NTNU
Norwegian University of
Science and Technology

Sun Glare Detection and Visualization

Kjetil Aune

Master of Science in Computer Science
Submission date: June 2017
Supervisor: Jo Skjermo, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Presence of sun glare while driving can be both annoying and potentially dangerous. This paper presents a method for calculating the presence of sun glare on roads. A plug-in for QGIS that can be used to visualize the results and perform analysis on the data is also presented. By calculating the sun's position relative to a point on the road and checking whether or not the surrounding terrain is occluding the sun, the system can calculate the visibility of the sun. Having established visibility of the sun, road data is analyzed to determine if the sun will be a distraction to the driver at any given time. The results of the glare detection are written to a shapefile and returned to the plug-in for visualization and examination. Experiments on the validity of the glare detection have been conducted, and these show that the system is able to correctly detect sun glare. Obtained results can be used to assess the safety of present or future roads, and the system can be a useful tool for evaluating potential safety measures.

Sammendrag

Tilstedeværelsen av solblending kan være både irriterende og potensielt farlig. Denne oppgaven presenterer en metode for utregning av solblending på vei. En plug-in til QGIS som kan brukes til å visualisere resultatene fra solblendingsutregningene blir også presentert. Plug-in'en har også verktøy for å analysere resultatene. Ved å regne ut solens posisjon relativt til et punkt på veien, og sjekke om det omkringliggende terrenget skygger for solen, kan systemet regne ut om solen er synlig. Om solen er synlig, blir veidata analysert for å se om solen vil være et distraksjonsmoment for bilføreren. Resultatene av solblendingsdeteksjonen blir skrevet til en shapefile-fil og returnert til plug-in'en for visualisering og undersøkelser. Eksperimenter på systemets validitet har blitt gjennomført, og de viser at systemet kan korrekt oppdage solblending. Resultater funnet ved analyse av dataen kan bli brukt til å vurdere sikkerheten til nåværende og fremtidige veier, og systemet kan være et nyttig verktøy for å evaluere potensielle sikkerhetstiltak.

CONTENTS

I Introduction 1

II Background 2

III Methodology 3

III-A System Workflow . . . 3

III-B The Sun Glare Detection System . . . 3

III-B1 Workflow . . . 3

III-B2 Calculating the Sun's Position . . . 4

III-B3 Reference systems . . . 4

III-B4 Terrain Extraction . . . 5

III-B5 Occlusion test . . . 6

III-B6 Road direction check . . . 7

III-B7 Optimizations . . . 8

III-B8 Shapefile Generation . . . 8

III-C Plug-in . . . 9

III-C1 Plug-in Introduction . . . 9

III-C2 Rendering . . . 9

III-C3 Information Extraction Tools . . . 10

IV Evaluation and Results 10

IV-A Sun Position Algorithm . . . 11

IV-B Geographical Occlusion . . . 11

IV-C Road direction . . . 12

IV-D Visualization in QGIS . . . 12

IV-D1 Filtering . . . 12

IV-D2 CRS Conversions . . . 13

IV-E Complexity . . . 13

IV-E1 Generation Time . . . 13

IV-E2 Azimuth and Aperture overlap . . . 13

V Future Work 13

VI Conclusion 14

References 14

Appendix A: Installation and Build Guide 17

A-A Sun Glare Visualization Plug-in Installation . . . 17

A-B Sun Glare Detection System Build Instructions . . . 17

Appendix B: Configuration File 19

B-A MORNING_RUSH_START . . . 19
B-B MORNING_RUSH_END . . . 19
B-C EVENING_RUSH_START . . . 19
B-D EVENING_RUSH_END . . . 19
B-E TIME_RESOLUTION . . . 19
B-F UTM_ZONE . . . 19
B-G HORIZONTAL_GLARE_APERTURE . . . 19
B-H VERTICAL_GLARE_APERTURE . . . 19
B-I EYE_HEIGHT . . . 20
B-J YEAR . . . 20

Appendix C: Images of the Plug-in Functionality 21

C-A Generating Glare Data . . . 21
C-B Open Existing Shapefile . . . 22
C-C Select Values from Time Range . . . 22
C-D Select Value Range for Attribute . . . 22
C-E Select Top X points . . . 22

Appendix D: Shapefile 25

Sun Glare Detection and Visualization

Kjetil Aune
Department of Computer and Information Science
Norwegian University of Science and Technology
Sem Sælands vei 9, 7491 Trondheim, Norway
kjetiaun@stud.ntnu.no

***Abstract*—Presence of sun glare while driving can be both annoying and potentially dangerous. This paper presents a method for calculating the presence of sun glare on roads. A plug-in for QGIS that can be used to visualize the results and perform analysis on the data is also presented. By calculating the sun's position relative to a point on the road and checking whether or not the surrounding terrain is occluding the sun, the system can calculate the visibility of the sun. Having established visibility of the sun, road data is analyzed to determine if the sun will be a distraction to the driver at any given time. The results of the glare detection are written to a shapefile and returned to the plug-in for visualization and examination. Experiments on the validity of the glare detection have been conducted, and these show that the system is able to correctly detect sun glare. Obtained results can be used to assess the safety of present or future roads, and the system can be a useful tool for evaluating potential safety measures.**

I. Introduction

Challenging weather conditions are one of the major safety hazards while driving. Slippery roads and low visibility cause numerous traffic accidents each year [14]. One of the main reasons for low visibility is the presence of sun glare [5]. When the sun is low in the sky, it can have a big impact on a driver's ability to make out details in front of the vehicle. It can be hard to spot another car breaking, a speed-limit change or a pedestrian. This can quickly lead to dangerous situations. Even when the sun is not close to the horizon, it can have a great impact on road safety. If a car is driving up a steep hill, the angle between the vehicle and the sun could be small enough for sun glare to be present. After heavy rainfall, the wet surface of the road can also reflect the sunlight. This type of sun glare is just as, if not more, dangerous, since the driver can not see the road at all. However, glare caused by wet roads is not accounted for in this paper. The amount of sun glare can also vary from vehicle to vehicle. If a car has an old windshield with lots of scratches, or the windshield is dirty, the sun glare can be even more prominent [17]. Windshields with these properties will scatter the sun light over a larger area (i.e. veiling glare), thus reducing a driver's visibility.

To counteract the effect of sun glare, precautions have been made. Today, most cars have a sun visor attached above the windshield on the driver and passenger side of the car. By using this visor, the sun's light can be blocked out. A downside to this is that the driver's field of view is significantly reduced. This can in turn lead to other dangerous scenarios. Some vehicles also have a tinting in the top part of the windshield. This acts like a pair of sunglasses, making the sun glare more tolerable. However, this tinting usually covers only a small part of the top of the windshield. When the sunlight is coming in from a lower angle, it does not protect the driver anymore.

Several other safety measures could be taken against sun glare. In the planning phase of road building, one can calculate the amount of sun glare on a particular road segment. If the planned road segment is highly exposed to sun glare, a change in the direction of the road can lead to less glare. For roads that have already been built, however, it might be too expensive to change their direction. If calculations show that these roads are exposed, dynamic road signs can be put up. These signs can have built-in sensors that detect sun light. If the amount of luminance reaches a certain threshold, the signs can display messages telling drivers to slow down. Another possibility is to find out where the sun light is coming from and plant trees or bushes that can occlude the sun in highly exposed places. However, to get any of these plans to work, one needs to be able to calculate the presence of sun glare on the road.

This paper will present a method for calculating the presence of glare on the roads and a plug-in for the GIS tool QGIS. By using road- and terrain data, the method calculates if roads have sun glare or not at any given time. Solar position and a vehicle's relative position to the sun takes part in deciding whether or not there is glare at a given point. After doing these calculations for a large geographical area, the plug-in can be used to render the results from the calculations on top of a map in QGIS. Drawing the results on a map gives a visual overview of what roads or city districts are most exposed to sun glare. Several tools are developed to aid the user in understanding the data. Analysis of the glare detection results can be helpful when deciding if any safety measures should be taken. Having a good tool for this purpose can greatly help with increasing both the efficiency and the correctness of the analysis. This can in turn lead to a reduction in the amount of sun glare on the roads and thus lowering the risk for accidents.

This paper is written in cooperation with the Norwegian Public Roads Administration. They are fully aware of the dangers of sun glare. However, they lack tools for analyzing glare. Work done in this paper aims at developing methods that employees at the Norwegian Public Roads Administration can use to easily gain a better understanding of what areas are especially exposed to glare. By providing a QGIS tool, the goal is that both technical and non-technical personnel can work with glare data and improve the safety of the roads.

Following this introduction, there will be more argumentation as to why sun glare is an important issue and clarification of relevant topics. Then, we will discuss the approach taken towards making the plug-in, with a thorough explanation of the underlying algorithm for detecting glare and also the visualization of the results. Finally, results will be presented,

before discussing future work and concluding.

II. Background

There has been a lot of previous efforts in understanding the effects sun glare has on drivers. Churchill, Tripodis and Lovell [6] developed a geometric model for identifying sun glare along an arbitrary highway segment. They used this model to detect what times and days of the year sun glare was present. Using this model, they wanted to see if there was a connection between sun glare and freeway congestion. Their results show that the presence of sun glare did affect the mean speed of the vehicles on the road. They conclude that changes in speed due to sun glare is a factor in highway congestion. Their model is in many ways similar to the one presented in this paper. However, their model does not take the surrounding topography into account, and thus reports more cases of sun glare than what there might actually be due to terrain occlusion.

Jurado-Piña and Mayora [11] also present a study on when sun glare can affect drivers. Unlike [6] they also consider the characteristics of the surrounding environment. One of the interesting calculations done in this paper finds the maximum tolerable sun glare caused by the angle between the sun and the driving direction. By using an equation for disability glare found in [13], they estimated thresholds for what constitutes sun glare. There are biological differences in the eyes of people at different ages. Therefore the study states that there should be different thresholds for people at different ages. According to the study, glare occurs if the sun is within a specified angular distance between the drivers line of sight and the sun. For a 40 year old driver, this angular distance is 19° and for a 60 year old driver it is 25°.

Hagita and Mori [10] studied the effect sun glare had on collisions between turning vehicles and pedestrians and bicyclists in Japan. Accident data and weather data was gathered to study the relationship between accidents and sun glare. Findings reveal that during daytime and in clear weather, the number of accidents was higher than at night or when it was cloudy. They also discovered that the amount of collisions was drastically higher when the sun was below 45° above the horizon and within 90° of the driving direction.

A study on a driver's field of visual detection was conducted by Zwahlen [19]. A geometric model is presented to investigate how far from a driver's line of sight a road sign should be, for the driver to quickly detect the sign and its contents. He concludes that the angular displacement of signs should not exceed 10°.

For a system like the one presented in this paper, it is crucial to be able to accurately compute the sun's position by using only geographical coordinates (longitude, latitude), date and time as input. Blanco-Muriel et al. [4] present an algorithm that takes the mentioned parameters as input and returns the horizontal coordinates of the sun at the given time and place. From the input, they convert the coordinates from ecliptic- to celestial coordinates, before further converting them to horizontal coordinates. The calculated position of the sun is the actual position at the given time. The calculated position does not give the apparent position that is caused by atmospheric refraction. The correctness of the results in the paper is based on the assumption that the difference between Universal Time and Dynamic Terrestrial Time grows steadily by one second per year on average. Results were validated from 1999 to 2015, and the results show an average error of -0.008 minutes of arc with a standard deviation of 0.107 minutes of arc. However, for the practical applications of the system presented in this paper, an error of 60 or even 120 minutes of arc would still yield acceptable results and thus the algorithm presented in [4] is used in this paper.

The Norwegian Mapping Authority provide an extensive data set of the Norwegian road network [3]. All the roads in Norway are well documented, and all the data is publicly available. The data is issued in the SOSI-format, which is developed by The Norwegian Mapping Authority themselves. For roads, the files contain objects for different road segments. An object contains information about the name and ID of the road, what type of road it is, etc. The object also contains a list of coordinates for points along the road segment. Each coordinate point contains the northing and easting value for the point along with the elevation of the point. These coordinates are along the centerline of the road.

The Norwegian Mapping Authority also provide terrain data for all of Norway [2]. The data comes in the form of Digital Elevation Model-files and the resolution of the data used is 10-by-10 meters. The terrain data is split into patches of 50-by-50 kilometers (i.e. a grid with 5000-by-5000 cells). Each of the cells contain the elevation above sea level.

When working with geospatial data, it is convenient to use a Geographic Information System (GIS). GIS systems are used to manipulate and analyze both spatial and geographic data. The relevant data is imported into the system as layers, and one can do analysis on one or several layers at a time. User-defined queries can be performed to gain a better understanding of the data. The data can also be explored and edited by using the many tools that come with the GIS system. The result of these actions are then displayed back to the user. QGIS is a free and open-source Geographic Information System, and is therefore the chosen GIS system for this paper. QGIS runs on Linux, Unix, macOS, Windows and Android [15]. In addition to its core functionality, QGIS has the possibility of being extended with plug-ins. These plug-ins are written either by QGIS developers or independent developers. Plug-ins can easily be installed from QGIS' plug-in repository.

Coordinate reference systems (CRS) are important when working with geographical data. A CRS provides a map projection for plotting an object on the surface of the earth, as well as having transformations to other CRSs. Standard geographic coordinates comprised of latitude and longitude use the World Geodetic System (WGS84), whereas the data from the Norwegian Mapping Authority uses Universal Transverse Mercator (UTM). Exact transformations between these CRSs are important. GIS tools, including QGIS, can do these transformations between numerous coordinate reference systems. By specifying what CRS the data is in and the desired output CRS, the GIS system automatically transforms the data between them.

When moving geospatial data between different systems, shapefiles are often used. Shapefiles are supported by most GIS systems. The shapefile specification is developed by

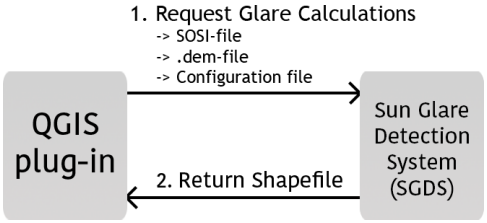


Fig. 1. The workflow of the full system.

Environmental Systems Research Institute (ESRI) and the technical description is publicly available [9]. Shapefiles store non-topological, geographical data as vector coordinates. They support several shape types (i.e. points, lines, polygons, multipoints, etc.). All items in a shapefile must be of the same shape. These items are called features, and each feature is accompanied by its vector data and additional attributes. The attributes describe the feature and are either a string or a number. Each feature can have a maximum of 255 attributes. The method presented in this paper generates glare data which is written to a shapefile and the QGIS plug-in reads the generated shapefile and visualizes the results from the glare detection.

III. Methodology

This section will present the approach taken towards creating the glare detection module and the QGIS plug-in. First, the system's workflow will be introduced. Secondly, the method for detecting sun glare will be presented. This will later be referred to as the Sun Glare Detection System (SGDS). Finally, the plug-ins visualization- and information extraction techniques will be described.

A. System Workflow

The system consists of two major parts, namely the actual sun glare detection and the visualization of results in QGIS. As seen in Figure 1, the user can use the QGIS plug-in to request a shapefile from the Sun Glare Detection System. Road- and terrain data and a configuration file is sent manually by the user together with this request. This will prompt the SGDS to run its glare detection and generate a shapefile containing the results. In turn, the file is returned to the QGIS plug-in, where the results are displayed.

B. The Sun Glare Detection System

The SGDS is a stand-alone application written in C++. It takes as input a SOSI-file with road data, a Digital Elevation Model-file with terrain data, a configuration file and a name and directory for the resulting shapefile. Appendix B shows a description of this configuration file. It is worth noting that to check a planned road for glare, one can simply generate a SOSI-file with points along this road and input this file to the system.

Calculations with the Sun Glare Detection System are done for a whole year at a time for all the road segments in the SOSI-file. The system is designed to calculate the sun glare in the rush hours in the morning and the afternoon. The start and end times for the rush hours, as well as the resolution, can be changed in the configuration file simply by changing the lines starting with *MORNING_RUSH_START, MORNING_RUSH_END, EVENING_RUSH_START, EVENING_RUSH_END and TIME_RESOLUTION* and provide the new configuration file to the system. The defaults times are from 7-9 a.m. and 3-5 p.m, with a resolution of 5 minutes. This means that for every day of the year, each road segment is checked for glare every five minutes between the start and end of both of the time intervals.

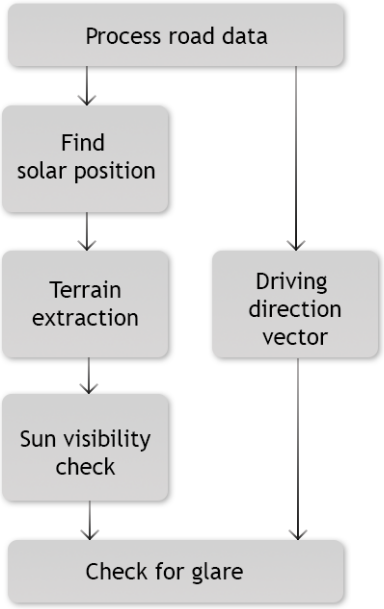


Fig. 2. The workflow of the SGDS.

Following is a thorough explanation of the glare detection for a point along the road. After this, optimization considerations will be discussed, before the shapefile generation is described.

1) Workflow: To be able to detect sun glare, there are three main parts that need to be taken into account:

- The sun's position
- The surrounding terrain
- The driving direction

A simplification of the glare detection for a single point on the road can be seen in Figure 2. First, the system starts off by interpreting the road data and extracting information regarding the road segment of interest. Then it calculates the sun's position relative to a point on the road segment. Next, the terrain that is between the vehicle and the sun is located and an occlusion test is performed to see whether or not the sun is actually visible from the view point. From the road data processing, the slope and direction of the road is determined. This produces the driving direction vector. This vector and the sun's position is used to see if there is sun glare at the given point on a given time and date.

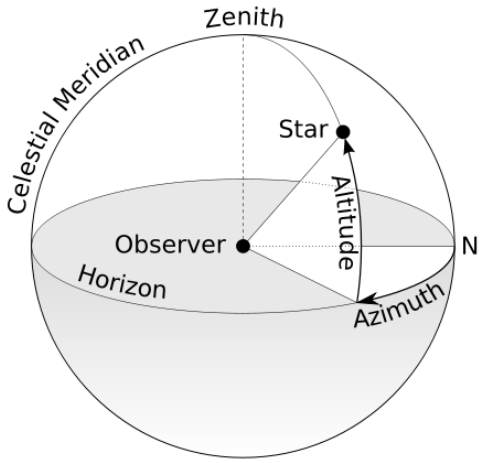


Fig. 3. Visualization of the horizontal coordinate system, showing both azimuth angle and elevation.

2) Calculating the Sun's Position: The algorithm described in [4] gives fairly accurate results in calculating the horizontal coordinates of the sun. Horizontal coordinates is a way of plotting a celestial object as seen from earth. To do this, two angles are required: the azimuth angle and the elevation angle. As seen in Figure 3, the *azimuth angle* is the number of degrees going clock-wise around the horizon from north to the celestial object. *The elevation* is the number of degrees from the horizon to the celestial object. The elevation has positive direction going upwards, with zero degrees being at the horizon and 90° being directly above the observer. With these two angles, it is easy to calculate the observer-sun unit vector.

To be able to calculate the position of the sun as seen from a point on earth, the algorithm requires some input. The algorithm needs to know the observer's position as well as the time and date for the calculation. Coordinates must be provided in the latitude-longitude format. Information regarding time zone and daylight savings time is also required. The algorithm uses GMT+0 as its reference point, and therefore time zone information has to be provided in the GMT format. Correction of the provided time is done as follows: if the point lies in GMT+1, one hour is then subtracted from the time. If daylight savings time is currently being used in that location, one more hour is subtracted. Since the SGDS performs calculations for a whole year, it is important that it adjusts for daylight savings time automatically. The regulation found in [16] defines when there is daylight savings time in Norway, and these are the boundaries used to provide the correct time to the sun position algorithm.

Figure 4 shows a plot of the azimuth angle for June 1st and December 1st in 2016. Since the sun rises in the east and sets in the west, the azimuth angles in both summer and winter should be pretty similar, which is confirmed by the graph. Figure 5, however, shows that there is larger differences in elevation between summer and winter. In June, the sun's maximum elevation is close to 50°, whereas in December it is approximately 5° at its peak. This causes shorter days during winter and longer days during summer. As a result, it is clear that sun glare will happen at different times of the day in different seasons.

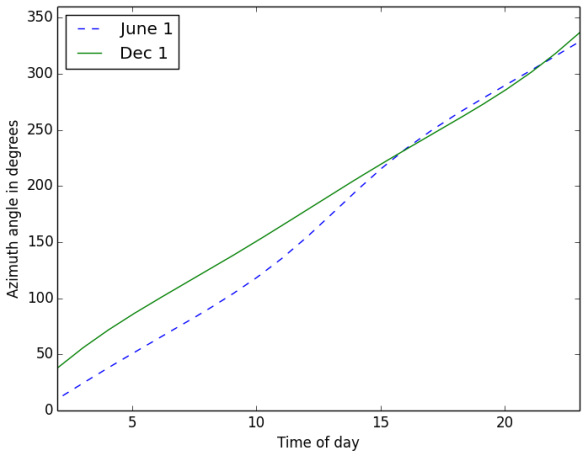


Fig. 4. A plot of the azimuth angle on June 1st and December 1st 2016 in Trondheim, Norway.

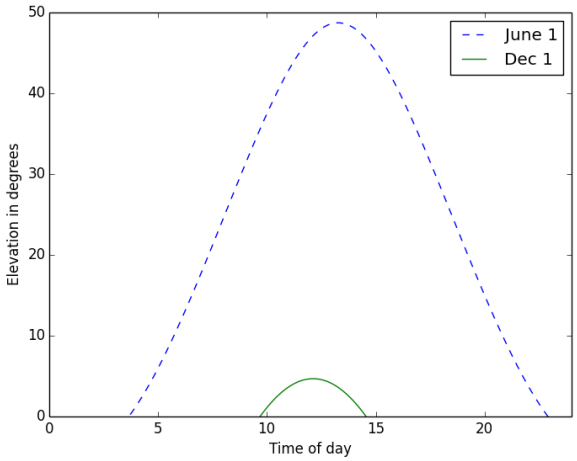


Fig. 5. A plot of the elevation angle on June 1st and December 1st 2016 in Trondheim, Norway.

3) Reference systems: Since this system collects data from several different sources, it is important to decide on a common reference system. The coordinates for road- and terrain data are provided as northing and easting coordinates. For calculating the sun's position, latitude and longitude is required. The coordinates of the sun's position is retrieved as horizontal coordinates. The azimuth angle can be represented as polar coordinates, and the terrain data is best described in the Cartesian coordinate system. In this section, we will look at the reference system used for this system, and how the conversions are done.

a) Coordinate conversion: Terrain data and road data provided by The Norwegian Mapping Authority use Universal Transverse Mercator (UTM) projection for its coordinates. Using the World Geodetic System, a position on earth is represented by degrees north and degrees east. These two values are enough to plot an object anywhere on earth. However, when using UTM, some more information is required. A point on earth is represented in UTM using the UTM zone number

along with values for northing and easting. The earth is divided into zones that run vertically from pole to pole and each zone is 6 degrees wide. The northing value is the number of meters from equator to the specified point. The easting value is the number of meters from the central meridian of the specified UTM zone. UTM has the positive easting direction going east. This means that points that lie to the west of the central meridian would get negative values. To avoid having to deal with negative numbers all together, 500.000 meters is added to the easting value. This is called a false easting. With the UTM zone number, the northing value and the easting value, a point on earth can be specified unambiguously.

As mentioned, the system presented in this paper uses both geographic coordinates and UTM coordinates. For the conversion between latitude and longitude and UTM coordinates, the C++ library GeographicLib [12] is used. The conversion needs to know the central meridian of the UTM zone the coordinates belong to. Section IV presents results from tests performed in southern Norway. UTM zone 32 is used in this part of Norway. Zone 32 covers the longitudes between 6° and 12° East and the central meridian is 9° East. GeographicLib's projection function takes as input the central meridian in degrees, the northing value and the easting value. Care must be taken to ensure that the easting value provided is in the correct form. The false easting of 500.000 must be subtracted from the easting before the projection. With the northing, easting and zone number, the function will return the coordinates in the form of latitude and longitude.

b) Terrain data convention: The terrain data from The Norwegian Mapping Authority is a grid of elevation data. Each cell in the grid contains the elevation for that point in the terrain. The cells are square and cover an area of 10-by-10 meters. A cell to the right of another cell lies directly to the east of it. A cell above another lies directly north of it. This type of layout is best described in the Cartesian coordinate system. This means that the easting values are increasing with x and that the northing values are increasing with y.

c) Azimuth angle to polar coordinates: When using the azimuth angle with the grid of terrain data, it is convenient to have all the values in the Cartesian coordinate system. However, an angle is best described as polar coordinates. Conversion between polar- and Cartesian coordinates is easy, but the convention for the azimuth angle and polar coordinates are slightly different. Therefore, the azimuth should follow the conventions of polar coordinates before converting it to the Cartesian coordinate system. The convention for the azimuth angle is that it increases clockwise from the north-axis. However, the convention for polar coordinates is that the angle goes counterclockwise from the x-axis (i.e. east). This is illustrated in Figure 6. Here, ω represents the convention used for azimuth angles and ϕ represents the convention used for polar coordinates. To transform the azimuth angle to go counterclockwise from east, the conversion in Equation 1 is performed. This function returns the smallest, positive modulus in the correct reference system. This will yield results in the range of $[0, 360\rangle$.

$$\phi = mod(mod((90 - \omega), 360) + 360, 360) \tag{1}$$

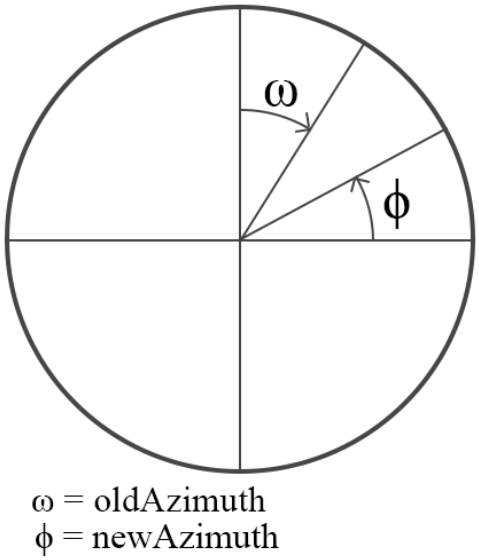


Fig. 6. Result of the conversion of the azimuth angle. ω is the convention used for the azimuth angle. ϕ is the convention for polar coordinates.

where:

ϕ : is the corrected azimuth angle
mod : is the floating point modulus
ω : is the old azimuth angle

d) Polar- to Cartesian coordinates: Now that the azimuth angle follows the conventions for polar coordinates, the conversion to the Cartesian coordinate system can be performed. To represent the azimuth angle in polar coordinates, an angle, ϕ, and a radius, *r* are used. The conversion from polar- to Cartesian coordinates can be seen in Equations 2 and 3. It is preferred to represent the azimuth angle as a unit vector in the Cartesian coordinate system. This unit vector is produced by inputting 1 for the radius, *r*, and the corrected azimuth for ϕ for both the x- and y component. Equation 4 shows this vector, and will later be referred to as the observer-sun 2D vector. It represents a vector going from the origin towards the sun in the horizontal plane (i.e. the two-dimensional plane made up by the east-west and north-south directions).

$$x = r * cos(\phi) \tag{2}$$

$$y = r * sin(\phi) \tag{3}$$

$$\mathbf{v} = [cos(\phi), sin(\phi)] \tag{4}$$

4) Terrain Extraction: To be able to detect if the sun is visible, it is crucial to know what part of the terrain is between the observation point and the sun. There are many ways of doing this sort of calculation. One could use ray tracing techniques, Beziér curve interpolation, line interpolation, line-plane intersection, parallax mapping, etc. most of which are heavily used in computer graphics. The approach taken in this

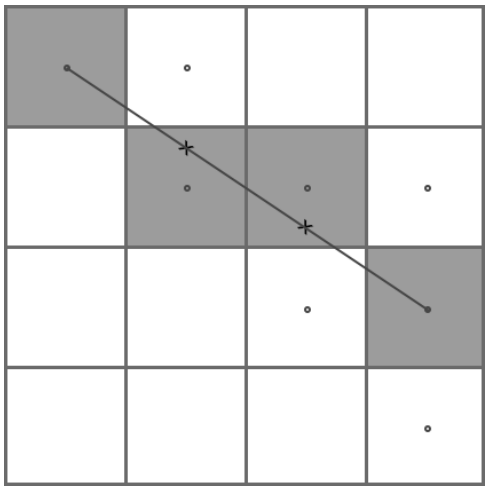


Fig. 7. A visualization of what pixels are drawn with Bresenham's line algorithm.

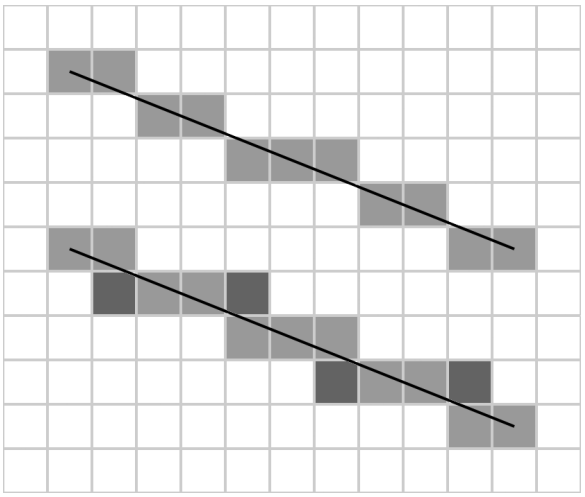


Fig. 8. The different outputs of Bresenham's line algorithm (top) and the Bresenham-based supercover algorithm (bottom).

paper also draws from the domain of computer graphics, by using Bresenham's line algorithm.

Bresenham's line algorithm is a well-known rendering algorithm that approximates a line between two points in a grid of pixels. It is a fast algorithm, since it only employs integer additions, subtractions and bit shifts. It also requires little memory. It was initially developed to draw lines on a screen. Since computer screens have a finite set of pixels, there was a need to find a way to draw pixels in a way that looked like a continuous line. Bresenham's line algorithm does this by keeping track of the error between where a pixel is drawn on the grid, and where the pixel would be if it was on the continuous line. Figure 7 illustrates this concept. Imagine drawing the pixels corresponding to the line, from left to right. The first gray square is drawn as the starting pixel. In the second column, the line goes through both squares marked with a dot. To decide on what square to color, the algorithm will choose the one that has its center closest to the point on the line marked with a cross. By continuing to draw the squares that are closest to the line, Bresenham's line algorithm yields a good approximation to a continuous line.

As mentioned, the terrain data comes in a grid with cells of elevation points. By combining this grid with the observer-sun 2D vector, one can calculate what cells of the grid that lie between the observer and the sun. If the top-left cell of Figure 7 is the observer's position and the line represents the observer-sun vector in the horizontal plane, it is clear that Bresenham's line algorithm returns only a subset of the cells that lie between the observer and the sun. Two of the white cells marked with a dot is also on the path of the line. By using Eugen Dedu's Bresenham-based supercover line algorithm [7], it is possible to find all the squares the line passes through. Figure 8 illustrates the different output of the two algorithms, where four more squares are marked with the latter algorithm.

The Bresenham-based supercover algorithm requires a starting point and an end point to do its calculations. The starting point is the cell that the observer is in. The end point is somewhere along the vector towards the sun. To avoid having to deal with the curvature of the earth, 20 km is chosen as the length of the vector. Even though the curvature of the earth has some impact on the results at 20 km, it is disregarded. In Section III-B3d the observer-sun 2D unit vector, **v**, was obtained. By multiplying **v** with the length of the search (i.e. 20 km), the end point is obtained. These two points are then sent as arguments to the Bresenham-based supercover algorithm. The returned result from this algorithm is a list of all the cells the vector passes through.

By using the Bresenham supercover algorithm, all the cells the line passes through are obtained. However, this is under the assumption that the observer is in the center of the 10-by-10 meter cell. In Figure 8, if the starting point was in the lower, left corner, the line would pass through different cells. The cell below the starting point would be marked, and not the one to the right. Even though the starting point is in the same cell, the placement within the cell matters. This could result in wrong sun glare predictions. This has a greater effect the closer the cell is to the observer due to the parallax effect between the two viewpoints. Since the displacement within a cell is so small compared to the distance of the vector (i.e. up to four orders of magnitude smaller), it is assumed that the displacement within a cell will not yield very different results.

5) Occlusion test: When all the terrain points between the observer and the sun is acquired, it is time to see if the sun is visible to the observer. In short terms, this is done by drawing an imaginary line from the observer towards the sun, and then check if the terrain is higher than this line somewhere along it. The steps for doing this check for one elevation data point is described below.

Step 1: The first step is to find the linear distance in the horizontal plane from the observation point to the point in the terrain. The line marked L in Figure 9 (a) is this distance. It is obtained by solving the Pythagorean equation for the hypotenuse as seen in Equation 5, where Δx and Δy are the distances between the two points in meters. They are obtained by simply subtracting the easting and northing of the two points.

$$L = \sqrt{\Delta x^2 + \Delta y^2} \tag{5}$$

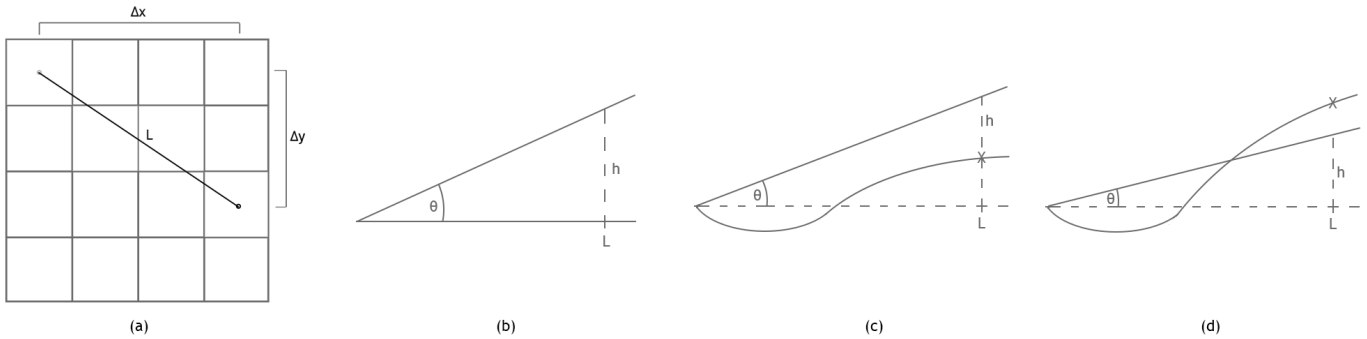


Fig. 9. Shows the principles behind the occlusion test. (a) shows the distance, L, between the observation point and the terrain point, (b) shows the height, h, to be calculated for a given terrain point, using the elevation angle, θ, and the distance, L, (c) shows a case where the sun is visible at point L and (d) shows a case where the terrain occludes the sun at point L.

Step 2: When the distance, L, is known, the next step is to calculate the height of the observer-sun vector, L meters from the observation point. A visualization of this is found in Figure 9 (b), where the sun's elevation angle is denoted as θ, and h is the height of the vector. The height is found by using Equation 6. This calculation returns the height of the vector, relative to the observer. This means that if the observer is not at sea level, its elevation must be accounted for. The true, real world, height of the vector, L meters from the observer, is the sum of the result of Equation 6 and the observer's own elevation. The observer's elevation is the height retrieved from the road data for the position the driver is at plus the height from the road to the driver's eyes. For the calculations done in this paper, it is assumed that the driver's eyes are 1.5 meters above the road when driving a passenger car. Modifications of this value can be done in the configuration file if the system is to be used with other types of vehicles.

$$h = L * tan(\theta) \tag{6}$$

Step 3: When the height of the vector is known, it is trivial to check if the sun is occluded by the terrain. Figure 9 (c) shows a case where the sun is not occluded by the sun at point L. The curvy line represents the terrain between the observer and the sun, as seen from the side. The spot on the curve marked with an "X" is the point from which the terrain's height is retrieved. Because the observer's elevation plus the height of the vector is larger than the terrain at point L, the terrain does not occlude the sun. Figure 9 (d), however, shows a case where the sun is occluded by the terrain. Because the observer's height plus the vector's height is smaller than the terrain height, the sun is not visible to the observer.

To find out if the sun is visible to the observer, this test is done for all the terrain points returned by the Bresenham supercover algorithm. If the test shows that the true height of the vector is larger than the terrain points for all points, the sun is visible. However, if there is one case where the terrain is higher, the sun is occluded and thus not visible.

6) Road direction check: It is, however, not enough that the sun is visible to a driver, for there to be sun glare. If, for instance, the sun is directly above the driver, it is not a nuisance. The angles of which the sun is bothersome has been addressed in several papers, [6], [10] and [11]. As stated in [11], a maximum angle of 25° from the driver's line of sight is suggested for a 60 year old driver. An aperture of 50° is therefore chosen as the glare limit. Figure 10 shows this aperture. Figure 10 (a) shows a vehicle as seen from above, with the driving direction going straight to the right from the front of the car. The angle α is the horizontal sun glare angular limit. In Figure 10 (b), the same vehicle is seen from the side, where β is the vertical sun glare angular limit. In this case, α and β are both chosen to be 50°, but could also be different values.

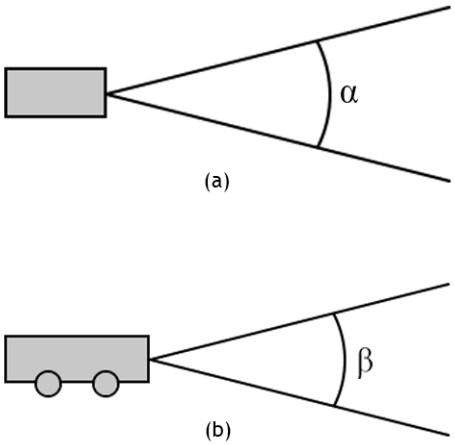


Fig. 10. Illustrates the aperture of the glare angular limit. (a) is a top-down view of a car on the road and (b) is the same vehicle as seen from the side.

As previously mentioned, the road data includes a list of points along a road segment. These points are used to find the driving direction for a given point. Figure 11 shows three of these points along a road, named p1, p2 and p3. The vectors $\boldsymbol{v_{1,2}}$, $\boldsymbol{v_{2,3}}$ and $\boldsymbol{v_{1,3}}$ are the normalized three dimensional vectors between the points. To find the driving direction at point p2, the vector $\boldsymbol{v_{1,3}}$ is used, which is the average of $\boldsymbol{v_{1,2}}$ and $\boldsymbol{v_{2,3}}$.

For sun glare to be present, the sun needs to be within both the α and the β angles shown in Figure 10. The α angle is in the horizontal plane. To see if the sun is within this limit, the difference between the azimuth angle and driving direction is obtained. In Section III-B3b, the azimuth angle was converted

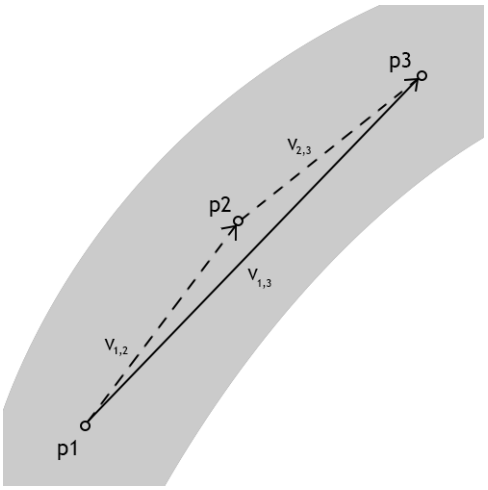


Fig. 11. Three points along the road segment marked p1-p3, with the vectors between the points, as seen from above.

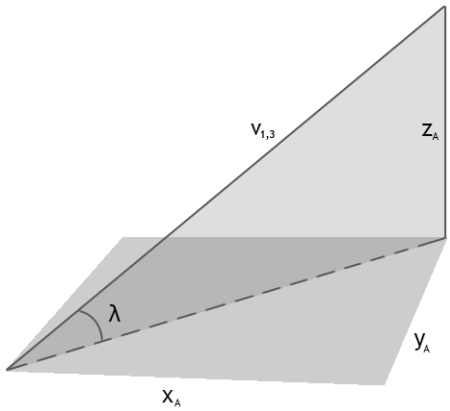


Fig. 12. The elevation profile of $\boldsymbol{v_{1,3}}$ shown in three dimensions. λ is the elevation angle and x_A, y_A and z_A are the individual components of the vector.

to the correct reference system, so the driving direction angle must be calculated in the same reference system. This is done by finding the angular distance between the average vector, $\boldsymbol{v_{1,3}}$, and the x-axis. To do this, the arctangent function with two arguments is used. It can be seen in Formula 7, where y_A and x_A are the y- and x components of vector $\boldsymbol{v_{1,3}}$, respectively. $\phi_{drivingDirection}$ is the angular difference between the x-axis and $\boldsymbol{v_{1,3}}$. It is worth noting that the arctangent function returns degree values in the range of $[-180, 180]$, so a value of 360 is added to $\phi_{drivingDirection}$ if the returned angle is negative. When $\phi_{drivingDirection}$ is obtained, it can be used to check for sun glare in the horizontal plane. If the absolute value of the difference between $\phi_{drivingDirection}$ and the corrected azimuth angle ϕ is less than 25°, the sun is within the limits of horizontal glare.

$$\phi_{drivingDirection} = atan2(y_A, x_A) \tag{7}$$

After finding the position of the sun relative to the vehicle in the horizontal plane, the relative position is calculated in the vertical plane. Figure 12 shows the elevation profile of the average vector, $\boldsymbol{v_{1,3}}$. λ is the road's elevation angle to be obtained. x_A, y_A and z_A are the values of the vector, $\boldsymbol{v_{1,3}}$, as shown in three dimensions. By using Equation 8, the slope of the road relative to the horizon, λ, is calculated.

$$\lambda = tan^{-1}\left(\frac{z_A}{\sqrt{x_A^2 + y_A^2}}\right) \tag{8}$$

The absolute value of the difference of λ and the sun's elevation, θ, is then calculated. As in the horizontal plane, if this value is smaller than 25, then the sun is within the vertical sun glare limit. However, if either of these differences are above 25°, there is no sun glare.

7) Optimizations: The default values for the rush hour intervals are set to 7-9 a.m. and 3-5 p.m. With a resolution of 5 minutes, the Sun Glare Detection System does the previously mentioned calculations for each point of each road 18250 times a year. When there are several points per road segment and thousands of roads, the amount of computation grows. This section will look at some of the optimizations done for the SGDS.

A naive approach of checking every point for glare would be to start at January 1st at 7:00 a.m. and check every point in the SOSI-file, before moving on to 7:05 a.m., checking every point, and so on. Since the SOSI-file is organized by segments, one would then need to read all points in all segments for every time interval. To minimize the number of reads to the SOSI-file, each segment is fully processed before reading the next one. A segment is first read to memory. Then, each point in the segment gets checked for glare for the entire year. Doing the calculations in this order makes it a lot easier to generate a shapefile with the layout that will be described in Section III-B8. Also, by finishing the calculations for a particular point for the whole year at once, all data from that point can be written to the shapefile.

Finishing one point at a time allows for another optimization. Figure 13 shows a vehicle as seen from above. The cone marked with α shows the aperture of the glare limit of the vehicle. The other cone represents where there is sun between 7:00 a.m. and 9:00 a.m. The left leg of the light gray cone is the azimuth angle at 7:00 a.m. and the right leg is at 9:00 a.m. Before checking a point for glare every five minutes between 7:00 and 9:00, a test is performed. If there is no overlap between the two cones, there can not be sun glare between those two times. An example of no overlap can be seen in Figure 13 (a). If, however, the two cones are overlapping, as seen in Figure 13 (b), further computation needs to be performed. If there is no overlap, the system will not check for glare for that point between 7:00 and 9:00. The same principle is used between 3:00 p.m. and 5:00 p.m. If there is no overlap for either the morning or evening rush hours, the point is certain to not have glare that day, and the system moves on to the next day.

8) Shapefile Generation: As previously mentioned, the results are sent back to the QGIS plug-in as a shapefile. The shapefile consists of several features with several attributes. Appendix D shows six features and all their attributes. For

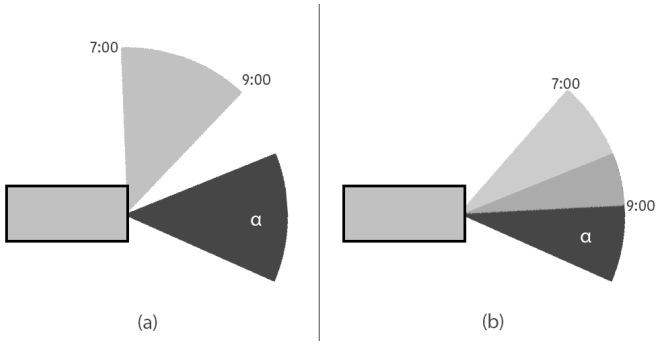


Fig. 13. Illustrates the main idea behind the overlap check. (a) shows a case where there is no overlap between the vehicle's glare aperture (α) and the sun's position between 7:00 a.m and 9:00 a.m. (b) shows a case where there is overlap.

every point on the road that has glare during the year, a new feature is created. Each feature has 67 attributes. There is one attribute for each week of the year (53), one for each month (12), one for the whole year and one label that says if the point had glare in the morning- or evening time interval. Each of the week-, month- and year attributes act as counters. The attribute for week 1, for instance, starts of as zero and is increased by one each time the system reports glare in the first week of the year. This means that if a point on the road only has glare at 7:00 a.m. and 7:05 a.m. the first of January and no glare the rest of the week, the attribute for week 1 will have a value of 2. The month- and year attributes also aggregate the data in the same way. By doing it this way, the count for each attribute acts as $time\_resolution * count$ minutes of glare over the specified attribute time period. So, for the previously mentioned example, the value 2 roughly represents $5min * 2 = 10$ minutes of glare in the first week of the year.

C. Plug-in

To visualize the results from the Sun Glare Detection System, a QGIS plug-in is developed. It contains three widgets that simplifies the user's interaction with the glare data, as well as ways of generating and loading the data. Screenshots of the plug-in and the different widgets can be seen in Appendix C. The plug-in is written in Python and communicates with the SGDS using the Python library $ctypes$. This section will describe the plug-in. First, a brief introduction to the plug-in's functionality will be presented. Secondly, the rendering techniques is described. Finally, the information extraction tools are presented.

1) Plug-in Introduction: When installed from the QGIS plug-in repository, the plug-in is located under the "Plugins"-tab. To use its functionality, one must first generate some sun glare data. Generating data can be done in two ways. The first is an action in the plug-in's menu called "Generate sun glare data". This will prompt the user with a window instructing the user to provide a SOSI-file for the roads, a Digital Elevation Model-file for the terrain, a configuration file, an output path and a file name. When clicking OK, the user must then wait until the Sun Glare Detection System is done processing all the roads. This also causes QGIS to freeze during computation. The second way is to run a python script that is located in the plug-in's main folder. It will require the same arguments as with the first option. However, this will not freeze QGIS, and one can run multiple instances of the script simultaneously, to lower the total time needed for generating the data.

When the user has some data ready, the next step is to load it into QGIS. There is an action in the plug-in menu called "Open Existing Shapefile" that will automatically load Google Maps and render the data in the shapefile on top of the map. A third-party plug-in, OpenLayers Plugin [18], is used to load Google Maps into QGIS. OpenLayers Plugin has a variety of different maps to use. For this plugin, the choice fell on using Google Streets, since it shows street names, which is helpful when finding high-glare roads. Other than that, it has a fairly neat and tidy layout, with few distractions. With the data loaded, the plug-in's widgets are ready to be used.

2) Rendering: Loading the data into QGIS causes the contents of the shapefile to be displayed on top of a map. A vector layer is created and populated with the features in the shapefile. Each point in the vector layer is rendered on the map as a small circle. This causes the map to be filled with small circles at every point that had glare at least once in an entire year. To be able to extract some meaningful information from the view, a Graduated Symbol Renderer is added to the layer. A Graduated Symbol Renderer in QGIS has the ability to render the contents of the vector layer differently. Size, color and symbol (i.e. circle, square, etc.) are some of the parameters that can be changed programmatically. First, the renderer is told which attribute(s) it should focus on. If, for instance, users are interested in yearly glare, the $year$ attribute is set as the target attribute. Then, the renderer can be configured to render points differently based on the value of the $year$ attribute. It is common to use a range from zero to the maximum value of the target attribute is used. This range is divided into several buckets of equal intervals. For each of these buckets, a different size, color and symbol can be set. By setting a darker color and larger size for the higher values, points with a lot of glare will appear as bigger and darker circles than points with little glare.

Instead of rendering based on one attribute, an expression can be created. This expression can contain several attributes and can be a mix of additions, multiplications, subtractions, etc, of these attributes. To render based on the total glare from January to March, the expression would be $"m1" + "m2" + "m3"$. This will cause the renderer to add the values of the three attributes together and use this value to categorize each point in one of the buckets. It is worth noting that when using expressions, the range for the renderer should be changed to reflect the new maximum value.

Filtering is another technique in QGIS. Queries can be written to exclude some of the data from rendering. For instance, the filter expression $"value" > 0$ would remove all features that have the attribute "value" set to 0 or a negative number. For string-attributes, it is possible to filter on equality and inequality. *"answer" ILIKE '%yes%'* will exclude all features that have the attribute called "answer" set to anything other than "yes".

When looking at the data at a large scale, the points from all the roads would just be merged into each other. Therefore, an aggregated layer is generated. An example of this can be seen in figure 14. Instead of rendering all the points at large scales,

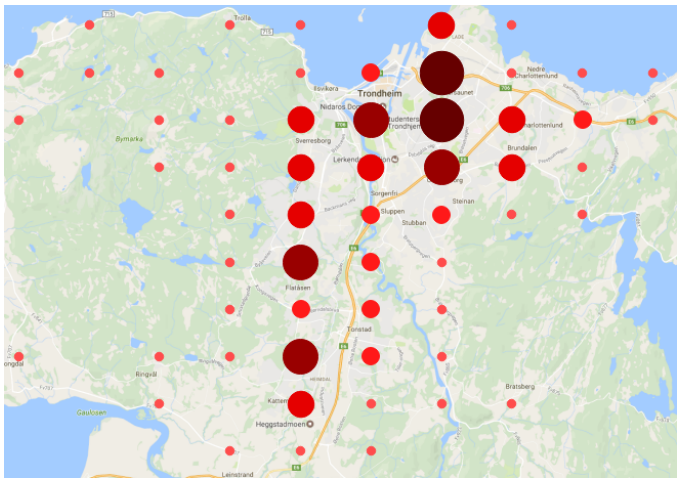


Fig. 14. An image of the aggregated layer, where bigger, darker circles represent areas with more glare.

it merges the values of all points within a geographic area, and creates a circle representing the total glare for that area. This layer is created by dividing the contents of the shapefile into a grid. Each cell in the grid represents the amount of glare of all the roads within the cell's boundaries. By retrieving all the points that are within one cell, one can find the total amount of glare for that geographical area. The yearly value for glare for each point is taken to the power of four. Each point in the aggregated layer gets its value using Equation 9.

$$\sum_{i=1}^{n} year_i^4 \tag{9}$$

This equation is used to favour roads with more glare, but still include the roads with some glare. If one were to take the average of all points in a cell, areas with a few roads with a lot of glare would be favoured over areas with the same amount of high-glare roads, but with many low- and medium-glare roads as well. Simply taking the sum of all glare would greatly favor areas with many roads. Therefore, taking the yearly value to the power of four will favor the roads with lots of glare, but still include the roads with some glare as well. This layer then gets its own Graduated Symbol Renderer, and will display larger, darker circles for the areas with the most exposed roads. This lets the user get a quick overview of what areas are the most interesting. By using a QGIS feature called "Scale Based Visibility", on can toggle a layer's visibility when zooming in and out. Both the aggregated layer and the glare data layer have their visibility scale set. The aggregated layer is only visible at scales larger than 1:200 000 and the glare data layer is visible at scales smaller than 1:200 000.

3) Information Extraction Tools: Once the data is loaded into QGIS, there are three menu actions (widgets) that are developed for information extraction and information understanding. Screenshots of the widgets' UI can be seen in Appendix C. This section will explain their usage.

The first widget is designed to study the data on a temporal basis. Here, the user can choose to filter the data on a weekly or monthly basis. The user can also choose to show the data from either the morning- or evening intervals, or from both. For instance, if the user chooses a monthly basis, selects the months 1-6, and checks the "Morning" radio button, the plug-in will only render cases of morning glare in the months January to June.

This is achieved by using both the attribute based rendering and filtering of the Graduated Symbol Renderer. If the user chooses to view the data on a weekly basis, the plug-in will use 'w' as a prefix. If the user chooses a monthly basis, 'm' is used as a prefix. Then, the plug-in builds an expression for each of the numbers in the range. If the user chooses weekly basis from week 45 to 46, the expression would be "$w45$"+"$w46$". If the user checks the "Morning" radio button, the filtering expression, *"rushType" ILIKE '%morning%'*, is executed. This will remove all features that have their rush type set to "evening". If the user wants to see both morning- and evening glare, there is simply no filtering.

The second widget lets the user select a range of glare values to be displayed for a specified attribute. First, the user selects the desired attribute. Then, the user can use two sliders to specify the minimum and maximum value for the selected attribute. By using this widget, the user can easily exclude points based on their value. Also, since the values represent total time of glare, this widget is a good tool to find roads that have more glare than a certain threshold. For instance, when using a resolution of 5 minutes, setting the attribute to "year" and the minimum slider to 288 will exclude all points that had less than 24 hours of glare that year. This is all achieved by filtering. An expression on the form "$attribute$" $>$ "$minValue$"AND"$attribute$" $<$ "$maxValue$" is created and set in the renderer.

The last widget is used to display the points with the most glare. First, an attribute is selected. The user can then choose to either get the "X" points with the highest values for the attribute or get the points in the top "X"-th percentile. Choosing 50 as a value for "X" would then either render the 50 points with the highest value or the top half of the points. This is done using filtering. If the user wants to see the "X" points with the highest value, the plug-in finds the glare value of the "X"-th highest feature. If the user wants to see the "X"-th percentile, the plug-in finds the lowest value in the top "X"-th percentile, and uses this as the threshold value. Then, a simple expression on the form, "$attribute$" $>$ "$thresholdValue$", is created.

IV. Evaluation and Results

For a system like the one presented in Section III, there are several different data sources that are mapped into a common reference system. Data can be flawed and the conversions can be imprecise. To check the validity of such a system, thorough testing and validation must be performed. This section will explain the approach taken towards testing the validity of the system and provide the results of the evaluation. Tests have been conducted as both comparisons between algorithms, field tests in the real world and validation of the rendering in QGIS. The system has been tested in both Asker, Norway and in Trondheim, Norway, with good results both places.

TABLE I. THE DIFFERENCES IN AZIMUTH- AND ELEVATION ANGLE FOR THE IMPLEMENTED ALGORITHM AND THE ONLINE CALCULATOR. SHOWS RESULTS FOR JUNE 21ST 2016 AND DECEMBER 21ST 2016.

June 21	ΔAzimuth	ΔElevation	Dec 21	ΔAzimuth	ΔElevation
08:00	0.00	0.03	12:00	0.01	0.23
12:00	0.00	0.02	13:00	0.01	0.24
16:00	0.01	0.01	14:00	0.00	0.39
20:00	0.01	0.06	15:00	0.01	0.13

A. Sun Position Algorithm

The glare detection system presented in this paper relies heavily on the calculations of the sun's position. Small, accumulating errors or large deviations from the true position can not be tolerated. Testing of the implementation of the algorithm found in [4] needs to be performed.

After implementation of the algorithm, a compass and a mobile app for angular inclination was used to check the validity of the acquired results. To further validate the results, an online calculator provided by the National Oceanic and Atmospheric Administration [8] was used. Just as with the algorithm implemented in this paper, it requires a date, time and coordinates for the test. It also needs to know about time zone and daylight savings time. Calculations for both the online calculator and the implemented algorithm were done for two dates, June 21st 2016 and December 21st 2016. These are around the days of the solstice. At these days, the elevation angle is at its most extreme. For June, calculations were done for four times of the day, namely 08:00, 12:00, 16:00 and 20:00. For December, the times were 12:00, 13:00, 14:00, 15:00. The results are displayed in Table I. The differences between the angles calculated by the implemented algorithm and the online calculator are shown for June and December. The maximum difference is 0.39°. Similar experiments have been conducted for other years, showing similar results.

B. Geographical Occlusion

After establishing the correctness of the calculation of the sun's position, the occlusion test was evaluated. At sunrise and sunset, the sun alternates between being occluded by the terrain and being visible. Validation of the occlusion test is therefore done at these times. Results of one test at sunrise and one test at sunset will now be presented.

The occlusion module described in Section III-B5 takes as input a position, a date and a time. It uses the sun position algorithm to find the azimuth- and elevation angles. The module returns whether or not the sun is visible to the observer at the given time. To predict when sunrise will occur, the only information known is the place the observer is at. To find the exact time when the sun will rise at the given point, an iterative approach is taken. First, a time when it is dark is provided to the module, and thus the module returns that the sun is not visible. Then, the time is incremented by one minute until the module returns that the sun is visible. The last provided time is the time of the sunrise. This was done for November 8, 2016 in Trondheim. The module returned that the first time the sun was visible at that given point this day was at 09:13 a.m. Figure 15 shows an image of the sun at 09:14 a.m. this day. It shows the sun just above the distant mountains. Validation that the system reports occlusion at the correct place is then

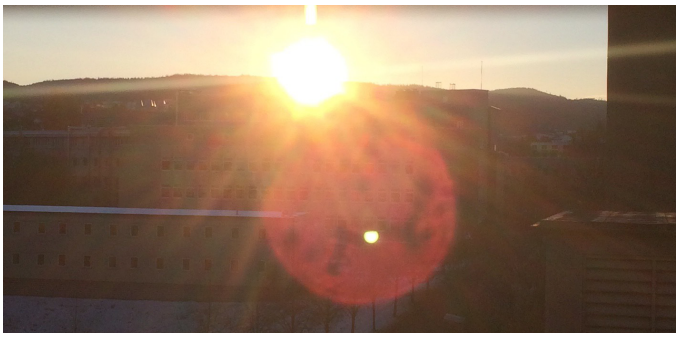


Fig. 15. Image of the sunrise taken in Trondheim November 8, 2016 at 09:14 a.m.

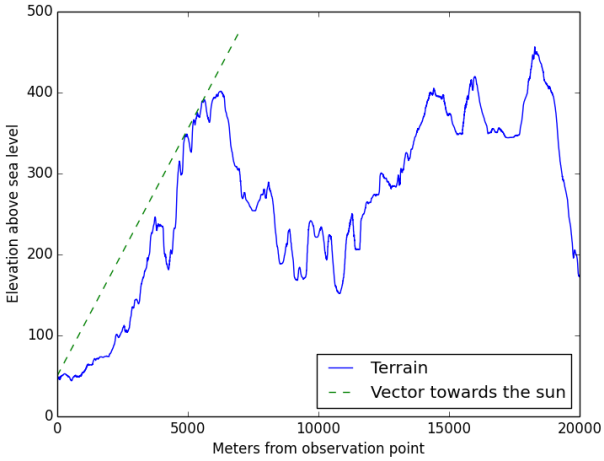


Fig. 16. A plot of the terrain between the camera and the sun. The straight line is the vector towards the sun. The x-axis is meters from the observer and the y-axis is the elevation in meters. The line intersects the terrain at 5536 m.

conducted. Investigation of the data confirms that the place occlusion is reported at is indeed at the far mountain. The graph in Figure 16 shows the terrain data and observer-sun vector. Distance from the observer is shown on the x-axis and elevation above sea level is shown on the y-axis. The straight line is the observer-sun vector. As seen, the vector intersects the elevation data 5536 meters from the observer. By using a distance tool in Google Maps, it is confirmed that 5536 meters from the observer is indeed at the top of the mountain seen in Figure 15.

Another test was conducted during sunset on March 28th, 2017 in Trondheim, Norway. This test shows positive results for the occlusion module's ability to detect the sun's visibility over a larger geographical area. The time of sunset was found using the same iterative approach as for the sunrise. This time, however, the system was evaluated for several road segments. A subset of these segments can be seen in Figure 17 (b). In this figure, the red lines represent roads where the sun was visible at 7 p.m. The green lines are places where the sun was visible at 6:55 p.m., but not at 7 p.m. The points A and B therefore represent points on the road where the sun was and was not visible at 7 p.m., respectively. Figure 17 (b) shows a top-down view of the city, with north going upwards. Figure 17 (a) shows a side view of the mountain that occluded the green

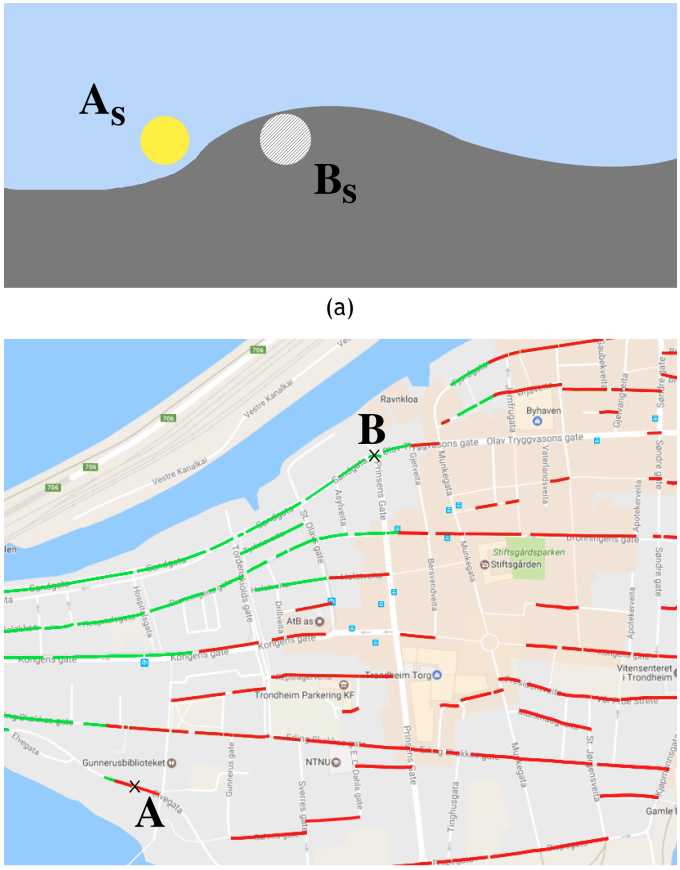


Fig. 17. Representation of a test of the sunset in Trondheim, Norway on March 28th, 2017. Top: An image of the sun as seen from point A and point B at 7 p.m. Bottom: Result from the glare detector, where green lines represent shadow at 7 p.m. and red lines represent sun visibility at 7 p.m. The sun is visible at point A and occluded at point B.

roads. In this figure, north is going to the right, so the image is taken towards west. This means that the mountain ridge is going in the north-south direction and is located directly west of point A and B. From both point A and B, the sun went down behind the left slope of the hump in the mountain, and the apparent position of the sun at 7 p.m. is shown as A_s and B_s. Since point A is located further south than point B, the sun is visible longer before going down behind the hump. Point A is also closer to the mountain, yet the sun is still visible. When going further north, one would also need to go further east for the sun to not be occluded by the hump, as the figure illustrates. These results further support the validity of the terrain occlusion module. They also show that the module can be used to locate shade in larger geographical areas.

C. Road direction

It is now confirmed that the system can both locate the sun and determine if it is visible. A final evaluation of the SGDS is seeing whether or not the system can correctly determine sun glare on the road. Figure 18 shows a map of the place where the system was evaluated. The figure shows the azimuth angle, ω, of 211.23°. The time of the experiment is 14:26 on December 6. *Segment 1* and *Segment 2* are the two road segments that are evaluated. Vertical differences between the

Fig. 18. Map of the place where the system was evaluated. Shows the azimuth angle, ω, of 211.23° and the two segments the system was tested with.

road segments' elevation angles and the elevation of the sun are small in both cases. The angular distance between Segment 1 and the azimuth angle is very small, so it is expected that glare should occur. On the other hand, the difference between Segment 2 and the azimuth angle is approximately 50°, so there should be no glare. By inputting Segment 1 to the system, it returns that there is glare. By inputting Segment 2, it returns that there is no glare. This is in line with what is both expected and observed on site.

Testing of the system in the vertical plane is also conducted, since the vertical difference in the validation of Segment 1 is negligible. To test the vertical aspect, the parameters from the test of Segment 1 are used. Since the test above has already proven to report sun glare, the inclination of the road and the elevation of the sun can be altered to see if the system still reports glare. First, the sun's elevation is increased, so that the difference between the road's inclination and the sun's elevation is larger than 25°. Then, the sun's original elevation is used, but with a large inclination in the road, such that the difference is larger than 25°. Both these cases reports no glare. A similar test is conducted with a large declination, and no glare is reported. Other variations of the sun's elevation and the road's inclination, where the differences are below 25°, are also tested. All the results are in line with what is expected, and the testing of the vertical aspect is redeemed a success.

D. Visualization in QGIS

When the data is generated, it can be loaded into the QGIS plug-in. This section will look at how the data is represented visually and show the validity of the plug-in's rendering methods.

1) Filtering: As described in Section III-C2, the user can filter the data on different attributes. The plug-in described in

this paper lets the user filter on morning and evening rush hour. Figure 19 shows the results from filtering on the rush hour types. Figure 19 (a) displays the standard rendering, where both morning and evening rush hour glare is shown. A rendering of only morning glare can be seen in Figure 19 (b), and evening glare is presented in Figure 19 (c).

It is interesting to see that when morning glare is selected, only roads segments that go towards east are shown. The same holds for evening glare, where only segments going westward are shown. It is worth noting that the rendered points do not have a directional dimension to them. Therefore, the user must be aware of the fact that the sun rises in the east and sets in the west. This means that the direction of the road segment shown in Figure 19 (c) has to be towards the south-west, since the sun is never visible towards north-east in the evening. Road segments that go east-west might have glare going both ways throughout the year. In the upper left corner of Figure 19 (a), such a segment is shown. There are two circles per point, since it has registered glare both for the morning and evening rush hour.

2) CRS Conversions: The points that are written to the shapefiles use the standard geographic coordinates, which is the standard latitude, longitude format. This coordinate reference system is called WGS84, with ID EPSG:4326. The OpenLayers plugin, that provide the rendering of the maps, uses a different CRS, namely WGS84 / Pseudo Mercator, with ID EPSG:3857. When importing the data, the CRS of the shapefile is set to EPSG:4326, and when rendering the points, QGIS does the conversions automatically. A close up of some points can be seen in Figure 20. Here, we can see that even after being converted from Universal Transverse Mercator to WSG84 to WGS84 / Pseudo Mercator, the points are still rendered in the middle of the road.

E. Complexity

Calculating glare for large sets of roads is time consuming. This section will look at the data generation times and the benefits of the optimizations described in Section III-B7. All tests were run on a MacBook Pro (late 2013), with 16GB RAM and a 2,3 GHz Intel Core i7-4850HQ processor.

1) Generation Time: Checking for glare every five minutes in both the morning- and evening rush hour results in 50 calls to the glare detector for each point on the road for one date. This means that the system will check for glare $50 * 365 = 18250$ times for each point. When generating data, the run times were measured. As mentioned, the data is divided into road segments, and each segment contains, on average, 12 points. Data has been generated for 10, 100 and 1000 segments, and results show that the system scales linearly (i.e. the average time used per point remains the same). On average, the system uses 1.39 seconds to generate a feature for the shapefile. This includes both reading the coordinates of the point, checking for glare for the entire year and writing the result to the shapefile.

The system has also been tested to run in parallel. As mentioned, the plug-in comes with a Python script that can be used to generate the data. The MacBook used for testing has four CPU cores. By running four instances of the Sun Glare Detection System, the system's parallelism can be tested. By measuring the run times of all instances, the system's ability for parallel computing was tested. Results show that there is no significant increase in time used per feature for generating data on several cores simultaneously.

2) Azimuth and Aperture overlap: Experiments have also been conducted to check the potential speed up from the overlap test described in Section III-B7. First, the SGDS was modified so that it did not include the overlap test. Then, the system was set to generate glare data for 1000 road segments and the run time was measured. The resulting shapefile was stored. Then, the SGDS was run again, but this time including the overlap test. The same 1000 road segments were used for this run. The run time was again measured and the shapefile stored. The generated shapefiles were inspected. Both files were identical, containing exactly the same features with the same values. It took the system 5.95 hours to generate the shapefile without the overlap test. When checking for overlap, it only took 4.61 hours. This corresponds to a significant speed up of 22.5%.

V. Future Work

The system presented in this paper takes the sun's position and the surrounding topology of the terrain into account to see if there is sun glare at a given road segment. However, there are more than just the surrounding terrain that can obstruct the view of the sun. Buildings often block the view of the sun, especially in cities. Forests and other vegetation can also obstruct the view. Extending the model to include building- and vegetation data could yield more accurate results. This could be implemented in various ways. One way would be to merge the terrain data with building data. If there is a building at one of the cells in the digital elevation model-file, then the value of that cell would be incremented with the height of the building. The terrain data used for this plug-in has a resolution of 10 meters. When merging with building data, the resolution should be more fine-grained, and 1 meter might be more reasonable. This will increase the accuracy of the results, at the expense of longer run times. Parts of the system, for instance terrain extraction, could also be implemented on the GPU. Precomputed mipmaps of the terrain- and building data might lower the run times. However, this would be at the expense of precision. Another possibility would be to create a box terrain. Each 10-by-10 meter cell is one box, having the height of the terrain at that cell. Buildings would then be added on top of the terrain boxes as their own boxes. Ray-box intersection could then be implemented for the occlusion test. This approach would keep the resolution of the terrain data at 10 meters, while the buildings are not constrained by the 10-by-10 meter resolution and could be more fine-grained.

With the rapid development of cameras and computer vision, it is possible to gather the vegetation data using computer vision techniques or laser scanning. This could in turn be used to improve the precision of the sun glare predictions.

As mentioned, the system is designed to be able to run in parallel. Data generation times could be drastically lowered by running the SGDS in parallel or in batches. Using a supercomputer or by utilizing the vast parallelism of cloud based services, one could generate glare data for large sets of roads in short amounts of time.

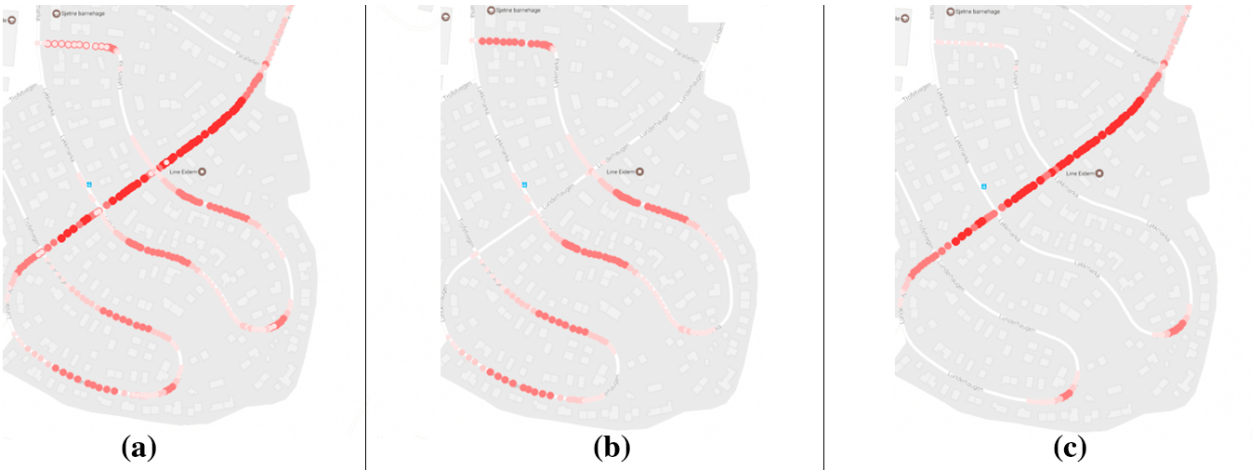


Fig. 19. Shows the results of applying different filtering to the data. (a) shows glare for both morning and evening rush hour. (b) shows only data for morning glare, whereas (c) shows evening glare.

Fig. 20. Shows several glare points and the accuracy of their CRS conversions.

Interpretation of the road data is also an area with potential improvement. The system could provide different models for straight roads, intersections and roundabouts. In addition, the system also only predicts glare at the centerline of the road. The Norwegian Road Data Base [1] contains much more information about the roads, as the number of lanes. This data could be helpful to calculate the glare at the points where the cars actually are, rather than at the centerline of the road.

The results from the system could also be added to the Norwegian Road Data Base as parameters. These parameters could then be retrieved through a REST API. Enriching the data in the data base, could provide additional information to other projects at the Norwegian Public Roads Administration. Adding data to the data base might give new insights, since the road data has increased in dimensionality.

VI. Conclusion

This paper has presented a method for detecting sun glare on the roads, as well as a plug-in for QGIS that has methods for visualizing the sun glare data. The plug-in provides tools for examining the data to ease the information extraction process. Parameters for the glare detection can be altered by the user. By calculating the sun's position and taking the surrounding terrain into account, the results show that the system can accurately mark a point on the road as having sun glare or not at any given time. The results also show the accuracy of the system's ability to calculate terrain occlusion, including prediction of sunrise- and sunset times. Results also show the validity of the visualization. The conducted run time tests show that the system scales linearly, both sequentially and in parallel. This plug-in and its models can be used to do glare evaluations for existing or planned roads. The results might show that there is a need for safety measures to be taken. Future work on the system include a more extensive data set of what could occlude the sun, which could lead to more accurate results.

References

[1] Norwegian Public Roads Administration. The norwegian road database. http://www.vegvesen.no/fag/Teknologi/Nasjonal+vegdatabank. Accessed: 2016-12-08.

[2] The Norwegian Mapping Authority. Digital elevation model files. http://data.kartverket.no/download/content/digital-terrengmodell-10-m-utm-32. Accessed: 2016-09-22.

[3] The Norwegian Mapping Authority. Norwegian road network. http://data.kartverket.no/download/content/vbase-utm-32. Accessed: 2016-09-22.

[4] Manuel Blanco-Muriel, Diego C. Alarcn-Padilla, Teodoro Lpez-Moratalla, and Martn Lara-Coira. Computing the solar vector. *Solar Energy*, 70(5):431 – 441, 2001.

[5] Eun-Ha Choi and Santokh Singh. Statistical assessment of the glare issue-human and natural elements. In *Proc. of the Federal Committee on Statistical Methodology Workshop*, 2005.

[6] Andrew M. Churchill, Yorghos Tripodis, and David J. Lovell. Sun glare impacts on freeway congestion: Geometric model and empirical analysis. *Journal of Transportation Engineering*, 138(10):1196–1204, 2012.

[7] Eugen Dedu. Bresenham-based supercover line algorithm. http://lifc.univ-fcomte.fr/home/~ededu/projects/bresenham/. Accessed: 2016-10-15.

[8] National Oceanic Earth System Research Laboratory and Atmospheric Administration. Noaa solar calculator. http://www.esrl.noaa.gov/gmd/grad/solcalc/. Accessed: 2016-09-22.

[9] Esri. *ESRI Shapefile Technical Description*. Environmental Systems Research Institute, Inc., July 1998.

[10] Kenji Hagita and Kenji Mori. Analysis of the influence of sun glare on bicycle or pedestrian accidents with turning vehicle. *World Conference on Transport Research Society*, 2013.

[11] Rafael Jurado-Piña and Jos Pardillo Mayora. Methodology to predict driver vision impairment situations caused by sun glare. *Transportation Research Record: Journal of the Transportation Research Board*, 2120:12–17, 2009.

[12] Charles Karney. Geographiclib. http://geographiclib.sourceforge.net/. Accessed: 2016-09-26.

[13] International Commision on Illumination. Cie 146:2002. cie equations for disability glare. *CIE Collection on Glare 2000*, 2000.

[14] Paul A Pisano, Lynette C Goodwin, and Michael A Rossetti. Us highway crashes in adverse road weather conditions. In *24th Conference on International Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology, New Orleans, LA*, 2008.

[15] QGIS. Qgis. http://www.qgis.org/en/site/about/index.html. Accessed: 2017-1-05.

[16] Nærings-og fiskeridepartementet. Forskrift om sommertid. https://lovdata.no/dokument/SF/forskrift/2007-12-14-1420. Accessed: 2017-21-03.

[17] Josef Schumann, Michael J Flannagan, Michael Sivak, and Eric C Traube. Daytime veiling glare and driver visual performance: Influence of windshield rake angle and dashboard reflectance. *Journal of Safety Research*, 28(3):133–146, 1997.

[18] Sourcepole. Openlayers plugin for qgis. https://plugins.qgis.org/plugins/openlayers_plugin/. Accessed: 2017-03-05.

[19] Helmut T Zwahlen. Conspicuity of suprathreshold reflective targets in a driver's peripheral visual field at night. *Transportation Research Record*, 1989.

APPENDIX A
INSTALLATION AND BUILD GUIDE

This installation is for MacOS only. However, the workflow should be pretty similar on other operating systems as well. The installation of the plug-in comes in two parts. First, installation of QGIS and the plug-in will be described. Then, build instructions for the Sun Glare Detection System will be explained. To be able to use the plug-in as is, follow the instructions in Appendix A-A. To be able to edit the SGDS, follow both Appendix A-A and Appendix A-B.

A. Sun Glare Visualization Plug-in Installation

This section will describe the steps taken towards installing QGIS and the Sun Glare Visualization Plug-in.

1) Download and install QGIS from http://www.kyngchaos.com/software/qgis. The version of QGIS used is for development is *2.14.10-Essen*. When installing QGIS, some other packages must be installed as well. These are GDAL, numpy and matplotlib. Install them in the order the install package says.
2) The QGIS plug-in folder must be created. There are two ways of doing this. The first option is to use the Plugin Manager (Plugins → Manage and install plugins...) and install one arbitrary plug-in. The other option is to navigate to the folder ".qgis2/python/" and create a folder named "plugins".
3) Install the plug-in called "OpenLayers Plugin". This is a dependency for the Sun Glare Visualization Plug-in.
4) Download the zipped plug-in from https://drive.google.com/open?id=0ByBkxB6DytDKUVFseHhMb2I2NzQ. Unzip the plugin in the ".qgis2/python/plugins/"-folder.
5) Restart QGIS and install the plug-in from the Plugin Manager.

B. Sun Glare Detection System Build Instructions

If changes to the plug-in or underlying algorithms are to be done, follow this installation as well. This will let the user make changes to the Sun Glare Detection System and build the system.

1) Make sure to have homebrew on your machine. Can be found at https://brew.sh/.
   Version used: *1.1.11*
2) Install cmake using brew.
   brew install cmake
   Version used: *3.4.0-rc1*
3) Install GeographicLib. Download source from https://sourceforge.net/projects/geographiclib/files/distrib/. Unzip the folder. Navigate in to the folder using terminal and issue the following command:
   mkdir build && cd build && cmake .. && make && make install.
   Can also use brew:
   brew install geographiclib
   Version used: 1.46
4) Install Boost using brew.
   brew install boost
   Version used: 1.63.0
5) Install automake using brew.
   brew install automake
   Version used: 1.15
6) Install libtool using brew.
   brew install libtool
   Version used: 2.4.6
7) Install autoconf using brew.
   brew install autoconf
   Version used: 2.69
8) Clone FYBA from github. Can be found at https://github.com/kartverket/fyba.
   Open the file *fyut.h* and input the following at the top of the file.

```
#ifdef __APPLE__
    #include <stdint.h>
    #define UT_INT64 int64_t
#endif
```

   Then issue the following command:
   autoreconf –force –install && ./configure && make && make install.
9) Navigate to the build-folder of the SunGlareVisualizer. "/.qgis2/python/plugins/SunGlareVisualizer/ccode/SunGlareDetector/SunGlareDetector/build/".

Issue the following command:
cmake .. && make

10) If this results in error, try to install GDAL from source.
Can be found at http://trac.osgeo.org/gdal/wiki/DownloadSource. Unzip the folder and issue the following command: *./configure && make && make install*.
Then try the previous step again.

APPENDIX B
CONFIGURATION FILE

This appendix will look at the configuration file used for the plug-in. Figure 21 shows the default configuration file used for the system. These values are the ones the user can tweak. Every line represents one variable in the system. The exact name of a variable must be written in upper-case, just like in Figure 21, followed by a space and the variable's value. The rest of this appendix will explain the use of each of the variables.

```
MORNING_RUSH_START 7
MORNING_RUSH_END 9
EVENING_RUSH_START 15
EVENING_RUSH_END 17
TIME_RESOLUTION 5
UTM_ZONE 32
HORIZONTAL_GLARE_APERTURE 25
VERTICAL_GLARE_APERTURE 25
EYE_HEIGHT 1.5
YEAR 2017
```

Fig. 21.

A. *MORNING_RUSH_START*

Specifies the start time of the morning rush. With a value of 7, the system will start checking for glare at 7:00 a.m., and run until the value of *MORNING_RUSH_END*. Care must be taken to ensure that the value of *MORNING_RUSH_START* is smaller than that of *MORNING_RUSH_END*.

B. *MORNING_RUSH_END*

Specifies the end time of the morning rush. With a value of 9, the system will start checking for glare at the value of *MORNING_RUSH_START*, and run until 9:00 a.m. Care must be taken to ensure that the value of *MORNING_RUSH_END* is larger than that of *MORNING_RUSH_START*.

C. *EVENING_RUSH_START*

Specifies the start time of the evening rush. With a value of 15, the system will start checking for glare at 3:00 p.m., and run until the value of *EVENING_RUSH_END*. Care must be taken to ensure that the value of *EVENING_RUSH_START* is smaller than that of *EVENING_RUSH_END*.

D. *EVENING_RUSH_END*

Specifies the end time of the evening rush. With a value of 17, the system will start checking for glare at the value of *EVENING_RUSH_START*, and run until 5:00 p.m. Care must be taken to ensure that the value of *EVENING_RUSH_END* is larger than that of *EVENING_RUSH_START*.

E. *TIME_RESOLUTION*

Specifies how often the system checks for glare. The value is in minutes. With a value of 5, the system will check for glare every five minutes from *MORNING_RUSH_START* to *MORNING_RUSH_END* and from *EVENING_RUSH_START* to *EVENING_RUSH_START*. With a value of 60, the system will only check for glare once every hour.

F. *UTM_ZONE*

Specifies the UTM zone number for the road data and terrain data. The default value of 32 is the correct number for southern Norway.

G. *HORIZONTAL_GLARE_APERTURE*

Specifies the horizontal limit for glare. For the value 25, the sun has to be within 25 degrees of the driving direction for it to be glare.

H. *VERTICAL_GLARE_APERTURE*

Specifies the vertical limit for glare. For the value 25, the sun has to be within 25 degrees of the driving direction for it to be glare.

I. EYE_HEIGHT

Specifies the height of the eyes of the driver. The system adds this amount to the height of the road before checking for glare. The value is in meters.

J. YEAR

Specifies the year the system should do calculations for.

APPENDIX C
IMAGES OF THE PLUG-IN FUNCTIONALITY

This appendix will show the functionality of the plug-in. All of the functionality of the plug-in can be accessed from the top menu of QGIS, as shown in Figure 22, or from the actions in the action bar in QGIS, as seen in figure 23.

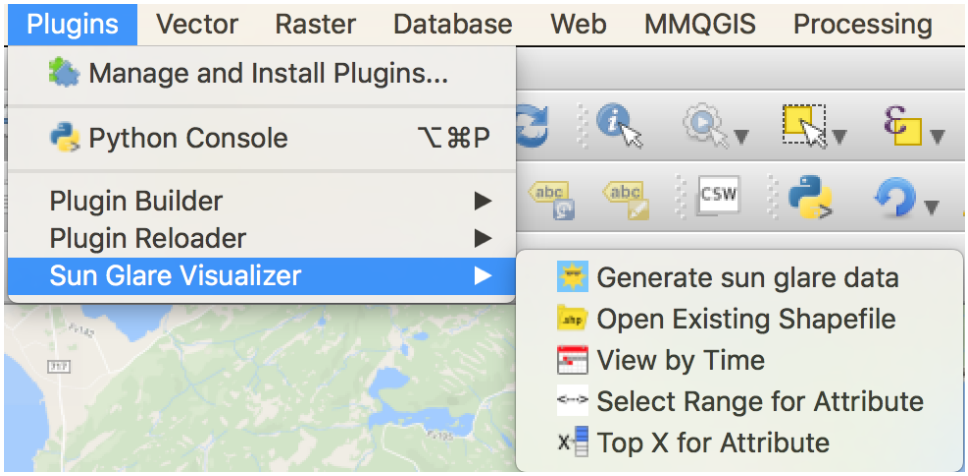


Fig. 22. Shows the available actions from the top menu in QGIS.

Fig. 23. Shows the available actions from the actions menu in QGIS.

A. *Generating Glare Data*

The action seen in Figure 24 is used when the user wants to generate data from within QGIS. Here, the user specifies a path to the SOSI-file, terrain data file, configuration file, output folder and a name for the output shapefile.

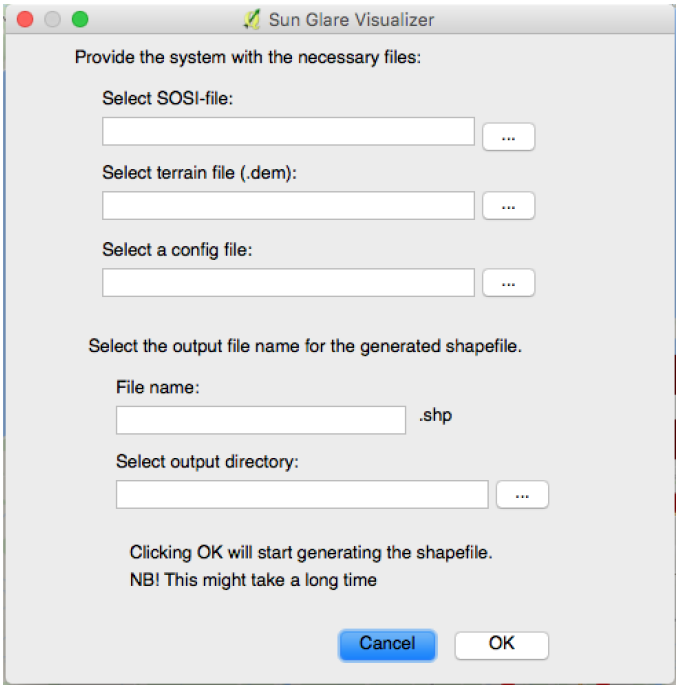


Fig. 24. Screenshot of the UI for the ”Generate sun glare data”-action.

B. Open Existing Shapefile

When the data is generated, it can be imported into the plug-in by using the action seen in Figure 25. The path to the shapefile has to be selected, and an optional layer name can be specified. If no name is specified, ”no-name” is set as the layer name.

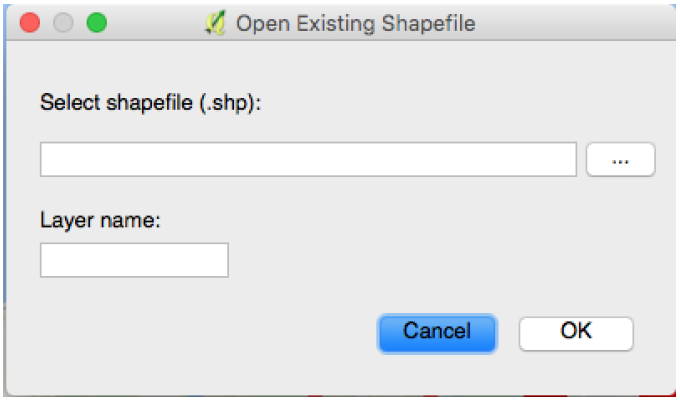


Fig. 25. Screenshot of the UI fir the ”Open Existing Shapefile”-action.

C. Select Values from Time Range

The action seen in Figure 26 can be used to filter out values based on time. The user can chose to filter on a weekly- or monthly basis. Then, the user can select what range of weeks or months it wants to see. There is also an option to only show results from either the morning- or evening rush hour, or both.

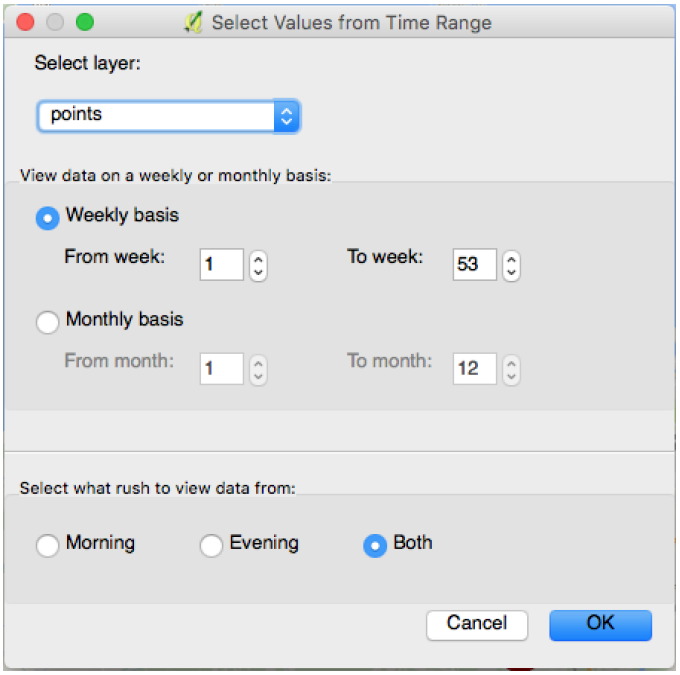


Fig. 26. Screenshot of the UI for the ”Select Values from Time Range”-action.

D. Select Value Range for Attribute

The action seen in Figure 27 can be used to only show points within a specified range. The layer and attribute is set in the combo-boxes. Then, the two sliders are used to set the minimum and maximum value for that attribute.

E. Select Top X points

The action seen in Figure 28 can be used to only render the points with the most glare. The layer and attribute is selected in the combo-box. Then, the filtering type is selected and a value for X is inputted.

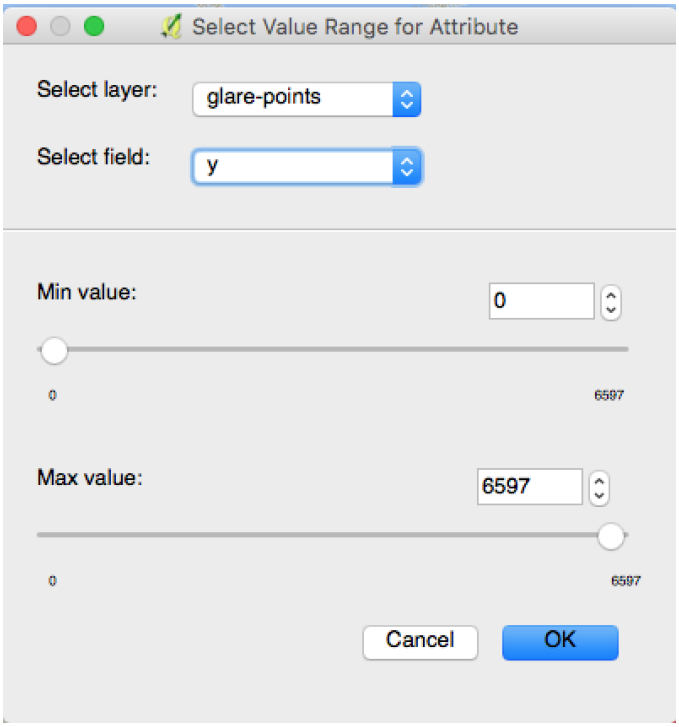


Fig. 27. Screenshot of the UI for the ”Select Value Range for Attribute”-action.

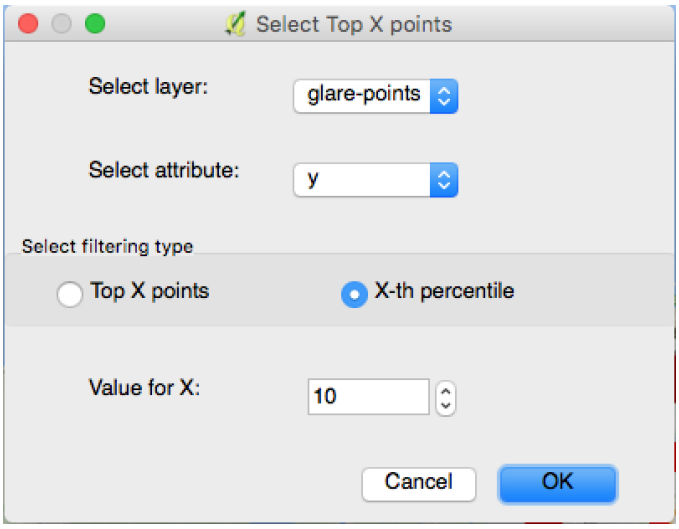
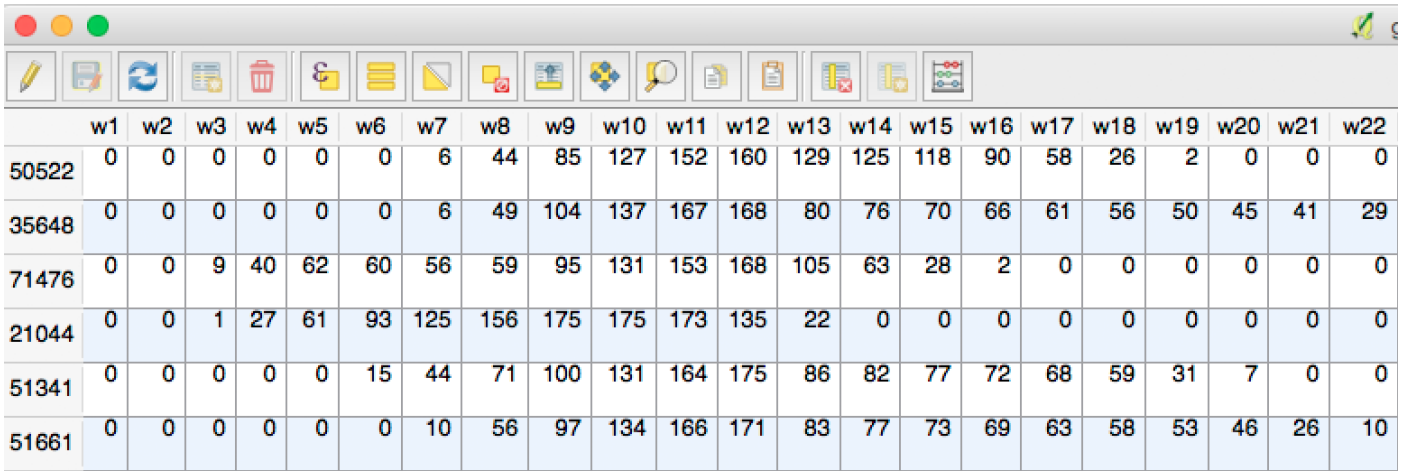


Fig. 28. Screenshot of the UI for the ”Select Top X points”-action.

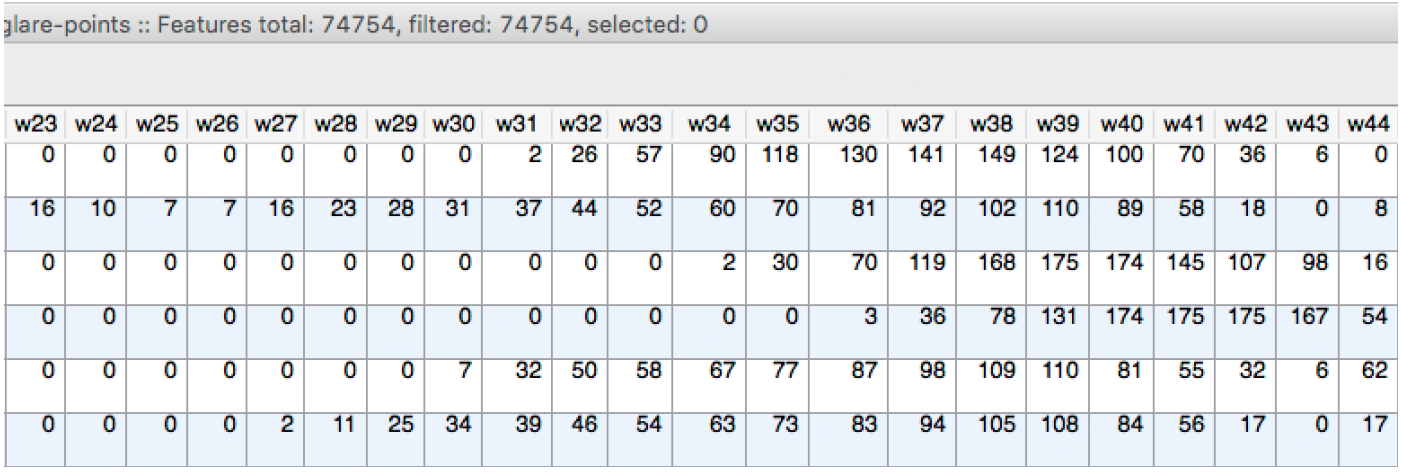
APPENDIX D
SHAPEFILE

This appendix will look at the shapefile generated by the Sun Glare Detection System. A screenshot of six of the features can be seen in Figures 29, 30 and 31. The first 53 attributes are for the glare count for each week. ”w1” represents the first week of the year, and is the first seven days (i.e. January 1st to January 7th). The next 12 attributes are glare count for each month, where ”m1” represents January. The next attribute is the total yearly glare count for the point. The last attribute says if the point had glare during the morning- or evening rush hour.



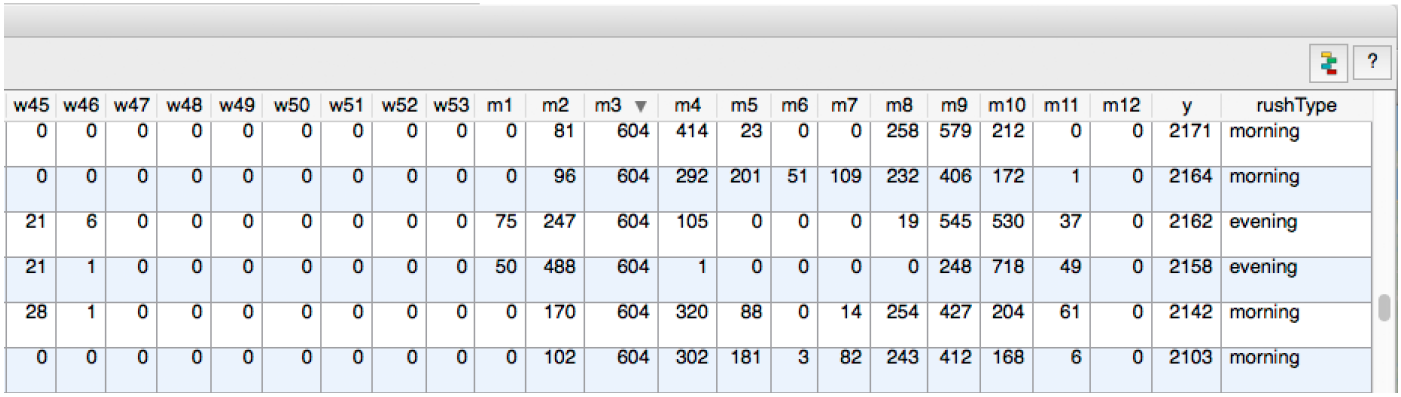
	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14	w15	w16	w17	w18	w19	w20	w21	w22
50522	0	0	0	0	0	0	6	44	85	127	152	160	129	125	118	90	58	26	2	0	0	0
35648	0	0	0	0	0	0	6	49	104	137	167	168	80	76	70	66	61	56	50	45	41	29
71476	0	0	9	40	62	60	56	59	95	131	153	168	105	63	28	2	0	0	0	0	0	0
21044	0	0	1	27	61	93	125	156	175	175	173	135	22	0	0	0	0	0	0	0	0	0
51341	0	0	0	0	0	15	44	71	100	131	164	175	86	82	77	72	68	59	31	7	0	0
51661	0	0	0	0	0	0	10	56	97	134	166	171	83	77	73	69	63	58	53	46	26	10

Fig. 29. Shows the first 22 attributes of the generated shapefile.



w23	w24	w25	w26	w27	w28	w29	w30	w31	w32	w33	w34	w35	w36	w37	w38	w39	w40	w41	w42	w43	w44
0	0	0	0	0	0	0	0	2	26	57	90	118	130	141	149	124	100	70	36	6	0
16	10	7	7	16	23	28	31	37	44	52	60	70	81	92	102	110	89	58	18	0	8
0	0	0	0	0	0	0	0	0	0	0	2	30	70	119	168	175	174	145	107	98	16
0	0	0	0	0	0	0	0	0	0	0	0	0	3	36	78	131	174	175	175	167	54
0	0	0	0	0	0	0	7	32	50	58	67	77	87	98	109	110	81	55	32	6	62
0	0	0	0	2	11	25	34	39	46	54	63	73	83	94	105	108	84	56	17	0	17

Fig. 30. Shows attribute 23-44 of the generated shapefile.



w45	w46	w47	w48	w49	w50	w51	w52	w53	m1	m2	m3 ▼	m4	m5	m6	m7	m8	m9	m10	m11	m12	y	rushType
0	0	0	0	0	0	0	0	0	0	81	604	414	23	0	0	258	579	212	0	0	2171	morning
0	0	0	0	0	0	0	0	0	0	96	604	292	201	51	109	232	406	172	1	0	2164	morning
21	6	0	0	0	0	0	0	0	75	247	604	105	0	0	0	19	545	530	37	0	2162	evening
21	1	0	0	0	0	0	0	0	50	488	604	1	0	0	0	0	248	718	49	0	2158	evening
28	1	0	0	0	0	0	0	0	0	170	604	320	88	0	14	254	427	204	61	0	2142	morning
0	0	0	0	0	0	0	0	0	0	102	604	302	181	3	82	243	412	168	6	0	2103	morning

Fig. 31. Shows attribute 45-67 of the generated shapefile.