Computer Physics Communications 183 (2012) 2494–2497

Contents lists available at SciVerse ScienceDirect

Computer Physics Communications

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

# ARVO-CL: The OpenCL version of the ARVO package — An efficient tool for computing the accessible surface area and the excluded volume of proteins via analytical equations☆

Ján Buša Jr. [a,b,c], Shura Hayryan [a], Ming-Chya Wu [a,d,e], Ján Buša [b], Chin-Kun Hu [a,*]

[a] Institute of Physics, Academia Sinica, Nankang, Taipei 11529, Taiwan

[b] Faculty of Electrical Engineering and Informatics, Technical University of Košice, 040 01 Košice, Slovak Republic

[c] FURT Solutions, s.r.o., Strojárenská 3, 040 01 Košice, Slovak Republic

[d] Research Center for Adaptive Data Analysis, National Central University, Chungli 32001, Taiwan

[e] Department of Physics, National Central University, Chungli 32001, Taiwan

ARTICLE INFO

*Article history:*
Received 3 April 2012
Accepted 23 April 2012
Available online 28 April 2012

*Keywords:*
ARVO
Proteins
Solvent accessible area
Excluded volume
Stereographic projection
OpenCL package

ABSTRACT

Introduction of Graphical Processing Units (GPUs) and computing using GPUs in recent years opened possibilities for simple parallelization of programs. In this update, we present the modernized version of program ARVO [J. Buša, J. Dzurina, E. Hayryan, S. Hayryan, C.-K. Hu, J. Plavka, I. Pokorný, J. Skivánek, M.-C. Wu, Comput. Phys. Comm. 165 (2005) 59]. The whole package has been rewritten in the C language and parallelized using OpenCL. Some new tricks have been added to the algorithm in order to save memory much needed for efficient usage of graphical cards. A new tool called '`input_structure`' was added for conversion of pdb files into files suitable for work with the C and OpenCL version of ARVO.

**New version program summary**

*Program title*: ARVO-CL

*Catalog identifier*: ADUL_v2_0

*Program summary URL*: http://cpc.cs.qub.ac.uk/summaries/ADUL_v2_0.html

*Program obtainable from*: CPC Program Library, Queen's University, Belfast, N. Ireland

*Licensing provisions*: Standard CPC licence, http://cpc.cs.qub.ac.uk/licence/licence.html

*No. of lines in distributed program, including test data, etc.:* 11834

*No. of bytes in distributed program, including test data, etc.:* 182528

*Distribution format*: tar.gz

*Programming language*: C, OpenCL.

*Computer*: PC Pentium; SPP'2000.

*Operating system*: All OpenCL capable systems.

*Has the code been vectorized or parallelized?*: Parallelized using GPUs. A serial version (non GPU) is also included in the package.

*Classification*: 3.

*External routines*: cl.hpp (http://www.khronos.org/registry/cl/api/1.1/cl.hpp)

*Catalog identifier of previous version*: ADUL_v1_0

*Journal reference of previous version*: Comput. Phys. Comm. 165(2005)59

*Does the new version supercede the previous version?*: Yes

☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).

* Corresponding author. Tel.: +886 2 27896720; fax: +886 2 27834187.
*E-mail addresses:* jan.busa.2@tuke.sk (J. Buša Jr.), shura@phys.sinica.edu.tw (S. Hayryan), mcwu@ncu.edu.tw (M.-C. Wu), jan.busa@tuke.sk (J. Buša), huck@phys.sinica.edu.tw (C.-K. Hu).

0010-4655/$ – see front matter 
doi:10.1016/j.cpc.2012.04.019

*Nature of problem:* Molecular mechanics computations, continuum percolation

*Solution method:* Numerical algorithm based on the analytical formulas, after using the stereographic transformation.

*Reasons for new version:* During the past decade we have published a number of protein structure related algorithms and software packages [1,2,3,4,5,6] which have received considerable attention from researchers and interesting applications of such packages have been found. For example, ARVO [4] has been used to find that ratios of volume $V$ to surface area $A$, for proteins in Protein Data Bank (PDB) distribute in a narrow range [7]. Such a result is useful for finding native structures of proteins.

Therefore, we consider that there is a demand to revise and modernize these tools and to make them more efficient. Here we present the new version of the ARVO package. The original ARVO package was written in the FORTRAN language. One of the reasons for the new version is to rewrite it in C in order to make it more friendly to the young researchers who are not familiar with FORTRAN. Another, more important reason is to use the possibilities for speeding-up provided by modern graphical cards. We also want to eliminate the necessity of re-compiling the program for every molecule. For this purpose, we have added the possibility of using general pdb [8] files as an input. Once compiled, the program can receive any number of input files successively. Also, we found it necessary to go through the algorithm and to make some tricks for avoiding unnecessary memory usage so that the package becomes more efficient.

*Summary of revisions:* 1. *New tool*. ARVO is designed to calculate the volume and accessible surface area of an arbitrary system of overlapping spheres (representing atoms), the biomolecules being just one albeit important, application. The user provides the coordinates and radii of the spheres as well as the radius of the probe sphere (water molecule for biomolecules). In the old version of ARVO the input of data was organized immediately in the code, which made it necessary to re-compile the program after every change in the input data. In the current version a module called '*input_structure*' has been created to input the data from an independent external file. The coordinates and radii are stored in the file with extension **.ats* (see the directory 'input' in the package). Each line in the file corresponds to one sphere (atom) and has the format

24.733 −4.992 −13.256 2.800.

The first three numbers are the $(x, y, z)$ coordinates of the atom and the last one is the radius. It is important to remember that the radius of the probe sphere must be already added to this number. In the above example, the value 2.800 is obtained by the formula *"sphere radius+probe sphere radius"*. In the case of the arbitrary system of spheres the file **.ats* is created by the user. In the case of proteins the '*input_structure*' takes as an input a file in the format compatible with Protein Data Bank (pdb) format [8] and creates a corresponding **.ats* file. It also assigns automatically, radii to individual spheres and (optionally) adds to all radii the probe sphere (water molecule) radius. As output, it produces a file containing coordinates of spheres together with radii. This file works automatically as an input for ARVO. Using an external tool allows users to create their own mappings of atoms and radii without the need to re-compile the tool '*input_structure*' or program ARVO. It is again the user's responsibility to assign proper radii to each type of atom. One can use any of the published standard sets of radii (see for example, [9,10,11,12,13]). Alternatively, the user can assign his own values for radii immediately in the module *input_structure*. The radii are assigned in a special file with extension ***pds (see the documentation) which consists of lines like this: ATOM CA ALA 2.0 which is read as "the $C_{alpha}$ atom of Alanine has radius 2.0 Angstroms". Here we provide for testing of the file *rashin.pds* where the radii are assigned according to [12].

The output file contains only recognized atoms. Atoms that were not recognized (are not part of mapping) are written to a separate log file allowing the user to review and correct the mapping files later.

2. *The Language*. Implementing the program in C is a natural first step when translating a program into OpenCL. This implementation is rewritten line-by-line from the original FORTRAN version of ARVO.

3. *OpenCL implementation*. OpenCL [14] is an open standard for parallel programming of heterogeneous systems. Unlike other parallelization technologies like CUDA [15] or ATI Stream [16] which are interconnected with specific hardware (produced by NVIDIA or ATI, respectively), OpenCL is vendor-independent, and programs written in OpenCL can be run on any hardware of companies supporting this standard, including AMD, INTEL, and NVIDIA. Programs written in OpenCL can be run without much change both on CPUs and GPUs.

*Improvements as compared with the original version:* Support for files in the format as created by '*input_structure*'; input of parameters (name of input file) via command line; dynamic size of arrays—removal of the necessity to re-compile the program after any change in size of structures; memory allocation according to the real demands of the application; replacing north pole test by slight reduction of the radius (see below).

To compile an OpenCL program, one needs to download and install the appropriate driver and software development kit (SDK). The program itself consists of two parts: a part running on the CPU and a part running on the GPU. The CPU initializes communication between the computer and the GPU, load data, processes and exports results. The GPU does the parallel part of calculation, consisting of the search for neighboring atoms and calculating the contribution of the area and volume of the individual atom to the total area and volume of the molecule. For details of the algorithm, please read Refs. [3,4].

In programming using OpenCL, more attention must be given to memory used than in a classical approach. Memory of the device is usually limited and therefore, some changes to the original algorithm are necessary. First, unlike in the FORTRAN version of the program, no structures containing the list of

**Table 1**
Comparison of volumes and surface areas of different proteins obtained by original ARVO and by the new version. Different strategies for dealing with the "north pole" are applied. The first column contains the PDB ID of the protein and the number of atoms. Second column contains the volume of the protein obtained with original ARVO (upper number) and the difference with the new approach (lower number). Third column contains the same as in the second column for the surface area. Fourth column contains the number of rotations of the molecule in original ARVO (upper number) and the number of atoms whose radii have been reduced in the new version (lower number). Fifth column contains the relative errors for the volume (upper number) and the area (lower number).

| Protein atoms # | Volume diff | Area diff | Rotat. reduct. | $\delta_{\text{volume}}$ (%) $\delta_{\text{area}}$ (%) |
|---|---|---|---|---|
| 3rn3 | 23,951.180469 | 6858.322636 | 3 | $-1.04 \cdot 10^{-7}$ |
| 957 | −0.000025 | −0.000007 | 1 | $-1.02 \cdot 10^{-7}$ |
| 3cyt | 40,875.867395 | 11,455.474832 | 3 | $-3.85 \cdot 10^{-6}$ |
| 1600 | −0.001575 | 0.001415 | 4 | $1.24 \cdot 10^{-4}$ |
| 2act | 38,608.243038 | 9054.007350 | 4 | $1.28 \cdot 10^{-4}$ |
| 1657 | 0.049480 | 0.001733 | 2 | $1.91 \cdot 10^{-5}$ |
| 2brd | 43,882.735479 | 10,918.203529 | 21 | $-7.84 \cdot 10^{-7}$ |
| 1738 | −0.000344 | −0.000097 | 1 | $-8.88 \cdot 10^{-7}$ |
| 8tln | 56,698.988883 | 12,496.978064 | 15 | $-1.70 \cdot 10^{-6}$ |
| 2455 | −0.000966 | 0.000459 | 4 | $3.67 \cdot 10^{-6}$ |
| 1rr8 | 105,841.502192 | 27,983.159772 | 18 | $-6.60 \cdot 10^{-7}$ |
| 4108 | −0.000699 | −0.000214 | 4 | $-7.65 \cdot 10^{-7}$ |
| 1xi5 | 1743,445.092001 | 863,139.882703 | 1 | $4.42 \cdot 10^{-7}$ |
| 15,696 | 0.007709 | 0.000070 | 1 | $8.11 \cdot 10^{-9}$ |

neighbor atoms are created. The search for the neighbors is done on-line, when the calculation of the contribution from individual atoms is being performed.

The strategy behind the *North Pole check and molecule rotation* [4, Sec. 4.7] has been changed. If during the north pole test, the north pole of the active sphere lies close to the surface of a neighboring sphere, the radius of such a neighboring sphere is multiplied by 0.9999 instead of rotating the whole molecule. This allows the algorithm to continue normally. Changing the radius of one atom changes the area and the volume of this atom by 0.02% and 0.03%, respectively. As the atom's contribution to the total area (volume) of the protein is usually only a part of the atom's total area (volume) and since there are many atoms in the protein itself, the change of total area (volume) is much smaller than 0.02% (0.03%). Testings showed relative errors ranging from $10^{-4}$ down to $10^{-8}$. An additional benefit of this approach is, that the whole molecule is not rotated and therefore no errors are introduced there which would occur during such rotation. We were even able to find a protein (1S1I having 31,938 atoms), where, after several hundreds of rotations, ARVO was not able to find such a position that the original north pole test could pass. For such proteins the new approach is the only one possible.

Some data obtained using the north pole test (with rotation) and those without the north pole test (with radii reduction) are summarized in Table 1. The radius of water molecule was set to 1.4 Å, and Rashin's set of the van der Waals radii of atoms [12] was used. The first column contains the protein name and the number of atoms. Each cell of the second and the third columns contains two numbers. The upper number is the volume (surface area) obtained using the original ARVO algorithm [4] with conventional north pole test and rotation. The lower number shows the difference coming from using the new approach. The upper number in the fourth column shows the number of rotations when using the original version and the second number is the number of atoms for which the radius has been reduced. The relative error of volume (upper number) and area (lower number) obtained by using radius reduction are shown in the last column. It can be seen clearly that the error is negligible.

The disadvantage is that calculations using OpenCL are done with single precision only. This comes from the fact that the OpenCL standard does not support double precision float number operations as a basic part but as an extension only. This means that availability of double precision calculations depends on the device (CPU, GPU) vendor. Switching to double precision calculations downgrades speed performance (calculations in double precision are 8–2 times slower than the same calculations in single precision). Another problem is that after using the double precision switch, all calculations are done with double precision which leads to problems with insufficient memory. This problem can be bypassed by explicitly switching to single precision where possible but this requires careful modification of the whole program source. Since on our GPU (NVIDIA GTX 480) double precision was available, we have decided to use the double precision only for the critical parts of algorithm (s.a. integral calculation), leaving non-critical parts in single precision. This allowed us to speed up the calculation and to obtain acceptable results.

Results of the test calculations are given in Table 2. All calculations except for 2brd0 have been performed using water radius 1.4 Å. The first column contains the protein name and the number of atoms. The second column contains computation time in seconds (in FORTRAN/CPU—upper part and OpenCL/GPU—lower part). The third column is a speed-up (time on the CPU divided by time on the GPU). The fourth and fifth columns contain the volume and area calculated in FORTRAN (upper number) and the difference when compared to results obtained by OpenCL (lower number). As one can see, the area and the volume obtained using FORTRAN (in double precision) and the OpenCL implementation (combination of single and double precisions) are practically the same. This is even more clear from the relative error of the OpenCL implementation as shown in the last column (upper number for volume, and lower number

**Table 2**
The table contains comparative data on precision and computational times obtained by FORTRAN vs. OpenCL implementations of ARVO. The structure of the columns is similar to Table 1. Note that last protein (1s1i) was not calculated using FORTRAN implementation and comparison presented is between C and OpenCL version. This is because we were not able to find such rotation that north pole test would pass.

| Protein atoms # | Time F95 (s) OpenCL | Speed up | Volume diff | Area diff | $\delta_{volume}$ (%) $\delta_{area}$ (%) |
|---|---|---|---|---|---|
| 1eca | 8.23 | 6.01 | 26,072.003069 | 7004.168138 | $1.65 \cdot 10^{-5}$ |
| 1031 | 1.37 | | 0.004310 | 0.000498 | $7.11 \cdot 10^{-6}$ |
| 2ptn | 13.72 | 9.01 | 39,273.220933 | 9227.570716 | $-2.01 \cdot 10^{-5}$ |
| 1629 | 1.52 | | −0.007906 | −0.005795 | $-6.28 \cdot 10^{-5}$ |
| 2brd | 15.77 | 9.91 | 43,882.735136 | 10,918.203432 | $-1.44 \cdot 10^{-5}$ |
| 1738 | 1.59 | | −0.006326 | 0.001471 | $1.35 \cdot 10^{-5}$ |
| 2brd0 | 0.29 | 0.91 | 22,412.825807 | 22,546.123881 | $-9.13 \cdot 10^{-5}$ |
| 1738 | 0.32 | | −0.020471 | −0.008437 | $-9.17 \cdot 10^{-4}$ |
| 8tln | 23.32 | 13.74 | 56,698.988550 | 12,496.977990 | $-5.34 \cdot 10^{-6}$ |
| 2455 | 1.70 | | −0.003028 | −0.008708 | $-4.64 \cdot 10^{-4}$ |
| 1rr8 | 30.89 | 17.67 | 105,841.501492 | 27,983.159558 | $1.93 \cdot 10^{-5}$ |
| 4108 | 1.75 | | 0.020445 | −0.000802 | $-2.87 \cdot 10^{-6}$ |
| 1s1i | 286.81 | 33.95 | 816,980.348702 | 253,160.674893 | $-1.40 \cdot 10^{-4}$ |
| 31,938 | 8.45 | | −1.140763 | 0.049478 | $1.95 \cdot 10^{-5}$ |

for area). As to computational time, FORTRAN (C) implementation is appropriate in the case when the calculation takes approximately less than 2 s. This is because in the case of OpenCL some time – about 0.3–1.5 s on testing configuration – is needed for the initialization of the device and for starting the communication. Speed-up is clearly visible for large proteins when the parallel approach can be exploited, but complexity of protein needs to be taken into account as well. Compare the times for 2brd (water radius 1.4 Å) and 2brd0 (water radius 0 Å). The difference is in the number of neighbors (overlapping spheres). While, for water radius 1.4 Å the number of neighbors is high and using the GPU is efficient, for water radius 0 Å it is better to use CPU. All results were obtained on a test configuration with CPU Intel Core i7 930 processor running at 2.8 GHz and a GPU NVIDIA GeForce GTX 480.

At the time of writing, OpenCL allowed the allocation of only 1/4 of the total memory of the devices (CPU, GPU) by one call to malloc. This can be bypassed by four individual calls of memory allocation requesting 1/4 of the total devices' memory. It is advisable to use a dedicated GPU for the calculations since sharing a GPU for calculations and displaying graphics can lead to unexpected results due to common access to the memory of devices.

*Restrictions:* The program does not account for possible cavities inside the molecule. The current version works in a combination of single and double precisions (see Summary of revisions for details).

*Running time:* Depends on the size of the molecule under consideration. For molecules whose running time was less than 2 s in the old version the performance is likely to decrease. This changes considerably when larger molecules are calculated (in test configuration speed-ups up to 34 were obtained).

*References:*

[1] F. Eisenmenger, U.H.E. Hansmann, S. Hayryan, C.-K. Hu, Comput. Phys. Commun. 138 (2001) 192.
[2] F. Eisenmenger, U.H.E. Hansmann, S. Hayryan, C.-K. Hu, Comput. Phys. Commun. 174 (2006) 422.
[3] S. Hayryan, C.-K. Hu, J. Skrivánek, E. Hayryan, I. Pokorný, J. Comput. Chem. 26 (2005) 334.
[4] J. Busa, J. Dzurina, E. Hayryan, S. Hayryan, C.-K. Hu, J. Plavka, I. Pokorný, J. Skrivánek, M.-C. Wu, Comput. Phys. Commun. 165 (2005) 59.
[5] J. Busa, S. Hayryan, C.-K. Hu, J. Skrivánek, M.-C. Wu, J. Comput. Chem. 30 (2009) 346.
[6] J. Busa, S. Hayryan, C.-K. Hu, J. Skrivánek, M.-C. Wu, Comput. Phys. Commun. 181 (2010) 2116.
[7] M.-C. Wu, M.S. Li, W.-J. Ma, M. Kouza, C.-K. Hu, EPL 96 (2011) 68005.
[8] http://www.rcsb.org.
[9] B. Lee, F.M. Richards, J. Mol. Biol. 55 (1971) 379.
[10] F.M. Richards, Annu. Rev. Bipohys. Bioeng. 6 (1977) 151.
[11] A. Shrake, J.A. Rupley, J. Mol. Biol. 79 (1973) 351.
[12] A.A. Rashin, M. Iofin, B. Honig, Biochemistry 25 (1986) 3619.
[13] C. Chotia, Nature 248 (1974) 338.
[14] http://www.khronos.org/opencl/.
[15] http://www.nvidia.com/object/cuda_home_new.html.
[16] http://www.amd.com/stream.