A parallel programming application of the A* algorithm in digital rock physics

Computers & Geosciences 187 (2024) 105578

Contents lists available at ScienceDirect

Computers and Geosciences

journal homepage: www.elsevier.com/locate/cageo

A parallel programming application of the A* algorithm in digital rock physics

Alice Raeli *, Eloisa Salina Borello, Filippo Panini, Cristina Serazio, Dario Viberti

Politecnico di Torino, DIATI – Dipartimento di Ingegneria Dell'Ambiente, Del Territorio e Delle Infrastrutture, Corso Duca Degli Abruzzi 24, 10129, Torino, TO, Italy

ARTICLE INFO

Keywords:
Pathfinding algorithm
Energy storage
Porous medium
Geometric tortuosity
A* algorithm

ABSTRACT

Pore-scale analysis and characterization of reservoir rocks provide valuable information for the definition and management of underground hydrogen storage and CO_2 storage or sequestration.

This article presents an optimized implementation of the A* algorithm, the most popular pathfinding method in the presence of obstacles. In this context, the algorithm is applied to recognize the minimum length connected paths through each flow direction of 3D images of a porous medium representative of a reservoir rock. The identification of the main paths allows the characterization of the pore space and the calculation of fundamental petrophysical properties such as tortuosity and effective porosity, which can be used for permeability estimation. Compared to other algorithms available for pore-scale characterization, such as the pore centroid, A* provides a better approximation of the pore space available for the flow and, therefore, a reliable characterization of the petrophysical properties. On the other hand, path identification is significantly consuming in terms of time and memory. In this paper, an efficient and optimized implementation based on C++/OpenMP programming language is presented.

The proposed implementation aims to the analysis of large-scale models profiting from parallelization, memory optimization, and enhanced managing of dead paths. Three test cases of increasing sizes are presented, to analyze the advantages and the disadvantages of the algorithm as the number of explored points increases. The 3D binary images analyzed are related to a synthetic domain (1 million voxels) and two actual sandstone samples (about 4 and 64 million voxels respectively). The code is validated against a Matlab serial implementation, showing a significant improvement in efficiency. Remarkable test cases of several millions of voxels were afforded, overcoming the memory and execution slowness issues. Moreover, the proposed implementation is suitable for large pore-scale models run in HPC environments.

1. Introduction

Underground porous media, such as depleted gas reservoirs and aquifers, represent an interesting option for large-scale hydrogen storage and CO_2 storage or sequestration. The definition of an optimal reservoir management strategy is typically based on large-scale (or macroscale) reservoir flow dynamic modeling. The full-field modeling is based on a macroscale parameterization, derived by the extension for the multiphase flow of the Darcy equation, accounting for permeability, relative permeability, porosity, saturation, and fluids properties. The definition of reliable simulation models requires a deep understanding of the fluid flow phenomena dominating the reservoir behavior and a throughout characterization of reservoir rocks. To minimize the uncertainties affecting rock characterization, all the possible information sources must be considered (Bratvold et al., 2009; Bratvold and Begg, 2010; Santos et al., 2021; Viberti et al., 2007, 2008, 2018; Viberti and Verga, 2012). In this view, pore scale (or microscale) analysis of reservoir rocks can provide valuable information.

Natural porous media (i.e.: in this contest reservoir rocks) are characterized by a significant variability of grains and pores size and a complex tortuous structure (Ghanbarian et al., 2013). Macroscale parameters, such as permeability and effective porosity, are strongly affected by the geometrical features of the pore space (Bear, 1988) at the microscale, thus affecting the corresponding fluids' dynamic behavior. For instance, absolute permeability (k) can be expressed in terms of parameters related to the microscale pore channels' geometry (such as tortuosity of pore channels and average pore size), as summarized by the Kozeny-Carman equation (Carman, 1937; Tu et al., 2018):

* Corresponding author.
E-mail address: alice.raeli@polito.it (A. Raeli).

https://doi.org/10.1016/j.cageo.2024.105578
Received 2 October 2023; Received in revised form 19 March 2024; Accepted 19 March 2024
Available online 24 March 2024

$$k = \frac{\varphi r_H^2}{c\tau^2} \tag{1}$$

where τ is the tortuosity, introduced by (Carman, 1937), which accounts for the convoluted fluid paths in the pore structure; r_H is the hydraulic radius, which is representative of the average pore radius; c is the Kozeny-Carman coefficient, depending on parameters describing the complex porous microstructure, such as pore to throat size ratio (Ozgumus et al., 2014) or constriction (Berg, 2014); and φ is the porosity. The geometric tortuosity of a rock sample in a certain direction can be calculated as the ratio between the shortest path across the sample avoiding the solid obstacles and the length of the sample itself (Adler, 1992; Clennell, 1997). According to equation (1), a characterization of the pore network geometry is necessary to estimate the permeability and to describe the flow behavior.

Thanks to the significant improvement in computational power and high-resolution 3D imaging techniques, such as *Micro Computed Tomography* (Micro-CT), Focused Ion Beams (FIB)/Scanning Electron Microscopy (SEM), Confocal Laser Scanning Microscopy (CLSM) (Blunt et al., 2013) and the deep learning based fusion of digital rock images at different scales (Liu and Mukerji, 2022; Wu et al., 2019, 2023), pore-scale simulation and characterization have seen significant development in the last decades, in particular for reservoir rocks (Al-Raoush and Madhoun, 2017; Guibert et al., 2015; Lindquist et al., 1996). Despite this, calculations on digital images of the pore space are typically heavy (Ramstad et al., 2019), and require significant computational time, comparable to or even higher than full-field multiphase flow reservoir simulation. As a consequence, the discipline is extremely interesting from a scientific point of view but not yet ready to be implemented in the workflow of reservoir study (Benetatos and Viberti, 2010). Due to that, the possibility of improving the computational performances of pore-scale analysis could make it an interesting tool to be integrated into the aforementioned workflow.

The present paper is focused on a methodology for the geometrical analysis of the pore space. Technical literature reports several methods for pore structure quantification of two-dimensional (2D) and three-dimensional (3D) image analysis. Among them, algorithms for pore skeleton extraction (Raeini et al., 2017), such as thinning process, medial surface (Al-Raoush and Madhoun, 2017), medial axis (Lindquist et al., 1996), and pore centroid method (Cooper et al., 2014) construct a morphological skeleton representation of the pore space. However, these approaches are sensitive to minor object boundary perturbations, often caused by binarization and noise (Shaked and Bruckstein, 1998). For instance, in the case of pore centroid, the computed values of tortuosity are affected by sudden changes in the homogeneity of the medium (Espinoza-Andaluz et al., 2022). Segmentation-based algorithms (Gostick, 2017; Jia et al., 2023; Øren and Bakke, 2003; Rabbani et al., 2014; Sheppard et al., 2004; Xu and Yu, 2008) allow the determination of pore connectivity. However, they are sensitive to pore surface roughness. (Wang et al., 2020). Therefore, any artificial roughness potentially induced by image binarization could affect the reliability of the results. An alternative approach to characterize the pore structure is the maximal ball method, which consists of expanding spheres into pore throats according to their sizes (Arand and Hesser, 2017; Dong and Blunt, 2009). However, the pore space can be underestimated when it is tortuous (Wang et al., 2020).

A promising approach is based on the A* pathfinding method to explore the porous domain using binarized images. The A* pathfinding algorithm (Hart et al., 1968; Nilsson, 1982) is largely used to determine the shortest, and so optimal, path between a starting and a target point (Russell and Norvig, 2021). A set of inlet-outlet pairs are defined on opposite sides of the analyzed samples in each direction. The identified paths can be employed to estimate various geometrical parameters characterizing the porous space such as effective porosity, tortuosity, and permeability, from 2D and 3D binary images of well-connected rock samples (Panini et al., 2022; Peter et al., 2023; Salina Borello et al., 2022; Viberti et al., 2020). Since the search of each path is independent of the others, the algorithm is very well-suited for parallelization.

A comparison between A*, skeleton shortest path, Dijkstra algorithm, and pore centroid method to compute the geometric tortuosity in 2D porous media is deepened in (Espinoza-Andaluz et al., 2022). The comparison made on 540 synthetic 2D samples gave results in good agreement, with discrepancies within 10%. A* and the Dijkstra algorithm produce paths closer to the obstacles with respect to Skeleton shortest path and Point centroid, which consequently generate slightly higher tortuosity values (Espinoza-Andaluz et al., 2022). A comparison between A* and point centroid in 3D synthetic porous media was presented in (Ávila et al., 2022), where the A* algorithm characterizes most paths better. However, in their proposed implementation, it was not possible to obtain satisfactory geometric tortuosity values for media of one million voxels using the A* algorithm, so they suggest the pore centroid method as the size of the sample increases (Ávila et al., 2022).

This paper aims to enhance the computing performance of the algorithm presented in (Salina Borello et al., 2022), exploiting code parallelization and reducing memory usage, thus allowing to address 3D samples of several tens of millions of voxels.

The new implementation is applied firstly on a synthetic sample characterized by a limited size (1 million voxels) and then on two real sandstone samples of sizes up to 64 million voxels. The results are satisfactory in terms of both computational time and memory occupancy.

2. Materials and methods

The code here proposed is a C++ (Kernighan and Ritchie, 1978) implementation through an OpenMP multithreading method of the A* pathfinding approach presented in (Panini et al., 2022; Salina Borello et al., 2022; Viberti et al., 2020), which relied on a serial Matlab code. Analogously to (Salina Borello et al., 2022) the methodology is applied to a 3D binary description (grain = 0, void = 1) of a synthetic or real portion of the rock. For each of the three flow directions (x, y, and z), the A* algorithm is used to find any existing shortest path connecting each point of a set of *inlets* to each point of a set of *outlets*. The inlet and outlet points for each direction were selected by running an algorithm based on the medial axis approach and cluster analysis, as presented in (Salina Borello et al., 2022). The implementation presented in this paper contains relevant optimization in terms of efficiency, runtimes, and memory handling.

2.1. Dataset

The proposed implementation was validated and here we present the application to three different test cases of cubic rock images (i.e. $L_1 = L_2 = L_3 = L$) characterized by different sizes and petrophysical properties. The analyzed cases denoted by *Sample100 (L = 100 voxels),*

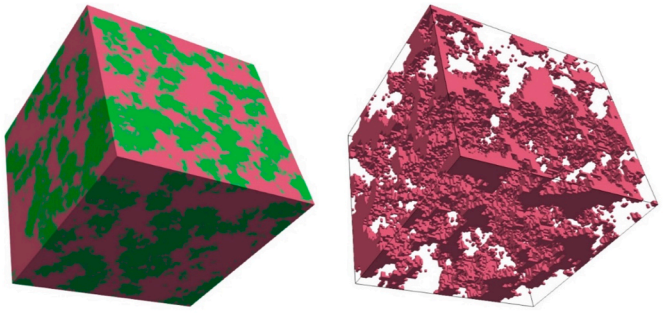

Fig. 1. Sample100 (left) with pores extraction (right). The green zone represents the grain voxels.

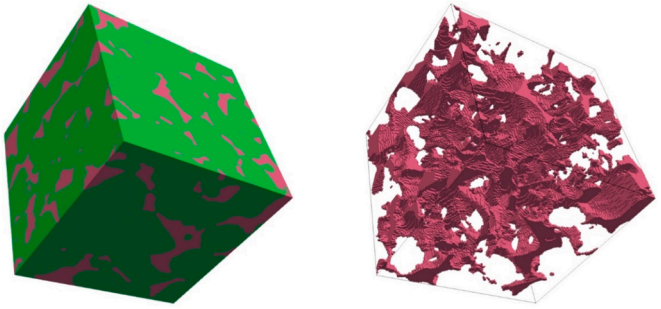

Fig. 2. Sample161 (left) with pores extraction (right). The green zone represents the grain voxels.

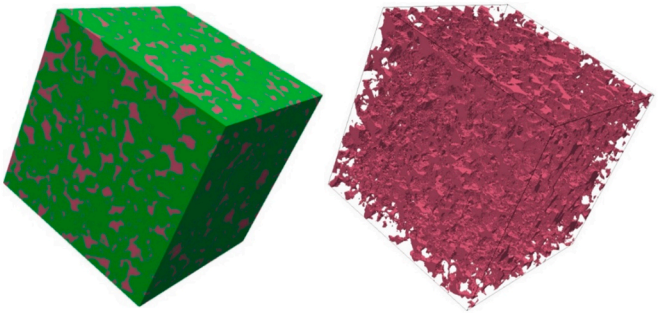

Fig. 3. Sample 400 (left) with pores extraction (right). The green zone represents the grain voxels.

Sample161 (L = 161 voxels), and *Sample400 (L = 400 voxels)*, are shown in Fig. 1, Fig. 2, and Fig. 3, respectively. Petrophysical properties are summarized in Table 1.

Sample100 is a 3D synthetic isotropic porous domain generated using the Quartet Structure Generation Set QSGS algorithm (Wang et al., 2007), presented in (Salina Borello et al., 2022). This domain is the coarsest of the analyzed cases.

Sample161 is a subset of the 3D binary image of a real sandstone sample, named S1 in (Dong and Blunt, 2009), available online (Imperial College London, 2023a). The 3D binary image was obtained from the 3D grayscale output of a micro-CT scanner by segmentation (Dong and Blunt, 2009). The investigated portion was taken from the central part of the sample; the size of the subsample was selected as the smallest subset preserving the total porosity (maximum discrepancy 1.5%).

Sample400 is the 3D binary image of a real Berea sandstone sample (Dong and Blunt, 2009), available online (Imperial College London, 2023b). Berea sandstone is a standard material consisting of quartz with minor amounts of dolomite, clays, and feldspar, characterized by fine grains, with closely spaced planar bedding (Okabe and Blunt, 2004).

Table 1
Petrophysical properties of analyzed samples.

	Sample100	Sample161	Sample400
Formation	synthetic	real sandstone S1	real sandstone Berea
Resolution (μm)	7	8.683	5.345
Source	QSGS generated	Segmentation of MicroCT	Segmentation of MicroCT
Length (vox)	100	161	400
Volume (vox^3)	1e6	~4.17 e6	64e6
Porosity (%)	29.97	14.3	19.6
Average Permeability (mD)	832	1678	1286
Anisotropy (k_x/k_z)	1.1	1.52	1.13
Reference	Salina Borello et al. (2022)	Dong and Blunt (2009)	Dong and Blunt (2009)

2.2. A* algorithm outline

The A* algorithm detailed below has applications in gaming, mapping, and artificial intelligence (Duchoň et al., 2014; Erke et al., 2020). It is the most popular technique for crossing finite graphs and for solving minimum paths in the presence of obstacles (Candra et al., 2020) because of its optimality (i.e. the output is the best solution according to the criteria), efficiency (i.e. other path finding algorithms, e.g. Dijkstra, will take more time, computational resources or both) (Candra et al., 2020; Foead et al., 2021; Martell and Sandberg, 2016; Permana et al., 2018; Rachmawati and Gustin, 2020), and completeness (if at least one solution exists then the algorithm is guaranteed to find a solution in a finite amount of time) (Russell and Norvig, 2021).

Given a grid with obstacles (represented as a graph), A* aims to find a path, starting from a specific node (inlet cell) to the given goal node (outlet cell), having the smallest cost in terms of traveled distance. The algorithm accomplishes the objective by iteratively exploring cells: at each iteration one of the potential paths originating at the inlet is extended until the termination criterion is satisfied. The tree of all explored potential paths is kept in memory. Thus, at each iteration, A* needs to determine which of the explored paths to extend, i.e. which cell to explore next. The selection is based on the cost of the path traveled so far and an estimate of the cost required to extend it to the goal. In other words, A* selects the path that minimizes the cost function:

$$f(P_i) = g(P_i) + h(P_i) \tag{2}$$

where $g(P_i)$ is the distance calculated over the incremental path already identified until the point P_i , and $h(P_i)$ is the forward cost estimation from P_i to the target. The $h(P_i)$ can be computed in different ways since the real distance to the target in the presence of obstacles is unknown. In our implementation, an Euclidean estimation approximating the distance to the target as the forward cost was applied. If the heuristic function used is admissible (i.e. it never overestimates the distance to the goal point), then A* is guaranteed to return an optimal solution (Katz and Domshlak, 2010). In our application, being the sample finite and the chosen heuristic function admissible, if at least a path between the inlet and the outlet exists, A* is ensured to find the minimum one.

The space complexity of A* (i.e. the memory usage), comparable with other graph search algorithms, is characterized by an oversized memory as it keeps all explored nodes in memory (Nilsson, 1982; Russell and Norvig, 2021).

Even if competitive in comparison with other pathfinding approaches, the time complexity of A*, i.e. the computational time used during the search process, is quite high. In the worst case, depending on the heuristic function, the number of nodes explored can be exponential in the depth of the solution, which is equal to the sample length (L) in our case. Given $I_{x,y,z}$ the number of inlets on a side of the sample and $O_{x,y,z,}$ the outlets, the total number of paths reckoned per direction is $I \bullet O$. Thus, the time complexity (t_{max}) for the path exploration of the entire sample could reach:

$$t_{max} \propto (I_x \bullet O_x) \bullet (I_y \bullet O_y) \bullet (I_z \bullet O_z) e^L \tag{3}$$

Algorithm 1. Pathfinding pseudocode. The dead paths control function pseudocode is reported in Algorithm 2.

Input: Set inlet and target points from $\mathcal{G}$

openclose($\mathcal{P}$) = 0, current = inlet, cond = *true*

openclose(current) = 1

Compute f(current), h(current)

(1)

While cond=*true*

find cost= min $f(P), \forall P \in \mathcal{P}$

Call: Dead paths control (Algorithm 2)

if current=target **then**

cond = *false*, **goto**(1)

end if

openclose(current) = 1, f(current) = FLTMAX

for $P_i \in$ **neights(current) do**

Compute $g(P_i)_{new}$

if $g(P_i)_{old} \leq g(P_i)_{new}$ *and* openclose(P_i) ≠ 0

continue;

else if $g(P_i)_{new} \leq g(P_i)_{old}$ *and* openclose(P_i) ≠ 2

$g(P_i) = g(P_i)_{new}$, compute $f(P_i), h(P_i)$

P_i.Parent ←current, openclose(P_i) =1

end if

end for

goto(1)

End while

Output: Optimal Path

The pseudocode implemented for each couple inlet/outlet is sketched in Algorithm 1, where a dead path managing code block called *Dead paths control* was added to the original algorithm (see Section 2.4).

2.3. Code parallelization

Being the path search procedure natively independent for each path, that is not affected by the others, we adopted a parallel code approach by OpenMP (Open Multiprocessing) programming code (Chandra, 2001). According to the parallelization, inlet/outlet couples are subdivided into different threads computing the algorithm in Algorithm 1.

2.3.1. Computer architectures

Firstly, we used a multicore architecture representative of a personal computer. The architecture supports Linux OS 11th Gen Intel(R) Core (TM) i7-11800H @ 2.30 GHz, 8 cores using a maximum of 16 threads = 16 Gb RAM.

As long as we focus on HPC development, in the second stage we tested the largest case on a more complex architecture, representative of a small cluster: Intel Xeon Scalable Processors Gold 6130 2.10 GHz 16 cores, using 32 threads. The computational resources were provided by HPC@POLITO, a project of Academic Computing within the Department of Control and Computer Engineering at the Politecnico di Torino (http://www.hpc.polito.it).

2.4. Non-interconnected paths managing

The complex nature of porous media may present isolated pores, not accessible to the fluid flow. When a couple *inlet/outlet* cannot be linked by a path, we classify the corresponding path as *dead*, and the corresponding inlet or outlet encapsulated by grain is named as *isolated*. The computation of dead paths is not functional to the post-processing characterization of properties such as tortuosity, and thus it represents a waste of time. As previously said, for each path, the number of nodes explored in the worst case is proportional to the length of the sample, yielding a computational effort given by Eq. (3). The computational cost for a path turns out to be particularly severe if the outlet cannot be reached (i.e. the outlet is isolated) because in such a case the exploration of the porous volume connected with the inlet is almost exhaustive. Conversely, the explored region is very limited in the case of isolated inlets. In (Salina Borello et al., 2022) when a selected inlet (or an outlet) is isolated from the connected porosity (i.e. bounded by grains) all the pathways starting from (or leading to) that point are anyway explored. As a consequence, the selection of an isolated outlet can dramatically increase the overall computational cost, especially in the case of a large domain. Being able to automatically detect an isolated outlet and consequently skip the calculation of all the associated dead paths, allows us to avoid this relevant waste of time.

Algorithm 2. Dead paths control pseudocode.

If cost > tol **then**

Declare the path dead

If visitedPoints $< \frac{|\mathcal{P}|}{2}$ **then**

Set the inlet as isolated

Else

Switch(inlet, outlet)

goto(1) (Algorithm 1)

End if

End if

This is accomplished by the algorithm outlined in Algorithm 2: when a dead path is encountered, the reverse path between the inlet and the outlet is searched, aiming at determining if one of the two is isolated. The criterion to detect if the point is isolated is based on the number of explored points. Isolated inlet points induce a very limited exploration, while isolated outlets induce an extended exploration. Identifying isolated inlets (or outlets) relevantly speeds up the process because it allows excluding any further path search from such inlet (or outlet) points.

Conversely, whenever an inlet/outlet pair is not linked due to compartmentalization in the pore volume, even if none of the two is isolated, the exploration covers a significant number of points from both directions. Being not isolated points, they are not excluded from further path searches from such inlet (or outlet) points.

Explored dead paths are not considered in the post-processing assessment of geometrical parameters, such as tortuosity.

2.5. Optimization of memory handling

The rock sample is encoded from a binary image as a set of points $\mathcal{G}$,

each one characterized by a linearized global index uniquely related to its position $P(x,y,z) = x + y * n_x + z * n_x * n_y$, where $n_{x,y}$ is the number of points in the subscript direction, and by a Boolean value S representing the *pore* or *grain* status, set as $S = 0$ and $S = 1$ respectively.

Let $\mathscr{P}$ be the sub-set of points belonging to the pore volume (i.e. $S = 0$). In addition to the global index, each point $P \in \mathscr{P}$ is associated with a porous index, which is related to its position within the set $\mathscr{P}$. To take advantage of memory, the points store their global and porous indices as internal properties.

Given a point $P \in \mathscr{P}$ we define the *neighborhood* of P as the set of points in $\mathscr{P}$ immediately adjacent to it in a predetermined distance called *connecting distance*. In this work, the connecting distance is set as $c_d = 1$ to provide a very detailed path, voxel by voxel, thus avoiding possible path approximation errors in the case of small isolated grains (Salina Borello et al., 2022). The neighborhood of each point is calculated just after the sample binary load and the information is saved as a point internal property in the form of a vector of pointers. Storing the neighborhood of each point has a negligible cost in terms of memory, being limited to C++ pointers handling. However, the speed up in terms of time is the real advantage of this choice, due to the recurrent calls to neighbors' information during the optimal path construction from a point to its successor as presented in Algorithm 1.

Once the $\mathscr{P}$ set is populated, the set $\mathscr{G}$ representing the entire domain is destroyed because only the pore volume points have to be explored to find possible paths. Thus, the part of memory storing the entire structure $\mathscr{G}$ can be deallocated. Considering that in natural rocks pore volume ranges in most cases between 10 and 30% of the domain, the memory advantage is relevant.

A modification in the data structure created during the point exploration by the A* algorithm was introduced. The A* is commonly implemented by using two lists of data, named *Open* and *Close* respectively, the first one containing the data not explored yet, and the second one containing the examined points with their cost values (Foead et al., 2021). However, *Open* and *Close* are commonly non-ordered lists, thus presenting drawbacks on searching strategies, since the connection with the corresponding position on the sample geometry could require additional memory access. Open and Close are implemented in Matlab as 3D-matrices large as the entire sample size, including grain cells, thus involving a huge memory waste. In our implementation, these two lists are replaced by a linearized vector *openclose*, as long as the pore structure, such that:

$openclose(P) = 0$ *if* $P \in \mathscr{P}$ has not been evaluated

$openclose(P) = 1$ *if* $P \in \mathscr{P}$ has to be evaluated as a neighbor of a previously evaluated point

$openclose(P) = 2$ *if* $P \in \mathscr{P}$ has been evaluated and/or removed as a candidate point.

The pore structure $\mathscr{P}$ containing the voxels information is the largest element stored. This set is the shared memory occupancy of the domain, accessible from all threads, as well as the paths list, which is updated from each thread. In contrast, data related to the algorithm (e.g. *openclose* list, costs, predecessors) are local, i.e. each thread handles them independently, allocating memory for each vector of maximal size $|\mathscr{P}|$ (the cardinality of the $\mathscr{P}$ set).

3. Results and discussion

In this section, we present the results obtained on three samples already resumed in Section 2.1. The first domain considered is the synthetic, isotropic *Sample100* (Fig. 1). The domain is encoded by a logic binary 3D matrix of dimension 100^3 voxels. *Sample100* is not "small" in comparison to the literature, as 1 million voxels were considered out of computation (Ávila et al., 2022), but it is here considered the "small" one, compared to the other cases. The second sample presented, named *Sample161* (Fig. 2) comes from a real sandstone and has a size of about 4 million voxels (see Table 1). For *Sample100* and *Sample161* a comparison with Matlab performances is provided. For a more fair comparison, C++ and Matlab code were run on the same Linux multicore architecture, representative of a personal computer (Section 2.3.1); the results presented in Section 3.2 are related to the specific hardware environment adopted. It is pointed out that the C++/OpenMP code is studied to reach a better performance by increasing the number of threads involved, meanwhile, the compared Matlab code is serial. However, Matlab itself is already optimized on Linux platforms, i.e. some built-in functions run in parallel, taking advantage of the available cores. As a result, a performance improvement is also possible for Matlab code increasing the number of threads.

The last case here reported is *Sample400* (Fig. 3), which is from a real Berea sandstone. Due to its significant size (64 million voxels), this test is particularly inefficient for serial runs, both in terms of timing and RAM employed. The results presented for this case were obtained using 32 threads on two architectures representative of small HPC clusters (see Section 2.3.1). In this case, a comparison with Matlab performances on the whole domain was not accomplished because a single path runtime required from several hours to days in worst cases; the comparison was limited to specific paths.

3.1. Geometric characterization

By way of example, some individuated optimal paths are depicted in Fig. 4, Fig. 5, and Fig. 6 for *Sample100*, *Sample161*, and *Sample400* respectively.

Aiming at measuring the minimum number of inlets/outlets required for a reliable characterization of the geometric properties of the sample, the graphs of tortuosity values convergence as a function of the considered paths are presented in Fig. 7a–c for *Sample100*, *Sample161*, and *Sample400,* respectively.

Fig. 7a suggests that 20 inlets (and 20 outlets) in each direction (i.e. 1200 paths in total) are sufficient to obtain satisfactory tortuosity values for *Sample100*, stabilizing within a variation of 1%. The obtained tortuosity values (Table 2) are indicative of an isotropic medium. The agreement with numerical simulation results was already shown in (Salina Borello et al., 2022).

Fig. 7b suggests that choosing approximately 25 inlet/outlet per direction is sufficient to obtain a satisfactory tortuosity estimate. A slight vertical anisotropy in the medium is observed (Table 2), in accordance with (Dong and Blunt, 2009). Moreover, the obtained pore radius distribution (Fig. 8b) well compares with the distribution of throat radius reported in (Dong, 2007). A comparison in terms of permeability (eq. (1)) is quite complicated since for an anisotropic medium with significant constriction the Kozeny Carman constant, which is sensitive to the characteristics of the porous media, is not defined in the literature. A robust characterization of the constant value from pore structure is still under investigation (Berg, 2014; Koponen et al., 1997; Ozgumus et al., 2014; Singh and Mohanty, 2000; Vidal et al., 2009), and is beyond the scope of this paper. Monophase CFD simulations were conducted with the opensource simulator OpenFoam for validation purposes. The

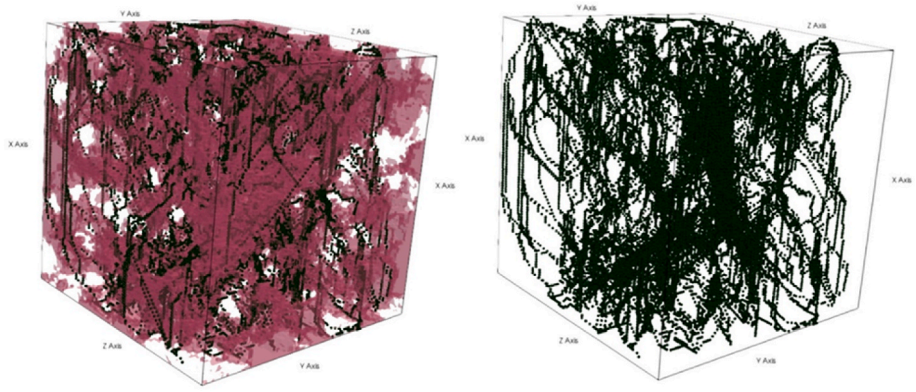

Fig. 4. Sample 100: 2500 optimal paths on X-direction.

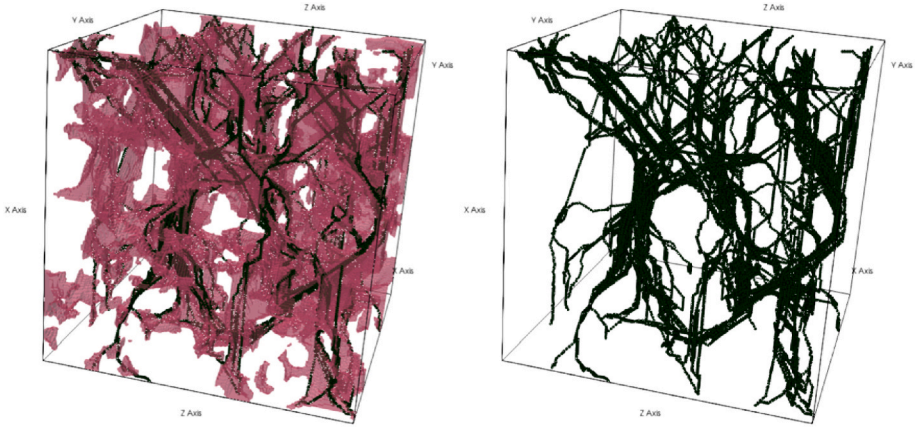

Fig. 5. Sample161: 2500 optimal paths on X-direction.

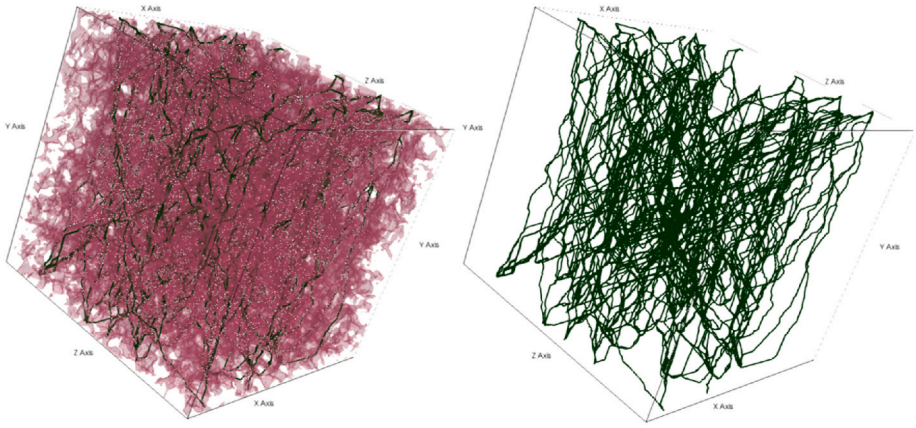

Fig. 6. Sample400: 625 optimal paths on Y-direction.

domain was flushed with water at 20 °C (cinematic viscosity ν = 1e-6 m^2/s). For each investigated direction, ambient pressure was assumed at the outlet and a pressure increment $\Delta p = 10$ Pa was imposed at the inlet; no flow was imposed at the boundaries in the other directions. The obtained tortuosity for each direction were $\tau_x = 1.58$, $\tau_y = 1.55$ and $\tau_z = 1.88$. As expected (Clennell, 1997; Ghanbarian et al., 2013), the geometrical estimate of tortuosity was a bit lower than the hydrodynamic one; however, values are comparable and the detected anisotropy is confirmed.

In the last case (*Sample400*) the tortuosity values started to converge with 25 inlets and 25 outlets (Fig. 7c). A substantial isotropy is observed (Table 2), in accordance with (Dong and Blunt, 2009). The obtained tortuosity is comparable with the value obtained for a 2D section of Berea presented in (Viberti et al., 2020). The obtained pore radius distribution (Fig. 8c) is compatible with the distribution of throat radius reported by (Dong and Blunt, 2009). Assuming Kozeny-Carman constant c = 5.8 for an isotropic medium as in (Koponen et al., 1997), a permeability of about 1700 mD is obtained, which compares with the average literature value provided by literature (Table 1). A comparison with CFD simulation of the hole sample at the same resolution was not possible because the grid generation exceeded the available 32 Gb of RAM.

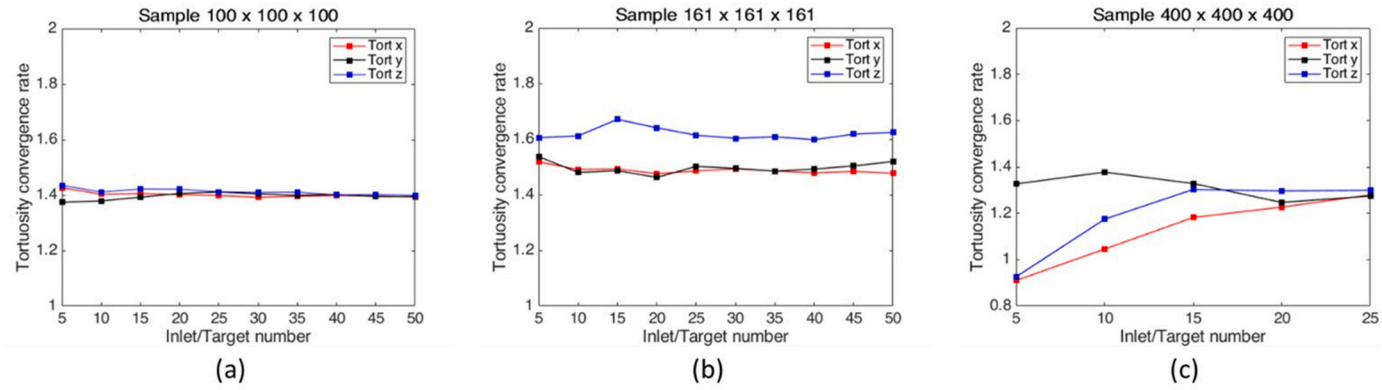

Fig. 7. Tortuosity convergence rate: (a) Sample100; (b) Sample161; (c) Sample400.

Table 2
Tortuosity results at convergence.

	τ_x	τ_y	τ_z
Sample 100	1.37	1.37	1.36
Sample 161	1.48	1.52	1.62
Sample 400	1.39	1.39	1.41

3.2. Computational performance analysis

Table 3 summarizes the obtained path statistics for the three considered samples in terms of path length (min, max, and average), time required per path calculation (min, max, and average), and percentage of detected dead paths. *Sample161,* compared to *Sample100* is characterized by an almost halved porosity (see Table 1) and a higher number of detected dead paths (almost three times) corresponding to about 21% of the investigated paths. This highlights a significant presence of isolated zones in *Sample161*.

The runtime comparison of Matlab code against our C++ code, single thread, and multithread with different numbers of threads (up to 16), is shown in Fig. 9a–b for *Sample100* and *Sample161* respectively.

In the smaller case (*Sample100*), the efficiency of Matlab Toolbox (Natick, 2022) in finding the minimum cost function makes it competitive up to 4 threads. Matlab times are comparable to the multithread run using 4 threads, requiring about 8 h for 7500 paths. Computational times of Matlab are halved compared to the 2-Threads and about one-third of the serial C++ run. Such a result is not surprising since Matlab on Linux Platforms can take advantage of the available cores to run some built-in functions in parallel. Moreover, notice that in this case, only a small percentage of paths were dead (Table 4), a number too low to take advantage of the code optimization on the dead path managing (Section 2.4). However, the 8-Threads and 16-Threads runtimes are uniformly well placed below the 4 h, thus significantly outperforming the Matlab implementation.

Furthermore, Matlab is no longer competitive on larger domains, such as the intermediate case *Sample161*, even with a single thread. Fig. 9b points out firmly the advantages of our implementation compared to Matlab for *Sample161*, independently from the number of threads employed. For instance, the C++ implementation run on a single thread already halves Matlab runtimes, emphasizing the advantages of recognizing isolated zones of the medium in the serial run. On the other hand, the serial C++ runtimes become critical for more than 4800 paths, requiring a multithread approach. By doubling the number of threads, the runtimes are almost halved, as expected from the parallelization approach. Limiting the total number of characterized paths to the minimum quantity required for a satisfactory convergence of the

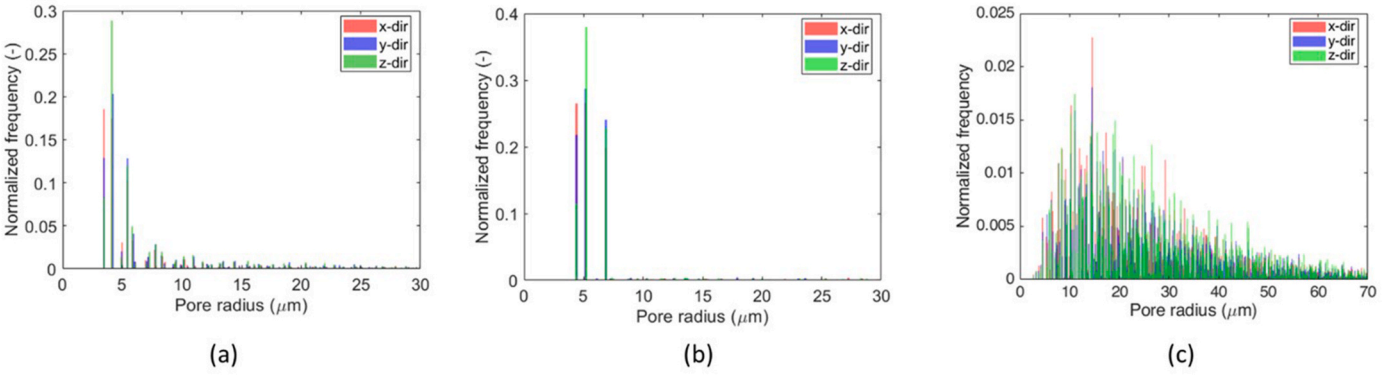

Fig. 8. Pore radius: (a) Sample100; (b) Sample161; (c) Sample400.

Table 3
Paths summarized data computation.

Sample	Dead Paths	Min Path Time	Max Path Time	Average Path Time	Min Length (vox)	Max Length (vox)	Average Length (vox)
Sample100	6.5%	1.83 s	60.07 s	16.41 s	100	142	107.33
Sample161	21%	12.41 s	417.97 s	86.76 s	161	327	189.3
Sample400	8%	3.83 h	88.91 h	21.77 h	400	592	419

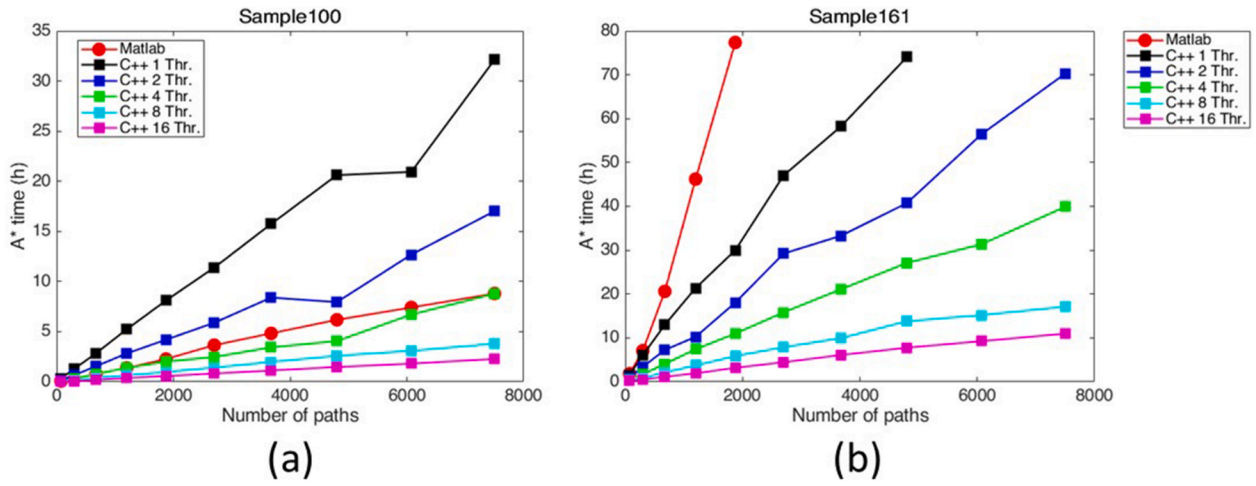

Fig. 9. AStar Times Comparison: (a) Sample100; (b) Sample 161. Runs overcoming three days of computation were stopped; thus, Matlab results over 1875 paths (i.e. 25 inlets/outlets through each direction) and serial C++ results over 4800 paths (i.e. 40 inlets/outlets through each direction), are not represented in (b).

Table 4
Sample400 Matlab and C++ Results – 9 representative paths (x-direction).

Path ID	Inlet	Outlet	Explored Points	Time (s) MATLAB	Time (s) C++
1	[1,81,24]	[400,224,132]	12502004	224423.6	296102
2	[1,81,24]	[400,218,373]	2988470	46888.85	70756.6
3	[1,81,24]	[400,105,276]	1749848	27673.6	35977.1
4	[1,134,289]	[400,224,132]	12502004	211118.23	0.0678
5	[1,134,289]	[400,218,373]	1276093	20788.81	27815.2
6	[1,134,289]	[400,105,276]	1700525	27387.88	28576.3
7	[1,346,30]	[400,224,132]	12502004	196704.48	0.0643
8	[1,346,30]	[400,218,373]	2667391	42060.98	63128.9
9	[1,346,30]	[400,105,276]	2301864	36613.44	45416.7

tortuosity values (see Section 3.1), the required runtime with 16 threads was about 23 min for *Sample100* and 2 h for *Sample100*.

From the memory usage point of view, Matlab required 2 Gb for *Sample100* and 2.4 Gb for *Sample161* while C++, storing all the local data for each thread and maintaining the domain as shared among them, required a maximum of 365 Mb for *Sample100* and 1.5 Gb for *Sample161*.

The last test (*Sample400*), characterized by 64 million voxels (see Table 1), is strongly inefficient when run on a single thread: the runtime for Matlab and serial C++ implementation can employ from 5 to 65 h for a single path. As a consequence, in this case, the comparison with Matlab is reported only for 9 representative paths in the x-direction (Table 4). The entire domain was addressed only with the C++ code, run on the 32 threads configuration of the HPC architecture (Section 2.6); results are reported in Table 4.

In Table 4, for each one of the 9 considered paths, the following information is reported: path ID; inlet and outlet points coordinates; the total number of explored points; computational times obtained using our Matlab and C++ codes. The total running time for the 9 paths on Matlab is about $8.34e^{+05}$ s, corresponding approximately to 230 h or 9 days. Thus, without a parallel run of different paths, the 1875 paths, corresponding to 25 inlets/outlets for direction would require approximately 5 years of simulation, becoming impracticable. For most of the paths, Matlab is slightly more efficient than C++, due to its architecture based on already optimized and parallelized build-in functions in Linux Environment. In some cases (e.g. Path 1 and Path 2) the non-interconnected paths managing approach described in Section 2.4 induces a not negligible additional computational effort. However, when an isolated inlet/outlet is detected (as in Path 1) a significant time saving is obtained by bypassing all the paths sharing the same inlet/outlet (as for Path 4 and Path 7). In the multithread approach, to avoid possible re-computation of isolated areas, the assignment strategy for the inlets and outlets distribution on threads remains a key point.

4. Conclusions

This paper presents an optimized implementation of the A* algorithm, the most popular pathfinding method in the presence of obstacles, extended to 3D domains for application to digital rock physics. The computing performance of the algorithm presented in (Salina Borello et al., 2022) was significantly enhanced by exploiting code parallelization and introducing a non-interconnected paths managing approach.

Three 3D binary images characterized by increasing sizes and different petrophysical properties were adopted to test the algorithm, and to analyze the performances as the number of explored points increased, putting the computing architecture gradually under strain. The first test case is a synthetic domain (1 million voxels), while the other two are actual sandstone samples (about 4 and 64 million voxels respectively). The first two cases were run on a multicore PC employing up to 16 threads, while the third was run on a small cluster with a 32 threads configuration.

The advantages obtained in terms of computational time with respect to the serial implementation presented in Salina Borello et al. (2022) are significant. Matlab implementation can be competitive on the small domain (1 million voxels) up to 4 threads, but it is largely over-performed when 8 or more threads are employed. Moreover, when a larger domain is analyzed (about 4 million voxels) the proposed implementation outperformed the Matlab one for any number of threads, thus allowing to address 3D samples of several tens of millions of voxels.

The proposed code could appear less efficient in terms of runtimes than other methods (Al-Raoush and Madhoun, 2017; Guibert et al., 2015; Horgue et al., 2015; Lindquist et al., 1996). However, it details the effective paths through the flux direction point-by-point as A* guarantees, thus allowing an accurate tortuosity characterization.

The presented implementation aims at a larger HPC-type development: by increasing the number of threads the times are reduced almost linearly.

Code availability section

Astar Library.
Contact: alice.raeli@polito.it.

Program language

C++/OpenMP.

Software required

Linux (g++) – Windows (MINGW compiler).

Program size

73.05 KB.

The source codes are available for download under the GNU license at the link: https://github.com/REDD-PoliTO/Astar.

CRediT authorship contribution statement

Alice Raeli: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Data curation. **Eloisa Salina Borello:** Writing – review & editing, Validation, Methodology, Conceptualization. **Filippo Panini:** Writing – review & editing, Visualization, Resources. **Cristina Serazio:** Validation. **Dario Viberti:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Adler, P.M., 1992. Porous Media: Geometry and Transports, Butterworth-Heinemann Series in Chemical Engineering. Butterworth-Heinemann, Boston.

Al-Raoush, R.I., Madhoun, I.T., 2017. TORT3D: a MATLAB code to compute geometric tortuosity from 3D images of unconsolidated porous media. Powder Technol. 320, 99–107. https://doi.org/10.1016/j.powtec.2017.06.066.

Arand, F., Hesser, J., 2017. Accurate and efficient maximal ball algorithm for pore network extraction. Comput. Geosci. 101, 28–37. https://doi.org/10.1016/j.cageo.2017.01.004.

Ávila, J., Pagalo, J., Espinoza-Andaluz, M., 2022. Evaluation of geometric tortuosity for 3D digitally generated porous media considering the pore size distribution and the A-star algorithm. Sci. Rep. 12, 19463 https://doi.org/10.1038/s41598-022-23643-6.

Bear, J., 1988. Dynamics of Fluids in Porous Media. Dover books on physics and chemistry. Dover, New York.

Benetatos, C., Viberti, D., 2010. Fully integrated hydrocarbon reservoir studies: myth or reality? Am. J. Appl. Sci. 7, 1477–1486. https://doi.org/10.3844/ajassp.2010.1477.1486.

Berg, C.F., 2014. Permeability description by characteristic length, tortuosity, constriction and porosity. Transport Porous Media 103, 381–400. https://doi.org/10.1007/s11242-014-0307-6.

Blunt, M.J., Bijeljic, B., Dong, H., Gharbi, O., Iglauer, S., Mostaghimi, P., Paluszny, A., Pentland, C., 2013. Pore-scale imaging and modelling. Adv. Water Resour. 51, 197–216. https://doi.org/10.1016/j.advwatres.2012.03.003.

Bratvold, R.B., Begg, S., 2010. Making Good Decisions. Society of Petroleum Engineers, Richardson, TX.

Bratvold, R.B., Bickel, J.E., Lohne, H.P., 2009. Value of information in the oil and gas industry: past, present, and future. SPE Reservoir Eval. Eng. 12, 630–638. https://doi.org/10.2118/110378-PA.

Candra, A., Budiman, M.A., Hartanto, K., 2020. Dijkstra's and A-star in finding the shortest path: a tutorial. In: 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA). Presented at the 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA). IEEE, Medan, Indonesia, pp. 28–32. https://doi.org/10.1109/DATABIA50434.2020.9190342.

Carman, P.C., 1937. Fluid flow through granular beds. TRANS. INSTN CHEM. ENGRS 15, 150–166.

Chandra, R. (Ed.), 2001. Parallel Programming in OpenMP. Morgan Kaufmann Publishers, San Francisco, CA.

Clennell, M.B., 1997. Tortuosity: a guide through the maze. SP 122, 299–344. https://doi.org/10.1144/GSL.SP.1997.122.01.18.

Cooper, S.J., Eastwood, D.S., Gelb, J., Damblanc, G., Brett, D.J.L., Bradley, R.S., Withers, P.J., Lee, P.D., Marquis, A.J., Brandon, N.P., Shearing, P.R., 2014. Image based modelling of microstructural heterogeneity in LiFePO 4 electrodes for Li-ion batteries. J. Power Sources 247, 1033–1039. https://doi.org/10.1016/j.jpowsour.2013.04.156.

Dong, H., 2007. Micro CT Imaging and Pore Network Extraction. Imperial College, London.

Dong, H., Blunt, M.J., 2009. Pore-network extraction from micro-computerized-tomography images. Phys. Rev. E 80, 036307. https://doi.org/10.1103/PhysRevE.80.036307.

Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T., Jurišica, L., 2014. Path planning with modified a star algorithm for a mobile robot. Procedia Eng. 96, 59–69. https://doi.org/10.1016/j.proeng.2014.12.098.

Erke, S., Bin, D., Yiming, N., Qi, Z., Liang, X., Dawei, Z., 2020. An improved A-Star based path planning algorithm for autonomous land vehicles. Int. J. Adv. Rob. Syst. 17, 172988142096226 https://doi.org/10.1177/1729881420962263.

Espinoza-Andaluz, M., Pagalo, J., Ávila, J., Barzola-Monteses, J., 2022. An alternative methodology to compute the geometric tortuosity in 2D porous media using the A-star pathfinding algorithm. Computation 10, 59. https://doi.org/10.3390/computation10040059.

Foead, D., Ghifari, A., Kusuma, M.B., Hanafiah, N., Gunawan, E., 2021. A systematic literature review of A* pathfinding. Proc. Comput. Sci. 179, 507–514. https://doi.org/10.1016/j.procs.2021.01.034.

Ghanbarian, B., Hunt, A.G., Ewing, R.P., Sahimi, M., 2013. Tortuosity in porous media: a critical review. Soil Sci. Soc. Am. J. 77, 1461–1477. https://doi.org/10.2136/sssaj2012.0435.

Gostick, J.T., 2017. Versatile and efficient pore network extraction method using marker-based watershed segmentation. Phys. Rev. E 96, 023307. https://doi.org/10.1103/PhysRevE.96.023307.

Guibert, R., Nazarova, M., Horgue, P., Hamon, G., Creux, P., Debenest, G., 2015. Computational permeability determination from pore-scale imaging: sample size, mesh and method sensitivities. Transport Porous Media 107, 641–656. https://doi.org/10.1007/s11242-015-0458-0.

Hart, P., Nilsson, N., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4, 100–107. https://doi.org/10.1109/TSSC.1968.300136.

Horgue, P., Guibert, R., Gross, H., Creux, P., Debenest, G., 2015. Efficiency of a two-step upscaling method for permeability evaluation at Darcy and pore scales. Computational Geosciences 19, 1159–1169. https://doi.org/10.1007/s10596-015-9531-6.

Imperial College London, 2023a. Sandstone S1 [WWW Document]. URL. https://www.imperial.ac.uk/earth-science/research/research-groups/pore-scale-modelling/micro-ct-images-and-networks/sandstone-s1/, 7.26.23.

Imperial College London, 2023b. Berea sandstone [WWW Document]. URL. https://www.imperial.ac.uk/engineering/departments/earth-science/research/research-groups/pore-scale-modelling/micro-ct-images-and-networks/berea-sandstone/, 7.26.23.

Jia, M., Huang, W., Li, Y., 2023. Quantitative characterization of pore structure parameters in coal based on image processing and SEM technology. Energies 16, 1663. https://doi.org/10.3390/en16041663.

Katz, M., Domshlak, C., 2010. Optimal admissible composition of abstraction heuristics. Artif. Intell. 174, 767–798. https://doi.org/10.1016/j.artint.2010.04.021.

Kernighan, B.W., Ritchie, D.M., 1978. C Programming Language, second ed. CreateSpace Independent Publishing Platform.

Koponen, A., Kataja, M., Timonen, J., 1997. Permeability and effective porosity of porous media. Phys. Rev. E 56, 3319–3325. https://doi.org/10.1103/PhysRevE.56.3319.

Lindquist, W.B., Lee, S.-M., Coker, D.A., Jones, K.W., Spanne, P., 1996. Medial axis analysis of void structure in three-dimensional tomographic images of porous media. J. Geophys. Res. 101, 8297–8310. https://doi.org/10.1029/95JB03039.

Liu, M., Mukerji, T., 2022. Multiscale fusion of digital rock images based on deep generative adversarial networks. Geophys. Res. Lett. 49, e2022GL098342 https://doi.org/10.1029/2022GL098342.

Martell, V., Sandberg, A., 2016. Performance Evaluation of A* Algorithms. Faculty of Computing Blekinge Institute of Technology SE–371 79 Karlskrona, Sweden. Karlskrona, Sweden.

Natick, 2022. MATLAB Version: 9.13.0 (R2022b).

Nilsson, N.J., 1982. Principles of artificial intelligence, Reprinted. Symbolic Computation Artificial Intelligence. Springer, Berlin.

Okabe, H., Blunt, M.J., 2004. Prediction of permeability for porous media reconstructed using multiple-point statistics. Phys. Rev. E 70, 066135. https://doi.org/10.1103/PhysRevE.70.066135.

Øren, P.-E., Bakke, S., 2003. Reconstruction of Berea sandstone and pore-scale modelling of wettability effects. J. Petrol. Sci. Eng. 39, 177–199. https://doi.org/10.1016/S0920-4105(03)00062-7.

Ozgumus, T., Mobedi, M., Ozkol, U., 2014. Determination of Kozeny constant based on porosity and pore to throat size ratio in porous medium with rectangular rods. Engineering Applications of Computational Fluid Mechanics 8, 308–318. https://doi.org/10.1080/19942060.2014.11015516.

Panini, F., Salina Borello, E., Peter, C., Viberti, D., 2022. Application of a* algorithm for tortuosity and effective porosity estimation of 2D rock images. In: Indeitsev, D.A., Krivtsov, A.M. (Eds.), Advanced Problem in Mechanics II, Lecture Notes in Mechanical Engineering. Springer International Publishing, Cham, pp. 519–530. https://doi.org/10.1007/978-3-030-92144-6_39.

Permana, S.D.H., Bintoro, K.B.Y., Arifitama, B., Syahputra, A., 2018. Comparative analysis of pathfinding algorithms A *, Dijkstra, and BFS on maze runner game. IJISTECH 1, 1. https://doi.org/10.30645/ijistech.v1i2.7.

Peter, C., Salina Borello, E., Baietto, O., Bellopede, R., Panini, F., Massimiani, A., Marini, P., Viberti, D., 2023. Quantitative characterization of marble natural aging through pore structure image analysis. J. Mater. Civ. Eng. 35, 04023286 https://doi.org/10.1061/JMCEE7.MTENG-15161.

Rabbani, A., Jamshidi, S., Salehi, S., 2014. An automated simple algorithm for realistic pore network extraction from micro-tomography images. J. Petrol. Sci. Eng. 123, 164–171. https://doi.org/10.1016/j.petrol.2014.08.020.

Rachmawati, D., Gustin, L., 2020. Analysis of dijkstra's algorithm and A* algorithm in shortest path problem. J. Phys.: Conf. Ser. 1566, 012061 https://doi.org/10.1088/1742-6596/1566/1/012061.

Raeini, A.Q., Bijeljic, B., Blunt, M.J., 2017. Generalized network modeling: network extraction as a coarse-scale discretization of the void space of porous media. Phys. Rev. E 96, 013312. https://doi.org/10.1103/PhysRevE.96.013312.

Ramstad, T., Berg, C.F., Thompson, K., 2019. Pore-scale simulations of single- and two-phase flow in porous media: approaches and applications. Transport Porous Media 130, 77–104. https://doi.org/10.1007/s11242-019-01289-9.

Russell, S.J., Norvig, P., 2021. Artificial intelligence: a modern approach. In: Pearson Series in Artificial Intelligence, fourth ed. Pearson, Hoboken.

Salina Borello, E., Peter, C., Panini, F., Viberti, D., 2022. Application of A* algorithm for microstructure and transport properties characterization from 3D rock images. Energy 239, 122151. https://doi.org/10.1016/j.energy.2021.122151.

Santos, S.M.G., Gaspar, A.T.F.S., Schiozer, D.J., 2021. Information, robustness, and flexibility to manage uncertainties in petroleum field development. J. Petrol. Sci. Eng. 196, 107562 https://doi.org/10.1016/j.petrol.2020.107562.

Shaked, D., Bruckstein, A.M., 1998. Pruning medial axes. Comput. Vis. Image Understand. 69, 156–169. https://doi.org/10.1006/cviu.1997.0598.

Sheppard, A.P., Sok, R.M., Averdunk, H., 2004. Techniques for image enhancement and segmentation of tomographic images of porous materials. Phys. Stat. Mech. Appl. 339, 145–151. https://doi.org/10.1016/j.physa.2004.03.057.

Singh, M., Mohanty, K.K., 2000. Permeability of spatially correlated porous media. Chem. Eng. Sci. 55, 5393–5403. https://doi.org/10.1016/S0009-2509(00)00157-3.

Tu, J., Yeoh, G.-H., Liu, C., 2018. Some advanced topics in CFD. In: Computational Fluid Dynamics. Elsevier, pp. 369–417. https://doi.org/10.1016/B978-0-08-101127-0.00009-X.

Viberti, D., Cossa, A., Galli, M.T., Pirrone, M., Salina Borello, E., Serazio, C., 2018. A novel approach to a quantitative estimate of permeability from resistivity log measurements. GEAM 155, 17–24.

Viberti, D., Peter, C., Salina Borello, E., Panini, F., 2020. Pore structure characterization through path-finding and Lattice Boltzmann simulation. Adv. Water Resour. 141, 103609 https://doi.org/10.1016/j.advwatres.2020.103609.

Viberti, D., Salina Borello, E., Rocca, V., 2008. Applicability of Newton-based optimization method merged with the Monte Carlo approach to log interpretation. In: Presented at the 11th European Conference on the Mathematics of Oil Recovery, Bergen, Norway. https://doi.org/10.3997/2214-4609.20146440.

Viberti, D., Verga, F., 2012. An approach for the reliable evaluation of the uncertainties associated to petrophysical properties. Math. Geosci. 44, 327–341. https://doi.org/10.1007/s11004-011-9358-1.

Viberti, Da, Verga, F., Galli, M.T., Gossenberg, P., 2007. An effective methodology for evaluation of the uncertainty of petrophysical parameters: application to A real case. In: Presented at the Offshore Mediterranean Conference and Exhibition. OMC-2007-095.

Vidal, D., Ridgway, C., Pianet, G., Schoelkopf, J., Roy, R., Bertrand, F., 2009. Effect of particle size distribution and packing compression on fluid permeability as predicted by lattice-Boltzmann simulations. Comput. Chem. Eng. 33, 256–266. https://doi.org/10.1016/j.compchemeng.2008.09.003.

Wang, C., Wu, K., Scott, G.G., Akisanya, A.R., Gan, Q., Zhou, Y., 2020. A new method for pore structure quantification and pore network extraction from SEM images. Energy Fuel. 34, 82–94. https://doi.org/10.1021/acs.energyfuels.9b02522.

Wang, M., Wang, J., Pan, N., Chen, S., 2007. Mesoscopic predictions of the effective thermal conductivity for microscale random porous media. Phys. Rev. E 75, 036702. https://doi.org/10.1103/PhysRevE.75.036702.

Wu, Y., An, S., Tahmasebi, P., Liu, K., Lin, C., Kamrava, S., Liu, C., Yu, C., Zhang, T., Sun, S., Krevor, S., Niasar, V., 2023. An end-to-end approach to predict physical properties of heterogeneous porous media: coupling deep learning and physics-based features. Fuel 352, 128753. https://doi.org/10.1016/j.fuel.2023.128753.

Wu, Y., Tahmasebi, P., Lin, C., Zahid, M.A., Dong, C., Golab, A.N., Ren, L., 2019. A comprehensive study on geometric, topological and fractal characterizations of pore systems in low-permeability reservoirs based on SEM, MICP, NMR, and X-ray CT experiments. Mar. Petrol. Geol. 103, 12–28. https://doi.org/10.1016/j.marpetgeo.2019.02.003.

Xu, P., Yu, B., 2008. Developing a new form of permeability and Kozeny–Carman constant for homogeneous porous media by means of fractal geometry. Adv. Water Resour. 31, 74–81. https://doi.org/10.1016/j.advwatres.2007.06.003.

Glossary

φ: Porosity (−)
k: Absolute Permeability (m^2)
τ: Tortuosity (−)
r_H: Hydraulic radius (m)
c: Kozeny-Carman coefficient (−)
$f(P_i)$: Cost function on the generic i-th voxel P_i
$g(P_i)$: Distance function calculated from the inlet on the generic i-th voxel P_i
$h(P_i)$: Forward cost estimation on the generic i-th voxel P_i
$I_{x,y,z}$: Number of inlets on a side of the sample
$O_{x,y,z}$: Number of outlets on a side of the sample
t_{max}: Maximal time complexity (s)
L: Length of a sample (vox)
$\mathscr{G}$: Set of points representing the sample
$\mathscr{P}$: Set of points representing the pore structure