Establishing and Evaluating Access to Scientific Data
EUDAT Human Brain Project
Big Data Bases and Cloud Services
Project Report

Tom Wiesing

May 29, 2016

[1] EdN:1

[1] EdNote: Make an abstract

Contents

1 Introduction

1.1 Overview

Multi-dimensional arrays arise naturally in multiple occasions and thus the paradigm of Array Databases has become more and more important. They can be used for a variety of scientific applications, ranging from satellite imagery, over medical imaging techniques to mathematically interesting objects.

One such array database is the Rasdaman system [Bau+98]. It promises to allow users to "storing and querying massive multi-dimensional arrays, such as sensor, image, simulation, and statistics data appearing in domains like earth, space, and life science" [Gmb16b].

In this project we want to establish and evaluate access to scientific data. In particular we want to insert this scientific data into the Rasdaman database and evaluate how well the database can handle this kind of data. In this case we want to work together with EUDAT and the Human Brain Project.

1.2 Collaborators

The Human Brain Project [Pro16b] is a European Commission Future and Emerging Technologies Flagship Project with the aim of "providing research infrastructure in the fields of Neuroscience, computing and brain-related medicine". In particular we collaborated with Huanxiang Lu and Dr. Sean Hill from the École polytechnique fédérale de Lausanne to gain access to the Human Brain Database. This archive of multiple sources is a database of (not only) human brain scans.

We also collaborated with Peter Wittenburg and Daan Broede from EUDAT [EUD16]. EUDAT is a "collaborative Pan-European infrastructure for research data services, training and consultancy" and will be used to host the data and resulting interfaces created in this project.

1.3 Project Components

Concretly this project consists of (1) gaining access to the scientific data provided by the Human Brain Database, (2) determining how (and if) this data can be represented inside the Rasdaman system, (3) developing a method to properly ingest the data into Rasdaman, (4) asking the collaborators about useful queries that can be performed on the data, (5) running the queries and gaining new insight into the data and finally (6) evaluating how well Rasdaman was able to deal with the provided data and developed queries.

The section of this report is as follows: In Section 2 we introduce in detail the provided dataset. We then continue in Section 3 by describing how we ingested the data into Rasdaman. Next we describe the queries that we developed over the course of this project in Section 4 before concluding with an evaluation and outlook for future work in Section 5.

2 The Dataset

2.1 The Human Brain Database Dataset

The dataset we have worked with for this project comes from the Human Brain Database project. In total it is about 1 TB in size. It consists of a collection of different brain scans from different sources, each of different resolutions and shapes. Due to the size of this dataset we have not yet accessed all of it.

So far we have only worked with a small subset in order to test our methods. This subset is significantly smaller and is sourced from the SPM Anatomy toolbox [Jül16]. It has a size of under 10 MB and is a brain atlas consisting of 27 different brain scans. Each scan has a resolution of about 150 x 150 x 200 pixels.

2.2 Neuroimaging Methods

Before we describe the dataset in more detail we briefly introduce the reader to the most commonly used neuroimaging methods. This provides the neccessary background to understand what the data means. Neuroimaging is the process of scanning the brain and generating images of it. There are two different methods that are commonly used to achieve this.

X-ray computed tomography (commonly known as CT scan) is a method which produces multiple two-dimensional X-Ray images of an object to be scanned. With the help of computers these images are then assembled into a full three-dimensional scan. This allows to look inside an object without having to physically disassemble or open it.

Another common technique is called Magnetic resonance imaging and also known as an MRI scan. MRI scans do not use X-Rays to investigate an object but instead abuse the quantum spin of atoms. By applying a magnetic field, the spins of atoms are forced to align. Once this magnetic field is removed, the atoms return to equilibrium and produce very faint RF emissions. These can be detected, measured and then assembled into a three-dimensional image.

Both methods produce a 3-dimensional cube with scalar values at each point as the output, although MRI scans usually give higher resolution than X-Rays. For this reason our dataset consists mostly of MRI scans.

2.3 The HDF Format And BBIC Encoding

When we first received the dataset it was provided in HDF5 format. HDF stands for Hierarchical Data Format and is a data format that was originally developed at National Center for Supercomputing Applications and is now mainted by the HDF Group [Gro16]. It is designed to store and organize large amounts of data and has two different kinds of objects, datasets and groups. Datasets are multi-dimensional arrays of homogeneous type. Groups are containers for datasets and sub groups. This results in a filesystem-like structure for HDF5 files.

Even though the data is stored inside an HDF file, it is encoded using the BBIC format. This is an internal format used mostly for an image viewer by our collaborators. Our collaborators stated that "the bbic format contains a stack of tiled images of different level/resolution. Our image viewer works just like the Google Maps. Every time we zoom in or out, the viewer sends a query to the image service to retrieve certain tiles of images of certain slice at certain resolution" [Lua]. This makes it very difficult for us to extract the original three-dimensional cube that represents the brain scan. What makes it even more difficult is the fact that there is very little documentation available because the "bbic format was developed by a former colleague who has left for long time before I joined" [Lub].

2.4 The ATLAS Viewer

The BBIC format is used for the so-called ATLAS viewer. It is based on the imaging service mentioned above and is available at [Pro16a]. The viewer allows the user to interactively explore a subset of the Human Brain Database.

The atlas viewer has three main features: (1) selection of different brain scans, (2) looking at different two-dimensional cuts through the original data (Figure 1) and (3) overlaying of brain region masks (Figure 2).

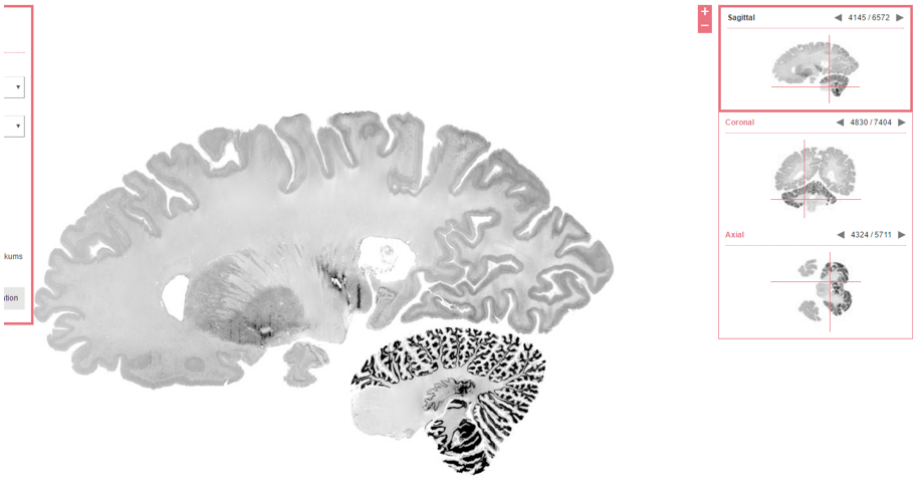

Figure 1: A screenshot of the ATLAS viewer showing a basic cut through a single dataset.

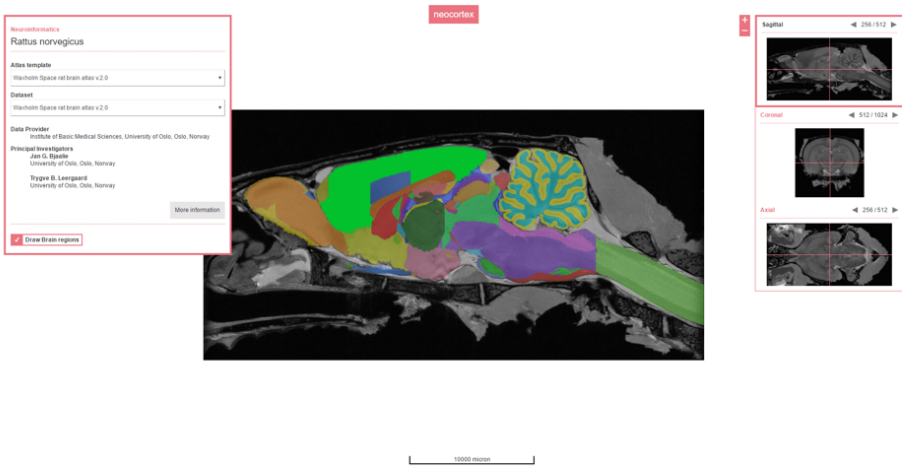

Figure 2: A screenshot of the ATLAS viewer showing an overlayed brain region mask.

3 Data Ingestion

3.1 The NIfTI Format

After viewing the image viewer it turned out that the HDF / BBIC encoded data is not the original data the brain scans were generated from. It was generated from files in NIfTI format.

NIfTI [Ini16], or Neuroimaging Informatics Technology Initiative, is a format created by a group of the same name. It is very commonly used for Neuroimages. As suggested by the name it is commonly used as the format of neuro images. Each file contains a single multi-dimensional array of homogeneous type as well as domain specific meta-data.

This format is a huge improvement on the BBIC format because it gives us direct access to the data we want to ingest into Rasdaman. In our case this gave us direct access to a three-dimensional array of floats. This allowed us to ingest the data in a collection of type `FloatSet3` [1].

Rasdaman does not support this format natively. Hence in order to ingest the data we need to find a way to access the data format and then convert it into a format that Rasdaman can understand.

[1] Rasdaman organises the multi-dimensional arrays it stores in so-called collections. Each collection corresponds to a list of multi-dimensional arrays of the same type. The type `FloatSet3` corresponds to a collection of three-dimensional arrays with floating point values.

3.2 Reading Data From Python

In order to access the data we used Python. A convenient library for acessing neuroimaging data called Nibabel is available from [BHL16]. It provides interoperatbility with the Numpy / SciPy stack and is thus perfeclty suited for our purposes. This allowed us to read in and access an image file via the following code:

```
>>> import nibabel as nb # Load the NiBabel Library
>>> import numpy as np # Load the Numpy Library
>>>
>>> image = nb.load("colin27_T1_seg.nii") # load the image file
>>>
>>> data.dtype # print the datatype of the image
dtype('float64')
>>> data.shape # print the data shape
(151, 188, 154)
```

3.3 Suitable Data Formats For Rasdaman

In terms of formats that Rasdaman supports that can be used to read in the data, two different formats come to mind, CSV and TIFF.

CSV, or comma seperated values, is a format that encodes up to two-dimensional data inside an ASCII file. For this it turns the data values into strings and joins them with commas. In the case of two-dimensional data, each line represents a single row of the dataset. This is a very wasteful encoding and unfortunatly not well suited for our data because it is three-dimensional. While Rasdaman supports three-dimensional CSV, it is non-standard and would thus proved to be too difficult to use.

TIFF, or Tagged Image File Format, is a lossless image raster format that was originally created by Aldus Corporation and is now maintained by Adobe [Ado16]. TIFF in theory supports both two and three-dimensional images however Rasdaman only supports the former. Furthermore it is possible to write to TIFF files from python using a library called `tifffile.py` available at [Goh16].

3.4 From Python To Rasdaman

In order to work around the fact that Rasdaman supports only two-dimensional TIFF files, we decided to import the data using so-called partial updates. This involves slicing the three-dimensional array into two-dimensional TIFF files which can then be imported on by one into Rasdaman. Using the *rasql* tool to insert data into Rasdaman this involves three steps.

```
# Step 1: Create a collection of type FloatSet3 inside Rasdaman
rasql -q "create collection NAME FloatSet3"

# Step 2: Insert an (empty) multi-dimensional array into the collection
```

```
rasql -q "insert into NAME values marray it in [0:0,0:0,0:0] values 0f"

# Step 3: Read each TIFF slice using partial updates
rasql -q "update NAME as c set c[INDEX,*:*,*:*] assign inv_tiff(\$1)"
    --file FILENAME.tiff
```

In the first step we create a collection of type `FloatSet3` inside Rasdaman. For this we need to give it a name (we just used a placeholder) and the collection type. In the second step we create an empty two-dimensional array inside the collection. In the third step (which we need to repeat for each slice) we update a single slice of the data cube with data read from a tiff file.

3.5 Automating Ingest

Since we usually have several hundered slices we do not want to do this manually. Thus we have written a python script that combines this process into a single step. The script consists of about 200 lines of python code and is well documented. It is called `nii_to_tiff.py` and concretly performs the following steps:

1. Reads a NIfTI file,
2. splits it into different slices and store each one as a TIFF file,
3. output a bash script to STDOUT which calls `rasql` with the arguments mentioned above to import the slices and
4. delete the TIFF files since they are no longer needed.

In order to import a single file we can then use the command

```
python3 nii_to_tiff.py FILENAME . | rasql_coll=COLLECTION
    rasql_args="--quiet" bash
```

4 Querying the Dataset

4.1 Slicing An Image Cube

Now that the data has been ingested into Rasdaman we proceeded with writing simple queries. Since Rasdaman is an Array Database system which has been used for multi-dimensional data we first wanted to start this part of the project by reproducing the functionaility of the ATLAS viewer described above. For this we want to start with simple slicing operations, namely just extracting Sagittal, Coronal and Axial[2] slices from the original scan.

Rasdaman is capable of expressing these cuts easily. Each of them is parameterized by two parameters, the two axes it is parallel to and the position within the third axis where to cut. In the query language this uses a slicing operation and is written as `select it[INDEX,*:*,*:*] from NAME as it`. Here `INDEX` refers to the position of where to cut and `NAME` refers to the name of the collection that contains the data cube.

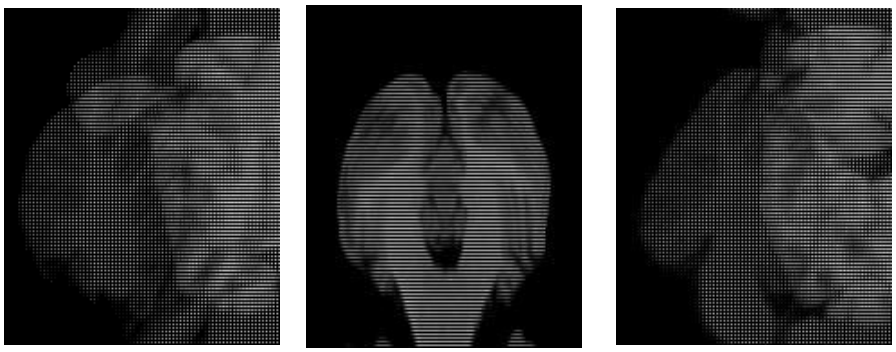

Figure 3: Simple cuts through the example dataset.

Additionally in order to display the output from this operation we need to choose an image format to display it in. To save such a cut as a png image a command like

```
rasql -q "select encode(it[INDEX,*:*,*:*],\"png\") from NAME as it"
    --out file --outfile FILENAME
```

can be used. A sample output of this operation can be found in Figure 3.

4.2 Overlaying Brain Region Masks

The second operation of the ATLAS viewer that is interesting to reproduce is the overlaying of Brain Region Masks onto the original scan. This operation is

[2]Sagittal, Coronal and Axial cuts are domain specific terms for cuts parallel to the axes of the data cube.

more complicated than the simple slicing operation and any query performing it needs to perform the following three steps:

1. retrieve the brain scan data
2. retrieve the brain mask data
3. overlay the mask over the brain data

A command similar to the following could be used to produce a single Brain region mask overlay:

```
rasql -q "select encode(S[INDEX,*:*,*:*] overlay M[INDEX,*:*,*:*],
    \"png\") from MASK_COLLECTION M, SCAN_COLLECTION S" --out file
    --filename FILENAME
```

In this command we assume that the brain scan data and the mask data are available in the collections `SCAN_COLLECTION` and `MASK_COLLECTION` respctively. Then we use the `overlay` operator to overlay the two images.

Figure 4: Example Brain Mask Data

In practive the second step of retrieve brain mask data provides significant difficulties. Even though the brain mask data is available in principal it is only given in the form of an SVG image. Unlike the brain scan data, this is a vector image format which does not store the rasterized representation as would be needed by Rasdaman. On top of this the SVG images do not contain three-dimensional data but are instead structured similar to the BBIC format. For each possible cut and zoom level a seperate image is provided. One of these image can be found in Figure 4. Unfortunatly these problems stopped us from properly ingesting the brain mask data into Rasdaman.

4.3 Extended Slices

The final type of query we wanted to experiment with expands on the slicing explained above. Instead of only allowing cuts parallel to two of the three axes we wanted to allow any type of two-dimensional cuts. While this is not available in the ATLAS viewer it is an enhancement our collaborators have been thinking about. Mathematically speaking such a cut consists of an arbitrary two-dimensional plane intersecting a three-dimensional cuboid. As such it is parametrised by 6 parameters, one (three-dimensional) point to start the plane at and three angles in each direction to rotate the plane along.

This mathematical description gives us one natural way to represent this query inside Rasdaman: Rotate the entire cube according to the three angles given above and then perform a simple cut as described above. Unfortunatly there is no rotation operation inside Rasdaman, but according to [Gmb16a] it might be added in the future.

There is a second way to represent these queries inside Rasdaman. Outside of Rasdaman one can compute the coordinates that would be included in the plane and then query the database for those exact values. Even though this could be simplified by using arrays to index the data cube, this still requires some non-trivial computational effort outside of Rasdaman and thus we did not perform this query either.

5 Outlook & Conclusion

[2] EdN:2

[2] EdNote: Write this

References

[Ado16] Adobe. *Developer Resources*. 2016. URL: http://partners.adobe.com/public/developer/tiff/index.html (visited on 05/27/2016).

[Bau+98] Peter Baumann et al. "The Multidimensional Database System RasDaMan". In: *Proceedings ACM SIGMOD'98*. Seattle, Washington, USA, June 1998.

[BHL16] Matthew Brett, Michael Hanke, and Eric Larson. *Neuroimaging in Python – NiBabel 2.0.3dev documentation*. 2016. URL: http://nipy.org/nibabel/ (visited on 05/27/2016).

[EUD16] EUDAT. *EUDAT - Research Data Services, Expertise & Technology Solutions*. 2016. URL: https://www.eudat.eu/ (visited on 05/27/2016).

[Gmb16a] rasdaman GmbH. *Issue 228 – Add function in rasql to rotate objects*. 2016. URL: http://www.rasdaman.org/ticket/228 (visited on 05/29/2016).

[Gmb16b] rasdaman GmbH. *the rasdaman raster array database – rasdaman*. 2016. URL: http://www.rasdaman.org/ (visited on 05/27/2016).

[Goh16] Christoph Gohlke. *tifffile.py*. 2016. URL: http://www.lfd.uci.edu/~gohlke/code/tifffile.py.html (visited on 05/27/2016).

[Gro16] The HDF Group. *HDF5*. 2016. URL: https://www.hdfgroup.org/HDF5/ (visited on 05/27/2016).

[Ini16] Neuroimaging Informatics Technology Initiative. *NIfTI-1 Data Format*. 2016. URL: http://nifti.nimh.nih.gov/nifti-1 (visited on 05/27/2016).

[Jül16] Forschungszentrum Jülich. *SPM Anatomy toolbox*. 2016. URL: http://www.fz-juelich.de/inm/inm-1/EN/Forschung/_docs/SPMAnatomyToolbox/SPMAnatomyToolbox_node.html (visited on 05/27/2016).

[Lua] Huanxiang Lu. *The BBIC data format*. private communication.

[Lub] Huanxiang Lu. *The BBIC developers*. private communication.

[Pro16a] Human Brain Project. *Atlas Viewer*. 2016. URL: https://nip.humanbrainproject.eu/atlas/ (visited on 05/27/2016).

[Pro16b] Human Brain Project. *The Human Brain Project*. 2016. URL: https://www.humanbrainproject.eu/ (visited on 05/27/2016).