MoloVol: an easy-to-use program to calculate various volumes and surface areas of chemical structures and identify cavities

Jasmin B. Maglic,[1] Roy Lavendomme[2]*

[1] Center for Geometrical Engineering of Cellular Systems, Department of Chemistry, University of Copenhagen, Universitetsparken 5, 2100 Copenhagen, Denmark

[2] Center for Ordered Materials, Organometallics and Catalysis (COMOC), Department of Chemistry, Ghent University, Krijgslaan 281-S3, 9000 Ghent, Belgium

* Correspondence e-mail: molovol@outlook.com

Abstract

Cavities are a ubiquitous feature of chemical structures encountered in various fields ranging from supramolecular chemistry to molecular biology. They are involved in the encapsulation, transport, and transformation of guest molecules, thus necessitating a precise and accessible tool for estimating and visualizing their size and shape. MoloVol, a free, user-parametrizable, open-source software, developed for calculating a range of geometric features for both unit cell and isolated structures is presented here. MoloVol utilizes up to two spherical probes to define cavities, surfaces, and volumes. The program was optimized by combining an octree data structure with voxel-partitioned space allowing for even high-resolution protein structure calculations at reasonable timescales. MoloVol comes with a user-friendly graphic interface along with a command line interface for high throughput calculations. It was written in C++ and is available on Windows, macOS, and Linux distributions.

Keywords: software; volume; surface area; cavity; void

1 Introduction

A variety of chemical structures contain enclosed sections that are either partially or completely accessible by smaller molecules. Such enclosures are known as "void spaces" or "cavities" and are found in a multitude of compound classes, such as enzymes, cage compounds and cavitands,[1] as well as porous solid materials.[2] In many cases, cavities are integral to the unique features of these compounds. Cage compounds and cavitands are, for instance, capable to accommodate single guest molecule within their cavities that can then be transformed,[3] transported[4] or sequestered.[5] Porous structures, on the other hand, can act as molecular sponges and accommodate large amounts of guest species.[6] Furthermore, the effectiveness of many catalysts is tied intimately to their porosity.[7]

One of the most, if not the most, important criterion to determine whether one or several guests can enter a cavity is the geometrical match between the cavity and the guest molecule. There is a plethora of computer programs that aid in calculating geometrical features of cavities such as their volume or surface area: PLATON with its "CALC VOID" and "CALC SOLV" commands,[8] VOIDOO,[9] GRASP,[10] CAST,[11] HOLLOW,[12] Volarea,[13] McVOL,[14] MDpocket,[15] ProteinVolume,[16] 3V,[17] Voronoia,[18] PoreBlazer,[19] Zeo++,[20] PyMOL,[21] Mercury,[22] Materials Studio,[23] and the ironically named "Another Void Program" or AVP.[24] This list is not exhaustive and yet, despite this wide choice of applications, each with its own uses and perks, we could not find an application meeting the following characteristics: (i) allows for freely parametrizable calculations, (ii) exports results in an easily visualizable format, (iii) provides all desired types of volumes and surfaces (*e.g.*, van der Waals, probe accessible and probe excluded surfaces), (iv) is able to analyze both unit cells and isolated molecules, and (v) operates through an easy-to-use, intuitive graphic user interface.

Here we present MoloVol, a free, user-friendly application available on all major operating systems, developed to meet the above list of features and more.

2 Accessibility and availability

MoloVol is an open-source application written in C++. The entire source code is available on GitHub and is free to use and modify under the MIT license. The user interface is based on the cross-platform GUI library wxWidgets, but the application is also fully functional from the command line. MoloVol is being actively developed and has been tested on Windows 10, macOS (10.14 and above) and Ubuntu 20.04 LTS. Additionally, we offer installation packages for Windows 7 and 8, macOS 10.11 and above, and Debian. On macOS, we provide native support for both Apple silicon and Intel processors. Current and past release are available as precompiled binaries or as source code at https://molovol.com. Future releases will be made available through the same webpage.

3 Overview of features

3.1 Method

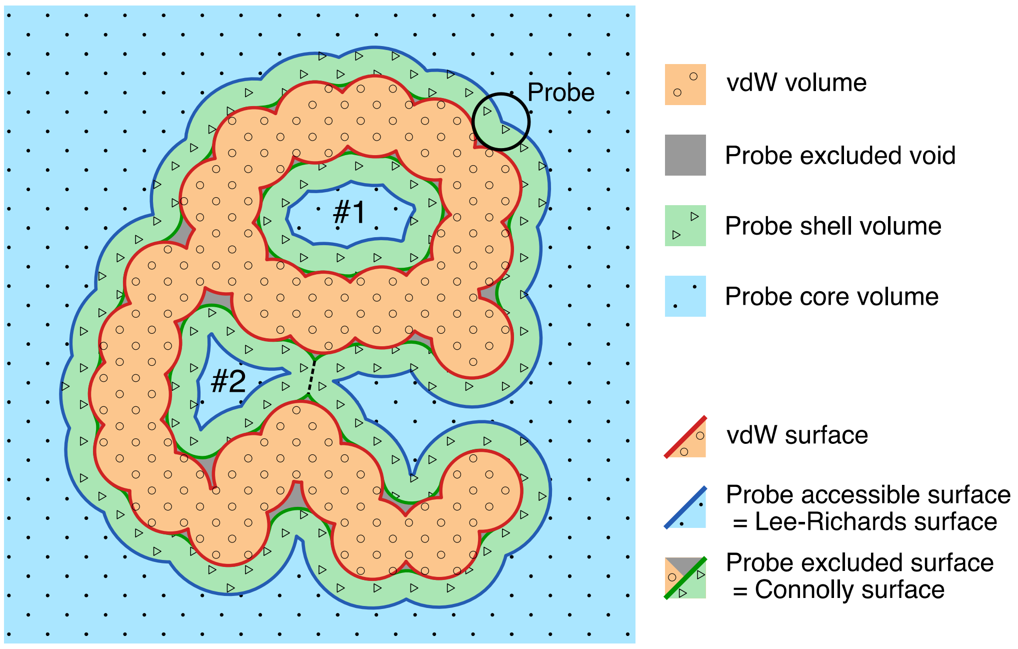

Figure 1. 2D analog of a potential structure analysis with MoloVol. Types of volumes and surfaces obtained from single-probe mode. Both cavities #1 and #2 are isolated. The Connolly surface for the outside space is the molecular open surface. Dashed line: Separation between cavity #2 and outside space.

3.1.1 Volume definitions in single-probe mode

MoloVol can determine different types of volumes that may be defined for a chemical structure. The simplest volume is the van der Waals (vdW) volume, where each atom is assumed to occupy a spherical volume (defined by its vdW radius) with its center at the atom position. The vdW volume includes any space that is occupied by atoms (orange section in Fig. 1). MoloVol is also able to distinguish between "probe accessible void" (blue/green section in Fig. 1) and "probe excluded void" volume (gray section in Fig. 1) by using a spherical probe. This "single-probe mode" again assumes spherical atoms and treats them as impenetrable by the probe. Within probe accessible void volume, MoloVol differentiates between "probe core" volume, *i.e.*, sections of space reachable by the probe center (blue section in Fig. 1), and "probe shell" volume, *i.e.*, sections reachable by the probe but not by its center (green section in Fig. 1). Cavities are defined by spatially isolated core volume sections. For a given cavity, its total volume is the sum of the isolated core volume and all connected shell volume within one probe radius distance. However, when two cavities are close enough, they may compete for the same shell volume. In this case, shell volume is attributed to the closest core volume, so that no shell volume is counted twice (dashed line in Fig. 1).

3.1.2 Surfaces and rolling probe

In addition to volumes, MoloVol can be used to analyze various surfaces. The vdW surface is defined as the surface enclosing the vdW volume (red line in Fig. 1). Two more values are obtained using a spherical probe, where the probe may be imagined as "rolling" over the entire structure while recording the position of the probe center and outer bound. This method was originally described by Lee and Richards[25,26] then further developed by Connolly.[27] The "probe excluded surface" is the surface traced by the probe's outer bound and encloses the vdW volume and probe excluded void (green line in Fig. 1). The "probe accessible surface" is traced by the probe's core and additionally encloses the probe shell volume (blue line in Fig. 1). These surfaces are often referred to as "solvent accessible surface" and "solvent excluded surface" when the probe approximates a solvent molecule.

3.1.3 Blocking open cavities in two-probe mode

A cavity is considered "closed" or "isolated" when a probe can neither enter nor exit, *i.e.*, all entrances are smaller than the probe diameter. In proteins, isolated cavities may be considered part of the molecular volume because guest molecules do not have access to the enclosed space unless access is given through conformational motion. Using a spherical probe as described in section 3.1.1, it is straightforward to identify isolated cavities. Open cavities that are connected to the outside such as protein tunnels, cavities of open cage compounds, and deep cavitands are, however, more difficult to analyze because there is no physical limit between the inside and outside of the molecule. Several methods have been used to define the volumes of these open cavities such as (i) calculating the largest sphere[28,29] or polyhedron[30] that fits inside the cavity, (ii) using a larger spherical probe that cannot exit the cavity,[31] (iii) arbitrarily blocking the entrances,[32] (iv) generating the convex hull of the molecule as the outside limit,[33] and (v) using the probe algorithm with a probe too large to reside in the molecule to thereby define the "outside".[17] All these options involve arbitrary decisions and there is no absolute best solution. The first two methods artificially reduce the cavity volume by neglecting asperity within the molecule. The third method appears the most arbitrary and may lead to the least reproducible results if details on the blockage are not provided, but it does account for asperity. The fourth method may create undesired supplementary cavities if the original structure has large concave surface regions. For MoloVol, we have opted for the fifth method using two probes: one large probe to define the outside space and one small probe to define inside cavities (see Fig. 2). In our opinion, this solution is the most elegant, as it allows defining the shape of the cavities clearly and in an easily reproducible manner using a minimal set of parameters. There are, however, some limitations: (i) in the case of very large open cavities such as for giant polyhedral structures,[34] the large probe used to define the outside space must be extremely large thereby missing the fine details of the outer shape of the molecule; (ii) the separation between the outside and inside space as desired by a user might not be obtainable for some intricate structures (*e.g.*, in the case of protuberant spikes surrounding the entrance of cavities). It is of note that methods (ii) and (iii) can also be realized in MoloVol, if preferred by the user, by changing the probe radius or loading a modified closed structure.

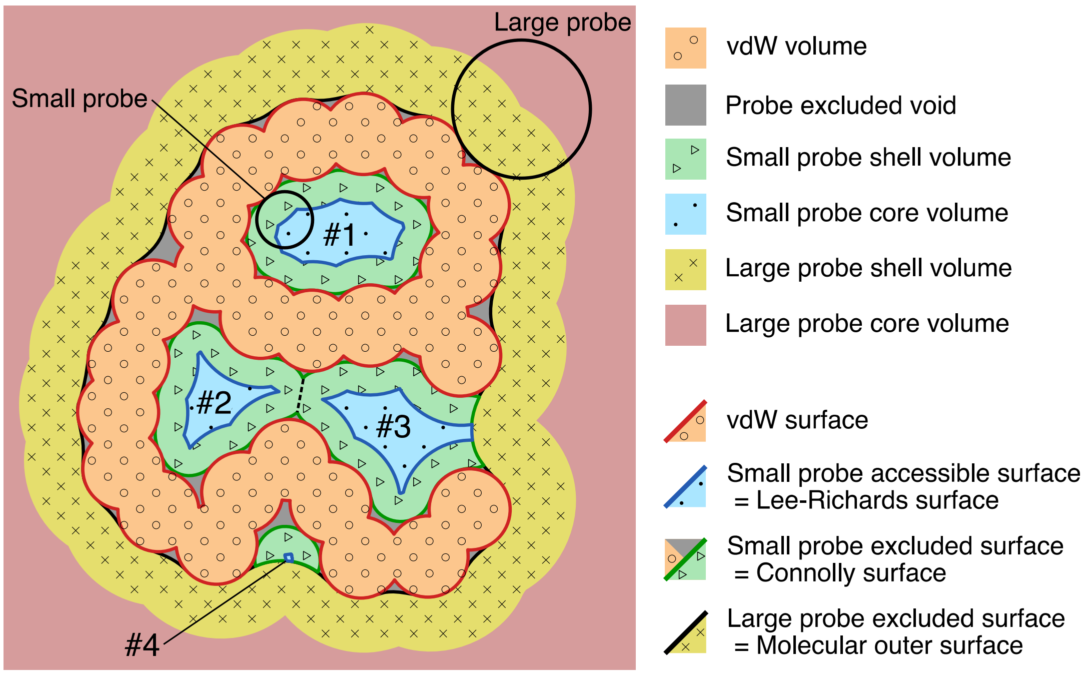

Figure 2. 2D analog of a potential structure analysis with MoloVol. Types of volumes and surfaces analyzed in MoloVol using two-probe mode to detect internal cavities. Detected cavities are numbered. Dashed line: Separation between the two touching but separate cavities #2 and #3. #3 and #4 are pockets that appear due to the large probe.

Due to the translational symmetry in crystal structures, a molecule and its cavities may be translated arbitrarily within the unit cell and thereby may appear split by its edge. Thus, "outside" and "inside" are not as easily defined within a crystal unit cell. Nonetheless, the two-probe mode in MoloVol can still be of use in crystal structure analysis. For example, using two probes allows to differentiate pores of different sizes in porous materials: a large probe to explore the mesopores and a small probe to explore the micropores. The two-probe mode implemented in MoloVol thus has a polyvalent use depending on the type of structure analyzed and the information desired. Further development for porous crystal characterization is planned (see section 5).

3.2 Calculation output

As described in 3.1 and as seen in Fig. 1 and 2, several volumes and surfaces can be defined for a structure. This section details the types of volumes and surface areas that can be obtained from a MoloVol calculation. An example calculation performed on an open cage compound[32] with the corresponding main volumes and surface areas is shown in Fig. 3.

3.2.1 Volumes

The default calculation of MoloVol provides the vdW (V_{vdw}), probe excluded void (V_{void}), probe core or accessible (V_{core}), and probe shell (V_{shell}) volumes. Additionally, combinations of these volumes are also calculated: molecular volume ($V_{mol} = V_{vdw} + V_{void}$), probe occupied void volume ($V_{occ} = V_{core} + V_{shell}$), and molecular volume with isolated cavities ($V_{mol\text{-}isolated} = V_{mol} + V_{occ\text{-}isolated\text{-}cavities}$) defined as the volume that cannot be accessed from the outside. Each cavity is characterized by an individual V_{core} and V_{occ} and a cavity type based on the number of entrances (only in non-unit cell mode). In single-probe mode, MoloVol only differentiates between "Outside" and "Isolated" cavities, where the former is the probe core that reaches the boundaries of the analyzed space, and the latter is every remaining cavity. In two-probe mode the cavity type is determined based on the number of entrances to the cavity from the outside defined by the large probe. It may be "Isolated" for none, "Pocket" for one, or "Tunnel" for two or more entrances. All volume values are provided both on the molecular scale in cubic Ångström and on the macroscopic scale in cubic centimeter per gram, which may be useful for comparison with experimental pore volumes in porous materials.[35]

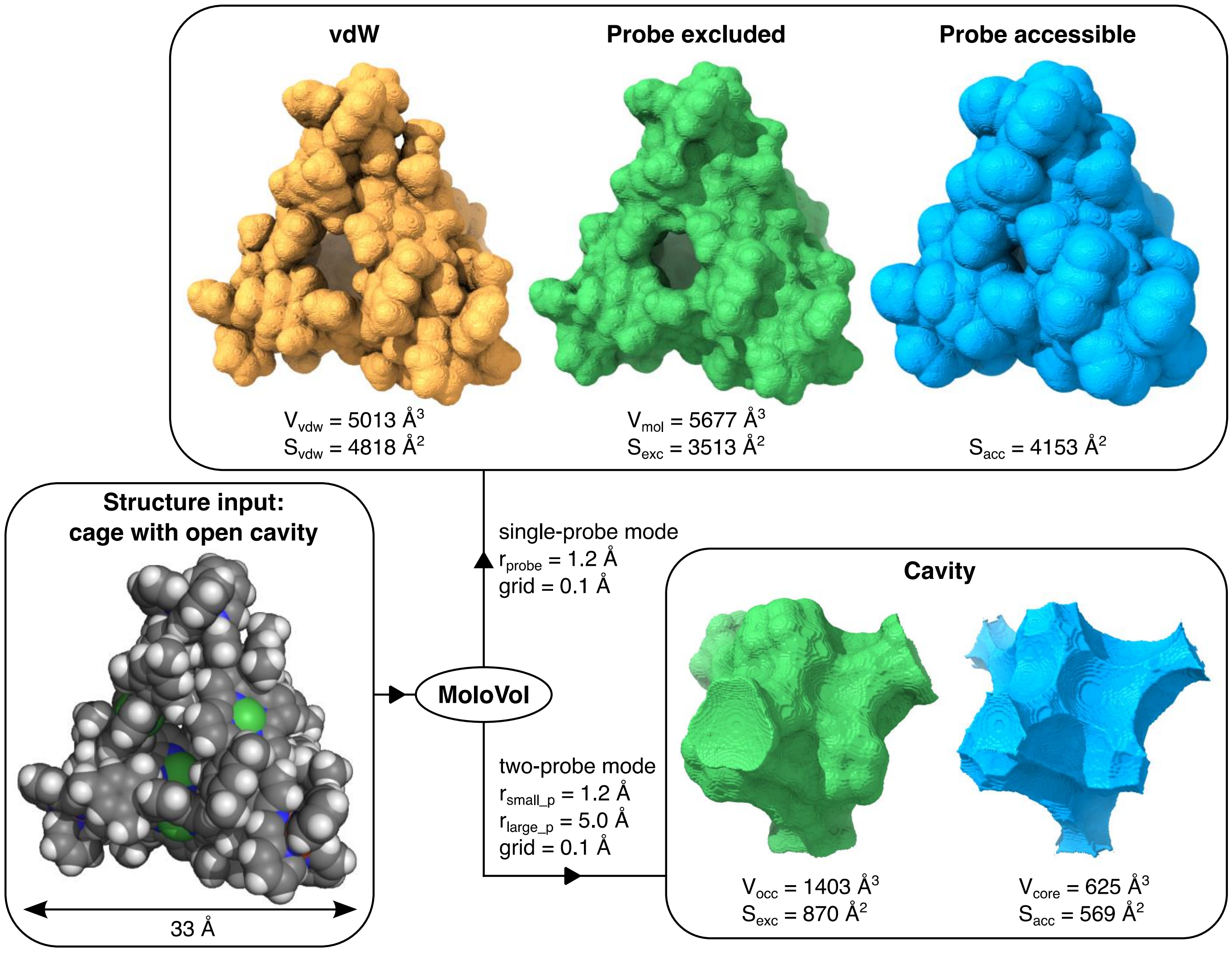

Figure 3. Example of the output from MoloVol calculations performed on an open cage compound. The resulting surfaces were rendered in ChimeraX based on the isosurface maps generated by MoloVol. "grid" refers to the grid resolution; subscripts "small_p" and "large_p" correspond to the small and large probe, respectively.

3.2.2 Surface areas

In addition to volumes, surface areas can be calculated if the corresponding option is enabled. MoloVol provides the vdW (S_{vdw}), probe excluded (S_{exc}), and probe accessible (S_{acc}) surface areas. In single-probe mode, the molecular surface area may either be considered equivalent to the total probe excluded surface (S_{mol} = S_{exc}), which is redundant and therefore not explicitly given, or the surface area that is only open to the outside ($S_{mol\text{-}open}$ = $S_{exc\text{-}outside}$) and encloses $V_{mol\text{-}isolated}$. In two-probe mode, S_{mol} is defined by both probes excluded surface but the value obtained in single-probe mode should be preferred as it better represents the asperities defined by the small probe. For each cavity, individual values for S_{exc} and S_{acc} are also given. These values are provided both on the molecular scale in square Ångström and on the macroscopic scale in square meter per gram for comparison to experimental pore surface areas in porous materials.[35]

3.2.3 Surface maps

It is often important to visualize the surfaces belonging to the calculated volumes, to confirm by shape that the values calculated correspond to the desired volumetric objects and to share visual information. MoloVol does not yet provide an in-built viewer for three dimensional models, but it is possible to export isosurface maps in the OpenDX format that can be opened in popular and powerful rendering programs for chemistry such as PyMOL,[21] Chimera,[36] and ChimeraX[37]. A detailed guide on how to display the surfaces in these programs is provided in MoloVol's user manual. The user can export both the isosurface map of the total structure and separate isosurface maps for each cavity. The map file contains all information needed to display each surface and volume type calculated by MoloVol.

3.3 Input files and parameters

3.3.1 Structure files

Chemical structures can be loaded from XYZ, PDB, and CIF files. XYZ files contain only a list of atoms with their symbol and cartesian coordinates in Ångström. They are often most convenient for chemical structures with no crystallographic information. PDB and CIF files may contain more information, notably crystallographic information. Consequently, crystal unit cell analysis in MoloVol requires either PDB or CIF files.

3.3.2 Element file

A file containing element radii and atomic weights is provided with the program binaries. Element radii were taken from a study by Alvarez because the list was more extensive than other references we found.[38] Atomic weights were taken from a IUPAC technical report.[39] These values might slightly diverge from the ones used in other programs. This file with element properties is loaded by default but it is possible to select a different file generated by the user with other values or custom elements if needed.

3.3.3 Calculation parameters

After importing a structure file, all atoms are listed along with their radii in a table. The radii are read from a provided element file (see section 3.3.2). It is possible to change the radii from the default in two ways: either by changing the values in the element table directly or by loading a custom element file. Using the former method, the radii will be reset when loading a new structure. The latter method may be used if changes should apply to multiple calculations. Furthermore, loading a custom element file allows defining custom element symbols. We do not expect this function to be heavily used but it might be handy in certain cases. Potential applications may be, for instance, setting different radii for aromatic and aliphatic carbons, for different oxidation states of the same element, or to increase the size of carbon atoms to compensate for missing hydrogen atoms in the structure as it is a common practice for large biomacromolecules. Custom element symbols may only contain alphabetic characters.

The following options can be toggled easily through tick boxes in the user interface: (i) include HETATM lines in PDB files (*i.e.*, atoms not belonging to the biomolecule, such as solvent), (ii) analyze the crystal unit cell, (iii) calculate surface areas (by default, only volumes are calculated), and (iv) switch between single- and two-probe mode.

The probe radii can be modified to meet the user's needs. Common probe radii (*e.g.*, hydrogen atom, water molecule, argon atom) are accessible via a dropdown menu but any value can be entered manually. Increasing the probe radii leads to an increase in calculation time (see section 4 for the algorithmic complexity).

MoloVol analyzes space by partitioning it into discrete cubes or voxels (see details in section 4). Each voxel is placed on a cubic grid. The grid resolution is defined as the voxel side length and can be changed. Smaller grid resolution value will increase the accuracy of the calculated values but also increase the calculation time (see section 4 for the algorithmic complexity).

Finally, a parameter named "optimization depth" can be chosen. This parameter is linked to the speed of the calculation and does not influence the calculated values. This parameter does not typically need to be modified by users for common calculations but is nonetheless available to give a full control on calculation parameters if desired. The effect of this parameter is detailed in section 4.5.1.

3.3.4 Command line interface

MoloVol was designed with a user-friendly graphic interface. However, providing only a graphic user interface limits the throughput of calculations and makes it difficult to communicate with other applications. To avoid this, MoloVol provides a fully functional command line interface (CLI), usable with the Windows command prompt or Unix terminal. This allows automating calculations and thereby increasing the throughput. A guide on how to use the CLI is provided in the user manual of MoloVol.

4 Algorithms

4.1 General

MoloVol analyzes the section of space containing a chemical structure by partitioning it in discrete sections. The space is divided into a 3D grid of small cubes (*i.e.*, voxels). With respect to computational cost, this voxel approach is more scalable than analytical calculations. Moreover, once each voxel is identified, a variety of physical values, such as volume and surface types, can be derived with minimal supplementary computational time. A voxelated space also allows tuning the grid resolution (*i.e.*, the side length of the voxels) to find a compromise between accuracy and calculation time. With this approach, the calculation time increases with the inverse cube of the grid resolution g (*i.e.*, complexity $O(g^{-3})$), as the number of voxels increase in each spatial dimension. However, we employ optimizations to reduce average complexity to sub-cubic and achieve decent calculation times even at a high resolution (*e.g.*, 0.1 Å, corresponding to 1000 voxels per Å^3) for structures as large as proteins. For clarity, voxels are presented as pixels in subsequent figures, but the text may refer to them as voxels.

4.2 Determining voxel types

Voxels are evaluated one by one to determine whether they belong to an atom, the probe core, the probe shell, or the probe excluded void volume. In single-probe mode, two algorithms loop over the voxels: the first one determines whether a voxel is of atom type, probe core type, or cannot yet be identified; the second algorithm goes over the remaining unidentified voxels to determine whether they are of probe shell or excluded void type. The algorithms performed in both loops are described in sections 4.2.1 and 4.2.2. In two-probe mode, there are four loops to evaluate (i) atom and large probe core types, (ii) large probe shell type, (iii) small probe core type, and (iv) small probe shell and excluded void types. Despite the additional loops, the same two algorithms as for the single-probe mode are used.

4.2.1 Atom and probe core types

For each voxel, its distance to each nearby atom is calculated. If, for any atom, the distance is smaller than the atom radius r_{atom} then the voxel is *de facto* inside an atom and set to atom type (orange in Fig. 4). Otherwise, if, for any atom, the distance d is smaller than r_{atom} + r_{probe} then the voxel is either of probe excluded void or probe shell type and remains undefined at this stage. If, after going through all nearby atoms, no distance matches these conditions then the voxel is assigned probe core type (light blue in Fig. 4). Nearby atoms are found using a *k*-d tree (see section 4.5.2) to reduce average algorithmic complexity from $O(m \times n)$ to $O(m \times \log n)$, where n is the number of atoms and m is the number of voxels.

This is the only algorithm where the distance between voxels and atoms needs to be calculated.

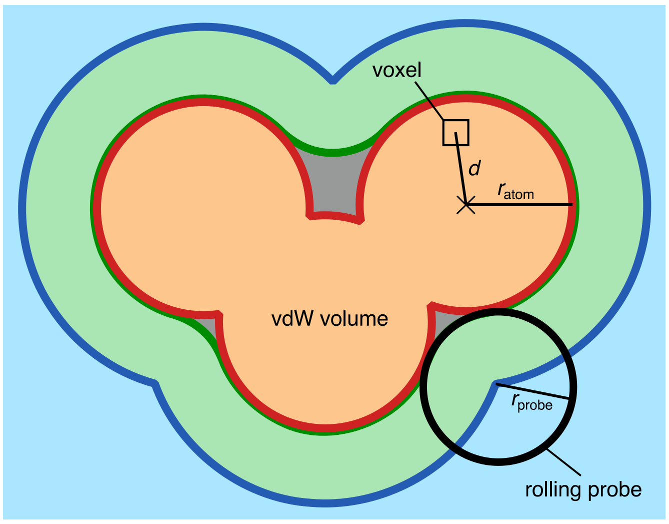

Figure 4. Necessary parameters and measures for determining voxel types in MoloVol algorithms. Voxel types: atom is orange, probe core is light blue, probe shell is light green, probe excluded void is gray.

4.2.2 Probe shell and excluded void

For all voxels that remain unassigned after the first algorithm, a second algorithm determines whether the voxel is of probe shell type (light green in Fig. 4), or probe excluded void type (gray in Fig. 4) by evaluating its relationship to surrounding voxels. If the voxel has any probe core neighbor within a spherical distance r_{search}, then it is set to probe shell type (Fig. 5a). Otherwise, the voxel is set to probe excluded void type (Fig. 5b).

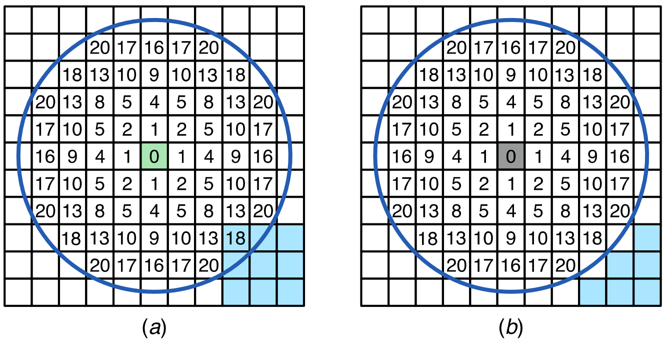

Figure 5. 2D analog of neighbor search algorithm. Neighbor voxels are evaluated with increasing distance to the central voxel within the search radius r_{search} (blue circle). The square Euclidian distances between central voxel and neighbor voxel are given in voxel units. (*a*) A core type voxel (light blue) is located within the search radius; therefore, the central voxel is assigned the shell volume type (light green). (*b*) There are no core type voxels within the search radius; therefore, the central voxel is assigned the probe excluded void type (gray).

Beginning from a central voxel, whose type is yet to be determined, its neighbor voxels are evaluated in order of increasing distance. In preparation for this, the program first generates a list of relative neighbor indices sorted by the distance from the center. Absolute neighbor voxel indices are obtained through vector addition of a relative index to the center voxel index. For each unassigned voxel, the neighbors are evaluated up to the limit distance of the r_{search} value. The search continues until a probe core voxel is found or all relevant neighbors have been evaluated. This procedure avoids (i) calculating voxel distances, which would have a significant computational cost, (ii) ensures that only voxels in the useful range are checked, and (iii) allows associating the newly assigned probe shell voxel with its nearest probe core voxel, which is used for assigning shell type voxels to cavities as explained in section 4.3.

Due to the discrete nature of the voxelated space, the distances between voxels are not continuous. This introduces spatial anisotropy with respect to the voxel grid. This is demonstrated in Fig. 5, as the furthest square Euclidian distance from the center is 16 horizontally but 18 diagonally. Another related issue stems from a propagated error in the first evaluation loop. When representing a solid in voxelated space, the voxels at the solid's boundary will virtually always be slightly displaced. After atom and core type voxels are assigned, this displacement on both sides can lead to the gap containing unidentified voxels to be significantly larger than the probe radius. Since the neighbor search algorithm starts from a probe core type voxel and not from the actual probe accessible surface (Fig. 6), it is necessary to increase the search radius, such that $r_{\text{search}} = (r_{probe} + \delta)$. We found that $\delta = 0$ led to false negative checks, *i.e.*, voxels that should be probe shell type were assigned probe excluded void type (Fig. 6b). This was tested by analyzing single atom structures in which no probe excluded void volume should be present. The ideal value of δ that eliminates false negatives entirely but does not introduce false positives depends on the grid resolution and r_{probe}. We found no analytical value for δ. However, empirical tests with parameters expected to be used for chemical structure analysis (*i.e.*, grid resolution in the range 0.05–1.0 Å and r_{probe} in the range 0–10 Å) have shown that $\delta = (\sqrt{2})/4$, given in units of voxel side length, yielded best results.

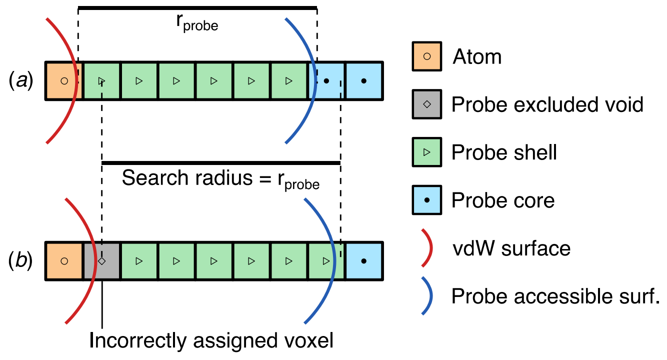

Figure 6. 1D demonstration of how shifting the underlying voxel grid relative to the atom positions can impact the voxel types with the neighbor search algorithm. (*a*) Example of voxel types being assigned as expected when r_{search} is equal to r_{probe}. (*b*) A false negative situation leading to an incorrectly assigned voxel using the same r_{search} and r_{probe} as before.

This part of the algorithm is typically the most time-consuming in a MoloVol calculation. The complexity is $O(m \times s^3)$ with s being the ratio between r_{probe} and grid resolution and m being the number of unidentified voxels, which itself depends on r_{probe}. Calculations performed with a small probe radius run extremely fast at this stage and other parts become more time-consuming.

4.3 Identifying separate cavities

One important feature of the application is to identify separate cavities and calculate their corresponding volumes and surface areas. Separate cavities are defined by portions of space between which a probe cannot travel. Thus, to identify each cavity, a flood fill algorithm is applied to voxels with probe core type.

The flood fill algorithm evaluates all 26 neighbors to a voxel that share at least a vertex. Conducting the algorithm with only the 6 direct neighbors led to incorrectly isolated cavities. Indeed, the discrete voxelated space can lead to small separate islands of voxels of probe core type for cavities with sharp geometrical features that would be connected in a continuous space. This flood fill algorithm is performed in-between the steps outlined in 4.2.1. and 4.2.2. Thus, when probe shell voxels are identified by finding their nearest probe core voxel neighbor, they are directly assigned to the same cavity. Consequently, separate cavities whose shell volumes overlap are properly segmented at equidistance of their probe core region (see cavities #2 and #3 in Fig. 2).

4.4 Calculating volumes and surface areas

4.4.1 Volumes

Calculating volumes is trivial once voxel types and separate cavities are identified. Each type of volume is simply calculated by summing all corresponding voxels and multiplying the tally by the volume of a single voxel.

4.4.2 Surface areas

Within a voxelated space, calculating surface areas is complicated by the fact that surfaces not aligned with the voxel grid will show rough steps instead of being smooth. If the surface area was calculated by simply adding the surface area of the voxels, then a voxelated sphere, for instance, would have a surface area approximating $6\pi r^2$ instead of $4\pi r^2$, similarly to how a pixelated circle's perimeter approximates 8r instead of $2\pi r$. Fortunately, algorithms were developed to extrapolate surface area data from voxelated objects. The marching cube algorithm was initially developed to display smoother surfaces from voxelated dataset without consideration for the surface area and was since then shown to be usable to extrapolate surface areas.[40] Briefly, the marching cube algorithm analyzes groups of eight voxels in a cubic arrangement. Each voxel in the octuplet is given a binary value depending on which side of the surface it is located on, and all values are stored in a byte. With respect to symmetry, there are 14 unique configurations out of the 256 possible bit combinations. Each configuration is attributed a preset surface area parameter. Finally, the surface areas of all octuplets are summed to obtain the total surface area. We implemented surface area calculation using this marching cube algorithm in MoloVol with the most recent optimal surface area parameters reported by Lindbald.[40] The vdW surface areas calculated with MoloVol at a resolution of 0.1 Å were very similar to analytical values for simple systems (< 0.3 % error for acetylene) and calculated values with other programs for large systems (1.6 % difference compared to PyMOL result calculated with same element radii as MoloVol for a cytochrome C complex containing 1356 atoms, PDB 6S8Y [41]). Worse results were obtained with other sets of parameters for the marching cube algorithm, notably a previous set described by Lindbald.[42] These results demonstrate that surface areas can be extrapolated from voxelated chemical structures with reasonable accuracy using a marching cube algorithm with the right set of parameters.

4.5 Optimizations

One important aspect in MoloVol development was to ensure that calculations would be completed in a reasonable time even for large molecules, like proteins, and at high resolution, like 0.1 Å.

4.5.1 Application of the octree data structure

Calculation time in voxel-based systems typically scales cubically with the number of voxels and thereby inverse cubically with the grid resolution. To achieve sub-cubic scaling, we adapt an octree approach. An octree is a data structure in which every node contains either eight or no children. In MoloVol, each voxel is an octree and may contain eight voxel children. The children have half the side length of their parents and are stacked in a 2x2x2 grid within their respective parent. For the optimization, the largest, top-level voxels are evaluated first. If their type is successfully evaluated, then the types of all descendants are also set. Only when the top-level voxel is a limit case, for instance by being located at the edge of an atom, then the voxel is formally divided into eight child voxels that are subsequently analyzed. The process is repeated until the lowest level of voxels is reached. At the bottom level, voxels are the smallest unit of space and are treated as 0-dimensional points, so that no limit cases can occur.

Because of this approach, only voxels near surfaces are analyzed using smaller voxels whereas voxels in uniform portions of the space are analyzed using larger voxels. For instance, Fig. 7 shows a 2D slice of a three-atom structure evaluated using a voxelated octree with a total of four levels. In the slice shown, we can count a total of 436 voxels (from top to bottom levels: 24, 76, 144 and 292) which is over three times smaller than the 1536 total number of voxels needed for this slice without the octree structure. The difference between these numbers is even more stunning when considering the full three-dimensional space. Notably, further increasing the number of levels is counterproductive as it would, albeit slightly, increase the number of voxels to 442 for this slice with five levels. Therefore, there is an optimal value for a given calculation. Considering the typical calculation parameters and various chemical structures, we found that the optimal optimization depth is generally in the range 2–5 for different calculations. Thus, we set the default value to 4 in MoloVol with the possibility to change this parameter if needed. Test results are given in section 4.6).

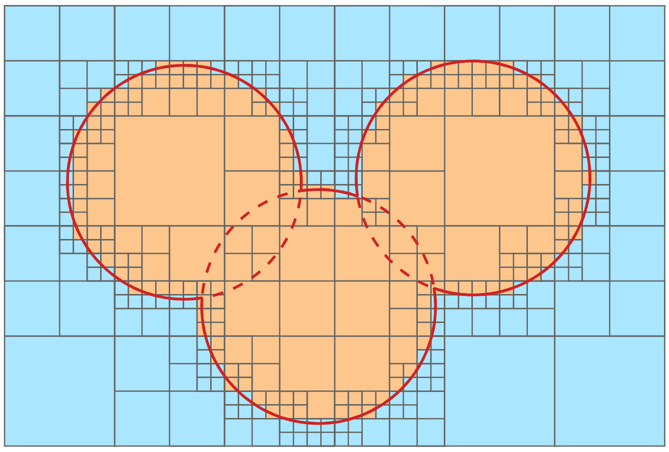

Figure 7. Slice of a voxelated space with octree development containing three atoms. The sizes of atoms and voxels correspond to a realistic MoloVol calculation with the following parameters: atom radius = 1.7 Å; probe radius = 0 Å; grid resolution = 0.2 Å; optimization depth = 3 (*i.e.*, four levels in the octree including level 0).

The algorithms to determine voxel types presented in section 4.2 were adapted to be compatible with this octree optimization. Importantly, it was ensured that the results obtained from the program were invariant with respect to the optimization depth parameter. In most cases, the octree is simply exploited by conducting all computations at the highest possible octree level, only descending the tree when a limit case is identified. In the case of the flood fill algorithm used in the cavity identification, it is necessary to evaluate neighbor voxels on varying octree levels. For this, we implemented a function that returns a list of all relevant neighbor voxels on any level.

4.5.2 Optimization of atom search using *k*-d tree

To accelerate the first algorithm in the voxel type evaluation step, a *k*-d tree was used. The *k*-d tree is a well-established data structure that can be used to store a set of spatially separated points and may be used to reduce the complexity of neighbor searches.[43,44] Here, the *k*-d tree is used to accelerate the search of atoms that may affect the type of a given voxel.

A list of atoms is obtained from the user via an input file. Each atom has a position in 3D space, consisting of three cartesian coordinates (x, y, and z) as well as a radius. To construct the *k*-d tree, the set of all atoms is first sorted along one of the coordinates, *e.g.*, x. The middle element of the sorted list (rounded down) is stored in the tree's root node. Next, this procedure is repeated with the two sets of atoms left and right from the previous middle element in the sorted list; however, in the second step the atoms are sorted along the y coordinate. The middle elements of each list become the left and right children of the root node. This algorithm is repeated recursively, cycling through the coordinates, until all atoms have been placed in a node, resulting in a binary tree. If an atom list is empty, then the appropriate node is assigned an empty reference (NULL).

The strength of this data structure is demonstrated best with an example. In MoloVol, we may want to determine whether a voxel is inside any atom to determine its type. Computing the voxel-atom distance for every atom scales with $O(n)$. Using the tree, we begin by computing the distance between the voxel and the root node atom along the x coordinate. If the distance exceeds the largest atom radius in the tree, we can not only dismiss the node atom but one entire branch of the tree. In the best case, the comparison may reduce the number of remaining candidate atoms by a factor of 2 for each tree level, resulting in $O(\log n)$. If the distance is not larger than the largest atom radius, then the 3D voxel-atom distance must be calculated for the node atom and the search may continue along both branches.

4.6 Benchmark Testing

Benchmark tests presented in this section were performed on a 13-inch MacBook Pro from Apple, built in 2020 (uniquely identified as MacBookPro17,1). The central processing unit (CPU) is an ARM-based Apple M1 and contains an in-built graphics processing unit. The computer provides 8 GB of random access memory (RAM). The latest development version available at time of writing (equivalent to MoloVol v1.0.0) was used. Platon[8] was downloaded and compiled on September 3rd, 2021. To accurately compare different programs, the user time output from the "time"-command within the bash shell was used. The value represents the CPU time spent in user mode and is not affected by idling.

Calculations on a structure file containing a single hydrogen atom (radius 1.2 Å) were performed at various grid resolution values to demonstrate the convergence behavior of the vdW volume and vdW surface results. Analytical values for volume and surface can be trivially obtained due to the geometric simplicity of the single-atom structure. Fig. 8 displays the relative error *versus* the grid resolution for both vdW volume (Fig. 8a) and vdW surface (Fig. 8b). As expected, results deviate strongly and unpredictably from the analytical values at large grid resolution values, because the voxels are larger. As the grid resolution decreases so do the voxel sizes and the obtained results soon converge towards the expected values. We have found that good results were generally obtained at below 0.2 Å grid resolution. As such, the default grid resolution in MoloVol is set to 0.2 Å.

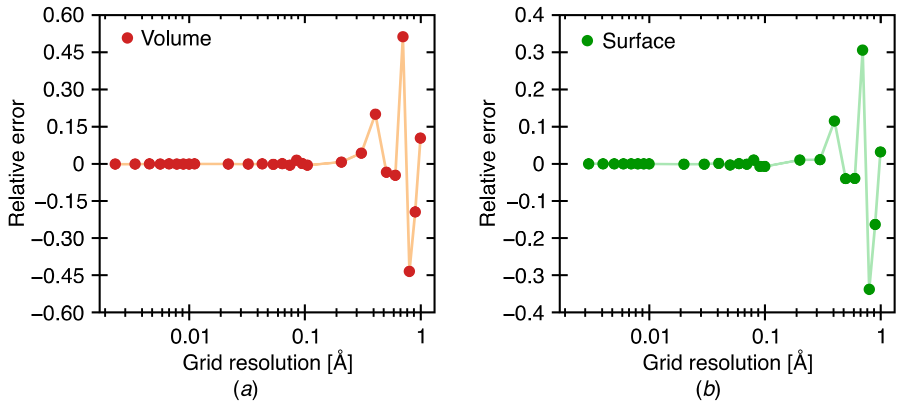

Figure 8. (a) Van der Waals volume and (b) van der Waals surface calculations at different resolutions for a single hydrogen atom (radius 1.2 Å). Connecting lines have been added for visual clarity. Due to the simplicity of the structure, analytical values can be easily calculated. As the grid resolution decreases so does the voxel size and the results converge towards the expected values.

A range of diverse calculations were conducted to find whether an optimal value for the octree depth (see section 4.5.1) could be determined. To test for unexpected effects of the optimization, different calculation parameters and options were varied, such as the number of atoms, addition of surface calculations, unit cell mode, and two-probe mode. Fig. 9 shows the results for three structures of different atom counts, with acetylene at 4, fullerene at 60, and a protein complex at 1356 (PDB 6S8Y). Each calculation set displays a sharp drop going from 0 to 1 octree depth, as this is equivalent to enabling the optimization. Furthermore, in all tests the optimal depth was found to be between 2 and 5. At higher depths the calculation time increases drastically as unnecessary voxels appear. Based on this, we have chosen the default octree depth to be 4 for MoloVol.

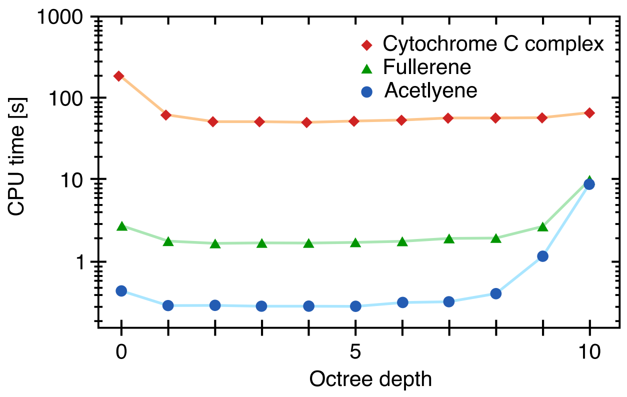

Figure 9. Calculation times for structures at varying octree depths. Connecting lines have been added for visual clarity. The structures were chosen to cover a wide range of atom amounts, with acetylene at 4 (blue dots), fullerene at 60 (green triangles), and a protein complex at 1356 (red diamonds, PDB 6S8Y).

Performance comparisons with Platon were conducted to give an impression of the performance improvement that MoloVol provides. Equivalent calculations were conducted both with MoloVol and Platon and the calculation times were compared. For this, a structure file containing a fullerene (C_{60}) molecule was analyzed in unit cell mode with a carbon radius of 1.77 Å at different grid resolutions. The vdW radius of carbon was adjusted using the SET VDWR command and the grid resolution was set using the GRID command, prior to using Platon's CALC SOLV routine to compute the cavity volumes. In MoloVol, the octree depth was left at the default value of 4. Both MoloVol and Platon yielded similar results and the differences were attributed to their different algorithms. The results of the comparison are shown in Fig. 10. We found that MoloVol performs consistently faster than Platon while providing more volumetric information. The direct comparison at grid resolution 0.1 Å shows that MoloVol performs more than 10 time faster and calculations using Platon at higher grid resolutions did not finish within at least 15 minutes.

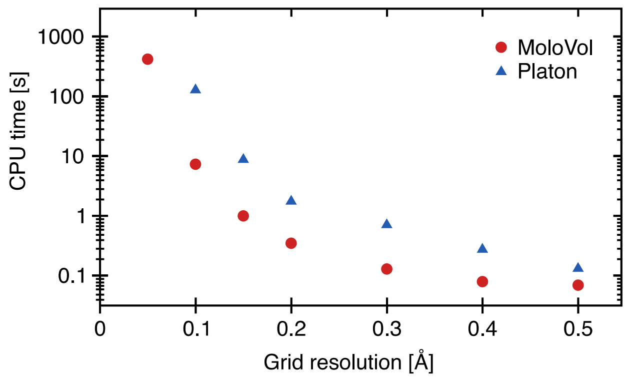

Figure 10. Comparison in performance between MoloVol and Platon with the CALC SOLV routine. Cavity calculations were conducted in both applications using the same input parameters when applicable (*i.e.*, atomic radii, probe radius, and grid resolution). MoloVol optimization depth was set to 4.

5 Limitations and planned features

MoloVol treats atoms as static, impenetrable spheres. This is a common and convenient model but does not represent the complex reality of atomic interactions and molecular motion. As such, the values calculated with MoloVol can be informative and help describe and compare molecular systems but should by no means be considered absolute limits to, for instance, what guest can or cannot reside in a cavity.

In its current state, MoloVol can be used to identify, locate, and analyze cavities within porous crystalline materials. Yet, for crystal structure analysis a user may need to manually distinguish continuous pores from isolated cavities.[35] We plan to include a feature that identifies pores automatically in a future version.

MoloVol can currently provide a list of cavities with their volumes, surface areas, center position, and cavity type, based on the number of openings in two-probe mode. We plan for MoloVol to indicate connecting and neighboring cavities and present a network of cavities in the report to provide more detailed spatial information. This feature is meant to help users identify useful cavities more easily.

Considering the anisotropy of a voxelated grid, rotating or translating the input chemical structure may lead to slightly different calculated volumes and surface areas. We are considering adding a feature to randomize the structure's orientation and provide an average result from multiple calculations. This reduces anisotropy and may increase accuracy.

Visualizing surfaces currently requires the use of third-party software. Adding the ability to visualize the calculated surfaces directly within the MoloVol user interface is under consideration.

Despite the algorithmic optimizations we have implemented in MoloVol that ensure fast calculations, there remains a clear optimization strategy that has yet to be exploited. It is possible to further accelerate calculations by parallelizing the program instructions, using either GPU or CPU driven multithreading. Such features are not critically needed but are nonetheless under consideration for future versions.

MoloVol is designed as a general tool for analyzing geometrical features of chemical structures. As such, it cannot compete with softwares dedicated to more specific functions, such as analyzing the chemical interactions within cavities. Yet, its simplicity of use and, to the best of our knowledge, its wider range of geometrical features that can be calculated compared to other available softwares makes of MoloVol a handy tool for several fields of chemistry including but not limited to host–guest chemistry, porous materials, and structural biochemistry.

Acknowledgements
We thank Laurens Bourda, Jack Davies, Prof. Ivan Jabin, Andreas Leamont, Zifei Lu, Dr. Tanya K. Ronson, Dr. Masahiro Yamashina, and Dr. Dawei Zhang for testing the beta version of MoloVol. We further thank Benedikt S. Vogler for productive and encouraging discussion during development.

Author details

Funding information
R.L. thanks Ghent University for a BOF postdoctoral fellowship.

ORCID
Jasmin B. Maglic: 0000-0003-1113-1148
Roy Lavendomme: 0000-0001-6238-8491

Notes

The authors declare no competing financial interest.

References

[1] Albrecht, M. & Hahn, F. E. (2012). Editors. *Chemistry of Nanocontainers*. Heidelberg: Springer.
[2] Van Der Voort, P., Leus, K. & de Canck, E. (2019). *Introduction to Porous Materials*. Cambridge: Wiley.
[3] Fang, Y., Powell, J. A., Li, E., Wang, Q., Perry, Z., Kirchon, A., Yang, X., Xiao, Z., Zhu, C., Zhang, L. Huang, F. & Zhou, H.-C. (2019). *Chem. Soc. Rev.* **48**, 4707–4730.
[4] Zhang, D., Ronson, T. K., Zou, Y.-Q., Nitschke, J. R. (2021). *Nat. Rev. Chem.* **5**, 168–182.
[5] Lavendomme, R., Ajami, D., Moerkerke, S. Wouters, J., Rissanen, K., Luhmer, M. & Jabin, I. (2017). *Chem. Commun.* **53**, 6468–6471.
[6] Li, J.-R., Kuppler, R. J. & Zhou, H.-C. (2009). *Chem. Soc. Rev.* **38**, 1477–1504.
[7] Sudarsanam, P., Peeters, E., Makshina, E. V., Parvulescu, V. I. & Sels, B. F. (2019). *Chem. Soc. Rev.* **48**, 2366–2421.
[8] Spek, A. L. (2009). *Acta Cryst.* D**65**, 148–155.
[9] Kleywegt, G. J. & Jones, T. A. (1994). *Acta Cryst.* D**50**, 178–185.
[10] Nicholls, A., Sharp, K. A. & Honig, B. (1991). *Proteins: Struct., Funct., Genet.* **11**, 281–296.
[11] Liang, J., Edelsbrunner, H. & Woodward, C. (1998). *Protein Sci.* **7**, 1884–1897.
[12] Ho, B. K. & Gruswitz, F. (2008). *BMC Struct. Biol.* **8**, 49.
[13] Ribeiro, J. V., Tamames, J. A. C., Cerqueira, N. M. F. S. A., Fernandes, P. A. & Ramos, M. J. (2013). *Chem. Biol. Drug Des.* **82**, 743–755.
[14] Till, M. S. & Ullmann, G. M. (2010). *J. Mol. Model.* **16**, 419–429.
[15] Schmidtke, P., Bidon-Chanal, A., Luque, F. J. & Barril, X. (2011). *Bioinformatics* **27**, 3276–3285.
[16] Chen, C. R. & Makhatadze, G. I. (2015). *BMC Bioinformatics* **16**, 101.
[17] Voss, N. R. & Gerstein, M. (2010). *Nucleic Acids Res.* **38**, W555–W562.
[18] Rother, K., Hildebrand, P. W., Goede, A., Gruening, B. & Preissner, R. (2009). *Nucleic Acids Res.* **37**, D393–D395.
[19] Sarkisov, L., Bueno-Perez, R., Sutharson, M. & Fairen-Jimenez, D. (2020). *Chem. Mater.* **32**, 9849–9867.
[20] Willems, T. F., Chris H. Rycroft, C. H., Kazi, M., Meza, J. C. & Haranczyk, M. (2012). *Microporous Mesoporous Mater.* **149**, 134–141.
[21] The PyMOL Molecular Graphics System, Version 2.0 Schrödinger, LLC.

[22] Macrae, C. F., Sovago, I., Cottrell, S. J., Galek, P. T. A., McCabe, P., Pidcock, E., Platings, M., Shields, G. P., Stevens, J. S., Towler, M. & Wood, P. A. (2020). *J. Appl. Cryst.* **53**, 226–235.
[23] BIOVIA, Dassault Systèmes, Materials Studio, San Diego: Dassault Systèmes, 2021.
[24] Cuff, A. L. & Martin, A. C. R. (2004). *J. Mol. Biol.* **344**, 1199–1209.
[25] Lee, B. & Richards, F. M. (1971). *J. Mol. Biol.* **55**, 379–400.
[26] Richards, F. M. (1977). *Annu. Rev. Biophys. Bioeng.* **6**, 151–176.
[27] Connolly, M. L. (1983). *J. App. Crystallogr.* **16**, 548–558.
[28] Pasquale, S., Sattin, S., Escudero-Adán, E. C., Martínez-Belmonte, M. & de Mendoza, J. (2012). *Nat. Commun.* **3**, 785.
[29] Ke, Y., Collins, D. J. & Zhou, H.-C. (2005). *Inorg. Chem.* **44**, 4154–4156.
[30] Liu, Y., Liu, X. & Warmuth, R. (2007). *Chem. Eur. J.* **13**, 8953–8959.
[31] Ronson, T. K., League, A. B., Gagliardi, L., Cramer, C. J. & Nitschke, J. R. (2014) *J. Am. Chem. Soc.* **136**, 15615–15624.
[32] Yamashina, M., Tanaka, Y., Lavendomme, R., Ronson, T. K., Pittelkow, M. & Nitschke, J. R. (2019). *Nature* **574**, 511–515.
[33] Petřek, M., Otyepka, M., Banáš, P., Košinová, P., Koča, J. & Damborský, J. (2006). *BMC Bioinformatics* **7**, 316.
[34] Fujita, D., Ueda, Y., Sato, S., Yokoyama, H., Mizuno, N., Kumasaka, T. & Fujita, M. (2016). *Chem* **1**, 91–101.
[35] Ongari, D., Boyd, P. G., Senja Barthel, S., Witman, M., Haranczyk, M. & Smit, B. (2017). *Langmuir* **33**, 14529–14538.
[36] Pettersen, E. F., Goddard, T. D., Huang, C. C., Couch, G.S., Greenblatt, D.M., Meng, E.C. & Ferrin, T.E. (2004). *J. Comput. Chem.* **25**, 1605–1612.
[37] Pettersen, E. F., Goddard, T. D., Huang, C. C., Meng, E. C., Couch, G. S., Croll, T. I., Morris, J. H. & Ferrin, T. E. (2021). *Protein Sci.* **30**, 70–82.
[38] Alvarez, S. (2013). *Dalton Trans.* **42**, 8617–8636.
[39] Meija, J., Coplen, T. B., Berglund, M., Brand, W. A., De Bièvre, P., Gröning, M., Holden, N. E., Irrgeher, J., Loss, R. D., Walczyk T. & Prohaska, T. (2016). *Pure Appl. Chem.* **88**, 265–291.
[40] Lindblad, J. (2005). *Image Vision Comput.* **23**, 111–122.
[41] Alex, J. M., Corvaglia, V., Hu, X., Engilberge, S., Huc, I. & Crowley, P.B. (2019). *Chem. Commun.* **55**, 11087–11090.
[42] Lindbald, J. (2003). Surface Area Estimation of Digitized Planes Using Weighted Local Configurations. Editors Nyström, I., Sanniti di Baja, G., Svensson, S. Discrete Geometry for Computer Imagery, p. 348–357. Berlin, Heidelberg: Springer.
[43] Bentley, J. L. (1975). *Commun. ACM* **18**, 509–517.
[44] Friedman, J. H., Bentley, J. L. & Finkel R. A. (1977). *ACM Trans. Math. Softw.* **3**, 209–226.