intersection with a circle. Due to sight-line $a \to b$ being obstructed if at least one object crosses the sight-line, the algorithm can terminate after finding one intersection.

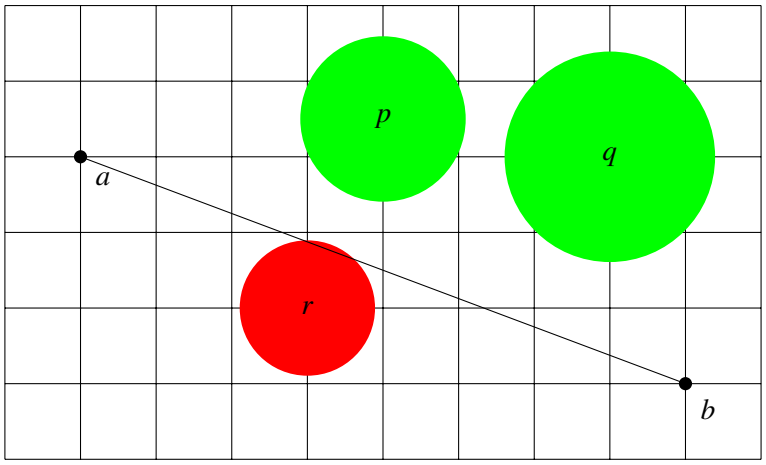


FIGURE 2.2: A 2-dimensional illustration of *LoS* and the issues that arise when the number of objects increases. For the sightline $a \to b$, objects p, q, r, ... are queried for their location and radius to check whether they intersects with the line. The calculation can be aborted once one of the circles intersects, however, all circles will have to be queried in order to ensure none of them intersect.

Objects in a simulated environment conventionally consist of meshes of polygons. These polygons are connected to each other to give the impression of a smooth surface and a complex object contour. In theory, to check whether any given object obstructs a sight-line, a query must be performed for each of the polygons in the mesh in a given area to check for intersection between the sight-line and the polygon. If the surface of the polygon crosses the sight-line, the polygon is 'in front' of the target with respect to the perspective of the observer and the sight-line is obstructed. This implies that for complex, high-polygon-count objects, the computational time to determine *Line of Sight* is much greater than for simple models. Therefore, it is beneficial to reduce the number of faces and vertices an object contains in order to reduce complexity and computational resources required. The formulae to determine intersection between a line and a primitive geometric shape are well-established. In this paper, intersection computation is limited to spheres and triangles. More complex shapes are a combination of multiple of these simplistic shapes in the underlying model representations.

Perfect sphere

For a sphere with origin $\vec{O}$ and radius r, the problem of determining whether a line intersects with the sphere is reduced to whether the distance between the origin or any point on the line is less than r. Formula 2.3 determines whether a line intersects with a sphere.

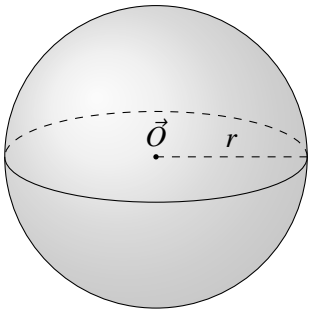


FIGURE 2.3: 3-dimensional sphere, with origin $\vec{O}$ and radius r.

$$\text{intersection}_{\vec{O}_{line}+\vec{v}\rightarrow\vec{O}_{sphere}+r} = \begin{cases} true & \text{if } (\vec{v}\cdot\vec{a})^2 \geqslant \vec{a}\cdot\vec{a} - r^2 \text{ where } a = \vec{O}_{sphere} - \vec{O}_{line} \\ false & \text{otherwise} \end{cases} \quad (2.3)$$

Here, the equation for the sphere is denoted as $\vec{O}_{sphere} + r$, and the line as $\vec{O}_{line} + \vec{v}$.

Orthogonal triangle

Determining whether a line intersects with a triangle (figure 2.4) can be computed by using algorithm 1. In this algorithm, an orthogonal triangle is used. I.e., the requirement $\vec{AB} \measuredangle \vec{AC} = 90^\circ$ must be satisfied. Two orthogonal triangles can form a rectangle, or any other combination of triangles can form a series of complex shapes. Then, for every individual triangle, algorithm 1 is applied.

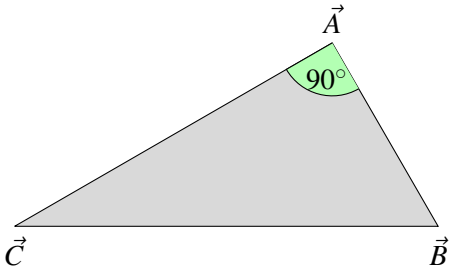


FIGURE 2.4: Orthogonal triangle in 3-dimensional space defined by the vertices $\vec{A}$, $\vec{B}$, $\vec{C}$.

Algorithm 1 Möller–Trumbore intersection algorithm [19]: Algorithm for determining whether a line defined by the equation $\vec{O} + \vec{v}t$ intersects a triangle defined by $\vec{A}$, $\vec{B}$ and $\vec{C}$

Require: $\vec{AB} \measuredangle \vec{AC} = 90^\circ$ ▷ Must be an orthogonal triangle
$\vec{e_1} \leftarrow \vec{B} - \vec{A}$ ▷ Vector between $\vec{A}$ and $\vec{B}$
$\vec{e_2} \leftarrow \vec{C} - \vec{A}$ ▷ Vector between $\vec{A}$ and $\vec{C}$
$\vec{N} \leftarrow \vec{e_1} \times \vec{e_2}$ ▷ Normal vector
$d \leftarrow -\vec{v} \cdot \vec{N}$
if d = 0 **then return** *false* **end if** ▷ Vector is parallel to the surface
$\vec{AO} \leftarrow \vec{O} - \vec{A}$
$\vec{DAO} \leftarrow \vec{AO} \times \vec{v}$
$u \leftarrow \frac{1}{d}\vec{e_2} \cdot \vec{DAO}$
$v \leftarrow \frac{-1}{d}\vec{e_1} \cdot \vec{DAO}$
$t \leftarrow \frac{1}{d}\vec{AO} \cdot \vec{N}$
return $t \geqslant 0 \wedge u \geqslant 0 \wedge v \geqslant 0 \wedge u - v \leqslant 1$

2.2 Radar visibility

2.2.1 Atmospheric ducting

Guglielmo Marconi sent a series of the Morse representation of the letter 'S' over the Atlantic ocean in 1901 [4], much further than was originally anticipated to be possible due to the Earth's curvature. This would not have been possible without radar refracting in the atmosphere. When specific conditions are met, a phenomenon coined as 'ducting' occurs, where the waves are trapped and greatly exceed their normal range. The layers in the atmosphere act as a waveguide for the radar signal. The sudden change in the refractive index bends the signal upwards or downwards in such a way that the wave is carried forward, similar to how light travels in a fiber-optic cable. Empirical analysis has been performed with the goal of approximating the behavior of waves propagating over large bodies of water [3]. A visualization of this phenomenon is displayed in figure 2.5.

The height of a duct differs for different regions on Earth, and is very situational and dependent on the weather. For use in this paper, atmospheric data for different environments can be found in previous research on the topic [12], and in cases where no data is available a plausible average is chosen based on surrounding environments. From existing literature, a set of known average evaporation duct heights is displayed in table 2.1.

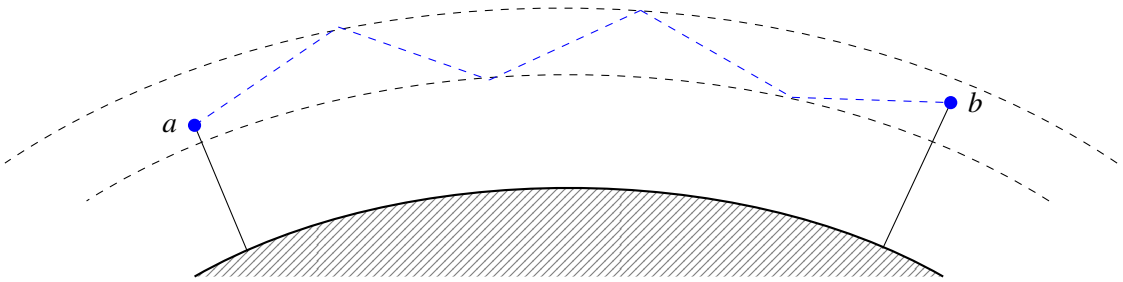


FIGURE 2.5: Visualization of the phenomenon of atmospheric ducting, where a radio wave's range is extended due to entrapment in a duct caused by a gradient in the Earth's atmospheric temperature and humidity.

Average duct height for known regions	
Region	Average duct height [m]
Labrador Sea	3.5
North Sea	6.4
Norwegian Sea	6.7
Bay of Biscay	8.2
Adriatic Sea	10.9
Aegean Sea	13.1
Arabian Sea	14.8
Gulf of Mexico	16.6
Caribbean Sea	17.6

TABLE 2.1: Average annual evaporation duct heights for a given area [12].

2.2.2 Parabolic Equation propagation model

Radar waves propagate in accordance with the laws governing all electro-magnetic radiation. In the field of electrodynamics, Maxwell's equations (equation B.1 [18]) are a well-established set of differential equations which describe the relation between magnetic and electric fields. From these equations, the solution for the split-step parabolic equation can be derived, resulting in equation B.14. The full extent of this derivation is available in appendix B. In this general form, the field is expressed as a function of surface distance x after applying the paraxial approximation [11]. Previous work on modeling the propagation of radar waves [20] resulted in a MATLAB®-tool using the one-way and two-way split-step parabolic equation (SSPE). One-way and two-way in this context entails neglecting or accounting for backward wave propagation (e.g., waves bouncing back after interacting with a mountain or object) respectively.

The numerical solution of the one-way SSPE for narrow angles ($\leqslant 10-15$ deg) [20] is given by the following equation [14]:

$$u(x+\Delta x,z) = \exp\{ik(n^2-1)\frac{\Delta x}{2}\}F^{-1}\{\exp\{-i(k\sin\theta)^2\frac{\Delta x}{2k}\}F\{u(x,z)\}\} \tag{2.4}$$

The numerical solution of the one-way SSPE for wide angles ($> 10-15$ deg) [20] is given by the following equation [24]:

$$u(x+\Delta x,z) = \exp\{ik(n-1)\Delta x\}F^{-1}\{\exp\{-\frac{i(k\sin\theta)^2\Delta x}{k\sqrt{1-\frac{(k\sin\theta)^2}{k^2}}+1}\}F\{u(x,z)\}\} \tag{2.5}$$

Here k is the wavenumber where $k = \frac{2\pi}{\lambda}$, x is the horizontal distance with respect to the radar origin, z is the elevation, θ is the angle with respect to the horizon, n is the refractive index, and F is a Fourier transformation (and F^{-1} the inverse Fourier transformation).

For the two-way SSPE for narrow and wide angles, the wave is split into a forward and backward component upon colliding with the terrain ($h_{terrain}(x) \geqslant z$) until the field strength dissipates. PETOOL [20] provides an extensive array of environmental effects to influence factors like atmospheric ducting, and is able to perform calculations based on Digital Elevation Models (DEMs). However, the license of PETOOL is relatively strict and as such, for the scope and use-case presenting in this paper, an alternative solution is preferred.

2.2.3 Ray tracing propagation model

The resource-usage of PETOOL is approximately $1.5-4$ ms per Δx-step for a 1.6GHz dual-core processor, depending on properties like terrain smoothness/conductivity [20]. Here, the number of steps depends on the distance between the observer and target and can be in the range of multiple thousands. Even when considering recent advancement in computing, the time and resources required for a single query would exceed the (near) real-time requirement. For the use-case presented in this paper, the goal is to find a method which is sufficiently accurate whilst being able to perform its calculations in (near) real-time. An alternative method to model radio wave propagation is ray tracing. Here, the path a ray traverses is modelled by solving Snell's equation for stratified atmosphere [8] [2]. In this model, the atmosphere is approximated as a set of layers, and the path of individual rays are traced after applying the effects of transitioning from one layer to another. Differences in density and temperature cause the rays to bend, thereby altering the sensor coverage with respect to a model in a vacuum.

PETOOL and ray tracing are found to largely agree within a range of 60 kilometers when not considering *beyond-LoS* [8]. In *beyond-LoS* areas located far beyond the curvature of the Earth and behind obstacles can still be visible due to the interaction between the radar waves and the troposphere. Given the assumption that modern radar systems can far exceed the range of 60 kilometers, this model does not suffice. However, beyond that range, the curvature of the Earth introduces a height drop of $h \approx r(1-\cos\{\frac{360^\circ}{d}\}) = 282.53$ meters for Earth radius $r = 6371km$, Earth circumference $d = 40030km$. Therefore, for a terrain with peaks and for objects near the coast that are located below this threshold, such an approximation will be accurate enough for collision detection. This weakness should be considered when modeling radar for long distances using ray tracing. PETOOL remains the dominant tool for high-accuracy long-distance simulations, but it does not meet the real-time latency constraints.

2.2.4 Radar cross section

In radar simulation models, the surface of an object influences the amount of reflected signal the radar is able to retrieve. The amount of reflected radiation depends on several factors, including

the size of the object, the material, the angle at which the radar hits the target and the properties of the radar itself like the polarization of the transmitted radio waves. For the purpose of modelling a radar system, these factors are often combined into a single physical variable; radar cross-section (RCS) σ [m^2]. The RCS provides an abstraction on top of what would be a complicated computations, such that the surface area represented by the RCS can be interpreted as a the cross-section of a perfectly isotropic sphere reflecting equally in all directions [27]. This constant is used in the Parabolic Equation method to determine the signal-to-noise ratio of the bidirectional path between an observer and a target. Calculating the RCS is a difficult process, and outside the scope of this paper. For situations where the value of this constant is unknown, $\sigma = 1\ m^2$ is used as a placeholder, as values of σ are commonly in this order of magnitude [22].

2.2.5 Terrain data representation

For the purpose of determining *Line of Sight*, it is common to express the terrain as a grid of points instead of a mesh of polygons [7]. When expressing the terrain as a grid of elevation data, it is possible to check whether the height at a given location is greater or less than the required height for an object further away from it to be visible. For example, in the case of an observer at point a, and target at point b. A mountain in-between point a and b obstructs the sight-line $a \rightarrow b$ when the angle between point a and the mountain top exceeds the angle between point a and point b. If the angle between the observer and the target is greater than the angle between the observer and the mountain top, there is *Line of Sight*. In a marching algorithm, the maximum angle is recorded and propagated forward. This simple algorithm circumvents calculating collision detection with the Earth's surface in the form of a mesh of polygons. The grid representation of the Earth imposes inherent limitations on the landscape. Any coordinate on the grid maps to exactly one elevation value, it is therefore implied that for a grid point with height $H : \forall h < H$ the terrain at height h is opaque. Therefore, a concave mountain or cave cannot be encoded in this format.

This representation is limited to terrain data, and does not apply to moving objects like vehicles or complex static objects like trees. For this purpose, vector or mesh representations of these objects can be employed. A possible format to encode the 3-dimensional model representation in is an ArcGIS Shapefile [1], an Esri vector data format used in Geographic Information Systems for embedding models and metadata of geographic features. The Shapefile can be parsed to extract its position and geometric shape to check for collisions by performing collision detection on every vertex of the selected model. Thereby requiring more computing power for a model with more vertices and faces. To counteract this, a mesh simplification pre-computation step such as an implementation based on the whale optimization algorithm [15] can be applied beforehand.

2.3 Coordinate reference systems

When considering a model of the Earth, the landscape can be represented as a collection of height data points relative to the Earth ellipsoid (an approximation of the Earth surface taking into account its ellipsoidal shape). A Coordinate Reference System (CRS) is an approximation of the shape of the Earth, relative to which a location can be determined. For example, the coordinate $(\text{longitude } 10^\circ, \text{latitude } 40^\circ)$ cannot be directly mapped to a location in 3-dimensional space without defining its reference system. Several Coordinate Reference Systems exist, the most conventional and widely-used being 'World Geodetic System 1984' (WGS84). A CRS comprises

of two constants that approximate the height (semi-minor, polar axis) b and width (semi-major, equatorial axis) a radii of the Earth. Most publicly available data sets which have utilized satellites to measure the height of the terrain do not express the height of a coordinate with respect to the centre of the Earth, but rather with respect to such a CRS, conventionally WGS84. In this paper, DLR TanDEM-X [26] with respect to WGS84 is used unless stated otherwise. This data set is free to use for academic purposes as per their license agreement and contains data with a resolution of approximately 90 meter. Due to the dynamic nature of the sea level (due to the season, tides, etc.), experiments will be exclusively performed on land.

2.3.1 Cartesian to LLA and vice versa

Conversion between Cartesian (x, y, z) and LLA (longitude l, latitude μ, altitude h) is required to use DEMs for *LoS*-calculations. The elevation data is encoded as a 2-dimensional grid but the sight-line propagates as a Cartesian vector in 3-dimensional space. A straight vector on the DEM would result in a curved line as the DEM is relative to the curved ellipsoid. Equation 2.6 describe the translation of LLA (l, μ, h) to the Cartesian coordinate system with intermediate variables λ and r (equation 2.7) representing the geocentric latitude at mean sea-level and the radius at a surface point respectively [17]. In this paper, Cartesian and ECEF (Earth-Centered, Earth-Fixed) are used interchangeably. ECEF is a reference system where point $(0, 0, 0)$ is the centre of the Earth, the x-axis intersects with coordinate (longitude 90°, latitude 0), the y-axis intersects with (longitude 0, latitude 0) and the z-axis intersects with the north and south pole.

$$\begin{aligned} x &= r\cos\lambda\cos l + h\cos\mu\cos l \\ y &= r\cos\lambda\sin l + h\cos\mu\sin l \\ z &= r\sin\lambda + h\sin\mu \end{aligned} \tag{2.6}$$

$$\begin{aligned} \lambda &= \arctan\{(1-f)^2\tan\mu\} \\ r &= \sqrt{\frac{a^2}{1+((1-f)^{-2}-1)\sin^2\lambda}} \end{aligned} \tag{2.7}$$

Conversion back to LLA from a Cartesian representation is a more computationally intense operation, and requires a predefined number of iterations until convergence using Bowring's method [5]. The algorithm of translating a Cartesian point to LLA is defined in algorithm 2 [16].

2.4 Viewshed analysis

Viewshed analysis is a well-established method of mapping visibility of surrounding points given a viewpoint and a collection of height data points. Computing a viewshed beforehand can yield results of *LoS*-query to all surrounding points on a live system, reducing an *LoS*-query to a look-up table request. Originating from the viewpoint, the minimal angle required for objects to be visible beyond the horizon are recorded as the sight-line originating from the viewpoint propagates outwards. The main advantage of this method is its sequential execution of simple arithmetic as

Algorithm 2 Iterative algorithm for translating a Cartesian location to its LLA representation. Here a is the equatorial radius, f the flattening of the planet, $e^2 = 1-(1-f)^2$ is the square of the first eccentricity, $s = \sqrt{x^2+y^2}$ and $N = \frac{a}{\sqrt{1-e^2\sin^2\mu}}$ [16]

Require: $i_{max} > 0$ ▷ Iterations for converging of geodetic latitude
  $l \leftarrow \arctan\{\frac{y}{x}\}$
  $\beta \leftarrow \arctan\{\frac{z}{(1-f)s}\}$ ▷ Initial guess for reduced latitude
  $\mu \leftarrow \arctan\{\frac{z+\frac{e^2(1-f)}{1-e^2}a\sin^3\beta}{s-e^2 a\cos^3\beta}\}$ ▷ Initial guess for geodetic latitude
  $\beta \leftarrow \arctan\{\frac{(1-f)\sin\mu}{\cos\mu}\}$ ▷ Recalculation of reduced latitude
  $i \leftarrow 0$
  while $i < i_{max}$ **do**
    $\mu \leftarrow \arctan\{\frac{z+\frac{e^2(1-f)}{1-e^2}a\sin^3\beta}{s-e^2 a\cos^3\beta}\}$ ▷ Calculate geodetic latitude until convergence
    $i \leftarrow i+1$
  end while
  $h \leftarrow s\cos\mu + (z+e^2 N\sin\mu)\sin\mu - N$ ▷ Calculate altitude

elaborated in algorithm 3. Several widely-used viewshed algorithms will be discussed in this section, along with their strengths and weaknesses.

Algorithm 3 Viewshed algorithm where $v[i]$ is a Boolean array for visibility of terrain at position i

Require: $d \geqslant 0$ ▷ Distance from origin is positive
Require: $h_0 \geqslant height(0)$ ▷ Viewpoint is located above terrain
  $i \leftarrow 0$
  $\alpha \leftarrow -\infty$ ▷ Arbitrarily low angle for initial comparison
  $v[0] \leftarrow true$
  while $i < d$ **do**
    $h \leftarrow height(i)$ ▷ Sample the height at the active location
    $i \leftarrow i+1$
    $\gamma \leftarrow atan(\frac{h-h_0}{d})$
    if $\gamma > \alpha$ **then**
      $\alpha \leftarrow \gamma$ ▷ Record maximum angle
      $v[i] \leftarrow true$ ▷ Terrain is visible
    else
      $v[i] \leftarrow false$ ▷ Terrain is in shadow
    end if
  end while

2.4.1 R3 & R2

R3 represents the most straightforward method for computing a viewshed; for every grid point, the minimal required angle is computed using algorithm 3. The drawback of this algorithm is its performance and scaling behavior; because every grid point constitutes a sight-line between the observer and said grid point, this algorithm scales with $O(r^3)$. It is, by definition, as accurate as performing an *LoS*-query on the original DEM grid containing the elevation data. During the

process of computing the minimal required angle in *R3*, the line between the observer and grid point crosses several other grid points. For a query between the observer and a far-away grid point, recording the intermediate values for the minimal required angle (α in algorithm 3) to the current grid point on the viewshed omits a large number of additional calculations. In *R2*, only queries to the outer ring of the viewshed are performed in order to process the full area of the viewshed by utilizing this principle [23]. However, in reality, intermediate points will rarely exactly align with the grid when only computing queries on the outer ring. Therefore, when only considering points which are aligned exactly, most of the grid will remain empty and will as a result contain no data. On the contrary, recording intermediate points to their closest corresponding grid point will cause an error; a point close to the origin will potentially block more of the viewshed than in reality. To mediate this issue, a constant ϵ can be defined. If distance d between point p and the closest grid point is smaller than ϵ, the error incurred by recording the angle to the grid point is deemed within an acceptable margin. Furthermore, in order to reduce ϵ, queries to points between points on the edge of the viewshed can be made in order to increase the number of grid points within a distance ϵ from any query. Setting ϵ to a value where no blind spots occur and where the results are as close to the results of *R3* results in optimal usage of the *R2* algorithm. An illustration of the grid alignment process of the R2 algorithm is given in figure 2.6.

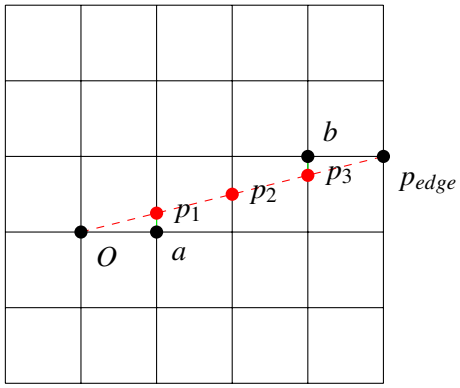


FIGURE 2.6: Illustration of the R2 algorithm where for a query $O \rightarrow p_{edge}$, points $p_{1,2,3}$ are sampled. For a given $\epsilon = 0.25$, p_1 and p_3 would align to grid points a and b respectively. Due to p_2 being $d = 0.5$ away from its neighbouring grid points, the result is computed for the query $O \rightarrow p_{edge}$ as intermediate angle but not stored on the viewshed grid.

Decreasing the value for ϵ improves the accuracy of the data points on the grid; the values recorded resemble values which are closer in proximity. In theory, setting $\epsilon = 0$ corresponds to the result from *R3*. However, lower values for ϵ create 'gaps' on the grid with `null` data (for example, p_2 in figure 2.6 does not map to a grid point). An alternative approach to increasing ϵ is to increase the number of sight-lines to the outer ring. Then, instead of performing 1 query for every point on the edge of the viewshed, N queries are emitted with an offset of $\frac{1}{N}$ grid point each. This increases the chance for a sight-line getting within a distance of ϵ of a grid point and decreases the number of `null` data points.

2.4.2 (ML-)XDraw

XDraw is an algorithm for computing *Line of Sight* for rings emitting from a central point. The algorithm aims to re-use calculations from the previous ring in its calculation of the next ring. There have been effective efforts in improving this algorithm in the form of parallel composition.

Either by utilizing one or more Graphical Processing Units (GPUs) [6] or a cluster of servers [9]. The latter, also called Multi-Level *XDraw* (*ML-XDraw*), is a form of dividing up the rings around the observer such that every section in said ring can be placed in a queue. Due to the reliance of points in the previous ring, this can cause issues, as points near the outside of a section may require points from a different sections' inner ring. Drawing a margin around every section such that every section partially overlaps with its neighbouring section solves this issue, but introduces redundant calculations. If performed optimally, the number of redundant calculations is equal to the number of points located on the boundary of every section.

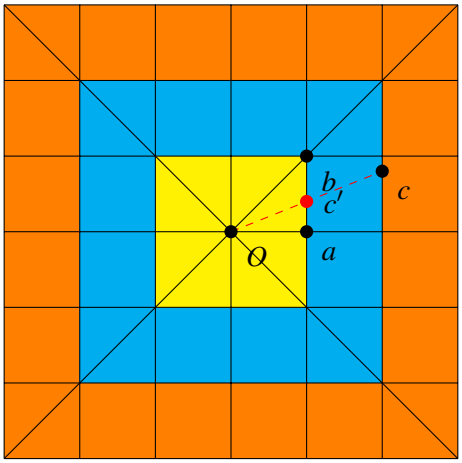


FIGURE 2.7: XDraw algorithm visualization where the rings emanating from the centre all have a different color. Here, the value for point c depends on the values for points a and point b because the projection c' lies between a and b.

Further research has focused on reducing effects of errors arising from parallel processing *XDraw*, using fault-tolerant techniques [10]. In this paper, *XDraw* will be parallelized on the CPU by subdividing the ring into octants, this way the borders between the sections are axis-aligned. When dividing the surface of the viewshed into octants, the results of any section are completely independent of another due to how *XDraw* operates. The algorithm compares the angle between a point in the current ring to an interpolated angle in the previous ring, and computes the maximum. This process of interpolating previously computed values can potentially compound an error as it propagates outward. However, it has the advantage of scaling well due to every grid point being manipulated exactly once and requiring just 3 queries per assignment (2 angles in the previous ring and the elevation of the selected grid point).

2.5 Detection probability

The answer to the question of whether an object is in *Line of Sight* is not always a binary *yes* or *no*. In some scenarios, it is desirable to model the probability of detection instead. In order to model this behavior, the electro-optical and radar sensors require different approaches.

2.5.1 Electro-optical sensor pixel coverage

In the use-case of this paper, an electro-optical sensor can be viewed as a simplified version of a camera sensor; a rectangular grid of pixels where every pixel contains information of part of the view. In a video game, the graphics card models the pixel values of this simulated camera sensor by calculating the 'ownership' of every pixel's color value. Even though this approach would be valid for the purpose of calculating *Line of Sight*, a simplified model will suffice. Determining whether the angle between the observer and target exceeds any intermediate angle between the observer and the terrain will imply whether a sensor with infinitesimally small pixel areas will detect the object. It simply answers the question of whether a geometric line can intersect both the location of the observer and target without intersecting the terrain. However, to mirror the behavior of a camera sensor with limited resolution and range, one can consider the principle of pixel coverage. In this approach, the area of a pixel at a given distance d increases with d^2 as per the inverse square law. Therefore, if a target is visible at distance $d < d_0$ and has an area of A_0, the minimal area to detect the object at $d = 2d_0$ is $4A_0$. If the area is smaller than the defined threshold, a probability can be calculated based on the area of the object and the area of the pixel area at distance d. The area visible from the perspective of the observer can then be determined by choosing a valid sampling rate and firing a set of sight-lines at the target (figure 2.8).

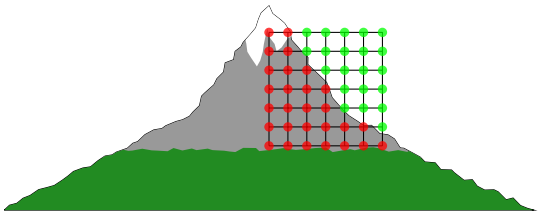


FIGURE 2.8: Illustration of the process of determining the area of a target visible from the perspective of an observer. The ratio between the obfuscated samples and the total of samples gives an indication of the ratio of the area of the object in view.

2.5.2 Radar signal-to-noise ratio

For the use-case of radar, a more complicated model is necessary. From a radar model, given the properties of the receiver and target, an SNR (signal-to-noise ratio) is calculated. The SNR indicates the strength of the signal with respect to background noise. For the purpose of this paper, it is of interest to answer the question of whether a target is visible; either with a *true* or *false* or a detection probability. Neither is a direct translation of SNR. As such, a detection model must be applied to the resulting SNR value. The detection model takes a variety of properties into account, such as the number of sensors on the radar and the observation time. As the detection models contain sensitive information about the internal workings and capabilities of radar systems, it was decided to be outside the scope of this paper. Instead, a step-function (equation 2.8) was chosen as an activation function for the radar detection process. In reality, the behavior of this activation function is more complicated, but for the scope of this paper, it is treated as a black box.

$$LoS = \begin{cases} \text{true} & \text{if } \text{SNR}_{radar} \geqslant \text{SNR}_{treshold} \\ \text{false} & \text{otherwise} \end{cases} \tag{2.8}$$

Chapter 3

Research goals

The following set of research goals have been identified based on available knowledge in literature and through discussions with experts based on the presented use-case. For every collection of research questions, more relevant information and motivation is introduced beforehand.

3.1 Accounting for the curvature of the Earth

The curvature of the Earth poses a challenge for determining *Line of Sight* when working with grid-based elevation data. Due to the data recorded being relative to a 2-dimensional plane, it does not accurately portray the landscape it represents due to the absence of the Earth's curvature on the planar representation. Ideally, the DTED grid could be viewed as a flat Earth, then formula 2.1 could be utilized to determine the angle between the observer and grid points. However, the further away a target is on the grid, the larger the discrepancy between the flat Earth model and the real world. To model terrain and objects disappearing behind the horizon, multiple methods are described which all have their strengths and weaknesses. These methods are compared in terms of performance and accuracy. Comparing different methods of accounting for the curvature of the Earth allows the selection of a valid approach for the purpose of calculating *Line of Sight* based on the required fidelity of the use-case.

3.1.1 Exact approach

A Coordinate Reference System (CRS) describes a set of formulae to translate a coordinate to a location in 3-dimensional space. Instead of computing the angle between an observer and target using formula 2.1, a more exact approach would be to first translate the two coordinate to ECEF and drawing two vectors; one from the observer normal to the surface and one from the observer to the target. The angle between these two vectors can be calculated using formula 3.1.

$$\theta_{observer \rightarrow target} = \arccos\left(\frac{\vec{N} \cdot \vec{V}}{|\vec{N}||\vec{V}|}\right) \tag{3.1}$$

Here $\vec{N}$ is the normal vector at the observer, and $\vec{V}$ is the vector from observer to target.

This approach works regardless of the CRS in use, as it operates in Cartesian space. With this strength, its inherent weakness comes to light; conversion between the coordinate and ECEF representation is required beforehand. This process is computationally expensive, as is illustrated in formula 2.6 and algorithm 2. This method does produce the most accurate results in terms of accounting for the curvature of the Earth, as it does not rely on compensating for curvature on a flat, 2-dimensional surface but rather directly calculates angles in 3-dimensional space.

3.1.2 General formulae for Earth curvature drop-off

Coordinate reference systems consist of 2 constants, representing the radii at the equator and the North and South pole. These two constants model an ellipsoidal shape, which is a better approximation of the Earth's surface than a perfect sphere due to centrifugal forces induced by the Earth's rotation on its shape. This causes the distance between the North and South pole to be less than the distance between two points on the equator located at two opposite sides of the Earth. This introduces difficulties when pre-calculating the drop-off induced by the curvature of the Earth. For example, an observer at the North pole would experience a smoother surface than an observer on the equator. Therefore, when determining a valid approximation for calculating the drop-off caused by the curvature of the Earth, its ellipsoidal shape must be taken into account.

3.1.3 Projecting to a spherical CRS

As part of the preparatory research for this paper, an assessment of assuming a spherical Earth was performed and shown in appendix A. Usage of a perfectly spherical CRS would allow the application of the aforementioned principles and approximation techniques related to viewing the Earth as a perfect sphere and formula A.1. Most widely-available DEM data, however, is recorded as elevation relative to the WGS84 CRS. In order to increase the applicability of the results of this research, it is of interest to use elevation data adhering to this standard. Projecting the elevation data from an ellipsoidal CRS to a spherical CRS is introduced as a pre-computation subroutine. This process is not as straightforward as correcting the elevation data, as the latitudinal location of a coordinate is also subject to change. Given that both reference systems have valid methods to convert between LLA and ECEF, converting from ellipsoidal to a spherical CRS is possible using intermediate conversion to ECEF.

Research questions

The following research questions have been stipulated with respect to this subject:

RQ1. What is the impact of converting from WGS84 to a perfectly spherical CRS in terms of conversion time and accuracy loss?

RQ2. Does the approximation approach provide a valid model of the Earth's curvature, and at what distances is this approximation valid?

3.2 DEM sampling and interpolation techniques

Depending on the smoothness of the terrain, the difference in height at location (x, y) is marginally different from the height data at $(x + \epsilon, y + \epsilon)$ for small enough ϵ. Depending on the resolution of the grid, the required fidelity and the gradient of the terrain elevation, reducing the resolution of the grid may provide comparable results at lower computational costs. By reducing the information to a lower-resolution grid there is a loss of information; in particular, the heights in-between points on the grid. The way this disparity is handled influences the *yes*- or *no*-answer of whether there is in fact *Line of Sight*. There are a number of possible reduction techniques from the high-resolution version to the simplification. Researching the effects of low-resolution sample data on the *LoS* queries allows for quantifying a set of requirements for the DTED resolution.

Naive The initial naive approach is to sample a point, then skip k points relating to the desired resolution and to sample the next point. This is the computationally least expensive approach as there are no additional calculations. However, when the resolution is relatively low this may yield undesired results. If, for example, there is a narrow mountain blocking the view between the observer and target, there is a possibility of sampling data in the valleys next to the mountain and excluding the mountain top itself. This effectively omits the mountain from the data, affecting *LoS*-queries.

Max/min To impede the negative effects introduced by naive sampling, the lowest or highest point in the neighbourhood of a point on the grid can be sampled. This results in either a pessimistic or optimistic answer to the *LoS* question. If there is *Line of Sight* between an observer and a target with a mountain of height h between them, there must also be *Line of Sight* between an observer and a target with a mountain of height $h - \epsilon$ between them (pessimistic approach). Furthermore, if there is no *Line of Sight* with a mountain of height h there must also be no *Line of Sight* with a mountain of height $h + \epsilon$ (optimistic approach).

Average Instead of choosing an optimistic or pessimistic value for the height, the (weighted) average of the surrounding points can be computed. This works well for smooth terrain. For uneven terrain, this will even out peaks and troughs, possibly resulting in a flat section of terrain where there should be a gradient.

Variable A possible approach worth exploring is to vary the grid density. It is possible to decrease the grid resolution for homogeneous terrain, i.e., there is less difference between a point and the points in its direct neighbourhood. An indifference in height data makes differences at smaller distances less important, and therefore these can be neglected. The higher the difference between nearby points, the higher the resolution of a DEM to fully represent its properties.

When sampling a point which does not perfectly align with the grid, the data at this point is undefined. Based on data of neighbouring grid points and the locality of the point to its neighbours, the elevation data can be interpolated. There are a multitude of interpolation methods, of which the most common methods have been selected. These are, in increasing order of complexity:

Aligned If the sampled point is exactly aligned with the grid, no interpolation is necessary and the point can be sampled directly. Any interpolation method would by definition result in the same value, but averting the increased computational complexity introduced by interpolation yields a temporal advantage.

Nearest neighbour The most simple method for sampling an intermediate data point is to sample the nearest neighbouring grid point. The advantage of this method is its computational simplicity and performance. For a relative grid distance d to the nearest grid point, when $d \ll \frac{1}{2}$, this approximation yields the best result. However, in the worst-case for $d = \frac{1}{2}$, this method yields inaccurate results based on the terrain gradient.

Linear Given that the sampled point is aligned with the grid in one dimension (relative distance $d_x = 0 \vee d_y = 0$), 1-dimensional linear interpolation can be applied. In this case, based on relative locality d to its neighbours, a weighted average between the two key values of the neighbours is returned. For this method, sampling two points is required.

Bi-linear Given that neither of the relative distances $d_x = 0 \vee d_y = 0$, the point is located between four grid points. Linear interpolation is applied to the two dimensions by first linearly interpolation one dimension, and subsequently linearly interpolation the other dimension. For this method, sampling four points is required.

Cubic Given that the sampled point is aligned with the grid in one dimension, a 1-dimensional cubic interpolation can be applied. For a point between f_0 and f_1, for relative distance $d > 0$, points $f_{-1}, \ldots, f_2$ are sampled. Cubic interpolation relies on the assumption that the function is continuous, which holds for most terrain (counter-examples include Fjords, Grand Canyon or other outliers). Based on locality d, and the surrounding sampled points, a function is fitted for which the predicates (the sampled elevations at $f_{-1}, \ldots, f_2$) hold. For this method, sampling four points is required. This is identical to the bi-linear interpolation technique. However, cubic interpolation requires more computation steps and is therefore considered more complex.

Bi-cubic Given that neither of the relative distances is zero, the point is located between four grid points. Bi-cubic interpolation is applied to the two dimensions by first applying cubic interpolation in one dimension, and subsequently applying cubic interpolation to the other dimension. For this method, sampling sixteen points is required. It is the most computationally expensive method, but considered the most accurate as it accounts for the gradient of terrain of a larger neighbourhood.

Researching the effects of different interpolation techniques on the performance and accuracy of *LoS* queries allows the selection of a valid approach based on the available resources and required fidelity. In particular, it is of interest how significant the difference is between bi-linear and bi-cubic interpolation in terms of performance and accuracy.

Research questions

The following research questions have been stipulated with respect to this subject:

RQ3. What is the impact of different DTED sampling techniques on performance and accuracy?

RQ4. What is the impact of different DTED interpolation techniques on performance and accuracy?

3.3 Pre-computing viewshed for (near) static observers

Utilizing viewshed algorithms like *R2* and *XDraw*, intermediate results between the observer and outer ring are cached and re-used when applicable. This allows for a more efficient method of computing the minimal inclination angle for a given area. For a static observer, like a stationary radar system, this would allow one computation to fulfill all future queries. However, in the scope of this paper, all entities (including the observers) are considered to be dynamic. Quantifying under which conditions an observer can be approximated as static in terms of re-using past viewshed computations allows for the *LoS*-service to impose a hard upper limit on the computation time of a viewshed. Furthermore, as an observer may come back to a previously visited location, keeping a select number of previously computed viewsheds may provide a temporal advantage in order to minimize duplicate calculations. To minimize the memory impact of retaining previously-computed viewsheds, finding a valid method and data format to compress these results whilst retaining precision and testing the performance and accuracy decrease versus the positive impact on memory usage is of particular interest.

Research questions

The following research questions have been stipulated with respect to this subject:

RQ5. What are the advantages and disadvantages of different viewshed approaches with respect to performance and accuracy?

RQ6. What is the number of static observer *LoS* queries at which pre-calculating an entire viewshed is more efficient than individual queries?

RQ7. What is the maximum offset at which an observer can be approximated as static for the purpose of re-using viewshed results?

RQ8. What is the performance and accuracy impact of compressing a viewshed?

3.4 Collision detection

Along with interacting with the DEM and atmosphere, a sight-line can also collide with an object. Collision detection is the act of checking whether the path between two objects is obstructed by another object. This object can be a collection of complex shapes, and intersecting with any of the faces of the shape results in the signal being obstructed, absorbed or scattered (depending on the surface roughness and angle with the surface normal vector).

In this paper, the objects to be tested for intersection are approximated as a set of spheres and triangles because of the low amount of resources required for calculating intersections between a line and the latter two shapes. A pre-processing step can reduce a complex shape with a high number of polygons to a shape consisting of a relatively low number spheres and/or triangles. The resulting simplification should have a maximum disagreement with the original model no more than the allowed margin of error prescribes.

Given the breadth and position of an object relative to an observer, objects that are certain to not intersect with the sight-line can be eliminated. For an object consisting of a set of shapes, a sphere is fitted which encompasses all its faces. If the sight-line does not intersect this sphere, all its faces do not intersect either and the object can be eliminated from the queue of candidates. If the sphere intersects with the sight-line, there can be two subsequent options; a) collision detection is performed for each of the faces to determine whether the sight-line is obstructed or b) the enveloping sphere is too far away in order to be perceived, and as such it is eliminated from the queue of candidates.

Research questions

The following research question has been stipulated with respect to this subject:

RQ9. What are the performance and accuracy effects of exact mesh-based collision detection versus exclusively calculating collision detection of the enveloping sphere?

3.5 Radar detection model

The question of whether a radar system detects an object does not yield a binary *yes*/*no* answer. Instead, in the simulation of radio waves propagating through space, the SNR (signal-to-noise ratio) is computed for a slice of vertical space. Then, depending on the elevation of the target, the appropriate SNR value can be queried from the result. Whether the object is visible is not directly derivable from the SNR, and a detection model must be applied first. This detection model accounts for the characteristics of the radar system, such as the amount of sensors, processing performance, emitting power and detection time, and is able to compute a probability $P(detect) = f(p_0, p_1, \dots)$ for parameters p_k. However, these detection models are complicated mechanisms by itself, and require a significant amount of knowledge in the field of radar technology and access to sensitive data to construct. Due to the limited scope and real-time requirements in the current use-case, a simplified (though unrealistic) model for detection in the form of an SNR threshold is considered.

Research questions

The following research question has been stipulated with respect to this subject:

RQ10. What is an appropriate SNR threshold to model the unique effects of radar propagation such as refraction and ducting?

Chapter 4

Methodology

In this paper, a set of research goals have been stipulated. For each of the research questions mentioned in chapter 3, a hypothesis and research method are elaborated upon in this chapter. Additionally, the architecture of the encompassing *Line of Sight*-service is illustrated in appendix C. The goal of this service is to provide a framework under which the research can be performed, as well as provide a starting point for Thales to further develop the results into a working product.

RQ1: What is the impact of converting from WGS84 to a perfectly spherical CRS in terms of conversion time and accuracy loss?

Accounting for different Coordinate Reference Systems was adopted early in the design process. A coordinate is always relative to a CRS (represented by a class of the subsequent type), and this class contains all relevant methods. To test the problem statement, a random collection of coordinates are generated and converted into spherical coordinates. This is done through first converting from ellipsoidal WGS84 to ECEF and then from ECEF to spherical. Accuracy of the conversion is tested by verifying the commutation relation; the conversion is bi-directional. Moreover, the resulting converted spherical grid should contain all physical features of the original ellipsoidal grid. This will be verified by converting both grid representations to an image and comparing high and low elevation points. For this purpose, the island of Corsica in the Mediterranean was chosen as a potential candidate as it contains a mountainous terrain close to sea.

To test for performance impact, a simple benchmark is run for various grid sizes where the time spent on converting to the spherical CRS is measured. Then, a set of *LoS*-queries are executed. The average time spent on $n \geqslant 1000$ queries is measured and compared between the two services.

To test for accuracy impact, a set of *Line of Sight* queries will be performed. For the land mass on Corsica, $N \times N$ evenly-spaced coordinates with elevation zero are chosen. The elevation of every point is set to the terrain elevation at the given coordinate on the grid; effectively producing $N \times N$ points on the surface of the Earth with varying elevation. Then, for every coordinate, N points are added to an array with varying elevation above the terrain. FOr example, in the case of a coordinate on the surface of the grid; (l, μ, h), coordinates $\{(l, \mu, h), (l, \mu, h + \Delta h), \ldots, (l, \mu, h + (N-1)\Delta h)\}$ are added to the array. The total number of coordinates in the array is N^3.

Then, all permutations are fed into the *Line of Sight* service for both the spherical and ellipsoidal CRS. Here, coordinate a with index i for $0 \leqslant i < N^3 - 1$ will combine with coordinates $b \in \{i+1, \ldots, N^3\}$. If the results of the *LoS* queries diverge (i.e., *true* for one, *false* for the other), a subroutine is run. This subroutine will express the error of the *LoS*-service in meters by varying the location of the target up or down depending on the results:

- If the ellipsoidal *LoS*-service answers with *true* and the spherical *LoS*-service answers with *false*: move the target **upward**.
- If the ellipsoidal *LoS*-service answers with *false* and the spherical *LoS*-service answers with *true*: move the target **downward**.

Height is varied until the two services agree, and the Δh is recorded as an error statistic. It is of interest to find the number of faulty queries, success percentage and the maximum error along with the mean error and standard deviation.

Hypothesis 1: *Due to the simpler mathematics of determining the angle between the observer and target in the spherical representation, it is expected for the spherical CRS queries to be faster. A rough estimate in the magnitude of the performance increase can range from ten to fifty percent based on the total lines of code of each method. The conversion from the ellipsoidal to the spherical CRS creates a time threshold, this process is estimated to consume several seconds for a regular* 1200×1200 *size grid, based on the assumption that a single ellipsoidal coordinate to spherical coordinate conversion takes less than one millisecond.*

The physical features of the terrain should be visible after converting from ellipsoidal to spherical, the latitudinal position of a coordinate is shifted and the scale is different. However, these differences (if implemented correctly) should remain small, and the aim and expectation is for the error rate to be below one percent.

Measurements and data characteristics

During this experiment, the distance between two locations will be measured in a simulated environment. The first location is an arbitrary point on the grid to which a sight-line will be drawn from the perspective of the observer. The second location being the point where (either by moving upward or downward) the question of whether there is *Line of Sight* agrees with the ground truth. This offset is measured with a precision of 10cm, and an average and standard deviation is calculated in order to make more informed observations about the validity of the method in question. *Line of Sight* is determined using the `lineOfSight()` method of the `ElectroOpticalVacuumSensor` class (figure C.1) and results in binary *true*/*false*. The queries of this experiment and all other experiments unless specified otherwise are performed on a 1200×1200 DEM grid with a resolution of approximately 90m over distances ranging from approximately 500m to 50km.

RQ2: Does the approximation approach provide a valid model of the Earth's curvature, and at what distances is this approximation valid?

To answer this question, first a valid approximation method must be found. To achieve this, both an angle-dependent and distance-dependent function will be defined. These functions express the

difference in elevation caused by the curvature of the Earth in both the (average) radius of the Earth and the angular distance over the surface of the ellipsoid or surface distance respectively. Two possible heuristic approaches are listed below.

$$\Delta h = f(\alpha) = R_{Earth}(1 - \cos\alpha) \tag{4.1}$$

This formula is based on the geometry of the Earth viewed as a perfect sphere, and is coined the 'central angle approach'.

$$\Delta h = f(d) = k\frac{d^2}{R_{Earth}} \tag{4.2}$$

This formula is based on the assumption that the difference in height increases in a quadratic manner with distance, but decreases with the radius of the Earth and is coined the 'geodesic approach'. Then, a similar approach to that of RQ1 is applied to find the statistics of utilizing this method to correct for the curvature of the Earth. In other words, the displacement required for the query of the exact approach and approximated approach is quantified as the error. Furthermore, a flat Earth model where $\Delta h = 0$ is used as a baseline to compare results of either approach.

Hypothesis 2: *Both the geodetic and central angle approach will yield an appropriate model for approximating the curvature of the Earth in the distances studied in this paper, and require fewer computational resources than both the exact vector-based ECEF method and the spherical method.*

Measurements and data characteristics

Similar to RQ1, the distance between the original location and location where the approximations' *LoS* query result and the ground truth agree is measured. As a result of using the same methodology, both methods which provide a different solution to the same problem statement of correcting for the curvature of the Earth can be compared and their advantages quantified.

RQ3: What is the impact of different DTED sampling techniques on performance and accuracy?

To quantify the effects of different sampling techniques, a single DEM tile is subdivided into multiple fragments. These fragments are then reduced in resolution based on the gradients and proximity to the observer. Flat terrain is reduced in complexity the most whilst complex terrain retains its original resolution. The border between the Alps and flatland provide opportune conditions to perform this experiment. Then, a similar approach to that of RQ1 is performed where the displacement required for the *LoS*-queries to agree is quantified.

Hypothesis 3: *The dynamic sampling method will result in the necessity of fewer samples, and as such will be faster. However, due to the fragmentation of the data among multiple tiles instead of the original singular tile, more overhead (induced by the need to find the correct tile corresponding to a coordinate) may cause these performance increases to be spoiled.*

Measurements and data characteristics

In this experiment, a 1200 × 1200 DEM tile with a resolution of approximately 90m is split up in a 1200 × 600 DEM tile with a resolution of 90m and a 1200 × 600 DEM tile with a (reduced) resolution of 180m. The latter DEM grid is reduced by using either of the techniques proposed; maximum, minimum or average. Then, the same methodology as RQ1 is applied in order to compare the results to both the results of RQ1 and RQ2.

RQ4: What is the impact of different DTED interpolation techniques on performance and accuracy?

To test the effects of different interpolation techniques, identical queries are performed for three interpolation methods which do not rely on existing alignment to the grid and therefore subsequently can be performed on any coordinate: nearest-neighbour, bi-linear and bi-cubic interpolation.

To benchmark the performance of a given interpolation method, the duration of a set of 1000 queries is measured. Additionally, to test the accuracy of a given interpolation method, a ground truth is needed. Assuming a regular 1200×1200 grid, its resolution is reduced to a 600×600 grid. The reduction is performed such that for four neighbouring points on the old, high-resolution grid are reduced to a single point $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \Rightarrow \alpha$, effectively removing points in an alternating fashion. Then, the points β, γ and δ are queried according to the respective interpolation technique and compared to the original values of the variable's point.

This effectively compares the interpolation algorithm's ability to reconstruct the original grid after information has been removed, and the difference between the original value on the full resolution grid and the value as a result of the interpolation on the reduced resolution grid is quantified as the error of the interpolation algorithm.

Hypothesis 4: *In the test scenario, nearest-neighbour will likely perform worst as it does not take into account one of its neighbouring values. Furthermore, the more neighbouring points are taken into account for a given interpolation algorithm, the more accurate the estimated elevation value will be. Scaling is estimated to be* $t \propto 2^0 N$, $2^2 N$ *and* $2^4 N$ *for nearest-neighbour, bi-linear and bi-cubic respectively, for* N *being the number of terrain queries.*

Measurements and data characteristics

In this experiment, the original 1200 × 1200 DEM tile with a resolution of approximately 90m is reduced to a 600 × 600 DEM tile with a resolution of approximately 180m. During this process, information is lost as the process of reducing the grid resolution does not account for data in neighbouring cells. For every four points in the original grid, one point remains in the reduced grid. The latter three are reconstructed using the interpolation methods, where the difference between the reconstructed interpolated value and the original value is quantified as the error of the interpolation method in question. This serves as a quantified assessment of an interpolation method's ability to effectively estimate values of terrain in-between grid points.

RQ5: What are the advantages and disadvantages of different viewshed approaches with respect to performance and accuracy?

To test the performance and accuracy of the *R2* and *XDraw* viewshed algorithms, both will be compared to the *R3* viewshed algorithm. In essence, *R3* does not benefit from any optimization techniques such as caching or re-use of calculations. It is comparable to performing all individual queries subsequently and considered to be the most accurate viewshed method.

To compare the performance and accuracy of the viewshed algorithms, a number of interesting locations (such as on peaks and in valleys) will be selected and a viewshed will be computed. The time spent on performing the calculations will be measured and compared for different grid sizes. The resulting grids will be rendered as an image and compared both visually and numerically. The latter entails super-imposing both grids over each other and calculating the difference in angle for every location. This results in an error map corresponding to the difference in angle for every location, enabling making quantitative observations with regards to locations where the difference between either algorithm in the greatest.

Hypothesis 5: *R2 and XDraw will both be viable methods for computing a viewshed efficiently, however the underlying algorithms are inherently different and will cause artifacts to form. XDraw will likely perform better in terms of performance due to its low number of required calculations, but will likely suffer from inaccuracies caused by interpolating angles.*

Measurements and data characteristics

Along with measuring the performance of the viewshed algorithms by recording the Unix system time stamp before and after computing the viewshed for an increasing viewshed size, the accuracy of the *R2* and *XDraw* algorithm is compared. This is achieved by exporting the viewsheds as text files in comma-separated values (CSV) format, importing the text files into MATLAB as 2-dimensional matrices and subtracting any given matrix by the ground truth. Here, the ground truth is represented by a viewshed created using the resource intensive but accurate *R3* algorithm. Subtracting one matrix by another is an element-wise operation which results in a value for the error for every location on the grid; from this, a mean error and standard deviation can be calculated. The resulting matrix will be visualized in order to examine locations with the greatest errors.

RQ6: What is the number of static observer *LoS* queries at which pre-calculating an entire viewshed is more efficient than individual queries?

The total time required to execute N *LoS*-queries, along with the required time to construct a viewshed will be measured. Depending on the size of the viewshed, a different amount of queries is required in order to 'break-even' in terms of computing time. Plotting the required queries as a function of grid size is therefore of interest in order to determine the viability of viewshed pre-computations on a live system to determine when it is wise to compute a viewshed in advance. First, an appropriate value for N must be determined, this variable is elaborated upon in section 2.4.1. This is achieved by iteratively decreasing ϵ starting at $\epsilon = \frac{1}{2}$ and increasing N starting

at 1 until a balance between computational time and accuracy has been achieved which satisfies the following set of criteria: *a)* the viewshed must have a mean error not exceeding 3° compared to ground truth *R3*, *b)* the *LoS*-queries on a pre-defined number of points must return the same answer as performing individual *LoS*-queries (verified by a unit test).

Hypothesis 6: *Depending on the algorithm and grid size, it may take several hundreds of milliseconds to compute a viewshed. Based on the assumption that a single query takes less than one millisecond, break-even will likely occur around* 5000 *queries for a standard* 1200×1200 *grid. Likely,* $N \geqslant 2$ *is required in order to achieve accurate results for* $\epsilon \leqslant \frac{1}{4}$.

Measurements and data characteristics

Similar to RQ5, the Unix timestamp before and after viewshed creation is measured. This time interval (as a function of grid size) is compared to the duration of individual queries. Due to individual queries requiring sub-millisecond time intervals to compute, the duration over several thousands is measured and the average is calculated. Then, from these measurements, a ratio can be calculated which indicates a viewshed algorithm's efficiency over performing individual queries.

RQ7: What is the maximum offset at which an observer can be approximated as static for the purpose of re-using viewshed results?

The goal of this research question is to quantify the decrease in reliability by using a viewshed originally meant for another location. Therefore, a number of grid points adjacent to each other are selected and a viewshed is computed for each location for all algorithms. To quantify the effects of using a previously-computed viewshed, the difference in minimal required angle will be subtracted between a viewshed at position (x, y) and position $(x + \epsilon, y + \epsilon)$ for every location on the viewshed, similar to RQ5. The mean error will be plotted as a function of an increasing ϵ, indicating the error as a function of displacement. Here, a displacement of $\epsilon = 0$ will always result in an error of 0 as the query location and viewshed location coincide. However, as the displacement ϵ increases, any query made on the original grid may yield very different results to a viewshed computed for the new location. In order to provide a tangible result, a worst-case scenario is considered where the elevation for the grid points varies greatly.

Hypothesis 7: *The distance* ϵ *for which the error is small enough to be considered accurate depends on the position of the observer and the gradient of the terrain. For terrain with a high gradient, a single positional shift on the grid can cause a high error. For a relatively flat terrain,* ϵ *can be multiple grid points and not make a difference.*

Measurements and data characteristics

Similar to RQ5, the viewsheds are compared in MATLAB using element-wise subtraction. Instead of comparing results between different algorithms, the two viewsheds being compared are the viewshed at zero displacement against viewsheds with the observer located on neighbouring grid points where the distance to the surface of the DEM is kept constant. The mean error over the absolute values on the resulting matrix is plotted as a function of the displacement.

RQ8: What is the performance and accuracy impact of compressing a viewshed?

To quantify the impact on performance and accuracy, identical queries will be performed on an uncompressed and compressed viewshed. The difference in latency will be measured for 10000 queries, and the value difference is recorded as the error. The mean error and standard deviation for a given viewshed is computed and from this, a conclusion is drawn to determine the effectiveness of compressing a viewshed in order to save resource usage.

In this paper, a viewshed is compressed by decreasing the precision of the variable storing the relevant information and defining a linear function by which a value close to the original value can be determined. Instead of a viewshed consuming 8 bytes for every grid point as a `double[]`, the grid is encoded as a 1 byte `byte[]`. The function $v = ax + b$ is used to encode the values. Here, v is the decoded value, in accordance with the original angle stored in the viewshed. a is a scaling variable defined upon creation of the compressed viewshed such that the complete range $\alpha \in [0, \pi]$ can be expressed within the precious of a single byte, b is the minimal value in the original viewshed and x is the compressed value. Furthermore, 16-bit `short[]` and 32-bit `float[]` are considered as alternatives to the relatively imprecise 8-bit `byte[]`.

E.g., for a viewshed with minimal value 10, and a maximal value 20, $b = 10$, and a is chosen such that $10 = a \times \texttt{00000000} + 10$ and $20 = a \times \texttt{11111111} + 10$. Variables a and b are shared across the entire compressed viewshed.

In order to compare the viability of using variables with varying precision, the original 64-bit `double[]` values are first converted to the reduced precision variant. Then, every grid points' value is compared; the difference between the full precision and reduced precision value is quantified. It is of interest to find the lowest-precision data type where the difference yields a negligible difference in accuracy and a sizable resource usage advantage.

Hypothesis 8: *Using this compression technique, sampling will take longer. However, it will still outperform a single query. Accuracy will decrease due to the less-precise usage of a* `byte[]` *instead of a* `double[]`*, but due to the small range of angles ($\alpha \in [0, \pi]$), the precision of a* `double[]` *does provide a noticeable difference.*

Measurements and data characteristics

Similar to RQ5, the difference between values stored on a viewshed are compared to each other using element-wise subtraction in MATLAB. In this case, the difference between angles stored on the original grid (64-bit `double[]`) are compared to data format with less precision. The absolute value of this difference is recorded for all grid points and the average and standard deviation is computed. This process is used for all three viewshed algorithms (*R3*, *R2* and *XDraw*), for every data format. The resulting mean error is recorded in radians, and indicates how reliable the data format is in general. Even a small angle can cause a large height difference over a long distance. As such, the possible height difference induced by the angular error is quantified. Based on this value, the data format is either determined to be valid or invalid for the intended use-case.

RQ9: What are the performance and accuracy effects of exact mesh-based collision detection versus exclusively calculating collision detection of the enveloping sphere?

To determine the effects of accounting for exact mesh collision detection and computing for collisions using an enveloping sphere, a simplified 3d model of an airplane consisting of a cuboid acting as the plane body and two triangles acting as its wings is chosen as a case study. This model is then placed behind a mountain such that it is not visible form the perspective of the observer. It then performs a 'peek' manoeuvre, and the electro-optical sensor detection probability is recorded for three models; the enveloping sphere, a bounding box and the exact mesh model of the simplified plane.

This sensor calculates the probability of detection from the perspective of the observer for the entity as a function of displacement d. At $d = 0$, $P(detect) = 0$. As the entity moves upward ($d > 0$), at some point the sensor will detect the entity. Depending on the collision detection method, the time of detection and corresponding probability of detection will vary. It is of interest to quantify this difference in detection behavior in a scenario where these methods disagree. As such, the entity is placed at a distance of 16km from the observer where $P(detect) < 1$ for partial visibility.

Hypothesis 9: *Even though the difference in volume of a rectangular airplane and its enveloping sphere are significantly different, approximating the model as a sphere enveloping its vertices does not significantly influence the detection probability. However, due to its elongated design, the 'peek' manoeuvre does create a worst-case scenario where the negative space of the sphere is detected first. The bounding box model has less negative space and will therefore likely exhibit behavior more true to the original, mesh-based model.*

Measurements and data characteristics

In this experiment, the `ElectroOpticalVacuumSensor` class' `signalToNoiseRatio()` method is queried for an entity at different elevations. In this case, the query result translates to a value from 0 to 1, representing the probability of detection. The shape of the object determines the number of sight-lines which intersect with the entity, and this influences the elevation at which the entity will be detected and the overall probability of detection. Using either a bounding box or enveloping sphere as an approximation of the entity's shape different behavior will arise, resulting in different values for the probability of detection.

RQ10: What is an appropriate SNR threshold to model the unique effects of radar propagation such as refraction and ducting?

To determine a valid SNR threshold, a radar propagation simulation will be run for which the Earth contains no features other than the curvature beyond the horizon. In this simulation, the radar should not see beyond the horizon. For the second simulation, an elevated duct is introduced, for this simulation the radar should be able to see beyond the horizon. To further narrow the range of a possible value for $\text{SNR}_{threshold}$, a mountain will be introduced and knife-edging (gentle curvature

around the peak) should occur. This yields a possible range of values for the SNR threshold for the placeholder detection model of equation 2.8.

First, a simulation is run for all scenarios where the raw SNR values resulting from the Parabolic Equation model is visualized. Then, for every SNR threshold, the signal is reduced to a binary *true*/*false* by mapping *true* to values equal or above to the threshold (indicated by the color white) and *false* to values below the threshold (indicated by the color black). Using a definition of required system behavior, a window of likely candidate SNR thresholds is determined for the simplified detection model.

Hypothesis 10: *It is possible to chose an* $SNR_{threshold}$ *such that complex behavior of a radar can be mirrored without implementing a complicated detection model. Likely, the SNR threshold should be* $SNR \ll 1$ *for radar characteristic behavior to become apparent.*

Measurements and data characteristics

Using the Parabolic Equation propagation model, the `RadarPESensor` class propagates the field forward from the observer towards to target location using the `propagate()` method (figure C.1). The initial conditions are defined by a vertical slice of values for the electromagnetic field for varying altitudes, initiated by the `initialField()` method. The results are visualized by calling the `toBufferedImage()` method, which produces an image containing the complex magnitudes ($a + bi \rightarrow \sqrt{a^2 + b^2}$) of the magnetic field for all locations between the observer and target between a minimum and maximum altitude. Furthermore, by dividing the complex magnitude of the electro-magnetic field by the free-space path loss (FSPL), the SNR can be approximated. In order to quantitatively determine a valid SNR range, several cut-off values will be chosen and the aforementioned figure will be plotted where instead of a gray value representing the SNR, binary black or white value will indicate the value being below or above this threshold.

Chapter 5

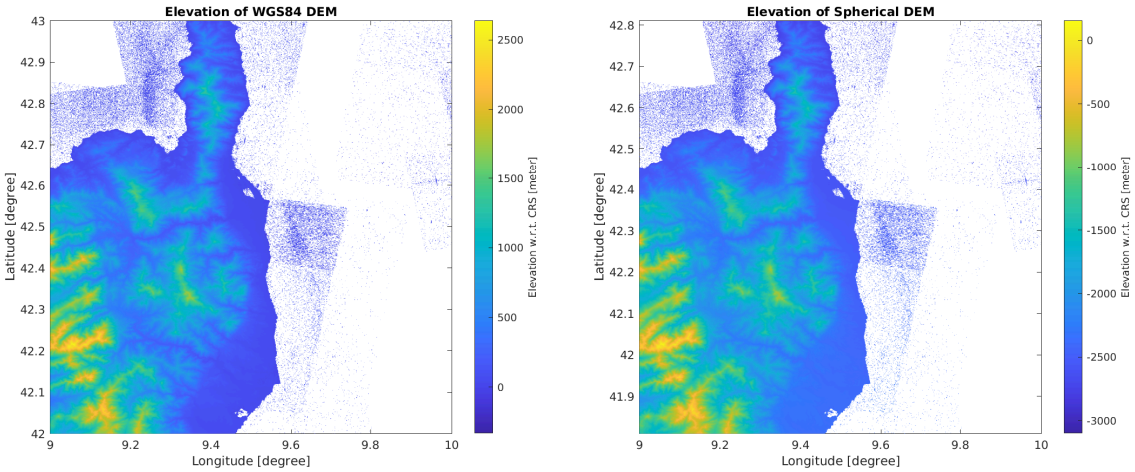
Results

5.1 RQ1: What is the impact of converting from WGS84 to a perfectly spherical CRS in terms of conversion time and accuracy loss?

Correctness of the projection method

To verify the correct translation of the WGS84 CRS to a spherical representation, a DEM tile of the Mediterranean island of Corsica was used. Due to the island's distance to the equator and the ellipsoidal shape of the Earth, the elevation with respect to a spherical CRS will generally be less than the elevation with respect to WGS84. In figure 5.1, a *before* and *after* view of the projection is illustrated. On the right-hand side of the respective figures, a color map legend is shown by which the elevation with respect to the CRS can be read. The white background indicates a lack of data at the given location (no data recorded; `null`). The blue dots surrounded by white are measurement errors of the satellite which mistook sea for land. Here, it is clearly visible that the elevation with respect to the spherical CRS is lower than it was relative to the WGS84 CRS. Noteworthy is the shift in latitude of all points. This is due to points on the WGS84 CRS not pointing directly to the centre of the Earth, instead the data is tangential to the surface. Upon projecting the DEM, a vector to the centre of the Earth is drawn, shifting the latitudinal position slightly. On a sphere, this does not occur as a tangent on its surface will always point to the centre of the sphere.

Visual inspection of figure 5.1b allows us to verify the physical features are retained in the spherical representation. Early implementations of the projection algorithm contained significantly more `null` data points after converting between ellipsoidal to spherical representation. In the projection process, the total area the DEM data spans is mutated and needs to be accounted for. The coverage vector, a vector spanning from the north-western coordinate to the south-eastern coordinate, requires adjustment as it is slightly different for the spherical CRS. Not accounting for this difference instigates a grid shift of several meters, effectively causing an incorrect mapping of grid points to coordinates.



(A) DEM tile of the island of Corsica relative to the WGS84 CRS.

(B) DEM tile of the island of Corsica after projecting to a spherical CRS.

FIGURE 5.1: The effect on coordinate position after projecting a DEM from WGS84 (a) to a spherical CRS (b). As a result of translation to a different CRS, the latitude and altitude of all coordinates are changed.

Sampling locations

The sampling locations for testing the *LoS*-services connected to either the WGS84 or spherical CRS were chosen to be on land as the source DEM tiles did not contain information on mean sea level. For this test described in the methodology, the value of N was chosen as 16. This yields a grid of 16×16 coordinates. Every coordinate has $N = 16$ different values for elevation, starting from 2m and incremented N times by 10m. The sampling locations are illustrated in figure 5.2.

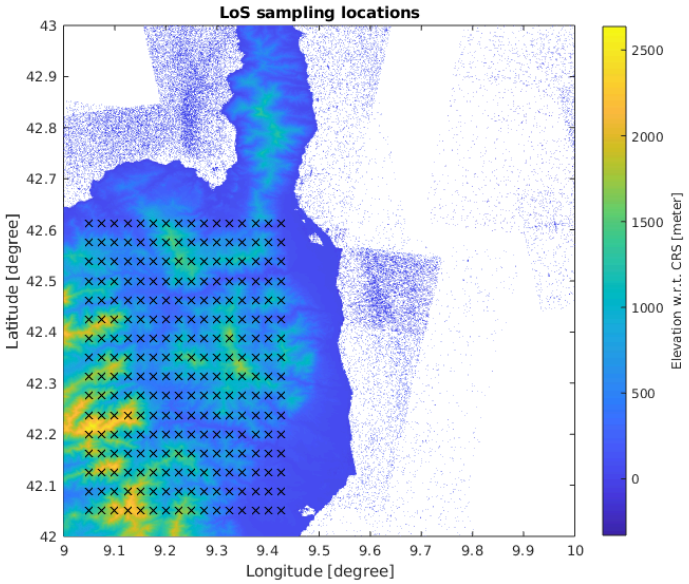


FIGURE 5.2: Sampling locations for the correctness test of the spherical CRS.

Correctness of *LoS*-queries

Performing the correctness test as described in the methodology yields interesting results. Even though the underlying method for determining geometric *LoS* is inherently different, the results overwhelmingly agree. Of all permutations of the 16^3 points ($\sum_{i=1}^{N^3-1} N^3 - i = \frac{1}{2}N^3(N^3-1)$ for $N = 16$ gives 8386560 queries), 213 queries are incorrect. Incorrect in this context implies the disagreement between the WGS84 and spherical CRS representation. The error E of most incorrect queries is $E < 5$ m. However, there are a number of outliers with $E \geqslant 300$ m. All queries for which $E \geqslant 50$ m are displayed in table 5.2. All incorrect queries for which $E \geqslant 5$m are traceable to four observer locations, each with between 1 and 4 target locations when not accounting for altitude. These locations are illustrated in figure 5.3.

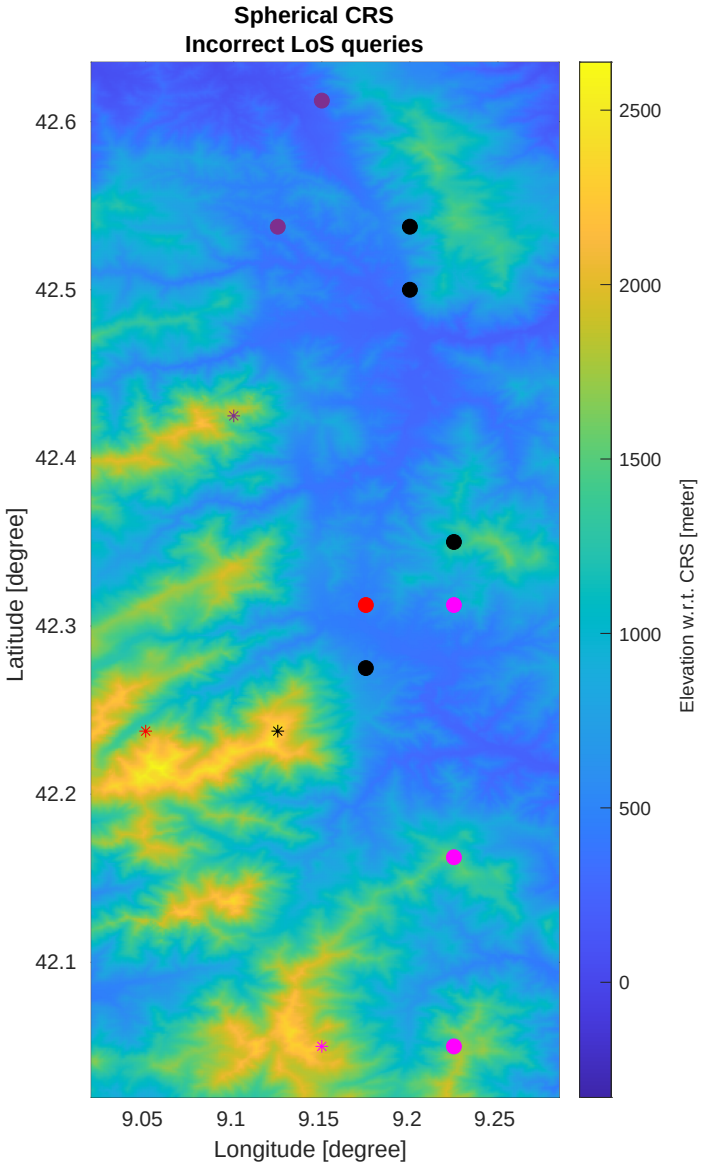


FIGURE 5.3: Locations of the incorrect queries; i.e., where the WGS84 and Sperical CRS *LoS*-services disagree. Locations indicated by an asterisks ($*$) are observers, locations indicated with a circle (●) with matching color are its targets for which the Spherical *LoS*-service response was incorrect.

From figure 5.3, an interesting observation is the location of the observers; these are all located on the edge of a point of high elevation and a valley. This causes interpolated values of intermediate samples to vary greatly with small deviations in elevation. A small error in angle close to the observer can cause a large error in meters further away due to how angles propagate. The coordinate with the most errors (9.15°, 42.05°) is represented by the magenta asterisk and the magenta circles are the coordinate position of its incorrect observers. The positive sign of the error in table 5.2 indicates that the target was moved upward in order to become visible for the spherical CRS-based *LoS*-services, thereby indicating that the spherical CRS obscures more of the *LoS* and gives a more pessimistic view of the question of whether *LoS* is possible. Table 5.1 shows the statistics resulting from the test.

Spherical CRS approximation statistics			
Property	# queries	Mean μ [m]	Standard deviation σ [m]
Error over all queries	8386560	0.001304	0.672515
Error over incorrect queries	231/8386560	47.320346	119.204413

TABLE 5.1: Statistics of the spherical CRS approximation technique.

For the WGS84-based *LoS*-service, 587307 of the 8386560 queries return *true*, and the remaining 7799253 return *false*. The *true*/*false* ratio is 7 percent. Of all queries executed on the Sperical CRS-based *LoS*-service, 8385396 of the 8385627 return the same response, 213 requests are incorrect. The mean error of all queries is 1.3×10^{-3}m, and the standard deviation is 6.7×10^{-1}m. When only considering the 213 incorrect queries, the mean error is 47.3m and the standard deviation is 119.2m. This implies that, if a query is incorrect, its error may deviate greatly.

Latency induced by projecting a WGS84 DEM tile to a spherical CRS

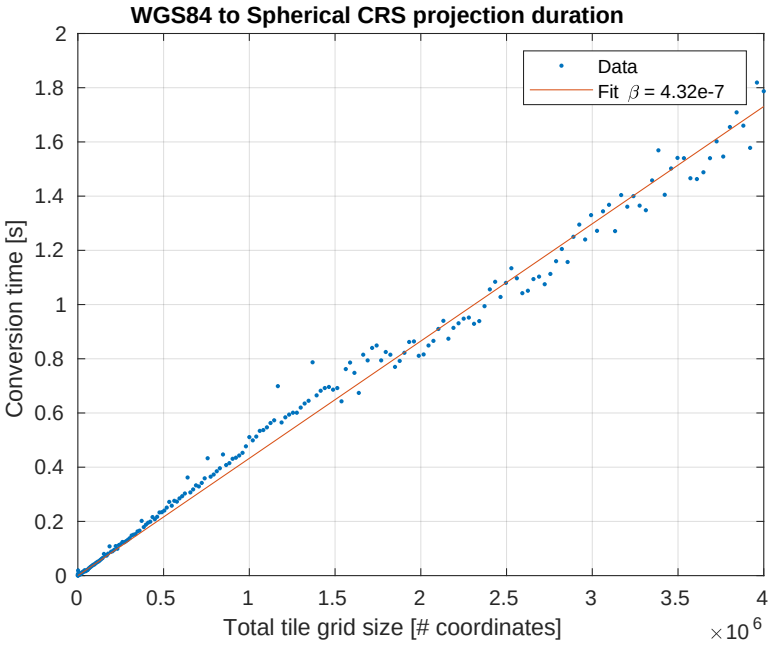


FIGURE 5.4: The time consumed for converting a tile from WGS84 to a spherical CRS.

Incorrect queries where error $E \geqslant 50$m (spherical CRS)		
Observer coordinate	Target coordinate	Error [m]
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1552.40m)	381.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1562.40m)	371.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1572.40m)	361.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1582.40m)	351.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1592.40m)	341.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1602.40m)	331.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1612.40m)	321.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1622.40m)	311.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1632.40m)	301.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1642.40m)	291.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1652.40m)	281.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1662.40m)	271.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1672.40m)	261.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1682.40m)	251.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1692.40m)	241.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.0500º, 1702.40m)	231.60
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1533.16m)	454.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1543.16m)	444.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1553.16m)	434.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1563.16m)	424.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1573.16m)	414.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1583.16m)	404.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1593.16m)	394.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1603.16m)	384.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1613.16m)	374.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1623.16m)	364.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1633.16m)	354.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1643.16m)	344.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1653.16m)	334.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1663.16m)	324.80
(9.1500º, 42.0500º, 1930.88m)	(9.2250º, 42.1625º, 1673.16m)	314.80

TABLE 5.2: Table containing information about the incorrect queries, i.e., queries where consulting the spherical CRS instead of the Ellipsoidal WGS84 CRS resulting in a different response with an error $E \geqslant 50$ m.

Projecting from the WGS84 Coordinate Reference System to the spherical CRS is necessary before queries can be performed. This conversion consumes time, and its scaling is studied in this section. Because every coordinate on the grid is sampled independently, and in a linear fashion, scaling occurs with $O(n)$ where n is the number of grid points. For a 4-core 4GHz CPU, the following graph has been produced where the conversion time is plotted as a function of grid size. Grid size is defined as the total number of grid points, i.e., n for a grid of dimensions $\sqrt{n} \times \sqrt{n}$. From this graph can be read that a graph containing 2.25 million points (e.g., a 1500×1500 grid) takes approximately one second to convert to a spherical CRS.

Performance improvements of the spherical CRS over the ellipsoidal CRS

To benchmark the performance of either method, several millions of requests were performed on a shuffled set of location permutations. In figure 5.6, the duration as a function of the number of performed queries is displayed. The slope of the linear regression model of the data points correspond to the mean time for a single *LoS*-query. For the WGS84 *LoS*-service this is approximately `0.1`ms, and for the spherical CRS *LoS*-service this is approximately `0.07`ms. Therefore, overall, the spherical CRS-based service is approximately 30 percent faster than the WGS84-based service.

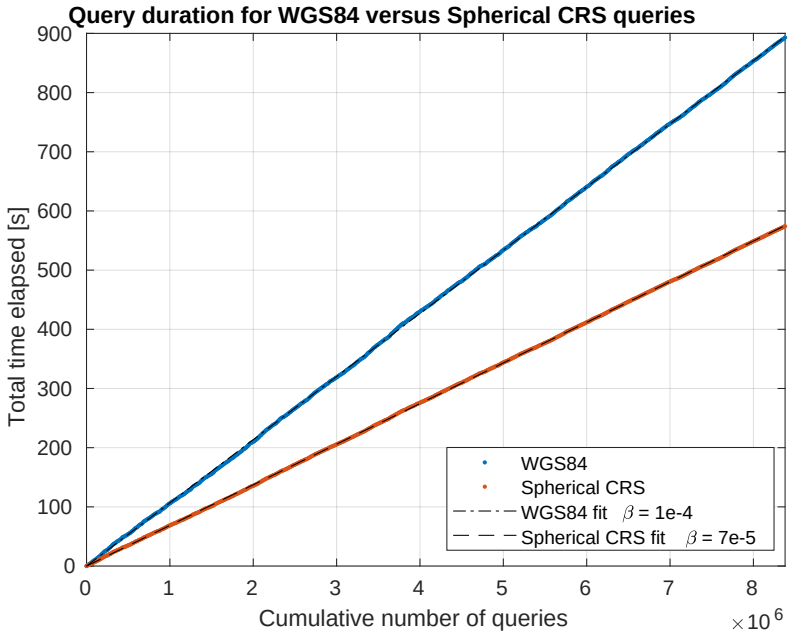


FIGURE 5.5: Query duration for the WGS84-based *LoS*-service and the spherical CRS-based *LoS*-service. Both exhibit linear scaling, but the slope of the spherical CRS-based *LoS*-service is less steep. As such, it is a faster method overall.

5.2 RQ2: Does the approximation approach provide a valid model of the Earth's curvature, and at what distances is this approximation valid?

In order to answer this research question, three approaches to approximating the curvature of the Earth are studied.

Flat Earth Negating the curvature of the Earth, $\Delta h = 0$

Geodesic distance A distance-dependent function of the Earth's curvature, $\Delta h = f(d) = k\frac{d^2}{R_{Earth}}$

Central angle An angle-dependent function of the Earth's curvature: $\Delta h = f(\alpha) = R_{Earth}(1 - \cos\alpha)$

Flat Earth approximation

This method serves as a baseline for the Geodesic distance and Central angle methods. If the curvature is approximated as zero (i.e., assuming a flat Earth), more queries are incorrect due to objects not disappearing behind a horizon. For a small surface distance, this does not influence the result of the *LoS*-queries, however, for longer distances this effect cannot be negated as will be shown in the statistical analysis below.

Flat Earth approximation statistics			
Property	# queries	Mean μ [m]	Standard deviation σ [m]
Error over all queries	8386560	0.022246	0.972134
Error over incorrect queries	13319/8386560	14.007583	19.662518

TABLE 5.3: Statistics of the flat Earth approximation technique.

From this, it may be concluded that assuming a flat Earth is not a reliable method, for there is a clear increase in incorrect queries to a total of 13319 for a similar scenario to RQ1. Because this method requires no projection to another reference system however, the errors are less extreme. The maximum error observed was 155.80m for query (9.1750°, 42.1625°, 862.94m) to (9.1750°, 42.6125°, 943.39m). This is due to the projection enabling the possibility of a grid shift of the DEM data, which in turn can cause a significant error due to coordinates being incorrectly aligned. The grid nature of a DEM coupled with the location shifting of the projection method makes this an inherent problem to the spherical CRS. This was observed in the results of RQ1. Therefore, of the incorrect queries, the mean error was 14m and the standard deviation was 19.67m.

Geodesic distance approximation

Approximating the curvature of the Earth using formula $\Delta h = f(d) = \frac{d^2}{2R_{Earth}}$, i.e., formula 4.2 for $k = \frac{1}{2}$ results in a promising model. For this approximation technique, the number of incorrect queries is 36. For these 36 incorrect queries, the mean error is 11.3cm and the standard deviation is 3.5cm. Additionally, the mean distance between the observer and target for the incorrect queries is 20.9km, with a standard deviation of 12.5km. For targets close to the observer, the value of the

formula 4.2 is low and nearly equal to that of the flat Earth model; which is a valid approximation at short ranges. For larger ranges, the error increases, explaining the relatively large mean distance between observer and target. Table 5.4 shows the results of the test.

Geodesic distance approximation statistics			
Property	# queries	Mean μ [m]	Standard deviation σ [m]
Error over all queries	8386560	0.000000	0.000247
Error over incorrect queries	36/8386560	0.113889	0.035074
Distance to incorrect queries	36/8386560	20869.994990	12509.092216

TABLE 5.4: Statistics of the Geodesic distance approximation technique.

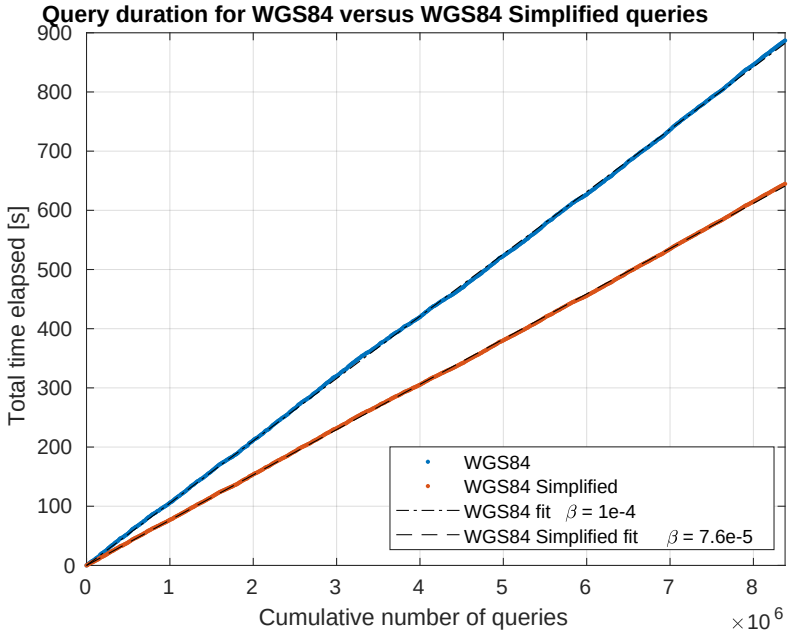


FIGURE 5.6: Query duration for the WGS84-based *LoS*-service and the Simplified WGS84-based *LoS*-service. Both exhibit linear scaling, but the slope of the Simplified WGS84-based *LoS*-service is less steep. As such, it is a faster method overall.

Central angle approximation

The Central angle approximation technique produced identical results to the Geodesic distance technique. This is because the difference between formula 4.1 and 4.2 with $k = \frac{1}{2}$ is negligible for the distances studied on the island of Corsica. As such, the curvature correction is near identical. Furthermore, given the $\Delta h = 10$cm (the incrementing step in determining the error), the difference was not large enough to differ from the Geodesic distance approximation. In figure 5.7, the similarity between the two methods is shown. These approximations are nearly equal, and differ by approximately 25cm over 200km. As such, it is shown that these two methods are both viable approximations and can be used interchangeably. In terms of performance, both methods require a comparable number of calculations and no preference between the two methods has been determined.

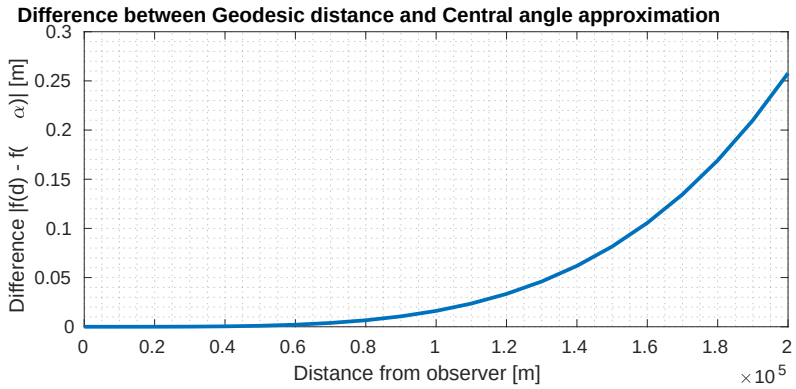


FIGURE 5.7: Difference between the Earth curvature approximation techniques; formula 4.1 and 4.2 for $k = \frac{1}{2}$.

5.3 RQ3: What is the impact of different DTED sampling techniques on performance and accuracy?

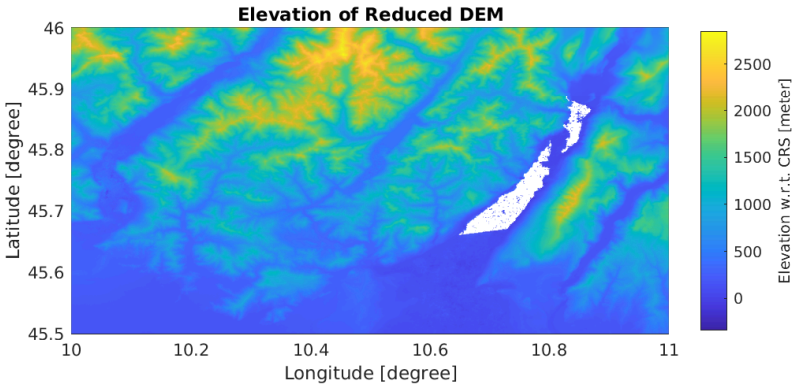
To test the effectiveness of dynamic sampling, a DEM tile was selected which could potentially benefit from fragmentation. In northern Italy, near the Alps, terrain with a high gradient borders terrain with a relatively low gradient. A DEM tile near the Alps where this border is approximately located in the centre of a horizontal divide was chosen. The tile was fragmented into two sections by separating the northern (mountainous) and southern (flat) section and saving the DTED to separate DEM tiles (figure 5.8). The goal of this research question is to test whether the *LoS*-service can benefit from reducing the resolution of the flat terrain tile, whilst retaining detail of the mountainous terrain. Multiple methods for reducing the resolution of the flat terrain tile are tested.

$$\text{Where, for every four points } \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} : \begin{cases} p = \frac{1}{4}(\alpha+\beta+\gamma+\delta) & \text{(average)} \\ p = max(\alpha,\beta,\gamma,\delta) & \text{(maximum)} \\ p = min(\alpha,\beta,\gamma,\delta) & \text{(minimum)} \end{cases} \tag{5.1}$$

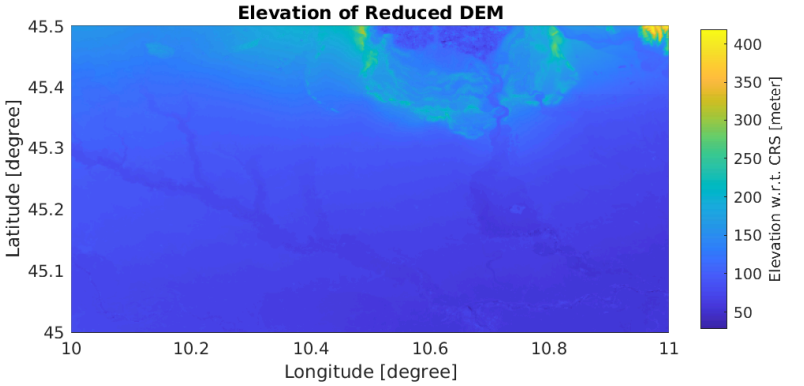
A similar statistical method to RQ1 is executed, with sampled points evenly distributed over the DEM tiles. In this test, the reduced tile results are compared to a reference tile with the original high resolution grid acting as the ground truth.

DTED grid resolution reduction statistics for 130804 **queries**								
	Correct			Incorrect			Distance	
Method	n [%]	μ [m]	σ [m]	n	μ [m]	σ [m]	μ [km]	σ [km]
Average	98.58%	0.167607	3.734526	1861	11.780602	29.050092	31.0	16.6
Maximum	99.01%	0.247078	6.539534	1292	25.012229	60.931554	33.7	18.1
Minimum	97.63%	0.31166	4.209165	3101	13.147436	24.058438	31.8	16.9

TABLE 5.5: Table containing results on the effects of reducing the resolution of a flat section of terrain and retaining detail on high-gradient terrain on accuracy.



(A) DEM of the Alps in northern Italy, because this section has high elevation gradients its resolution is retained. Every grid point spans an area of approximately 90 × 90m.



(B) DEM of the flatland south of the Alps in northern Italy, because this section is relatively flat its resolution is reduced. Every grid point spans an area of approximately 180 × 180m.

FIGURE 5.8: Elevation of the reduced resolution DEM where the northern section (a) retains its original high resolution grid, and the southern section (b) is simplified by reducing the grid resolution.

Table 5.5 shows that using the maximum method of $p = max(\alpha, \beta, \gamma, \delta)$ yields the most accurate results, whilst the minimum method yields the least accurate results. In the case of taking the maximum of all surrounding grid points, the terrain is perceived to be higher than it is in actuality. Terrain with a greater elevation tends to block more *LoS*-queries. The opposite holds for the case of taking the minimum of four surrounding grid points. However, because most of the queries are *false*, taking the minimum has a greater effect on comparing just the *true* and *false* answers. This is visible in table 5.5, where $n_{incorrect} = 1292$ for the maximum technique, and $n_{incorrect} = 3101$ for the minimum technique. The most accurate approximation (with the lowest mean error) is taking the average of surrounding grid points, the highest mean error over all incorrect queries is achieved when taking the maximum of surrounding grid points ($\mu = 25.01$m, $\sigma = 60.9$m). In this model, the absolute value of the error is studied. However, the maximum method gives a model where there are very few false positives and an increased amount of false negatives. The opposite holds for the minimum technique, where there are very few false negatives and an increased number of false positives. Whichever is preferred depends on the use-case; the maximum technique if false positives are to be averted and vice versa. The average technique has the lowest mean error μ overall, and is optimally utilized when the *true*/*false* responses are more equally distributed.

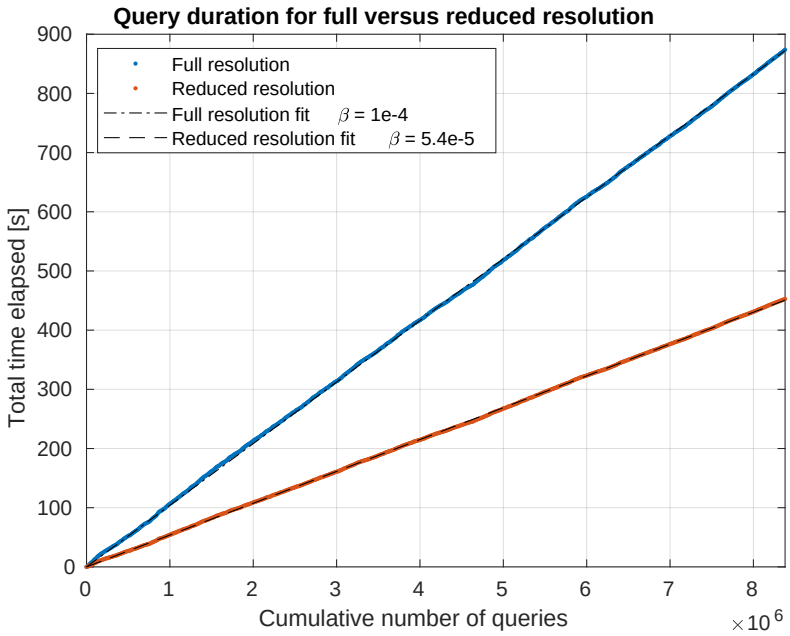


FIGURE 5.9: Query duration for the full-resolution and reduced resolution *LoS*-service.

With regard to performance, the query duration for a reduced resolution grid was compared to a full resolution grid. In figure 5.9, the cumulative query duration is plotted as a function of the number of queries performed. The slope of both lines is the average duration per query, which is significantly shorter for the reduced grid. This is due to the *LoS*-service performing fewer queries on the grid.

5.4 RQ4: What is the impact of different DTED interpolation techniques on performance and accuracy?

In order to test the impact on accuracy and performance for different interpolation techniques, a high-resolution grid is reduced to a lower-resolution grid. When reducing the resolution, instead of taking the maximum, minimum or average, for every four points $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$, the new point p is defined by $p = \alpha$. This way, intermediate points are effectively removed and the interpolation method attempts to reconstruct these data points based on the surrounding grid. The error between the ground truth (original high-resolution grid) and the interpolated value is analyzed for all points on the original grid. An interesting difficulty arose when reducing the resolution of a grid which required careful consideration of an appropriate approach; the DEM had a grid size of 1201×1201 and as such, could not be divided into two parts and/or reduced in resolution without additional steps. Because of this uneven number, the coverage vector had to be changed, and it was chosen for the two tiles to overlap partially after splitting up the data in order to adhere to the conventions of the source data. Additionally, the size of the grid is reduced to the nearest factor of 2 in order to accommodate the resolution reduction schema. Unit tests were written to verify querying a location on either the original or split-up chart returned exactly identical results in order to ensure the experimental results are valid.

Comparison between interpolation methods for 345600 **samples**			
Method	μ [m]	σ [m]	Total t [ms]
Nearest neighbour	16.589997	18.365425	62
Bi-linear	5.322557	6.952590	255
Bi-cubic	4.376876	5.768800	1211

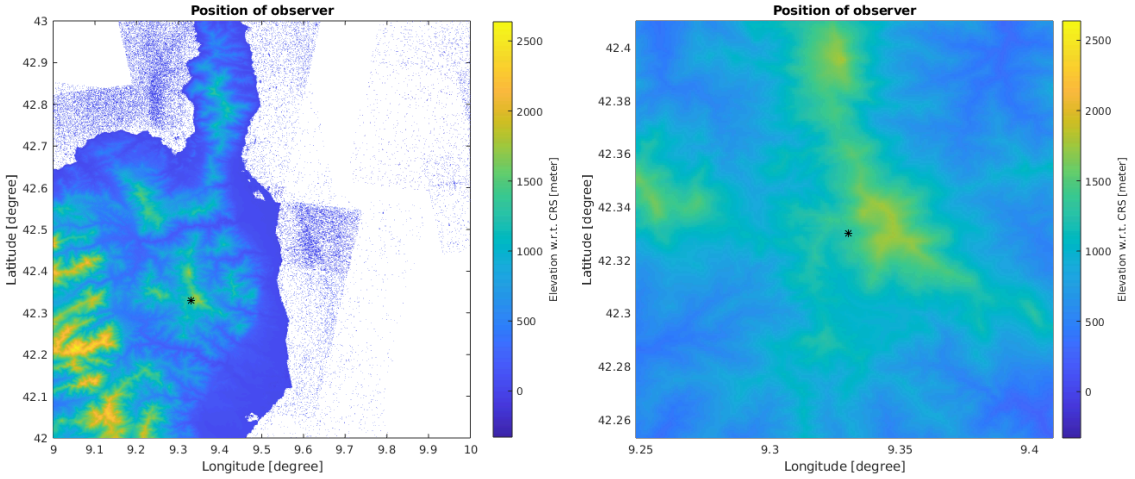
TABLE 5.6: Table containing results on the effects of utilizing different interpolation techniques on query accuracy and performance. The more complex the interpolation method, the more accurate it is and the more resources are required for the computation.

Table 5.6 shows that the more complex the interpolation technique, the smaller the error is between the ground truth and the interpolated value. Transitioning from nearest-neighbour to bi-linear yields the highest increase in accuracy versus additional time spent; a factor 4 time increase results in a factor 3 error decrease. However, when utilizing bi-cubic interpolation over bi-linear interpolation, the computational time is 5 times greater, but the error is decreased by 20%. Bi-cubic interpolation requires sampling 16 points whilst bi-linear interpolation requires 4. Likely, additional overhead created by the underlying matrix multiplication logic used in the bi-cubic interpolation method. For the purpose of this paper, bi-linear interpolation allows for a good balance between the average error and the computational time required.

5.5 RQ5: What are the advantages and disadvantages of different viewshed approaches with respect to performance and accuracy?

In this research question, the viewshed algorithms *R2* and *XDraw* are compared against *R3*. In this case, *R3* serves as an accurate but slow baseline. Both algorithms outperform *R3* in terms of speed by approximating the intermediate *LoS*-calculations. In this process, *R2* and *XDraw* attempt to retain as much of the accuracy as possible. The *LoS* service distinguishes between two different types of viewsheds, each with its own advantages; a radial and tile-based viewshed. The former is defined as a viewshed around an observer for a radius r, covering a total area of πr^2. Determining whether such a viewshed is viable for a given target is simple; test radius r against the geodetic distance between the observer and target. This type of viewshed uses a chart service to query coordinates to receive elevation data, and as such can combine DEM data from multiple tiles. This does, however, introduce additional overhead causing this method to be slower than its counterpart. In the latter method, coined the tile-based viewshed, instead of passing an observer coordinate and a radius, the viewshed is limited exclusively to the tile on which the observer is located. This reduces the overhead of finding the appropriate tile, but limits its uses to the range of the source tile. Both methods are compared in terms of performance and accuracy, due to the slight difference in the implementation. For the radial viewshed construction, lines are drawn as a function of an angle $\theta \in [0, 2\pi)$ whilst the tile-based methods (due to the constant grid spacing) can utilize the underlying grid itself for positioning ($x \in [x_0, x_{max}]$, $y \in [y_0, y_{max}]$).

Figure 5.10 shows the location of the observer, chosen due to the elevation gradient in its vicinity. In the east, a high-elevation mountain blocks the view, this should be visible in the viewshed in the form of high values for the minimal required angle. In the west, a valley dominates the terrain. Here, more of the terrain is visible, resulting in lower minimal required angles.



(A) Observer location relative to the island of Corsica. The terrain in the south-west has high elevation, but is far away, thus having a small impact on the visibility.

(B) Observer location in a more zoomed-in window, showing the terrain gradients in the east and west. The mountain range in the east has a high impact on visibility.

FIGURE 5.10: Observer location chosen for the purpose of comparing viewshed calculations. The location has good visibility in the west, but limited visibility in the east. This allows for a quantitative comparison of different viewshed algorithms.

5.5.1 Performance comparison

Performance comparison between different viewshed methods				
		Computation time [ms]		
Type	Method	WGS84	Spherical	WGS84 Simplified
Tile-based	*R3*	291529	65890	164321
	R2	3913	1355	2404
	XDraw	906	249	488
Radial	*R3*	80341	20571	53220
	R2	5965	2175	2927

TABLE 5.7: Table containing the time to compute a viewshed for the three considered algorithms for a radial and tile-based viewshed. Due to different parameters and grid size, a given radial viewshed method's performance can exclusively be compared to other radial viewshed method's performance and vice versa.

Table 5.7 shows the time to compute a viewshed for different algorithms. A distinction between tile-based[†] and radial[‡] viewshed is made as additional overhead, caused by finding the correct tile for a given coordinate influences the computation time. *R3* is the most accurate because it computes a separate *LoS*-query for every grid point, and does not re-use any results of previous computations. However, it is significantly slower than *R2* and *XDraw*. For further testing purposes, *R3* will be viewed as the ground truth, and *R2* and *XDraw* are compared to its results. In practice, *R3* is too computationally intensive to have a practical advantage for dynamic observers.

[†]In this experiment, the tile-based viewshed has a grid size of 1201×1201 and coverage of $1^\circ \times 1^\circ$.
[‡]In this experiment, the radial viewshed has a grid size of 600×600 and coverage of $0.5^\circ \times 0.5^\circ$.

XDraw only has a tile-based variant implemented, because the algorithm by itself is grid-based and interpolates previously-computed angles. *XDraw* relies on querying either a previously computed angle from its own viewshed grid or data from the parent DEM tile. The radial method constructs a new grid with a custom resolution, causing a misalignment and requiring more overhead to achieve this. In practice implementing *XDraw* for a radial grid is achievable, but the scope was limited to its tile-based counterpart for this paper.

5.5.2 Accuracy comparison

To compare the accuracy of one viewshed method to another, both are compared to the ground truth. In this case, *R3* is, due to the individual *LoS*-queries for every grid point, as accurate as performing a new query. An observer was placed in the position indicated on figure 5.10, and both a tile-based and radial viewshed are computed relative to the WGS84 CRS.

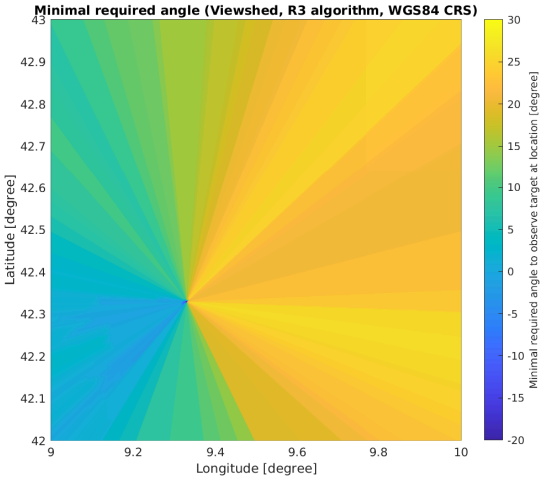
In figure 5.11, the results of all three algorithms are shown for an observer located as described in figure 5.10. In figure 5.11a and 5.11b, a tile-based and radial *R3* viewshed are shown respectively. Here, the gradient between different gridpoints is very smooth, especially in the south-western segment. This smooth transition between angles is due to very small differences in interpolated values for the points between the observer and a grid point. This strength is what causes its main weakness; because every sight-line between the observer and a grid point introduces a new sequence of interpolated locations, it is much more computationally intensive.

Furthermore, a qualitative measurement of the accuracy of *LoS*-queries performed on the viewshed was performed, similar to the method of RQ2. In this experiment, an identical query will be performed on a regular *LoS*-service and a viewshed created using the algorithm in question.

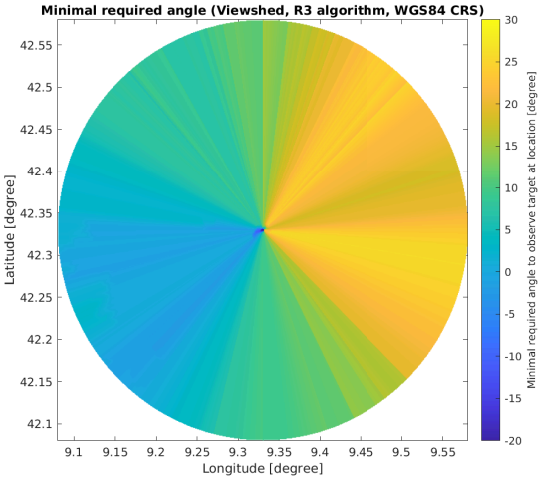
Viewshed accuracy statistics				
Property	Algorithm	# queries	Mean μ [m]	Standard deviation σ [m]
Error over all queries	*R2*	130816	1.901540	23.549441
	XDraw	130816	1.076787	17.419574
Error over incorrect queries	*R2*	1351	178.739156	143.302137
	XDraw	1131	124.026614	140.424946
Distance to incorrect queries	*R2*	1351	11878.132996	6839.987261
	XDraw	1131	16344.380819	7000.141811

TABLE 5.8: Statistics of utilizing a viewshed instead of performing individual *LoS*-queries.

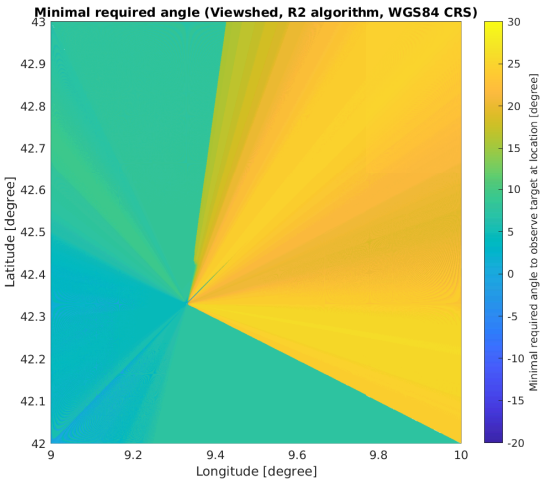
Table 5.8 shows how accurate viewshed methods are in practice when performing a large set of queries on the computed grid. In total, approximately 1 percent of the total number of queries is incorrect when using the *R2* or *XDraw* algorithm. *XDraw* performs marginally better in practice. The mean distance of incorrect queries is greater for *XDraw* than for *R2*, while the standard deviation is roughly equal. This indicates that *R2* has more difficulty at the short range than *XDraw*. Likewise, *R2* has a greater mean error overall.



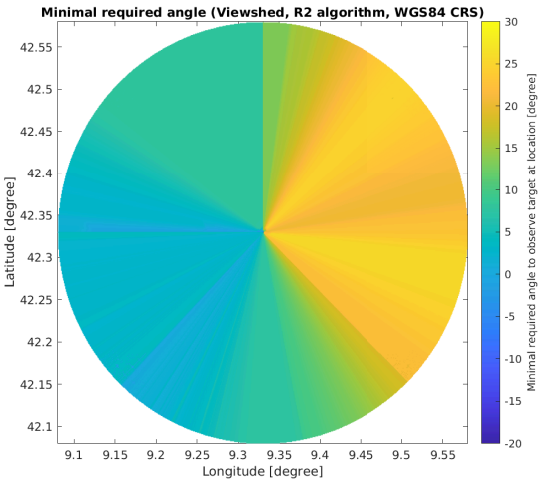
(A) Tile-based *R3* viewshed.



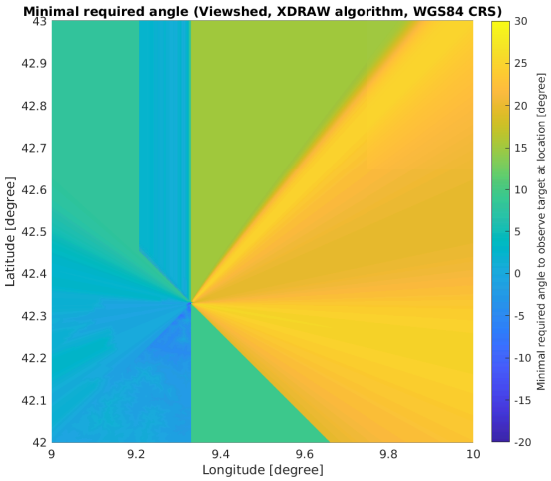
(B) Radial *R3* viewshed.



(C) Tile-based *R2* viewshed.



(D) Radial *R2* viewshed.

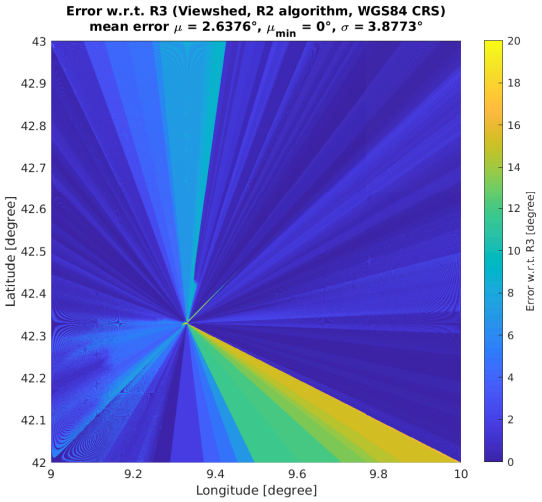


(E) Tile-based *XDraw* viewshed.

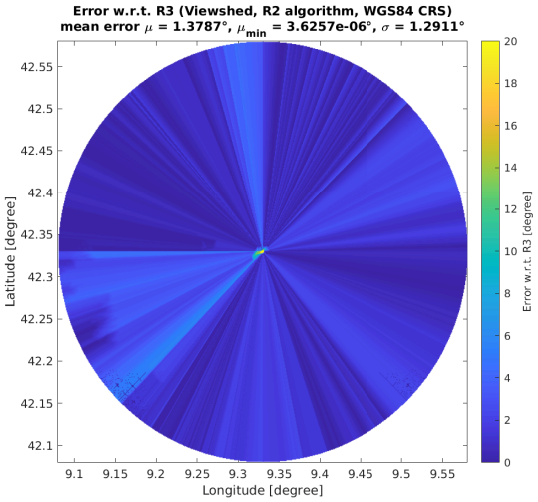
FIGURE 5.11: Viewshed computations of an observer on the location indicated by figure 5.10 at a height of $h = 10$m above the terrain, for all three viewshed algorithms relative to the WGS84 CRS. The values on the grid represent the minimal required angle between the observer an target in order to have *Line of Sight*.

Figures 5.11c and 5.11d show the results of a tile-based and radial *R2* viewshed implementation respectively. *R2* attempts to fill the entire grid solely by performing queries to the outer ring. Comparing the tile-based implementations (figures 5.11a and 5.11c) to their respective *R3* counterpart, the main differences are located in the north-west and south-east of the observer. The cause of this discrepancy is due to the high gradient of angles near the observer, as can be seen in figure 5.10b. This does not occur in the radial viewshed (figures 5.11b and 5.11d).

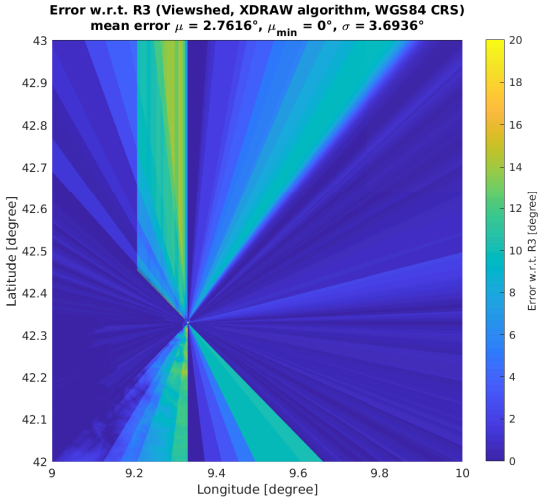
Figure 5.11e shows the result of the tile-based *XDraw* implementation. Due to how *XDraw* operates, intermediate angles on the computed grid are re-used for angles further outward. This can cause interesting fragments where an angle propagates outward when in reality it does not obstruct the view. Such an example is visible in the north-western section of figure 5.11e. Furthermore, because every section is completely independent, and within a section the values in the inner ring have the potential to propagate outward; clear borders between the octants become apparent.



(A) Tile-based *R2* viewshed error. The mean error $\mu \approx 2.64°$, $\sigma \approx 3.88°$.



(B) Radial *R2* viewshed error. The mean error $\mu \approx 1.38°$, $\sigma \approx 1.29°$.



(C) Tile-based *XDraw* viewshed error. The mean error $\mu \approx 2.76°$, $\sigma \approx 3.69°$.

FIGURE 5.12: The absolute difference between the viewshed angle values of the *R3* method and the method being analyzed.

5.6 RQ6: What is the number of static observer *LoS* queries at which pre-calculating an entire viewshed is more efficient than individual queries?

To calculate the break-even number of queries for a given grid, the computation time as a function of grid size for different viewshed methods is analyzed. The number of queries to which this time is equal also depends on the CRS in use. Because *R3* is functionally equivalent to performing a query to every location, it is not considered for this research question. As such, the focus will be on *R2* and *XDraw*. In this experiment, a dummy tile grid of variable size containing zeros will be converted to a viewshed. Then, given the previously measured query duration, the number of queries to break even* will be determined for an algorithm as a function of the grid size.

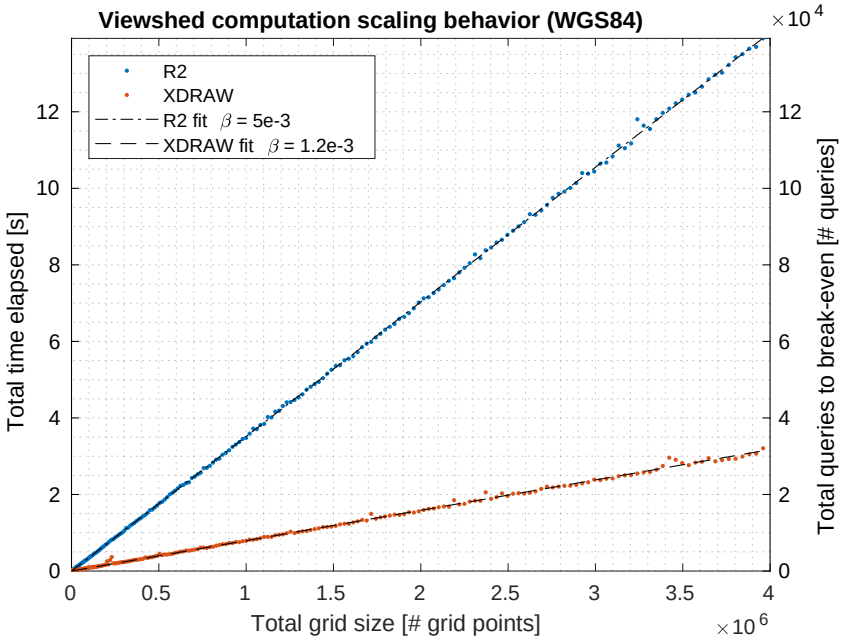


FIGURE 5.13: Viewshed computation scaling behavior, duration (left y-axis) and required number of queries to break even (right y-axis) as a function of the total number of grid points for a live system.

Figure 5.13 shows the scaling behavior of computing a viewshed for the *R2* and *XDraw* algorithm. On the x-axis the total number of grid points is displayed. Both methods exhibit linear scaling with respect to the total number of grid points (quadratic with respect to the radius r). On the left-hand side the total time to produce a viewshed is displayed, which is directly translatable to a number of *LoS*-queries as per previous results (1 query taking $t \approx 1 \times 10^{-4}$s). E.g., a 2000×2000 grid (total 4×10^6 grid points) requires just 3 seconds to compute for *XDraw*, equal to approximately 3×10^4 individual queries. The ratio between grid points and required number of queries to break-even is $400 : 14$ for *R2*, and $400 : 3$ for *XDraw*. As such, when considering a static observer producing a viewshed can provide a significant temporal advantage. Especially when considering one grid point contains the minimal required angle, and this angle can be utilized for targets at any height at the specified location.

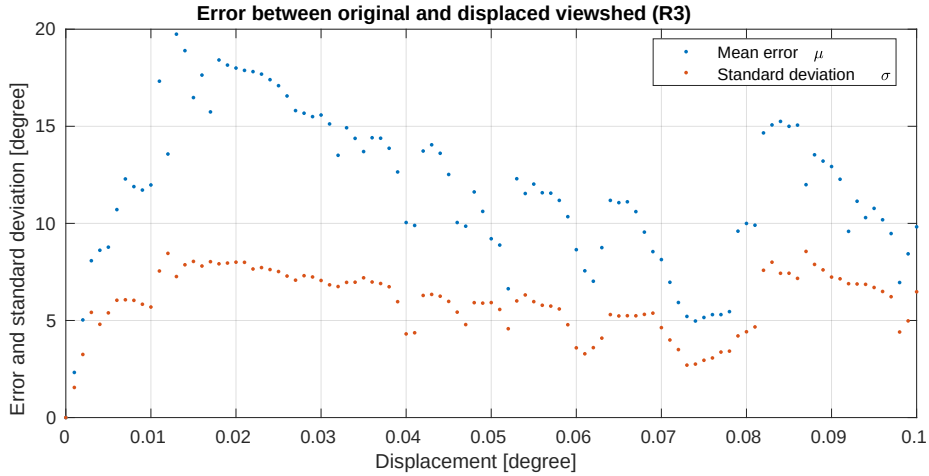
*This assumes viewshed calculations are performed on a live system and as such requires a given number of queries before it is perceived as more performant. In reality, viewshed representations can be calculated off-line, given the position or path of the observer is known beforehand.

5.7 RQ7: What is the maximum offset at which an observer can be approximated as static for the purpose of re-using viewshed results?

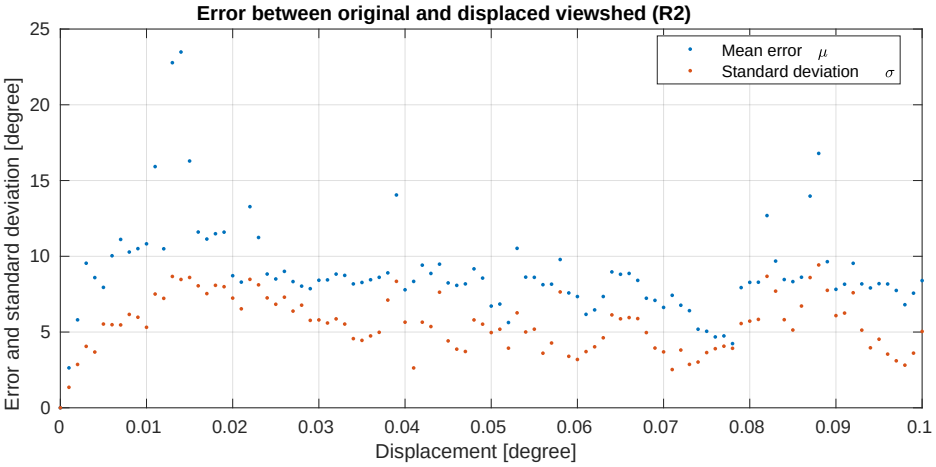
Re-using a previously-computed viewshed saves a significant amount of computational resources. As such, the validity of this approach is tested by moving an observer along the surface of the terrain for different offsets in degrees and calculating the mean of the element-wise difference between the two viewsheds. Moving along the surface was chosen as this is comparable to a real-world use-case, where the distance to the surface stays equal, and not the elevation with respect to the CRS. For every Δx, a new tile-based viewshed is created for all three algorithms, and the mean difference in degrees is plotted as a function of the Δx and shown in figure 5.14.

In figure 5.14a, the *R3* algorithm is analyzed for the purpose of re-using previously computed viewsheds. In the domain $\Delta x \in [0, 0.02^\circ)$, approximately 2km, this difference increases steadily to a total mean error of 20°, after which the mean error declines. The high difference between the initial viewshed and the viewshed at $\Delta x \approx 0.015^\circ$ is likely tied to the specific conditions of this experiment. Where, at this displacement, the peaks happened to align with the troughs. Further along the displacement graph, the mean error $\mu \approx 10 \pm 5^\circ$. Figure 5.14b shows the error caused by displacement and utilizing the original viewshed for the *R2* algorithm. Noteworthy is the persistence of the peak at $\Delta x \approx 0.015^\circ$, but the absence of the clear steady slope towards it. Additionally, for higher Δx, the values are more stable around $\mu \approx 10^\circ$. Furthermore, figure 5.14c illustrates the displacement error for the *XDraw* algorithm. These results are compelling due to their similarity to figure 5.14a. The behavior of the mean error μ as a function of the displacement Δx is similar for the initial domain $\Delta x \in [0, 0.02^\circ)$, and the same peaks and valleys as for higher Δx are visually distinguishable.

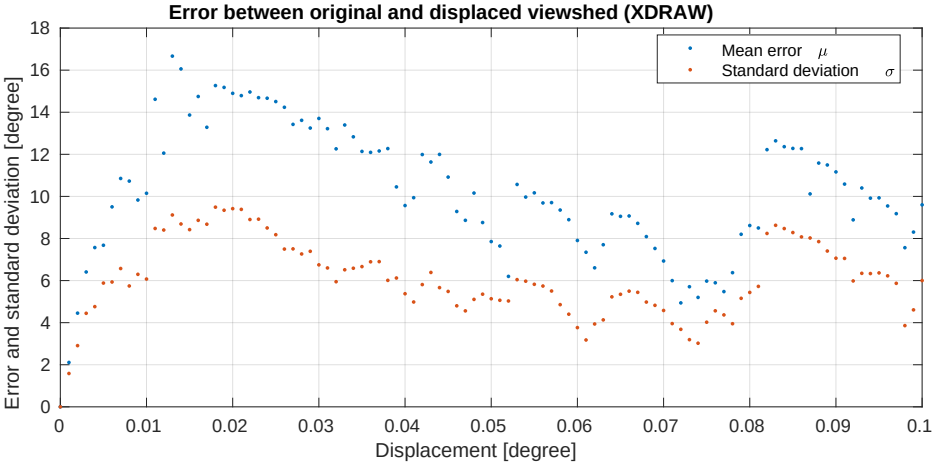
Depending on the allowed maximum error induced by re-using a previously-computed viewshed, this method may provide a viable approximation method. From the displacement error figures 5.14a - 5.14c, for small displacement, the slope of the mean error displays hints of linearity. Therefore, given a fidelity constraint, one can interpolate between these values to conclude a maximal grid shift within the stipulated margin of error. At a shift of $\Delta x \approx 0.001^\circ$ (approximately 100m, one grid point), the error can already be close to $2 - 3^\circ$, which can make the difference between a target being within or outside *Line of Sight*. The error caused by a displacement of 100m can be significant, depending on the surrounding terrain and the direction of movement. For example, a ship close to the coast moving in parallel to the coastline will inhibit a large mean error when re-using a previously-computed viewshed. If, instead, the ship is far away from the nearest landmass and the ship moves away or toward the landmass, this mean error is minimized. In this test, a worst-case scenario was considered over land where, overall, the gradient is relatively high. These results do not rule out re-usage of former viewshed computations on sea, but define a realistic upper bound for the mean error induced by such an approach.



(A) Error between the viewshed at (x_0, y_0) versus $(x_0 - \Delta x, y_0)$ as a function of displacement Δx for the R3 algorithm. A shift of two grid points already causes a mean error μ of $\mu \approx 5^\circ$.



(B) Error between the viewshed at (x_0, y_0) versus $(x_0 - \Delta x, y_0)$ as a function of displacement Δx for the R2 algorithm. A shift of two grid points already causes a mean error μ of $\mu \approx 5^\circ$.



(C) Error between the viewshed at (x_0, y_0) versus $(x_0 - \Delta x, y_0)$ as a function of displacement Δx for the XDraw algorithm. A shift of two grid points already causes a mean error μ of $\mu \approx 5^\circ$.

FIGURE 5.14: Error caused by re-using a previously-computed viewshed as if it were computed at the new coordinate. The mean error μ increases quickly for small Δx.

5.8 RQ8: What is the performance and accuracy impact of compressing a viewshed?

The precision of the variables used to store the minimal required angle for every grid point is reduced and the impact on accuracy caused by this change is studied. The range from the minimal to the maximal value on the grid is optimally used by setting all bits to `0` for the minimal angle, and setting all bits to 1 for the maximum angle in the original viewshed. All other possible values are a linear interpolation between these two values. This way, the impact of reducing the precision is minimized, as more detail is concentrated where it is needed the most; primarily between 0° and 180° or a smaller range dependent on the minimal and maximal recorded angles.

Compressed viewshed precision loss statistics		
Data format	Mean error μ [rad]	Standard deviation σ [rad]
`byte[]`; 8 bit	4.55×10^{-3}	2.68×10^{-3}
`short[]`; 16 bit	1.77×10^{-5}	1.0×10^{-5}
`float[]`; 32 bit	1.73×10^{-8}	9.98×10^{-9}

TABLE 5.9: The absolute difference in angles between the compressed and uncompressed viewshed, quantifying the loss of precision when utilizing the given data format in a real-world use-case of a viewshed of the island of Corsica.

The memory usage of a viewshed can be reduced by using a data format with lower precision. However, this causes the quality of the stored information to degrade. Small differences in angles are smoothed and the space of possible angles the viewshed can contain is reduced. This introduces a domino effect, where a small error in an angle early on (the location of the observer) can propagate outward. This is because the maximum angle is recorded whilst the sight-lines emanate outward, and all values in an outer rings are either explicitly or implicitly compared to an incorrect angle located nearer the observer. Table 5.9 shows the average error induced by utilizing a given data format as a form of compression. This experiment was performed on an identical viewshed, compressed using the three different data formats. The goal of this research question is to find the optimal precision to store the minimal required angles whilst occupying the least amount of memory. The tables shows the `byte[]` format occupies the least amount of memory, but loses a lot of precision. This is inferential, due to the limited number of possibilities ($2^8 = 256$ different angles). Over a distance of 10km, an error of $\mu = 4.55 \times 10^{-3}$ rad can cause a difference in perceived elevation of $\Delta h = 10^3 \times \tan \mu = 45.5$m. Doubling the memory usage increases the precision significantly ($2^{16} = 65536$ different angles). This decreases the mean error μ to $\mu \approx 1.77 \times 10^{-5}$ rad, the difference in perceived elevation over 10km then becomes $\Delta h = 10^3 \times \tan \mu = 17.7$cm.

Furthermore, there is no significant performance impact on encoding and decoding the angles in the modified data format. Running a benchmark took the uncompressed viewshed 135ms to look up every grid value on a 1200×1200 size grid, and 139ms for the compressed viewshed. The difference between these two measurements is negligible. Therefore, the decoding of the data does not give rise to ample latency such that it influences the query duration in a significant manner as the bottle-neck remains retrieving the variables from memory.

Compressed viewshed accuracy statistics					
Error over ...	Algorithm	Precision	# queries	Mean μ [m]	Std σ [m]
All queries	*R2*	`byte[]`; 8 bit	130816	2.177349	23.022003
		`short[]`; 16 bit	130816	1.874981	23.396296
		`float[]`; 32 bit	130816	1.910885	23.590728
		Uncompressed; 64 bit	130816	1.901540	23.549441
	XDraw	`byte[]`; 8 bit	130816	2.165639	19.355694
		`short[]`; 16 bit	130816	1.076114	17.403355
		`float[]`; 32 bit	130816	1.091881	17.540086
		Uncompressed; 64 bit	130816	1.076787	17.419574
Incorrect queries	*R2*	`byte[]`; 8 bit	1894	92.581759	87.450610
		`short[]`; 16 bit	1344	177.154985	143.813533
		`float[]`; 32 bit	1349	177.342550	143.360521
		Uncompressed; 64 bit	1351	178.739156	143.302137
	XDraw	`byte[]`; 8 bit	3048	124.026614	140.424946
		`short[]`; 16 bit	1122	124.943316	140.380712
		`float[]`; 32 bit	1269	110.546730	138.069459
		Uncompressed; 64 bit	1131	124.026614	140.424946

TABLE 5.10: Statistics of utilizing a compressed viewshed instead of performing individual *LoS*-queries. The experiment is performed with identical parameters as the results in table 5.8. The data has been compressed to occupy less memory in the form of a `byte[]`, `short[]` and `float[]` (8, 16 and 32 bit precision respectively).

Table 5.10 shows the impact on accuracy of querying a compressed viewshed instead of performing a new *LoS*-query without any viewshed. Here, the left column indicates the precision of the variables in use. The original, uncompressed format utilizes a 64-bit `double[]`. Encoding the angle in just 8 bit precision provides insufficient resolution to produce accurate results, indicated by the relatively large mean error for both *R2* and *XDraw*. From 16 bit precision and upward, the differences become less noticeable. These data formats provide ample differentiation between the angles to produce results similar to the original 64-bit `double[]`. As such, it is a valid strategy to reduce memory usage by a factor of 4 by utilizing the 16-bit `short[]` data format instead of the original 64-bit `double[]`. This is the optimal middle-ground between fidelity and memory usage as is apparent from table 5.10.

5.9 RQ9: What are the performance and accuracy effects of exact mesh-based collision detection versus exclusively calculating collision detection of the enveloping sphere?

To examine the effects of substituting precise mesh-based collision detection with an enveloping sphere, a suitable model was determined. The model used in the experiment represents a simplified mesh of an airplane, such that the enveloping sphere does not optimally contain the 3d model. The simplification was made to express the characteristics of an airplane, whilst negating most of the details to decrease the total number of calculations as per algorithm 1. Figure 5.15 shows a simplfified 3d model of the airplane used in testing.

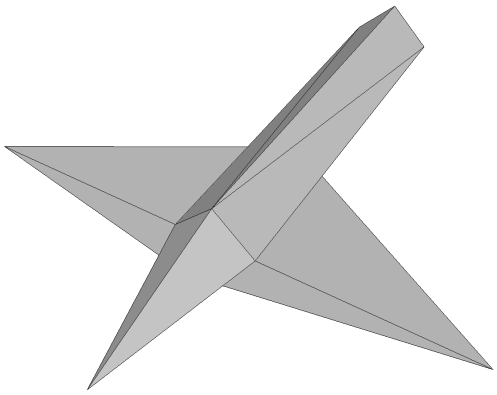


FIGURE 5.15: A simplified 3d model of an airplane, containing the main characteristics of a sharp nose, wings and a body. The model contains just 12 vertices and 15 faces (orthogonal triangles).

To determine whether an entity is visible to an observer, a simple implementation of a sensor has been developed. This sensor defines a rectangular area which has to be exposed in order for the sensor to detect the object. This rectangle expands in size the further away from the sensor an object is, to simulate objects further away being perceived as smaller in size and therefore more difficult to detect. Modeling the detection of visibility for an object from the point of view of an observer occurs in three steps.

1. The first check is whether the sight-line between the observer and the entity encounters terrain. This is achieved via *LoS* algorithm 3.1 for a uniformly sampled set of terrain points on the grid. If at any point one angle between the observer and the terrain exceeds the angle between the observer and the entity, there is no *Line of Sight*.
2. Second, obstruction by other objects is checked by approximating the model as a sphere where the radius is equal to the point furthest from the centre of the model. This way, if the sight-line does not intersect the sphere (calculated using equation 2.3), it is certain to not intersect the model either.
3. Finally, if the previous calculation determined the sight-line to intersect with the enveloping sphere, the individual faces (triangles) of the mesh are checked for intersection (calculated using algorithm 1).

Furthermore, the probability of detection is modeled by dividing the visible region of an object by the minimal required area to detect the object, capped at $P(detect) = 1$. Calculating the visible region from the point of view of an observer is achieved by drawing a circle with the radius of the object's enveloping sphere tangential to the vector between the observer and the target. This area is divided into a grid of points with a density determined by the sampling resolution or number of samples (32×32 in this case). Then, for every point on this grid, a sight-line is emitted from the observer to this point. The ratio between the samples with an intersection with the given entity and the total number of samples gives a rough estimate of the visible region of the entity. In this experiment, the necessity of calculating exact mesh collisions as opposed to collision detection with the enveloping sphere and bounding box is examined, and the latency improvements of this

change are quantified. This is done by conducting an experiment with a 'peeking' entity behind a mountain. The probability of detection $P(detect)$ is measured as the entity moves upward. The difference in the value of $P(detect)$ for both the exact mesh model and the enveloping sphere is measured and displayed in figure 5.16.

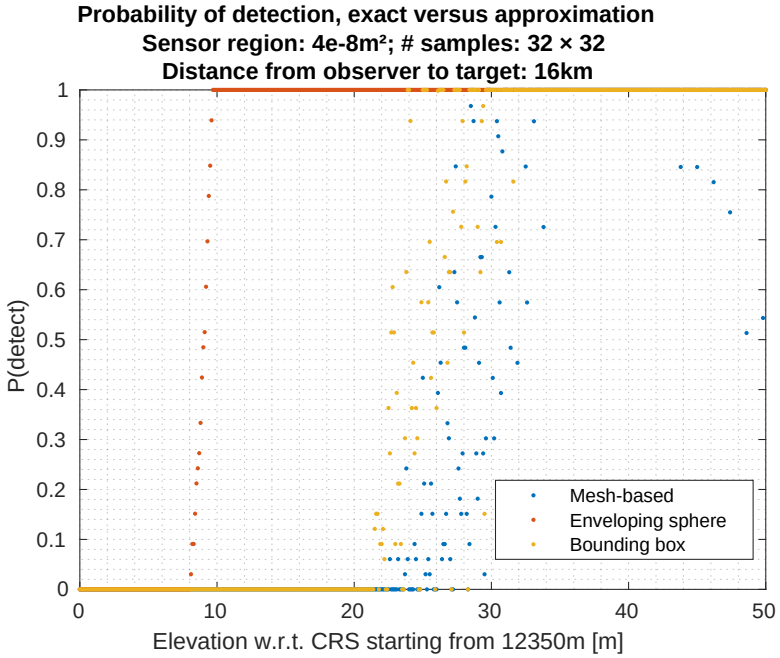


FIGURE 5.16: Probability of detection as a function of the elevation of an entity, where $P(detect)$ is calculated for mesh-based collision detection, an enveloping sphere and a bounding box a for an edge case where the target is approximately 16km away from the observer and near the maximum range of visibility.

Figure 5.16 shows the effect of approximating the surface of the simplistic representation of the airplane (figure 5.15) as a sphere. Drawing a sphere around the airplane leaves a relatively large amount of negative space. In the spherical approximation, detecting this space counts as a detection. Hence, in figure 5.16, the enveloping sphere model is detected first. Eartly on, for low elevation, $P(detect) = 0$, as the elevation increases $P(detect) \to 1$. During an intermediate period where $0 < P(detect) < 1$ holds, the sensor has *Line of Sight* with a part of the surface of the entity. However, depending on the predefined sensor region, it may not have exceeded a threshold. The sensor region dictates the minimal required size of an object as a function of the distance, its area scaling quadratically over distance from the sensor. This serves as a simplified model to limit the view distance. In this experiment, the length of the plane is approximately 30m, with a height of 4m. This creates a negative space of $\Delta h \approx 12$m, which is visible in figure 5.16 as the horizontal difference between the offset at which either model is detected. Here, the difference of elevation between the moments the mesh-based and enveloping sphere models are detected is approximately 12m. Furthermore, a bounding box drawn around the airplane was considered as a viable alternative. Figure 5.16 shows the bounding box model behaves very similar to the mesh-based method. This is because the negative space of the bounding box is much smaller, and as such, there are fewer false positives. The advantage of the bounding box method as opposed to the mesh-based method is its reliability; the bounding box will, by definition, always have 6 faces (each consisting of 2 triangles) defining its shape. The computation time of the mesh-based model depends on the complexity of the 3d model. Therefore, the bounding box model is a good

middle-ground between accuracy, performance and reliability for the purpose of calculating the sensor coverage for the entity.

The previous experiment examined an environment where the number of total entities is one, and is therefore not representative of a real-world use-case. Several methods will therefore be compared by measuring the computation time as a function of the number of environmental entities taken into account. To account for outliers, the average over $N = 100$ queries is calculated, the number of entities is step-wise increased by 100 until a total of 10^5 entities is reached.

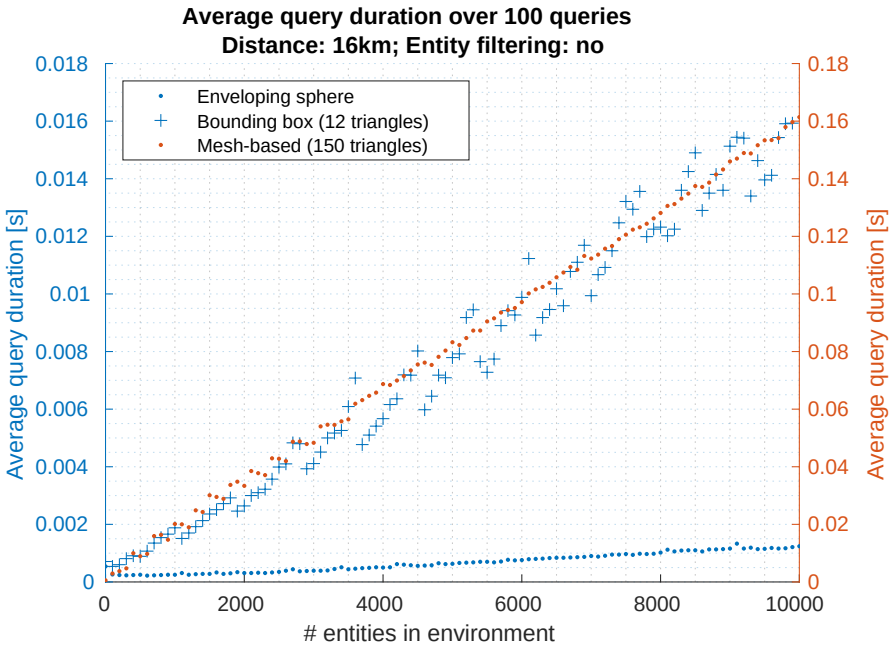


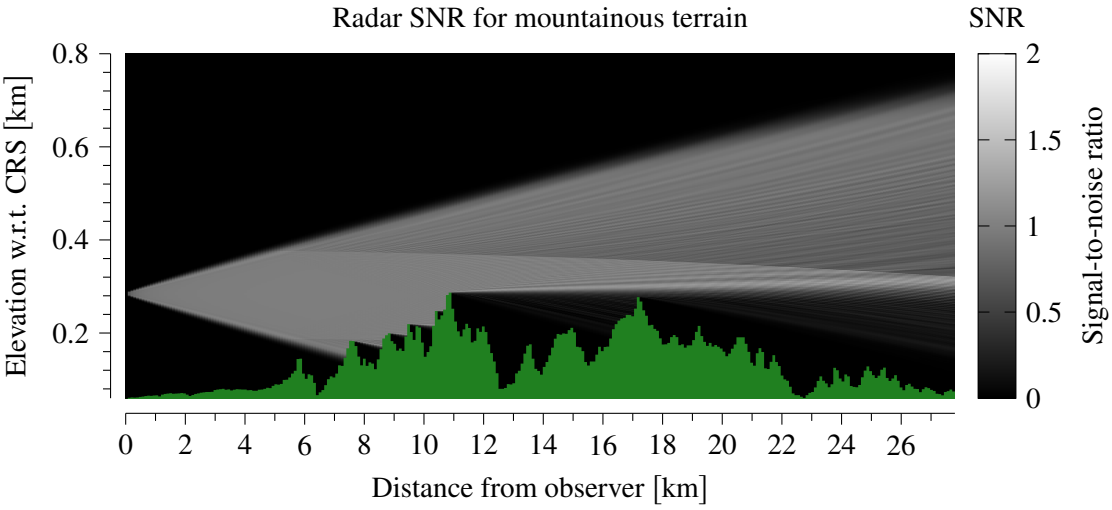
FIGURE 5.17

Figure 5.17 shows the scaling behavior of *LoS*-queries for a variable number of entities placed in the world. To form a worst-case scenario, no filtering was performed on the entities in order to omit performing redundant calculations for entities which are out of range. This represents a real-world scenario where a large group of entities is clustered together. Two y-axes are drawn, with the color indicating the results' corresponding scale. The graph displays the amount of time to process *LoS* for two points over a distance of 16km, given the presence of x entities, using three different methods. The base duration of $t < 1$ms as per previous results is overshadowed by significant margins due to the collision detection logic for a high number of entities. However, the impact is very limited for the enveloping sphere model, remaining $t < 2$ms for $N = 10^5$ entities. This is due to the simple arithmetic involved in calculating intersection with a sphere. First applying intelligent filtering based on the location and proximity of entities, and subsequently performing bounding box or mesh-based collisions detection will outperform any of the studied methods displayed in figure 5.17. However, this filtering of entities is very situation-dependent and difficult to quantify and therefore a worst-case scenario was studied. Based on the required fidelity of an *LoS*-service, three options remain. These options are, in increasing order of fidelity and resource usage; *a)* limiting collision detection to a sphere enveloping all vertices, *b)* limiting collision detection to the bounding box enveloping all vertices, and *c)* calculating exact mesh-based collision detection. The latter method (c) can call a subroutine for method a and/or b in order to omit exact mesh-based collision detection if either returns *false*.

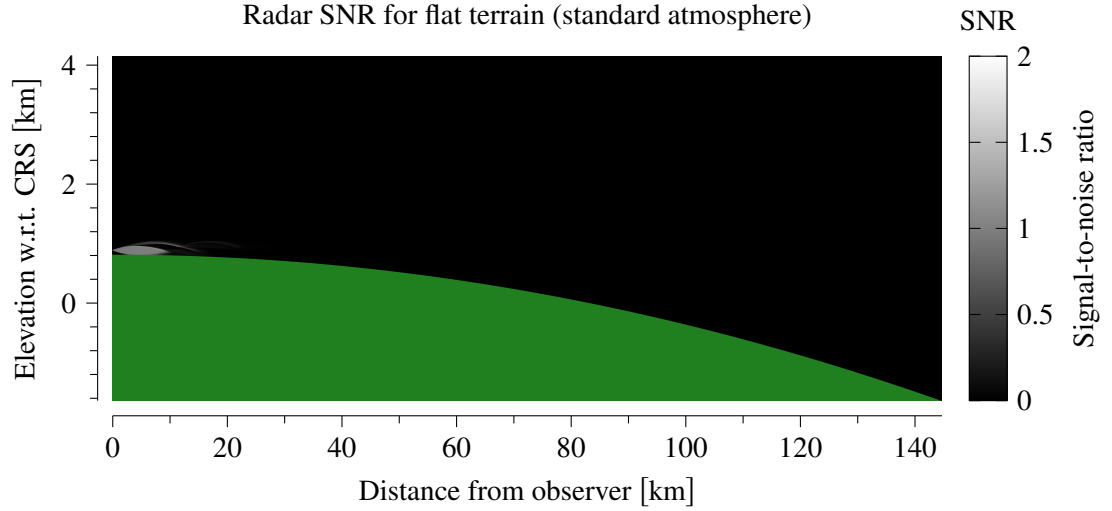
5.10 RQ10: What is an appropriate SNR threshold to model the unique effects of radar propagation such as refraction and ducting?

To test the viability of the Parabolic Equation Radar model, three scenarios were considered; a mountainous terrain with a realistic atmospheric density, a flat terrain with a monotonic atmosphere and a flat terrain with an elevated duct. The raw output of the simulations are given in figure 5.18. Here, the black to white gradient represents the signal strength in the form of the signal-to-noise ratio, and the green color represents the presence of terrain. In figures 5.18a - 5.18c, the SNR for a bi-directional query to the location on the grid at a given location is shown. The algorithm is initiated with a given set of radar parameters and initial conditions such as antenna height, emission angle and power. Then, using the Parabolic Equation (PE) model, step-wise the algorithm marches towards to target. Memory usage is limited to a single slice $z \in \{z_{min}, ..., z_{max}\}$ tangential to the surface of the Earth at the observer's position, and this matrix is mutated in-place. Depending on the maximum range of the query, the duration of the algorithm can vary from milliseconds up to $t \approx 20$ seconds for a total range of $d \approx 150$km. This depends on the physical parameters as well as the terrain sampling rate.

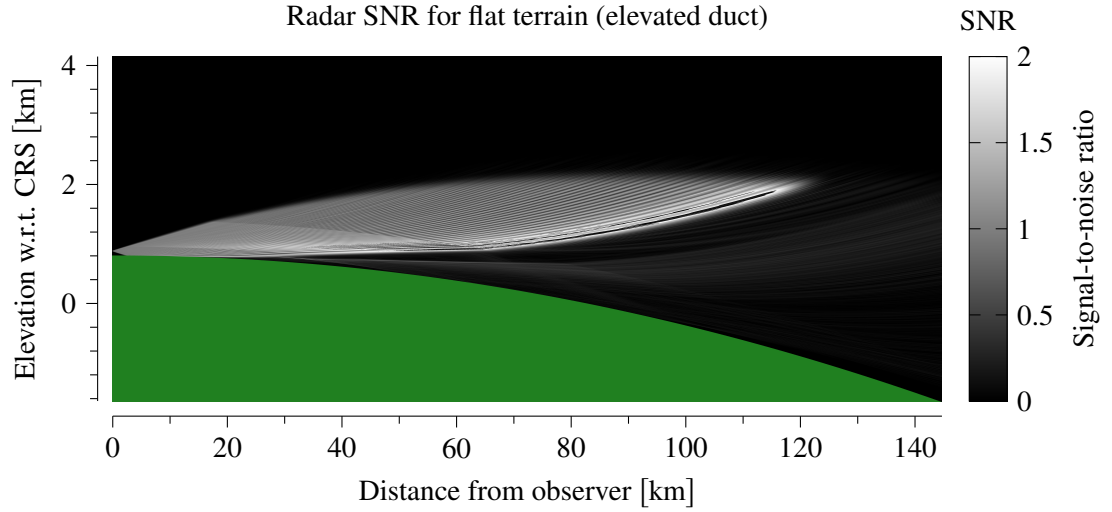
Figure 5.18a shows a query for a high-elevation antenna pointing towards a mountainous area. As the radar waves strike the mountains, a blind zone behind the summit arises. However, due to the physical properties of waves the signal 'bends' around the peak, akin to how sound waves can bend around a corner. In the atmosphere, there is a distinct horizontal line above the terrain where the SNR is stronger below and weaker above this line. This line represents a gradient in temperature and density in the atmosphere. The gradient tends to reflect radio waves downward, but parts do not reflect and pass through this layer. Once a radio wave escapes this duct, the atmosphere is monotonic and therefore it continues without further perturbations. Figures 5.18b and 5.18c show an example with identical parameters but a difference in atmospheric gradient; for figure 5.18b the atmosphere is monotonic and for figure 5.18c an elevated duct is introduced. The latter scenario, the duct, acts as a wave guide for the radio waves and carries the signal further along the terrain. This behavior is characteristic for radio waves. Next, a simplified model will be composed based on an SNR threshold which can translate these values for the SNR to a *true* or *false* response from a *LoS*-service. This simplistic detection model is not rooted in physics, but serves as a placeholder for a real detection model.



(A) Radar propagation for mountainous terrain with temperature gradient in the atmosphere.



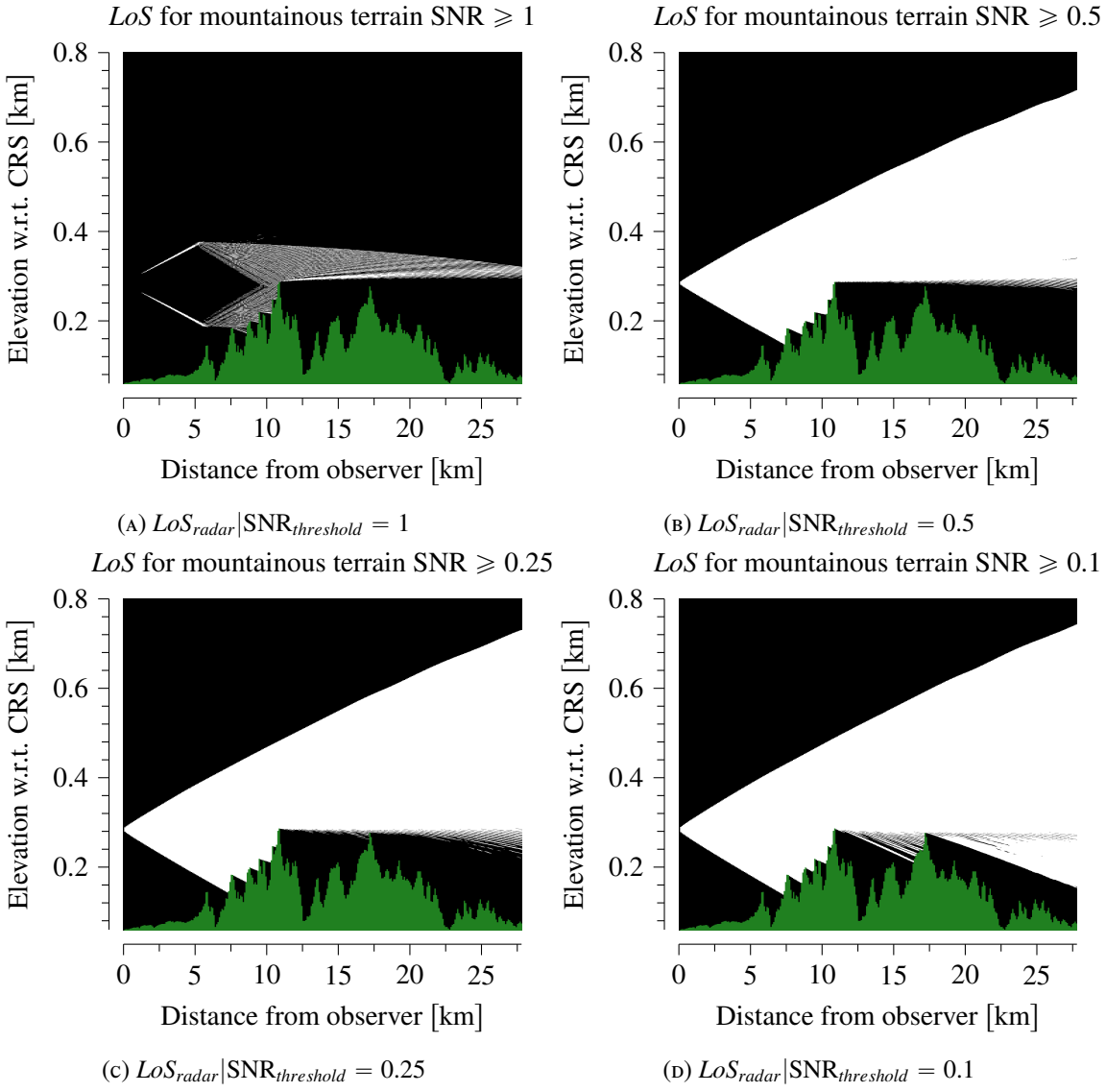
(B) Radar propagation for flat terrain with a monotonic atmosphere.



(C) Radar propagation for flat terrain with an elevated duct.

FIGURE 5.18: Radar propagation for three models; mountainous terrain with a temperature gradient, flat terrain with a monotonic atmosphere and flat terrain with an elevated duct.

In order to translate the matrix of values for the SNR to a matrix of boolean *true* or *false*, a detection model consisting of a step function was considered. In this model, represented by formula 2.8, the process of finding a correct value for the threshold consists of recognizing radar-like behavior in a set of possible values of the threshold. This will be executed on case-by-case basis, starting at the scenario illustrated in figure 5.18a, and ending with the scenario illustrated in figures 5.18b and 5.18c. Formula 2.8 will be applied to the data in figures 5.18a - 5.18c, and displayed as the color white for *true*, the color black for *false* and green for terrain.



(A) $LoS_{radar}|\text{SNR}_{threshold} = 1$

(B) $LoS_{radar}|\text{SNR}_{threshold} = 0.5$

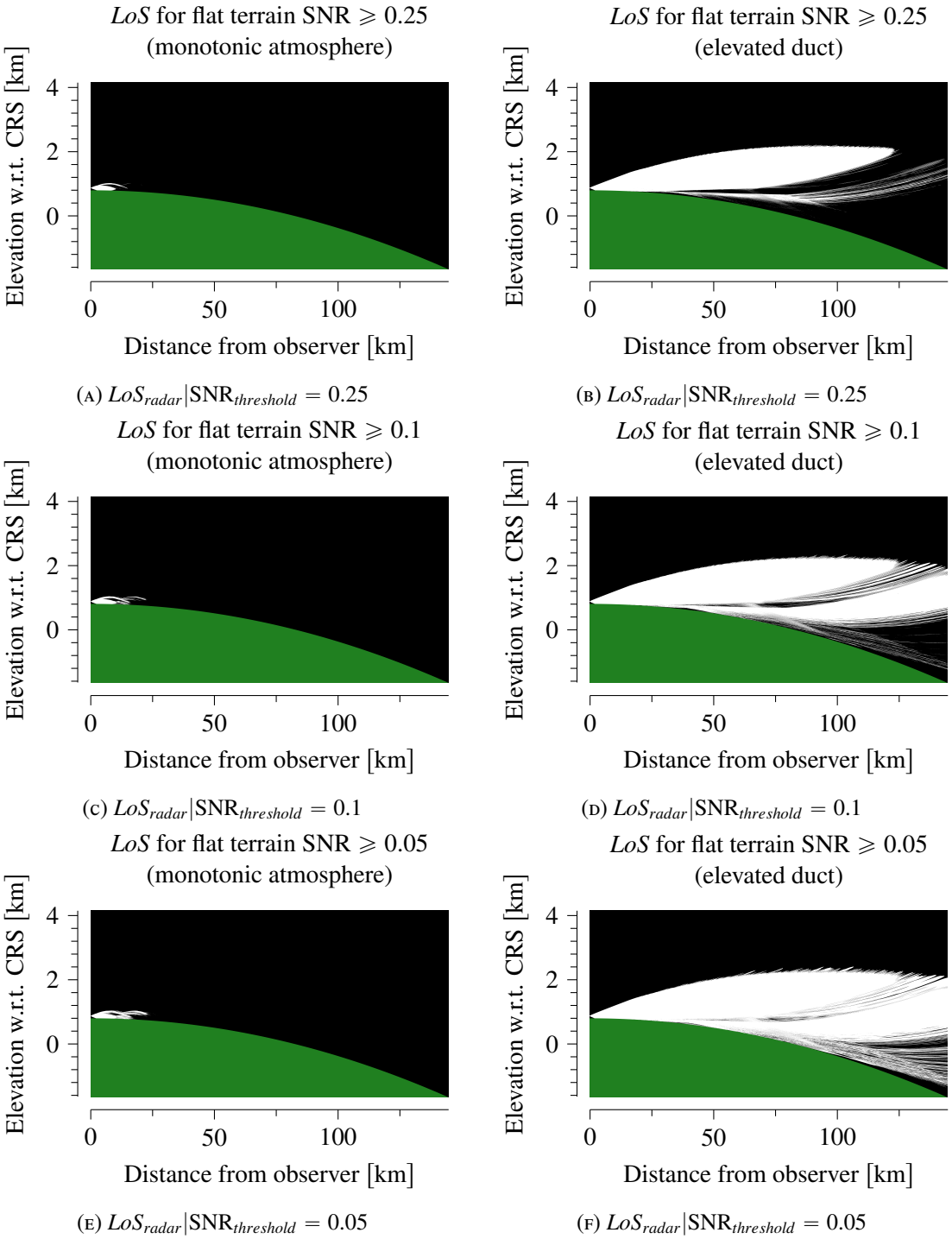
(C) $LoS_{radar}|\text{SNR}_{threshold} = 0.25$

(D) $LoS_{radar}|\text{SNR}_{threshold} = 0.1$

FIGURE 5.19: Radar *Line of Sight* for mountainous terrain illustrated in figure 5.18a for different SNR thresholds applied to formula 2.8.

From this figure, it is clear that a SNR threshold of $\text{SNR}_{threshold} \geqslant 1$ is too high, as figure 5.19a displays a very clear radar dead zone where a target should be visible; right next to the antenna without any terrain obstruction. Figure 5.19b displays similar behavior to that of an electro-optical sensor; only sight-lines drawn directly from the observer to a point are visible. The waves do not curve after passing a mountain summit. Further decreasing the threshold to $\text{SNR}_{threshold} = 0.25$ and $\text{SNR}_{threshold} = 0.1$ produce figures 5.19c and 5.19d respectively. Both show behavior similar

to waves curving around the peaks, the latter showing a more extreme variant than the former. Depending on the use-case, either of these values can produce interesting results for a simplistic Radar *LoS*-service. In the next scenario, only the latter two SNR thresholds will be considered, and behavior induced by an elevated duct as opposed to a monotonic atmosphere will be compared. In a monotonic atmosphere, the signal is less prominent for larger distances.



(A) $LoS_{radar}|\text{SNR}_{threshold} = 0.25$

(B) $LoS_{radar}|\text{SNR}_{threshold} = 0.25$

(C) $LoS_{radar}|\text{SNR}_{threshold} = 0.1$

(D) $LoS_{radar}|\text{SNR}_{threshold} = 0.1$

(E) $LoS_{radar}|\text{SNR}_{threshold} = 0.05$

(F) $LoS_{radar}|\text{SNR}_{threshold} = 0.05$

FIGURE 5.20: Radar *Line of Sight* for flat terrain illustrated in figures 5.18b and 5.18c for different SNR thresholds applied to formula 2.8.

In figure 5.20, a set of different values for the SNR threshold are compared. Decreasing the thresholds leads to greater radar visibility in general, which is to be expected. For the value which resulted as a potential candidate from figure 5.19; $\text{SNR}_{threshold} = 0.25$ in figure 5.20b, there is slight visibility beyond the horizon, with a dead-zone in-between. The dead-zone remains for $\text{SNR}_{threshold} = 0.1$ in figure 5.20d, but the visibility beyond the horizon is increased significantly. This is due to the elevated duct guiding the radar waves across the duct situated near the surface of the Earth. Further decreasing the threshold to $\text{SNR}_{threshold} = 0.05$ in figure 5.20f results in visibility beyond the horizon for a distance of $d \approx 125$km. Due to radar parameters and characteristics being unknown for the scope of this paper, whichever of these three is the most realistic is left open. However, from existing research 2.5 it is known that elevated ducts can greatly increase the range of a radar system. As such, depending on the use-case, scenario and radar characteristics, a realistic threshold must be chosen which satisfies the behavioral constraints. Notably, this behavior is only characteristic for the parameters used in this specific simulation of a radar sensor. As such, when altering these values, the presented results and observations are invalidated.

In the presented use-case of real-time *Line of Sight* calculations the Parabolic Equation model does not yield satisfying results. For medium range distances $d \leqslant 7$km, the time to perform a query equals $t \approx 1$s, and for small range distance $d \leqslant 1$km the query duration is in the order of magnitude of hundreds of milliseconds. For long range queries, the query duration far exceeds any real time latency requirement. In order to consider applying the PE radar model to short ranges, it must provide advantages over the much less computationally expensive geometric electro-optical model. However, radar characteristics are mostly not present at the short range, and only become apparent at the longer ranges. As such, in the domain where the PE model can be considered, the geometric *LoS* model provides a more performant approach.

Chapter 6

Conclusion

6.1 Accounting for the curvature of the Earth

RQ1: What is the impact of converting from WGS84 to a perfectly spherical CRS in terms of conversion time and accuracy loss?

The first research question stipulated in this paper was an out-of-the-box approach to computing the angle between an observer and a target. Given the complications imposed by the ellipsoidal shape of the Earth and therefore the more complicated mathematics of determining the angle; the question became whether projecting to a more convenient reference system can suffice. Initial results showed a promising trend; computations are cheaper and most results are identical in the initial unit tests. The error rate remained far below the 1 percent, as only 231 out of 8386560 queries were incorrect. However, more rigorous testing and quantification of the errors showed the results of table 5.1 and the incorrect queries displayed in table 5.2. The magnitude of some of the errors are significantly greater than anticipated and do not suffice for a *LoS*-service. However, one can make the argument of the applicability to the current use-case; all incorrect queries arose on locations near mountains, and for the naval domain this will not cause issues as a ship will not get as close to a mountain as occurred in the experiment. In the experiment, the steep elevation on the error-prone locations was approximately 1 kilometer over 90 meters (figure 5.3). As such, in scenarios where the terrain gradient near the coast is below this threshold, the error induced by the projection to the spherical CRS does not influence the queries in a significant manner. In reality, this would exclude the usage of this model near locations with cliffs, as these can exceed the aforementioned steep terrain gradient. The spherical CRS provides a framework of calculating the angle between the observer and target represented by equation A.1 which is 30 percent less computationally expensive than the exact vector-based ECEF approach of equation 3.1. Time consumption for the projection of a grid tile to its spherical representation cost costs approximately 0.4ms per 1000 grid points, and approximately 1s for a common 1200×1200 DEM tile on a 4-core 4GHz CPU (not multi-threaded). As such, this pre-computation does not pose a bottleneck for real-world usage when several seconds are available. Moreover, these tiles can be converted in a background process as most of the miscellaneous computations for *LoS* are single-threaded due to their consecutive nature. The physical features of the original tile were visible in the converted tile, as is visible from figure 5.1a (before) and 5.1b (after). As a result of the elevation data being relative to a sphere with a constant radius, the elevation and latitudinal positions have changed.

RQ2: Does the approximation approach provide a valid model of the Earth's curvature, and at what distances is this approximation valid?

The formulae for approximating the curvature of the Earth provide a valid model for calculating *Line of Sight*, where both the geodetic distance and central angle approach produce identical results in this experiment. The difference between the values of the curvature correction of either method was negligible for the presented use-case, as is displayed in figure 5.7. Here, the difference between the two approximation techniques is shown to be up to 30cm for a distance of up to 200km; too small to influence *LoS*-queries, as other factors such as the approximation of the shape of objects (method-dependent) and the terrain sampling resolution of the DEM (90 meters) overshadow this small value. On the island of Corsica, where the experiment was conducted, a total of 36 errors were found between the exact vector-based ECEF approach and the approximation approach. Previous calculations shown in figures A.1a and A.1b illustrated a worst-case scenario for approximating the curvature of the Earth using a spherical model, and these effects are hereby shown to not influence the results in a magnitude considered influential. Moreover, this approach outperforms the exact vector-based approach by 25 percent. As such, it is marginally slower than the spherical model but its errors are much smaller. This is because the solution proposed in this research question utilizes the original grid, and only translates the elevation value of a grid point downward according to its surface distance to the observer; thereby omitting the possibility of shifting the grid incorrectly as was the case in the projection method proposed in RQ2. Furthermore, a model of a flat Earth was considered and shown to not perform well overall. For queries at the short range, the flat Earth model can provide a valid approximation due to the small amount of curvature-induced drop-off. However, in the short range, all alternative methods are less computationally expensive to perform as well due to their cost increasing linearly with distance. As such, there is no quantified advantage of using a flat Earth model for the short range. The geodesic and central angle approach were generally faulty at the larger distances ($10 - 20$ km), indicated by the high value for the mean distance between the observer and target for the incorrect queries displayed in table 5.4. The mean error over all incorrect queries remained small at only 11cm. In other words, this method was only incorrect for queries where (on average) an altitudinal difference of 11cm made the difference between *LoS* and no *LoS*. Therefore, this method is a viable candidate for approximating the curvature of the Earth in a consistent and fast manner, as this value is well within the error margin induced by the other factors mentioned before. In the use-case presented in this paper, where a balance must be struck between fidelity and performance, this method provides the optimal middle-ground between a the computationally expensive exact vector-based ECEF method and the much less expensive but less reliable spherical projection method.

6.2 DTED sampling and interpolation techniques

RQ3: What is the impact of different DTED sampling techniques on performance and accuracy?

The three functions applied to every block of grid points (average, maximum and minimum) produced results in agreement with the hypothesis; an increased amount of incorrect *LoS*-queries for either method. The minimum approach yielded the worst results, a 2.37 percent decrease in accuracy as is displayed in table 5.5. In contrast, the maximum technique only resulted in a 0.99 percent accuracy decrease. This disparity is caused by the format of the experiment where, for

the permutations of all grid points, most *LoS*-queries result in *false*. As such, the minimum approach which causes more queries to be *true* has more impact on the accuracy than the maximum approach which has the opposite effect. The average method produced the most reliable results where, most notably, the mean error over all and just the incorrect queries are the lowest; 16.7cm and 11.8m respectively. However, in use-cases where a false positive is more detrimental than a false negative, the choice to limit *LoS* using the maximum approach is preferred. Furthermore, applying the maximum or minimum reduction technique can be a valid strategy in certain scenarios. For example, in order to determine *Line of Sight* between an observer and a far-away target, a low-resolution tile as a result of the maximum reduction approach can be used as an initial test; if the response of the *LoS*-service is *true*, then the result of using the original tile will be *true* as well. The inverse holds for the minimum approach; if the response of the *LoS*-service is *false*, the result of using the original tile will be *false* as well. Depending on the conditions and exact use-case, applying these resolution reduction techniques to a section of sea in the naval context can provide latency improvements as is displayed in figure 5.9.

RQ4: What is the impact of different DTED interpolation techniques on performance and accuracy?

Three prominent interpolation techniques were compared by first removing information from a DEM grid and then attempting to recover the original data. This approach directly quantified the applicability of either method to real-world usage, and its ability to fill in the 'missing gaps' on a continuous grid. From table 5.6, the hypothesis was confirmed; the more computational resources utilized in the interpolation process, the smaller the error. The bi-linear and bi-cubic interpolation technique rely on the assumption of continuous terrain, as both attempt to fit a function between their neighbouring points. As such, both techniques produced reliable results with a mean error of 5.3m and 4.4m respectively. Additionally, both techniques are significantly more reliable than the nearest-neighbour approach, with its mean error of 16.6m. This large error is caused by the relatively large delta between grid points in this experiment; 90m on the unaltered grid and 180m on the reduced grid. Traveling a distance of 90 meters can cause a significant shift in elevation for a terrain as mountainous as Corsica. The decrease in error between bi-linear and bi-cubic interpolation is relatively small compared to the increased cost. Therefore, bi-linear interpolation provides an optimal middle-ground between the inaccurate nearest-neighbour and the costly bi-cubic interpolation technique. If a higher fidelity is required, the decision to utilize bi-cubic interpolation provides more reliable results at high cost. The error values in table 5.6 represent a mean over a worst-case scenario where an 180×180 meter grid was reconstructed into a 90×90 meter grid spanning the same area. The higher the resolution of a grid, the smaller the interpolation errors will be overall. Given that most DEM grids considered for usage during the process of writing this paper have approximately 100×100 meters as their sampling resolution, performance will be better than illustrated in table 5.6. To reduce the number of calculations and the complexity of interpolation, an optimization was introduced in the sampling technique of the *LoS*-service in order to half the computational resources required for interpolation. To achieve this, only points aligned to at least one grid line are sampled. This way, for any grid point in the list of samples, the distance between either its closest horizontal or vertical neighbour is zero and the interpolation step in said dimension can be omitted.

6.3 Pre-computing viewshed for (near) static observers

RQ5: What are the advantages and disadvantages of the different viewshed approaches with respect to performance and accuracy?

In the process of comparing *R2* and *XDraw* to the baseline *R3* viewshed algorithm, a number of conclusions can be draw. For one, the accuracy of both algorithms is comparable, but a significant mean error of 2.64m and 2.76 was established for *R2* and *XDraw* respectively. An error of 2° over a long distance represents a significant difference in elevation. However, most of these errors were concentrated in certain regions, which can be found in figure 5.12.

For the *R2* algorithm in particular, figure 5.12a shows a high magnitude error region in the south-eastern segment of the tile. This is caused by uneven terrain in this section, which is located very close to the observer. The largest section of the viewshed contains an error of less than 2°, indicated by the dark blue regions. Here, pure dark blue represents an error of zero and complete overlap between the results of *R3* and *R2*. Furthermore, for the *R2* algorithm, a careful balance must be struck between two parameters; the grid alignment constants ϵ and the number of intermediate edge queries N. The former constant represents the maximal proximity by which a sampled grid point is considered for recording to the viewshed, here $0 \leqslant \epsilon \leqslant \frac{1}{2}$. Ideally, the value of this constant would be as small as possible, or zero. However, then the final grid would have a large section of `null` data due to intermediate points not being recorded to the grid. The second constant N represents the number of queries to emit for every point located on the edge, with an offset of $\frac{1}{N}$ grid point. Increasing the value for this integer value can enable usage of a decreased value for ϵ, due to an increased possibility of any query crossing a given grid point and therefore increase the accuracy of the resulting viewshed. Conversely, this does raise the required amount of computational resources by a factor of N. In this experiment, $N = 4$ provided the optimal balance between resource usage and accuracy.

The *XDraw* algorithm performs more reliable in the presented use-case, this is shown in figure 5.12c. Here, the south-eastern region does not exhibit a large discrepancy between the results of *R3* and *XDraw*, as was the case with *R2*. However, this phenomenon occurs in the north-western region instead. It is unclear why, as there is no direct conclusion to be drawn based on the observer's position illustrated in figure 5.10. Instead, this is likely caused by the interpolation between two angles in the ring-based marching algorithm of *XDraw*. The algorithm compares the angle between the current grid point and the observer to an interpolated angle in the previous ring. This interpolation happens linearly, and this process can open up the possibility of propagating an incorrect angle outward. This is a common artifact of the *XDraw* algorithm, together with the rise of clear boundaries between the octants in the viewshed representation of figure 5.11e.

RQ6: What is the number of static observer *LoS* queries at which pre-calculating an entire viewshed is more efficient than individual queries?

One of the advantages of the *XDraw* algorithm is its performance, requiring just below 22 percent of the resources required for the *R2* implementation. This is shown in figure 5.13, where the viewshed computation time is plotted as a function of the total number of grid points. This difference of a factor of 4 originates from the total number of queries being emitted for every grid point on the edge, represented by the constant N. In the current implementation, $N = 4$ was chosen as the

ideal balance between accuracy and performance by a process of iteratively decreasing ϵ (starting at $\epsilon = \frac{1}{2}$) and increasing N (starting at $N = 1$) until a pre-defined set of *LoS*-queries returned the correct results and no empty grid points remained on the grid. However, this increases the computational resources required to fill the grid. Decreasing N or increasing ϵ negatively affected the quality of the resulting viewshed representation in the testing process, during which the mountainous landscape on the island of Corsica was primarily used. More careful modification of the parameters depending on the required fidelity can yield more optimal results. For example, in the scope of the naval use-case, *R2* can likely function with $1 \leqslant N < 4$, due to few gradients in the region surrounding the position of the observer. Thereby decreasing the computational resources required to perform *R2*. The *XDraw* algorithm cannot be modified by changing parameters, but its interpolation-reliant ring-based comparison of angles leaves room for future improvement. Ideally, interpolation would not occur for a large delta between two neighbouring values in the inner ring, as these are likely the cause of errors which are prone to propagating to outer rings. This will further increase resource usage, but improve the fidelity.

Figure 5.13 shows the number of queries to 'break-even' on the right-hand y-axis, based on previous query duration measurements. For a grid of size 2000×2000 this is approximately 3×10^4 queries, and for a grid size of 1200×1200 this is approximately 2×10^4 for the *XDraw* algorithm. This corresponds to a grid points to query ratio of $400 : 3$. This is increased by a factor of 4.7 for the *R2* algorithm in its current form with a grid point to query ratio of $400 : 14$, due to modifications made to increase its accuracy at the cost of performance as explained in the previous paragraph. This is a relatively high number of queries to perform, and depending on the use-case and velocity of the observer it may be a rare occurrence to reach this 'break-even' threshold before the viewshed is invalidated. However, the possibility to compute these viewsheds off-line beforehand or in parallel must be considered before writing off this method as a viable approach.

RQ7: What is the maximum offset at which an observer can be approximated as static for the purpose of re-using viewshed results?

This research question represents questioning an important assumption of any viewshed representation; the axiom its inherent validity is based on the position of the observer. Changing the position of the observer inherently invalidates the previously-created viewshed due to a change in perspective, and therefore *all* calculations must be revised. To test this axiom to experimental data, the mean error of a full viewshed was compared for a range of values of the longitudinal offset, the value by which the observer moves westward. For every value for the displacement, a viewshed was computed and the average difference between the original and new viewshed is calculated and recorded. The resulting graph, displayed in figure 5.14, shows this mean error as a function of the horizontal displacement. Figure 5.14a and 5.14c in particular show behavior similar to linear scaling for the displacement up to 0.01 degrees, roughly corresponding to a surface distance less than 1km on the latitude of Corsica. Depending on the exact conditions, such as the position of the observer and target, shifting a select few grid points can still provide a satisfying answer when queried from the viewshed corresponding to the previous position. However, in general, the increase of the mean error is represented by a steep slope in the graphs of figure 5.14. This is caused by the relatively low resolution of the grid with respect to the gradient of the surrounding, mountainous terrain. However, applying these results to the naval use-case implies a resulting decreased overall error due to a larger separation between the ship and high-elevation points. Furthermore, the direction of the observer influences the error between the current and pre-

vious viewshed. In particular, considering the naval use-case, when the primary landmass is in a given direction; moving away from this landmass causes a relatively low delta due to the marginal change in perspective. Conversely, moving in parallel to the landmass maximizes the delta due to significant changes in perspective induced this movement.

RQ8: What is the performance and accuracy impact of compressing a viewshed?

The initial test performed for the purpose of answering this research question examines the loss of precision when encoding a value between 0 and π radians (or conversely 0° and 90°) for different data types. Table 5.9 shows a quantification of this precision loss of a real-world use-case on the island of Corsica where a viewshed was compressed to occupy a fraction of the original 64-bit array. The most imprecise primitive data type in Java is the 8-bit `byte` type, containing just $2^8 = 256$ different possible configurations. From the initial experiment, it can be concluded that this precision does not suffice for the purpose of recording angles to the grid, as the mean error of $\mu = 4.55 \times 10^{-3}$ rad posed a significant difference of approximately 45.5m over a distance of 10km. The `short` data type occupies twice the memory of a `byte`, and therefore is considerably more precise at $2^16 = 65536$ possible configurations. The mean error $\mu = 1.77 \times 10^{-5}$ rad over a distance of 10km becomes just 17.7cm. The `float` data type is even more precise, but exhibits diminishing returns. Twice the memory usage from 16-bit to 32-bit further decreases the mean error to $\mu = 1.73 \times 10^{-8}$, but this does not influence the *LoS*-queries as the error has already become negligible at 16-bit precision.

Furthermore, on top of quantifying the loss of precision when compressing the viewshed, the compressed viewshed accuracy was compared to the baseline model of performing individual *LoS*-queries. In table 5.10 these results are shown for the *R2* and *XDraw* algorithm, for both the tile-based and radial implementation. The same conclusion from table 5.9 can be formed; the `byte[]` does not provide ample precision in order to encode the minimal required angles in the viewshed. This can be concluded from the number of incorrect queries in the fourth column of the 'Error (incorrect)' row. The number of incorrect queries is much greater for the `byte[]` format for both the *R2* and the *XDraw* algorithm. The difference being even more noticeable for the latter, where 3048 queries are incorrect on the 8-bit `byte[]`-encoded viewshed, and just 1122 are incorrect for the 16-bit `short[]`-encoded viewshed. In some cases the `short[]` performed better than the uncompressed viewshed, this is likely due to the rounding induced by the less precise data type causing an unforeseen positive effect on the *LoS*-query. However, this difference is minor compared to the difference between transitioning from the `byte[]` to the `short[]`. As such, the `short` data type provides enough precision to encode a viewshed, and offers a reduction in memory usage of a factor of 4 with respect to a 64-bit `double`. Additionally, testing was performed on the latency induced by this simplistic method for encoding and decoding the angles to and from the compressed viewshed, but no significant enough impact was measured.

6.4 Collision detection

RQ9: What are the performance and accuracy effects of exact mesh-based collision detection versus exclusively calculating collision detection of the enveloping sphere?

During this experiment, the viability of two different approximation methods for the purpose of collision detection were researched. In particular, it was of interest to determine how close the behavior of a detection model consisting of an enveloping sphere around the vertices of a 3d model can come to exact mesh-based collision detection. To test this, a 'peek' manoeuvre was performed behind a mountain at a large distance, and a sensor was simulated to detect the peeking object. The shape of the object was chosen such that it portrays a worst-case scenario; an elongated model which does not fit a sphere and therefore creates a significant amount of negative space ($\approx$ 80 percent). This negative space caused the enveloping sphere to be detected first, as is visible in figure 5.16. Here, the 3d model of a plane moves upward as the elevation value of the x-axis increases. For elevation $\Delta h = 9$m, the probability $P(detect)$ increases, and the plane is reliably detected at $\Delta h \geqslant 10$m. For the original, mesh-based model, this process occurs for $\Delta h \geqslant 22$m. The elevation difference between the first moment of detection (i.e., where $P(detect) > 0$ holds), is approximately 12m; the amount of vertical space between the top of the 3d model and the top of the enveloping sphere. Depending on the distance between the observer and the entity, the elevation difference between the first moment of detection can correspond to an angle within a predefined margin of error. For example, at 16km, this difference in angle corresponds to roughly $\Delta\theta = \arctan\left\{\frac{12}{16\times10^3}\right\} = 0.04^\circ$. The error induced by using a viewshed or different CRS overshadows this minor angular approximation. For entities close to the observer, an alternative must be defined.

Additionally, a bounding box model was proposed to approximate the shape of the 3d model. This bounding box contains a constant number of faces for any shape it contains, and is therefore reliable in terms of estimated computational resources. Figure 5.16 shows near-identical behavior of the bounding box model and the original 3d model. However, the bounding box model is more expensive to compute collision detection for, as is indicative of figure 5.17. In a scenario where entities are located close to the observer, these additional expenses are worth the extra fidelity in order to not obfuscate too much of the view. Ideally, an initial computationally inexpensive task of performing collision detection with the enveloping sphere is performed before considering computing collision detection on the bounding box. If the latter detection model returns *false*, the bounding box model can be omitted. Due to the bounding box model performing well for the scenario illustrated in figure 5.16, it is considered to be a viable alternative to mesh-based collision detection at the medium to long-range. At the short range, the differences between the bounding box and exact mesh-based collision detection become more apparent. In this range, a simplified 3d model of the original mesh can be used for this purpose. Algorithms to reduce the complexity of a 3d model exist in literature [15]. As a result, the total error is then not only dependent on the distance to the entity, but also on the quality of the mesh approximation of the model. Based on the correctness of the mesh approximation, it can be decided the use the bounding box model instead due to its constant and predictable performance.

6.5 Radar detection model

RQ10: What is an appropriate SNR threshold to model the unique effects of radar propagation such as refraction and ducting?

In order to incorporate radar-like behavior such as ducting and waves curving behind peaks without embedding confidential or otherwise sensitive radar characteristics in a detection model, it was chosen to model radar propagation using the parabolic equation (equation B.14) and to interpret the SNR using a placeholder threshold-based detection model (equation 2.8). This model created a set of interesting images, shown in figures 5.18a - 5.18c, which display this characteristic radar behavior. In figure 5.18a, a realistic atmosphere was introduced, and the effects of knife-edging (curving behind a mountain peak) occurs as a result. The $\text{SNR}_{threshold}$ at which this occurred is between $\text{SNR}_{threshold} = 0.25$ (figure 5.19c) and $\text{SNR}_{threshold} = 0.1$ (figure 5.19d). As such, for this scenario (the atmospheric conditions, radar parameters) a $\text{SNR}_{threshold}$ of $0.1 \leqslant \text{SNR}_{threshold} \leqslant 0.25$ suffices for the stipulated set of requirements. Furthermore, in order to push the effect of an increased range induced by radar ducting to its limit, a scenario on a flat surface was considered in figures 5.18b and 5.18c. In both figures all radar parameters are identical, the only difference being the atmospheric conditions; the former sensor being situated in a monotonic atmosphere, and the latter sensor in a atmosphere containing an elevated duct. This duct acts as a wave-guide for the radar waves, carrying them along the surface of the Earth and extending the maximum range of the radar sensor. Depending on the exact atmospheric conditions of a given scenario, and the detection sensitivity, either of the proposed values of $\text{SNR}_{threshold} = 0.25$ (figure 5.20b), $\text{SNR}_{threshold} = 0.1$ (figure 5.20d) or $\text{SNR}_{threshold} = 0.05$ (figure 5.20f) can be applied.

For the naval use-case presented in this paper, it is interesting to study the effects of this extended range due to atmospheric conditions. However, this method is paired with increased resource usage; up to a total of 20 seconds for one long-range query for a target at a distance of 125km. This model does not meet the real-time requirements, and can therefore not be considered for usage in the *LoS*-service. The performance of the algorithm can be improved by decreasing the sampling rate of the terrain and increasing the paraxial time step Δx for the PE equation. However, this negatively affects the fidelity of the model due to sections of terrain not being correctly accounted for, negating the prospects of the PE model over the electro-optical vacuum model. Alternatively, the Parabolic Equation model can be implemented and compiled in a low-level programming language such as C or C++, thereby allowing usage of high-performance *Fast Fourier Transform* (*FFT*) libraries. This *FFT* operation concerns a significant part of the computation cost in the algorithm, as is indicative of equation B.14. This can potentially decrease the computation time of short to medium-range queries to within reach of the real-time requirements.

Chapter 7

Discussion

The contents of this thesis concerned the answering of a set of research question and the delivery of a proof-of-concept *Line of Sight*-service. Using this service, a set of experiments were conducted to test the research questions of this paper. During the development of this proof-of-concept alongside speaking with experts, the initially stipulated set of research questions were iteratively scoped and reformulated in order to better fit the use-case of a naval simulation environment. In particular, achieving accurate and performant *LoS*-queries on a grid-based DEM whilst accounting for the curvature of the Earth was the first priority. In order to solve this problem statement, several alternative approaches which were not put forward in this paper were considered as well. For example, a model where the elevation data is projected to a planar surface in order to embed the curvature of the Earth into the grid data itself. The spherical projection method instead attempted to utilize the simplistic geometry of a perfectly spherical Earth in order to compute *LoS*. However, as the Earth is not a perfect sphere, the elevation data had to be projected to a spherical CRS first. The resulting *LoS*-service had very few incorrect queries, but some results presented were unexpected. The solution found in literature; fitting a function to account for the curvature of the Earth, provided a more reliable method instead.

In the process of implementing the viewshed algorithms, the underlying logic of handling the grid data was modified and rewritten multiple times. Low-level control over the grid data structure used in the *LoS*-service allowed for more granular control over several factors such as the logic for interpolation and optimally sampling grid points based on grid alignment. The latter optimization is a method to reduce the resources required to interpolate values by only considering data points which are aligned to either the horizontal or vertical axis when performing an *LoS*-query. This omits the requirement to interpolate in two dimensions, as the point is already aligned to one of the two grid lines. Furthermore, the *R2* and *XDraw* algorithms were implemented and found to be relatively imprecise compared to *R3*. In practise, this imprecision is likely the result from the algorithms sampling different elevation values as a result of interpolating different continuous locations on the grid. This is caused by the terrain being very mountainous. Either a more flat or more high-resolution DEM negates these negative effects. In the naval use-case where high gradient terrain is absent in a radius around the ship, the errors presented in the results are likely not a common occurrence.

On the subject of collision detection, two methods for approximating the shape of a 3d model were studied; the enveloping sphere and bounding box model. Initial recommendations around this subject were to develop an Octree data structure, where the locations of objects in the environment

were stored in a recursive data format. However, due to time limitations it was decided to not include this in the *LoS*-service as of now. The initial problem was related to whether to encode the 3-dimensional ECEF location or the 2-dimensional coordinate position. The collision detection logic relied on the ECEF positions of the models, as the sight-lines propagate in 3-dimensional space. Conversely, all locations and queries were recorded in their coordinate form. Furthermore, upon altering the position of an entity, its relation to the Octree must be re-calculated. Therefore, in a scenario containing one observer and moving entities, there was no quantified advantage to using the Octree in this stage of development. Instead, a reliable solution for the collision detection problem statement was initially testing for collision with the enveloping sphere, and then the bounding box and/or exact mesh model. For the naval context, the bounding box model suffices for the purpose of collision detection. Additionally, the sensor size constant enables limiting the maximal view distance of the electro-optical sensor by defining the minimal required size of an object at a given distance.

The mathematics of the radar propagation physics introduced a steep learning curve for the subject. Previous work and existing tools therefore provided a good starting point in understanding the mechanisms behind the radar Split-Step Parabolic Equation model (equation B.14), and enabled to development of a solution in Java. During the development process, the final key step was to interpret a value for the SNR (signal-to-noise ratio) as a *true* or *false*. Discussing this with experts on the field of radar propagation modeling yielded the conclusion that the development of such a detection model would exceed the scope of the thesis due to the required expertise and insights into radar characteristics. Instead, it was chosen to create a placeholder model which resulted in satisfying results for the presented use-case. In any future implementations or spin-offs, a high-fidelity detection model is of the essence in order to produce reliable results true to the physical behavior of a radar system. However, the radar propagation model remained relatively slow, and potential latency improvements will likely not decrease the computation time of long-range radar *LoS*-queries to an order of magnitude considered (near) real-time. As such, either a different model must be considered, or tolerance for a higher latency must be accepted into the system requirements if the radar behavior must be modeled as accurately as possible.

The *LoS*-service developed as a result of the research conducted in this paper allows the creation of an API-endpoint, to which can be queried whether two points and/or entities are within *Line of Sight*. In a simulation environment, such an API can determine whether an entity is able to detect and therefore should react to an object or entity in its surroundings. Furthermore, collecting the results of a set of *LoS*-queries using the radar model enables the construction of a so-called 'radar display' for the purpose of human interaction.

Priorities and viability ultimately determined several potential research questions to not be applicable to the presented use-case. For example, the ray tracing model was developed but ultimately not studied due to initial simulations not presenting satisfying results. The atmospheric effects are accounted for using Snell's law, but these effects were negligible in the current implementation and the model was relatively computationally expensive. As such, more effort was put into the electro-optical vacuum model and parabolic equation radar model. Future work can focus more on a viable ray tracing implementation, potentially utilizing the efficiency of the GPU for this purpose. Furthermore, as was described in an earlier chapter, GPU usage was omitted during the research process due to requirements of the use-case; determining *LoS* using the CPU is possible according to existing research, and provides a multitude of advantages over *LoS* on the GPU. Software services running on the CPU are more easily deployed, and can utilize CPU cycles of existing infrastructure instead of requiring a dedicated (potentially expensive and resource-intensive) graphics card. In real-world use-cases where the size of a system must be taken into account, such

as on a marine ship, a CPU-only service is advantageous as well due to its much smaller space- and energy footprint. In future research, the alternative of approaching the problem on a GPU can be studied more extensively, and can potentially be compared to the CPU-only approach in terms of performance and cost.

General recommendations for an *LoS* API

From the conclusion and discussion of this paper, a set of general recommendations can be stipulated for the purpose of defining a reliable *Line of Sight*-service. For one, the Parabolic Equation radar model as it is currently implemented does not meet the current scenario's needs; the signal-to-noise ratio resulting from the model is yet to be translate either to a binary *true/false* or a probability of detection using a valid detection model. Additionally, at the longer ranges the model struggles to meet real-time requirements without under-sampling the terrain. Stronger hardware or possibly implementing the algorithm efficiently on a GPU can potentially alleviate this issue. Geometric *LoS* allows a simplistic model of reality to perform queries in sub-millisecond latency, and does suffice if accounting for atmospheric effects and ducting can be omitted. Furthermore, this model allows for granular control over calculating detection probability in the form of sampling a surface tangential to the sight-line. This way, because the individual queries are highly performant, the surface which is visible to the observer from the point of view of the observer can be approximated reliably.

Viewsheds yield interesting advantages over individual queries for static observers, and as such can be utilized to their full strengths when simulating a radar system which is installed on a vantage point. For a moving ship, depending on the required fidelity and the gradient of the surrounding terrain, re-using a viewshed calculation of 90 meters away remains a valid option. Assuming a ship does not exceed 100 km/h, a time interval of approximately 3 seconds is available before a next viewshed is required, which (depending on the maximum range) is possible according to the measurements presented in this paper. The advantages of computing the full viewshed over performing individual queries are: *a)* only a single query to an *LoS* API is required and then cached and *b)* the system can account for an arbitrary number of entities as all possible answers to future queries are embedded in the viewshed grid.

For the purpose of modeling the entities in the environment for collision detection, three solutions were proposed. The most performant method, the enveloping sphere, can serve as an initial check to separate potential candidate entities from entities which are certain to not intersect the sight-line. Then, depending on the number of meshes and available computing resources, either the bounding box model or the 3d mesh will be checked for intersection. The advantage of the former method is its reliable scaling, as the resources required to compute for collision detection do not depend on the complexity of the entity itself. However, at closer distances, the marine ships' bridge (which extends vertically above all other parts of the ship), can cause false positives when calculating whether the ship is in *Line of Sight*. Therefore, at closer distances, a simplified mesh model may be preferred over both the enveloping sphere and bounding box model. Alternatively, a distance-dependent mesh where the fidelity decreases as a function of distance from the observer may attenuate the aforementioned issues.

Acknowledgements

First and foremost, I would like to express my gratitude to my daily supervisor Ir. Harm Jan Spier (Thales) and my supervisor at the university dr. Tom van Dijk (University of Twente). Their guidance helped shape the contours of this thesis, which I had the opportunity to fill in during a six-month period. During this time, I had the pleasure to work on two subjects close to my heart and interests; computer science and physics, which I was able to incorporate into a thesis on the subject of *Line of Sight* in a simulated environment. Moreover, I would like to thank dr. Nicola Strisciuglio for his valuable feedback during the final period of writing this report, and for partaking in the committee overseeing my masters thesis.

Furthermore, the openness and willingness to aid and answer questions related to the subject from employees at Thales was invaluable to the course of this study. I was given appropriate tooling and connections to experts on the fields of computer science and physics, where discussions felt like colleague-to-colleague interactions.

Bibliography

[1] Shapefiles—ArcGIS online help | documentation. https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm. Accessed: 2024-02-16.

[2] J.B. Anderson. *Digital Transmission Engineering*. IEEE Series on Digital & Mobile Communication. Wiley, 2006. URL: https://books.google.nl/books?id=VW7_F8qBsEUC.

[3] L. W. Austin. Preliminary note on proposed changes in the constants of the austin-cohen transmission formula. *Journal of the Washington Academy of Sciences*, 16(8):228–231, 1926. URL: http://www.jstor.org/stable/24529248.

[4] P. K. Bondyopadhyay. Sir J.C. Bose diode detector received Marconi's first transatlantic wireless signal of December 1901 (the "Italian Navy Coherer" Scandal Revisited), January 1998. doi:10.1109/5.658778.

[5] B. R. Bowring. Transformation from spatial to geographical coordinates. *Survey Review*, 23(181):323–327, 1976. arXiv:https://doi.org/10.1179/sre.1976.23.181.323, doi:10.1179/sre.1976.23.181.323.

[6] Aran J. Cauchi-Saunders and Ian J. Lewis. Gpu enabled xdraw viewshed analysis. *Journal of Parallel and Distributed Computing*, 84:87–93, 2015. URL: https://www.sciencedirect.com/science/article/pii/S0743731515001197, doi:10.1016/j.jpdc.2015.07.001.

[7] Fang Chao, Yang Chongjun, Chen Zhuo, Yao Xiaojing, and Guo Hantao. Parallel algorithm for viewshed analysis on a modern gpu. *International Journal of Digital Earth*, 4(6):471–486, 2011. arXiv:https://doi.org/10.1080/17538947.2011.555565, doi:10.1080/17538947.2011.555565.

[8] Ghassan Dahman, François Gagnon, and Gwenael Poitau. Ship-to-ship beyond line-of-sight communications: A comparison between ray tracing simulations and the petool. In *2017 XXXIInd General Assembly and Scientific Symposium of the International Union of Radio Science (URSI GASS)*, pages 1–4, 2017. doi:10.23919/URSIGASS.2017.8105400.

[9] Junduo Dong and Jianbo Zhang. A multi-level distributed computing approach to xdraw viewshed analysis using apache spark. *Remote Sensing*, 15(3), 2023. URL: https://www.mdpi.com/2072-4292/15/3/761, doi:10.3390/rs15030761.

[10] Wanfeng Dou and Yanan Li. A fault-tolerant computing method for xdraw parallel algorithm. *The Journal of Supercomputing*, 74, 06 2018. doi:10.1007/s11227-018-2321-x.

[11] John E Greivenkamp. Field guide to geometrical optics. SPIE Bellingham, WA, 2004.

[12] H.V. Hitney. Refractive effects from VHF to EHF. Part A: Propagation Mechanisms. In *In AGARD*, September 1994.

[13] Gerard Kunz, Eric Heemskerk, and Lex van Eijk. Comparison of atmospheric refraction at radar and optical wavelengths. In Karin Stein and Anton Kohnle, editors, *Optics in Atmospheric Propagation and Adaptive Systems VIII*, volume 5981, page 59810B. International Society for Optics and Photonics, SPIE, 2005. doi:10.1117/12.638613.

[14] Mireille Levy. *Parabolic equation methods for electromagnetic wave propagation*. Number 45. IET, 2000.

[15] Yaqian Liang, Fazhi He, and Xiantao Zeng. 3d mesh simplification with feature preservation based on whale optimization algorithm and differential evolution. *Integrated Computer-Aided Engineering*, 27(4):417–435, 2020.

[16] MathWorks. Ecef position to lla. https://www.mathworks.com/help/aeroblks/ecefpositiontolla.html, 2023. Accessed: 2023-09-11.

[17] MathWorks. Lla to ecef position. https://www.mathworks.com/help/aeroblks/llatoecefposition.html, 2023. Accessed: 2023-09-11.

[18] James C Maxwell. *A dynamical theory of the electromagnetic field*. Wipf and Stock Publishers, 1996.

[19] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, pages 7–es. 2005.

[20] Ozlem Ozgun, Gökhan Apaydin, Mustafa Kuzuoglu, and Levent Sevgi. Petool: Matlab-based one-way and two-way split-step parabolic equation tool for radiowave propagation over variable terrain. *Computer Physics Communications*, 182(12):2638–2654, 2011. URL: https://www.sciencedirect.com/science/article/pii/S0010465511002669, doi:10.1016/j.cpc.2011.07.017.

[21] Timothy M Pritchett. *Spectral solution of the Helmholtz and paraxial wave equations and classical diffraction formulae*. Army Research Laboratory, 2004.

[22] Mirabel Cerqueira Rezende, Inácio Malmonge Martin, Roselena Faez, Marcelo Alexandre Souza Miacci, and Evandro Luıs Nohara. Radar cross section measurements (8-12 ghz) of magnetic and dielectric microwave absorbing thin sheets. *Revista de Fısica Aplicada e Instrumentaçao*, 15(1):24–29, 2002.

[23] Vivek Saxena and Navjit Kaur. Comparison of r2 and r3 approaches over cpu vs. gpu implementation. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 377–379, 2022. doi:10.23919/INDIACom54597.2022.9763106.

[24] David J Thomson and NR Chapman. A wide-angle split-step algorithm for the parabolic equation. *The Journal of the Acoustical Society of America*, 74(6):1848–1854, 1983.

[25] Alpaslan Türkboyları. Radar propagation modelling using the split step parabolic equation method. Master's thesis, Middle East Technical University, 2004.

[26] Birgit Wessel, Martin Huber, Christian Wohlfart, Ursula Marschalk, Detlev Kosmann, and Achim Roth. Accuracy assessment of the global tandem-x digital elevation model with gps data. *ISPRS Journal of Photogrammetry and Remote Sensing*,

139:171–182, 2018. URL: https://www.sciencedirect.com/science/article/pii/S0924271618300522, doi:10.1016/j.isprsjprs.2018.02.017.

[27] Christian Wolff. Radar cross section. https://www.radartutorial.eu/01.basics/Radar%20Cross%20Section.en.html, 2023. Accessed: 2023-10-19.

Appendix A

Perfectly spherical Earth approximation assessment

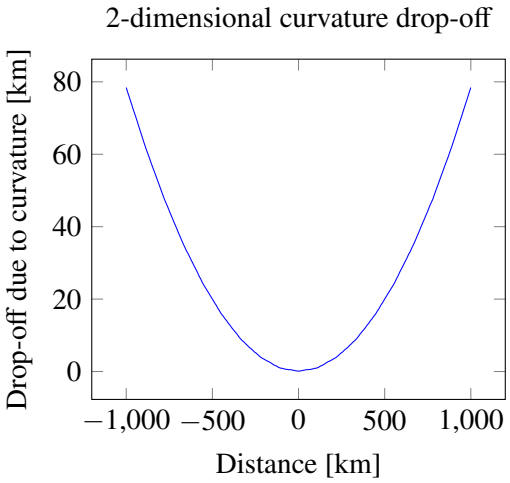
Depending on the use-cases and range, approximating the shape of the Earth as a perfect sphere can yield satisfying results when calculating the curvature of the Earth. When calculating an average radius, it is beneficial to weigh the equatorial radius more than the polar radius. Given constants $a = 6378137$m and $b = 6356752.314245$m of the WGS84 ellipsoid, this would result in an average radius of $\frac{2a+b}{3} = \frac{2\times6378137+6356752.314245}{3} \approx 6371000$m. Weighing the equatorial radius twice positively influences the accuracy near the equator, and negatively near the North and South pole. However, heuristically, most queries are located nearer to the equator than to the North or South pole. In figure A.1a, a 2-dimensional representation of the spherical drop-off has been plotted. The further away a target on the spherical Earth's surface would be, the lower it would be from the perspective of the observer. A similar representation can be viewed in figure A.1b, where a 2-dimensional plane contains the drop-off when moving in either of the two dimensions. Due to the spherical nature of this approximation, this layer is identical to any position the observer can be positioned, and therefore this representation only has to be pre-computed once. Superimposing this layer upon a DEM containing elevation data results in an approximation of the effect of the curvature of the Earth. As a result, on a spherical Earth approximation, a line which is pointing downwards from a high position (figure A.1c), would appear to curve upward when accounting for the curvature of the Earth (figure A.1d).

The error introduced by approximating the ellipsoidal shape of the Earth by a perfect sphere is shown in figures A.2a - A.2c. The difference between the radius of the Earth on the ellipsoidal and spherical representation varies as a function of the angle between the North pole and the point of interest. At $\alpha \approx 36^\circ$ the two models intersect, this can be viewed in figure A.2a. The difference in drop-off caused by the approximation has been visualized for two locations on Earth; for the North pole (figure A.2b) and the Equator (figure A.2c). In the former figure, the error increases as a function of displacement, then briefly decreases due to the intersection of the spherical and ellipsoidal models before it increases again. In the latter, the error slowly increases over time as the difference between the radius at the equator between the two models is constant. Therefore, the two models never intersect. To conclude; for a distance along the surface corresponding to an angle $\alpha \leqslant 10^\circ$ the error introduced by the spherical approximation is very small ($\approx 50m$) with respect to the total distance travelled. At the equator, a longitudinal angle of 10° corresponds with a distance of $10\frac{2\pi r}{360} = 10\frac{2\times\pi\times6371000}{360} \approx 1112000$m. Therefore, when considering a radius within this bound, the spherical Earth approximation can be considered as a viable alternative.

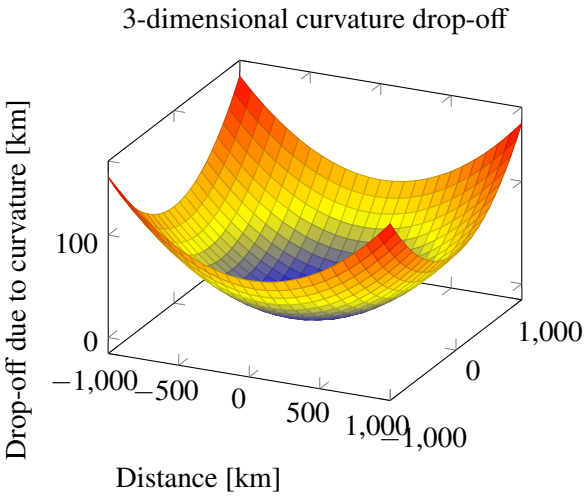
In a spherical reference system, the angle between an observer and a target can be calculated using formula A.1.

$$\theta_{observer \to target} = \arctan\left(\frac{r_1 \sin\alpha}{r_0 - r_1 \cos\alpha}\right) \tag{A.1}$$

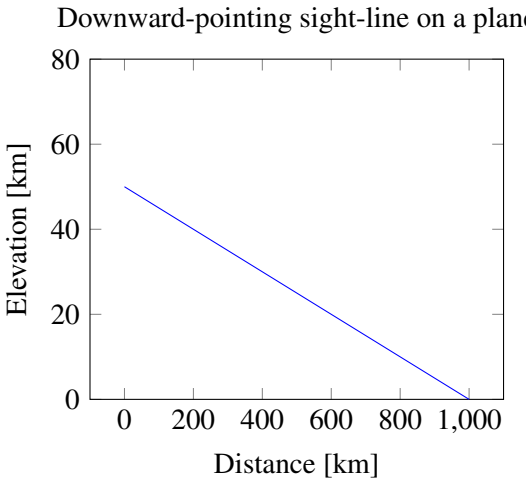
Here, α is the angle on a great circle between the observer and target positions, r_1 is the distance from the target to the centre of the Earth and r_0 is the distance from the observer to the centre of the Earth.



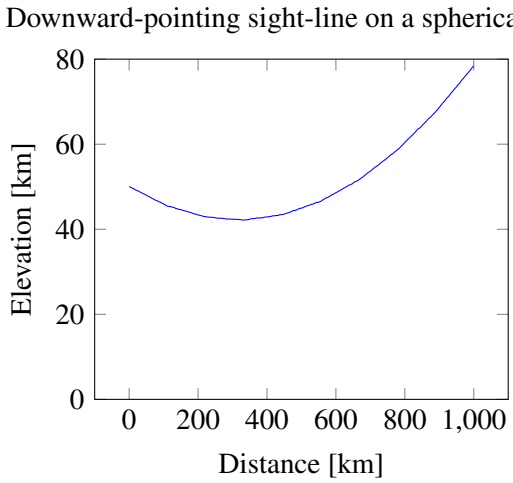
(A) A 2-dimensional representation of the drop-off introduced by the curved nature of the Earth along a straight line.



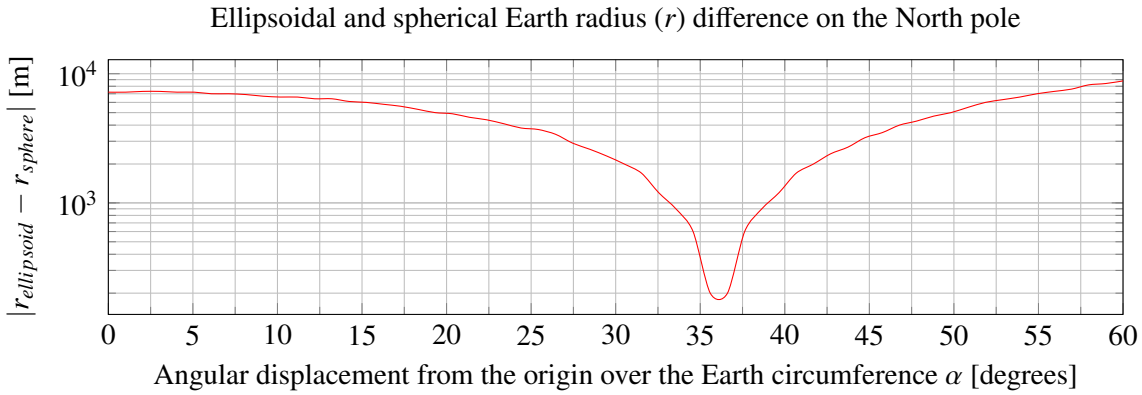
(B) A 3-dimensional representation of the drop-off introduced by the curved nature of the Earth.



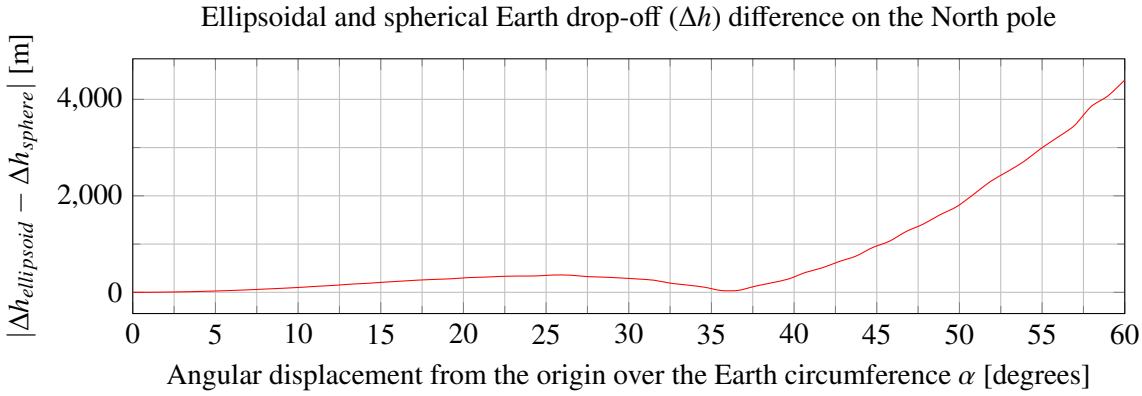
(C) An illustration of a sightline starting at $y = 50km$ with a negative slope, when approximating the Earth as a plane. When plotting the distance between the sightline and the plane, the result is a straight line.



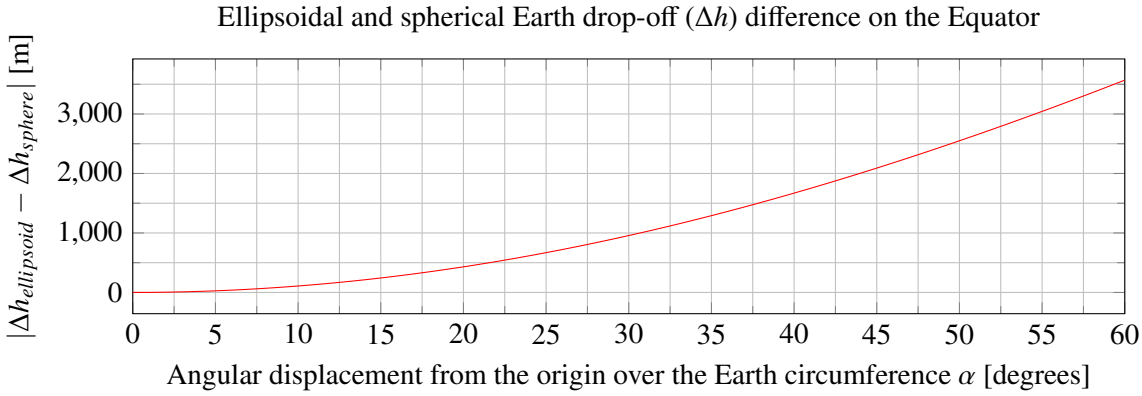
(D) Taking into account the curvature of the Earth, the sight-line from A.1c will curve up due to the curvature of the Earth when plotting the height relative to the Earth's surface.



(A) The difference in radius between the ellipsoidal and spherical models on the North pole, the region with the largest offset. The difference is minimal at angle $\alpha \approx 36^\circ$, where the two representations intersect.



(B) The difference in the drop-off between considering a uniform spherical drop-off versus an exact ellipsoidal drop-off on the North pole. The difference is minimal at small angles $\alpha \leqslant \epsilon$ for small ϵ and angle $\alpha \approx 36^\circ$, where the two representations intersect.



(C) The difference in the drop-off between considering a uniform spherical drop-off versus an exact ellipsoidal drop-off on the Equator. The error steadily increases as a function of the angle α.

FIGURE A.2: The difference in drop-off for an ellipsoidal model of the surface of the Earth and a spherical approximation for several points on the surface of the Earth. On the equator, the error steadily increases as a function of distance, whilst for observers on the North or South pole a local minimum at $d \approx 36$km arises due to an intersection between the ellipsoidal model and the spherical approximation.

Appendix B

Derivation of the Split-Step Parabolic Equation Radar propagation model

In order to derive the solution to the split-step parabolic equation for wave propagation, Maxwell's equations (equation B.1 [18]) are used as a starting point.

$$\nabla \cdot \vec{E} = \rho/\epsilon_0 \tag{B.1a}$$
$$\nabla \cdot \vec{B} = 0 \tag{B.1b}$$
$$\nabla \times \vec{E} = -\frac{\delta \vec{B}}{\delta t} \tag{B.1c}$$
$$\nabla \times \vec{B} = \mu_0(\vec{J} + \epsilon_0 \frac{\delta \vec{E}}{\delta t}) \tag{B.1d}$$

Here, $\vec{E}$ is the electric field, $\vec{B}$ the magnetic field and $\vec{J}$ the current density. Constants ρ is the electric charge density, and ϵ_0 and μ_0 are permittivity $[F \cdot m^{-1}]$ and permeability $[kg \cdot m \cdot s^{-2} \cdot A^{-2}]$ in a vacuum respectively.

To describe the propagation of an electro-magnetic wave, starting from Maxwell's equations (B.1) for charge-free space ($\rho = 0$, $\vec{J} = 0$) to form equations B.2.

$$\nabla \cdot \vec{E} = 0 \tag{B.2a}$$
$$\nabla \cdot \vec{B} = 0 \tag{B.2b}$$
$$\nabla \times \vec{E} = -\frac{\delta \vec{B}}{\delta t} \tag{B.2c}$$
$$\nabla \times \vec{B} = \mu_0 \epsilon_0 \frac{\delta \vec{E}}{\delta t} \tag{B.2d}$$

Taking the curl of $\nabla \times \vec{E}$ and $\nabla \times \vec{B}$ in equation B.2 and re-substituting the equations for $\nabla \times \vec{E}$ and $\nabla \times \vec{B}$ of the resulting right-hand side of the equation results in equation B.3.

$$\nabla \times (\nabla \times \vec{E}) = -\mu_0 \epsilon_0 \frac{\delta^2 \vec{E}}{\delta t^2} \tag{B.3a}$$

$$\nabla \times (\nabla \times \vec{B}) = -\mu_0 \epsilon_0 \frac{\delta^2 \vec{B}}{\delta t^2} \tag{B.3b}$$

Then, using the mathematical equation for the curl of a curl (equation B.4), and substituting equations $\nabla \cdot \vec{E} = 0$ and $\nabla \cdot \vec{B} = 0$ of equation B.2 in equation B.4 results in equations B.5.

$$\nabla \times (\nabla \times \vec{A}) = \nabla(\nabla \cdot \vec{A}) - \nabla^2 \vec{A} \tag{B.4}$$

$$\mu_0 \epsilon_0 \frac{\delta^2 \vec{E}}{\delta t^2} - \nabla^2 \vec{E} = 0 \tag{B.5a}$$

$$\mu_0 \epsilon_0 \frac{\delta^2 \vec{B}}{\delta t^2} - \nabla^2 \vec{B} = 0 \tag{B.5b}$$

Substituting the speed of light c for $\frac{1}{\sqrt{\mu_0 \epsilon_0}}$ results in equation B.6.

$$\frac{1}{c^2} \frac{\delta^2 \vec{E}}{\delta t^2} - \nabla^2 \vec{E} = 0 \tag{B.6a}$$

$$\frac{1}{c^2} \frac{\delta^2 \vec{B}}{\delta t^2} - \nabla^2 \vec{B} = 0 \tag{B.6b}$$

Therefore, for a field u, a partial differential equation coined the 'wave equation' arises (equation B.7).

$$\frac{\delta^2 u}{\delta t^2} = c^2 \left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} + \frac{\delta^2 u}{\delta z^2} \right) \tag{B.7}$$

Equation B.7 is the scalar wave equation, sometimes denoted as $\Box u = 0$ where $\Box$ denotes the d'Alembert operator $\frac{1}{c^2} \frac{\delta}{\delta t^2} - \nabla^2$ or in scalar form $\frac{1}{c^2} \frac{\delta}{\delta t^2} - \frac{\delta}{\delta x^2} - \frac{\delta}{\delta y^2} - \frac{\delta}{\delta z^2}$.

The Helmholtz equation (equation B.8) represents a time-independent variant of the wave equation:

$$\nabla^2 u = -k^2 u \tag{B.8}$$

Here, k is eigenvalue and in the use-case of waves the wave number ($k = \frac{2\pi}{\lambda}$ for wavelength λ). It is obtained by applying the principle of separation of variables to the wave equation.

Furthermore, after obtaining the Helmholtz equation for the time-independent solution to the wave equation, applying the paraxial approximation yields a more applicable form for computer modeling. In the paraxial approximation, the assumption is made that the derivative in the x-direction slowly varies as a function of x: $|\frac{\delta^2 u}{\delta x^2}| \ll |\vec{k}\frac{\delta u}{\delta x}|$. Here, k, is the wave vector and the angle is assumed to be small. The paraxial Helmholtz equation (equation B.9) can be derived by substituting complex amplitude $A(\vec{r}) = u(\vec{r})e^{ikx}$ into equation B.8 and assuming that under the right conditions $\frac{\delta^2 u}{\delta z^2} + \frac{\delta^2 u}{\delta y^2} = 2ik\frac{\delta u}{\delta x} = 0$ [21].

$$\frac{\delta^2 A}{\delta z^2} + \frac{\delta^2 A}{\delta y^2} + 2ik\frac{\delta A}{\delta x} + 2k^2 A = 0 \tag{B.9}$$

Expressing the wave equation in two-dimensional coordinates, by assuming azimuthal symmetry in the vertical axis, allows rewriting the equation B.9 in the 2-dimensional, cylindrical form [25] as equation B.10.

$$\frac{\delta^2 A}{\delta z^2} + 2ik\frac{\delta A}{\delta x} + 2k^2 A = 0 \tag{B.10}$$

Isolating $\frac{\delta A}{\delta x}$ in equation B.10 results in equation B.11.

$$\frac{\delta A}{\delta x} = \frac{-i}{2k}\frac{\delta^2 A}{\delta z^2} + 2ikA \tag{B.11}$$

Computing the Fourier transform of $A(\vec{r})$ results in equation B.12.

$$\frac{\delta U(x,p)}{\delta x} = \frac{-ip^2}{2k}\frac{\delta^2 U(x,p)}{\delta z^2} + 2ikU(x,p) \tag{B.12}$$

Here, $U(x,p) = F\{A(\vec{r})\}$ where F is the Fourier transform and p is the transformation variable $p = k\sin\theta$ where θ is the propagation angle with respect to the horizon [25].

Furthermore, the Fourier Split-Step solution of the Parabolic Wave Equation can be derived by taking the inverse Fourier transformation of both sides for range $x + \Delta x$ by utilizing the rule that the Fourier transformation of a derivative is a multiple of the transformation (equation B.13). Applying this rule to equation B.12 results in equation B.14.

$$\hat{f}'(p) = F\left\{\frac{\delta}{\delta x}f(x)\right\} = i2\pi p\hat{f}(p) \tag{B.13}$$

$$A(x+\Delta x, z) = e^{ik\Delta x}F^{-1}\left\{e^{\frac{-i\Delta x p^2}{k}}F\{A(\vec{r})\}\right\} \tag{B.14}$$

Here, F^{-1} is the inverse Fourier transformation.

Appendix C

Line of Sight-service architecture

This section explains the overall architecture and inner workings of the *LoS*-service. The primary methods containing the sensor behavior are located in the `Sensor` abstract class and its sub-classes, in particular the `ElectroOpticalVacuumSensor` and the `RadarPESensor` in figure C.1. The former class models an electro-optical sensor as if it were operating in a vacuum using geometry. The latter class utilizes the Parabolic Equation method to calculate the SNR (signal-to-noise ratio) for a radar system. Due to a common super-class, both sensors can be queried in a similar way and return an `SNRResultEnvelope`; which contains all relevant information relating to the resulting signal strength. This result can then be applied to a `DetectionModel`, which returns an `LOSResultEnvelope`. The resulting object contains relevant information on the query result such as a binary *true* or *false*, as well as the point of collision with the terrain or an object.

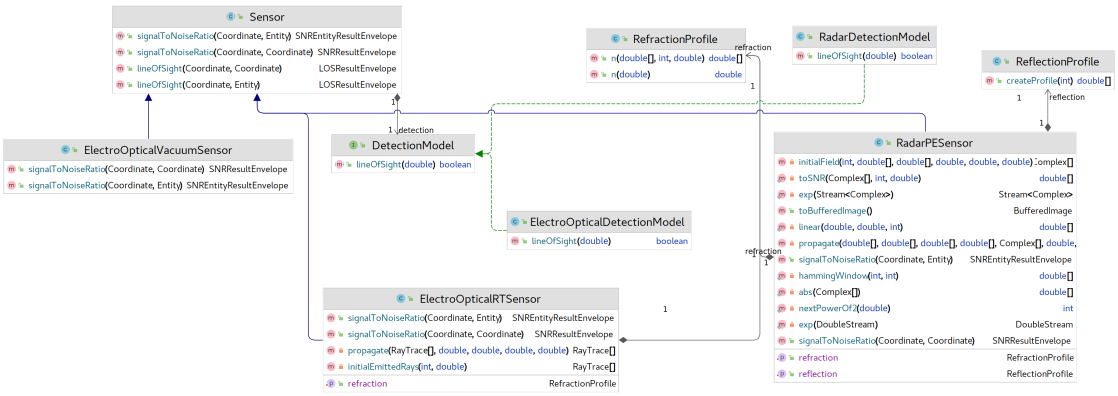


FIGURE C.1: UML class diagram of the `Sensor` abstract class and its related classes.

Electro-Optical Sensor

The electro-optical sensor is modeled as a geometric *LoS*-sensor, assuming a vacuum as its medium. This model has the benefit of being very computationally inexpensive to perform queries on. However, the phenomenon of extended *LoS* beyond the horizon due to refraction in the atmosphere is unaccounted for. Calculating *LoS* consists of a call to the `lineOfSight()` method with two coordinates as arguments; the position of the observer and the target. In this case, the target will be

approximated as a point in space with zero volume, and *LoS* between the two points consists of a binary *true/false* based on whether or not a direct geometric line can be drawn between these two points in 3-dimensional space. Alternatively, an `Entity` object can be passed as an argument, which represents a target with volume. Determining *LoS* between the observer and an `Entity` requires emitting a group of sight-lines to a plane orthogonal to the sight-line between the observer and the centre of the entity. Using a sampling technique to subdivide the plane tangential to the sight-line (indicating the shape of the entity from the point of view of the observer), an estimation of the sensor coverage is calculated based on the samples which intersect the model and samples which are obstructed by terrain. An illustration of this is given in figure 2.8. Furthermore, depending on the required fidelity, exact mesh-based collision detection can be omitted if the system requirements allow for this approximated approach.

Parabolic Equation (PE) Radar Sensor

Performing an *LoS*-query on a PE radar sensor using the `lineOfSight()` method is very similar to the electro-optical sensor, as both classes can be used interchangeably due to their common superclass. However, instead of using geometry, the PE radar sensor utilizes a step-wise marching algorithm towards the target. Querying *LoS* between an observer and a point by passing two coordinates as arguments assumes a default radar cross-section (RCS) of $\sigma = 1\ m^2$, and passing an `Entity` object as second argument for the target can account for a custom value for the RCS. The detection model assumes a step-function as described in formula 2.8, and is to be replaced by a proper detection model for real-world usage due to its heavily simplified nature.

Entity collision detection

In order to perform collision detection queries, the shape of an object has to be denoted as a set of geometric shapes. In principle, all complex shapes can be expressed as a combination of triangles and spheres. Collision detection with either of these shapes is performed using algorithm 1 and formula 2.3 respectively.

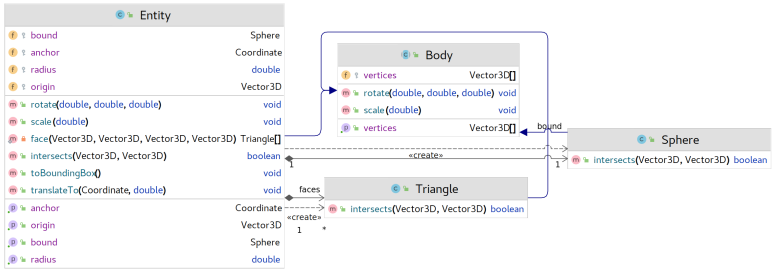


FIGURE C.2: UML class diagram of the `Entity` class and its related classes. All logic associated with collision detection is housed in a `intersects()` method implementation.

Here, the `Triangle` and `Sphere` classes contain the collision detection logic, which is abstracted for all other classes. The `Entity` class determines whether it collides with a geometric

line by first checking for an intersection with its bounding sphere (a sphere containing all vertices within its radius r), and then optionally performing mesh-based collision detection on its faces. Additionally, the `Entity` class contains a method called `toBoundingBox`, which reduces the model to a bounding box of its previously complex shape. This has the effect of reducing a high-polygon model to a model containing (by definition) 12 triangles (6 faces, two triangles for each face); thereby reducing the computational complexity for collision detection. The `Entity` class contains a constructor which is able to parse *Wavefront '.obj'*-files, a file format which expresses 3d models as a set of vertices and faces consisting of previously-defined vertices. To limit the complexity of the developed parser, only orthogonal triangles are supported.

Coordinate Reference Systems

The initial development phase was focused on the foundation of all calculations; correctly accounting for different Coordinate Reference Systems. The current implementation contains a model where a `CoordinateReferenceSystem` abstract super-class contains correct yet computationally expensive algorithms to convert between LLA and ECEF for WGS84, as well as calculating the angle between two coordinates, the distance on a great circle and the distance from a coordinate to the centre of the Earth. Then, a different implementation is able to overwrite these methods with a more candid algorithm based on the advantages the CRS provides. E.g., the spherical CRS allows for computationally inexpensive calculation of the angle between two coordinates using equation A.1. In the current implementation, a `Coordinate` object is always declared to be relative to a CRS, and has this information embedded in a private field. Because of this, all logic for converting to a new CRS can be abstracted and delegated to the `Coordinate` class.

FIGURE C.3: UML class diagram of the `CoordinateReferenceSystem` abstract class and its related classes.

Chart & DEM Grid

The key to correctly handling DEM data is to implement a solid foundation for the underlying grid in the form of an abstraction layer. This way, any grid-related methods can be used in a sub-class without re-implementing similar logic. This class hierarchy is illustrated in figure C.4. For example, the `QueryableGrid` class contains methods to convert between grid indices and coordinates, as well as methods to interpolate between grid points in order to emulate a continuous surface gradient. This interpolation logic is used in `ChartTile` to query the elevation at a coordinate as well as in `XDrawTileViewshed` to interpolate the angle from a previous ring. This modularity increases maintainability, and abstracts mathematics in the low-level classes. In the current implementation, a `Chart` object is a collection of `ChartTile` objects, with its main responsibility to find the tile(s) corresponding to a given coordinate or set of coordinates and delegating the retrieval of elevation data to the correct tile. The `interpolate()` method in `Chart` creates an array of coordinates with elevation data based on the grid resolution of every respective tile it passes. Then, sampling is performed in alignment with the grid in order to limit the complexity of interpolation to linear or cubic interpolation as opposed to bi-linear or bi-cubic interpolation.

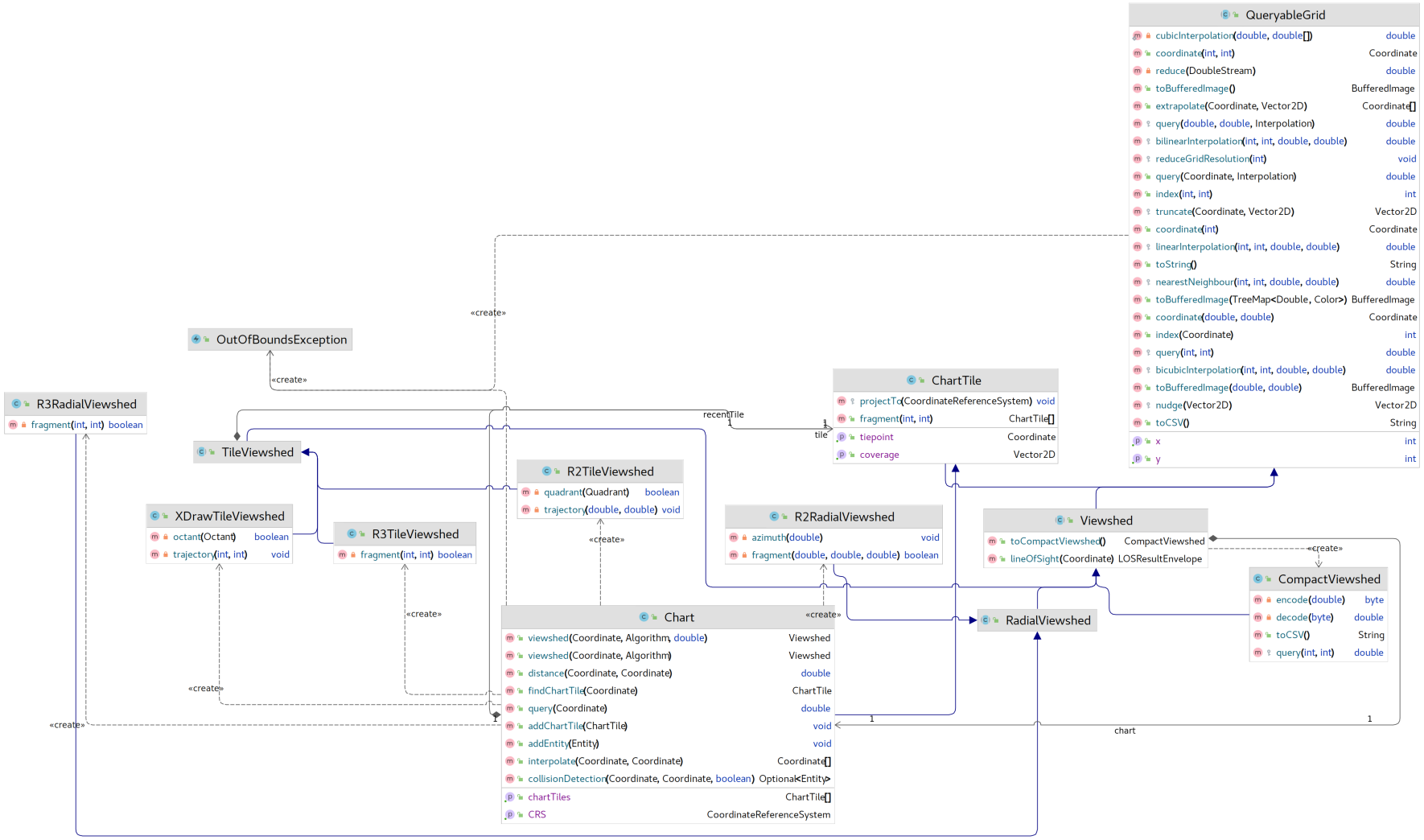


FIGURE C.4: UML class diagram of the `Chart` class and its related classes. Most notably the `QueryableGrid` class is where all of the grid-related calculations take place, the chart and all viewshed implementations all heavily rely on this class. For example, the logic for querying of elevation data, translating a coordinate to a grid location and interpolation methods are situated here.