Anomaly detection for treatment planning and a learning health system in radiotherapy

Vincent Qi, Daniel Yuan
Mentors: Dr. Todd McNutt, Pranav Lakshminarayanan

Spring 2018

Computer Integrated Surgery II, 601.356/456/656
Dr. Russell Taylor, Ehsan Azimi
Johns Hopkins University

Contents

I. Introduction

A. Background

In the past century, the average lifespan of an individual has increased significantly due to medical advances and improved healthcare accessibility. However, with this increase in average lifespan comes an increase in the probability of random oncogenic (cancerous) mutations occurring. Studies show that the median age of cancer diagnosis is 66 years and that over 75% of all cancers are detected in patients past the age of 55.

In past 5 years in the United States, cancer accounts for 22.5% of all total deaths. As such, cancer diagnosis, prognosis, and treatment are becoming increasingly important medical problems. Currently, there are several main forms of cancer treatment. These are surgery, radiotherapy, and chemotherapy. There are also many initiatives supporting research and development targeted/precision therapy, hormone therapy, and stem cell transplantation.

One of the most common types of treatment is radiation therapy. Almost 35% of cancer patients have radiation therapy as a component of their primary treatment plan. Radiation therapy involves bombarding cells with high energy electromagnetic radiation, such as x-rays, in order to cause cell death by damaging the DNA of the cell and preventing cell division. However, radiotherapy has several drawbacks. High energy waves are not selective and can kill healthy cells in the neighborhood of the target location. If the target location has sensitive organs and tissue, such as the reproductive system or spine, then any side effects could cause major patient mental and physical trauma.

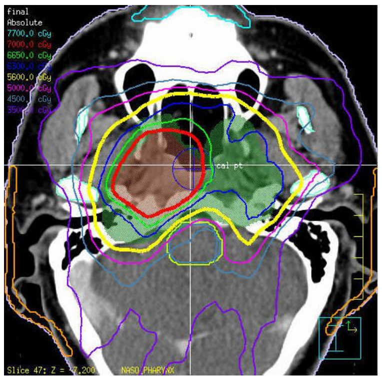


Figure 1: Nasopharynx IMRT treatment

Source: http://images.iop.org/objects/med/talkingpoint/4/2/1/imrt2_1602.jpg

One improvement modern radiotherapy implements is 3-dimensional radiotherapy. 3D radiotherapy (3D-RT) involves taking a 3D image of the tumor and region of interest, mapping the organ and tumor contours and designing radiation beams to conform to tumor shape and patient anatomy. This results in increased accuracy and consistency. Another improvement is intensity modulated radiotherapy (IMRT). IMRT delivers precise radiation doses to specific regions using computer controlled linear accelerators based on patient morphology and oncologist expertise. An example can be seen in Figure 1.

B. Problem

Many decisions and treatments that oncologists and surgeons have to make are dependent on the collected patient data. For example, 3D-RT and IMRT requires contour models of the tumor region. Therefore, any low-quality contour maps will increase the risk involved for the cancer patients. The current issue with data collection is that it involves many manual human involved steps. There is also a lack of quality and integrity validation for collected data sets. Some examples of manual steps include splicing contour slices together, contouring regions, collection of patient assessments, and patient treatment plans. Since data collection is a repetitive and tiring process, it can be prone to human error. Technicians could build contour models that are missing slices and physicians could input incorrect patient variables.

There exists some research into creating generalized standards for oncologic data collection based off of previously collected data and standardized variables, however, there is almost no existing software that has an active approach to the problem. We aim to be able to build a framework to allow a module and active approach for integrity checking of mapped organ contours to detect these anomalies. By using updated and live clinical databases combined with this framework software, our system allows the user to implement unique integrity checks that checks various patient data for validation.

II. Technical Approach

We developed a code framework that allows users to identify potentially anomalous data using their own integrity algorithms. The framework also allows for simple statistical analysis of the results. A generalized description of the automated workflow for our framework can be seen in figure 2. First, we query the database and isolate the appropriate patient data to be validated. Next, we pass the data to the validation modules. The validation modules analyzes and returns whether the sample is anomalous or not. The collected results are then analyzed using various statistics, such as measuring the true/false positive and true/false negative rates.

In figure 3, we can see a basic UML diagram that describes the actual framework code base. There is a main interface that takes the users decisions as command line arguments. The interface passes the specific user case to the engine. The engine queries the database for the appropriate data, which is then passed to a module manager. The module manager selects the specific integrity module that the user wants for validation and passes the data to it. It then returns whether the data was an anomaly back to the user. If the user wants to run statistics, the engine also calls a statistics manager. There are many other supporting objects and relationships that are not covered for sake of visual simplicity.

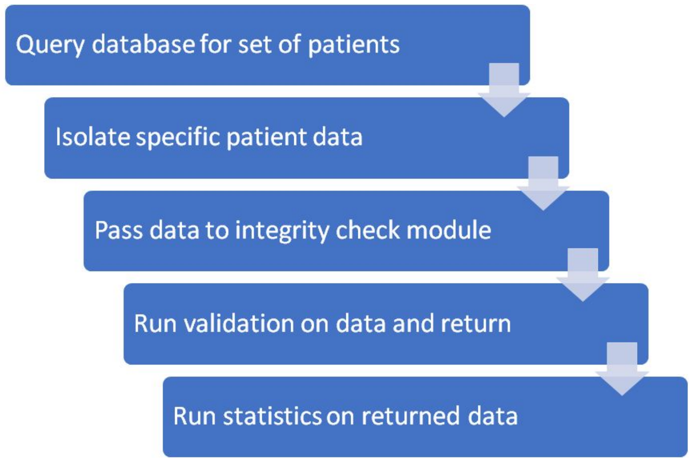

Query database for set of patients
Isolate specific patient data
Pass data to integrity check module
Run validation on data and return
Run statistics on returned data

Figure 2: Generalized Framework Workflow

The framework is designed to allow for modular insertion of various detection rules. The code base has a generalized class format for integrity modules for the user to follow. The user can call the integrity module superclass for their integrity module. The only method they need to implement is an algorithm to determine whether the data is valid or not. The data is passed in as a parameter to the method. The method then returns a validation value (usually true/false). The framework allows users to sample individual patients and detection methods or test over a battery of tests at once.

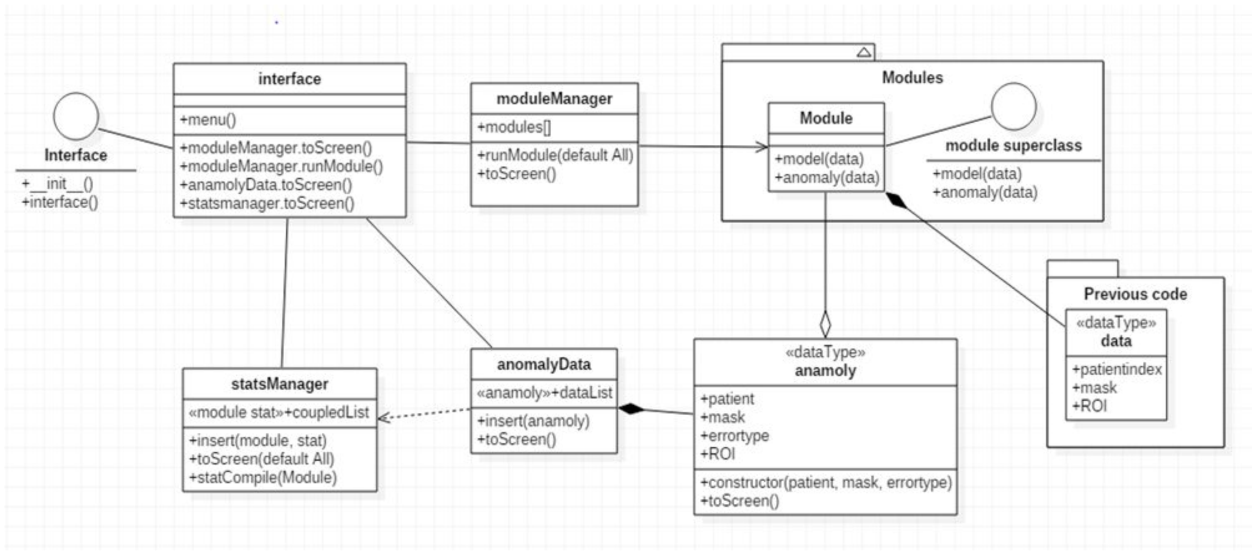


Figure 3: Simplified Framework UML Diagram

Beyond the integrity algorithm, there is one other area that the user may want to modify. This is the specific type of data passed down through the framework. Since different databases will contain different patient information, there a specific section the engine the can be modified to select the data. It can also be easily converted to a method which can be written outside the framework in a seperate user generated file and then passed in as a command line argument. Figure 4 demonstrates one possible example of data from the database, a contour mask for the oncospace database.

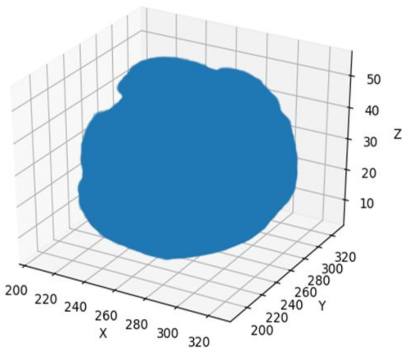
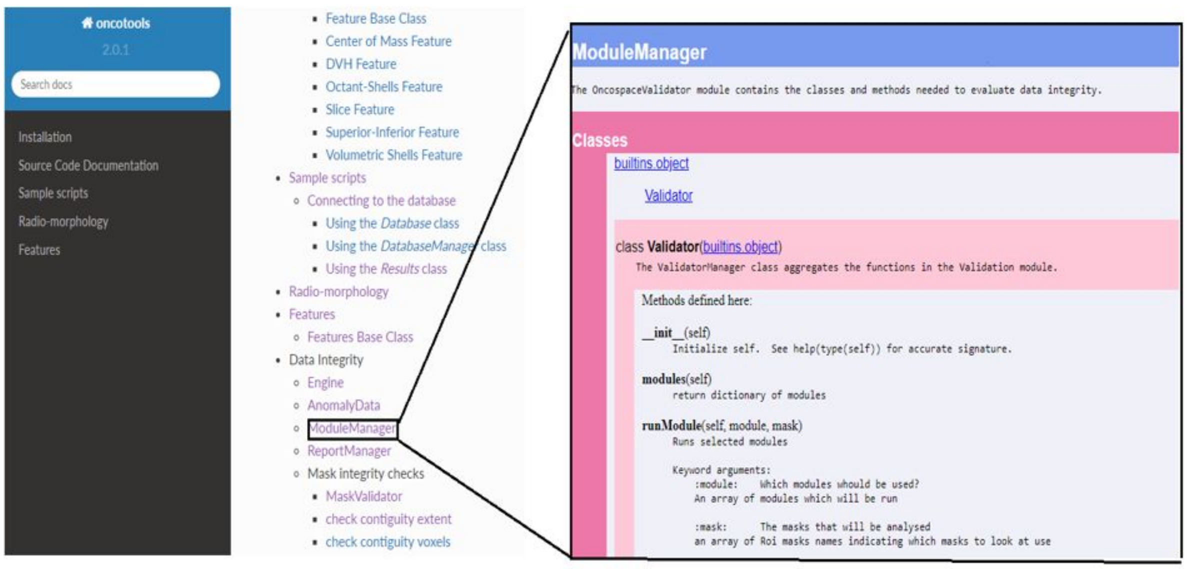


Figure 4: Example of contour mask

We selected to use Python for two main reasons. First, we wanted to conform with the oncotools package developed previously by the McNutt Lab. Our database querying uses prebuilt code from the package. We also wanted to be able to have our code stay incorporated as part of the oncotools package for future users. Second, for user convenience. Python has a very large community that supports lot of public prebuilt packages available that do a lot of the integrity checking algorithms that users may want to use.



ModuleManager

```
The OncospaceValidator module contains the classes and methods needed to evaluate data integrity.
```

Classes

builtins.object

Validator

class **Validator**(builtins.object)

```
The ValidatorManager class aggregates the functions in the Validation module.
```

Methods defined here:

__init__(self)

```
Initialize self.  See help(type(self)) for accurate signature.
```

modules(self)

```
return dictionary of modules
```

runModule(self, module, mask)

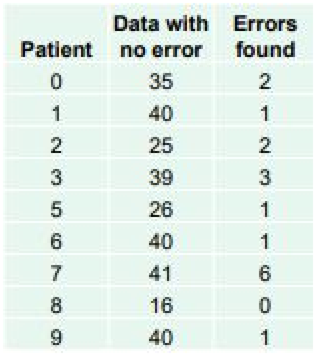
```
Runs selected modules

Keyword arguments:
    :module:    Which modules whould be used?
    An array of modules which will be run

    :mask:      The masks that will be analysed
    an array of Roi masks names indicating which masks to look at use
```

Figure 5: Expanding oncotools documentation to include our framework

To demonstrate the functionality of the framework we developed some of our own detection modules as well as inserting pre-existing error modules. The framework combined with our own algorithms allowed us to explore areas where anomalies occur. They can also serve as examples for future users.

III. Results and Significance

We were able to create a framework that allows for the insertion of various detection algorithms. Most of the process is automated, as long as the location of the detection algorithms is added to the manager class for the framework data can be passed to the algorithm. The user only needs to write the actual detection method with a structured input and output format, give it a label, and place it in the same location as other existing modules.



Patient	Data with no error	Errors found
0	35	2
1	40	1
2	25	2
3	39	3
5	26	1
6	40	1
7	41	6
8	16	0
9	40	1

Figure 6: Example of table of errors found for individual patients

We were able test our framework with several of our own detection algorithms in order to validate that it works. As seen in figure 6 running the framework on a sample of the database each data passed into the detection modules will return if an error has been detected or not detected. The example shown in figure 6 is the accuracy of the contour maps for the first 10 patients of the database. As can be seen above, there is about a 6% chance for a contour map to be have errors in contiguity using a simple extant algorithm. Another example is a simple drug dosage check. We were able to go through the patients and determine if certain patients had errant dose grids, such as dose grids with no data. Testing demonstrated that the algorithm worked, but after running through the database, we were unable to find patients with incorrect grids. However, we were able to determine patients with no grids. We noticed that some integrity checking modules took significantly longer to run than others.

As a general rule, we were able to query the database for all 1275 patients and run them through the framework to determine the existence of anomalous contour maps using the detection algorithms currently implemented as well as determining which patients had anomalous dosing maps and questionable assessments. For detection modules that were more complex, such as voxel analysis, we narrowed the sample size according to the runtime of one patient analysis.

IV. Conclusion

The framework we have built will allow oncologists to automatically determine patients with anomalous data. They will be able to implement their own algorithms, whether it be simple statistical analysis or machine learning, and find the patients of interest to them. They will also be able to actively check new patients using the framework. We have shown that our framework works with several examples. By using the implemented framework, the quality of data used by physicians to make decisions should increase and better the lives of radiation therapy patients.

V. Management Summary

Responsibilities:

Vincent worked on framework design, implementation, and testing. Daniel worked on anomaly detection methods, framework implementation, and testing.

Deliverables: Accomplished vs. Planned

Minimum:

- Working framework that allows for modular insertion of new integrity checks
- Documented API to develop new integrity checks

Expected:

- Incorporate existing errant detection modules into our working framework
- Implement new anomaly detection modules

Maximum:

- Develop and implement numerous new integrity checks
- Implement compatibility packet to allow other programs access to results easily

We were able to accomplish all of our minimum and expected deliverables. Our framework allowed for the insertion of new integrity checks and is well documented for others to develop their own integrity checks. Our framework has previous existing errant detection modules integrated as well as new anomaly detection modules we developed as well. For maximum deliverables, our code is implemented so that it is easily compatible with other programs and allows easy access to results. For example, many results can be output and written into a text document. We were able to implement our own novel integrity check modules for the maximum deliverable but were not able to implement as many integrity check modules as we would have wanted to initially due to time restrictions and computational limits.

What might be next:

The Oncospace group wants to implement their own error detection modules. The next step would be to implement these error detection modules. We also would like to enable physicians to analyze more types of erroneous data by expanding the scope of our framework. Further improvements include increasing the flexibility of framework to enable more complex input and output, such as a scale of the likelihood of a data point to have errors. Finally, we would like to optimize of the code for efficiency for running over a large database such as implement parallelism and improve query structures.

What we learned:

Our initial design wanted to be very flexible and robust. However, the complexity added a lot of complicated relationships to the framework, make the code harder to implement. Even with structured communication between team members, there can be a lot of confusion and errors due to the framework structure. We found the more flexible we tried to make the framework, the harder it became to implement and the more cases we needed to take into account. We also learned that certain detection algorithms we wanted to implement were highly computationally intensive and a balance of memory usage and speed are required for optimization. Additionally, traversing across a database is highly memory intensive and a balance of memory usage and speed in required for optimization and testing.

VI. Acknowledgements

We would like to thank our mentors Dr. Todd McNutt and Pranav Lakshminarayanan for providing us with their guidance throughout the project period. We would also like to thank Dr. Russell Taylor and Ehsan Azimi for reviewing our work in Advanced Computer Integrated Surgery.

VII. References

1. Risk Factors: Age - Age and Cancer Risks. Retrieved from https://www.cancer.gov/about-cancer/causes-prevention/risk/age
2. Cancer Facts & Figures 2018. (n.d.). Retrieved from https://www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/cancer-facts-figures-2018.html
3. Cancer Incidence by Age. Retrieved from http://www.cancerresearchuk.org/health-professional/cancer-statistics/incidence/age#heading-Zero
4. Data Integrity Systems for Organ Contours in Radiation Therapy Planning (submitted)
5. Gaffney, D. K., King, B., Viswanathan, A. N., Barkati, M., Beriwal, S., Eifel, P., . . . Bosch, W. (2016). Consensus Recommendations for Radiation Therapy Contouring and Treatment of Vulvar Carcinoma. International Journal of Radiation Oncology*Biology*Physics, 95(4), 1191-1200. doi:10.1016/j.ijrobp.2016.02.043
6. McNutt TR, Benedict SH, Low DA, Moore K, Shpitser I, Jiang W, Lakshminarayanan P, Cheng Z, Han P, Hui X, Nakatsugawa M, Lee J, Moore JA, Robertson SP, Shah V, Taylor R, Quon H, Wong J, DeWeese T, Using Big Data Analytics to Advance Precision Radiation Oncology, International Journal of Radiation Oncology • Biology • Physics (2018), doi: 10.1016/ j.ijrobp.2018.02.028.
7. Nicholas, Hannah. Reviewed by Timothy J. Legg. (2017) The top 10 leading causes of death in the United States. Retrieved from https://www.medicalnewstoday.com/articles/282929.php
8. Radiation Therapy for Cancer. (n.d.). Retrieved from https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy/radiation-fact-sheet
9. Smith, B. D., Haffty, B. G., Wilson, L. D., Smith, G. L., Patel, A. N., & Buchholz, T. A. (2010). The Future of Radiation Oncology in the United States From 2010 to 2020: Will Supply Keep Pace With Demand? Journal of Clinical Oncology, 28(35), 5160-5165. doi:10.1200/jco.2010.31.2520
10. Sun, Y., Yu, X., Luo, W., Lee, A. W., Wee, J. T., Lee, N., . . . Ma, J. (2014). Recommendation for a contouring method and atlas of organs at risk in nasopharyngeal carcinoma patients receiving intensity-modulated radiotherapy. Radiotherapy and Oncology, 110(3), 390-397. doi:10.1016/j.radonc.2013.10.035
11. Types of Cancer Treatment. Retrieved from https://www.cancer.gov/about-cancer/treatment/types

VIII. Technical appendices

Appendix A: Code Repository Link

All code developed for this project, documentation, and images collected can be found in the GitHub repository at https://github.com/necroteddy/oncotools-master

The repository is private, so access must be granted by specific members of the team. The repository is private because it is a branch of the original oncotools github repository, which is also private.

Appendix B: Description of Files

We implement the framework structure in Python. Python is used because it allows for simple portability and OS compatibility. It also has many existing packages that can be used to augment and improve the framework. The framework is based off of oncotools, a prebuilt library by the McNutt lab for querying the oncospace database.

DataIntegrity.py

The main driver for the entire code base. The user runs this program with command line arguments and the program runs the correct corresponding tools.

engine.py

The Engine class initializes a database connection and the queries required to navigate the database. The engine class also create instances of the Manager and Stat classes and contains functions that allows user to pass commands to operate the classes.

Manager.py

The Manager class contains references to all existing anomaly detection modules, and enforces strict input requirements when a detection module is called. When new detection modules are to be added to the framework adding a reference to the location of the detection modules as well as the input required into the manager file will connect the detection module to the framework.

Statistics.py

The Statistics file contains classes that allow for managing the results of implemented integrity checks and the compiling of various statistics. It also contains functions for reading and writing data to file or printing result to screen.

Integrity_Module.py

The Integrity Module class is the basic integrity superclass. It enforces consistency in the structure of implemented integrity modules to be handled by the framework. It is written such that it will not significantly hinder the implementation of modules but makes sure that the implemented module will be able to receive input from the framework and return values that the framework can handle

Example integrity modules:

check_contiguity_extent.py

Module that checks that a mask is contiguous using projections along x, y, and z axes. Intakes a mask and returns validity of input mask with message/

check_contiguity_voxels.py

Module that checks that a mask is contiguous using region neighbor-crawling technique. Has option to use surface mask or volume mask. Volume mask requires significantly more run time and computational power.

check_dose_grid.py

Module that checks Dose Grid Data for basic standard errors