ON ACCELERATION OF THERMAL SIMULATION OF URBAN SCENES WITH THE APPLICATION OF AN EVOLUTIONARY ALGORITHM TO TREE PLANTING STRATEGIES

Dimitri Bulatov, Marko Hecht, Benedikt Kottler, Jonas Mispelhorn, Eva Strauss

Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB),
Gutleuthausstrasse 1, 76275 Ettlingen, Germany

KEY WORDS: Digital twin, Evolutionary algorithm, GPU, Urban morphology, Vegetation.

ABSTRACT:

Tree planting is one of the most popular in urban morphology measures for urban heat island reduction since, at a relatively low monetary cost and with a marginal alteration of the scene, trees provide shadows and neutralize harmful gases. Findings for their optimal distribution within the urban scene exist in numerous environmental studies. However, merely the digital twin of the scene possesses the capability to analyze further developments of the scene, such as changes in dominant wind directions. Today's extensive computational resources allow for thermal simulations of digital twins on multiple (not-yet-existing) urban scene designs, aiming at minimizing the average or peak temperatures. From the point of view of computer graphics, this paper proposes four tools to accelerate an evolutionary algorithm for tree planting strategies. Using GPU arrays for rendering, pre-rendering default scenes, and pre-filtering trees in the early morning and late evening hours helps accelerating the rendering process. Computation of fitness function on different computers allows a further acceleration of the evolutionary algorithm. The total acceleration factor of a scene using computational set-up exceeds 218, thus demonstrating the enormous potential the evolutionary algorithm may bring about in future investigations.

1. INTRODUCTION

Due to numerous reasons, such as the abundance of dark paved surfaces or lack of greenery and water bodies, the air temperature in densely built areas is generally higher than in the surrounding countryside. According to (Osmond and Sharifi, 2017), the temperature difference can reach up to 12°C. This phenomenon is known as the "Urban Heat Island" or UHI effect. This effect creates many negative repercussions (Williams et al., 2012, Kjellstrom et al., 2009) and should be mitigated while planning and designing a city. Commonly used measures for improving scene designs with respect to UHI include, but are not limited to: sustainable roofing and paving materials, drafting of parks with trees and water bodies, smart building design and solutions for transportation, and raising the awareness of the population to the problem. In particular, tree planting has proved to be an effective measure for preventing UHIs, since trees act as shade-provider and effective heat-storage systems, and they grant the city an endearing, livable visual aspect (Wong and Yu, 2005).

Since the tree planting process is always associated with some costs, the positions and kinds of trees to be planted require thorough planning. Moreover, since a city cannot be built from scratch, many researchers have worked on modeling urban temperatures using sensor and cadaster data (Helmholz et al., 2021). Nowadays, urban planners often have to perform multiple year-long simulations to evaluate and optimize the competing design (new urban area) or retrofit (existing urban area) alternatives. From the points of view of automatic algorithms, such as genetic or evolutionary algorithms, methods based on reinforcement learning, and many others, the machine is supposed to assess tens of thousands of scene designs, requiring acceleration in the evaluation.

In this paper, we aim to outline the scenario of tree planning measure assessment within an evolutionary algorithm (EA). For each scene design, the simulation is run, for example, during a diurnal cycle. A fitness value, such as the negative average of the temperatures, is assigned to the design. Scene designs yielding a higher fitness value are preferably kept in the next iteration of the algorithm, and so on.

We emphasize that in this article, we are merely interested in accelerating this evolutionary algorithm. The investigations on maximizing the fitness value and creating a new scene design are not the focus of this paper and will be covered in further work, in broad experiments. We present four improvements aimed at acceleration of the fitness value computation. Firstly, using the GPU instead of the CPU already leads to a considerable acceleration of the occlusion analysis module. Secondly, considering that only a marginal update of the scene is required, we pre-render the default scene and only update the occlusion map by newly rendered trees. Thirdly, we can take a preliminary decision about the occluded trees. For the last, fourth module on acceleration, the loop for the computation of fitness is addressed whereby the availability of several computers with GPUs to compute fitness in a controller-and-worker-like approach is assumed.

The article will be organized in the following way. We provide a concise summary of the previous works and all the reader is supposed to know about the digital twins, thermals simulations, and EAs in Sections 2 and 3, respectively. The methodology will be presented in Section 4. Results and conclusions are given in Sections 5 and 6, respectively.

2. RELATED WORK

Occlusion analysis is a very simple algorithm. One triangle after another is projected into the scene. The current depth is calculated from its baricentric coordinates. After this, for all image pixels, for which the current depth lies below the initial depth, both the foreground index (or feature) map and depth map are overwritten by the relevant values of the current triangle. The initial values of the depth map and index map are set to the positive infinity. Starting at this, numerous variations have been proposed. For example, triangles may be reordered according to the distance of their center of gravity to the image plane. The transparent ones are usually rendered from background to foreground, and the opaque ones from foreground to background. In order not to search huge images for pixels exceeding the value of the initial depth map, constraining the search range is practice. One can reverse the procedure and emit the rays from image pixels searching for intersection with objects within a scene, giving origins to ray tracer and path-tracer algorithms (Feldmann, 2015), implemented on modern GPUs. To minimize the number of intersections, ray tracers employ spatial indexes to organize the scene to be rendered (Hapala et al., 2011). The problem of dealing with large models is covered in (Landaverde et al., 2014), where an efficient way of storing the GPU memory resources in the RAM is presented.

With respect to the calculation of sun radiation in thermal simulation, a very interesting approach was presented by (Guo et al., 2018). The authors have subdivided the half-sphere into patches and for the sun position in each patch, the scene was pre-rendered and stored. During simulation at an arbitrary time, only the position of the sun must be computed, and the corresponding foreground index map is loaded. The problem with this approach is that in our application, many scene designs must be evaluated which even degrades the problem of memory load. The work of (Jones et al., 2012) focuses on the computation of solar radiation during longer cycles. For a fixed area, the sunlit pixels are pre-computed on every 20th day while for intermediate times, the B-spline is calculated to model the sunlit areas as continuous functions of time. The modern view synthesis approaches try to compute the intermediate views using machine learning, in particular, generative-adversarial techniques; thus, the results are realistically looking fantasy products (Srinivasan et al., 2019).

Finally, many authors tried to accelerate EAs. While (Cantu-Paz, 2000) discussed a master-slave implementation of EAs, articles like (Wong, 2009, Arora et al., 2010) focused on efficient exploitation of the available GPU. In particular, in (Arora et al., 2010), the authors have not only parallel evaluation of solutions, but also adjusted the remaining modules of EA to perform efficiently on the GPU architecture. For more sources on efficient GPU-based implementation of EAs, we refer to (Cheng and Gen, 2019), whereby we are more interested in algorithmic than in hardware-relevant improvements and will therefore pursue a controller-worker solution in Section 4.4.

3. PRELIMINARIES

3.1 Notation

Throughout this work, we will use the following notation:

- A denotes the default mesh;
- B denotes the new meshes (trees);
- There are J trees per solution (so we can refer to meshes B_j for j-th tree);
- i denotes the foreground index map;
- d is the foreground depth map;
- The total number of tree positions is U;
- The EA has G generations;
- Every generation has N solutions, or individuals, so we will occasionally refer to $B_{jn(g)}$;
- Finally, we parallelize the processing using K computers

3.2 Digital twin computation

As in (Bulatov et al., 2020), a semantic mesh may be extracted from the airborne sensor data, optionally, we some additional input, such as GIS building outlines. Together with mathematical (normal vectors) and physical (density, emissivity, and other relevant for simulation) properties of its triangles, we have a parametric mesh A, which is denoted as a three-dimensional digital twin. At the beginning of the simulation, the weather information is retrieved (e.g. from a weather server), thus creating a four-dimensional digital twin. The scene A consists of the ground model, building polygons, and trees. The trees are given in a twofold manner. For large groups of trees and even forests, there is not much sense to model every single tree; hence, we use a forest box, a prismatic structure whose base is described by an irregular polygon. On the contrary, single trees are modeled using a closed tree-like surface consisting of approximately 150 triangles. This is also the case for the virtual trees to be planted within EA. Since we have the landcover classification result, which we compute using one of the state of the art procedures, (Volpi and Tuia, 2016, Bulatov et al., 2019), we know where the trees can or cannot be planted in our semantic mesh A. Tree planting is only permitted upon the classes soil or grass of the ground model.

3.3 Evolutionary algorithm in a nutshell

We evaluate N scene designs within one generation of the EA. Every design differs from the default scene by J trees, added at random, but meaningful positions. Then we can assume that the input for the first generation is a $J \times N$ matrix of random integer numbers, all of which are between 1 and U. From this matrix, the corresponding meshes B_{jn}, with $j \in \{1, \ldots, J\}$ referring to the individual tree index and $n \in \{1, \ldots, N\}$ denoting the running index for scene design, are formed and added to A.

The thermal simulation algorithm developed in (Kottler et al., 2019, Bulatov et al., 2020) allows computing the unknown temperature using the heat balance equation for each discrete time, each triangle, and each scene design. Since we want to keep the temperatures low, the *fitness value* is a negative norm of the temperature array over time and over the triangles. After evaluating N scene designs, we obtain N fitness values. The second generation of the EA begins, which presupposes the *evolution* of the best solutions, denoted as parents. The two most common tools for evolution are *crossovers* and *mutations*. Crossover means taking“the chromosomes from two or more parents”, which in our case means random combining integer numbers corresponding to tree positions from the best solutions received so far. Mutation, or “arbitrary chromosomes alteration”, means that in a given solution, some few integers are arbitrarily replaced by the others from the pool $1, \ldots, U$. The two steps, fitness value computation and evolution, are carried out for G iterations.

4. METHODOLOGY

4.1 Occlusion computation on the GPU

The occlusion is the negation of the exposure to the light source and has to be calculated for each triangle. The equation for the heat radiation from the sun in dependence of cloud coverage index I_c, surface albedo a, solar irradiation E_{Sun}, and elevation angle of the sun θ,

$$S = (1 - a)I_c E_{Sun} \cos(\theta)\gamma_{Sun}, \tag{1}$$

allows for the calculation of the thermal energy S emitted from the sun towards the simulation scene. The factor γ_{Sun} is an array of booleans values that determines the occlusion of a triangle face. The value is 1 if the face is exposed to the sun and 0 if it is concealed. The array contains the same number of values as triangles in the mesh that define the scene, with each value of γ_{Sun} indicating the exposure for a triangle.

In order to accelerate the computation for the occlusion or exposure of the scene triangles, it is executed via OpenGL on a graphics processing unit. There are two outputs of the OpenGL executable, which is a depth mask d and an index mask i. Both masks are provided in a 2048×2048 matrix. The depth mask is the distance of the center of each triangle from the perspective of the light source is rescaled from -1 to 0, with lower values being closer to the light source. The index mask codes the index of the triangle closest to the light source lodged at the position of the grid and allows to retrieve γ_{Sun} in (4.1).

Although it would be possible to compute the term in (4.1) entirely on the GPU, we prefer making the short detour over i and d because the simulation is supposed to be computed every minute. Since γ_{Sun} does not change too frequently for almost all triangles, the time-consuming process of occlusion analysis may only be performed every 10 minutes. The presence of highly time-dependent variables, such as I_c, does not allow using the term in (4.1) throughout 10 minutes.

4.2 Pre-rendering

The second acceleration of the occlusion computation roots in the separate handling of the default scene and the artificial trees. By pre-rendering the default scene and storing the depth mask d_A and the foreground index mask i_A of the default scene, this rendering process can be skipped for later simulations. Instead, only the occlusion for the artificial trees has to be determined, which also results in a depth mask d_B and an index mask i_B output.

To obtain the occlusion map for the entire scene, d_A and i_A for the same relative position of the sun have to be imported. It is important to notice that the spatial buffer (or 3D scene extensions) while rendering B must coincide with those for rendering A to avoid co-registration problems. In other words, if we imagine that at a certain moment of time, the highest object of the scene seen by the sun is one of the new trees, this object will be cut (unlike to rendering the scene $A \cup B$), leading to marginal inaccuracies.

After pairs of images of the size 2048×2048 are stored in a directory in regular time steps, at the moment of rendering a solution $B = B_{jn}$, we need to compare d_A and d_B to determine the combined occlusion map of the artificial trees.

The indices of the scene with a lower value of the d at the same position of the mask then conclude the index mask $i_{A\cup B}$ of the merged scene, as explained in (2).

$$i_{A\cup B} = \begin{cases} i_A & \text{if} \quad d_A < d_B \\ i_B & \text{otherwise.} \end{cases}, \tag{2}$$

After comparing the depth values and updating the index mask for the entire scene, the indices present in $i_{A\cup B}$ then indicate which triangles are exposed to the sun. Their values of γ_{Sun} are set to one.

Also, here one could wonder why we do not pre-render the temperatures of the default scene, which would require merely the computations the of triangles in B and those triangles of A which are occluded by them. There are two objections: Temporal shadows and the heat conduction term. Because of the temporal shadows, triangles formerly but not anymore occluded have a different temperature than the triangles that have not been occluded. The conduction term is responsible for the distribution of the (negative) heat to the neighboring triangles, even if those are not occluded by newly-added trees.

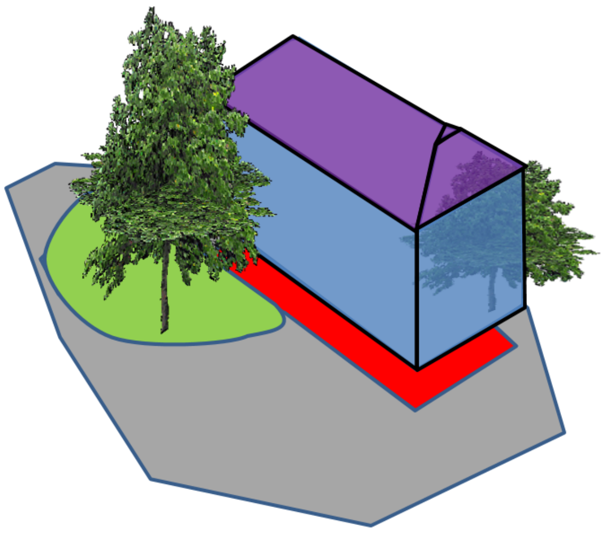

Figure 1. The advantages of pre-rendering and preliminary decisions for single tree rendering. In the case of the tree on the left, the default scene is not supposed to be rendered every time. The tree on the right is partly visible even though its highest point is occluded. Discarding this tree while pre-filtering would result in inaccuracies. However, as we will see in Results section, inaccuracies of this kind happen rarely.

4.3 Preliminary decision for single tree rendering

Even with our improvement using the pre-rendered default scene, we may still have several thousands of small triangles to render because trees are very detail-rich objects, which is especially problematic in the case of CPU-based rendering. At the same time, these are compact types of objects, located in different parts of the scene. Bearing in mind this as well as the fact that the position of the sun is over the scene, we can take a preliminary decision about the trees that may be occluded and further accelerate the algorithm at cost of accuracy.

Meshes from J trees are given an index number. Let $\mathbf{c}_j$ denote the highest center of gravity over all triangles in the mesh B_j. The depths of projections of $\mathbf{c}_j$ are compared to the values of A at corresponding rounded pixel positions. If a tree top $\mathbf{c}_j$ is visible, we can proceed with rendering B_j as described in Sec. 4.2. However, if $\mathbf{c}_j$ is occluded – by a building wall or a box forest, for example, – then we conclude that the tree itself is not visible, too. Thus, we ignore situations as in Fig. 1, resulting in slight inaccuracies. The loss of accuracy could be significantly reduced if, instead of the highest triangles, the bounding boxes were taken. However, this would take longer and, besides, we wish to exploit the fact that the sun shines from above.

Computation of projections of $\mathbf{c}_j$ takes place on the CPU. That is, the per-tree balance of speed compared to GPU would roughly be estimated as

$$s_0|B_j| - \text{overhead}, \tag{3}$$

where s_0 is the proportion of occluded trees and $|B_j|$ is the average number of triangles in a tree mesh. The term overhead emphasizes the necessity of numerous relabel routines: From the highest triangle index to the subset of j, from this subset to the whole set, and further to $A \cup B$. Thus, the quintessence of equation (3) is that from a certain percentage of occluded trees on, pre-filtering will allow saving computing time. Section 5 will help us to find this critical percentage.

4.4 Controller-worker subdivision

We now suppose availability of K computers possessing a mighty GPU. Since the computation of fitness values can be computed for different individuals independently within one generation, we can additionally accelerate the processing by distributing the simulation task between different GPUs in a simple controller and worker way. We are conscious that many approaches (Arora et al., 2010) exist that perform multi-threading even during parents' selection and evolution of EA. However, in our case, the by far most computationally intensive part of the procedure is the simulation of temperatures, needed for fitness value assignment to a scene design. We compute the fitness value on every machine and obtain values $t_1, t_2, \ldots t_K$ for computing times. Now, supposing the population size to be again N, we have to determine how many fitness computations N into $a_1, a_2, \ldots, a_K$ are run at each GPU to minimize the overall computing time. Overall, we have to determine

$$\min \max_l (n_k t_k) \text{ s.t. } \sum_k n_k = N. \tag{4}$$

There is a closed form solution for partition of N into K positive numbers satisfying (4):

$$n_k = N t_k^{-1} \left(\sum_k t_k^{-1} \right)^{-1}, \tag{5}$$

Of course, since n_k are integer numbers, we must carry out rounding, resulting in a constraint for one of the numbers. Another small issue to be aware of is a particularity of many EAs. Starting from the first generation, the first couple (N_0) of the best individuals from the previous generation remain unchanged to prevent the solution to become worse with increasing g. Thus, the simulation must not be carried out for these individuals since their fitness values can be kept. Because of this, (5) is valid only for the first generation, while from the second generation on, we slightly modify this equation by replacing N by N_0 and simply adding N_0 to n_1, thus copying the solutions. This means that if all GPUs are equally powerful, then the total time is Nt/K. However, in practice, data exchange between the computers as well as components related to the CPU workaround of the EA (see next paragraph) always needs some extra time, as Amdahl law and other similar estimations state.

The key element of our controller-worker approach is that for a given generation, the controller (master computer) generates the solution prototypes, after which these are redistributed between the master and the worker using a shared directory accessible by all computers. The workers carry out the computation of fitness from solutions (we remind that the solutions form a $N \times J$ matrix of integers while the fitness values are, in essence, N floating point numbers). The fitness values are again imported by the master. Parents' selection and offspring creation take place by the master only. Together with the data exchange, this is the main reason our theoretical estimation of the total time is optimistic for the practice.

5. RESULTS

5.1 Experimental setup

The default scene is, in essence, the digital twin of the dataset described in (Bulatov et al., 2020). It has been reconstructed using the remote sensed data (laser point cloud and multispectral imagery), close range data (aerial and drone imagery), and open street data (GIS outlines of buildings). The single trees, both in the default scene (mesh A) and in B_J are all equal prototypes scaled to the correct size in x, y, z direction. Each prototype consists of a hexagonal trunk and a crown, needed to simulate shadows. The crown mesh is sampled from 500 random vertices x, y in a unit circle, with z coordinates given by a higher degree parametric bivariate function $z(x, y)$, and edges stemming from the Delaunay triangulation in 2D. Doing so ensures consistent orientation of the normal vectors pointing outwards. Now, using standard tools for mesh decimation, we can keep the number of vertices flexible while the form of the tree crown is widely preserved. We usually keep 10 to 50% of initial vertices; the larger the percentage, the more noticeable the effect of pre-filtering trees.

5.2 Acceleration of occlusion analysis

The default scene A consists of 1.65 million triangles, and we add to A 2000 trees with around 110 triangles each. During a diurnal cycle, we measure the time needed to compute the solar-exposed triangles in one-hour intervals. This temporal resolution is insufficient to compute the temperatures accurately because the changes in occlusion status should better be recorded every ten minutes. However, for us, it is acceptable since we only want to measure the performance at arbitrary times of the day. At the same time, we will compare the lists for occluded triangles to measure the accuracy of our method.

Figure 2 shows the results of the accelerated occlusion analysis. Using GPU arrays, we can decrease the computation time by a factor of 21.3 at midday to 41.6 at the time of the minimum sun inclination angle (2.5°C). Here we must put into perspective the fact that during the CPU rendering, we have to perform inpainting using repeated dilation (since many triangles' projections consist of only one pixel), additionally slowing down the computation during morning and evening hours. Without this inpainting, the acceleration constitutes values between 22.2 and 24.2.

At all times of the day, pre-rendering helps to shorten the computing time considerably. The GPU-based processing accelerates to the factor around 3.8 and the CPU-based to the factor of 7.1. As for the preliminary decision for single tree rendering using the highest triangle, the positive impact could only be measured at the times of small inclination angles, as one could have expected. At these times, when 30 to 40% of B_j are not supposed to be rendered (blue curves), the speed-up using CPU is more visible than for GPU because the time for rendering a single tree is considerably higher. At high sun positions, when almost all trees are visible ($s_0 \approx 0$), the relabeling overhead (3) slows down the procedure, even though values of $|B_j|$ are high.

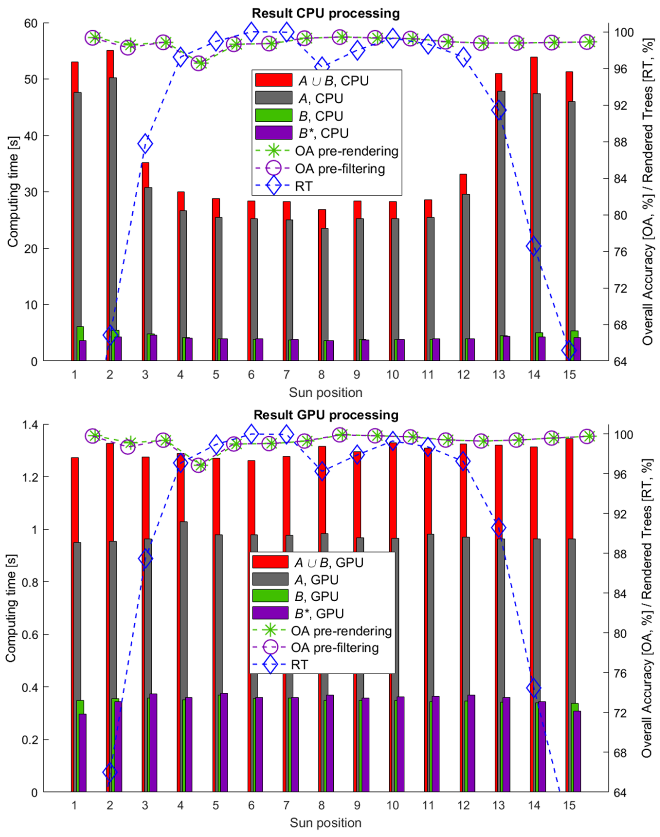

Figure 2. Performance of our occlusion analysis, CPU- and GPU-based processing on the top and the bottom graphics, respectively. In the x axis, the current sun position is denoted. In the left y-axis, bar diagrams denote time in seconds. On the right, we depict the percentages of remaining after pre-filtering $(1 - s_0) * 100\%$ with s_0 from (3) and values of overall accuracy.

Turning to the results for overall accuracy, we can see that the values are quite high and with one exception over 98%. The decay in overall accuracy values while pre-filtering is almost invisible. We could notice that the number of false negatives and false positives is approximately the same for most times, which could hint that the inaccuracies are due to some numerical issues and not systematical. However, for very few measurements, mostly at midday, the deviation reaches some 70%, for a yet unclear reason.

Figure 3 shows the fragments of rendering results. We remind that by using pre-rendering, some triangles from A may get cut out if they do not fit in the bounding box computed from the default scene. Obviously, this happens only at very early and late hours of the day. Around midday, when the sun is right above the scene, the extensions of A and $A \cup B$ widely coincide. Note also how narrow the trees appear in the scene rendered in the early morning. The reason is that the scene must be stretched along the z axis to fill completely a 2048×2048 image.

5.3 Acceleration of evolutionary algorithm

We consider a small excerpt of the previous scene containing some 10000 triangles and 10 trees with approximately 100 triangles each. To prove the principle of the EA, a toy example with $N = 6$ and $G = 6$ was run using two computers (CPU: Intel Core i9-10980HK - 16Core@2.40GHz; 32GB RAM; GPU: NVIDIA GeForce RTX 2080 Super) equipped with the up-to-date MATLAB versions. The temperature computation takes place for the diurnal cycle every 30 seconds while the pre-rendering version of the occlusion analysis algorithm runs every 10 minutes. Two best solutions were kept from the second generation on. Using merely one laptop, the program needed 1.25 hours for $NG - n_0(G - 1) = 26$ simulations. Using this same laptop with two MATLAB instances has resulted in 0.84 hours, or 32.8% acceleration, while the application of two laptops brought an improvement of 28% (0.9 hours). The difference between the running time on one or two computers is explainable by the necessity of storing some data and accessing it on a network directory. In a further experiment, we opened two MATLAB instances on each computer and run the program with one master and three workers, letting the running time to decrease to 0.62 h or roughly half of the original time. We see that the waiting time and the overhead calculation do not allow for the reduction in the running time by a factor proportional to the number of workers. This confirms Amdahl's law, but still, the improvement is quite significant.

To provide a qualitative illustration of the results, we show in Fig. 4 two screenshots of our model with two scene designs at two different times. We can see how even after the sunset, the surface temperature remains higher in the areas of the scene where no trees are planted while at the afternoon, treed provide welcomed shadows.

Summarizing, we multiply the acceleration factors yielded by all modules (28.95, 3.71, 1.09 and 2.00) respectively, to obtain the overall acceleration factor to be 218.

6. DISCUSSION AND OUTLOOK

We presented a workflow for an evolutionary algorithm for temperature evaluation on the 3D digital twin of an urban scene, which is a very up-to-date and, literally and metaphorically speaking, burning topic, as we could see in Sec. 1. In total, four tools for acceleration of the workflow were presented. Three of them affect the time-consuming procedure of occlusion analysis and one the EA itself. Two of them (GPU-based rendering and controller-worker evaluation of evolutionary algorithm) are hardware-based. The remaining two, pre-rendering and pre-filtering of trees have an algorithmic background. We could see that the application of GPU-based rendering leads to the highest acceleration, followed by tree pre-rendering, resulting in double- or high single-digit factors of speed-up. While in the first case, the efficient usage of graphic card allows rendering even very complex scenes, the occlusion computation of the artificial trees, the reading of the depth and index mask of the default scene, and the successive comparison are less time-expensive than the occlusion computation for the entire scene. Two remaining acceleration suggestions have the advantage to be quantitatively predictable. For example, the preliminary decision on single tree rendering can be activated at very early

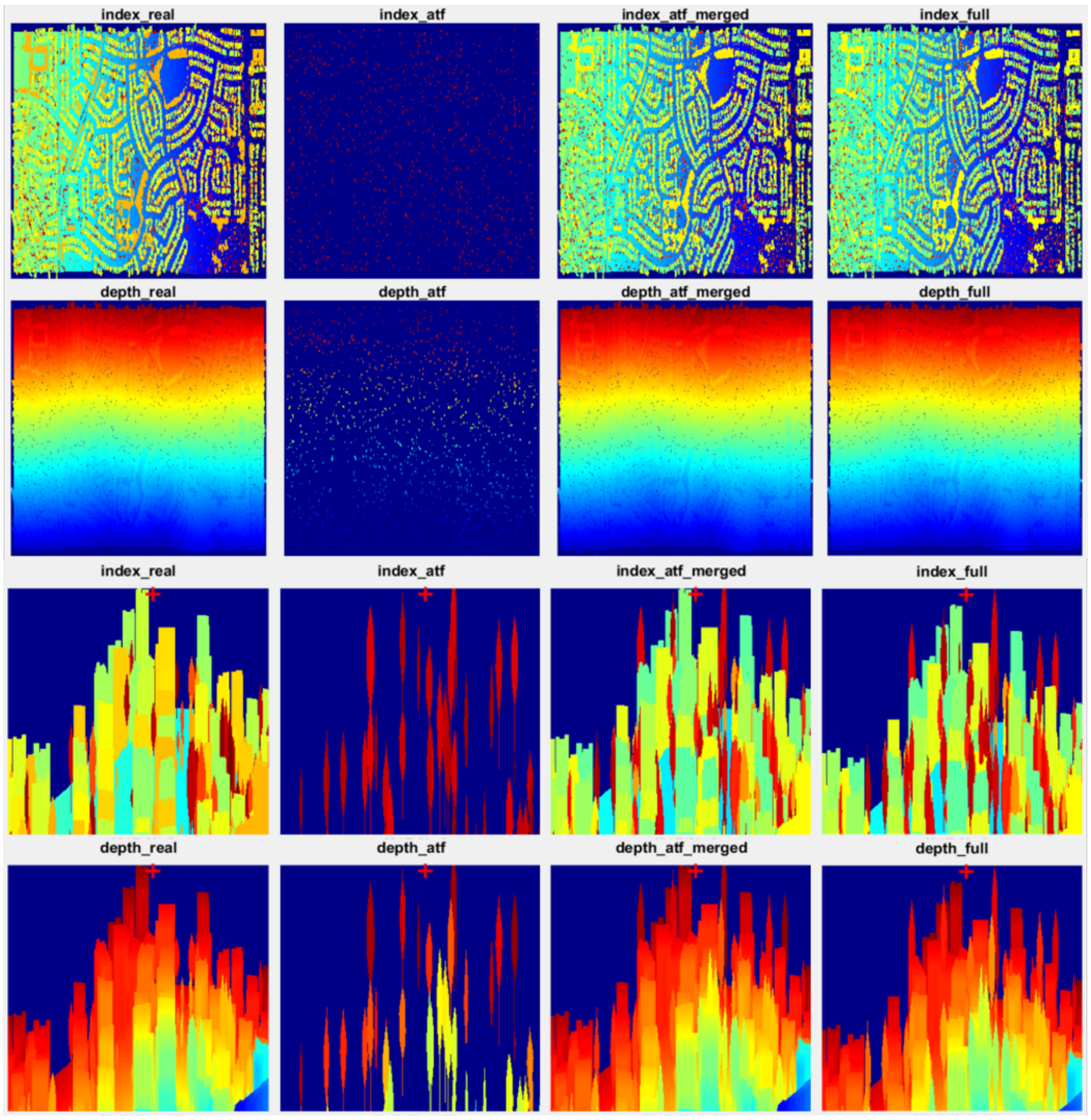

Figure 3. Depth and index mask images for various settings of the rendered scene. At the top: CPU-based, around the midday. At the bottom: GPU-based, at early hour. In each configuration, in the top row, from left to right: i_A, i_B and $i_{A \cup B}$ obtained with and without pre-rendering. Bottom row: corresponding foreground depth values. The lower the sun position (two bottom rows), the higher the deviation between the results of stratified and simultaneous rendering of $A \cup B$. For example, a tree at the top of the images is cut up, and the building corner close to the red cross, despite coinciding in the first three images, appears displaced in the right-most image.

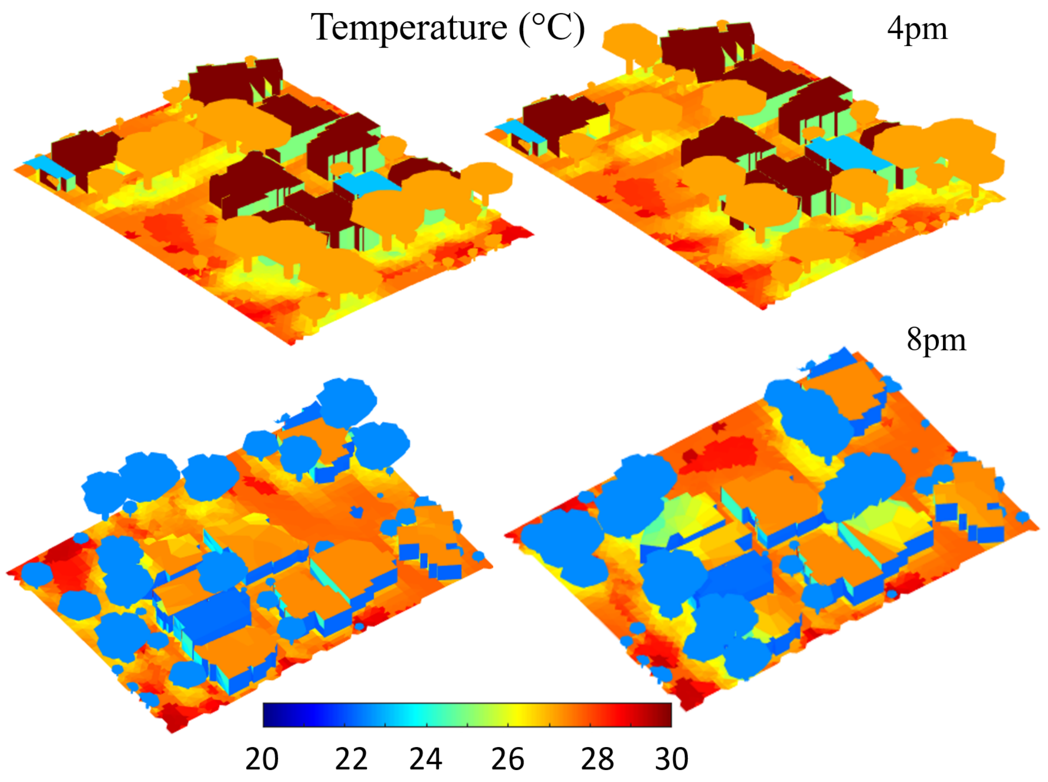

Figure 4. Scene views for two different tree planting designs at two different times.

and very late hours only. Optimal distribution of fitness values computations between different computers can be achieved using (5).

To get both speed and flexibility, the proposed algorithm uses both the CPU (flexibility) and the GPU (speed). Data transfer between the two devices is usually the biggest performance bottleneck in such setups, because we have to load the model every time in the GPU instead of merely modifying the angle between the sunrays and the surface. The reason for this suboptimal data processing is that due to the application of the EA, we have dozens of thousands of different models. However, taking into account that only a few tree models are dealt with, it should be investigated in the future to what extent tensors of transformations B_{jng} may help to work with a scene model loaded in the GPU only once or old, not used anymore meshes may be cleared in VRAM to make place for updated ones. Additionally, we are planning to outsource parts of the processing to the other devices to reduce the volume data transfers. Finally, the first experiments with the results of the evolutionary algorithm are being carried out, and the relevant findings will be discussed in the future.

ACKNOWLEDGEMENTS

We express our deep gratitude to Prof. Mikut (Karlsruhe Institute of Technology) for the fruitful discussions on evolutionary algorithms.

REFERENCES

Arora, R., Tulshyan, R., Deb, K., 2010. Parallelization of binary and real-coded genetic algorithms on GPU using CUDA. *IEEE Congress on Evolutionary Computation*, IEEE, 1–8.

Bulatov, D., Burkard, E., Ilehag, R., Kottler, B., Helmholz, P., 2020. From multi-sensor aerial data to thermal and infrared simulation of semantic 3D models: Towards identification of urban heat islands. *Infrared Physics & Technology*, 105, 103233.

Bulatov, D., Häufel, G., Lucks, L., Pohl, M., 2019. Land cover classification in combined elevation and optical images supported by OSM data, mixed-level features, and non-local optimization algorithms. *Photogrammetric Engineering & Remote Sensing*, 85(3), 179–195.

Cantu-Paz, E., 2000. *Efficient and accurate parallel genetic algorithms*. 1, Springer Science & Business Media.

Cheng, J. R., Gen, M., 2019. Accelerating genetic algorithms with GPU computing: A selective overview. *Computers & Industrial Engineering*, 128, 514–525.

Feldmann, D., 2015. Accelerated ray tracing using R-trees. *GRAPP*, 247–257.

Guo, S., Xiong, X., Liu, Z., Bai, X., Zhou, F., 2018. Infrared simulation of large-scale urban scene through LOD. *Optics express*, 26(18), 23980–24002.

Hapala, M., Davidovič, T., Wald, I., Havran, V., Slusallek, P., 2011. Efficient stack-less bvh traversal for ray tracing. *Proceedings of the 27th Spring Conference on Computer Graphics*, 7–12.

Helmholz, P., Bulatov, D., Kottler, B., Burton, P., Mancini, F., May, M., Strauß, E., Hecht, M., 2021. Quantifying the impact of urban infill on the urban heat island effect – a case study for an alternative medium density model. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46, 43–50.

Jones, N. L., Greenberg, D. P., Pratt, K. B., 2012. Fast computer graphics techniques for calculating direct solar radiation on complex building surfaces. *Journal of Building Performance Simulation*, 5(5), 300–312.

Kjellstrom, T., Holmer, I., Lemke, B., 2009. Workplace heat stress, health and productivity–an increasing challenge for low and middle-income countries during climate change. *Global Health Action*, 2(1), 2047.

Kottler, B., Burkard, E., Bulatov, D., Haraké, L., 2019. Physically-based thermal simulation of large scenes for infrared imaging. *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, INSTICC, 53–64.

Landaverde, R., Zhang, T., Coskun, A. K., Herbordt, M., 2014. An investigation of unified memory access performance in CUDA. *IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, 1–6.

Osmond, P., Sharifi, E., 2017. Guide to urban cooling strategies. *Low Carbon Living CRC*.

Srinivasan, P. P., Tucker, R., Barron, J. T., Ramamoorthi, R., Ng, R., Snavely, N., 2019. Pushing the boundaries of view extrapolation with multiplane images. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 175–184.

Volpi, M., Tuia, D., 2016. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2), 881–893.

Williams, S., Nitschke, M., Weinstein, P., Pisaniello, D. L., Parton, K. A., Bi, P., 2012. The impact of summer temperatures and heatwaves on mortality and morbidity in Perth, Australia 1994–2008. *Environment International*, 40, 33–38.

Wong, M. L., 2009. Parallel multi-objective evolutionary algorithms on graphics processing units. *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, 2515–2522.

Wong, N. H., Yu, C., 2005. Study of green areas and urban heat island in a tropical city. *Habitat International*, 29(3), 547–558.