Fast traffic noise mapping of cities using the Graphics Processing Unit of a personal computer

Erik M. SALOMONS[1]; Han ZHOU[2]; Walter J.A. LOHMAN[3]

TNO, Netherlands Organisation of Applied Scientific Research, Delft, The Netherlands

ABSTRACT

Traffic noise mapping of cities requires large computer calculation times. This originates from the large number of point-to-point sound propagation calculations that must be performed. In this article it is demonstrated that noise mapping calculation times can be reduced considerably by the use of parallel computation on the Graphics Processing Unit (GPU) of a personal computer. Comparisons are presented between a GPU implementation and a conventional CPU implementation for various urban areas, from which a GPU speedup factor of 720 is obtained (compared with a single CPU). The complete traffic noise map of Amsterdam is calculated with the GPU implementation in 2 h. Local modifications of the noise map, for example to investigate the effect of a new building or a road, are calculated in a time of the order of a minute.

Keywords: traffic noise, noise mapping, parallel computation
I-INCE Classification of Subjects Numbers: 52.3, 76.1

1. INTRODUCTION

Traffic noise in cities affects the lives and health of a large number of people. To regulate and control the effects of urban traffic noise, the European Commission requires that major EU cities produce noise maps and corresponding noise exposure distributions of their inhabitants (1). The first two rounds of noise mapping took place in 2007 and 2012. The noise exposure distribution of a city is based on building exposures and numbers of people living in the buildings. Building exposure is represented by a noise level at the most exposed facade of the building. Both the day-evening-night level L_{den} and the night level L_{night} are considered.

To calculate a noise map, the road network is represented by a large number of road segments, and the vehicles on the road segments are represented by average vehicle flows and driving speeds. A road segment is divided into subsegments for the calculation of sound propagation from a road to a receiver. Each subsegment is represented by a point source, and the basic noise mapping operation is a point-to-point calculation from a point source to a receiver. For the noise map of Amsterdam (see Sec. 4), for example, the number of point-to-point calculations is of the order of 10^{10}. Consequently, there is a great practical need for efficient noise mapping algorithms.

A noise mapping algorithm is basically a computer implementation of a mathematical model for traffic noise emission and propagation in an urban area. The model is usually referred to as an engineering model, since engineering approximations are inevitable for noise mapping of a complete city. The engineering model should describe how road segments are divided into subsegments and how the contribution from a subsegment to the sound level at a receiver is calculated, taking into account various attenuation terms such as geometrical attenuation, air absorption, ground attenuation, and barrier attenuation. In addition, the model should specify how reflections by vertical walls, such as noise barriers and building facades, are taken into account. Each reflection corresponds to an additional point-to-point calculation. Consequently, the total number of reflections that is specified by the model is an important parameter for the calculation time.

For the first two rounds of EU noise mapping in 2007 and 2012, various national engineering

[1] erik.salomons@tno.nl
[2] han.zhou@tno.nl
[3] walter.lohman@tno.nl

models for traffic noise have been used. Owing to differences between the national models, the comparison of noise maps from different EU member states is a bit ambiguous. To solve this problem, the European Commission currently develops a harmonized traffic noise model to be used in future noise mapping rounds by all member states. For the present article, however, the differences between the various noise models are not essential. All models have a similar basic structure, with the point-to-point calculation as the basic operation, while only the details of the calculation of the attenuation terms differ. In this article we use the Dutch standard traffic noise model (2,3), which we refer to as DSM (Dutch Standard Method). The DSM model is similar to the international ISO 9613-2 model (4).

The objective of this article is to describe an efficient implementation of the DSM model. The implementation runs on a 'normal' personal computer, and employs *parallel computation* on the Graphics Processing Unit (GPU) of the computer. The performance of GPUs has increased considerably in recent years, and the CUDA architecture (Compute Unified Device Architecture) developed by NVIDIA (5) has made it possible to utilize GPUs for general purpose computations and various environmental applications, such as air pollution modelling (6). For noise mapping, it is shown in this article that parallel computation on the GPU results in a considerable increase of computational speed, compared with the speed of a conventional CPU implementation. The noise mapping calculations are decomposed into a large number of basic operations, which are handled efficiently by the large number of calculation threads of a GPU.

The application of parallel computation of noise maps on multi-core computers has been described before by other researchers (7,8). In the present article we focus on parallel computation on a GPU. Figure 1 illustrates the high computational speed, expressed as number of floating-point-operations (FLOPS) per second, that can be achieved with GPU in comparison with CPU. The curves in the figure are based on GPU and CPU processors developed in the period 2002-2012, as reported on the NVIDIA website (5).

It should be noted that it is not straightforward to achieve the high GPU speeds in Fig. 1 in practice. One has to design the computer code very carefully, taking into account the constraints imposed by the GPU hardware.

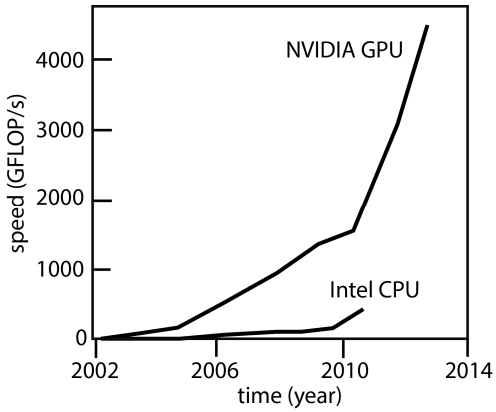

Figure 1 – Development in the period 2002-2012 of computational speed of GPU and CPU processors, expressed as GFLOPS/s (1 GFLOP = 10^{12} floating point operations), for single precision calculations.

The performance increase achieved with parallel computation has been utilized in an *interactive* implementation of the DSM noise mapping model (9). The idea of interactive noise mapping is that the effects on a noise map of various infrastructural changes or traffic flow changes are calculated within a very short time, preferably of the order of a minute. In this way, stakeholders of the noise map (city planners or road authorities, for example) can directly observe the effects of proposed changes in an interactive session. Interactive noise mapping is a powerful tool for developing cost effective noise reductions measures, for example in the framework of EU noise action plans (1) aimed at the development of solutions for so-called hot spots on the noise map of a city.

2. NOISE MAPPING METHOD

This section presents a global description of the DSM model. Details can be found in the original

(Dutch) description of the model (2), or in an English description of the model (3).

The model distinguishes three types of vehicles: light vehicles (automobiles), medium-heavy vehicles (light trucks and buses), and heavy vehicles (heavy trucks). A vehicle is represented by a point source at a height of 0.75 m. Different speed-dependent octave-band emission spectra are used for the three vehicle types, including spectral correction terms for road surfaces such as low-noise asphalt. Numbers of vehicles per unit road length are determined from vehicle flows (numbers of vehicles per hour) and driving speeds for the three vehicle types.

The road network consists of a number of segments. For the noise calculation each segment is divided into a number of smaller segments. In this article we refer to the original road segments as *segments*, while we refer to the smaller segments as *subsegments*. Two types of subsegmentation may be used, according to the model description: fixed angular sectors of 2 degrees or variable angular sectors of at most 5 degrees (angular sectors are defined in the horizontal plane from the viewpoint of the receiver; see Fig. 2). For the calculations presented in this article (see Sec. 4), fixed angular sectors of 2 degrees were used.

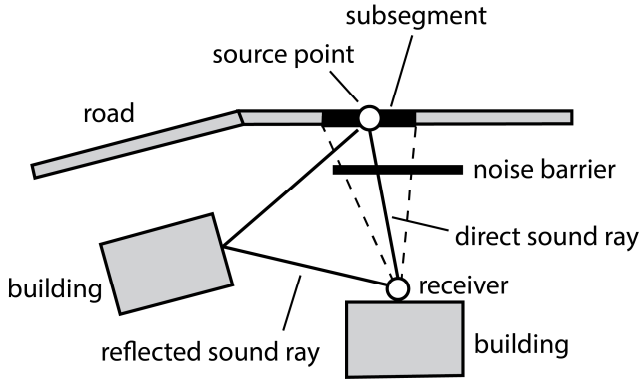

Figure 2 – Top view of a model situation, with a direct sound ray and a reflected sound ray.

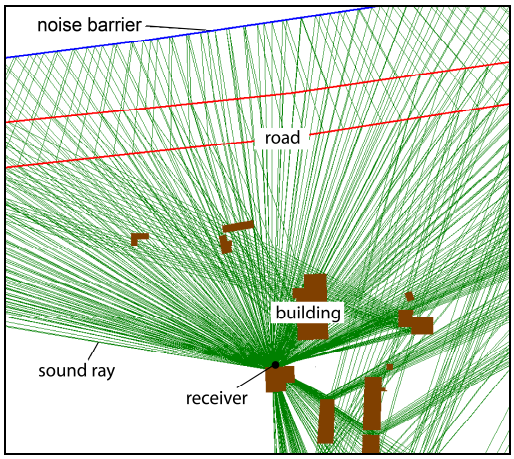

Figure 3 – Direct and reflected sound rays in a practical situation.

Source points are placed at the centers of the subsegments. The sound level at a receiver is calculated by logarithmic summation of contributions from sound rays between source points and the receiver. The model takes into account sound rays with zero or one reflection by vertical surfaces (building facades, noise barriers). Multiple reflections may be included, but are ignored here. Figure 3 shows direct and reflected sound rays in a practical situation.

Ground reflections are taken into account indirectly by a ground attenuation term, which is a function of the source and receiver positions and a parameter that characterizes the acoustic absorption of the ground, averaged over the source-receiver line. The ground attenuation term is valid for downward-refracting propagation conditions. A meteorological correction term is included to account for upward-refracting conditions, so the result is an estimate of a long-term average sound level.

If the ground projection of a sound ray intersects one or more vertical walls (buildings walls, noise barriers), then the model takes into account a screening attenuation. First the screening attenuation is calculated for all intersected walls, and next the highest value of all screening attenuations is taken into account. The screening attenuation is calculated with a formula for sound diffraction by a thin screen, with an upper limit of 25 dB.

The sound level at a receiver is calculated with the following formula

$$L = 10\lg \sum_{i=1}^{8} \sum_{j=1}^{N} 10^{L_{i,j}/10} \qquad (1)$$

where the sum is over eight octave bands (63 Hz to 8 kHz) and N sound rays. The set of sound rays includes direct and reflected rays from all source points. The contribution from sound ray j for octave band i is given by

$$L_{i,j} = L_{\mathrm{E},i,j} - \Delta L_{\mathrm{geo},j} - \Delta L_{\mathrm{air},i,j} - \Delta L_{\mathrm{ground},i,j} - C_{\mathrm{met},j} - \Delta L_{\mathrm{screen},i,j} - \Delta L_{\mathrm{refl},i,j}\,. \qquad (2)$$

The first term on the right represents the emission level (sound power level per unit length, summed logarithmically over the three vehicle types). The remaining terms represent the geometrical attenuation, air absorption, ground attenuation, meteorological correction term, screening attenuation, and reflection attenuation, respectively. The default reflection attenuation is 1 dB per reflection for all frequencies, corresponding to an absorption coefficient of 0.8.

Equation (1) is a general formula for the sound level at the receiver. To calculate the day-evening-night level, Eq. (1) is used with separate values of the vehicle flows for the day, evening, and night periods. This results in day, evening, and night sound levels, which are combined into the day-evening-night level, taking into account penalties of 5 and 10 dB for the evening and night periods, respectively.

3. NUMERICAL IMPLEMENTATION

3.1 Input Data

Input data for a noise map calculation consist of four types of elements: i) roads, ii) receivers, iii) ground, and iv) noise barriers and buildings.

Road are represented by line segments. For each segment the following data is available: three-dimensional coordinates of the end points of the segment, vehicle flows and speeds for the three vehicle types, and road surface type. Road segments are typically 100 m long. Different driving lines on a road are often combined into one or two average driving lines. These average driving lines are referred to as roads in this article.

Receivers are represented by points. Each point is represented by three-dimensional coordinates.

Ground data are represented by polygons. Each polygon is represented by the coordinates of the vertices and the ground type. From the ground type the acoustical ground absorption parameter is determined.

Noise barriers and buildings are represented by sequences of vertical walls. The walls are referred to as *screens* in this article. Each sequence is represented by the coordinates of the vertices on the ground plane and the heights of the vertices. A building is represented by a closed sequence of screens. A building that encloses an open space, such as a courtyard, is represented by two closed sequences, with opposite rotational directions (clockwise and anti-clockwise). The absorption coefficients of the barrier surfaces and building facades are also included in the data.

3.2 Basic Calculation Scheme

In this section we describe the basic calculation scheme. In the next section we describe the efficient implementation with parallel computation on a GPU.

A noise map represents sound levels at a large number of receivers. The calculations for different receivers are independent of each other. The sound level at a receiver is calculated with Eq. (1). The sum over sound rays j in Eq. (1) includes direct rays and reflected rays. There are two restrictions to the sound rays that are included:

a) only rays with at most $N_{\max}$ reflections per ray are included,

b) only rays with lengths up to $R_{\max}$ are included.

For the results presented in this article we used $N_{\max} = 1$ and $R_{\max} = 1000$ m, unless indicated otherwise.

The basic steps of the calculation of the sound levels, with contributions $L_{i,j}$ from direct rays and reflected rays, are described below. Figure 4 shows the steps of the calculation scheme in a flow diagram.

Direct sound rays - For the calculation of direct rays to a receiver, a subset of the road segments and a subset of the screens is used. The subsets include road segments and screens that lie within a circle with radius $R_{\max}$ around the receiver. Each road segment is divided into a number of subsegments. A source point is chosen at the center of each subsegment. For each source point a direct sound ray to the receiver is taken into account. The attenuation terms in Eq. (2) are calculated based on the ground sections and screens that are intersected along the sound ray. For computational efficiency, the ground polygons and screens are discretized on a ground grid with square cells of 10 x 10 m. The grid allows fast selection of screens that are intersected and fast calculation of the average ground absorption along the sound ray.

Reflected sound rays - The road segments, subsegments, and screens within a circle with radius $R_{\max}$ around the receiver, as selected already for the direct rays, are also used for the reflected rays.

After geometrical construction of the reflected rays, only rays with total path length shorter than R_{max} are taken into account. The geometrical construction is based on equal angles of incidence and reflection at a reflecting screen. The construction is compute intensive. For each source point – screen – receiver combination, first the *image* receiver with respect to screen reflection is determined, and next it is determined if the direct ray from the image receiver to the source point intersects the screen. This is illustrated in Fig. 5. The geometrical construction can be generalized to higher-order reflections. Each source point – screen – receiver combination that satisfies the geometrical construction corresponds to a reflected sound ray, and therefore to a contribution to the sum over source rays *j* in Eq. (1). The attenuation terms in Eq. (2) are calculated in the same way as for the direct sound rays, making use of the ground grid with cells of 10 x 10 m for computational convenience.

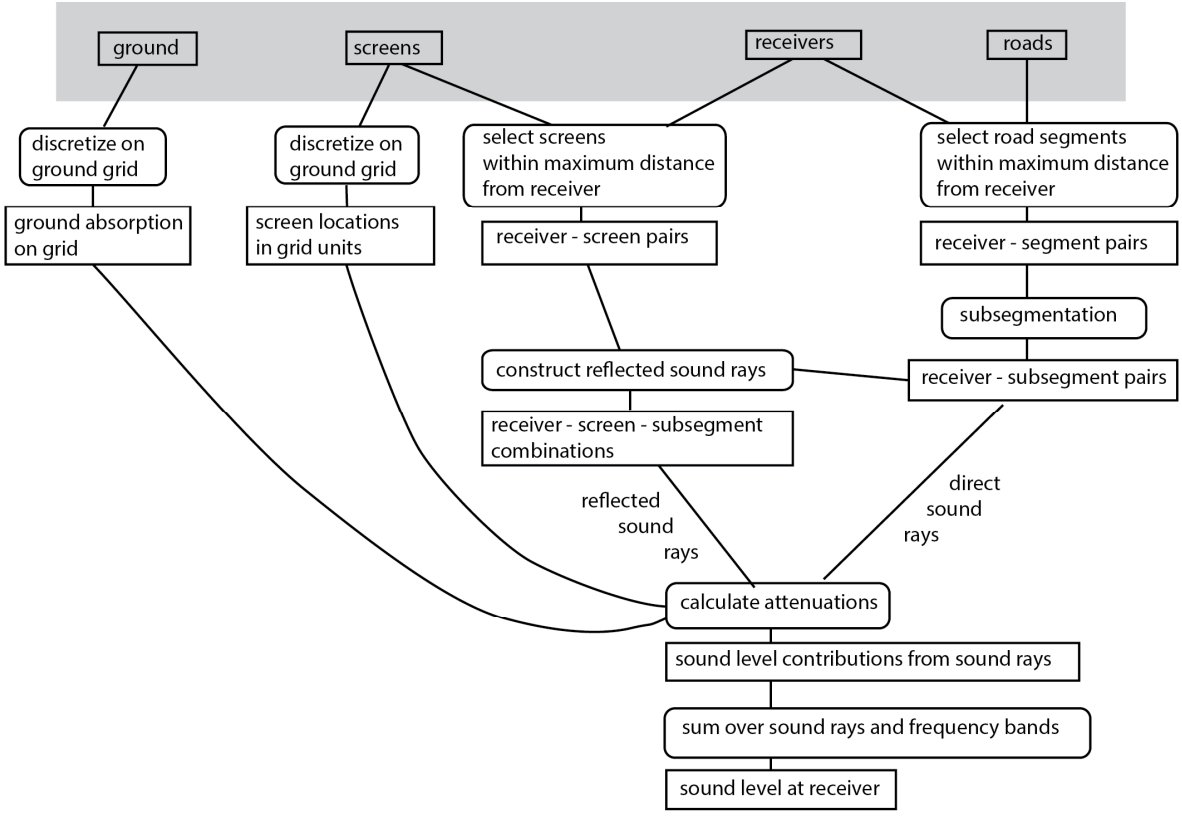

Figure 4 – Flow diagram of the calculation of the sound level at a receiver. Rectangular blocks represent data or intermediate calculation results and blocks with rounded corners represent calculation steps.

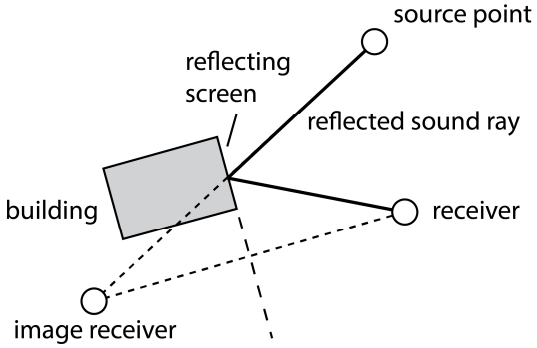

Figure 5 – Schematic illustration of a sound ray reflected by a building facade.

3.3 Parallel Implementation on a GPU

The calculation scheme described in the previous section specifies how the sound level at a receiver is calculated. For each receiver a large number of sound rays is involved, corresponding to a large number of subsegments (point sources) and reflecting screens. Moreover, for a noise map the

calculation must be performed for a large number of receivers.

A conventional implementation of the calculation scheme on a CPU leads to a large number of loops in the computer code, such as loops over receivers, subsegments, and screens. A more efficient implementation on a GPU employs the large number of calculation threads of the GPU, each performing the same computational operation but with different input and output data. In other words, the loops are unrolled in a parallel implementation on a GPU.

The calculation represented by the flow diagram in Fig. 4 can be divided into two parts:

- part 1: determination of receiver – (screen) – subsegment combinations,
- part 2: calculation of attenuations.

The calculations in part 1 are primarily geometrical calculations. The calculations for determining receiver – subsegment pairs for direct rays are simple and require only small calculation times. The calculations for determining receiver – screen – subsegment combinations for reflected rays are more complex. In particular the geometrical construction of the reflected paths is time consuming. Here parallel computation is applied efficiently in our implementation. Reflecting screens are found for different receiver – subsegment pairs by parallel geometrical calculations on different calculation threads. Finally, parallel computation is also employed in part 2, the calculation of the attenuations for all direct and reflected sound rays.

A major challenge with code optimization through parallel implementation on a GPU is to reduce the 'idle times' of different calculation threads. Ideally, all calculation threads are busy all the time. In practice, calculation tasks on different threads require different calculation times due to input differences, so threads have to wait until all threads have completed their tasks. If all idle times are reduced to zero, one reaches the high theoretical calculation speed with a GPU shown in Fig. 1. For the parallel implementation of the noise model described in this article we achieved about 50% of the theoretical speed. It should be noted that the GPU and CPU implementations have basically equal accuracy.

4. NUMERICAL EXAMPLE

4.1 Introduction

In this section we present a numerical example of the speedup that can be obtained with a parallel implementation of the DSM noise model on the GPU of a personal computer. The example is for the city of Amsterdam. Figure 6 shows the full noise map of Amsterdam, which was calculated with a GPU implementation of the noise model. The noise map is based on calculated noise levels at 1.2 million receivers. The total calculation took 123 minutes on the personal computer, with a 2.4 GHz CPU (Intel Xeon E5620, 6GB RAM) and an NVIDIA GPU (GeForce GTX 680, 4GB, 1536 cores).

To determine the speedup obtained with the GPU, we have compared calculation times of the GPU implementation with calculation times of a commercial CPU implementation of the DSM noise model (10). Both implementations were run on the same personal computer. The CPU implementation uses two CPU cores of the personal computer for a noise mapping calculation. To distinguish between the two computer implementations, we refer to the GPU version and the CPU version of the model implementation.

It was not possible to calculate the full noise map of Amsterdam with the CPU version, within a reasonable time. We estimated that this would take a calculation time of the order of 1000 h (see Sec. 4.4). Therefore we decided to perform calculations for different subareas of Amsterdam and compare calculation times between the GPU version and the CPU version.

In general, two areas are distinguished for a noise mapping calculation:

i) The receiver area
ii) The total urban area.

The receiver area is the area where receiver locations are chosen, with a density that is sufficient to determine the continuous noise map in the area by interpolation. Receivers are chosen along the roads, along the facades of buildings, and at various positions in open areas. In this case we have chosen a receiver height of 1.5 m.

The total urban area is the area that contains all the noise sources (road segments) and buildings that are taken into account in the calculation. The total urban area may be the complete area of the city (Amsterdam in this case), but for reasons of efficiency one usually chooses a smaller area, for example a circular area with a radius of 1 km around the receiver area (as described in Sec. 3).

The calculation time of a noise mapping calculation depends on the choices of the receiver area and

the total urban area: it is proportional to the number of receivers and it increases with increasing size of the total urban area. We have chosen three receiver areas for the calculations with the CPU and GPU versions. As explained in the next sections, we have chosen different total urban areas for the CPU and GPU versions.

The three receiver areas, denoted as A1-A3, are square areas centered at a point in Amsterdam at position (120.5, 486.5), expressed in Dutch RD coordinates in kilometers. The central point is referred to as point C, and is located near the central point of the noise map shown in Fig. 6. The three areas are specified in Table 1. Parameter X_r is the length of the side of the square area. Parameter N is the number of receivers in the area. As an example, Fig. 7 shows area A3 (400 x 400 m, 1999 receivers). Black dots are receivers, grey areas are buildings, and black lines are roads (or rather, source lines representing one or more driving lines).

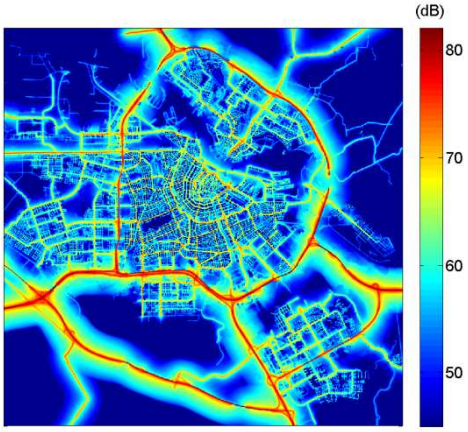

Figure 6 – Noise map of Amsterdam, showing the L_{den} noise level at height 1.5 m. The lower left and upper right corners have Dutch RD coordinates (in km) of (113, 477) and (130, 494), respectively.

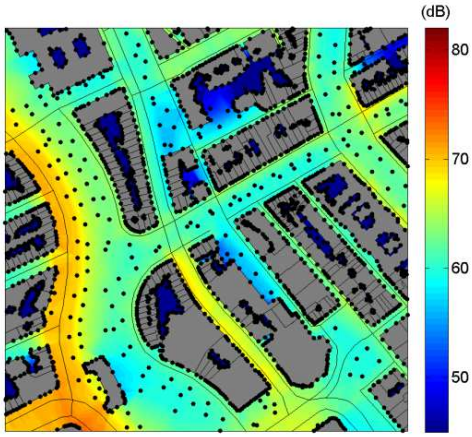

Figure 7 – Part of the noise map from Fig. 6 in receiver area A3 (400 x 400 m). Black dots are receivers, grey areas are buildings, and black lines are roads (source lines).

Table 1 – Specification of the three square receiver areas A1-A3, centered at point C.

Receiver area	side X_r, m	N, number of receivers
A1	100	135
A2	200	618
A3	400	1999

4.2 Calculations with the CPU version of the noise model

Table 2 specifies parameters and calculation times of the calculations performed with the CPU version. Time T is the time of a calculation including reflections (so sound paths with 0 and 1 reflections are taken into account) and time T_0 is the time of a calculation without reflections (so only direct sound paths are taken into account).

As indicated in the table, calculations were performed for the three receiver areas A1-A3 and various total urban areas. The total urban area is a cross section of a circular and a square area.

The square area, with side X_u, was selected manually before the actual calculation was performed. This approach was necessary since loading the full Amsterdam geometry slows down the calculation with the CPU version. The approach is consistent with the noise mapping approach of the CPU version in practice: the model is divided into square areas (tiles) which are handled sequentially.

The circular area is determined by the radius R_{max} introduced in Sec. 3. Radius R_{max} is a general model parameter and also an input parameter of the CPU version. With this parameter the user controls the size of the area around the receiver that contains the sources and other objects taken into account in the calculation. To derive an approximate expression for the total urban area, we replace the circles for different receivers by a single circle around point C. This is a good approximation since the receiver area is small compared with the total urban area.

The expression for area A of the total urban area is $A = 8\int_0^{\pi/4} \frac{1}{2} r^2 \, d\phi$ with $r = \min\left(\frac{1}{2} X_u / \cos\phi, R_{max}\right)$. The integral is easily evaluated numerically, which resulted in the values of A given in Table 2.

It should be noted that the CPU version allows the use of two additional cutoff distances (in addition to the cutoff distance R_{max} for the direct source-receiver distance): one for the distance reflector-source and one for the distance reflector-receiver. In this case we have chosen these two cutoff distances equal to R_{max}. This additional filtering is applied after reflection paths have been constructed, and has only a minor effect on the calculation time.

The calculation times T and T_0 are plotted in Fig. 8 as a function of the product $N\,A$. The lines are least square fits to the points. The slope of the line representing T as a function of $N\,A$ is 1.03 (in a double logarithmic plot), so close to unity. Consequently, the calculation time is proportional to $N\,A$ in good approximation. The slope of the line representing T_0 as a function of $N\,A$ is 0.91.

Further, calculation time T is about a factor of 10 larger than calculation time T_0. This implies that the calculation time of a full noise mapping calculation, including reflections, is dominated by the time required for the reflected sound paths.

Table 2 – Parameters and calculation times of the calculations with the CPU model. Time T is the time of a calculation with reflections (including sound paths with 0 and 1 reflections) and time T_0 is the time of a calculation without reflections (so only direct sound paths are included).

	Receiver area side X_u, m	N	Urban area side X_u, m	Radius R_{max}, m	Area A, km^2	Time T, s	Time T_0, s
A1	100	135	350	1000	0.123	28	6
A2	200	618	350	1000	0.123	140	21
A1	100	135	700	1000	0.491	126	13
A2	200	618	700	1000	0.491	546	56
A2	200	618	700	250	0.197	270	32
A1	100	135	1400	1000	1.96	532	63
A2	200	618	1400	1000	1.96	2450	259
A3	400	1999	1400	1000	1.96	8063	728

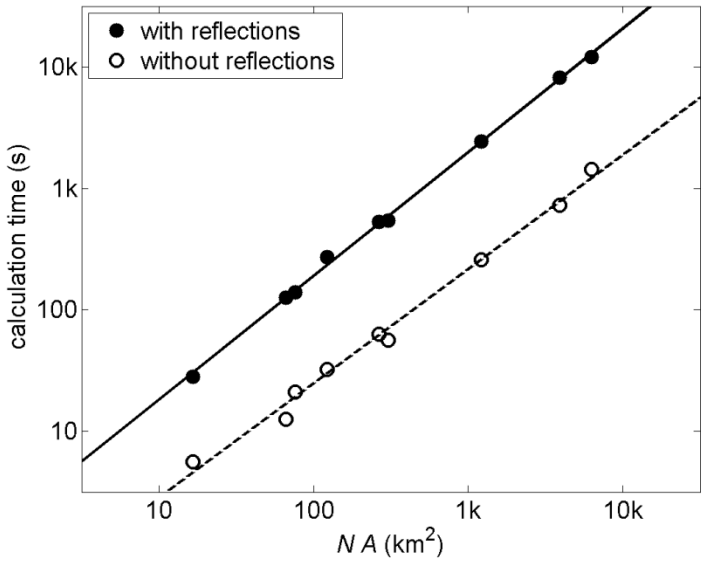

Figure 8 – Calculation times T (filled circles) and T_0 (open circles) as a function of product $N\,A$.

4.3 Calculations with the GPU version of the noise model

Next we describe the calculations with the GPU version. In this case we do not need to select a square urban area of limited size, since the full Amsterdam geometry is handled efficiently by the GPU version and the calculation is not slowed down by it. We simply selected the full Amsterdam geometry and performed calculations for different values of cutoff radius R_{max}. The cutoff radius is employed in the same way as with the CPU version: all sources and objects outside the circle with radius R_{max} around the receiver are eliminated in an initial stage of the calculation. Consequently, the surface area A of the total urban area introduced before is given by $A = \pi R_{max}^2$ in this case.

In addition to the application of cutoff distance R_{max} for the direct source-receiver distance, reflected paths longer than R_{max} are eliminated (in a later stage of the calculation). This approach differs from the approach of the CPU implementation with separate cutoff distances for the two parts of a reflected sound path.

Table 3 specifies parameters and the calculation times of the calculations with the GPU version. Also included is a calculation of the complete noise map of Amsterdam, with about 1.2 million receivers, and a calculation time of 123 minutes (7380 seconds).

Table 3 – Parameters of calculations with the GPU version.

Receiver area	N	Radius R_{max}, m	Time T, s
A1	135	1000	4
A2	618	1000	12.8
A3	1999	1000	37
A3	1999	2000	166
full city	1187574	1000	7380

4.4 Comparison of GPU and CPU Calculation Times

Figure 9 shows the calculation time T as a function of the product NA, both for the CPU version (see Table 2 and Fig. 8) and for the GPU version (see Table 3). The lines represent least squares fits to the points. The line for the GPU points has a slope of 0.84, so with increasing product NA the GPU calculation time per receiver (T/N) decreases.

To determine the GPU speedup factor we consider the region around NA = 10k, since the calculations in this region correspond to realistic practical noise mapping situations with urban areas of the order of 3 km^2 (see Table 2). From the least squares fits we find that the calculation time at NA = 10k is 2069 s for the CPU version and 57.4 s for the GPU version. Consequently, the GPU speedup factor is 2069/57.4, or 360, compared with the CPU version employing two CPU cores. Compared with a single CPU core, the GPU speedup factor is 720.

We also estimated the calculation time that the CPU version would need to calculate the full noise map of Amsterdam. The city was divided into 63 square tiles of 3 x 3 km, surrounded by a buffer of 1 km, and we estimated the calculation times for the different tiles (from calculation time extrapolations provided by the CPU version). We found that the total time is of the order of 1000 h, which is a factor of 500 longer than the GPU calculation time of 123 minutes. The factor of 500 is in good agreement with the factor of 360 given above.

5. CONCLUSIONS

We have shown that traffic noise mapping can be performed efficiently by parallel implementation on the GPU of a personal computer. The engineering models commonly used for noise mapping allow efficient unrolling of various loops in the computer code, employing the large number of calculation threads of a GPU. As an example we have presented calculations for the traffic noise map of Amsterdam. The complete noise map of Amsterdam was calculated with a GPU version of the noise model in 123 minutes. We estimated that the calculation time with a conventional CPU version would be of the order of 1000 hours. From various calculations for subareas of Amsterdam with the GPU and the CPU versions, we concluded that the GPU speedup factor is 720, compared with a single CPU core.

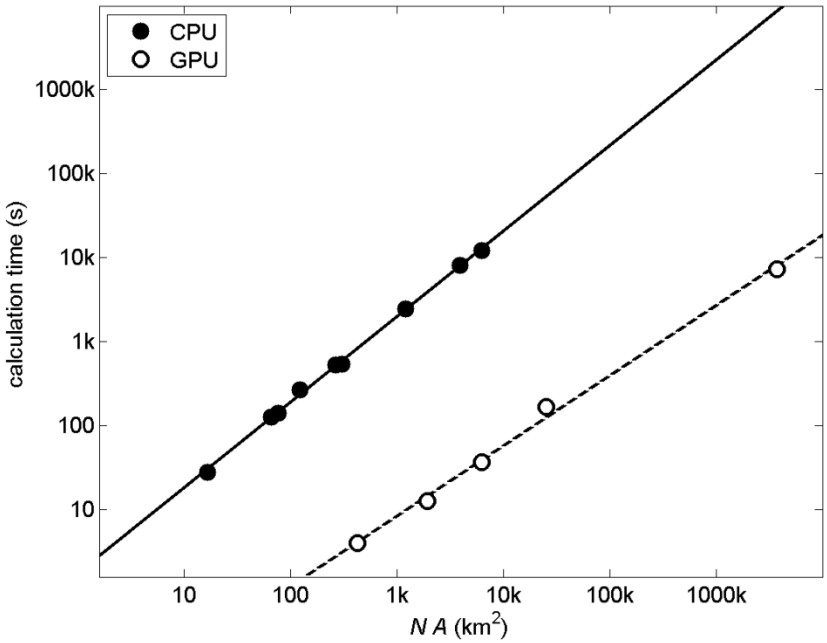

Figure 9 – Calculation times of the CPU version (2 cores) and the GPU version (1536 cores), as a function of the product $N\,A$.

ACKNOWLEDGEMENTS

We are grateful to George van Venrooij (Organic Vectory) for help with the GPU implementation.

REFERENCES

1. Directive 2002/49/EC of the European Parliament and the Council of 25 June 2002 relating to the assessment and management of environmental noise. Official Journal of the European Communities, L 189/12, 18 July 2002.
2. Dutch standard model for road traffic noise – version 2012. The model is described in the document 'Calculation and measurement method for environmental traffic noise 2012', which can be downloaded from http://www.infomil.nl. The description is in Dutch. An English description of a previous version of the model can be found in Ref. (3).
3. E.M. Salomons, H. Polinder, W.J.A. Lohman, H. Zhou, H.C. Borst, and H.M.E. Miedema, "Engineering modeling of traffic noise in shielded areas in cities", J. Acoust. Soc. Am. 2009; 126: 2340-2349.
4. ISO 9613-2, Acoustics—Attenuation of sound during propagation outdoors—Part 2: General method of calculation, ISO, 1996. URL: http://www.iso.org.
5. NVIDIA website http://www.nvidia.com. The data used for Fig.1 in this article were derived from the CUDA programming guide, available through the NVIDIA website (http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html).
6. F. Molnár et al Jr., T. Szakály, R. Mészáros, and I. Lagzi, "Air pollution modelling using a Graphics Processing Unit with CUDA", Computer Physics Communications 2010; 181: 105-112.
7. A. Czyzewski and M. Szczodrak, "Software for calculation of noise maps implemented on supercomputer", TASK Quarterly, Scientific Bulletin of the Academic Computer Centre in Gdansk, 2009; 13(4): 363-377, 2009. See http://www.task.gda.pl/quart/09-4.html
8. W. Probst, "Multithreading, parallel computing and 64-bit mapping software – advanced techniques for large scale noise mapping", Proc 34th DAGA meeting, 2008, Dresden.
9. H.C. Borst, E.M. Salomons, W.J.A. Lohman, H. Zhou, and H.M.E. Miedema, "Urban Strategy: Noise mapping in instrument for interactive spatial planning" Proc Euronoise 2009, Edinburgh, Scotland.
10. Geomilieu version 2.31, developed by the Dutch firm DGMR. See http://dgmrsoftware.nl.