User guide

Editor: Viktor Bernholtz

Version 1.0

Status

Granskad	DOK	2018-05-25
Godkänd	Rickard Armiento	2018-05-28

PROJECT IDENTITY

2018/Spring, Group 2
Linköpings Tekniska Högskola, IFM

Group members

Name	Role	Phone no.	E-mail address
Anders Rehult	Project leader (PL)	076-3161206	andre449@student.liu.se
Marian Brännvall	Document manager (DOK)	070-7280044	marbr639@student.liu.se
Andreas Kempe	Secretary (SE)	073-9796689	andke133@student.liu.se
Viktor Bernholtz	Viktor Bernholtz (VB)	073-0386030	vikbe253@student.liu.se

Client: IFM, Linköpings universitet, 581 83 Linköping
Contact person of client: Rickard Armiento, 013-281249, rickard.armiento@liu.se
Course examiner: Per Sandström, 013-282902, persa@ifm.liu.se
Main supervisor: Johan Jönsson, 013-281176, johan.jonsson@liu.se

Contents

Document history

Version	Date	Changes	Done by	Reviewed
0.1	2018-05-22	First draft.	Project group	PL
0.2	2018-05-25	Second draft.	Project group	DOK
1.0	2018-05-28	Approved for version one.	Project group	DOK

Licens

This documet is licensed as CC0, see appendix A.

1   Introduction

ENVISIoN is an open source toolkit for electron visualisation.

ENVISIoN is implemented by using a modified verision of the Inviwo visualisation framework, developed at the Scientific Visualization Group at Linköpings universitet, LIU.

ENVISIoN has been developed as a part of the course TFYA75: Applied Physics - Bachelor's Project, given at Linköpings universitet, LiU.

The present version was developed during the spring term of 2018 by a project group consisting of: Anders Rehult, Marian Brännvall, Andreas Kempe and Viktor Bernholtz. Supervisor: Johan Jönsson; Requisitioner and co-supervisor: Rickard Armiento; Visualization expert: Rickard Englund; and Course examiner: Per Sandström. The work is based on a previous version by the project group taking the course in the spring term of 2017 consisting of: Josef Adamsson, Robert Cranston, David Hartman, Denise Härnström, Fredrik Segerhammar. Supervisor: Johan Jönsson; Requisitioner and co-supervisor: Rickard Armiento; Visualization expert: Peter Steneteg; and Course examiner: Per Sandström.

ENVISIoN provides a set of Python scripts that allow the user to:

- Read and parse output from electronic structure calculations made by the program VASP and storing the result in a structured HDF5 file.

- Generate interactive Inviwo visualisation networks for common tasks when analyzing electronic structure calculations. Presently there is support for visualising the crystal structure of the unit cell of a material, electron localization function (ELF)-data, electronic charge density, and density of states - both total and partial. The system also provides the ability to interconnect some of the networks mentioned above.

2   How to build and run ENVISIoN

These instructions show how to build Inviwo and ENVISIoN on Ubuntu 18.04 LTS.

2.1   Install git

Start by installing git, which will be used to fetch ENVISIoN in the next step.

```
sudo apt install git
```

2.2   Download ENVISIoN

Go to your home folder and clone ENVISIoN from Github. This guide will assume that both ENVISIoN and Inviwo will be placed directly under the home folder.

```
cd
git clone https://github.com/rartino/ENVISIoN
```

2.3   Prepare Inviwo using the ENVISIoN install script

ENVISIoN provides an install script for Ubuntu 18.04 LTS. Executing the installation script will install all required dependencies, clone Inviwo from Github and configure the Inviwo build.

The script should *NOT* be run as root, but as your own user and it will ask for your password when it needs root rights. It is possible that the script will ask for other user input during the process, if that's the case, just accept the default.

```
cd ~/ENVISIoN/scripts

./install_envision_ubuntu_1804.sh /home/$USER/ENVISIoN /home/$USER/inviwo
```

Once the installation script has run, it prints build instructions. Follow the instructions and start the build. The instructions will tell you to *cd* to the build directory and execute make.

An easy way to modify the build settings, if needed, is to install the cmake curses gui and run it in the build directory.

To install the cmake gui:

```
sudo apt install cmake-curses-gui
```

Running cmake in the build directory:

```
cd ~/inviwo/build
ccmake .
```

When in the GUI, press *c* to apply the current configuration, *g* to generate build files and *q* to quit. If settings have changed, it is possible that you will need to press *c* more than once before the *g* option becomes available.

After having generated the build files, the project can now be rebuilt with the new settings by executing *make* like earlier.

3   Start Inviwo and run ENVISIoN scripts

After the Inviwo build is done, the binary will be available as *bin/inviwo* in the build folder. Start it by executing the command below, while still standing in the build directory.
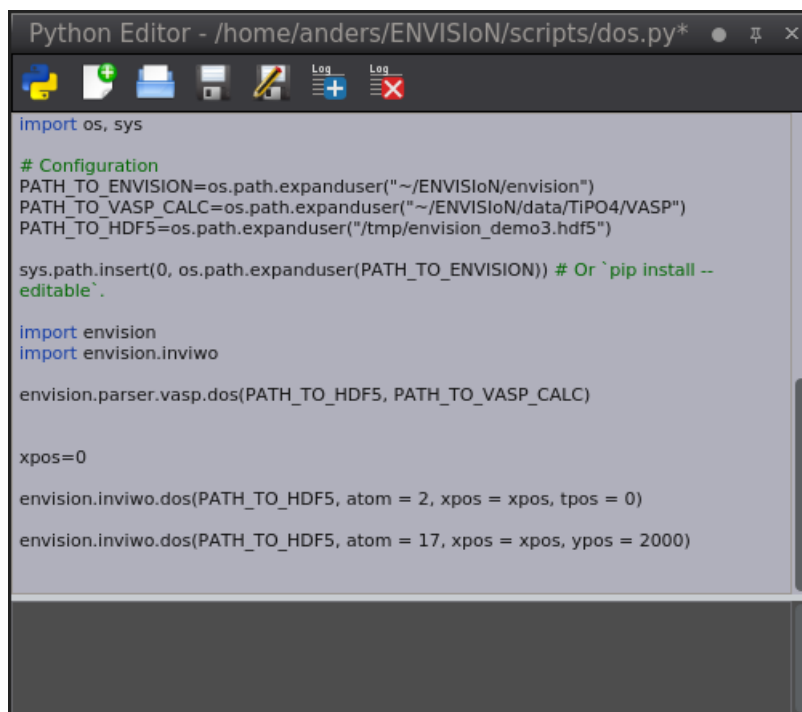
```
cd ~/inviwo/build
./bin/inviwo
```

- Open Python Editor under Python menu.

- In the Python editor, click Open Script.

- Select one of the scripts.

- Click open.

- Click the python logo in the top left corner to run.

More information about each script and how to use them in chapter 4.

4 Scripts

Descriptions and examples of all the avaliable scripts are found below. The scripts are called through the python editor in Inviwo as shown in figure 1.

Figure 1: Python script for calling on the DOS visualisation module of ENVISIoN. Note that the pathway PATH_TO_VASP_CALC here is set to "~/ENVISIoN/data/TiPO4/VASP", the folder containing the relevant VASP output files.

In all of the scripts the user has to specify the path to ENVISIoN, a path where VASP output files are found, and a desired path where ENVISIoN will generate a HDF5-file, or, if one already exists, use that file and skip parsing VASP data. **NOTE:** change the value of PATH_TO_HDF5, delete the HDF5-file, or rename the HDF5-file in between visualisations if a new material is to be visualised.

4.1 Unit cell

In the Python editor of Inviwo the user can choose to generate a visualisation of the unit cell by opening the folder *scripts* and running the file *unitcell.py*.

Figure 2 is an example of a unit cell visualisation, in this case of TiPO4. Figure 3 shows the network responsible for the aforementioned visualisation.
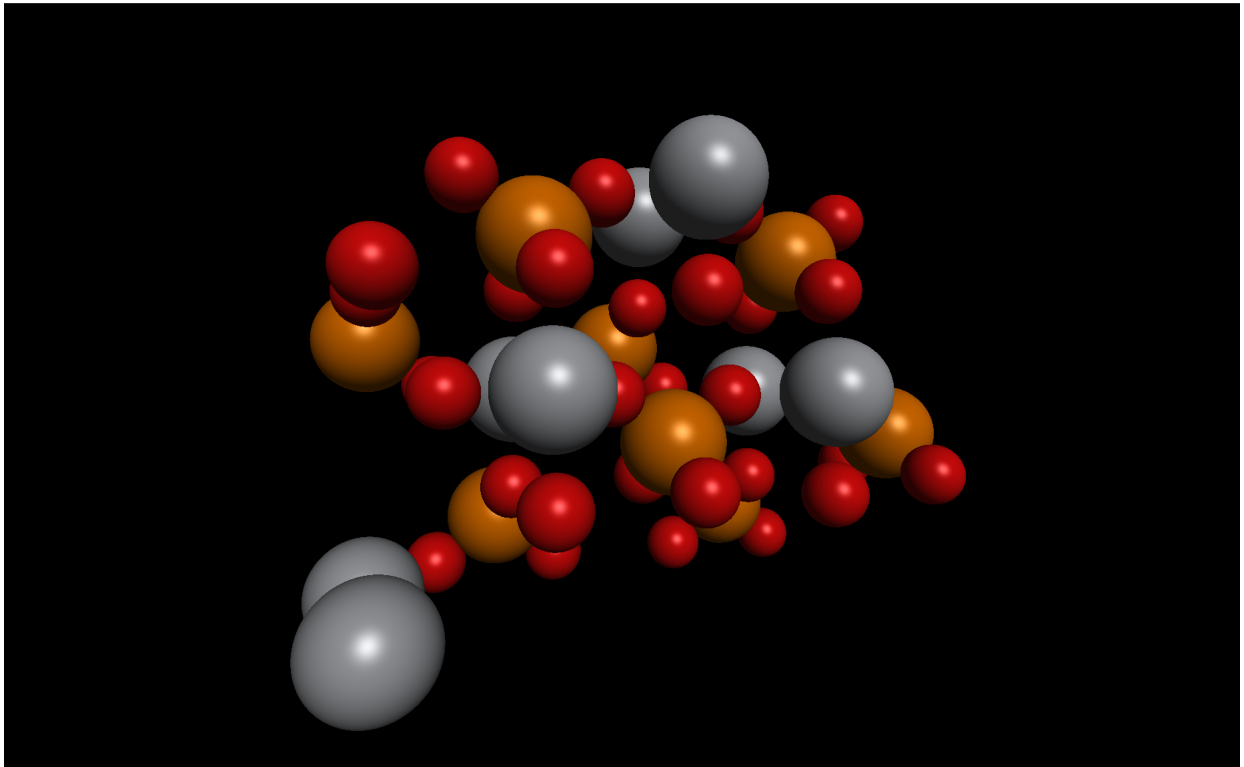
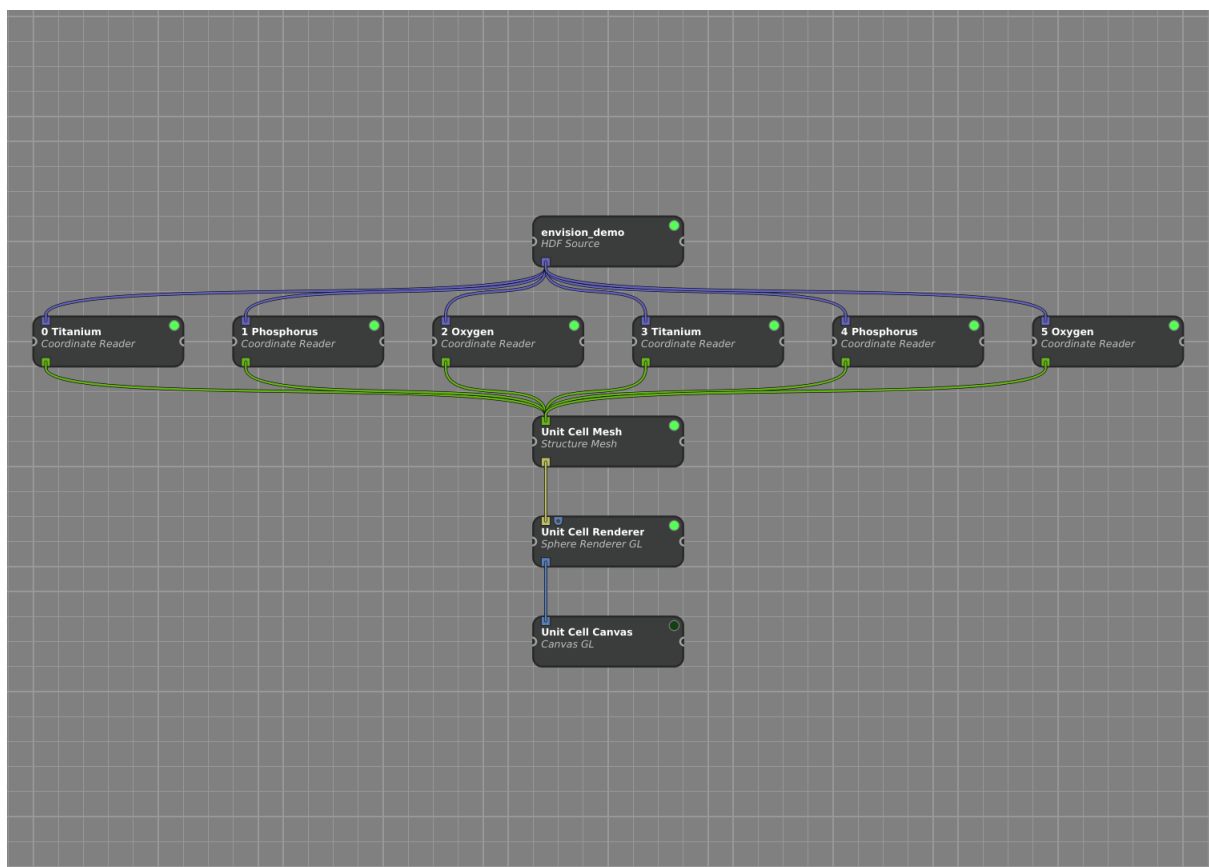Figure 2: Visualisation of the crystal structure of the unit cell of TiPO4 in Inviwo.

Figure 3: Network for visualisation of TiPO4 in Inviwo.

4.2 ELF

In the Python editor of Inviwo the user can choose to generate a visualisation of the Electron Localisation Function (ELF)-data as shown in figure 4 with corresponding network shown in figure 5 by opening the folder *scripts* and running the file *elf.py*. A second image showing the ELF-data of a slice of the unit cell can be added by changing the argument *Slice* of the function *elf* to True. This slice corresponds to a plane intersecting the unit cell, which can be moved using the W and S keys on the keyboard. The orientation of this plane can be changed by selecting the processor "Volume Slice" in the network by clicking on it, then choosing another value from the drop down menu in the property "Slice along axis".

The ELF-data can also be visualised as an isosurface. This is achieved by changing the argument *iso* of the function *elf* from None to any value between 0 and 1. The slice function described above is not compatible with isosurface visualisation.

The colors and opacities of different values can be changed by selecting the processor "ELF raycaster" in the network, clicking on the property "Transfer function", and interacting with the window that pops up. This window shows multicolored dots which can be manipulated to control opacity. Dots can be added by double clicking, and removed by clicking a dot and pressing the Delete key on the keyboard. Dots can be dragged and dropped using the mouse. The horizontal axis of the plane on which the dots are placed corresponds to ELF values between 0 and 1, while the vertical axis corresponds to opacities between 0 (transparent) and 1 (completely opaque). Figure 9 in chapter 4.3 shows what this window looks like.
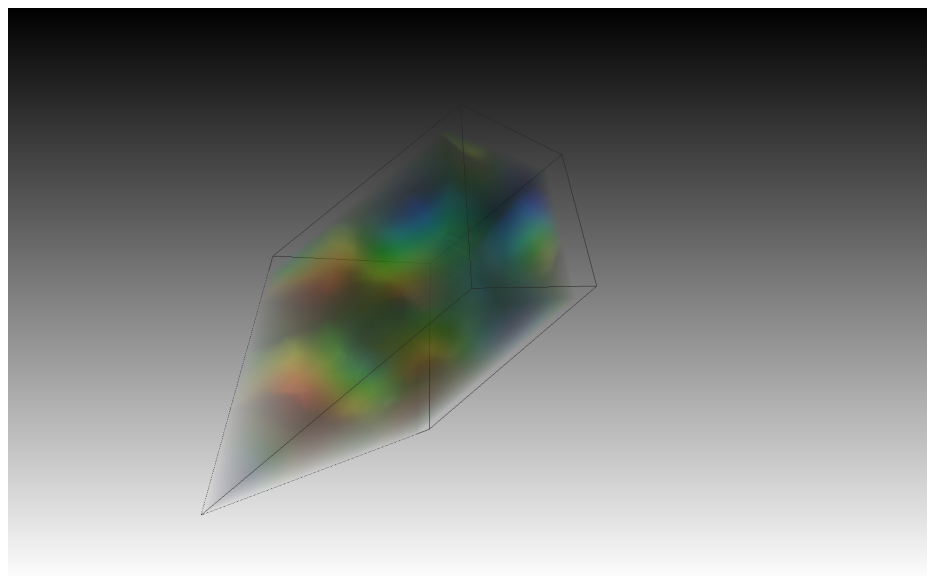

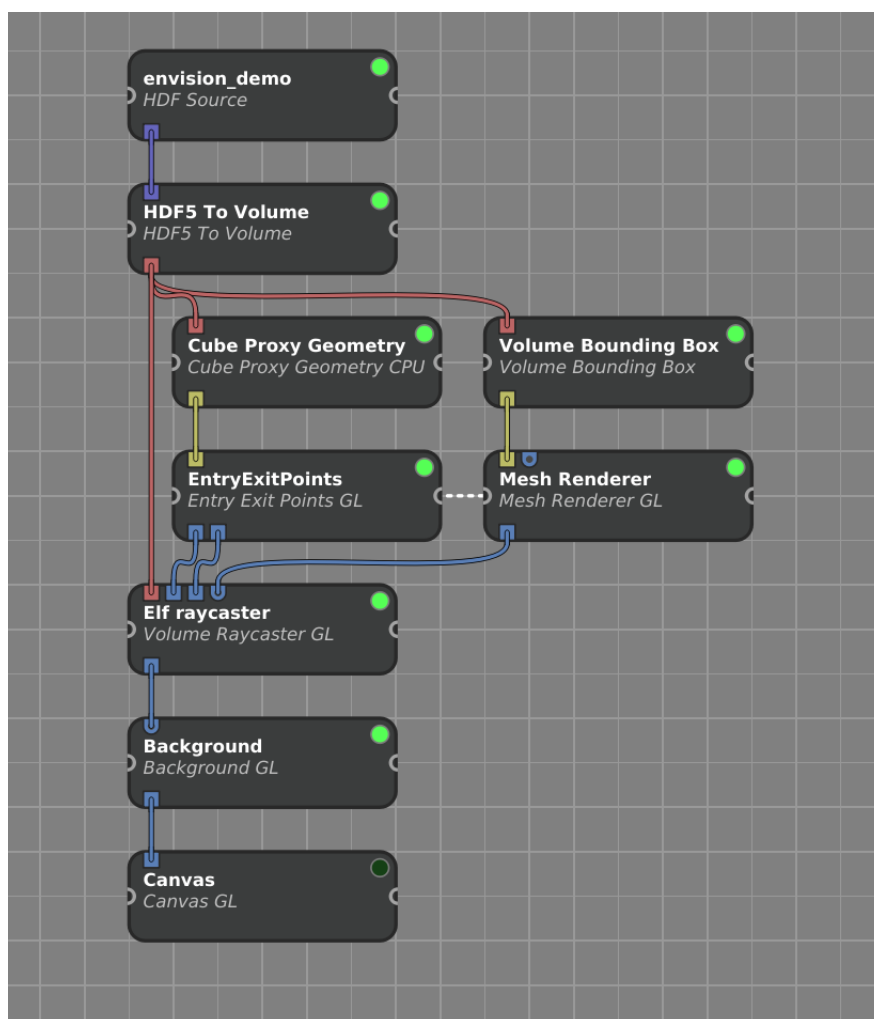Figure 4: Visualisation of ELF-data for diamond in Inviwo.

Figure 5: Network for visualisation of ELF-data for diamond in Inviwo.

4.3 Charge density

In the Python editor of Inviwo the user can choose to generate a visualisation of the charge density by opening the folder *scripts* and running the file *charge.py*. A second image showing the charge density of a slice of the unit cell can be added by changing the argument *Slice* of the function *charge* to True. This slice corresponds to a plane intersecting the unit cell, which can be moved using the W and S keys on the keyboard. The orientation of this plane can be changed by selecting the processor "Volume Slice" in the network by clicking on it, then choosing another value from the drop down menu in the property "Slice along axis". This type of visualisation is shown in figure 6, and the generating network is found in figure 7.

The charge density can also be visualised as an isosurface. This is achieved by changing the argument *iso* of the function *charge* from None to any value between 0 and 1. Figure 8 shows an example of this. The slice function described above is not compatible with isosurface visualisation.

The colors and opacities of different values can be changed by selecting the processor "Charge raycaster" in the network, clicking on the property "Transfer function", and interacting with the window that pops up. This window shows multicolored dots which can be manipulated to control opacity. Dots can be added by double clicking, and removed by clicking a dot and

pressing the Delete key on the keyboard. Dots can be dragged and dropped using the mouse. The horizontal axis of the plane on which the dots are placed corresponds to charge density values between 0 and 1, while the vertical axis corresponds to opacities between 0 (transparent) and 1 (completely opaque). Figure 9 shows what this window looks like.
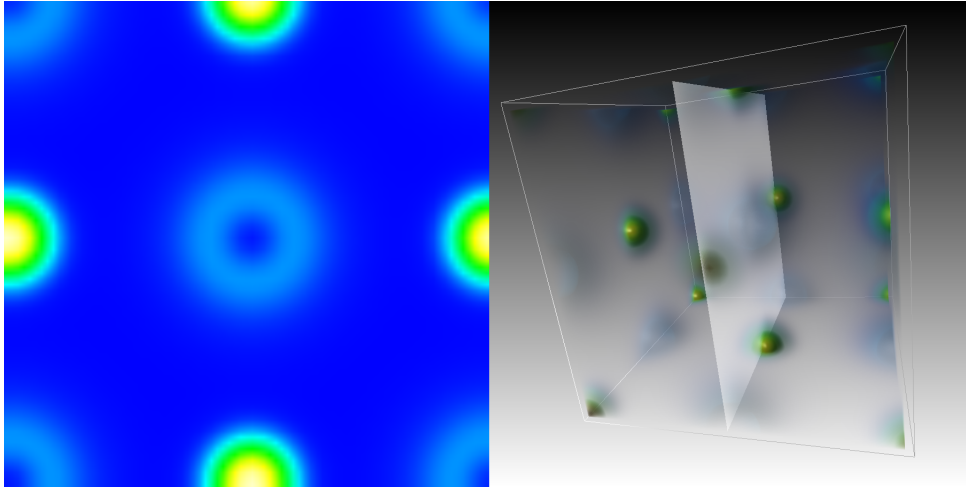

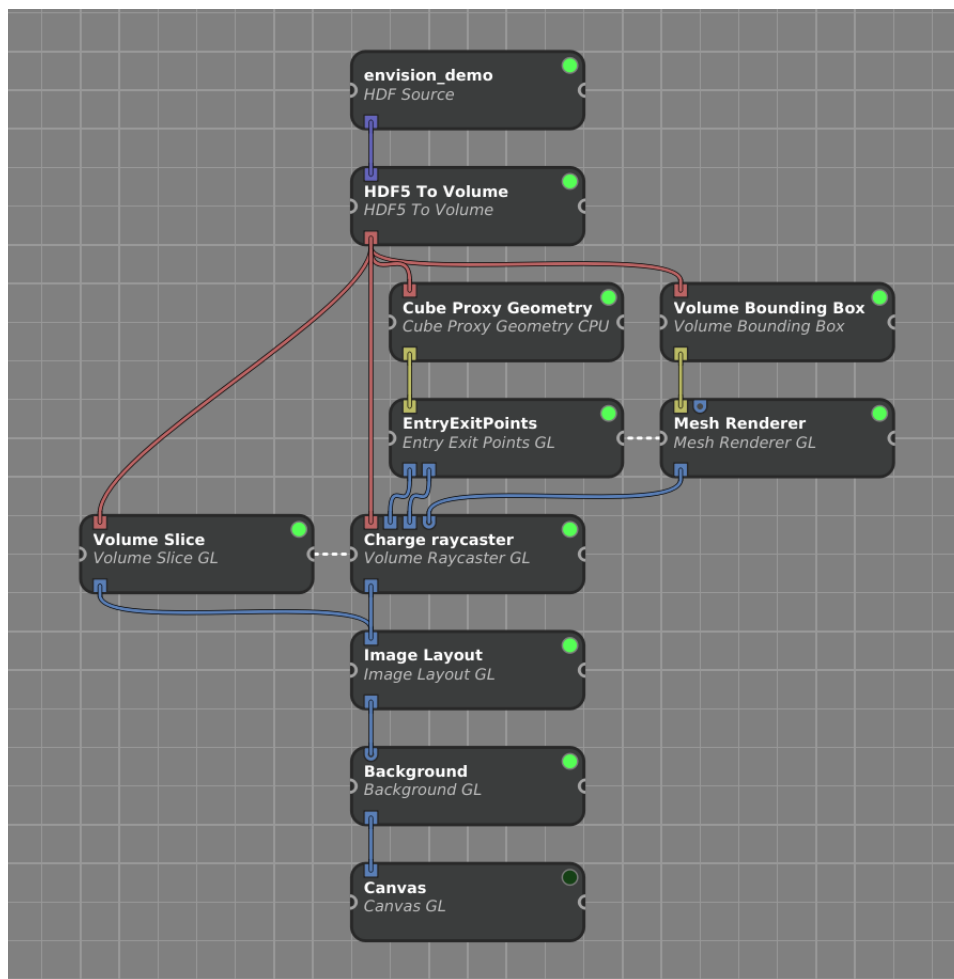Figure 6: Visualisation of charge density for NaCl with slice function in Inviwo.

Figure 7: Network for visualisation of charge density for NaCl with slice function in Inviwo.
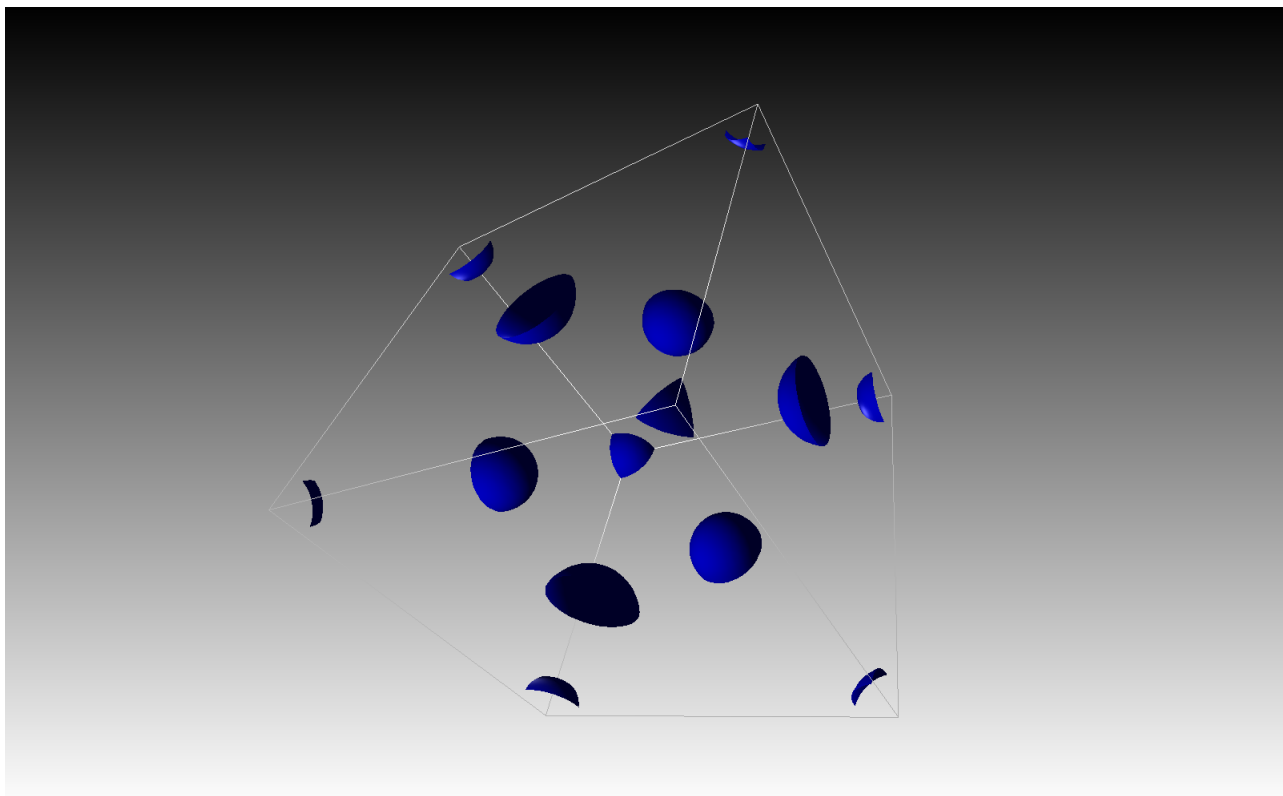
Figure 8: Visualisation of charge density for NaCl with ISO raycasting.
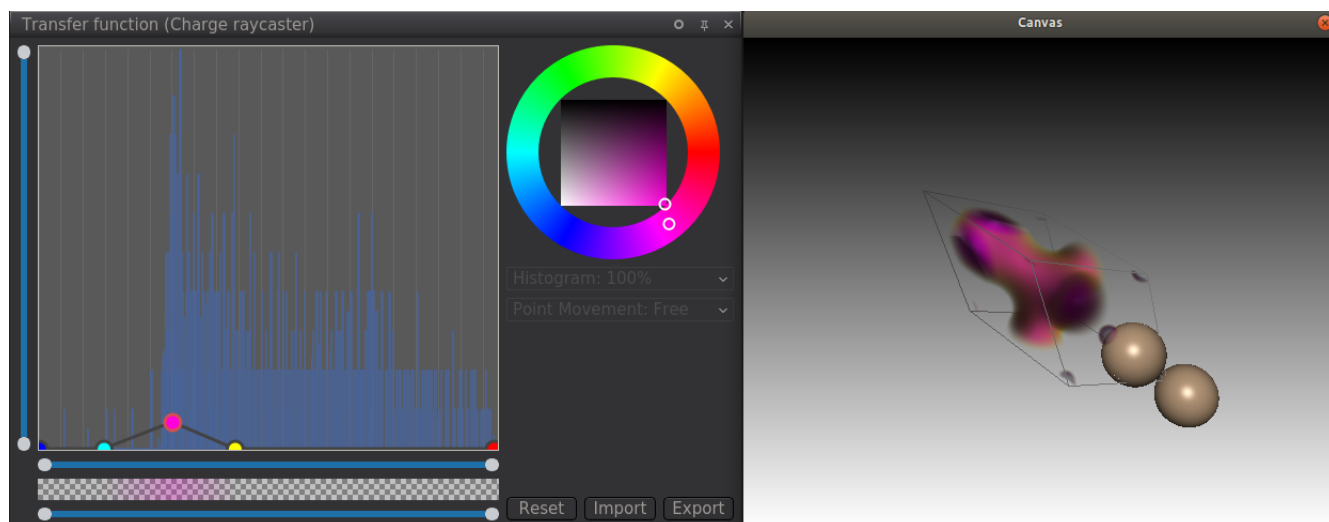
Figure 9: Transfer function of the Charge raycaster processor and visualisation of charge density of Si connected with unit cell visualisation. Charge density values between 0.15 and 0.45 are illustrated as a somewhat transparent purple volume.

4.4 DOS

In the Python editor in Inviwo, the user can choose to generate a visualisation of the density of states by opening the folder *scripts* and running the file *dos.py*.

Figure 10 is an example of a visualisation of the total density of states, in this case for Cu. Figure 11 shows the network for the visualisation. This network generates both a 2D-graph of the total density of states and a similar graph but for the partial density of states. The partial density of states is by default shown for one atom, in two graphs for spin-polarized calculations, determined by the argument *atoms* in the function *dos*. See chapter 4.5.3 for a description of how to pick the atom for which to show projected DOS.

Visualisation of the partial DOS for one specific orbital (s, p, d, or f) of an atom can be achieved by deleting the connections between the undesired processors of type "HDF5 To Function" and the corresponding processor with name "Partial Add". To do this, select a connection by clicking on it and press the Delete key.
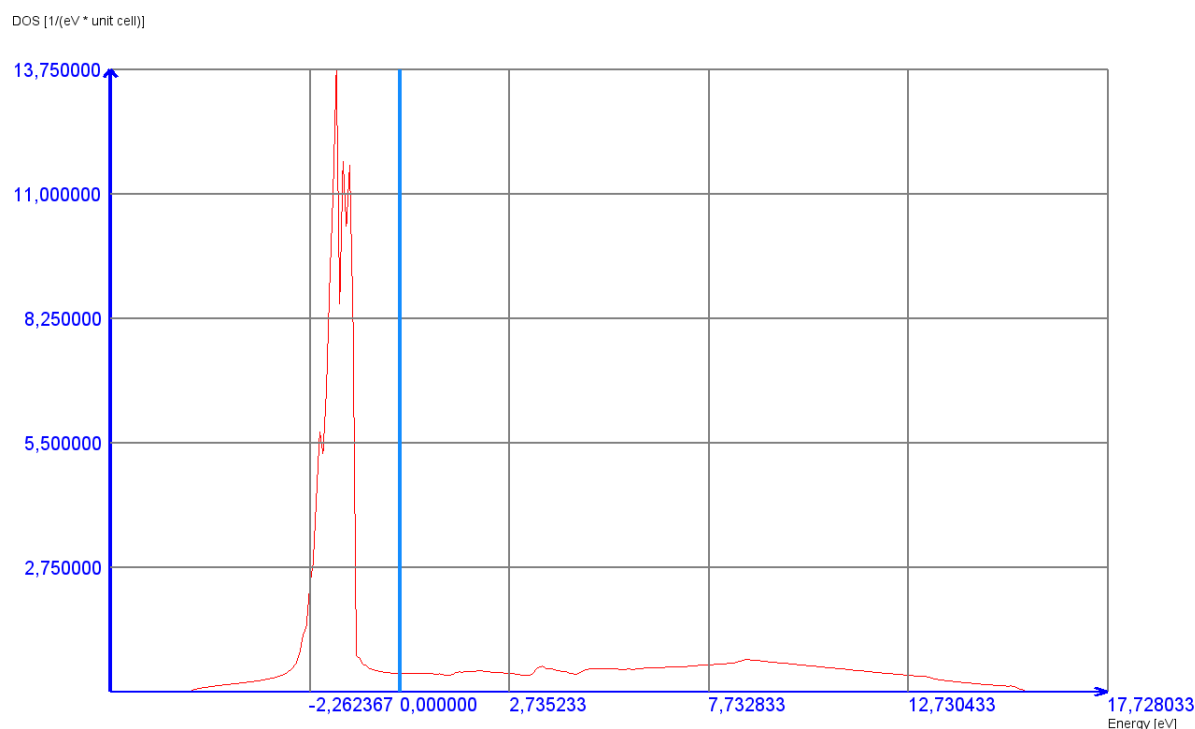
Figure 10: Visualisation of total DOS for Cu.
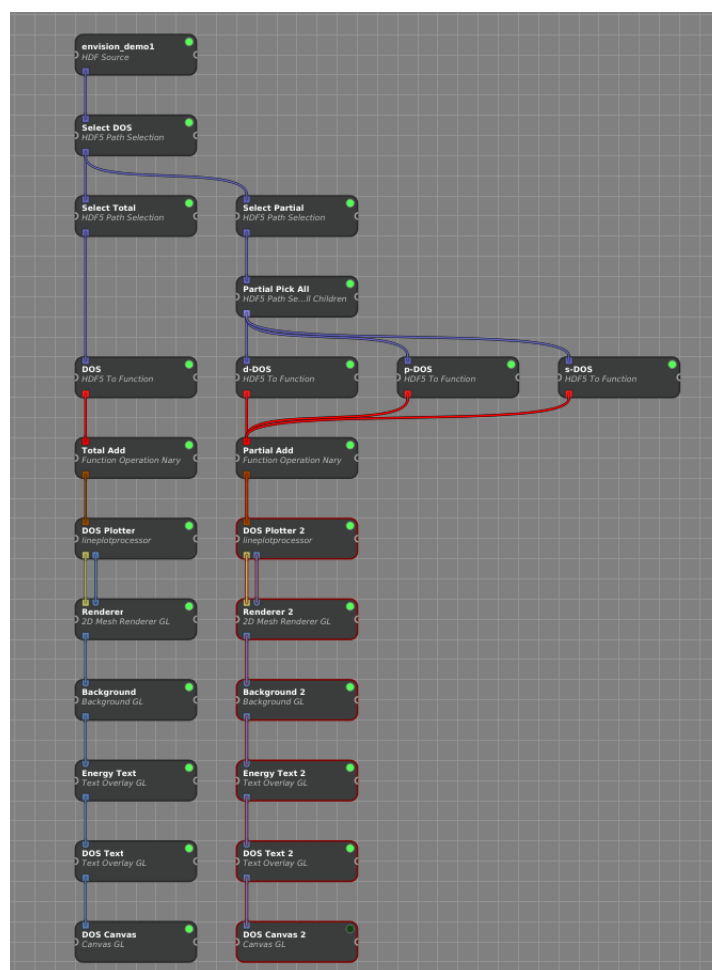
Figure 11: Network for visualisation of total and partial DOS for Cu.

4.5 Interconnected networks

Descriptions and examples of the python scripts generating interconnected networks can be seen below.

4.5.1 Unit cell and charge

In the Python editor in Inviwo, the user can choose to generate a visualisation of unitcell and charge by opening the folder *scripts* and running the file *unitcellAndCharge.py*.

4.5.2 Unit cell and ELF

In the Python editor in Inviwo, the user can choose to generate a visualisation of unitcell and ELF by opening the folder *scripts* and running the file *unitcellAndElf.py*. Figure 12 shows what this can look like.
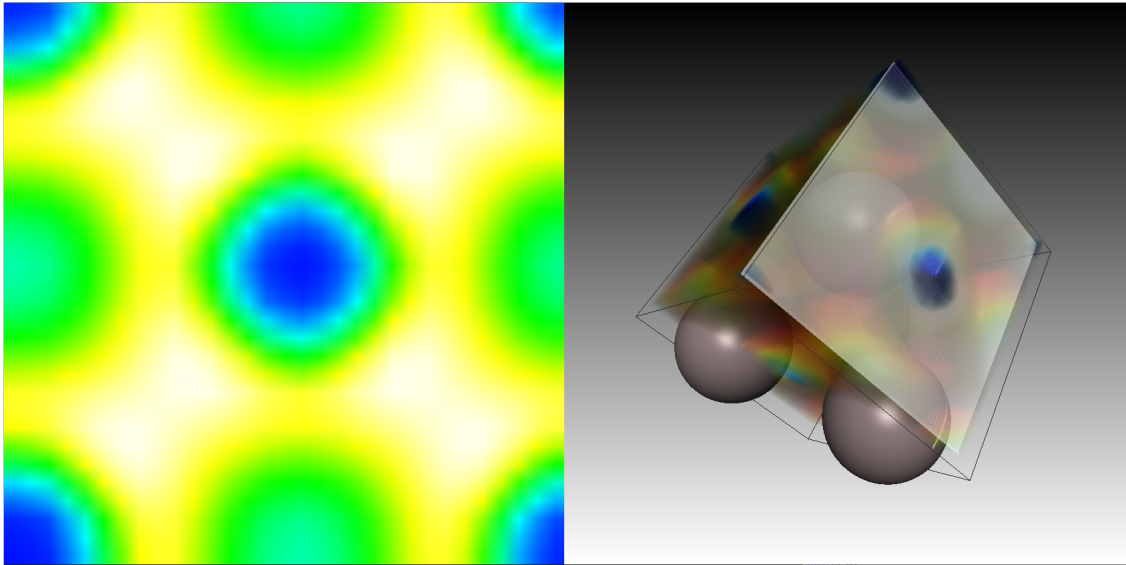
Figure 12: Unit cell and ELF data for aluminum visualised side by side, with volume raycasting and slice function.

4.5.3   Unit cell and DOS

In the Python editor in Inviwo, the user can choose to generate a visualisation of unitcell and density of states by opening the folder *scripts* and running the file *unitcellAndDos.py*. Here the user can pick an atom for which to show partial DOS by clicking on the desired atom in the unit cell. See figure 13 for an example of what this can look like.
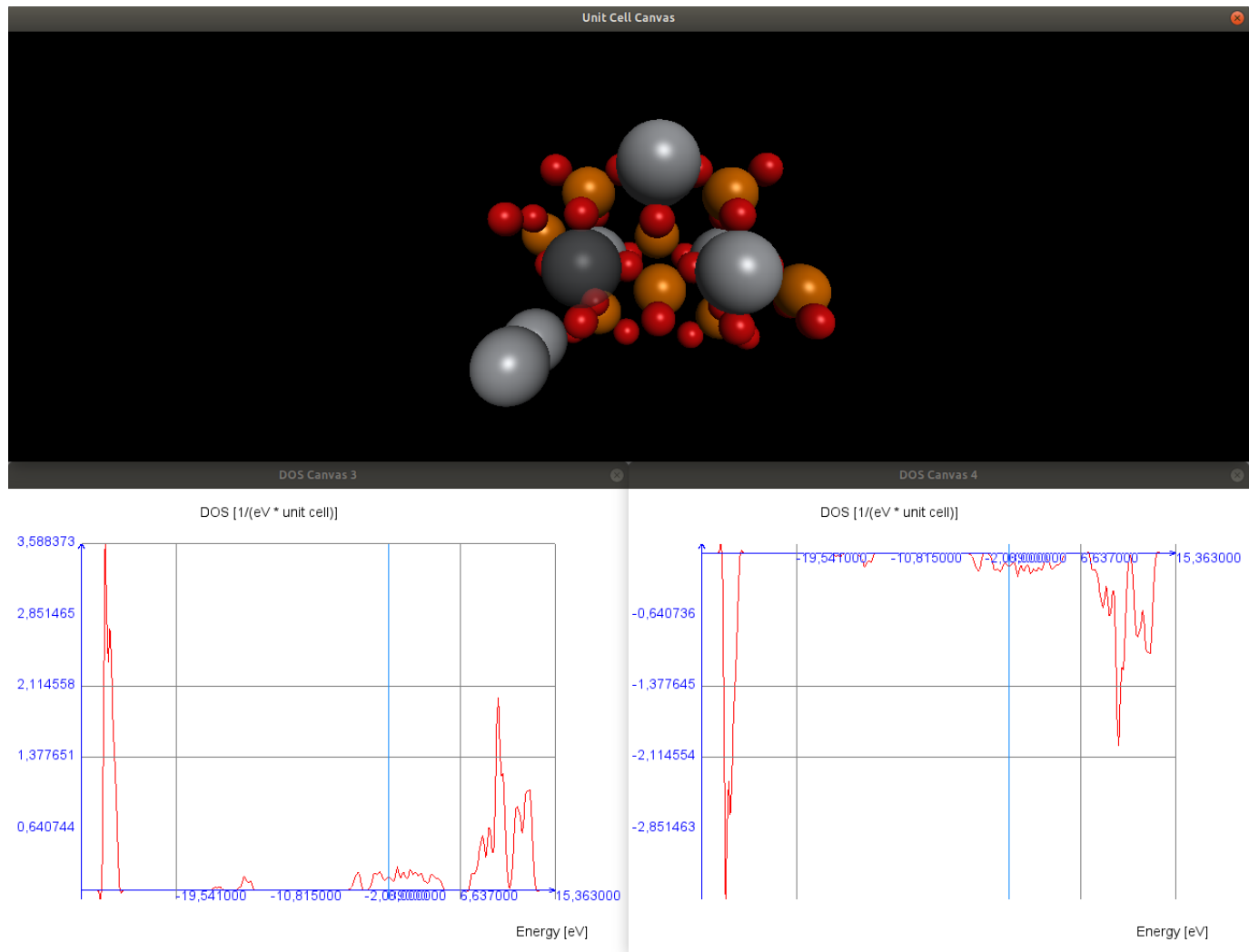
Figure 13: Visualisation of unit cell data and partial DOS of a titanium atom. The two graphs show DOS for electron spin up and electron spin down.

A Licens

2. Waiver. To the greatest extent permitted by, but not in contravention of, applicable law, Affirmer hereby overtly, fully, permanently, irrevocably and unconditionally waives, abandons, and surrenders all of Affirmer's Copyright and Related Rights and associated claims and causes of action, whether now known or unknown (including existing as well as future claims and causes of action), in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "Waiver"). Affirmer makes the Waiver for the benefit of each member of the public at large and to the detriment of Affirmer's heirs and successors, fully intending that such Waiver shall not be subject to revocation, rescission, cancellation, termination, or any other legal or equitable action to disrupt the quiet enjoyment of the Work by the public as contemplated by Affirmer's express Statement of Purpose.

3. Public License Fallback. Should any part of the Waiver for any reason be judged legally invalid or ineffective under applicable law, then the Waiver shall be preserved to the maximum extent permitted taking into account Affirmer's express Statement of Purpose. In addition, to the extent the Waiver is so judged Affirmer hereby grants to each affected person a royalty-free, non transferable, non sublicensable, non exclusive, irrevocable and unconditional license to exercise Affirmer's Copyright and Related Rights in the Work (i) in all territories worldwide, (ii) for the maximum duration provided by applicable law or treaty (including future time extensions), (iii) in any current or future medium and for any number of copies, and (iv) for any purpose whatsoever, including without limitation commercial, advertising or promotional purposes (the "License"). The License shall be deemed effective as of the date CC0 was applied by Affirmer to the Work. Should any part of the License for any reason be judged legally invalid or ineffective under applicable law, such partial invalidity or ineffectiveness shall not invalidate the remainder of the License, and in such case Affirmer hereby affirms that he or she will not (i) exercise any of his or her remaining Copyright and Related Rights in the Work or (ii) assert any associated claims and causes of action with respect to the Work, in either case contrary to Affirmer's express Statement of Purpose.

4. Limitations and Disclaimers.

a. No trademark or patent rights held by Affirmer are waived, abandoned, surrendered, licensed or otherwise affected by this document.

b. Affirmer offers the Work as-is and makes no representations or warranties of any kind concerning the Work, express, implied, statutory or otherwise, including without limitation warranties of title, merchantability, fitness for a particular purpose, non infringement, or the absence of latent or other defects, accuracy, or the present or absence of errors, whether or not discoverable, all to the greatest extent permissible under applicable law.

c. Affirmer disclaims responsibility for clearing rights of other persons that may apply to the Work or any use thereof, including without limitation any person's Copyright and Related Rights in the Work. Further, Affirmer disclaims responsibility for obtaining any necessary consents, permissions or other rights required for any use of the Work.

d. Affirmer understands and acknowledges that Creative Commons is not a party to this document and has no duty or obligation with respect to this CC0 or use of the Work.