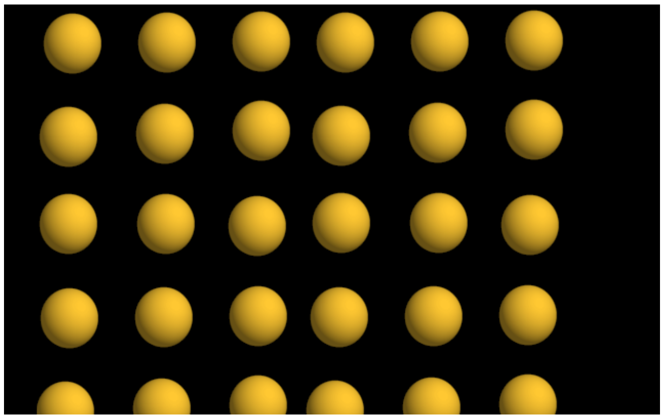
Final Project

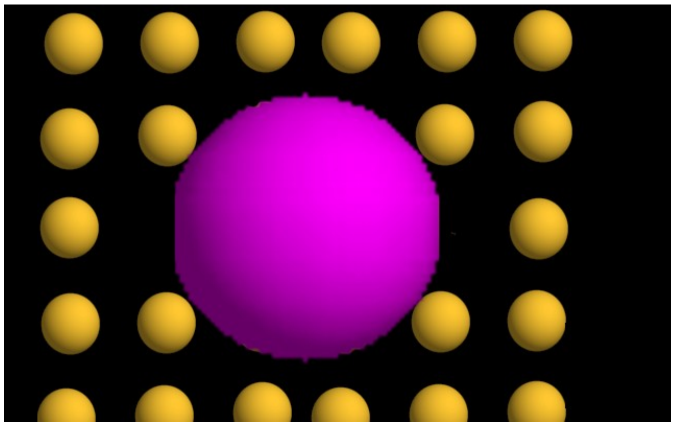
Derek Chaconas – Group Leader
John Mooring
Bryan Malyn

Algorithm

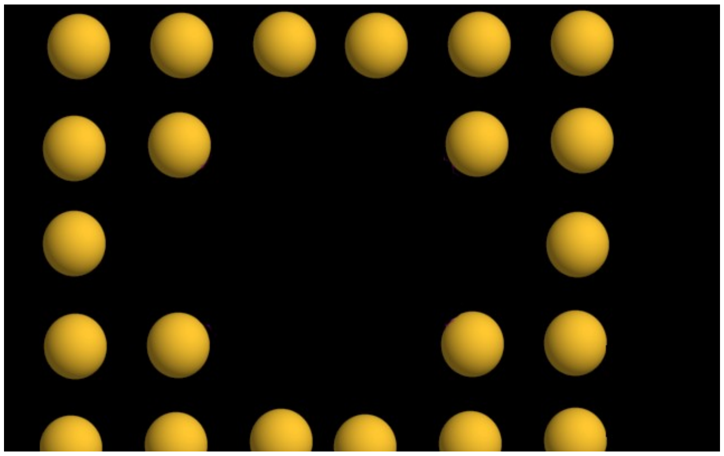
We start off with a 3D array filled with probe atoms. The size of the array is the maximum and minimum bounds of the protein, divided by the resolution, plus a buffer of one on all size to facilitate the recursive elimination of all non-cavity probe atoms.



Next, for every atom in the input file, we get the bounding box of the atom translated onto the grid. Then, for every atom inside the bounding box, we test to see if they intersect by checking if the distance between them is smaller than the sum of their radii.



If the atoms do intersect, we remove the intersecting probe atom. We are left with what is essentially a 3D bitmap negative of the protein molecule.



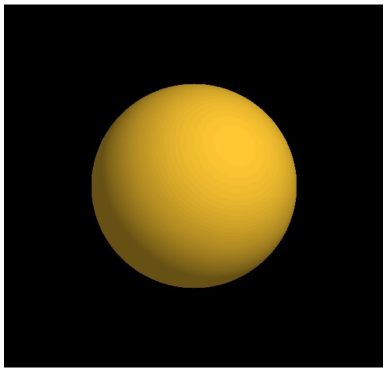
After the probe atoms that intersect the probe atom are removed, we use what is essentially a paint fill algorithm to remove all atoms that are not part of a cavity. We start at a corner, which we know is outside the atom because of the single resolution-unit buffer, remove that atom, and all atoms touching that atom. This is then reapplied to all atoms adjacent to those atoms. If there are any atoms in cavities, there will not be a chain of atoms connecting them to the outside of the protein, and will therefore not be removed.

In tests, the recursive algorithm failed due to stack overflows when used with high resolutions and large proteins. This problem was solved by implementing the algorithm non-recursively by using a stack on the heap in place of the actual function stack. This was a very straightforward translation.

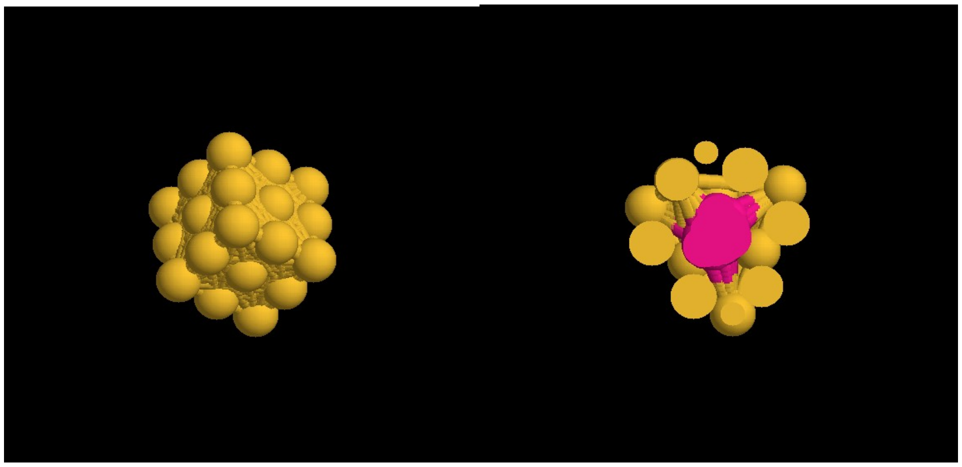
Our program now runs very fast, solving the example input with default parameters in seconds. The stack will not be blown, even for large proteins, and has been shown to be accurate for all test structures.

Test Results

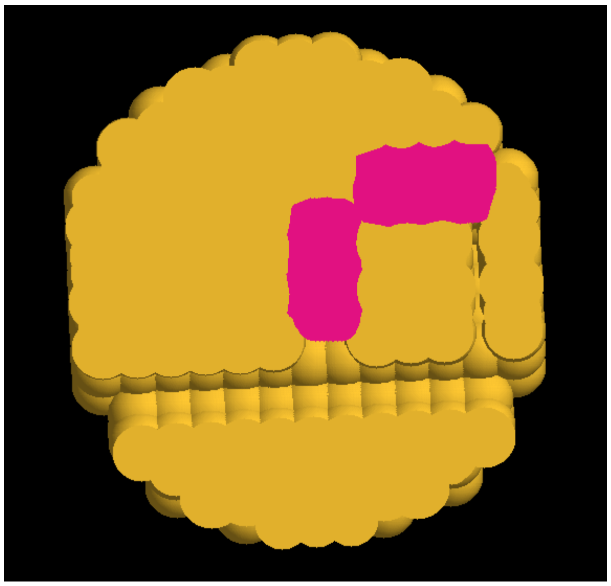
We first tested on an input of one atom. We expected to see no cavities. This was our result visualized using Rasmol. It contains no cavities.



We then tested a 3x3 cube of atoms with the middle atom missing. We expected to see a cavity in place of the removed atom. This was our result with the cavity appearing pink and atoms appearing yellow. The full structure is on the left, and a sliced view, showing the cavity atoms in pink, is on the right.



We proceeded to test the algorithm on the provided example input file. No cavities were found with the default parameters and atom radii, which (after inspecting the example input) proved to be accurate. To further test our algorithm, we tried the example input with atoms of a larger radius.



The actual radius of the protein atoms used is not accurately represented in the picture, however, we can see a clear distinction between cavities and voids. The void that goes all the way through the molecule is not a cavity, however, the radii above are sufficient to partition off another section of the molecule and create a cavity (again, the actual radius of the protein atoms is not accurately represented in the picture).