G. Gori, A. Guardone
VirtuaSchlieren: a hybrid GPU/CPU-based Schlieren Simulator for Ideal and Non-Ideal Compressible-Fluid Flows
Applied Mathematics and Computation, Vol. 319, 2018, p. 647-661
doi:10.1016/j.amc.2017.07.041

VirtuaSchlieren: a Hybrid GPU/CPU-Based Schlieren Simulator for Ideal and Non-Ideal Compressible-Fluid Flows

Giulio Gori and Alberto Guardone

Department of Aerospace Science and Technology, Politecnico di Milano
via La Masa, 34, 20156 Milano, Italy

Abstract

A schlieren post-processing tool for CFD simulations of ideal and non-ideal compressible-fluid flows is presented. The software *VirtuaSchlieren* simulates light propagation across a non-homogeneous medium to predict the schlieren images from an actual measurement apparatus. Trajectories of a large number of light rays—in the order of tens of millions—are reconstructed by numerical integration, from the light source to the screen plane. To this purpose, kd-tree search algorithms are implemented to retrieve the position of each ray within the CFD computational grid at each time step. The local value of the refraction index was retrieved from the interpolated value of the density at the center of each cell. The simple Gladston-Dale and the Lorentz-Lorenz models are implemented to compute the value of the refractive index. Two search algorithms are evaluated, namely, the Approximate Nearest Neighbor (ANN) and a simplified kd-tree search technique. The latter is implemented on both CPU and GPU architectures. The hybrid GPU/CPU implementation was successfully tested against a reference experimental schlieren image of the supersonic flow around a conical body. Numerical simulations of the supersonic expansion of non-ideal flows are also presented.

Keywords: Numerical Schlieren, supersonic flows, non-ideal compressible-fluid dynamics, CUDA, GPU

1. Introduction

Current research activities in fluid mechanics integrate theoretical predictions, numerical simulations and experimental measurements to tackle physical problems of ever increasing complexity. Indeed, in laboratories worldwide computational fluid dynamics (CFD) simulations are often used to guide experimental activities and to complement the actual measurements. Unfortunately, the comparison of CFD against experimental results is not always a straightforward task. Measurement procedures may possibly deliver a representation of the flow field that is not as detailed as CFD ones and measured quantities are often not immediately related to flow variables such as density, pressure and velocity. In these cases, complex and/or time consuming algorithms are to be applied to CFD results for comparing simulations and experiments.

A good example is the post-processing of numerical simulations of fluid flows for comparison against measurements obtained from an experimental technique called Schlieren visualization.

Schlieren is a German word used to indicate optical non-homogeneities. In fluid mechanics, the word schlieren is commonly associated to an optical techniques for capturing the density variation along a specified direction. This technique is applicable to fluid flows embedding regions of variable density, such as, for instance, supersonic flows or liquid/gas mixtures. The advantage of this technique lies in its simplicity and in the fact that it is non-intrusive—only optical access is to be guaranteed and no seeding of the flow is required. The schlieren technique is widely used in many different fields from automotive (Lee et al., 2012) to medical research (Tang et al., 2009) or from fundamental physics (Hirano et al., 1985) to micro-fluidics (Huang et al., 2007).

In a standard schlieren apparatus a collimated light beam is cast through the test chamber. The apparatus usually includes a set of mirrors and optical lens used to drive the light from the source, which is

assumed to be point-wise, across the domain. A shutter, called *knife*, is placed at the focal point of the light beam to isolate rays that were deflected by inhomogeneities in the continuum. In this way, only a fraction of the initial light beam eventually reaches the screen, where the schlieren image is projected. The schlieren image is therefore a two-dimensional picture of the three-dimensional flow domain, characterized by darker and brighter regions, whose intensity depends on the sign of density variations encountered by light rays within the continuum. The orientation of the knife determines the components of the density gradient the measurement system is sensitive to. Several experimental techniques were derived from Schlieren such as, for instance, the Backround Oriented Schlieren BoS (Venkatakrishnan and Meier, 2004), the Laser Schlieren Deflectometry LSD (Schäfer et al., 2012) or Moire deflectometry (Kafri and Glatt, 1985).

The goal of this paper is to present the *VirtuaSchlieren* software: a hybrid GPU/CPU-based schlieren simulation tool for obtaining schlieren images from three-dimensional Computational Fluid Dynamics (CFD) simulations of variable density flows, including non-ideal fluid flows. The algorithm moves from the work of Yates (1992); Brownlee et al. (2011) and it is capable of accounting for diverse experimental arrangements such as the single pass or the double-pass reflected schlieren apparatuses. Briefly, to obtain the virtual schlieren image from CFD simulations of the flow field, a large number of light paths is computed starting from the light source and across the computational domain. The ray trajectories and their light intensity are influenced by the local value of the fluid refraction index, which is computed from the interpolated value of the density within the computational cell. To this end, a fast search algorithm is to be implemented to determine the pertinent grid element. Finally, the virtual schlieren is reconstructed on the screen plane. The virtual schlieren image may be compared against the experimental one to assess the accuracy of the mathematical fluid model and to determine specific properties of the fluid. Imaging science offers a wide set of different image processing methods that could accomplish this goal, among the others it is worth mentioning statistical methods such as those based on convolution and cross-correlation, see for instance Keating et al. (1975) or those derived from the Keypoint Matching approach (Lowe, 1999).

Non-ideal compressible-fluid dynamics (NICFD) flow fields can be studied thank to a general formulation of the density-refraction index relation based on the Lorentz-Lorenz model, see Liu and Daum (2008). NIFCD applications range from super-critical CO_2 flows in nuclear reactors (Dostal et al., 283-301) or pharmaceutical processing (Subramaniam et al., 1997) to compressible flows of molecularly complex fluids in the close proximity of the liquid-vapour saturation curve, which are of interest for Organic Rankine Cycle for renewable energy applications (Drescher and Bruggeman, 2007; Lang et al., 2013; Quoilin et al., 2013; Casati et al., 2014). Current experimental activities in NICFD include pressure and temperature measurements and schlieren visualization of supersonic flows in converging-diverging nozzles (Spinelli et al., 2016).

This paper is organized as follows: in Sec. 2 the physical modeling of the experimental procedure for producing schlieren images is presented. The hypotheses and the governing equations are presented and commented. Sec. 3 presents the structure of the software. In section Sec. 4 the fast search algorithm and related aspects that are key to the code performances are presented. In Sec. 5 exemplary results are shown. In particular, the influence of the grid resolution on the final Schlieren image is discussed and the dependence on the type of grid elements is assessed. Moreover, different algorithms and data structures are evaluated and tested in terms of computational efficiency. An assessment of the code performances and scalability is carried out in Sec. 5, using both multi-CPU and multi-GPU architectures. Conclusions are reported in Sec. 6.

2. Ray tracing model

Light rays traveling across a media are deflected due to changes in the value of the refractive index n, which strongly depends on the local value of density. As a general rule, stronger density variations lead to larger light beam deflections.

To compute the light beam trajectories in media with variable density, Snell's law can be applied provided that the medium properties are discontinuous (Feynman et al., 1964). Examples are liquid-gas interfaces in multiphase flows, shock waves and contact surfaces in compressible flows. A more general approach is required to deal with flow-fields that exhibit smoother gradients, such as, for instance, turbulent flows. In

this respect, Fermat's *Principle of Least Time*, also known as the *principle of the shortest optical path* (Born and Wolf, 1999), states that *a light ray travels from point A to point B along the only path that brings the ray to its destination B in the shortest time.* According to Fermat's principle it is possible to derive the law of refraction and to define the light beam deviation in a more general framework, see Settles (2001).

To introduce the governing equation for the propagation of light in a variable density medium, a Cartesian reference frame defined by three direction vectors $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ is now introduced. Consider a collimated light beam propagating along a nominal direction $\hat{\mathbf{s}}_0$, initially aligned with the $\hat{\mathbf{z}}$ axis. According to Yates (1992), the incremental change in the angular deflection ϵ across a non-homogeneous region can be expressed as

$$\epsilon_x \approx \tan \epsilon_x = \frac{1}{n} \frac{\partial n}{\partial x} dz \quad if \quad \epsilon_x \to 0 \tag{1}$$

$$\epsilon_y \approx \tan \epsilon_y = \frac{1}{n} \frac{\partial n}{\partial y} dz \quad if \quad \epsilon_y \to 0 \tag{2}$$

where $n = n(x, y, z)$ is the local value of the refraction index of the material. Under the hypothesis of this model, interactions due to electromagnetic forces are negligible and each light beam can be dealt with independently. Therefore, from the knowledge of the local value of the refraction index n, it is possible to reconstruct the trajectory of each light beam by integration of (1) and (2). At each integration step i the deflection angle is computed and the new direction of propagation is retrieved as

$$\hat{\mathbf{s}}^{i+1} = \hat{\mathbf{s}}^{i} + \nabla n \tag{3}$$

where $\hat{\mathbf{s}}$ is a unit vector identifying the light beam direction and i is the index related to the integration step. Let Δs is the length of the integration step; the new position $\mathbf{p}$ for the next integration step is immediately obtained via explicit integration as follows

$$\mathbf{p}^{i+1} = \mathbf{p}^{i} + \frac{\Delta s}{n} \hat{\mathbf{s}}^{i} \tag{4}$$

Under the above assumptions, the trajectory of each light beam depends only on gradients of the refraction index. Literature provides a set of mathematical and empirical models that relate the refractive index to the value of density. The Gladstone-Dale model is accurate for compressible flow problems in the gaseous phase and it reads

$$n = \rho K + 1 \tag{5}$$

where K is a fluid-dependent constant determined experimentally. If the K constant of the fluid is not known a priori from experiments, as it is possibly the case for NICFD flows, relation (5) can be substituted by the linearized Lorentz-Lorenz relation, see Liu and Daum (2008), which reads

$$n = \rho \frac{3}{2} \frac{A}{\mathcal{M}} + 1 \tag{6}$$

where A is the molar refractivity and $\mathcal{M}$ is the molar mass of the fluid.

3. Numerical implementation

Fig.1 depicts the flow-chart of the VirtuaSchlieren code, which has a structure similar to the approach proposed by Yates (1992); Brownlee et al. (2011).

First, the computational grid of the fluid domain is generated and the density field is computed using the CFD solver SU2, see Palacios et al. (2013) and Palacios et al. (2014). A pre-processing phase is required to compute the position of the centroid of each grid element (block *Geometry Preprocessing*). Then, a second pre-processing routine (block *Density gradient computation*) is called to evaluate density gradients at each cell centroid. The gradients are assumed to be constant within each element: this assumption introduces no approximations for a linear representation of the solution over triangular and tetrahedral elements. From these data the trajectory of the light beams can be reconstructed using the ray tracing procedure, which

consists of a recursive algorithm made of two steps. First, the pertinent computational cell to which the current integration point $\mathbf{p}^i$ is located, see (4), is identified. Hereinafter, we will refer to this step as the *search step* (Ray search block in Fig. 1). Following the search step, the local elemental value of the density gradient is retrieved and the gradient of the refractive index to be used in (3) is computed using either the Gladstone-Dale model (5) or the linearized Lorentz-Lorenz (6) relations. This loop is repeated until all rays traveled through the test domain, from the light source to the screen where the schlieren image is projected.

According to Brownlee et al. (2011), the relation between the deflection of the beam $\mathbf{e}$ and its intensity I is as follows

$$I = 0.5 - C\left(-e_x \sin\alpha + e_y \cos\alpha\right) \tag{7}$$

where C is a constant that is set to maximize the image contrast and it is usually chosen so to keep the intensity value between 0 and 1. At the screen location, the variation of the intensity of the light is computed from the overall deviation $\mathbf{e}$ as $\mathbf{e} = \mathbf{p}_F - \mathbf{p}_0$, with $\mathbf{p}_F$ and $\mathbf{p}_0$ being respectively the final position over the screen and the expected position corresponding to a uniform condition. In the above relation, α is the angular position of the knife. According to (7), $I = 0.5$ if a light beam reaches the screen unperturbed. The virtual schlieren image is finally created by assigning the value of the light intensity to each graphical pixel according to a uniform gray scale.

4. Fast search algorithms

In Fig. 1, a set of operations is tagged as *perfectly parallel workload*, meaning that the algorithms gathered in the dashed box are very suitable for parallel execution. Indeed, as pointed out in Sec. 2 , the trajectory of each light beam is uncoupled from that of the other beams and therefore the computation of the beam trajectories can be easily implemented in a parallel algorithm. The VirtuaSchlieren code makes use of two different hardware architectures to run massively parallel computations: multi-cores and multi-processors systems such as clusters, using the standard *Message Passing Interface* (MPI), and multi-GPUs architectures via CUDA technology (Compute Unified Device Architecture, see Nickolls (2008)).

Not surprisingly, profiling of the software revealed that the *search step* presented in Sec. 3 is the most demanding task during code execution. Indeed, the *search step* requires one to identify—among the large number of computational cells in which the CFD domain is decomposed—the one in which the current integration point $\mathbf{p}^i$ is located, see relation (4).

Starting from the list of cell centroids, which is computed once and for all at the beginning of the simulation, the pertinent computational cell can be located by finding the index of the centroid which is the closest to the integration point $\mathbf{p}^i$. Note that, for highly stretched meshes or for meshes with non-convex elements, the above choice does not guarantee that the integration point $\mathbf{p}^i$ lies within the element associated to the selected centroid.

The computational cost of the *search step* increases with the number of query points N, which relates to the number of tracked rays, the number M of points in the so-called point cloud, which in the present case is the list of centroids and it depends on the number of grid elements in the CFD domain, and the dimension of the problem in terms of the number D of degrees of freedom associated to each point. Here, $D = 3$ since the degrees of freedom of each point are its three Euclidean coordinates. Since the quality of the numerical schlieren image largely depends on the level of grid refinement and on the number of tracked rays, large values of both N and M are usually expected in applications and therefore the implementation of a fast search algorithm is mandatory.

Starting from a simple brute force algorithm, different search procedures and data structures were evaluated to improve code performances and efficiency for the nearest neighbor search. So-called kd-tree data structures, see Bentley et al. (2008), were eventually chosen, see also Garcia et al. (2008) for a detailed comparison of the diverse search approaches. The nearest neighbor search based on a kd-tree structure relies on three different phases: a *pre-processing step*, to create the structure of the tree, a *climbing step*, during which the algorithm searches from the root down to a leaf, and a *descending step*, from the leaf back to the root. As an example, Fig. 2 depicts a graphical representation of a 2-level tree data structure.

The reader is referred to Bentley et al. (2008) for further details regarding balanced/unbalanced structures, multidimensional trees and for a thorough description.

The *pre-processing step* is needed to order data within the point cloud according to a given criterion and it is performed once and for all for each set of cloud points. Since the problem of interest here involves points defined within an Euclidean space, a natural choice is to order points with respect to the spatial coordinates x, y and z. If cloud points are ordered, for instance, along the x axis, it is possible to find a particular value $\bar{x}$, the root, which divides the cloud into two equally large subsets. Each subset may be further divided and the procedure can be carried out to create several levels, until each subset contains one element only or few of them. The ordering criterion may change at each level by alternating the reference coordinate x, y or z. The branch fork can therefore be of type x, y or z and the corresponding fork value is indicated as $\bar{x}$, $\bar{y}$ and $\bar{z}$, respectively.

The *climbing step* proceeds as follows. At each branch fork, the coordinate value of the query point (e.g. x) is compared to the fork value (e.g. $\bar{x}$) to determine which path must be chosen: if the query point coordinate is larger (smaller) than the fork value, the left (right) branch is followed. The procedure continues until a leaf is reached. If the leaf contains more than one element, brute force is used to retrieve the index of the closest cell. The *climbing step* returns a first guess regarding the position $\mathbf{p}^* = (x^*, y^*, z^*)$ of the centroid closest to $\mathbf{p}^i$.

Note that the point $\mathbf{p}^*$ identified during the climbing step may not be the point closest to $\mathbf{p}^i$ within the cloud. Indeed, there may be alternative paths that identify point closer to $\mathbf{p}^i$, see Bentley et al. (2008). In the following, point $\mathbf{p}^*$ is therefore referred to as the current best guess and it is possibly improved during the *descend step* from the leaf to the root. On the way down, the difference between the fork value and the corresponding coordinate of the best guess value is evaluated at each fork. For example, for a x-type fork, the difference $d = |\bar{x} - x^*|$ is computed. If $||\mathbf{p}^* - \mathbf{p}^i|| > d$, then it is possible that inside the alternative subset a cloud point closer to $\mathbf{p}^i$ exists. In this latter case, the alternative branch is climbed up again until a new leaf is reached to check for better candidates. The procedure is carried out until the algorithm eventually returns to the root: at this point the search is complete and the point found actually corresponds to the nearest centroid.

Two different approaches based on kd-tree structures were considered in this work: the Approximate Nearest Neighbor (ANN) search, implemented for CPU architectures in the open-source ANN library, see Arya et al. (1998), and the simpler *PoliTree* implemented on both CPU and GPU architectures. Differently from the ANN library, which implements a linked-list data structure that is built during the pre-processing step, the PoliTree is based on an array structure, which is defined after the pre-processing step but it is more suitable for GPU implementation. Since the ANN library was designed to exploit multi-CPUs platform only, no calculation were carried out on GPUs using the ANN library. Moreover, it has to be recalled that in the simplified PoliTree approach, the descending step, which requires to evaluate a large number of conditional jumps, is not performed. As mentioned earlier, the trajectory of each light beam is reconstructed independently and the nearest neighbor search is accomplished by evaluating the position of a light ray against the centroid of each grid element. The list of cell centroids, casted in a kd-tree structure on the master rank, must be therefore available to each thread. For a multi-CPUs configuration such list is cloned from the master rank into the address space of each process while it is copied from the CPU space to the global memory of each GPU for the implementation exploiting CUDA. Once the kd-tree structure is available to all threads, together with the density gradient field, light rays are equally distributed among processes, to reconstruct the virtual schlieren image. For both the CPU and the GPU implementation, each thread computes the trajectory of a set of light rays, from the light source to the projection screen. In the next section, some exemplary test cases are provided to evaluate differences between the resulting virtual schlieren images.

5. Results

Some exemplary results for compressible flows including smooth variation of the density as well as shock discontinuities are presented in this section to assess the capabilities of the code. The first test case regards a planar problem, namely, the supersonic flow of an ideal, dilute gas over a compressive-rarefaction ramp. The

CFD solution is obtained by simulating a three-dimensional domain. This test case provides the benchmark to evaluate the predicted schlieren image against different levels of grid resolution and different types of element (either tetrahedral or hexahedral). The second test case concerns a genuinely three-dimensional problem, namely an axisymmetric body of revolution in a supersonic parallel flow. Numerical results are assessed against the experiments presented in Venkatakrishnan and Meier (2004). For this latter test case, which requires a fairly large computational grid, the algorithm efficiency and scalability are assessed for both CPU and GPU clusters. A third, final case regards the supersonic expansion of a parallel flow around a sharp corner in non-ideal compressible-fluid conditions. In all considered cases, under the boundary layer approximation, the effects of viscosity and thermal conductivity are assumed to be relevant only within a thin layer close to the wall and therefore the Euler equations are solved to compute the fluid dynamics in the domain. The CFD solver SU2 is used in all computations.

5.1. Planar supersonic flow

The first test case is the supersonic, two-dimensional flow of air in dilute conditions around a compressive-rarefaction ramp. The computational domain for CFD simulation is three-dimensional and it is obtained by extruding the two-dimensional geometry along the axis normal to the plane. The domain length in the x or flow direction is 8, the height is 4 and the thickness is 1 unit length. The angle of the ramp is $\arctan(1/3)$. The surface grid in the x-y plane is depicted in Fig. 3. On the upper and lower boundaries a slip-wall condition holds while no condition is applied at the outlet boundary, on the right, since the flow there is supersonic. Two different computational grids, made of tetrahedral and hexahedral elements, respectively, were used in the simulations. Both grids are composed of approximately 80 000 nodes.

With reference to Fig. 4 showing the Mach distribution from the numerical simulations, the flow features two sharp variations of the fluid velocity to accommodate the corresponding slope discontinuities at the compressive and rarefaction corners. An oblique shock wave is observed at the compressive corner. Past the shock wave, the flow is parallel to the inclined ramp. A continuous rarefaction fan, centered at the rarefaction corner, deviates the flow so that the velocity is again parallel to the initial flow direction. Therefore, the flow field is characterized by four separate regions: the uniform, parallel flow region between the inlet and the shock-wave, the uniform-flow region bounded by the shock and by the left boundary of the rarefaction fan, the rarefaction fan and the uniform-flow region from the end of the fan to the outlet. Fig. 7 compares the density from the CFD solutions over the two grids against the analytical solution along the $y = 1.1$ line. In the numerical solutions the shock wave appears as a continuous variation of the density, as expected due to the use of an artificial viscosity approach. The angle of the oblique shock-wave from the numerical simulation is $\hat{\beta} = 30.75°$, which compares fairly well to the expected value of $\beta = 30.71°$. Fig. 5 and Fig. 6 report respectively the computed schlieren image for the hexahedral and the tetrahedral grid. The position of the knife is such that positive gradients in the x directions appear as lighter regions, whereas negative ones are associated to darker regions. The bright line matches the oblique shock-wave, which is correctly represented using both grids. The shock-wave appears to be smeared out onto a wider region in the image produced from the tetrahedral grid. Fig. 8 reports light intensity profile along a reference line crossing the domain at $y = 1.1$. Vertical dashed lines depicted in Fig. 8 represent the position, respectively from left to right, of the shock and of the characteristic lines bounding the rarefaction fan. The tetrahedral grid is found to diffuse light into a wider region. A significant dependency of the ray tracing process from the orientation of grid edges is observed.

5.2. Axisymmetric body in supersonic flow

In this section an experimental test case is reproduced numerically and the predicted schlieren images are compared against actual pictures taken during experimental activity. Moreover, this test case was exploited to assess code performances in terms of computational speed and scalability.

The domain consists of a small rectangular supersonic wind tunnel, the domain has an optical access on both sides so that a single passing schlieren can be used. An axisymmetric body, a cone-shaped bow that merges into a cylindrical rear segment, is placed along the channel axis. The flow is parallel to the cylinder axis x and the Mach number is 2. The fluid is air in dilute conditions. Fig. 9 and Fig. 10 briefly report

geometrical data and the set up of the experimental apparatus. A detailed description of the test-rig set up and a complete discussion about results can be found in Venkatakrishnan and Meier (2004).

Domain symmetries were exploited to reduce the dimension of the problem and only a 90° sector of the body was simulated. The computational grid is composed by approximately 7 million tetrahedral elements. Differently from the planar case presented in Sec. 5.1, in this test case three dimensional effects are relevant: the leading shock, for instance, is no longer a planar discontinuity but a cone-shaped surface with vertex at the bow of the body. As a consequence, the local gradient of density can have a non-zero component along any direction.

Fig. 13 reports the experimental schlieren image to be compared against Fig. 11 and Fig. 12, showing the numerical schlieren images produced using the ANN and the PoliTree algorithm, respectively. Numerical images fairly resembles the experimental one. The conical shock-wave and the rarefaction fan are correctly reproduced in terms of light gradients and wave angles, thus confirming the correctness of the present approach. The ANN and the PoliTree fast search algorithm produce very similar results: no significant differences are notable between Fig. 11 and Fig. 12. A more quantitative comparison is shown in Fig. 14, where the light intensity values obtained by the two approaches are compared along a straight line located on the x-y plane 0.024 m above the axis of symmetry. Only negligible differences are observed in Fig. 14 between the two approaches. Significantly large number of elements in the computational grid, more than 7 million, makes this problem a suitable test bench for assessing the code performances and for evaluating the use of GPU/CPU architectures. For this purpose, using a fixed number of light rays (4 million), the PoliTree algorithm and the ANN library were compared in terms of computational speed and scalability.

Tab. 1 presents an analysis of code performances. Data reported in Tab. 1, under the label RT (Ray Tracing), refer only to the computational time needed to reconstruct ray trajectories. The OH (Overheads) column report the time required to arrange data structure, i.e. read the mesh and the solution, allocate variables, parallel partitioning and communication, data preprocessing and output saving. Information about performances are reported first for the complete ANN search and then for the PoliTree algorithm. Fig. 15 and Fig. 16 report the result of the scalability analysis on a pictorial representation on a logarithmic scale. As expected, the PoliTree algorithm is consistently much faster than the ANN library: results show that the total elapsed time can be reduced by almost two orders of magnitude. This speed-up is mainly due to the fact that in the PoliTree approach, albeit the climbing phase is performed following the exact climbing algorithm described in Bentley et al. (2008), no descent algorithm is implemented to verify that the current best guess is indeed the closest centroid, as discussed in section 4. In all test cases considered here, this strong simplification was found not to hamper code predictions with respect to the full ANN algorithm.

The cluster used to carry out this study consists of 192 cores distributed among 16 nodes, each node containing two Intel Xeon X5650 2.67 GHz (6 core units). Due to the fact that the ray tracing process is a perfectly parallel workload, both algorithms follow an almost ideal scaling law, as it can be appreciated in Fig. 15 and Fig. 16 .

On the other hands, overheads increase as additional operations are needed during parallel execution. In particular, a certain amount of time is spent creating data structures for parallel execution and for transferring information back and forth among CPUs. Moreover, as the number of processors is further increased, communication starts also suffering lag due to the transmission of information between processors on the same node and between nodes themselves. Overheads are more noteworthy for the code version implementing the PoliTree search because the tree structure was designed to suit Graphical Processing Units (GPU) requirements at best. Indeed, linked list as those implemented in the ANN library are more efficient concerning the creation of the tree structure since this particular process involves data ordering and thus it requires a large number of insertions and cancellations. GPUs performances are instead enhanced using arrays, since data is stored in contiguous memory registers. Therefore, the tree structure for the graphical unit is created using arrays but this slows down the building process. Results reported in Tab. 1 point out that both algorithms keep scaling down almost perfectly, as the number of CPUs is increased. Also the overall execution time keeps decreasing quite regularly for the complete ANN search since, in this case, overheads are just a small portion of the total cost ($\approx 8\%$ for 96 CPUs). Results regarding the PoliTree algorithm suggest instead that a saturation limit is being approached. Indeed, though RT process scales down almost perfectly from 48 to 96 processors, overheads now represents a considerable amount ($\approx 90\%$

on 96 CPUs) of the total time.

VirtuaSchlieren performances were then assessed using GPUs architectures. Two different hardware configurations are considered: a personal computer equipped with an Intel Core i3-4150 3.50 GHz Processor and a NVIDIA GeForce 750Ti graphical card and a professional workstation equipped with two Intel Xeon E5-2630 v2 2.60 GHz 6 core processors and two NVIDIA Tesla k40M units. Tab. 2 reports the RT time, overheads and the overall execution time for GPU runs. Though significantly cheaper than a HPC cluster, a personal computer equipped with a GeForce 750Ti GPU allows to reach performances that are comparable to a 24 core machine. In part, this is due to lower overheads since data are transferred only once, back and forth, from the CPU to the GPU. Using the NVIDIA Tesla k40M GPU results in a reduction of one order of magnitude for the RT time and the total execution time is halved. Comparison against CPU hardware show that one NVIDIA Tesla k40M GPU can manage to carry out the RT process in a time comparable to that required by 96 CPUs. Moreover, due to the significant reduction of the overheads, the total execution time is reduced by almost 36 %. The time needed to enroll the RT algorithm is further halved using a second Tesla GPU, though the total execution time is only slightly reduced. Results thus suggest that a significant computational cost reduction may be achieved exploiting GPUs capabilities. A proper analysis of scalability, for the PoliTree algorithm, is left for further investigation.

5.3. Non-Ideal Compressile Fluid flow

In this section an exemplary test case regarding isentropic expansion of a fluid in non-ideal conditions is presented. The inviscid supersonic flow of MDM (octamethyltrisiloxane) over an expansion edge is simulated. Fluid properties at the inlet, identified by subscript 0 hereinafter, are reported in Tab. 3 and are well within the non-ideal thermodynamic region.

The domain length in the x or flow direction is 1, the height (y) is 0.8 and the thickness (z) is 0.3 unit length. The flow is turned by an angle $\Theta = 15.95°$ over an edge. With reference to Fig. 17, the inlet boundary is on the left hand side while the outlet boundary on the right. Slip-wall condition holds on all the remaining boundaries. The numerical grid is made of 363,558 hexahedral elements (380,800 nodes). The non-ideal thermodynamic behavior of the fluid is modeled through the iPRSV (Peng-Robinson Stryjek-Vera) equation of state implemented in SU2, here briefly recalled.

$$P\left(T, v\right) = \frac{RT}{v - b} - \frac{a\alpha^2\left(T\right)}{v^2 + 2bv - b^2} \tag{8}$$

$$\alpha\left(T, \omega\right) = \left[1 + \varphi\left(\omega\right)\left(1 - \sqrt{\frac{T}{T_{\mathrm{cr}}}}\right)\right] \tag{9}$$

Eq. 8 is the equation of state for the pressure P, as a function of the temperature T and specific volume v. The constant $R = \mathcal{R}\mathcal{M}$ is the gas constant ($\mathcal{R}$ is the universal gas constant), a is a measure of the molecular attractive forces and b is the covolume. The function $\alpha\left(T\right)$ further models the inter-molecular attraction forces, which depend on the value of temperature and on the value of the acentric factor ω. The function $\varphi\left(\omega\right)$ is the energy in the dilute, ideal gas limit. The reader is referred to Peng and Robinson (1976) and Stryjek and Vera (1986) for a throughout description of the iPRSV equation of state.

Fig. 17 reports the density field computed using the SU2 NICFD solver. An expansion fan centered at the expansion corner is clearly visible and it separates the upstream (0) region from the downstream one.

Fig. 19 shows respectively the final schlieren image obtained using ANN library, on the left, and using the PoliTree library, on the right: 4 millions light rays were used to produce the schlieren image on the screen. Fig. 18 depicts light intensity trends along a line crossing the domain at $y = 0.25$. Despite its lower precision in determining the position of the nearest neighbor, the PoliTree simplified search is proved to deliver results comparable to those obtained from the ANN library.

6. Conclusions

In this paper a hybrid GPU/CPU-based virtual schlieren simulator, called VirtuaSchlieren, is presented. The software uses standard ray tracing algorithm already implemented in Yates (1992); Brownlee et al. (2011) to produce the virtual schlieren image from a CFD simulation of the flow.

The code structure allows to exploit both multi-processor and graphical-processing unit capabilities at the same time and it implements both the simple Gladston-Dale model for dilute gases and the Lorentz-Lorenz model for predicting the refraction index in non-ideal flow conditions, including non-ideal compressible-fluid flows in the close proximity of the liquid-vapor saturation curve. Note that for the virtual schlieren technique presented here, the use of the Gladstone-Dale or the Lorentz-Lorenz model amounts simply to a different definition of the proportionality constant relating the density ρ and the refraction index n, namely, the constants K and $3A/(2M)$ in (5) and (6), respectively. Given that the constant C in (7) is then arbitrary chosen to maximize the image contrast, the two models deliver the same results in terms of virtual schlieren images, though the local value of the refraction index n is possibly different.

All CFD simulations were carried out using the open-source SU2 software. Numerical results compare fairly well with the analytical model and to a reference experimental Schlieren image for a conical body in a supersonic flow.

Code performances were assessed by carrying out a scalability analysis on a multi-CPUs HPC cluster and on a professional GPUs workstation. A simplified version of a fast search algorithm based on kd-tree structures was introduced to reduce computational costs and to improve performances for GPU hardware. Though less accurate, this algorithm proved to be much faster than a complete, optimized, kd-tree search while delivering comparable results in terms schlieren image quality.

7. Acknowledgements

The authors thank Prof. Fabio Cozzi for helpful discussions on the modeling of the refraction index and Luca Virtuani for his contributions to an early version of the code. This research is supported by ERC Consolidator Grant N. 617603, Project NSHOCK, funded under the FP7-IDEAS-ERC scheme. The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the two Tesla K40 GPU used in this work.

References

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., Wu, A. Y., Nov. 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J. ACM 45 (6), 891–923.
URL http://doi.acm.org/10.1145/293347.293348

Bentley, J., Buck, I., Garland, M., Skadron, K., 2008. Multidimensional binary search trees used for associative searching. ACM queue 6 (2), 40–53.

Born, M., Wolf, E., 1999. Principles of Optics, 7th Edition. Cambridge University Press.

Brownlee, C., Pegoraro, V., Shankar, S., McCormick, P., Hansen, C., 2011. Physically-based interactive flow visualization based on schlieren and interferometry experimental techniques. IEEE Transactions on Visualization and Computer Graphics 17 (11).

Casati, E., Vitale, S., Pini, M., Persico, G., Colonna, P., 2014. Centrifugal Turbines for Mini-Organic Rankine Cycle Power Systems. Journal of Engineering for Gas Turbines and Power 136, 122607–1–11.

Dostal, V., Hejzlar, P., Driscoll, M., 283-301. The supercritical carbon dioxide power cycle: comparison to other advanced power cycles. Nuclear technology 154 (3).

Drescher, U., Bruggeman, D., 2007. Fluid selection for the Organic Rankine Cycle (ORC) in biomass power and heat plants. Applied Thermal Engineering 27 (1), 223–228.

Feynman, R., Leighton, R., Sands, M., 1964. The Feynman Lectures on Physics, 1st Edition. Addinson-Wesley.

Garcia, V., Debreuve, E., Barlaud, M., 2008. Fast k nearest neighbor search using gpu. CVPR Workshop Comput. Vision GPU.

Hirano, K., Matsumoto, C., Shimoda, K., Yamamoto, T., 1985. Cross check of electron density distribution by moirÃł-schlieren technique and mach-zehnder interferometry. Japanese Journal of Applied Physics 24 (11), 1518–1521.

Huang, C., Gregory, J., Sullivan, J., 2007. A modified schlieren technique for micro flow visualization. Measurement Science and Technology 18 (5).

Kafri, O., Glatt, I., 1985. Moire deflectometry: a ray deflection approach to optical testing. Opt. Eng. 24 (6).

Keating, T., Wolf, P., Scarpace, F., 1975. An improved method of digital image correlation. Photogrammetric Engineering and Remote Sensing 41 (8).

Lang, W., Almbauer, R., Colonna, P., 2013. Assessment of Waste Heat Recovery for A Heavy-duty Truck Engine Using An ORC Turbogenerator. Journal of Engineering for Gas Turbines and Power-Transactions of the ASME 135 (4), 042313–1–10.

Lee, J., Kim, N., Min, K., 2012. Measurement of spray characteristics using the background-oriented schlieren technique. Measurement Science and Technology 24 (2).

Liu, Y., Daum, H., 2008. Relationship of refractive index to mass density and self-consistency of mixing rules for multiocomponent mixtures like ambient aerosols. J. of Aerosol Science 39, 974–986.

Lowe, D., 1999. Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, pp. 1150–1157.

Nickolls, J., 2008. Scalable parallel programming with cuda. Communication of the ACM 18 (9), 509–517.

Palacios, F., et al., 2013. Stanford university unstructured (su2): An open-source integrated computational environment for multi-physics simulation and design. AIAA Paper 2013-0287 51st AIAA Aerospace Sciences Meeting and Exhibit.

Palacios, F., et al., 2014. Stanford university unstructured (su2): Open-source analysis and design technology for turbulent flows. AIAA Paper 2014-0243 52nd AIAA Aerospace Sciences Meeting.

Peng, D., Robinson, D., 1976. A new two-constant equation of state. Ind. Eng. Chem. Fundam. 15 (1), 59–64.

Quoilin, S., Broek, M. V. D., Declaye, S., Dewallef, P., Lemort, V., 2013. Techno-economic survey of Organic Rankine Cycle (ORC) systems. Renewable and Sustainable Energy Reviews 22 (0), 168–186.

Schäfer, J., Foest, R., Kewitz, T., Šperka, J., Weltmann, K., 2012. Laser schlieren deflectometry for temperature analysis of filamentary non-thermal atmoshperic pressure plasma. Review of Scientific Instruments 83.

Settles, G., 2001. Schlieren and shadowgraph techniques. Springer.

Spinelli, A., Cozzi, F., Dossena, V., Gaetani, P., Zocca, M., Guardone, A., 2016. Experimental investigation of a non-ideal expansion flow of siloxane vapor mdm. GT2016-57357 ASME Turbo Expo June 13-17, 2016: Turbomachinery Technical Conference and Exposition.

Stryjek, R., Vera, J., 1986. Prsv: an improved peng-robinson equation of state for pure compounds and mixtures. The Canadian Journal of Chemical Engineering 64 (2), 323–333.

Subramaniam, B., Rajewski, R., Snavely, K., Aug. 1997. Pharmaceutical processing with supercritical carbon dioxide. Journal of Pharmaceutical Sciences 86 (8), 885–890.

Tang, J., Liebner, T., Craven, B., Settles, G., 2009. A schlieren optical study of the human cough with and without wearing masks for aerosol infection control. Journal of the Royal Society Interface (6), 727–736.

Venkatakrishnan, L., Meier, G., 2004. Density measurements using the background oriented schlieren technique. Experiments in Fluids 37 (2), 237–247.

Yates, L., 1992. Images constructed from computed flowfields. AIAA Paper 92-4030 17th AIAA Aerospace Ground Testing Conference.

List of Figures

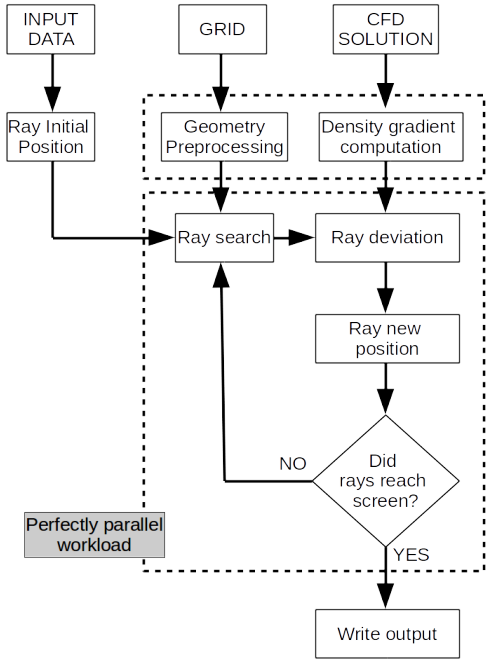

Figure 1: VirtuaSchlieren flow-chart. The diagram presents the logical work-flow implemented within the code.

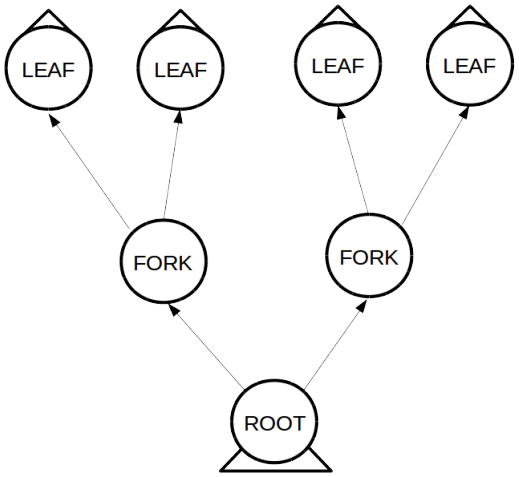

Figure 2: graphical representation of an exemplary kd-tree structure. At each fork a conditional statement drive the search towards a specific leaf.

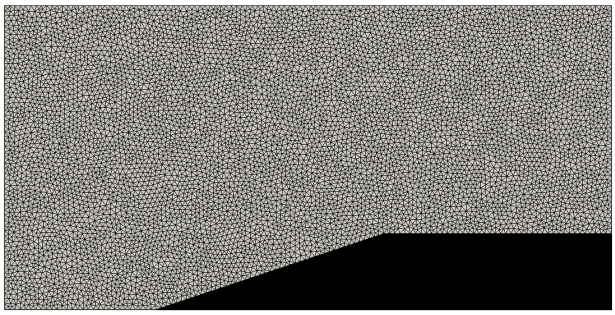

Figure 3: Surface grid for the planar compressive-rarefaction ramp test case. The flow is from left to right.

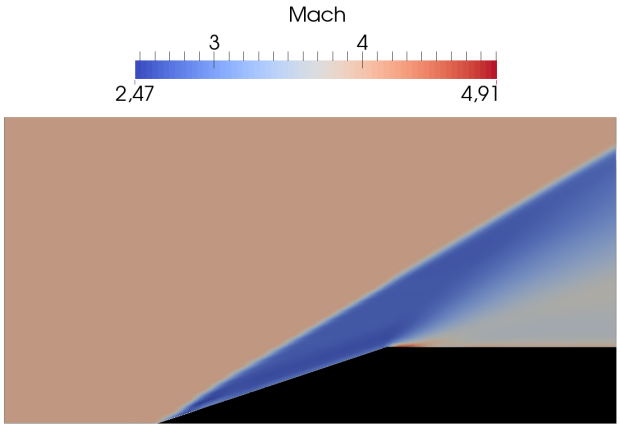

Figure 4: Flow solution computed by SU2 solver over the tetrahedral grid for the planar nozzle test case.

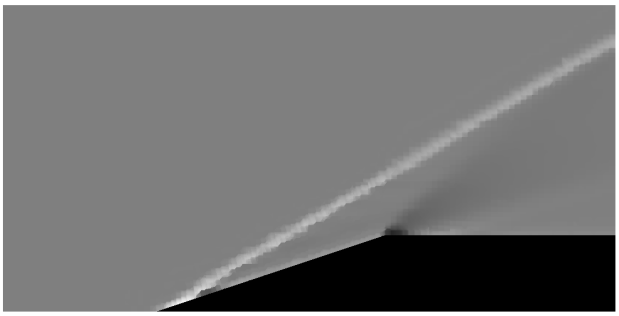

Figure 5: Schlieren image from CFD solution computed over the hexahedral grid.

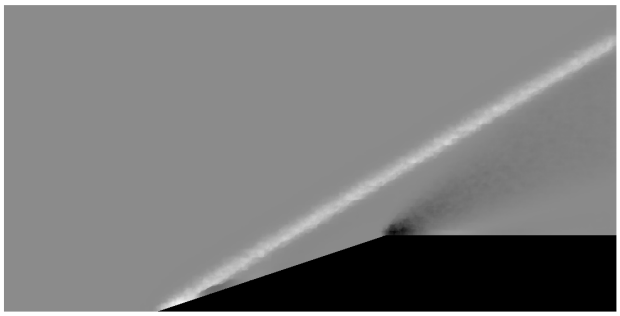

Figure 6: Schlieren image from CFD solution computed over the tetrahedral grid.

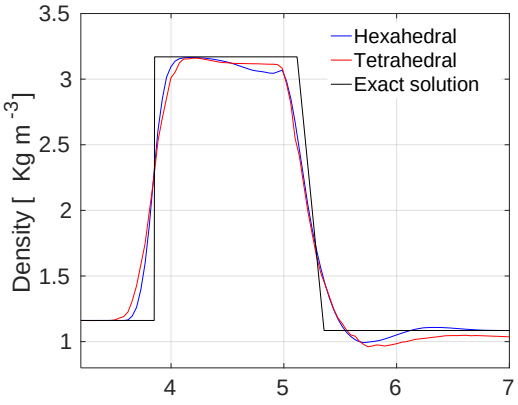

Figure 7: Compressive-rarefaction ramp. Comparison of the numerical solution computed over the hexahedral and the tetrahedral grid against the analytical solution along $y = 1.1$.

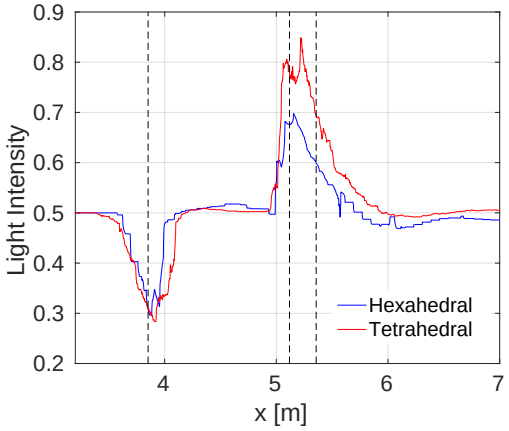

Figure 8: Compressive-rarefaction ramp. Comparison of the the light intensity at $y = 1.1$ of virtual schlieren images from the two grids (cf. figures 6 and 5). Dashed lines represent, from left to right, the analytical position of the shock, the first and the second characteristic lines bounding the rarefaction fan.

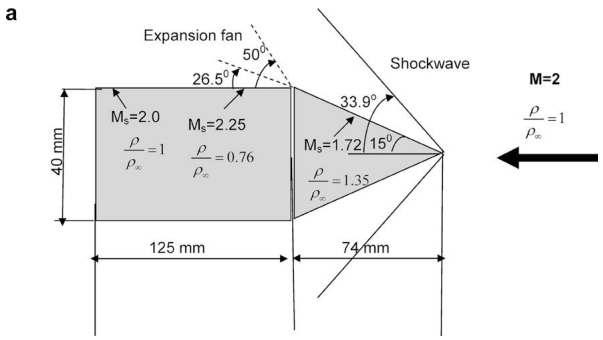

Figure 9: picture reports geometrical data of the cone-shaped body and depicts main features of the analytical solution (Venkatakrishnan and Meier, 2004)

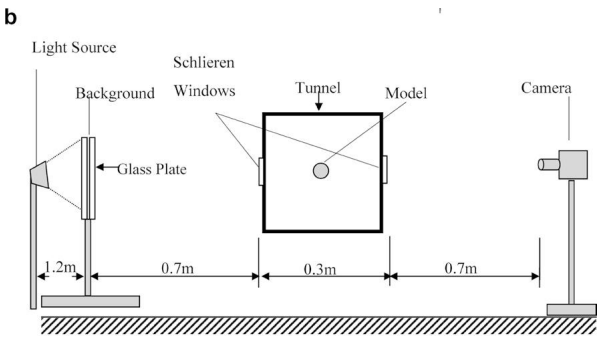

Figure 10: pictorial representation of the test rig used to investigate the flow field around the supersonic body (Venkatakrishnan and Meier, 2004)

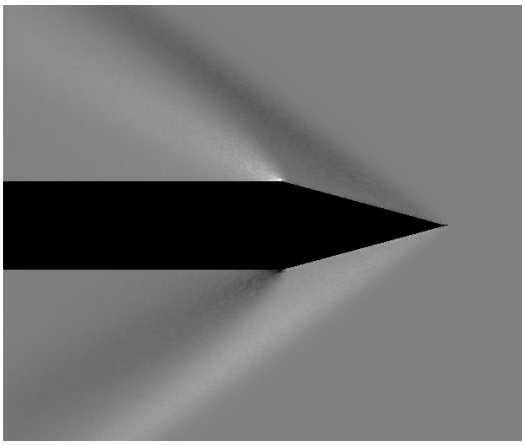

Figure 11: Virtual image predicted for the supersonic cone-shaped body, using the ANN library.

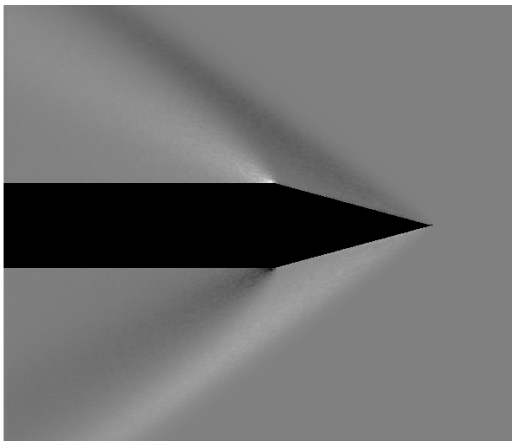

Figure 12: Virtual image predicted predicted for the supersonic cone-shaped body, using the PoliTree algorithm.

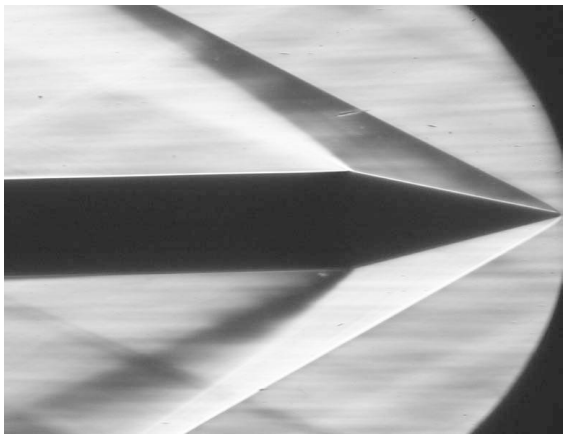

Figure 13: Experimental schlieren image of the density field around a cone-shaped body within a supersonic flow (Venkatakrishnan and Meier, 2004).

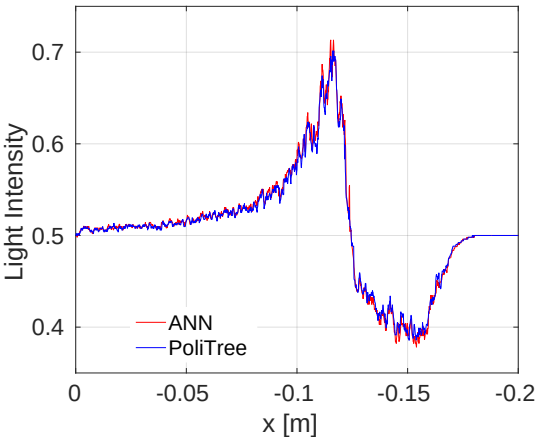

Figure 14: supersonic body. Comparison of light intensity trends extracted from the numerical schlieren image predicted exploiting the ANN and the PoliTree search algorithm.

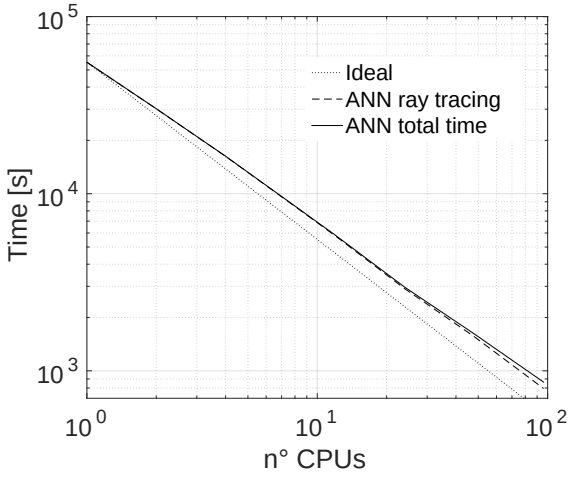

Figure 15: ANN. Ideal, ray tracing and overall execution time for predicting the schlieren image of the supersonic body using different number of processors.

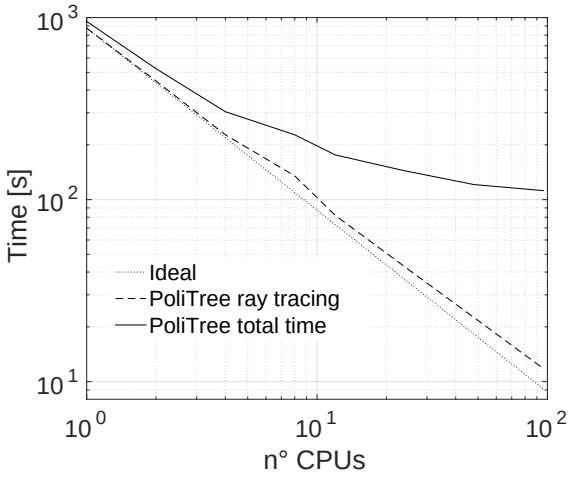

Figure 16: PoliTree. Ideal, ray tracing and overall execution time for predicting the schlieren image of the supersonic body using different number of CPUs.

Figure 17: Computed density field for the non-ideal supersonic expansion test case. The field is scaled using the value ρ_0 of the density at the inlet.

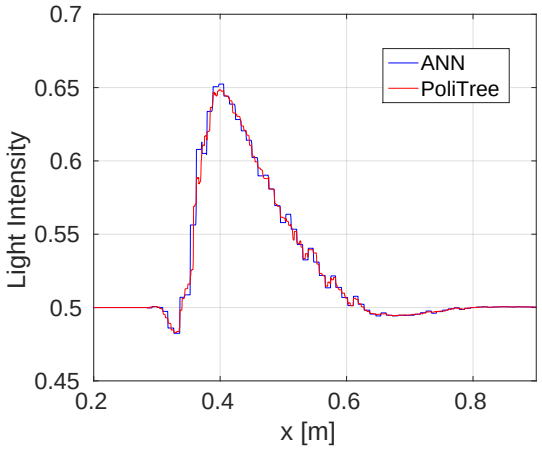

Figure 18: Non-ideal expansion. Comparison of light intensity trends extracted from the numerical schlieren image predicted using the ANN and the PoliTree search algorithm.

Figure 19: Comparison of virtual schlieren images predicted using the VirtuaSchlieren tool. Left figure was obtained using the ANN library while picture on the right using the approximate PoliTree algorithm.

List of Tables

	ANN			PoliTree		
n Procs	RT [s]	OH [s]	Tot [s]	RT [s]	OH [s]	Tot [s]
1	55185	43	55228	875	78	953
2	30155	41	30196	448	76	524
4	16245	39	16284	227	77	304
8	8475	44	8519	135	92	227
12	5771	51	5822	81.9	94	176
24	2905	58	2963	42.8	101	144
48	1560	57	1617	22.5	99	121
96	795	69	864	11.8	100	112

Table 1: VirtuaSchlieren performances for the supersonic body problem. Table shows ray tracing time (RT), overheads (OH, related to the pre-processing phase and to time lag due to partitioning and communication among processors) and code total execution time (Tot), using the ANN library and the PoliTree algorithm, for different number of CPUs (n Procs). Results refer to double precision computations.

GPU	RT [s]	OH [s]	Tot [s]
GF 750Ti	99	49	148
1 Tesla k40M	13	59	72
2 Tesla k40M	6.7	60	67

Table 2: VirtuaSchlieren performances for the supersonic body. Table shows ray tracing time (RT), overheads (OH, related to the pre-processing phase and to time lag due to partitioning and communication among processors) and code total execution time (Tot), using the ANN library and the PoliTree algorithm, for different GPU configurations.

Fluid	γ	R [J/KgK]	ω	P_0 [Pa]	T_0 [K]	ρ_0 [Kg/m^3]	M_0
MDM	1.0125	35.152	0.529	1500112	569.373	202.8870	1.7

Table 3: MDM properties and its conditions at the inlet for the non-ideal supersonic expansion test case. γ is the specific heat ratio, R is the gas constant and ω is the acentric factor of the fluid. P_0, T_0, ρ_0 and M_0 are, respectively, the static pressure, the static temperature, the value of density and of the Mach number characterizing the fluid at inlet of the domain.