A Novel 2.5D Shadow Calculation Algorithm for Urban Environment

Sukriti Bhattacharya, Christian Braun and Ulrich Leopold
Department for Environmental Research and Innovation (ERIN), Luxembourg
Luxembourg Institute of Science and Technology (LIST), Luxembourg

Keywords: 2.5D Shadow, Tensorflow, Bresenham's Algorithm.

Abstract: This paper proposes a novel efficient algorithm to calculate a 2.5D shadow map based on a coherent mathematic formula concerning the sun's position in a specific location, date and time. This work attempts to improve the understanding of the underlying equations and data structures from an analytical, a geometric and a dynamical systems perspective. By using scalable tensor data structure and inherent parallelism offered by data-flow based implementation the proof of concept is developed to test the technical feasibility of the proposed algorithm. Results show noticeable and significant improvements in overall performance keeping accuracy at negligible differences.

1 INTRODUCTION

The Sun is the most significant light source in the solar system which beams light from a single intent in space; there's only one ray of light that can potentially hit the surface point. So the associated intensity level can therefore only be fully lit (100%) or fully shadowed (0%), depending on whether an obstacle blocks the ray. This type of shadow is referred to as hard shadow or umbra.

Solar Access has become a co-occurring issue for urban planning. Access to sunlight is an indispensable part of a healthy human thermal comfort for inhabitable buildings. Also, direct sunlight is of vital importance in architecture for reasons of mental health as well as to reduce electric lighting and energy saving. Therefore, urban environmental spatial analysis (Biljecki et al., 2015) often requires estimating whether a given point is in shadow or not, given a representation of spatial obstructions such as buildings on a specific date and time precisely associated with the solar position. Many planning authorities now need light concerns to be addressed as part of the outlining. Shadow Studies explain the influence of development concerning sun and daylight access to the neighboring context including surrounding buildings.

The combination of ray casting (Appel, 1968), the process of determining the visibility between two points and ray tracing (Whitted, 1980), the process of following illumination paths by performing repeated ray casts is the most frequently used method for shadow calculation in off-line rendering. For decades, the main drawback of this approach is the speed of the algorithm. Precisely, it is incredibly time-consuming to find the intersection between rays and geometry.

In this paper, unlike any traditional ray tracing algorithm, we propose a 2.5-D shadow construction algorithm for buildings with effect to sun's position for real-time outdoor rendering. A trigonometric shadow calculation algorithm is proposed on top of a quick method of obtaining solar angular values. The entire proof of the concept is developed in Python using Tensorflow (Abadi et al., 2015), an open source software library developed by the Google Brain Team using data flow graphs and the tensor data structure.

The paper is organized as follow; Section 2 contains the preliminary concepts required to understand the algorithm proposed in Section 3. Section 4, describes the proof of concept by running it on a suitable use case, the performance of the prototype is also discussed in the same section. Finally, the paper concludes in Section 5 by describing the main features and limitations of the present work.

2 BACKGROUND

This section provides a few necessary concepts and describes the useful notations before we the actual shadow calculation algorithm.

Bhattacharya, S., Braun, C. and Leopold, U.
A Novel 2.5D Shadow Calculation Algorithm for Urban Environment.
DOI: 10.5220/0007748602740281
In *Proceedings of the 5th International Conference on Geographical Information Systems Theory, Applications and Management (GISTAM 2019)*, pages 274-281
ISBN: 978-989-758-371-1

2.1 Sun Orientation

Solar orientation is the process of aiming something at the Sun. Any solar technology like a solar panel or a sun oven receives the highest amount of energy when pointed or oriented at a right 90° angle towards the sun. Now we need to know how actually to measure the Sun's position in the sky. Let's assume we are standing exact center of the Skydome, and the Sun is somewhere at the inside surface of the dome. The sun's position on the Skydome is a combination of two measurements as shown in Figure 1.

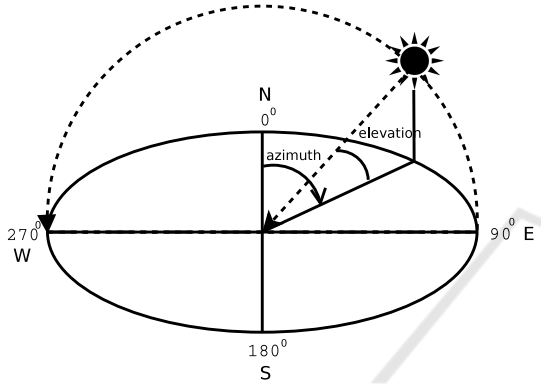

Figure 1: The Sun Position Diagram.

The first measurement is the Sun's direction on a compass. A plumb line drawn from the sun to the horizon intersects a specific degree on compass rows and this angle is a measure of the Sun's azimuth. We also need to know Sun's altitude or vertical angle, its angular height above the horizon. The combination of azimuth and elevation describes a specific spot on the Skydome.

Algorithm 1 computes the sun position from sunrise to sunset of a specific latitude(ϕ) for a given date(d) and a given interval of time in minute(τ). We need to know two things before we calculate the Sun position, the geographical location of the observer, and the orientation of the Sun at that position. To determine these two positions two sets of angles have to be specified. The first set, containing two angles, pinpoints the location on earth. They are a) latitude and b) longitude(θ). The second set of angles is related to the position of the Sun concerning a particular location on the earth. These angles are 1) declination angle, 2) elevation/altitude angle(α), 3) zenith angle(β), 4) azimuth angle(γ), and 5) hour angle(ω). The Solar declination angle is the angle between the equatorial plane of the Earth and the rotational plane of the Earth around the Sun explained in line 4. Δ in line 3 is the day angle and n is the day number which varies from 1 on January 1st to 365 (366) December 31. The hour angle is the angular displacement of the Sun east or west of the local meridian due to the rotation of the Earth around its axis 15° per hour; morning negative, afternoon positive and during solar noon zero. The hour angle for sunrise (ω_{sr}) and sunset (ω_{ss}) is explained in line 5 and line 6, respectively. Line 7 converts the time interval τ to angle in radian τ_ω. The loop at line 8 calculates the hour angles, I_ω from sunrise to sunset. Line 12 calculates the elevation angle at hour angle ω_i. The solar azimuth angle calculated through line 13 to 17 is the angular distance of the Sun's projection on the horizontal plane at a place on earth from a reference direction. Different authors use different conventions for calculating solar azimuth angle. In this paper, the azimuth angle is measured from the north clockwise from zero to 360°.

Algorithm 1: Sun Position Calculation from Sunrise to Sunset.

1: **procedure** SUNPOS(ϕ, τ, d)
2: n = dayofyear(d)
3: $\Delta = (2.0 \times \pi \times n)/365.25$
4: $\delta = \sin^{-1}(0.3978 \times \sin(\Delta - 1.4 + 0.0355 \times sin(\Delta - 0.0489)))$
5: $\omega_{ss} = \cos^{-1}(-\tan(\phi) \times \tan(\delta))$
6: $\omega_{sr} = -\omega_{ss}$
7: $\tau_\omega = \tau \times 0.261799$
8: **for** ($i = \omega_{sr}$; $\omega_{ss} \leq \omega_{sr}$; $i = i + \tau_\omega$) **do**
9: $I_\omega = i$
10: **end for**
11: **for** each $\omega_i \in I_\omega$ **do**
12: $\alpha[i] = \sin^{-1}(\sin\delta \times \sin\phi + \cos\delta \times \cos\omega_i \times \cos\phi)$
13: $\gamma' = \cos^{-1}\left((\sin\delta \times \cos\phi - \cos\delta \times \cos\omega_i \times \cos\phi)/cos\alpha\right)$
14: **if** ($i_\omega \geq 0$) **then**
15: $\gamma' = 360° - \gamma'$
16: **end if**
17: $\gamma[i] = \gamma'$
18: i++
19: **end for**
20: **Return** α, γ
21: **end procedure**

2.2 Shadow Geometry

The foundation of all science and technology is mathematics, and one of the most important beaches of mathematics is trigonometry. This section defines a set of trigonometric equations for shadow length calculation. These equations can be generalized to get the shadow top co-ordinate.

In Figure 3, $\triangle$OPQ form a right angle triangle considering the fencepost OQ of height, h drove perpen-

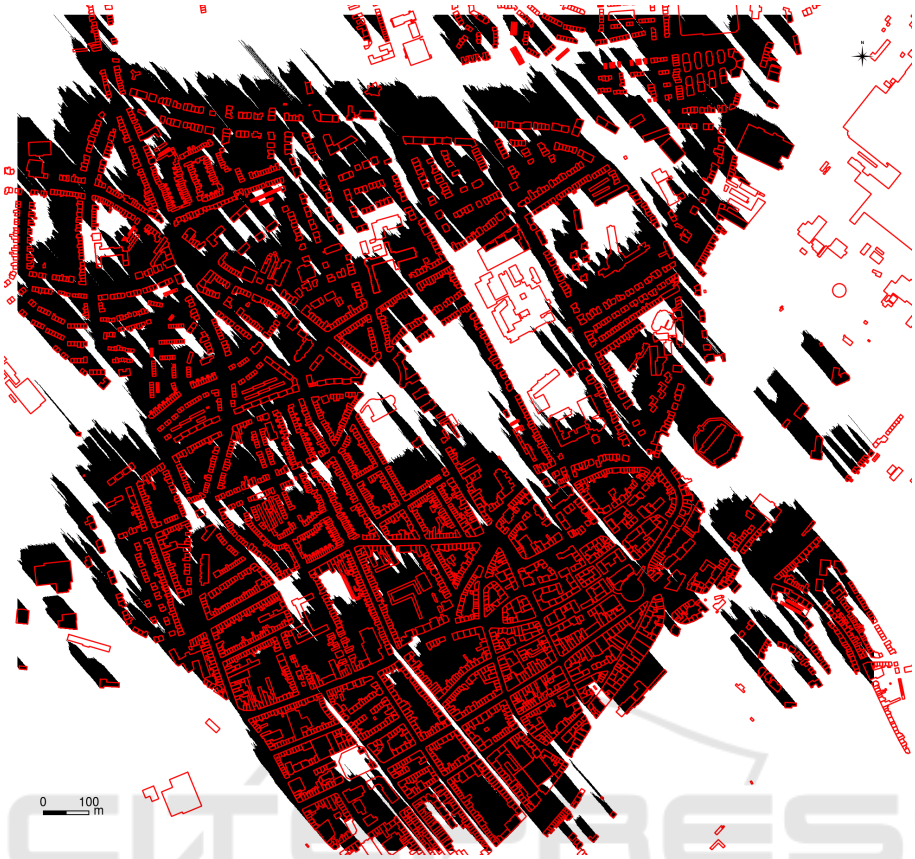

Figure 2: Date: 22nd December Time: 08:30.

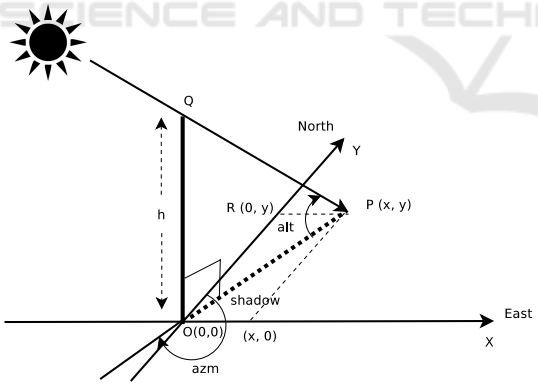

Figure 3: Shadow Calculation.

dicularly into the ground. Now the question is where would the shadow fall? However, its very easy to figure it out by drawing a line from sun across the top of the fencepost all the way to the ground. In this case, the length of the shadow $|OP|$ can be calculated using the acute angle between the horizon and the line to the sun, $alt(\angle OPQ)$ in Equation 1. The angle alt is commonly known as elevation angle or altitude angle.

$$\tan(alt) = \frac{h}{|OP|} \Rightarrow |OP| = \frac{h}{\tan(alt)} \text{ meter.} \quad (1)$$

The goal is to calculate the tip of the shadow located at $P(x, y)$, i.e. x meter East and y meter North of the the base of the fencepost. Using right triangle trigonometry on $\triangle ORP$, we get,

$$\sin(azm - 180^\circ) = \frac{x}{|OP|} \Rightarrow$$

$$x = |OP| \times \sin(azm - 180^\circ) \text{ meter.} \quad (2)$$

azm is a horizontal angle measured clockwise from a north base line, commonly known as solar azimuth angle. Substituting $|OP|$ from Equation 1 to Equation 2.2 yields x in Equation 3.

$$x = \frac{h}{\tan(alt)} \times \sin(azm - 180^\circ)$$

$$= -\frac{h \sin(azm)}{\tan(alt)} \text{ meter} \quad (3)$$

The same way, y can be calculated by Equation 2.2 and Equation 5.

$$\cos(azm - 180^\circ) = \frac{y}{|OP|} \Rightarrow$$

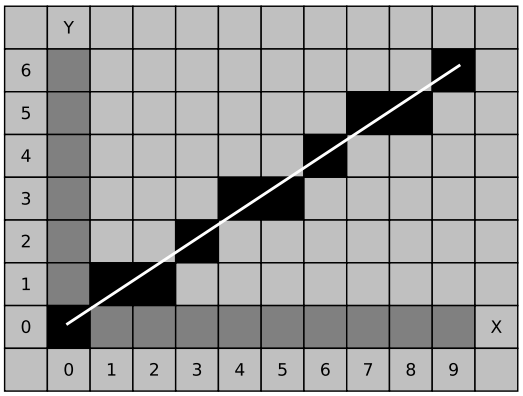

Figure 4: True line and its approximation.

$$y = |OP| \times \cos(azm - 180^\circ)\,\text{meter} \quad (4)$$

$$y = \frac{h}{\tan(alt)} \times \cos(azm - 180^\circ) = -\frac{h\cos(azm)}{\tan(alt)}\,\text{meter} \quad (5)$$

Therefore, from Equation 3 and Equation 5 the shadow tip coordinate can be observed at

$$P = \left(-\frac{h\sin(azm)}{\tan(alt)}, -\frac{h\cos(azm)}{\tan(alt)}\right)$$

2.3 Bresenham's Line Drawing Algorithm

Bresenham's line algorithm named after the inventor Jack Elton Bresenham (Bresenham, 1965) is a fundamental line drawing algorithm in computer graphics. Its primary use it to draw lines on raster graphics devices. The line-drawing algorithm is based on drawing an approximation of the real line, which is difficult to be drawn due to low precision caused by pixel spacing on a PC monitor especially when dealing with low resolutions.

In Figure 4, the true line is indicated in white straight line starting from `(0,0)` to `(9,6)`, and its approximation is indicated in black cells along the true line. Bresenham's line-drawing algorithm works in the following ways; first, it decides which axis is the major axis and which is the minor axis based on the length. `X` axis is the major axis in Figure 4. Starting from the original position, in each iteration, the current value of the major axis is incremented by exactly one cell (pixel). Then it decides the most appropriate pixel on the minor axis for the current pixel of the major axis by checking which pixel's center is closer to the true line. Bresenham's algorithm for approximate line drawing is all about this decision making. It is not 100% precise but works effectively for higher resolution.

3 METHODOLOGY

The 2.5D shadow of the buildings are generated by executing `SHADOW2D`, explained in Algorithm 2 and the entire process is illustrated using a block-diagram in Figure 5. In line 2, the building heights are obtained by subtracting the base height of the building from the DSM. `BLDHEIGHT` performs this task as illustrated in Algorithm 3. The DSM data is derived from a LiDAR survey which gives directly the surface model in a desired resolution. To obtain the building base heights we used a tool called `3Dfier`[1] (Biljecki, 2017) which was developed by Delft University. It takes building footprints and a LiDAR point cloud to extrude the footprints to a LOD1 model in which buildings are represented by block models (usually extruded footprints). The base height of the resulting building cube is based on descriptive statistics like the height values of the 10% quantile of the points falling inside the footprint. With available base heights linked to each building footprint it is then easy to compute the difference between the top of the DSM and bottom of each building to get the building height. `sunpos_t` gets solar orientation by means of azimuth and altitude angle for a given date and time for each and every point in the give building footprint.

`sunpos_t` calls `SUNPOS`, described in Section 2.1 with two additional parameter time and building footprint in line 3. The `coordinates` function generate the Cartesian co-ordinates for each and every point in the given building footprint. From north (N), south (S), east (E) and west (W) values first, the four coordinate corners, `(W,S); (E,S); (E,N); (W,N)` and hence the coordinate of each and every point in the building footprint is computed, respectively in line 4. From line 5 to 8 the algorithm is computing the shadow tops ($x_\top$,$y_\top$) for each point (x,y) in building footprint flowing the trigonometric formulation derived in Section 2.2. Bresenham's line drawing algorithm explained in Section 2.3 is used to connect each (x,y), ($x_\top$,$y_\top$) pair in the grid to draw the shadow line.

4 CASE STUDY

A proof of concept is developed for the validation of technical feasibility of the proposed algorithm at a very high spatial resolution by using scalable tensor data structure and inherent parallelism offered by data-flow based implementation in Tensorflow.

We run our proof-of-concept on Esch-sur-Alzette (49°29′44.988″N, 5°58′50.016″E), located 5,505.41

[1] https://github.com/tudelft3d/3dfier

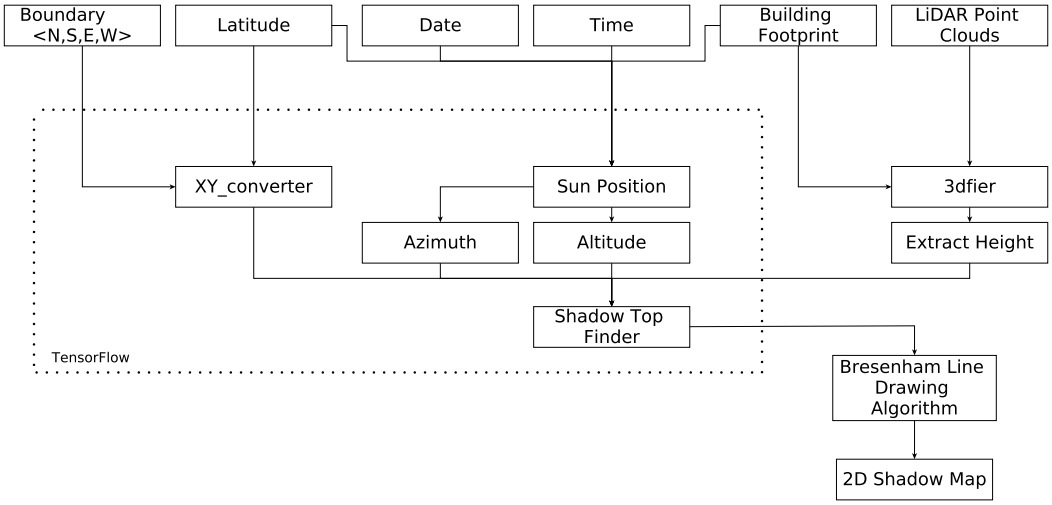

Figure 5: Block Diagram of Shadow Calculation Approach.

Figure 6: Date: 21st June December Time: 11:30.

km north of the equator, in the northern hemisphere [2]. It is situated in south-western Luxembourg on the border with France, the second largest town after the country's capital Luxembourg city. The total area of 14.35 km^2 with elevation above sea level ranging from 279m to 426m. In this paper, we used a digital elevation model of Esch-sur-Alzette. The grid is 1874 columns by 1828 rows, with a spatial resolution of 1 meter. The sun orientations from sunrise to sunset on

[2]https://fr.distance.to/Esch-sur-Alzette

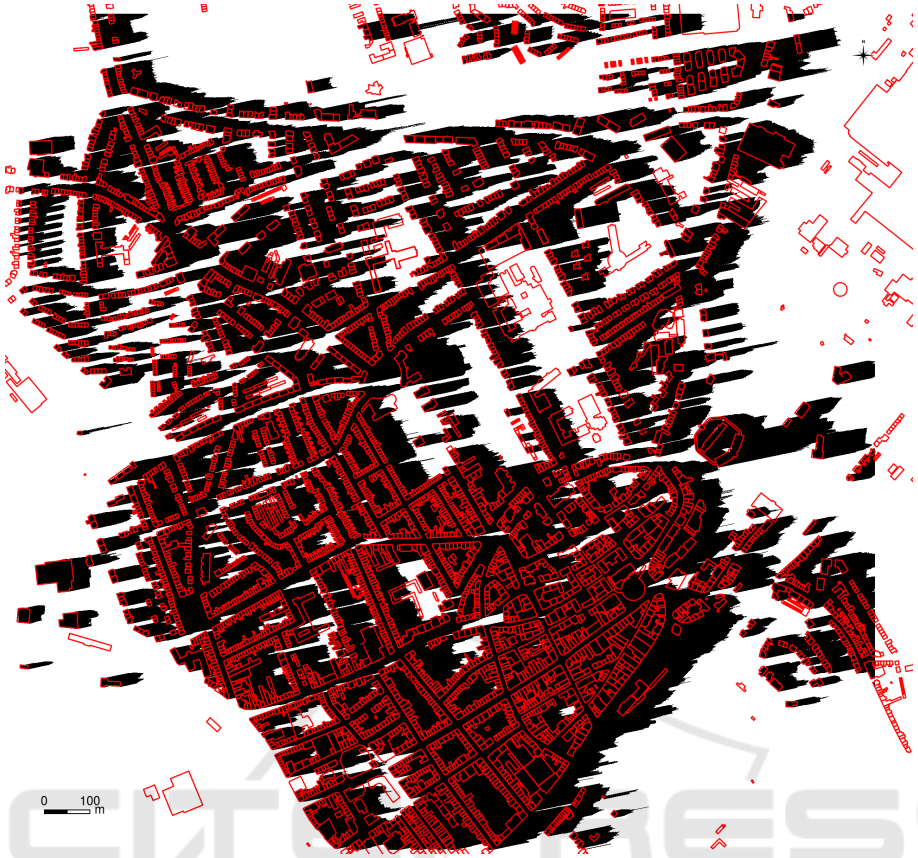

Figure 7: Date: 23rd September Time: 16:30.

Algorithm 2: 2.5D shadow calculation of buildings.

procedure SHADOW2.5D($\langle$N,S,E,W$\rangle$, ϕ, d, t, $\text{Bld}_{\text{footpnt}}$, $\text{LiDAR}_{\text{ptcld}}$)
  H = BLDHEIGHT($\text{Bld}_{\text{footpnt}}$, $\text{LiDAR}_{\text{ptcld}}$)
  azm, alt = sunpos_t(SUNPOS(ϕ, τ, d), t, $\text{Bld}_{\text{footpnt}}$)
  XY = coordinates($\langle$N,S,E,W$\rangle$, $\text{Bld}_{\text{footpnt}}$)
  for each $\langle x,y \rangle \in XY$ **do**
    $x_\top = -\frac{H[x,y] * \sin(azm)}{\tan(alt)}$
    $y_\top = -\frac{H[x,y] * \cos(azm)}{\tan(alt)}$
  end for
  for each $\langle x,y \rangle \in XY$ & each $\langle x_\top, y_\top \rangle \in XY_\top$ **do**
    $\text{print}_{\text{pixel}}$(bresenham(x,y,$x_\top$,$y_\top$))
  end for
end procedure

Algorithm 3: Building height calculation.

procedure BLDHEIGHT($\text{Bld}_{\text{footpnt}}$, $\text{LiDAR}_{\text{ptcld}}$)
  Bld_{LOD1} = 3dfier($\text{Bld}_{\text{footpnt}}$, $\text{LiDAR}_{\text{ptcld}}$)
  Bld_{ht} = $\text{extract}_{\text{ht}}$($\text{Bld}_{\text{LOD1}}$)
  Return Bld_{ht}
end procedure

21st June, 23rd September, 22nd March and 22nd December are shown in Figure 9. SUNPOS (Algorithm 1) is used to generate the result. We plotted Figures of cast shadow simulations for some exemplary sun positions over the year, namely, 23rd September at 08:30 (Figure 2), 21st June (Figure 6) at 11:30, 22nd December at 16:30 (Figure 7). The Figures showing building footprints of Esch-sur-Alzette in red outlines and areas in black are covered by cast shadows of the buildings. The direction and length of the cast shadows are determined by the position of the sun (Figure 2.1). The town gets longer cast shadows when the sun position is low over the horizon in the morning (Figure 2) and evening (Figure 7) hours, and short and more crisp shadows (Figure 6) when the sun reaches zenith about mid day. One can clearly notice the difference of sun position and resulting cast shadow length and the influence to neighbouring buildings or areas. Some unfavorable locations can end up receiving much less direct sunlight during the course over the year due to their relative position to other objects.

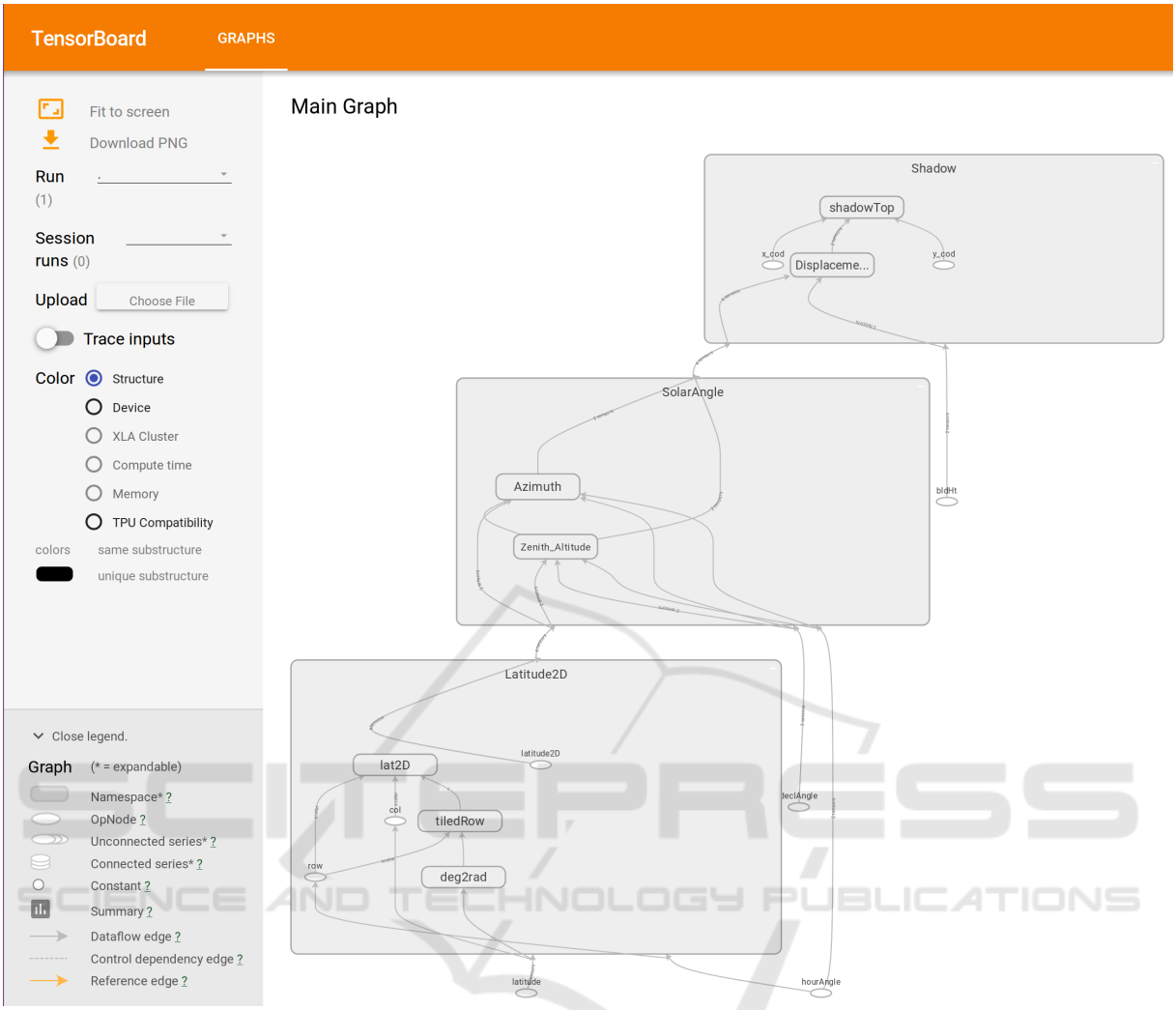

Figure 8: Dataflow graph generated by TensorBoard.

4.1 Performance

The proof of concept run on a 64 bit machine with `Intel(R) Core(TM) i7-6800K CPU @ 3.40GHz` processor. The performance is quite significant. Figure 2, Figure 6 and Figure 7 took `42.26 secs`, `29.63 secs` and `113.20 secs`, respectively. The performance is remarkably faster than `r.sunmask` module by Geographic Resources Analysis Support System, commonly referred to as GRASS (GRASS Development Team, 2017). `r.sunmask` Calculates cast shadow areas from sun position and DSM if the sun position is specified or based on given date/time calculates the sun position by itself.

The underlying data-flow graph of the implementation of the algorithm is shown in Figure 8 using Tensorboard. Tensorboard is the interface included with any standard TensorFlow installation used to visualize the data-flow graph and other tools to understand, debug, and optimize the model.

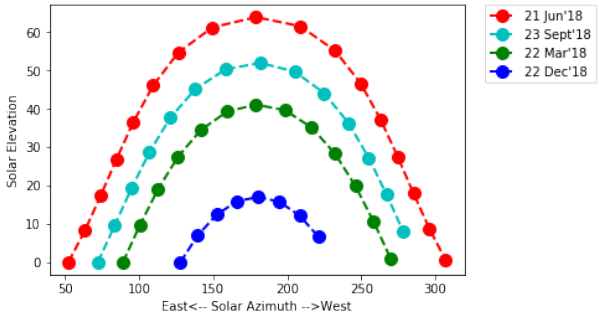

Figure 9: Sun path for Esch-sur-Alzette Calculated using Algorithm 1.

5 CONCLUSIONS

The main characteristics of the proposed 2.5D shadow implementation can be described as follows,

- Defining, optimizing, and efficiently calculating mathematical expressions involving multi-dimensional arrays (tensors).
- Transparent use of GPU computing. One can write the same code and run it either on CPUs or GPUs.
- Implicit parallelism and distributed execution.
- High scalability of computation across machines with huge data sets.

The building heights are valid for flat roofs (L0D1 model). This flatness of the terrain affects the accuracy of the cast shadows. In our future work, we are considering digital elevation model to get the actual hight of the object, towards a more accurate and robust shadow map.

ACKNOWLEDGEMENTS

This work has been funded and supported by the ENOVOS Foundation Luxembourg and the Luxembourg Institute of Science and Technology (LIST) through the SECURE project.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Appel, A. (1968). Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), pages 37–45, New York, NY, USA. ACM.

Biljecki, F. (2017). *Level of detail in 3D city models*. PhD thesis, Delft University of Technology, Delft, the Netherlands.

Biljecki, F., Stoter, J. E., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3d city models: State of the art review. *ISPRS Int. J. Geo-Information*, 4(4):2842–2889.

Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Syst. J.*, 4(1):25–30.

GRASS Development Team (2017). *Geographic Resources Analysis Support System (GRASS GIS) Software, Version 7.2*. Open Source Geospatial Foundation.

Whitted, T. (1980). An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349.