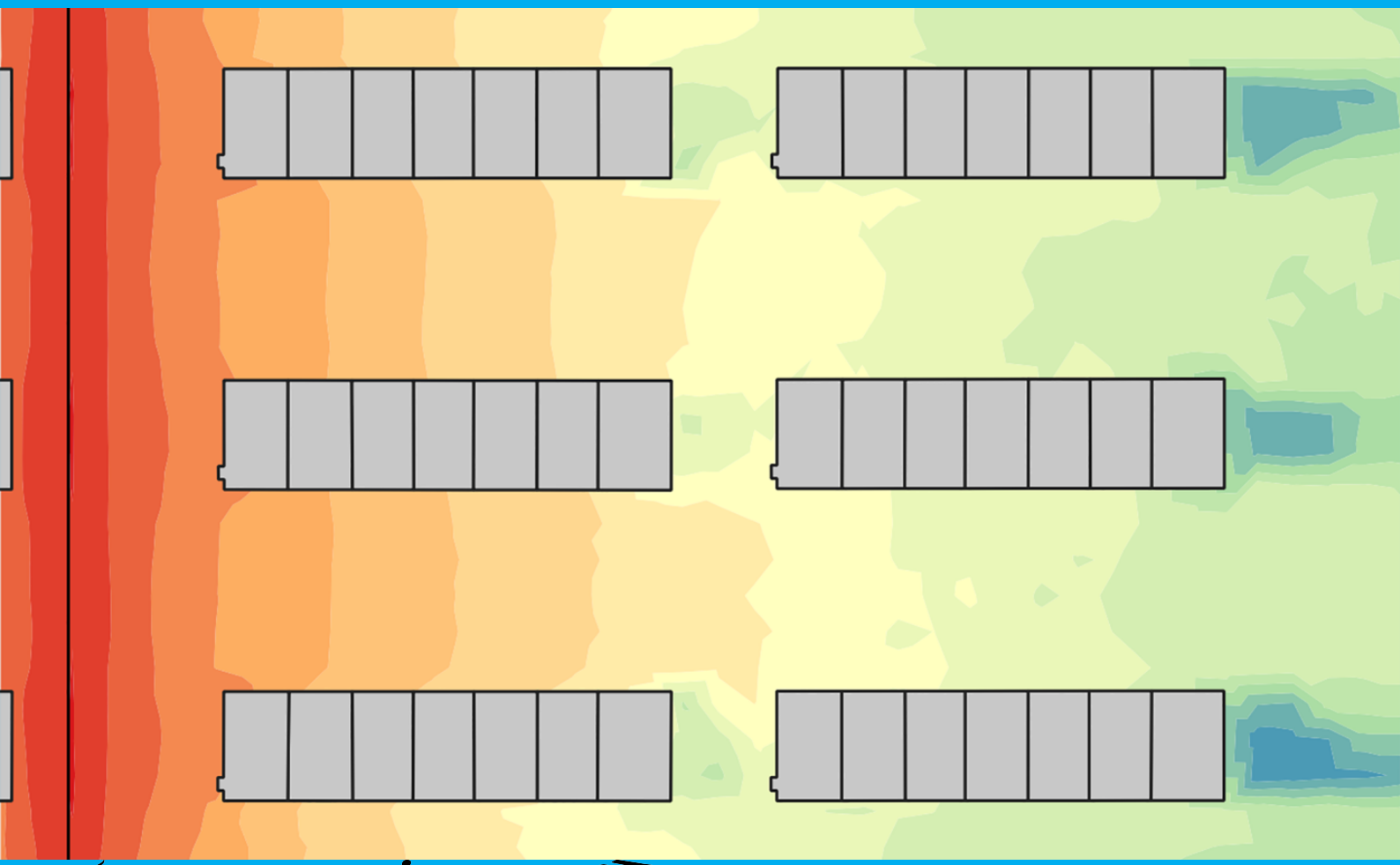
3D noise simulation

Final report

Constantijn Dinklo
Denis Giannelli
Laurens van Rijssel
Maarit Prusti
Nadine Hobeika

Synthesis project 2020
Master Geomatics, Faculty of Architecture and the Built Environment



TU Delft
rivm National Institute for Public Health and the Environment
Rijkswaterstaat

3D noise simulation

Final report

by

Constantijn Dinklo
Denis Giannelli
Laurens van Rijssel
Maarit Prusti
Nadine Hobeika

Submitted on June 23rd 2020, public document

Project duration:	April 20, 2020 – June 26, 2020	
Supervisors:	Prof. dr. ir. J. Stoter,	TU Delft
	Ir. B. Dukai,	TU Delft
Clients:	Dr. A. Kok	RIVM
	Dr. R. van Loon	RIVM
	Ir. R. Nota,	RWS

An electronic version of this report is available at `http://repository.tudelft.nl/`.

TU Delft rivm National Institute for Public Health and the Environment Rijkswaterstaat

Preface

Constantijn Dinklo
Denis Giannelli
Laurens van Rijssel
Maarit Prusti
Nadine Hobeika
June 2020

In March 2020, several projects were offered to the students in the course Synthesis Project, part of the MSc Geomatics at TU Delft to work on in the last period of the academic year. All the group members in this project chose to participate in the 3D noise simulation project. As a group, we have various backgrounds ranging from computer science to architecture. Although we have been working as a group, each group member has had individual tasks.

The final report will show the steps that are taken, the choices that are made, and the results of our Synthesis Project. This final report is written for our clients the Directorate-General for Public Works and Water Management (RWS) and the National Institute for Public Health and the Environment (RIVM), represented by Renez Nota, Arnoud Kok and Rob van Loon. Besides, it is written for our supervisors from the TU Delft, Jantien Stoter and Balázs Dukai. Finally, it is written for the whole noise modelling community.

In 2017, RWS and RIVM approached the TU Delft for a more efficient, cheaper, and standardised way to generate input data for noise simulation. After three years of developing several approaches based on height lines, it was concluded that another approach could be more efficient, namely an approach based directly on a Triangulated Irregular Network (TIN). The time frame of this project is from the 20th of April 2020 until the 26th Of June 2020.

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 About the stakeholders . . . 1
- 1.3 Current state of sound propagation in noise simulation . . . 2
- 1.4 Problem definition . . . 2
- 1.5 Goal . . . 2
- 1.6 Scope of this Project . . . 2
- 1.7 Requirements (MoSCoW) . . . 3
- 1.8 Expected Results . . . 3

2 Research Method **4**
- 2.1 Method . . . 4
- 2.2 Technical Procedure . . . 4

3 Development **6**
- 3.1 Input . . . 6
  - 3.1.1 Raw Data . . . 6
  - 3.1.2 Test Scenarios . . . 7
  - 3.1.3 Constrained TIN . . . 8
- 3.2 Pre-processing . . . 9
  - 3.2.1 Finding Source Points . . . 10
  - 3.2.2 Propagation Paths: Theoretical Approach . . . 11
  - 3.2.3 First-Order Reflection . . . 13
  - 3.2.4 R-Tree . . . 15
- 3.3 Processing . . . 17
  - 3.3.1 Finding the Receiver Triangle . . . 17
  - 3.3.2 Straight Walking . . . 17
  - 3.3.3 Constructing the cross-sections . . . 17
  - 3.3.4 Adding Semantics . . . 18
  - 3.3.5 Collinear sources . . . 19
  - 3.3.6 Exporting the cross-sections . . . 20
- 3.4 Post-Processing . . . 20
  - 3.4.1 Receiver Sound Level . . . 20

4 Results and limitations **22**
- 4.1 Find sources . . . 22
- 4.2 Finding propagation paths . . . 22
  - 4.2.1 First-Order reflections . . . 23
- 4.3 Cross-Sections . . . 23
- 4.4 Noise map . . . 25

5 Quality Assessment **27**
- 5.1 Input Data . . . 27
- 5.2 Quality of Noise Maps . . . 28
- 5.3 Performance . . . 30

6 Research Conclusion **33**
6.1 Discussion . . . 33
6.1.1 Proof of concept . . . 33
6.2 Recommendations for Further Research . . . 34

Bibliography **35**

7 Appendix **36**
7.1 Simplifying Data Structure . . . 36
7.2 Constrained TIN using Triangle library . . . 36
7.3 Second-Order Reflections . . . 38

1 Introduction

1.1. Context

In the current state of the art of our cities, several challenges arise when it comes to providing citizens with a healthy urban environment. Among these challenges, noise pollution is one of particular interest since, albeit it cannot be visually perceived nor experienced by touch, it still has a significant impact on general quality of life. “Europe is acting to fight noise pollution. The Environmental Noise Directive (2002/49/EC) requires EU Member States to determine the exposure to environmental noise through strategic noise mapping and elaborate action plans to reduce noise pollution.” (European commission [4]). The present research is inserted in such context of assessing noise pollution and aims to provide the noise research community with technical procedures for a better understanding of our built environment.

In the context of the Synthesis Project for the MSc Geomatics at TU Delft, a group of students focused on finding noise propagation paths that could be used in Test_cnossos. In this chapter, the project and its stakeholders will be introduced in more detail. In chapter 2, the research method and technical procedures are discussed. In the next chapter, the different steps of the algorithm are described. It states which choices, and why they have been made. In the fourth chapter, the results and limitations of this project will be discussed. It describes why possible alternative theories have not been applied and explains why the result is as it is. A quality assessment is done in the next chapter. Here the produced results are compared with the current Dutch method. Afterwards, a conclusion is drawn in the last chapter. Finally, an appendix will discuss several elements that were modelled but not used in the final product for different reasons.

1.2. About the stakeholders

The National Institute for Public Health and the Environment (in Dutch: Rijksinstituut voor Volksgezondheid en Milieu, [8]) is a public body affiliated to the Ministry of Health, Welfare and Sport. Arnaud Kok and Rob van Loon, who represent this client, work at the Noise Expertise Centre (Expertisecentrum Geluid, [3]). This Centre aims ”to ensure that noise levels and their effects on the built environment and public health are properly and reliably determined and monitored over time.” [3].

The Directorate-General for Public Works and Water Management (in Dutch: Rijkswaterstaat, [9]) is a public body affiliated to the Ministry of Infrastructure and Water Management. Renez Nota, who represents this client, integrates a group of technicians which is responsible for performing noise studies and monitoring noise production along national roads [7].

In collaboration with these two government agencies, the 3D Geoinformation research group at TU Delft is investigating an efficient approach for automatically modelling 3D data on noise sources and the environment for the whole of the Netherlands based on existing data from multiple sources [2].

As a group of students enrolled in the MSc Geomatics at TU Delft, for the present synthesis project, we find ourselves involved in an ongoing project, and our current task is to conduct scientific research on the efficiency of (geometrical) sound propagation techniques, according to the Common Noise As-

sessment Methods in Europe (CNOSSOS-EU).

1.3. Current state of sound propagation in noise simulation

Noise simulations make use of one or multiple receiver points at positions where the noise level is requested. Sources - such as roads, railways, and industries - are established as elements that emit noise. Therefore, the path(s) between source(s) and receiver(s) must be found.

In the current commercial approach, only 3D polylines can be used as input to describe the terrain. These 3D polylines are semi-automatically generated by noise experts, based on the principle of describing the terrain profile with as few height lines as possible, which leads to:

- The generation of breaklines at positions where height variance is high;
- The generation of more lines near the noise source, considering that the closer the height line is from the source, the greater its relevance in the model.

In order to propose a more efficient, standardised and economic modelling approach, a partnership between RIVM/RWS and the 3D Group was launched in 2017, aiming to generate these height lines automatically from the available datasets, namely AHN3, BAG and BGT, which are publicly available via PDOK for free. A digital terrain, modelled as a TIN directly from the AHN3, was then proposed by the 3D Geoinformation Group and was frequently tested in order to strike a balance between simplification (due to computing performance) and height line accuracy.

1.4. Problem definition

From the last reports of this research [2], it was concluded that extracting height lines from a TIN is still not a satisfactory technique in order to substitute the semi-automatic approach. Moreover, the original TIN preserves the most accurate height data, so there is no point in proceeding with all the conversion steps to satisfy the input requirements of commercial simulation software, especially because it will (re)create a TIN for calculation kernel.

The research hypothesis, therefore, is: *Using a TIN directly allows automated 3D noise modelling according to the guidelines of CNOSSOS-EU.* Nevertheless, this was never tested because existing software cannot take the TIN as input.

1.5. Goal

Considering the aforementioned panorama, the goal of this synthesis project is answering the research question: Is it possible to obtain propagation paths in noise simulation studies by directly using a 2.5 TIN representing a DTM, rather than the currently used 3D height lines?

Therefore, adding to our hypothesis, by implementing an algorithm that is capable of eliminating such undesirable conversion steps, the noise modelling process will be most likely more efficient and accurate. Once implemented and tested in terms of effectiveness, this new approach would potentially represent a milestone in noise simulation. This could be a game-changer in noise simulation not only in the Netherlands but also in Europe due to CNOSSOS-EU regulations. Therefore, it is useful to compare possible results with current methods.

1.6. Scope of this Project

In this project, the noise propagation pathfinder is based on a TIN. The noise propagation pathfinder provides the paths to the algorithm for noise emission and propagation of CNOSSOS-EU. At first, we prepare the raw input datasets. Then, we prepare the constrained TIN for different scenarios. At first, direct paths are computed, but later on, reflected propagation paths of the first-order are modelled as well. Besides direct and reflected paths, there are diffracted paths. Vertical diffracted paths are taken care by Test_cnossos, while the horizontal diffracted paths could have been done in this project. This is not the case however. Propagation paths can occur in homogeneous, favourable, and unfavourable

conditions. Homogeneous conditions are when the sound rays are straight segments and when the sound waves are considered constant in all directions [11]. When the sound rays are curved towards the ground, it is a favourable condition. An unfavourable condition occurs when the sound rays are curved towards the sky [11]. The main priority of this project is modelling propagation paths in homogeneous conditions. If the time allows it, propagation paths can also be modelled in favourable conditions. However, this has a low priority. Finally, a noise map is produced to visualise the sound levels per receiver.

The calculation of noise propagation can be time-consuming. Therefore, a balance must be found between the optimisation of the calculation time and the overall accuracy of the noise propagation paths. At first, the project is more focused on the accuracy of the code. Afterwards, an optimisation of the code can be made.

The main limit in this project might be the time period of nine weeks. In this project, big datasets are going to be tested on many variants of each scenario. This will take a lot of time.

1.7. Requirements (MoSCoW)

At the beginning of this project, a table conforming to the MoSCoW method was created. This method consists of four different categories of requirements: must have, should have, could have, and won't have requirements [1].

The requirements labelled as "must have" are critical. If these are not present in the code, the code is considered a failure. "Should have" requirements are important, but not as necessary in the current time period. Requirements labelled as "could have" are desirable, but not necessary. However, they could improve the experience of the result. These requirements are typically included, if the time allows it. "Won't have" requirements are the least-critical requirements for the code [1].

	Must Have	Should Have	Could Have	Won't Have
LoD	1.2	1.3	2.0	3.0 or higher
Paths	Direct propagation paths (in homogeneous by the group and favourable conditions by Test_cnossos)	1st order reflections (in both homogeneous conditions by the group and in favourable conditions by Test_cnossos	Diffraction (horizontal)	higher order reflections and unfavourable conditions
Additional elements	Evaluating the efficiency of TINs in these conditions	Evaluating the efficiency of TINs in these conditions	Optimisation to create a noise map of defined area	

1.8. Expected Results

The research question of this project is: "Is it possible to obtain propagation paths in noise simulation studies by directly using a 2.5 TIN representing a DTM, rather than the currently used 3D height lines?" In order to prove this concept, a pathfinder algorithm is modelled in this project. It is expected this algorithm offers a valid and efficient alternative to noise propagation. The research question is answered in section 6.

2
Research Method

2.1. Method

Considering the scientific purpose of the synthesis project, a first development step was addressing the present research with an appropriate rational path towards the answer of its question: 'Is it possible to obtain propagation paths in noise simulation studies by directly using a 2.5 TIN representing a DTM, rather than the currently used 3D height lines?'

Such interrogation brings the challenge of comparing both technical procedures, i.e. the currently commercially available one and the one proposed through this investigation. In this sense, sound propagation modelling would then become the core of the project.

In a modelling approach, features of the real-world built environment, such as buildings, terrain, roads, etc. are gathered into abstract classes with specific attributes and operations. Therefore, since objects belonging to the same class share a common data schema, it is possible to implement an algorithm that repetitively iterates through these objects and computes sound propagation, the phenomenon under study.

This deductive reasoning, i.e. applying general rules which hold over every single building, piece of terrain, etc., despite their particular characteristics, configures the methodological core of this project, and it is used while conducting all technical procedures dealt by the main algorithm.

Once the algorithm is implemented, the empirical comparison between these two modelling techniques (currently commercial one vs. proposed one) takes place, and the research question is answered.

2.2. Technical Procedure

Several technical procedures are used in this project. First of all, a literature study is done. This is done to understand the basic principles of noise propagation. The JRC Reference Reports, written by the European Commission to be used for strategic noise mapping in the European Union [5], is studied. It explains the different path types that the Test_cnossos software can handle:

- Direct path potentially with horizontal diffraction
- Reflected path
- Reflected path with horizontal diffraction
- Vertically diffracted path

Besides these path types, the image method for reflected paths is explained. Finally, the calculation of the power levels is explained.

The second technical procedure is the algorithm. In figure 2.1, the workflow of the project is visible. A 2.5D TIN representing a DTM is used to create cross-sections from the matching source-receiver pairs. Using road lines and receiver points, the source points are generated. The buildings are used

for establishing reflection points. These cross-sections, written to an XML file, are then used to run in Test_cnossos. Beside the cross-section, additional information about the power level is added to the XML file before it is run in Test_cnossos. Test_cnossos takes these XML files as input and calculates noise levels at the receivers. Afterwards, the Test_cnossos output XML files with the noise levels corresponding to the same receiver, are energetically added.

In the next chapter, the decisions made throughout the project are explained, step by step, starting from the raw input data to the results at the end.

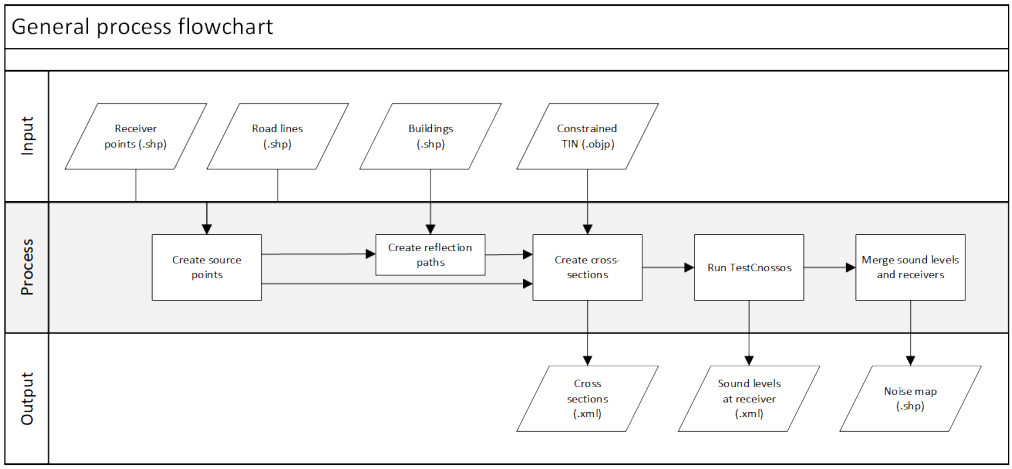


Figure 2.1: Flowchart of the process in general.

The third technical procedure is the verification of the proof of concept. The goal of this project is to investigate the possibility of obtaining noise propagation paths directly using a 2.5D TIN representing a DTM.

The last technical procedure that is used in this project is the comparison. The results of this algorithm are compared to other results, namely the results of the Dutch noise simulation software.

3
Development

3.1. Input

3.1.1. Raw Data

There are several input files as can bee seen in figure 2.1. At first, the clients provided the project with several files containing receiver points. These files were different in terms of the number of receiver points in the files. In the end, however, a 3m x 3m regular grid of receiver points is created, thus allowing a successive interpolation, at the receiver's position, of noise power values in order to generate a continuous field of sound, which is represented by a noise map.

A JSON file with a semantic constrained TIN was also provided.

Regarding noise sources, it was decided to work exclusively with road segments rather than a combination of these roads with railways and/or industrial plants. The reason for this decision is the fact that railway noise emission is more difficult to handle since trains are long and the noise produced by them is not omnidirectional. A similar issue applies to industrial plants, considering that their area is not negligible. In this sense, road segments were the only noise source taken into account. Among all four available road datasets in our disposal ('NWB wegvakken', 'TOP10NL Wegdeelhartlijn', 'RWS Verkeersongevallen wegvakgeografie' and 'Geluidregister homogenewegvakken'), the first three of them satisfy the condition of presenting local roads as line segments, the only requirement in order for the algorithm to run properly. The RWS Verkeersongevallen wegvakgeografie was used for testing the scenarios that we created, but using another road dataset would not modify the proof of concept.

The building models were prepared in a collaboration between the RWS, RIVM, and the 3D geoinformation research group from TU Delft. They prepared a file with building models with an LoD1.2 and LoD1.3. The latter was used in this project. Table 3.1 contains more specifics about the building model files.

	Format	Size
building1.2	ESRI Shapefile	18.6MB
building1.3	ESRI Shapefile	15.6MB

Table 3.1: Input Files: Building Models.

The terrain data was also provided. There are several options to use the terrain data. First of all, the terrain data is provided for TINs and 3D lines, the so-called height lines. For the scope of this project, the files concerning the TIN are used. The threshold of the errors could be 0.3m, 0.5m, or 1m. The terrain was provided in different formats, namely ESRI Shapefile, OGC GeoPackage, and Wavefront OBJ. The input files concerning terrain data are summarised in table 3.2. For this project, the TIN with an error threshold of 0.3m was chosen in Wavefront OBJ format. The TIN is quite accurate and the file itself is the most compact using this format.

Error Threshold	Name	Format	Size
0.3	tin_03m.shp	ESRI Shapefile	132.6MB
0.3	tin_03m.gpkg	OGC GeoPackage	159MB
0.3	tin_03m.obj	Wavefront OBJ	72MB
0.5	tin_05m.shp	ESRI Shapefile	61.6MB
0.5	tin_05m.gpkg	OGC GeoPackage	73.4MB
0.5	tin_05m.obj	Wavefront OBJ	33.2MB
1.0	tin_1m.shp	ESRI Shapefile	17.7MB
1.0	tin_1m.gpkg	OGC GeoPackage	20.8MB
1.0	tin_1m.obj	Wavefront OBJ	9.1MB

Table 3.2: Input Files: Terrain Data.

Besides the terrain data, the ground types were also provided and used. The file with the ground types is based on the BGT (Dutch: Basisregistratie Grootschalige Topografie, English: Basic Register for Large-Scale Topography). The acoustic classification is added to this BGT. It states whether an object is absorbing or reflecting noise. The minimum object area can be chosen between $6m^2$, $12m^2$, and $18m^2$. For this project, a minimum object area of $6m^2$ was chosen as it is the most accurate. These files were provided in two formats, namely ESRI Shapefile and OGC GeoPackage. ESRI Shapefile was chosen since it is the industry standard right now. An overview of these input files is given in table 3.3.

Minimum Object Area	Name	Format	Size
$6m^2$	tiles_bodemvlakken_6.shp	ESRI Shapefile	12.6MB
$6m^2$	tiles_bodemvlakken_6.gpkg	OGC GeoPackage	13.8MB
$12m^2$	tiles_bodemvlakken_12.shp	ESRI Shapefile	12.1MB
$12m^2$	tiles_bodemvlakken_12.gpkg	OGC GeoPackage	13.2MB
$18m^2$	tiles_bodemvlakken_18.shp	ESRI Shapefile	11.8MB
$18m^2$	tiles_bodemvlakken_18.gpkg	OGC GeoPackage	12.8MB

Table 3.3: Input Files: Ground Types.

3.1.2. Test Scenarios

While analysing the input data, three scenarios were defined, each one of them representing a particular environment. The first one, a suburb block, was used throughout the developing process and was repetitively tested for debugging. The other two scenarios were generated in order to check if there would be exceptions that had not been foreseen in the code.

Scenario 1 - Suburb block: Located in between Amazonenlaan, Cilostraat, Plutostraat, and Cassandrastraat, in the city of Rotterdam, this is a one-block scenario with building blocks and road segments. Considering its similarity with the scenario available in the CNOSSOS-EU guidelines (figures VI-1 to VI-5) in terms of urban design, it was specifically selected for testing the algorithm while this was still under development.

Scenario 2 - Rural land: Located on the edge of Golfpark Rotterdam, next to the corner of Broekkade and Rotterdam Rechter Maasoever, in the city of Rotterdam, this is a 100m x 100m scenario with no buildings and a local road segment. The purpose of this scenario is testing if the algorithm would run properly with a greater terrain height difference and a larger reflective surface, considering the artificial hills of the relief.

Scenario 3 - City Centre: Located at Beurs underground station, in the city centre of Rotterdam, this is a 100m x 100m scenario with high buildings and some road segments. The purpose of this scenario is testing if the algorithm would run properly with high buildings, considering the diffraction phenomenon

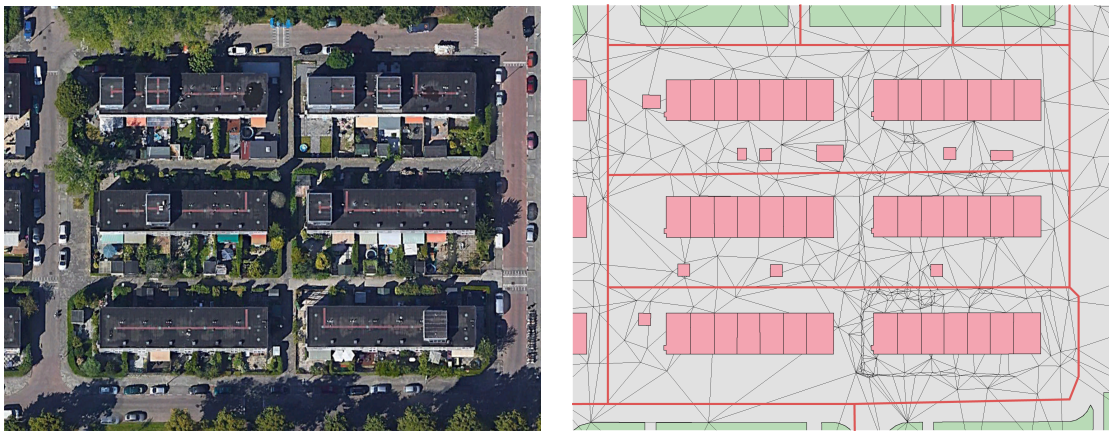


Table 3.4: Suburb block - Aerial image. Source: Google Earth. (left); Scenario 1 (right).

Table 3.5: Rural Land - Aerial image. Source: Google Earth. (left); Scenario 2 (right).

in noise propagation.

A comparison among these three scenarios highlights the difference between them (see table 3.7).

3.1.3. Constrained TIN

The project started with an LoD 0 TIN [6]; however, this led to many intersection tests that complicated and slowed down the process. Therefore, it was clear that creating an LoD 2 TIN [6] would simplify and optimise the algorithm. To create the constrained TIN, which will be used as input, two steps are taken. First, a CityJSON is created which has all the triangles and the semantics of each triangle. Secondly, an OBJP file is created which stores all triangles with their corresponding adjacent triangles and the semantics of each triangle.

CityJSON

The first step is to create a constrained TIN CityJSON file. It generates the constrained TIN from two input files. The first is a TIN of the corresponding area, of which only the vertices will be used. The second is the buildings and ground-type polygons file which will be used as constraints. Triangulating the TIN vertices with its constraints generates a constrained TIN which is then stored in a CityJSON file. In the cityjson, a CityObject presents a semantic type with all boundaries in the CityObject being

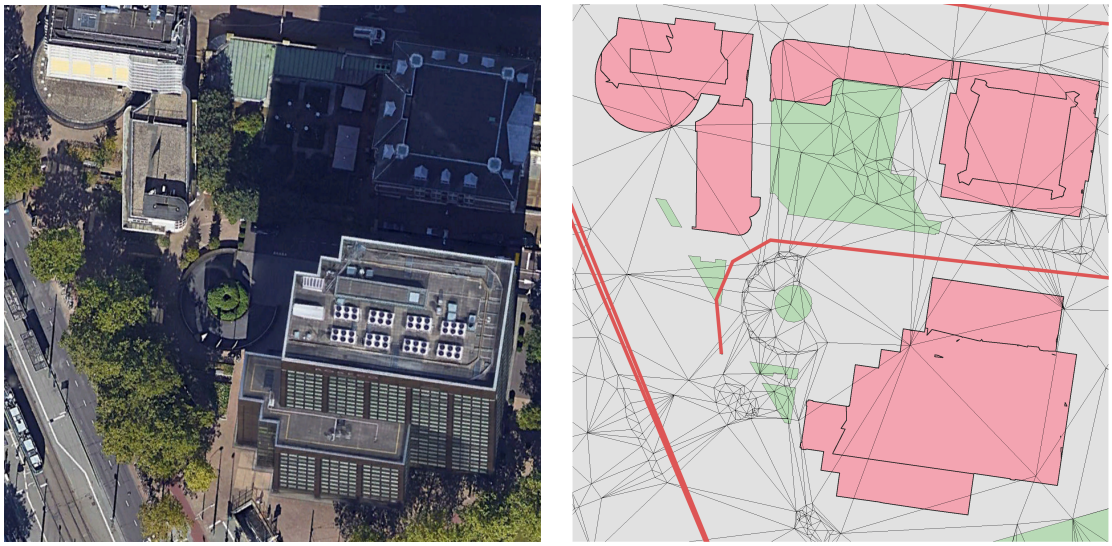


Table 3.6: City Centre - Aerial image. Source: Google Earth. (left); Scenario 3 (right).

	Area (sq.m)	Roads (m)	Building parts (#)	Maximum building height (m)	Ground Type (%)	Terrain height difference (m)
Scenario 1	11.400,0	622,2	56	5,14	Absorbing: 94,4% Reflecting: 5,6%	1,73
Scenario 2	10.000,0	137,4	0	-	Absorbing: 10,1% Reflecting: 89,9%	15,5
Scenario 3	10.000,0	255,6	25	92,9	Absorbing: 92,1% Reflecting: 7,9%	6,0

Table 3.7: Comparison among the three scenarios.

triangles that have that semantic type. Therefore, this also gives the triangles their semantic value from this file. The semantic of a triangle is which building or ground-type it belongs to.

OBJP file format

While the CityJSON file does provide all constrained triangles and their semantics it does not provide the adjacency of triangles. Since this is an unnecessary cost to the program if it needs to be done each time the program is run, it is done as a pre-processing step. This step takes the CityJSON as an input and determines all the adjacencies of the triangles. The result is then stored in an .objp file. This file is very similar to obj, but every face (f) has six (6) values instead of three (3). The last three (3) represent the index of the face that is adjacent to f. Furthermore, it also stores an attribute for each triangle. These lines start with an 'a'. This means that there are as many face (f) lines as attribute (a) lines. This file can then be used as the semantic constrained TIN input to the program.

3.2. Pre-processing

The pre-processing in this project is needed to add more information to the input data before the processing phase. In this project, the processing purpose is creating the cross-sections. This means that

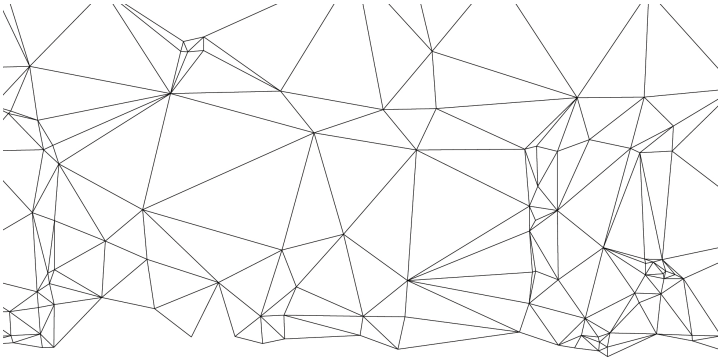


Figure 3.1: Conforming TIN.

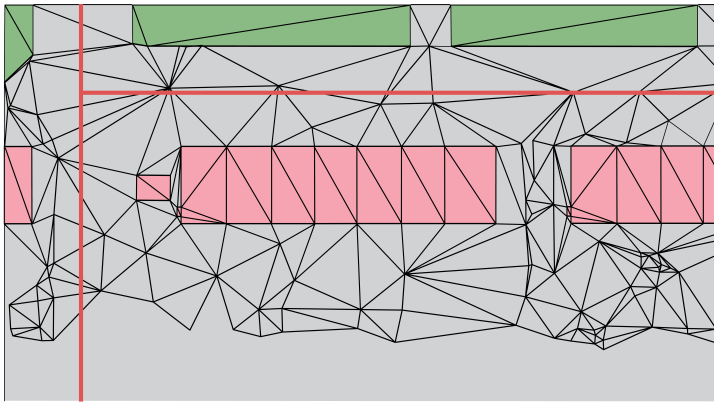


Figure 3.2: Constrained TIN with Buildings and Ground Types.

finding the source points and generating the first-order reflections are part of the pre-processing.

3.2.1. Finding Source Points

In order to create the cross-sections, the source points per receiver point must be found. At first, an input file containing receiver points is used. The receiver points are equidistant from each other and outside of buildings. This is visible in figure 3.3. The red lines represent the roads, the pink shapes represent buildings and the blue dots are the receiver points.

In the case of this project, there are several receiver points placed in a test scenario. From each receiver point, 2km long rays are shot at a two-degree interval. The distance of 2km is standardised by the CNOSSOS-EU guidelines. The length of the ray is a parameter and can be adjusted. The two-degree interval between the rays is a standard and recommended by RIVM and RWS to use. The result is a receiver point with 180 rays. Every ray is checked whether it intersects with a road in the test scenario or not. If the ray from the receiver intersects with a road, the intersection point represents a source point and will be saved. In figure 3.4, multiple rays are shot from one of the receiver points. Multiple source points belong to one receiver point.

Since the intersection points are from a direct propagation paths, some of them are collinear. Simplification is done by sorting the source points that intersect the same ray. They are sorted so that the cross-section only has to be made once. The other source points that lie within this cross-section are dividing the cross-sections into parts.

In addition to the source location, an estimation of the road length that the source represents is made. It is computed by making a (virtual) intersection with the same road element both half of the interval

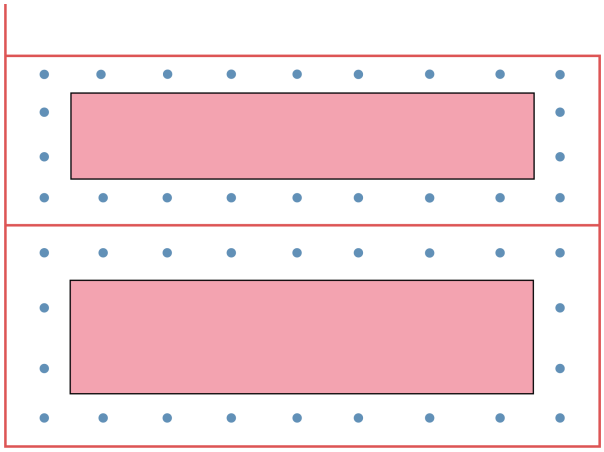


Figure 3.3: Receiver Points in Test Scenario.

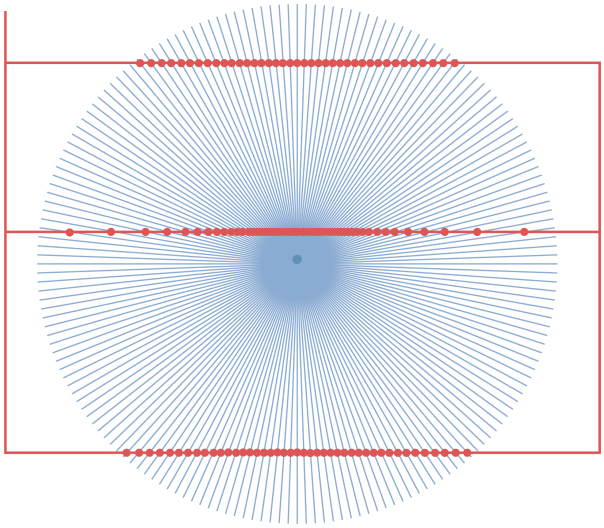


Figure 3.4: Finding the Source Points.

angle to the left and to the right, as can be seen in figure 3.5. The intersection is virtual since the road element might stop within one degree, but in many cases, a new road element in (roughly) the same orientation will be there. The distance between the left and right point is determined and stored with the source. In short, the length estimation takes into account the distance between source and receiver, and the relative orientation of the source-receiver line compared to the road line.

3.2.2. Propagation Paths: Theoretical Approach

The CNOSSOS-EU guidelines [4] elaborates on elementary propagation paths for sound-propagation modelling and indicates four types of paths to be considered, namely:

- Type 1 - 'Direct' paths
- Type 2 - Paths reflected on vertical (or slightly sloping) obstacles
- Type 3 - Paths diffracted by lateral edges of obstacles
- Type 4 - Mixed Paths.

Among these four, the first two types are the ones taken into account for the purpose of this synthesis project, in accordance with the MoSCoW table.

Input : $receiver$ point;
Set of $roads$ (only road within search radius are selected
Output: receiver point with the corresponding source points ordered on distance

for $angle$ *from 0 to 360 in steps of 2°* **do**
  end_point ← the end of a 2.000 m ray from receiver in direction of $angle$;
  for *each* $road\_segment$ *in* $roads$ **do**
    if $receiver$ - end_point *line intersects with* $road\_segment$ **then**
      $source$ ← intersection point;
      $source\_length$ ← estimation for the source length;
    end
  end
end

Algorithm 1: Finding noise sources, executed for each receiver

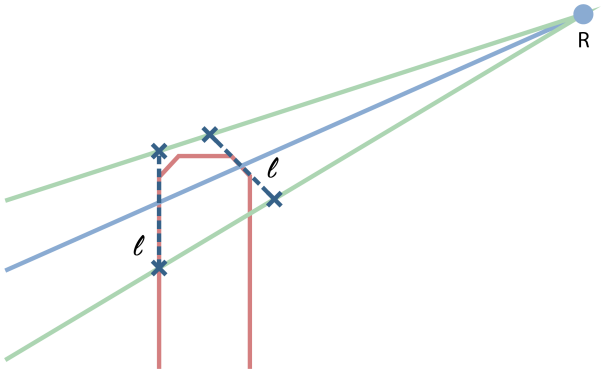


Figure 3.5: Estimation of the Road Length (red: road lines, blue dot: receiver, green: one-degree rays, blue dotted lines: estimated road length)

1. Type 1 paths are "'direct' paths from the source to the receiver, which are straight paths in plane view and which may nevertheless include diffraction on the horizontal edges of obstacles." (figure 3.6)
2. Type 2 Paths, in turn, are the ones "reflected on vertical or slightly sloping (< 15°) obstacles (...)" and may also include diffraction on the horizontal edges of obstacles (...)" (figure 3.7 and figure 3.8)

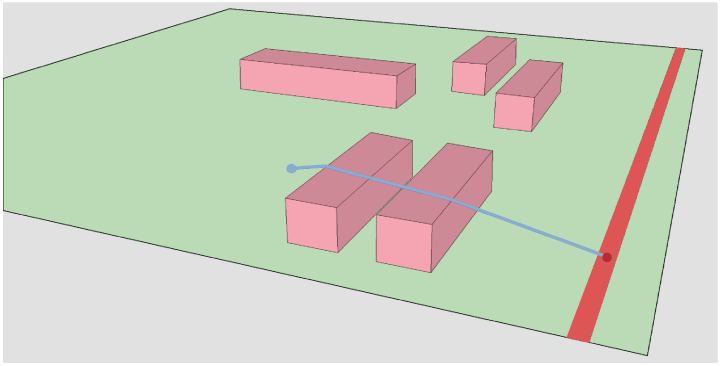


Figure 3.6: Example of Type 1 path (direct path). Adapted from CNOSSOS-EU.

First, direct paths are computed from the noise source's position towards the receiver's position. Second, for the purpose of this project, the reflection paths are generated by first analysing the model and computing, for each pair of source (S) and receiver (R), what are the candidate points on building facades and/or on other vertical surfaces such that, starting from the source point (S), once a sound

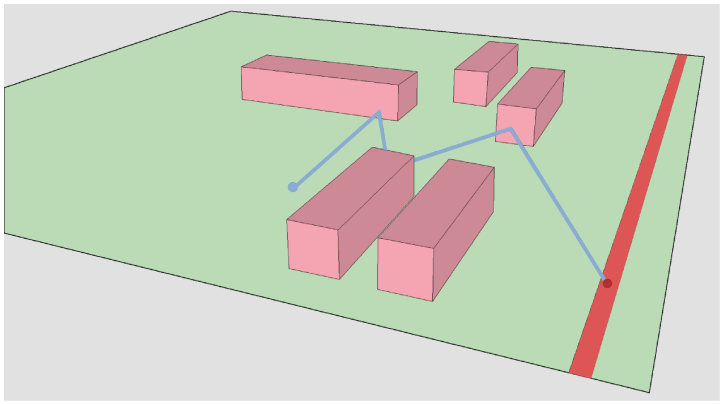


Figure 3.7: Example of Type 2 path (reflection path) without diffraction. Adapted from CNOSSOS-EU.

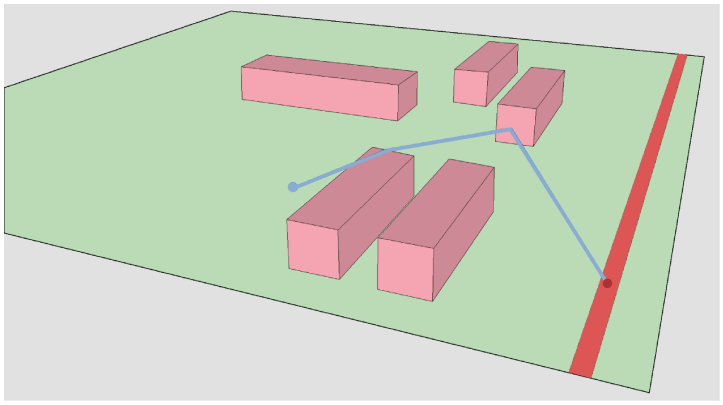


Figure 3.8: Example of Type 2 path (reflection path) with diffraction on horizontal edge. Adapted from CNOSSOS-EU.

wave hits them, the sound wave will be reflected towards the receiver point (R).

The technical solution for finding these candidate points is the 'image method'. Within this technique, each vertical obstacle, i.e. a 2D line segment in the XY plane, acts as a mirror and produces a virtual image source (S') from the original source (S) that is present in the model. By connecting the image source (S') with the final receiver point (R) or successive reflection point, it is possible to determine the intersection of the virtual propagation path and the wall segment under analysis, if there is any.

It is worth mentioning that, "when dealing with reflections on significantly sloping obstacles, the method should be applied in 3D". Since the modelling is implemented with LoD 1.2/1.3 buildings, all walls are treated as absolute vertical planes. Moreover, reflections on the ground are not dealt with by the 'image method', but rather dealt with by the Test_Cnossos software. According to the CNOSSOS-EU guidelines, they are taken into account in the calculations of attenuation due to the boundary (ground, diffraction).

Finally, once the reflection point(s) are found, it will be possible to define a 2D cross-section of the geometry, created by the succession of all vertical planes passing through the straight line segments located between source, reflection point(s) and receiver. The CNOSSOS-EU guidelines characterises this process with the metaphor of a Japanese screen being unfolded in order to create a single plane.

3.2.3. First-Order Reflection

A reflection path may have one or multiple reflection points along its trajectory. If there is 'n' reflection point(s), there is a n-order reflection.

When dealing with first-order reflection, the algorithm has a very straightforward approach: In the 2D plane, given a pair of source (S) and receiver (R), for each building facade, i.e. a line segment, there

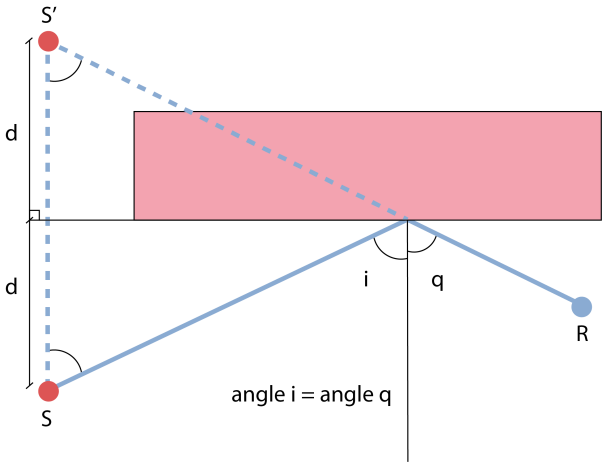


Figure 3.9: Example of reflection on an obstacle dealt with by the image source method (S: source, S': image source, R: receiver). Adapted from CNOSSOS-EU.

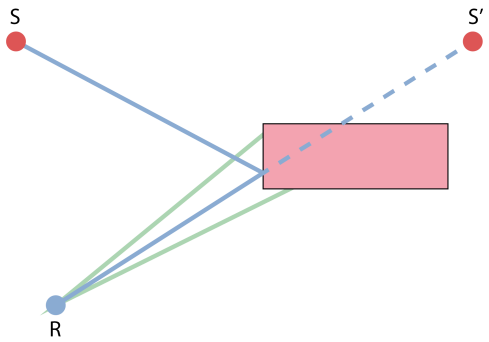


Figure 3.10: Example of the one-degree test applied to a first-order reflection. By rotating R-S' to the left and to the right, the algorithm verifies if these two rotated paths intercept one of the walls of the building.

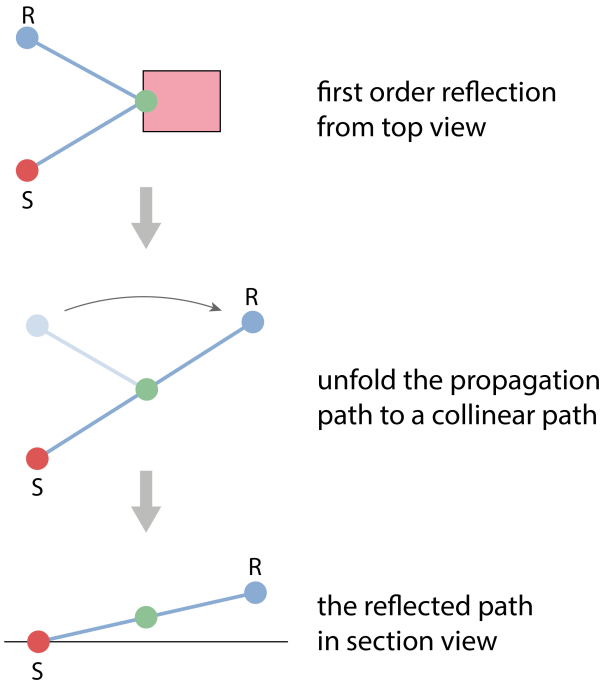


Figure 3.11: Japanese screen metaphor: Sound ray reflected to the order of 4 in a track in a trench: actual crosssection (top), unfolded crosssection (bottom). Adapted from CNOSSOS-EU.

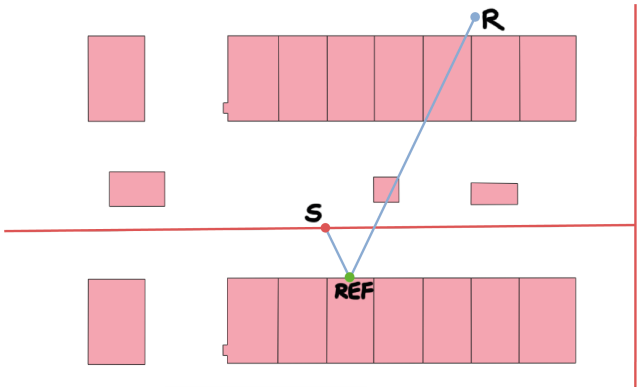


Figure 3.12: Computed first-order reflection. Given a source (S) and a Receiver (R) points, finding the reflection point (REF) does not require any testing. The system of linear equations determines the x and y coordinates of this point.

will be either zero or one reflection point (REF) within this facade that satisfies the image method. If the REF point lies in between the vertices of this line segment, the reflection takes place. If not, there's no possibility that this particular facade reflects the sound towards the receiver.

If the path could theoretically reflect upon the building, in advice of the clients, two validity tests are implemented to verify the reflection.

1. The buildings in which the sound waves are being reflected need to be checked in order to verify if their dimensions and/or distance from the source point are significant enough to produce reflection points. According to the CNOSSOS-EU guidelines, this test should be simple: "The obstacles where at least one dimension is less than 0.5 m should be ignored in the reflection calculation, except for special configurations." However, it should be noticed that, depending on the distance from the noise source to the reflection point, 0.5m could be rather (in)significant. As an alternative, requested by the clients a one-degree test is applied: having the receiver point as the center, the reflection point is rotated one degree to the left (counterclockwise) and one degree to the right (clockwise), and if these two new left and right lines still intercept any wall of the same building, then the reflection point (REF) is valid for a reflection path.

2. In urban areas, buildings are often connected through adjacent walls, commonly seen in housing blocks (rijtjeshuizen in Dutch) Therefore, the exterior side of a wall can, at the same time, be the interior wall of another building. In this case, the reflection is only valid when the reflecting building is at least one meter higher than the adjacent building. The one-meter threshold is chosen to cope with the simplifications due to the LoD level (flat roofs). This causes buildings with similar roof heights to have a larger spread of mean roof height. The building height, defined in the output file, is in this case relative to the roof of occluding building.

3.2.4. R-Tree

R-trees are data structures used to index geo-referenced data. They group nearby objects into minimum bounding boxes. All objects that are part of that minimum bounding box are leaves of a node. The node then sub-dives all objects within into smaller bounding boxes containing fewer objects. This is repeated until a leaf node contains only one object. This makes it very easy to determine which objects lie within a given area without having to search through every object.

The noise map that needs to be generated has many receiver points covering a large area. However, any individual receiver point covers a smaller area. While dealing with an individual receiver point, only a subset of the overall data is required. This is where R-trees are used. The building and road data sets are indexed through the use of R-trees. This provides the opportunity to quickly retrieve the necessary data for a given receiver point.

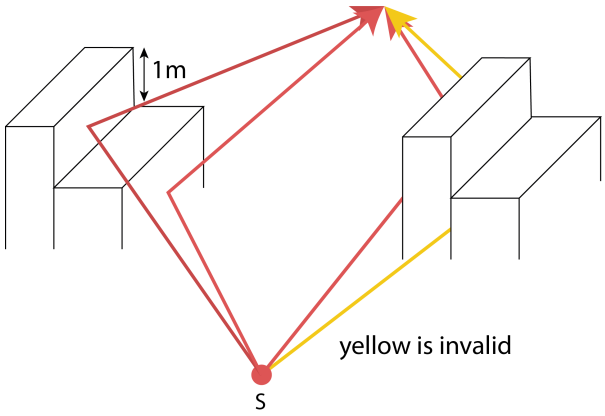


Figure 3.13: Second Validity Test Reflected Paths; the red paths are valid, the yellow path is not processed.

Input : buildings_dict : BuildingManager object - stores all the building objects.
Output: Stores reflection points, and their corresponding heights in the class; Returns True if reflections are found, False if not.

$reflection\_points \leftarrow [\,]$;
$reflection\_heights \leftarrow [\,]$;
for $id, building$ *in buildings* **do**
  for *each* $wall\_segment$ *in the building* **do**
    if *both* $source$ *and* $receiver$ *are on the exterior side of the wall* **then**
      $S' \leftarrow$ projected source image with respect to the wall;
      if *the* S' *-* $receiver$ *line intersects with the* $wall\_segment$ **then**
        $REF \leftarrow$ point of intersection;
        $left\_point \leftarrow$ rotate S' 1° left;
        if $left\_point$ *intersects with a* $wall\_segment$ *of the* $building$ **then**
          $right\_point \leftarrow$ rotate S' 1° right;
          if $right\_point$ *intersects with a* $wall\_segment$ *of the* $building$ **then**
            $ground\_mat \leftarrow$ the material of the triangle on the exterior side on the $wall\_segment$;
            if $ground\_mat$ *is a* $building$ **then**
              if *building* $height$ *- exterior building* $height$ *> 1 m* **then**
                $reflection\_points \leftarrow$ append REF;
                $reflection\_heights \leftarrow$ append building $height$;
              end
            else
              $reflection\_points \leftarrow$ append REF;
              $reflection\_heights \leftarrow$ append building $height$;
            end
          end
        end
      end
    end
  end
end

Algorithm 2: First-order reflection path, executed for each source - receiver pair.

Input : A 2D origin point $origin$;
A destination point $destinations$;
A receiver triangle $current\_tr$;
Output: next edge that intersects the $origin$-$destinations$ segment;
next triangle to walk to;

while *not in* $destination\_triangle$ **do**
  for *each* $edge$ *in* $current\_tr$ **do**
    if *side_test of* $edge[0] <= 0$ & *side_test of* $edge[1] > 0$ **then**
      return $edge$ & $edge\_index$
    end
  end
end

Algorithm 3: Straight-walk Algorithm

3.3. Processing

The processing purpose of this project is constructing a cross-section written into an XML file. It will then be possible for Test_Cnossos to read the XML files and calculate the sound levels per receiver point. The construction of the cross-sections will be explained step by step.

3.3.1. Finding the Receiver Triangle

First, the triangle that the receiver point is located in needs to be found. To find the triangle a point is located in, the ”walk” algorithm is used. The algorithm starts at a triangle (T) in the TIN. Using orientation calculations on the edges of T and the position of the receiver point, the algorithm determines which incident triangle to T the walk algorithm needs to go to next to get closer to the receiver point. Repeat this process until the walk algorithm is inside the triangle that the receiver point is located in.

It is computationally expensive to find the triangle a point is located in. Therefore, performing the operation as few times as possible is the goal. Since there are fewer receiver points than source points, it is more efficient to find the starting triangle for each receiver point.

Furthermore, It is assumed that the closest vertex to the receiver point is part of a triangle that is (or is very close to) the triangle in which the receiver is. Therefore, a 2D KDtree is created for the vertices in the TIN and the closest vertex to the receiver point is queried. Afterwards, a triangle having that vertex as boundary is selected as the starting triangle. Finally, once the starting triangle for an individual receiver point is found, it can be stored in a dictionary. The dictionary can later be used for instant lookup when the same receiver point is used again.

3.3.2. Straight Walking

To create a cross-section, a straight path is required from the receiver point to the source point. The straight walk algorithm is used to solve this issue, see Algorithm 3. To start the algorithm, the receiver point, source point, and starting triangle, which is the triangle the receiver point is located in, are needed. At each iteration of the walk the edge of a triangle that the straight path intersects with next is returned. The process is repeated until the walk has arrived at the destination point. This provides a list of all edges that the straight path intersects on its walk from receiver point to source point. These edge are used to make the cross-section. See Figure 3.14 for a visual of how the algorithm works.

3.3.3. Constructing the cross-sections

There are two types of cross-sections to construct:

- Direct path cross-sections, from receiver to source;
- Reflected path cross-section, from receiver to reflected point to source.

Therefore, the reflection point and source are stored in a list of destinations that needs to be looped over. The origin is set to the receiver at first and then, when looping through the second destination,

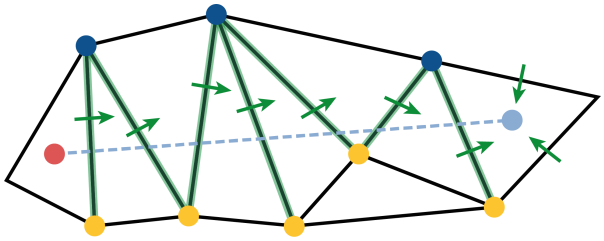


Figure 3.14: Straight Walking to the Receiver Triangle

the origin is set to the reflection point.

When an edge is found while straight walking, the 2D intersection point between the edge and the segment origin-destination is computed. In the case an edge is collinear with the [origin, destination] segment, the intersection point is in half way between the two end-points of the edge.

The height of the intersection point is interpolated using the side test that was used to find the designated edge to assign weights to the end-points of the edge. Before adding that point to the cross-section, the point still needs to be provided with semantics, its material (see section Adding Semantics). Once a semantic point, the interpolated point is added to the cross-section and then the algorithm keeps walking. When the algorithm reaches the destination triangle, the ground height of the source is interpolated in the TIN and it is appended with the material of the triangle. In the end, the order of the vertices in the cross-section is reversed to have a source-receiver cross-section. A simplification step consists of not adding a point in the cross-section if that point lies within 10 cm of the previous point.

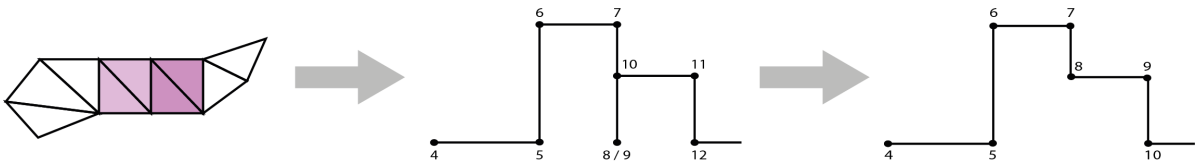


Figure 3.15: Creating a correct Cross-Section.

3.3.4. Adding Semantics

Since the straight-walk algorithm moves from receiver to source, and not from source to receiver, the intersection point inherits from the next triangle on the straight path instead of the current triangle to adhere to the CNOSSOS-EU guideline.

If the current triangle and the next triangle are both ground, each intersection point inherits the ground material of the next triangle. For this, the absorption index of the triangle is checked and the intersection point is assigned a letter, 'C' or 'G'. 'C' for absorption since, according to the CNOSSOS-EU guideline, refers to "uncompacted, loose ground (turf, grass, loose soil)" with an absorption index of 1.0 and a sigma of 80. 'G', for reflection according to the CNOSSOS-EU guideline, refers to 'hard surfaces (most normal asphalt, concrete) with an absorption index of 0.0 and a sigma of 20000. [11] Furthermore, since the datasets provided have accuracy limitations, it was decided that any difference of plus or minus ten centimeters between two points of the same material is superfluous in the cross-section. As a consequence, a threshold was set to skip these close points.

When a building is reached, the cross-section needs to go up, which means both the intersection point and the lifted intersection point to roof level, with their corresponding building material, are added to the cross-section. In the code, all building materials are given an 'A0' letter according to the CNOSSOS-EU guideline. A marker is also added to indicate that a cross-section is in a building. This marker helps avoid adding intermediate points that are not part of the Digital Surface Model (DSM) cross-section before going down again. (figure 3.15) If the cross-section is already in a building and the next triangle is the same building or has the same height (plus or minus 10 cm) and the same material, then the

cross-section ignores the corresponding intersection points.

If the cross-section is already in a building and the next triangle has a different roof height, two points are added each at the height of the current and the next triangle. If the cross-section is already in a building and the next building is ground, both the lifted point with the building material and the intersection point with the ground type are added to the cross-section. To make the code more robust, a condition to skip a building that has a roof level lower than the ground TIN level is added.

Input : A 2D receiver point $origin$;
A list of destination points $destinations$;
A receiver triangle $r_t r$;
A semantic constrained TIN class.
Output: A list of vertices of the cross-section between receiver and source;
A list of vertices materials;
information about the receiver and source

$cross-section \leftarrow$ initialise with projected receiver point;
$material \leftarrow$ initialise with projected receiver point corresponding material;
$in\_building \leftarrow$ False;
$current\_tr \leftarrow r_t r$;
for $destination$ *in* $destinations$ **do**
  while *not in* $destination\_triangle$ **do**
    $e \leftarrow$ edge of $current\_tr$ between $origin$ and $destination$;
    $p \leftarrow$ interpolate intersection point between e and $origin$-$destination$ segment;
    $p \leftarrow$ get semantics of intersection point;
    $current\_material \leftarrow$ material of $current\_tr$;
    $next\_tr \leftarrow$ next triangle on other side of e;
    $next\_material \leftarrow$ material of $next\_tr$;
    if $current\_material$ *and* $next_material$ *are ground* **then**
      $cross-section \leftarrow$ append p;
    else
      if *not* $in\_building$ **then**
        $in\_building \leftarrow$ True;
        $cross-section \leftarrow$ append p;
        $cross-section \leftarrow$ append p elevated to roof level;
      else
        if $next\_material$ *is a building* **then**
          $cross-section \leftarrow$ append p elevated to roof level of $current\_tr$;
          $cross-section \leftarrow$ append p elevated according to roof level of $next\_tr$;
        else
          $cross-section \leftarrow$ append p elevated to roof level of $current\_tr$;
          $cross-section \leftarrow$ append p to cross-section;
          $in\_building \leftarrow$ False;
        end
      end
    end
  end
  $cross-section \leftarrow$ append projected source ;
  $origin \leftarrow$ destination;
end

Algorithm 4: Extract cross-section.

3.3.5. Collinear sources

Since sources are created by intersecting rays from a receiver, constructing cross-sections can be optimised by walking from receiver to the furthest source point along a ray and then each source point along the path splits the cross-section. To split the overall cross-section, 2D sources are located in

Input : A list with succeeding vertices in Cartesian space vts;
A list with the material of each matching vertex mat;
A dictionary where key is vts id and value is the extension type and information ext;
A dictionary with the source noise settings.
Output: None (writes output xml file)

$vts \leftarrow$ subtract the source (first point) from each vertex to make the path local (e.g. relative to the source) *(the source therefore becomes [0, 0, 0])*;
if *min*$(vts.z)$ < *0* **then**
  $vts \leftarrow$ add min$(vts.z)$ to lift the path such that all vertices have a positive height;
end
$path \leftarrow$ initiate the xml tree with standard Test_cnossos setup, meteo, method and validation settings;
$source\_noise\_level \leftarrow$ adapt the default noise levels to the source length;
$path \leftarrow$ insert all control points (vts) and their extensions;
Write the xml tree to "path_i_j.xml" where:
$i \leftarrow$ the receiver number;
$j \leftarrow$ the cross-sections number within the receiver.

Algorithm 5: Prepare and write cross-sections to xml files.

the overall cross-section and their heights are linearly interpolated from the edges in that cross-section (figure 3.16). Weights are assigned to each endpoint of an edge according to the distance along an axis between those points respectively and the interpolated point.

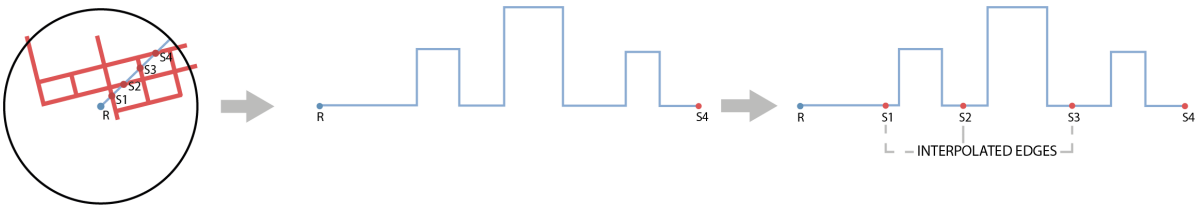


Figure 3.16: Linear Interpolation at the Edges, red lines are road, s1 to s4 are collinear sources

3.3.6. Exporting the cross-sections

The cross-section is stored as a list of Cartesian points (in euclidean space), supported by their materials and extensions. In order for the Test_Cnossos software to process this cross-section, it should be written to an xml file, according to requirements from the JRC-2012 calculation method mentioned in [11]. However, to produce a valid xml file, a few changes are required. These changes, which are described in algorithm 5, should produce a valid and correct xml file. The source, receiver and optionally wall extension are added to the respective points. It holds the relative height and optionally the material of the extension. In accordance with the clients, the sources are stored as type point sources, which broadcast their noise equally in all directions (omnidirectional). The result after this step in the program is one xml file for each cross-section. The requirements only allow for one cross-section to be stored in one xml file. These xml files are to be read by the Test_Cnossos software to compute the noise levels at a receiver position, this is described in 3.4: Post-Processing.

3.4. Post-Processing

Post-processing is the phase in which the processing products are edited or enhanced. In this project, the Test_cnossos software calculated the noise levels based on the cross-sections. In the post-processing phase, these noise levels are visualised in a noise map.

3.4.1. Receiver Sound Level

Once all the cross sections have been produced the sound level for each receiver can be calculated using the Test_cnossos software. The following pipeline is used to generate the sound level at each

receiver:

1. Write a shell file to run Test_cnossos automatically
2. Go over each xml output file and extract the A-weighted equivalent sound level (LeqA) value
3. Energetically sum all LeqA values for each receiver respectively
4. Create a shapefile containing the 2D receiver points with the sound level as attribute

The result is a shapefile with each receiver point having a sound level value.

4
Results and limitations

In this chapter, the results of the algorithm are described. It describes what the outcome of the algorithms ((sub)-results), and what can be said about how this implementation is a simplification of reality (known limitations). A limitation may explain functionalities that were introduced in the Project Initiation Document (PID) that have not been implemented (yet). It may also explain why some theories were not implemented, which may cause artefacts in specific cases. The sub-results are discussed for the technical procedures to find sources, the set of technical procedures to find possible propagation paths and creating the cross-sections. At last, the final result, the noise map, is discussed.

4.1. Find sources

Any method of localising noise sources is a simplification of reality, the implemented method was requested by the client but, as any, is a simplification. This simplification of reality is accepted, as it provides a reasonable estimation and is quick to determine. In most occasions, mainly highways and railways are used. These source line types tend not to have sudden stops or bends. But in cases of a junction, a turn is segmented. Depending on which segment intersects with the receiver ray, the estimated length may be higher or lower than reality (see figure 3.5).hou This is only an issue when the source segment stops before the 1° and is not extended with a new source segment in the same orientation.

The second limitation of finding the noise sources concerns the default noise values. Every source is treated equally in this method. However, some roads are much busier than other roads, which should result in a higher noise level. On top of that, the speeding limit of the roads can differ from road to road. The higher the speed of a vehicle, the higher the noise level. These aspects are currently not taken into account.

Finally, an artefact can be created when a receiver has a short perpendicular distance with the road. This especially happens when the receiver is below or above the road line. As a result, it will have very few sources that will have a long length. This will eventually cause artefacts when running the cross-sections in Test_cnossos(see figure 4.1).

4.2. Finding propagation paths

In collaboration with the clients, a selection of the CNOSSOS propagation path types is implemented. This selection applies to the horizontal (i.e. top view) plane since the vertical part is taken care of by the Test_cnossos software. The implemented types are type 1 and 2, respectively direct and reflected paths. In the case of type 2, it was decided to only implement single order reflections. This was decided since higher-order reflections have shown to cause a low impact on the noise levels. It also leads to multiple unknown variables in the formulas which are computationally expensive to solve. This theory is applied and implemented for second-order reflection paths, but not actively used. It is described in 7. Due to limited time, horizontally diffracted paths were not considered. This adds little value to the proof of concept, and quality control had a higher priority. At last, due to limited time, only horizontal homogeneous conditions were implemented.

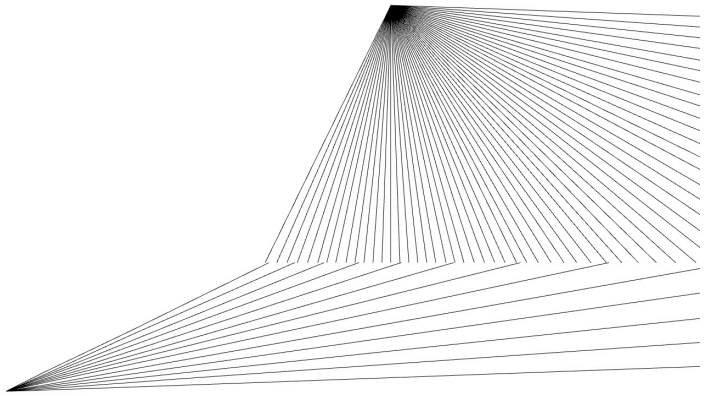


Figure 4.1: A receiver point beyond the end of a road segment (lower) receives much fewer paths (12 vs 86) The estimated source length will solve the difference, but as the intersections get further away from the receiver, the chance of missing a part of the road is inevitable.

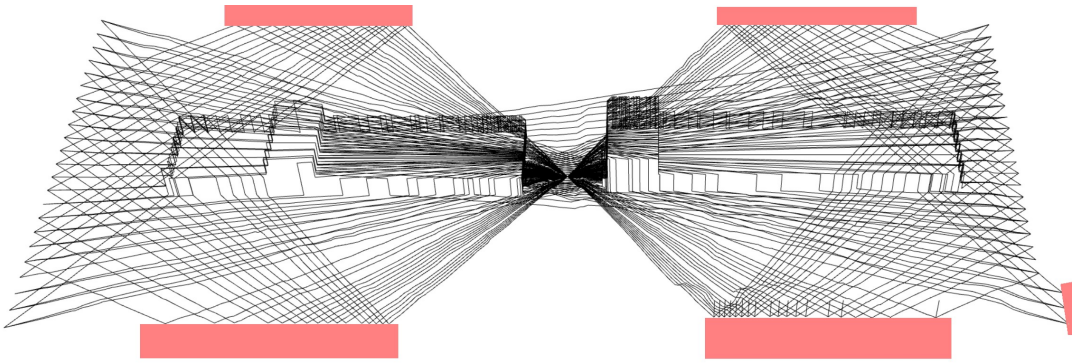


Figure 4.2: An example of a set of reflections for one receiver and two roads on both side. There are two housing blocks included in the paths, and 5 other buildings cause reflections

4.2.1. First-Order reflections

The reflection algorithm produces validated propagation paths for which cross-sections can be made. (see figure 4.2) After a theoretical reflection is found (i.e. disregarding surroundings, building size and height), it is validated according to two tests, as explained in 3.2.3. Both may cause artefacts as visualised in figures 4.3 and 4.4. The test to verify the size of the building can be shown, however, the validity check (whether the reflection is valid in connection to its surroundings) is a result of simplification in the input data since (slanted) roofs are modelled as flat.

Despite the general quality of the results coming from the algorithm, it is important to observe that modelling sound propagation by means of noise sources and receivers is a discrete approach, whereas in reality, the noise phenomenon is continuous. Understanding this discrepancy is important since the exact position of noise sources and receivers plays a fundamental role in determining the reflection path.

In order to overcome this limitation, the algorithm is fed by a regular grid of receivers. This will approach a continuous field while keeping a reasonable processing time. In practice, computing a highly dense grid would be impossible, not only due to computing efficiency but also because the source image method is meant for discrete objects and not for continuous fields.

4.3. Cross-Sections

Limited number of materials

In this code, only two absorption indices for ground type according to the CNOSSOS-EU guidelines are supported: 'G' for an absorption index of 0, and 'C' for an absorption index of 1. Other absorption

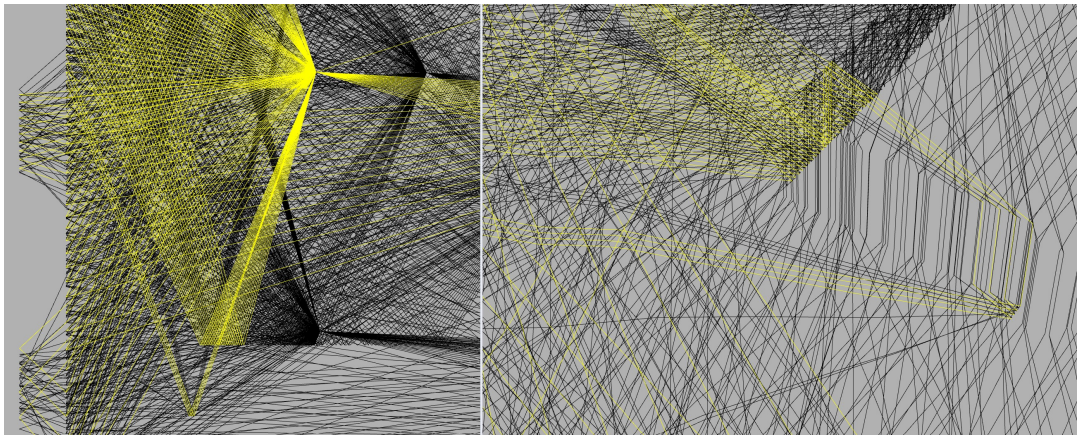


Figure 4.3: An example of reflections that are valid according to our tests, since the size test does return True, as both 1° left and right do intersect with the building, but in such a different spot that it is not in line with the idea behind the test. Note, however, that although these paths will be created and processed, it will result in a very low noise level and will therefore barely influence the noise map

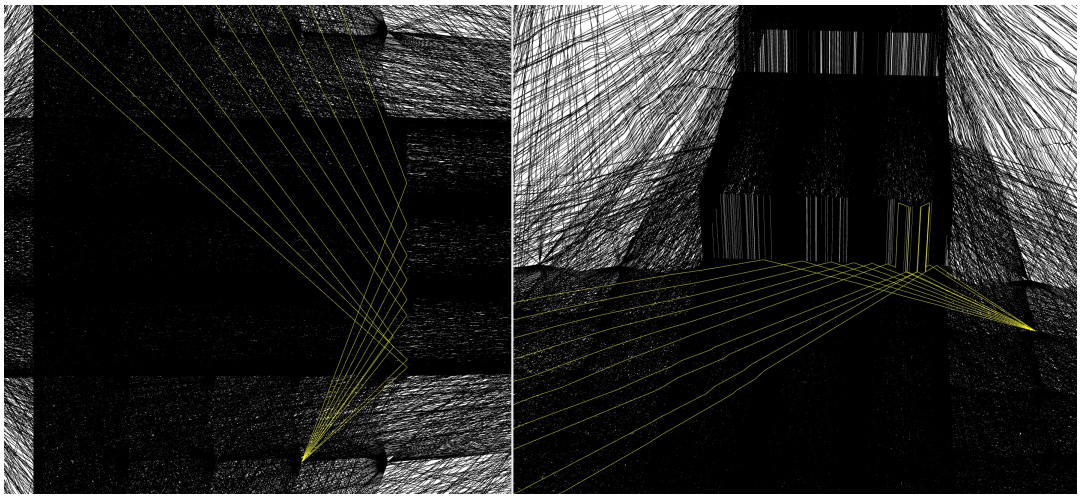


Figure 4.4: Example of a building size verification in which it produces slight artefacts, it is coherent with the current method, but as the building size is tested only from the perspective of the receiver, reflections quickly become invalid on the top side, where they continue to be valid up until the corner of the building on the south side of the building.

indices should be added later on.

Buildings are only assigned one material 'A0' according to the CNOSSOS-EU guidelines. It could be that the roof has a different absorption index than the walls.

Line Barriers Not Supported

Barriers are only supported when they are incorporated in the DTM, or as a building. This will then be accepted as a building, not as extension type barrier.

Spikes

The semantic constrained TIN provides coordinates with up to three digits after the comma whereas the buildings shapefile provides coordinates with up to six digits after the comma. This precision discrepancy causes spikes at the reflection point in cross-sections because it does not correctly identify the triangle in which the reflection point exists. Tolerances were introduced; however, some spikes remained. Increasing the tolerance would result in topological inconsistencies while straight-walking to the source. The shapefiles could have been rounded to the precision of the constrained TIN; however, this does not guarantee topological consistency between both datasets. Therefore, given the time limit,

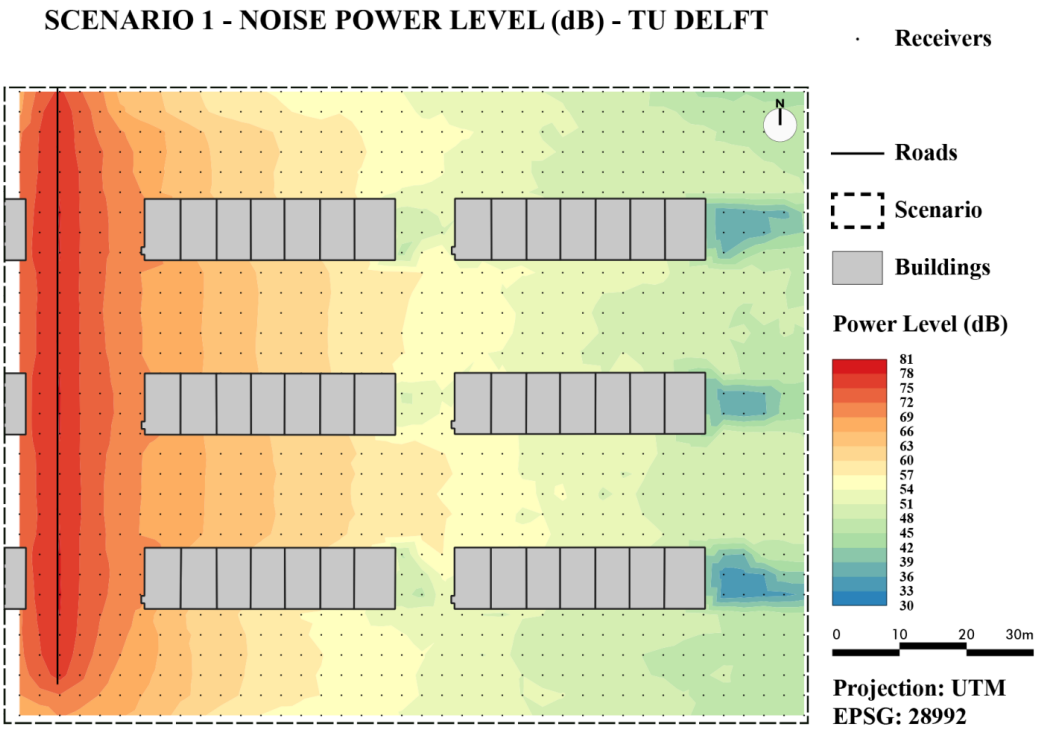


Figure 4.5: Noise map for a suburb block (Scenario 1), with contour representation.

it was decided to leave the remaining spikes for this stage of the project.

4.4. Noise map

Noise levels, as could be expected, are highest near the source of noise and decrease further away. Due to reflections around buildings, thus higher noise levels near buildings, noise contour lines are pulled towards the buildings where reflections exist. The inhibitory effect behind buildings could be seen also due to the reflective nature of buildings

The previously mentioned spikes at reflected points were found in the lower right building block of figure 4.5. At this point, the effect of these spikes on the noise map can not be determined. A comparison with the noise map after spikes are fixed would give a better idea of how spikes affect the output. The result for scenario 1, a suburb block with a single road, is shown in figure 4.5. The result for scenario 2, a rural area with no buildings and a 15.5m height difference, is shown in figure 4.6 The result for scenario 3 was not computed using this technical procedure because the constrained TIN provided had missing triangles and severe outliers which raised an error while walking the TIN.

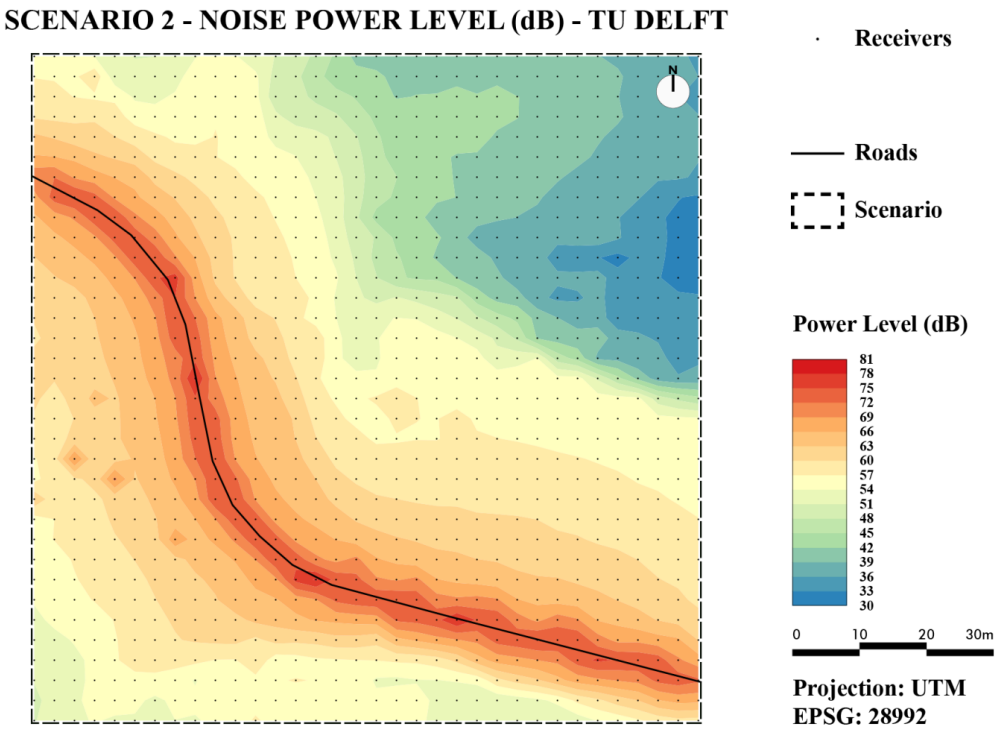


Figure 4.6: Noise map for a rural land (Scenario 2), with contour representation.

5
Quality Assessment

In this chapter, a quality assessment is done on the results shown in the previous chapter. Not only the results are assessed, but the algorithm itself as well.

5.1. Input Data

Dealing with geo-data means dealing with data imperfections that cause disruptions in the general pipeline of the workflow. Identifying these imperfections entails additional pre-processing steps; however, these steps allow the code to run without errors and with fewer conditions to check. The following table presents data issues found in the initial input data.

Ground-type data set	
Data issues	Fix
invalid polygons: self-intersecting	QGIS: buffer by 0
Bodemfactor == None	Skip

Table 5.1: Ground-type data set clean-up.

Buildings data set	
Data issues	Fix
no unique identifier for LoD1.3 (2 parts have same bag_id)	give id in qgis before starting under ”part_id” column
h_maaiveld == None and h_maaiveld < h_dak	Skip

Table 5.2: Buildings data set clean-up.

Fixing these issues was enough to be able to merge those two datasets and to generate Scenario 1, which is a simple representative neighbourhood . However, both datasets were not a perfect match, and joining them in more complicated scenarios, not necessarily covering larger areas, caused multiple problems. Most of the problems were caused when clipping the ground-type dataset with the buildings dataset such as dangling segments, spikes in polygons, and sliver polygons. Fixing these problems created gaps, and fixing gaps resulted in topological inconsistencies. Using our code with this data would either result in a ”Topological inconsistency” error or an infinite loop. Since cleaning so much data inconsistencies was not in the scope of the project, constraining the TIN for the rest of the scenarios was outsourced.

Outsourcing the constrained TIN also came with some issues. The semantic constrained TIN provides coordinates with up to three digits after the comma whereas the buildings shapefile provides coordinates with up to six digits after the comma. This precision discrepancy causes problems with identifying the triangle in which the reflection point exists. A tolerance was introduced; however, it is

not enough and it is better to have consistent precision in datasets or output with the constrained TIN its respective buildings dataset.

5.2. Quality of Noise Maps

The methodological comparison between the two technical procedures (the RIVM one and the TU Delft one) was carried out having the noise power level datasets as inputs. Since these were discrete point shapefiles with power level values at precise locations, a first step for comparison was interpolating the noise phenomenon in order to generate a continuous field.

This procedure was carried out in a GIS environment, with the possibility of interpolating values by means of an Inverted Distance Weight (IDW) or a Linear TIN interpolator. Between these two, the second one displayed better results, not only because the points were already pre-defined in a grid structure, but also because the IDW interpolator demands a search radius as an input, and choosing a value for this parameter was problematic: if the search radius were too small or too large, the continuous field would present undesired artifacts.

Therefore, for each scenario, the TIN interpolator was applied for both point datasets (the RIVM one and the TU Delft one). Both datasets had the same amount of points at the same position in order to guarantee the verisimilitude of the comparison, and no other user-defined parameter as chosen. The continuous fields are then represented with the same colour ramp, ranging from 30 dB to 79 dB, i.e. the minimum and maximum values for all fields under analysis. The homogeneous treatment for all maps guarantees the comparison among scenarios and between the two technical procedures.

Successively, the absolute power level difference is generated with map algebra, subtracting, for each pixel, the RIVM values from the TU Delft ones (TU Delft – RIVM). This first comparator generates one new continuous field per scenario, named Noise Power Level Difference (dB), which expresses how far the results generated by the synthesis project are from the currently method used by RIVM.

Albeit interesting to be compared in absolute terms, for the purpose of quality assessment, it is also advantageous to confront both results in terms of relative noise power level difference. This is achieved by subtracting values of one field from the other one, and dividing the difference with these subtracted values ((TU Delft – RIVM) / RIVM). This new field is thus normalised to the range 0 – 100.

Finally, the histogram of the Relative Noise Power Level Difference indicates how far the set of pixels is from the 0% value, i.e. the case in which there is no difference at all. The standard deviation value makes possible to measure the spread of the curve, indicating the closeness of TU Delft results with respect to the RIVM ones.

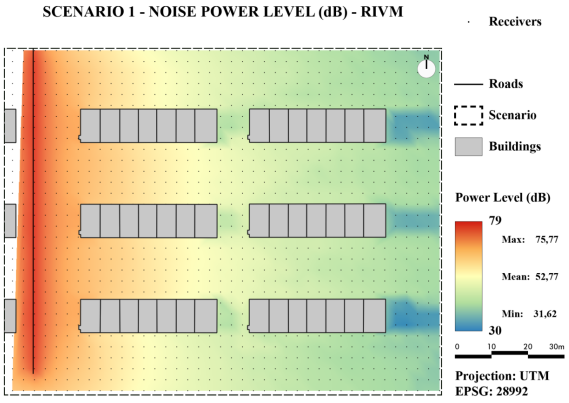


Figure 5.1: Noise Map of Scenario 1 by RIVM

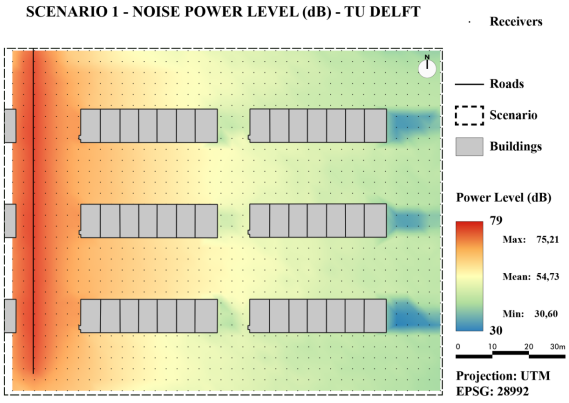


Figure 5.2: Noise Map of Scenario 1 by TU Delft

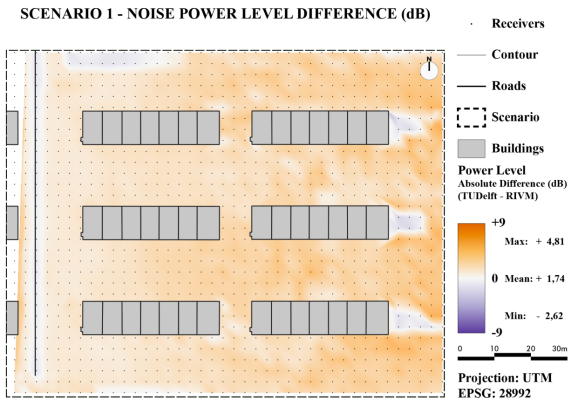


Figure 5.3: Power Level Difference in dB Scenario 1

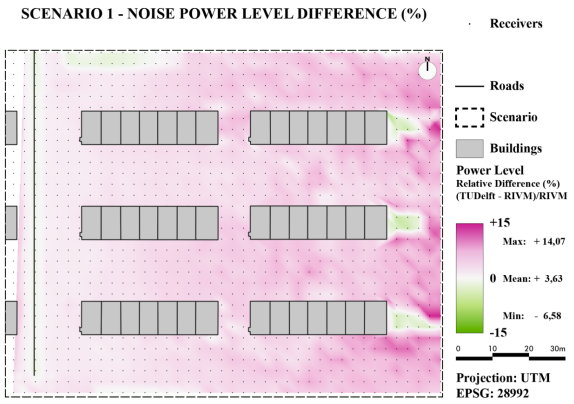


Figure 5.4: Power Level Difference in % Scenario 1

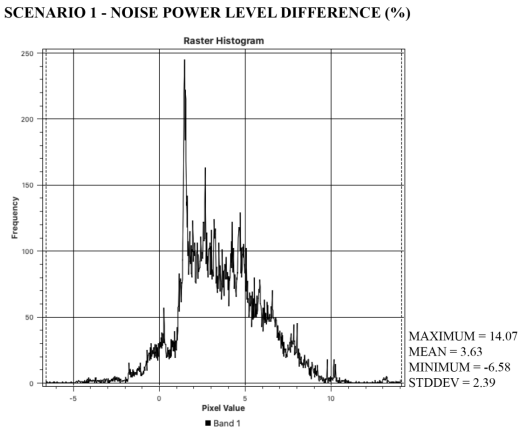


Figure 5.5: Power Level Difference Scenario 1

The comparison will start with Scenario 2 because it only deals with direct paths. A qualitative comparison shows that the distribution of the sound levels as well as the sound range is similar between both methods. However, the tested method seems to have more local minima and maxima whereas the RIVM method seem to be more continuous. This could be attributed to the use of the TIN instead of height lines. More tests should be done to verify this conclusion.

A quantitative assessment is carried out based on the histogram in figure 5.5. The majority of pixels are within plus or minus one decibel from zero. This means that the difference is mostly attributed to the extra details provided from the TIN.

In Scenario 1, a qualitative comparison shows that both methods present the same distribution of sound levels and sound range. However, in flat reflecting areas, the RIVM model tends to have a lower noise level as we get further away from the source. In the absorbing areas the tested method tends to generate lower noise levels The tested method seems more susceptible to local height differences. What can also be seen in figure 5.4 is that in occluded areas the RIVM model tends to a higher noise level. These areas will receive most of their noise coming from reflections. As described, the new constrained tin caused spikes for reflected cross sections, which occur much more on the buildings in the lower right side. These spikes will cause the reflection to have a smaller impact on the noise level. This could be the explanation for the lower noise levels in occluded areas.

A quantitative assessment, based on the histogram in figure 5.10, shows that the majority of pixels are within plus or minus one decibel from 1.5 dB. The only major difference between the two scenarios are the introduction of reflections. These reflections could lead to such differences since a longer distance

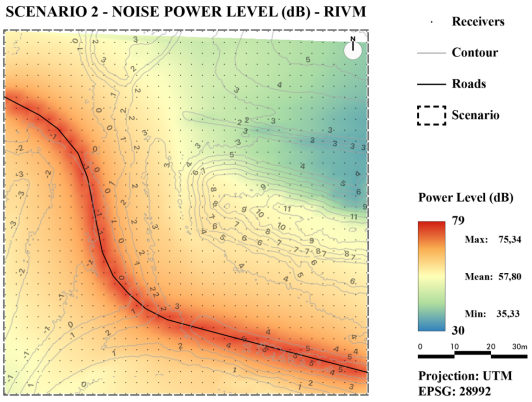


Figure 5.6: Noise Map of Scenario 2 by RIVM

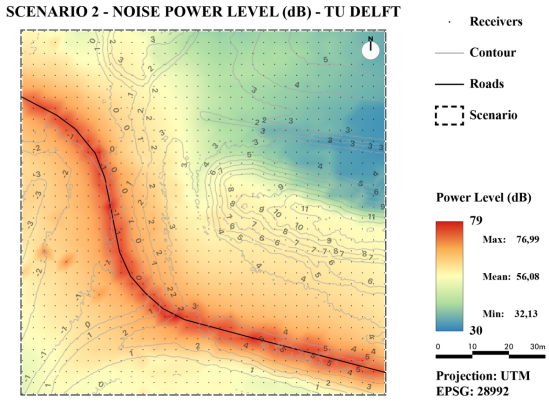


Figure 5.7: Noise Map of Scenario 2 by TU Delft

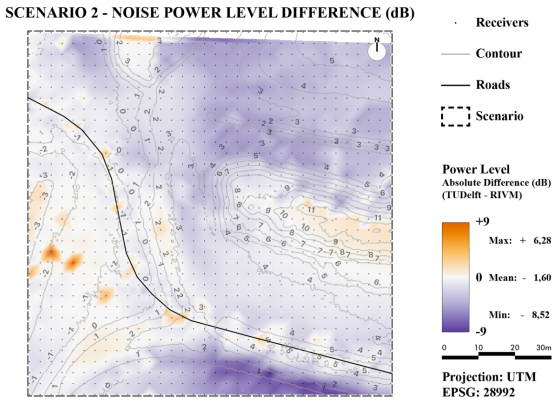


Figure 5.8: Power Level Difference in dB Scenario 2

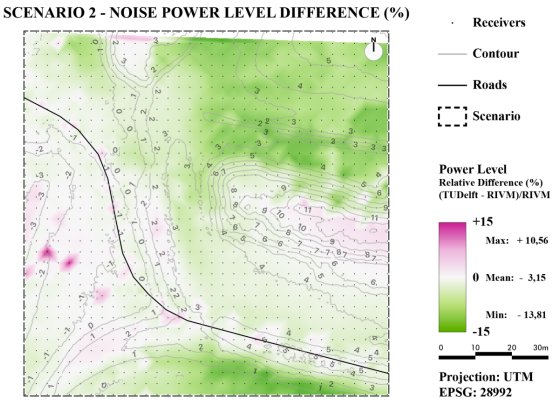


Figure 5.9: Power Level Difference in % Scenario 2

is travelled in reflections and the details from the TIN could finally add up and create higher noise levels.

For scenario 3, the noise map could not be produced using the developed method, as there were inconsistencies in the TIN as described in section 5.1. Therefore, no comparison could be made.

These comparisons clearly show how similar the outputs are between both approaches. This proves that the current method of modelling noise can be replaced with using the TIN directly.

5.3. Performance

While the performance of the program was not a big concern for the project, since it is only a proof of concept, it is worth taking a short look at it.

Both the construction of the constrained TIN and post-processing steps will not be evaluated here since they are irrelevant for the task at hand. These steps will be performed by others using their own technical procedures. Therefore, their performance is not relevant.

This test was run on Scenario 1, an area of 11400 square meters with a total of ten receiver points.

Quick Numbers:

1. Total Time: 2916.57 seconds
2. Reading Time: 0.74 seconds

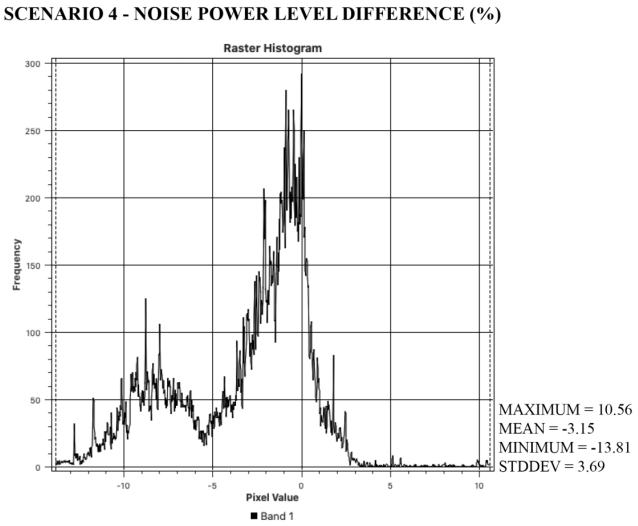


Figure 5.10: Power Level Difference Scenario 2

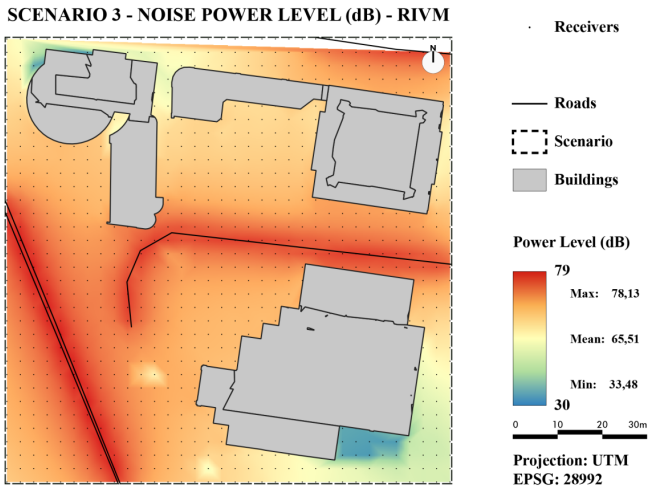


Figure 5.11: Noise Map of Scenario 3 by RIVM

3. Source Point Time: 125.55 seconds
4. Reflection Path Time: 183.32 seconds
5. Cross-Section Time: 1340.31 seconds
6. Writing Output Time: 1266.65 seconds

From the quick numbers, it becomes clear that there are 2 parts of the program that take up the majority of its time. The first of which is generating the Cross-Sections. When broken up into generating Cross-Sections for direct paths and for reflected paths, they have a run time of 661.97 seconds and 678.40 seconds respectively. For direct paths, an optimised technique is used. For all source points $(S_1, S_2, ..., S_n)$ on a single ray from the receiver point (R) only one Cross-Section (CS) is created, mainly for the furthest source point (S_n). This is the reason they are sorted at the start. For any other source point (S_i) that lies on CS created between S_n and R, the Cross-Section is simply split at S_i, creating a new Cross-Section in the process. This allows for simply copying the data from one CS to another and prevents the need to re-calculate. However, for the reflected paths there is no optimization done at this point. In the quick numbers this optimization is not represented because there is only one (1) source

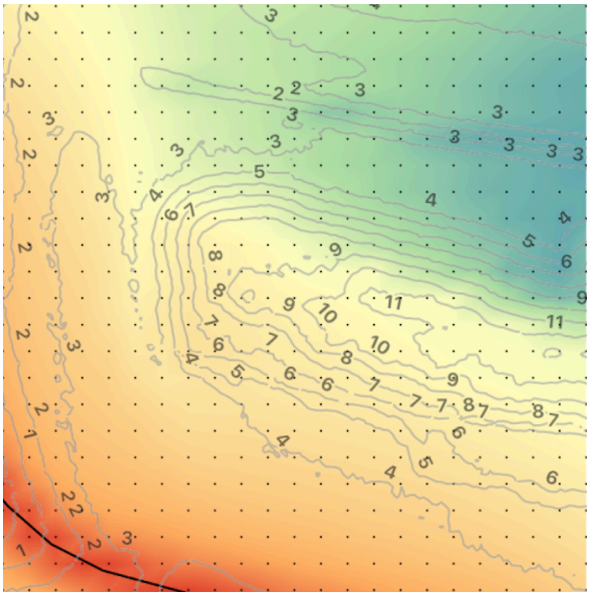


Figure 5.12: Zoomed In Area Scenario 2 by RIVM

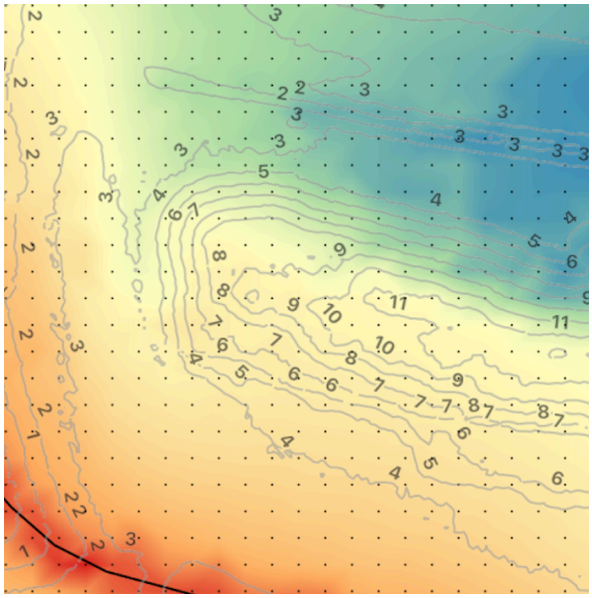


Figure 5.13: Zoomed In Area Scenario 2 by TU Delft

segment. This means that the optimization for direct paths has no benefit as there are no collinear direct paths. However, in other scenarios it provided a great benefit.

The other section that takes up a lot of time is writing to the output files. This is mainly because of the need to write a single XML file for each path, be it a direct or a reflected path. Since writing to disk is already expensive, it is not a surprise that this will take a long time. However, writing to files will not be needed when implementing the algorithm in C. This is because the Test_cnossos functions can than be accessed directly.

6
Research Conclusion

In this chapter, the proof of concept will be discussed. Furthermore, the recommendations for further research are mentioned.

6.1. Discussion

6.1.1. Proof of concept

The proof of concept in this report is an empirical assessment of the possibility to use a TIN for 3D noise modelling.

The genesis of the concept

As mentioned in section 1.4, using height lines generated either manually or through the use of a TIN is not an efficient approach for 3D noise modelling. Moreover, since this approach is highly monopolised by commercial companies in the Netherlands, the community of noise modelling is seeking a more open-source approach, in line with the open data policy of the EU. Finally, a TIN is already used to generate the height lines and it is easily and freely generated from the point cloud dataset AHN3.

The hypothesis

The hypothesis set at the beginning of this project is: *Using a TIN directly allows automated 3D noise modelling according to the guidelines of CNOSSOS-EU.*
The TIN used represents a DTM of the area studied. When the hypothesis was formulated, this TIN was initially considered as an LoD 0 TIN [6], which means that the triangles inside the TIN do not hold any semantic information. The semantic information, ground absorption index or building information, would be extracted later on from different datasets.

The intervention

In order to prove this hypothesis, the following steps were followed:

1. Extracting cross-sections of the DSM representing the area studied
2. Visually checking these cross-sections in Rhinoceros software
3. Checking the validity of these cross-sections in Test_cnossos software
4. Running the Test_cnossos software with these cross-sections, creating a noise map and comparing it with a noise map using height lines

For the first step, it was concluded to create an LoD 2 TIN [6], which constrains the TIN to buildings and ground absorption and stores semantic information in the triangles without creating a DSM (which includes not only the natural bare terrain but also the human-made objects). This simplifies the processing steps and increases the consistency and time efficiency of the algorithm.

The proof

Getting results from the Test_cnossos software and comparing them to noise maps from the height lines approach proves that a semantic TIN can be used directly for automated 3D noise modelling. Therefore, an extra step to generate height lines becomes unnecessary. For now, time-efficiency is not taken as a criterion because, depending on the resolution of the height lines, the performance can vary greatly. However, with optimisation and the use of a different programming language like C++, using height lines can become obsolete. In addition to the automation advantage of using a TIN over height lines, using a TIN gives more accurate cross-sections and thus more accurate noise levels in Test_cnossos software.

6.2. Recommendations for Further Research

The scope of this project was to conduct scientific research to check if it was possible to derive the noise propagation paths directly from the TIN instead of the ground height lines. However, some elements are not included in this project and could be valuable in further research. In this project, only direct paths, first-order reflections, and vertical diffraction were taken into account. These are all modelled in the horizontal plane and are therefore in homogeneous conditions. There is also horizontal diffraction and higher-order reflections that could be modelled. This is quite complex and not required, therefore it is not a part of this project. However, horizontal diffraction contributes to the sound levels and therefore, could be interesting in future projects.

In the scope of this project, only roads were taken into account. Besides these noise sources, railways, industrial buildings and air planes contribute to noise pollution. Therefore, these options could be explored as well in future research. On top of that, there is a difference between busy and calm streets that are not taken into account in this project. This could be an option to explore in further research.

In the current algorithm, buildings with an LoD1.3 are taken into account. However, it could be interesting for further research to integrate LoD2 buildings. This way it will be a truly 3D path-finding algorithm in which slanted roofs are taken into account.

The absorption in this algorithm is highly deduced. Currently, a ground type is either completely reflecting or absorbing. There are no values in between. On top of that, the current algorithm does not take vegetation into account.

The study area used in this project are tiles from the AHN3. The tiles resemble an area that partially overlaps with the city of Rotterdam. It would be interesting to scale this area up since there are cities bigger than Rotterdam in Europe that have noise pollution as well.

Finally, another approach to noise modelling could be suitable. Nowadays, the virtual 3D world is more often used. Therefore, it could be an option to make use of voxelisation. Noise simulation could be visualised in 3D for a whole city for example. This could be an interesting approach to investigate in the future.

Bibliography

[1] Moscow method. https://en.wikipedia.org/wiki/MoSCoW_method. Accessed: 2020-04-27.

[2] 3D geo-information. Automated reconstruction of 3d input data for noise studies. https://3d.bk.tudelft.nl/projects/noise3d/. Accessed: 2020-04-27.

[3] ECG. Expertisecentrum geluid. https://www.rivm.nl/geluid/expertisecentrum-geluid. Accessed: 2020-04-27.

[4] European commission. Common noise assessment methods in europe (cnossos-eu). https://ec.europa.eu/jrc/en/publication/reference-reports/common-noise-assessment-methods-europe-cnossos-eu. Accessed: 2020-04-29.

[5] Stylianos Kephalopoulos, Marco Paviotti, and Fabienne Anfosso Ledee. Common noise assessment methods in europe (cnossos-eu). page 180, 01 2012.

[6] Kavisha Kumar, A. Labetski, H. Ledoux, and Jantien Stoter. An improved lod framework for the terrains in 3d city models, 09 2019.

[7] Peters, R. and Commandeur, T and Dukai, B. and Stoter, J. 3d-inputgegevens voor geluidssimulaties gegenereerd uit landsdek. https://3d.bk.tudelft.nl/rypeters/pdfs/18_geoinfo_noise3d.pdf. Accessed: 2020-04-27.

[8] RIVM. Rijksinstituut voor Volksgezondheid en Milieu. Ministerie van Volksgezondheid, Welzijn en Sport. https://www.rivm.nl/. Accessed 2020-04-27.

[9] RWS. Rijkswatersaat. ministerie van infrastructuur en watersaat. https://www.rijkswaterstaat.nl/index.aspx. Accessed: 2020-04-27.

[10] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[11] D van Maercke. Task 2: Propagation software modules; user's and programmer's guide. unpublished (comes with TestCnossos software; requests mailed to: dirk.van-maercke@cstb.fr, 2012. company: Centre Scientifique et Technique du Bâtiment (CSTB).

7 Appendix

In this chapter, some implementations that were modelled for this project but not included in the finale product are discussed.

7.1. Simplifying Data Structure

A functioning algorithm that simplifies the data structure is implemented, i.e. it removes unnecessary points. This is done according to the principles of a Douglas-Peucker algorithm, it respects point materials and extensions (see figure 7.1). However, it has not proven to be required for the program to run nor has it proven to save time (i.e. the cost to simplifying the path is not necessarily less than the decrease of sound level computation). The code is still in the program, but the function is not called for.

Another reason for not implementing is that it is not yet tested with a wide variety of scenarios. The principle behind Douglas-Peucker is that it keeps, iteratively, adding the vertices which have the highest error with the simplified model to this model. This continues until all errors are within a tolerance. This is preferred, but in the current implementation, it does not yet respect vertical walls, as can be seen in figure 7.1, vertical wall segments can be combined into one sheer line. Although it has not been tested, this could lead to an error in the computed noise level.

7.2. Constrained TIN using Triangle library

To prove the advantages of having a semantic constrained TIN, at the start of the project, a trial semantic constrained TIN was created using the Triangle Python library [10]. To get the constrained TIN, preprocessing the datasets is required. The datasets are a ground TIN LoD 0, ground type dataset, and building dataset.

1. In QGIS, buffer the ground type dataset by 0 to fix for the invalid polygons
2. Apply a difference overlay on the ground type dataset by using the LoD1.3 buildings polygons
3. Merge the difference overlay with the buildings dataset.

The resulting dataset is polygons that are either ground type or buildings, no overlaps between those two which is important for the next steps.

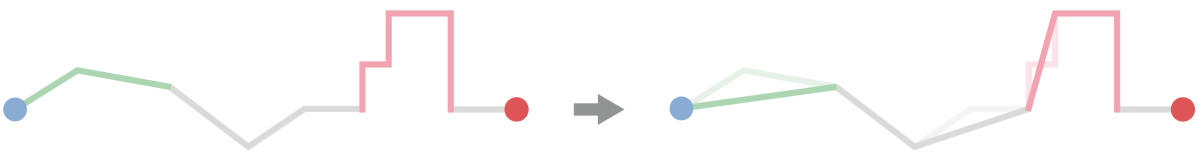


Figure 7.1: Simplification by Douglas-Peucker (Green: Grass, Grey: Road, Pink: Building, Red Dot: Source Point, Blue Dot: Receiver)

Input : Cross-Section including $vertices$, $materials$ and $extensions$;
$tolerance$ maximum error between input and output path;
Output: simplified cross-section

$simple\_path \leftarrow$ 1st and last vertex, vertex with extension, vertices where material changes;
$curser$ = 0 **while** $curser$ *has not reached end of path* **do**
  $segment \leftarrow$ segment from $vertices[curser]$ till next vertex;
  if *there are no original vertices in between* $segment$ *points* **then**
    $curser$ + 1;
  end
  $max\_offset \leftarrow$ highest offset of vertices in $segment$;
  max_offset is greater then $tolerance$ $simple\_path \leftarrow$ insert the vertex with the max_offset;
  $curser$ + 1;
end
$materials \leftarrow$ remove the materials of the removed vertices;
$extensions \leftarrow$ update with correct locations;

Algorithm 6: Implementation of simplification algorithm for cross-sections(not in use)

In Command Prompt, pip install Triangle, a python library that generates the constrained TIN with the correct input. In python, first import triangle, then read the resulting dataset (ground type + buildings) and the LoD 0 TIN using fiona.

The LoD 0 TIN is a 2.5D DTM. However, the TIN can be considered either as a 2D TIN or a 3D one. Considering as a 3D TIN is computationally expensive and the provided TIN has no vertical walls, i.e. for each (x,y) there is only one z. This means the 2.5 D TIN can be projected into 2D and as long as the 2D vertices inserted keep the same order as the 3D vertices, the triangle-base data structure (v0, v1, v2, t0, t1, t2) after constraining the 2D TIN will always refer to the correct 3D vertices making it possible to lift the 2D TIN into a 2.5D TIN easily and quickly. Therefore the first step is to project all vertices to a 2D plane, then add building and ground type segments as constraints to the TIN.

While adding the constraints, the datasets are filtered by making sure that each ground type polygon has an absorption index as an attribute and that a building has roof level higher than ground level. Along with the segments, the id of the features is extracted as semantics for the TIN to reference back to either a building class or a ground type class. For the method to work using the triangulate function from the triangle library, the attribute must be inside the boundary of the respective polygon and the attribute must be of integer type or can be transformed to integer type. To always get a point in a polygon, two vectors are created, one created by the first and second vertex of the polygon and the other created by the first and last vertex of the polygon. Those 2 vectors are summed, the result is multiplied by epsilon and then added to the first vector again.

In the case of a multi-polygon (this can occur with ground types because the difference overlay that might disconnect one polygon), each polygon gets a unique id.

Finally, the triangulate function of Triangle is applied. This step outputs all vertices from the TIN, ground type polygons and building polygons, all triangles created as a list of 3 indices that refer to the vertices list (2D or 3D), a list of triangle neighbours as 3 indices referring to the triangles list, and the attributes per triangle. However, we noticed that the function adds Steiner points in 2D that don't exist in the 3D vertices list which will create problems when triangles have indices to non-existing vertices. For now, the function seems to add these points at the end of the list. Therefore, the extra points are interpolated and added to the 3D vertices list.

Input : shapefile with buildings and ground polygons, obj file of an LoD0 TIN
Output: vertices, triangles, neighbours of triangles, and semantics of each triangle in the constrained TIN

$vts \leftarrow$ vertices of LoD0 TIN $vts \leftarrow$ add vertices of the shapefile;
$segments \leftarrow$ segments of polygons in shapefile as constraints;
$attributes \leftarrow$ attributes of polygons;
triangulate TIN;

Algorithm 7: Create constrained semantic TIN.

7.3. Second-Order Reflections

Although second-order reflections are not calculated in this synthesis project, it is worth explaining the reason why such paths were not considered in this 3D noise simulation.

Unlike first order-reflection, multiple-order-reflection are the ones in which there is a combination of reflection points (REF1, REF2, … REFn) such that, starting from the source point (S), the sound wave passes through all these reflection points and then finally hits the receiver point (R).

This fact creates a paradoxical situation, in which determining the position of one reflection point depends on the position of the successive reflection point and vice versa. Therefore, second-order reflections do not have a straightforward approach like the one of the first-order reflection.

For second-order reflection, there are two unknown points, which creates an under-determined system of equations. The algorithm, therefore, would have to perform a series of trials, in which candidate points would be tested for each building wall. If a candidate point (C) is capable of reflecting the sound wave in such a way that another point in a facade (REF) reflects the wave towards the receiver point (S), then the path is also possible.

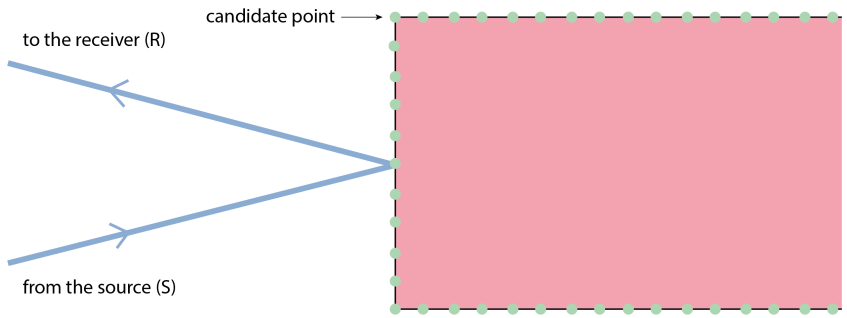


Figure 7.2: Candidate points, located along a building walls, to be tested as valid reflection points second-order reflection.

The heuristics of such an approach increase the computing time of the algorithm according to the number of candidate points to be tested. The more candidate points present in the model, the more second-order reflections are computed. However, the heuristics would also require a certain threshold from the receiver point.

The decrease of computing efficiency caused by multiple-order reflection does not compensate for the overall accuracy in terms of power level at the receiver points. Moreover, in a meeting with RIVM and RWS, it was explained that approximately 90% of the noise effect is due to straight paths and first-order reflections. Additionally, even if second-order reflection is important for a certain scenario (such as street canyons and open roofs), its results are not realistic, since facades, in reality, are not perfect mirrors. Therefore, it is worth working with power levels that are underestimated by a few decibels

rather than losing computing efficiency and accuracy.

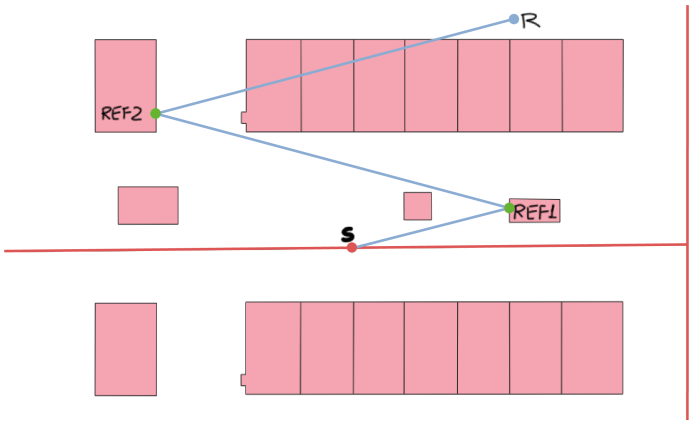


Figure 7.3: Computed second-order reflection. Given a source (S) and a receiver (R) points, finding the reflection points REF1 depend of the position of REF2, and vice versa. Considering the approximation of this procedure, candidate points are tested to check if they produce such a ray that gets close to the receiver point, considering a threshold.