# A Treatment Planning System for the Small Animal Radiation Research Platform (SARRP) based on 3D Slicer

*Release 0.1*

Nathan B. Cho and Peter Kazanzides

June 21, 2012

Johns Hopkins University, Baltimore, MD

**Abstract**

This paper describes the software integration of a treatment planning system (TPS), based on the open-source 3D Slicer package, with the Small Animal Radiation Research Platform (SARRP). The TPS is designed to enable researchers to replicate their clinical techniques, allow for image fusion with other imaging modalities, and provide dose computation and graphical visualization of treatment plans consisting of multiple x-ray beams and conformable arcs. The dose computation is implemented on a GPU to achieve high performance; the dose volume for a typical treatment plan can be computed in less than a minute.

Latest version available at the Insight Journal [ http://hdl.handle.net/10380/3364]


## Contents

# 1 Introduction

SARRP is a system for mirco irradiation with on-board cone-beam computed tomography (CBCT) guidance[4]. The SARRP incorporates CT imaging with precise radiation delivery to enable researchers to identify an anatomical target and deliver radiation to that point, using one or more beams with diameters as low as 0.5 mm. The SARRP research platform has been designed to minimize the gap between the current human clinical systems and systems for small animal research. Cancer research with small animals allows researchers to study the details of biological processes and facilitate describing disease progression and identifying response to therapy.

It is not enough to precisely locate an anatomical target, however; it is also necessary to control the amount of radiation deposited on that target. For beams of a given energy, this can be achieved by specifying the *exposure time*, i.e., the amount of time that the beam is turned on. The SARRP software provides a Dose Calculator that uses machine-specific commissioning data to compute the exposure time given the source-to-surface distance (distance from x-ray source to surface of animal), the target depth, and the desired dose. While this may be adequate for determining the exposure time, it does not allow the researcher to visualize the dose distribution within the animal. This is a significant limitation because it is often necessary to limit the amount of radiation delivered to other parts of the animal (i.e., outside the target region). These considerations motivated the development of a treatment planning system (TPS) that enables the researcher to specify a treatment plan, consisting of one or more beams, compute the dose distribution, and then visualize the resulting dose volume. Thus, the TPS enables researchers to replicate clinical techniques and validate the dose distribution with multiple targets and plans before delivering the radiation exposure. Image fusion with other imaging modalities is also provided to facilitate planning.

The SARRP TPS is based on 3D Slicer (Version 3.6), which is an open source application for medical image visualization and analysis [2]. 3D Slicer is available on multiple operating systems, including Windows, Linux, and Mac OS X. Thus, although the SARRP software is currently restricted to the Windows operating system due to hardware constraints, it is possible to run the SARRP TPS on any platform. 3D Slicer is based on the Model-ViewController (MVC) design pattern and the Medical Reality Modeling Language (MRML). It provides the ability for researchers to develop custom modules to extend its functionality. We chose to write the custom module in Python because it was easy to implement, maintain, and distribute. We did not even need to recompile 3D Slicer.

Integration with 3D Slicer was further facilitated by its OpenIGTLink interface. OpenIGTLink is a network communication protocol that is designed especially for image-guided therapy (IGT) applications to transfer data, such as image volumes and transformations, between devices [3].

The dose computation is implemented in CUDA and is performed on a graphical processing unit (GPU) to achieve fast computation times [1]. It uses the superposition-convolution method to compute dose, rather than Monte Carlo.

# 2 System Overview

Although it is technically possible to integrate all functionality within 3D Slicer, one design constraint was that users should still be able to run the SARRP using the simple graphical user interface (GUI) provided by the SARRP software (see left side of Fig. 1). This is intended for users with simple experimental setups; for example, when it is sufficient to align the targeting lasers with skin markers instead of acquiring a CBCT image and creating a plan. Thus, it was decided to maintain two separate programs: (1) the SARRP

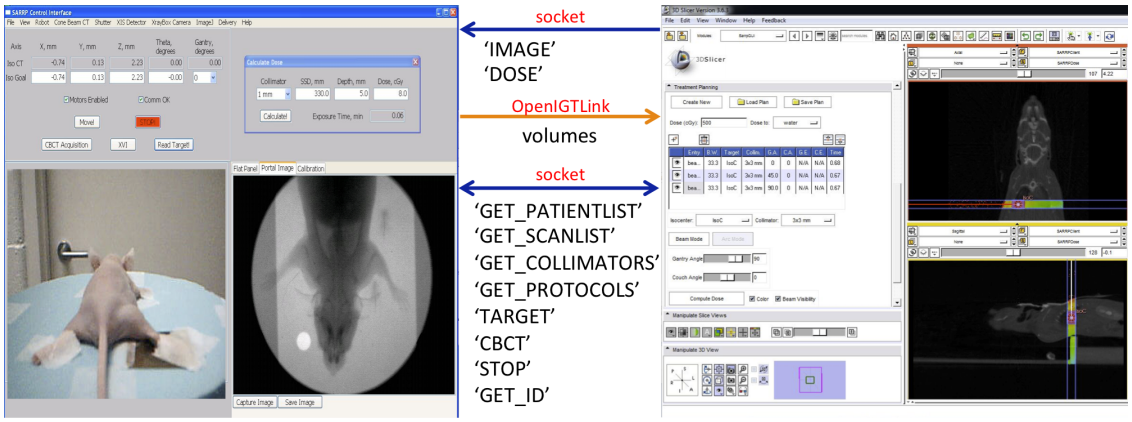


Figure 1: Communications between SARRP software (left) and TPS on 3D Slicer (right)

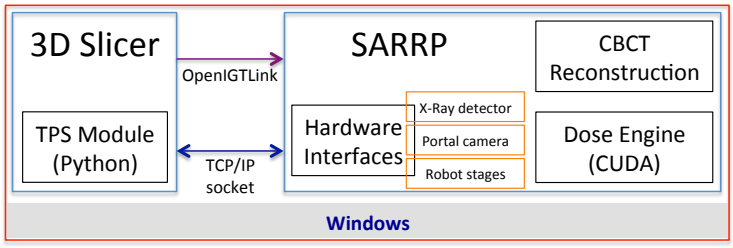


Figure 2: SARRP with Integrated TPS

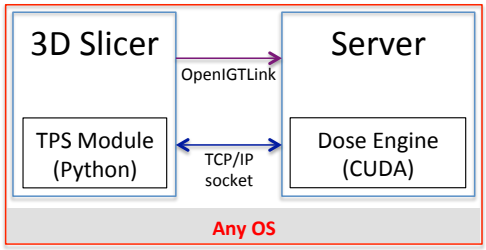


Figure 3: Standalone TPS

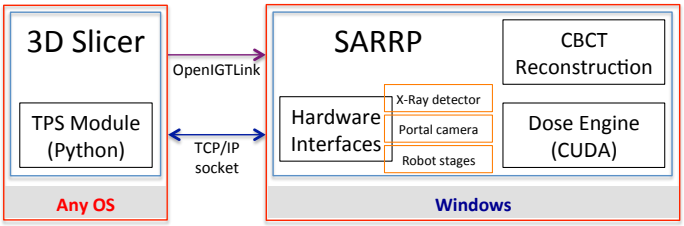


Figure 4: SARRP with Remote TPS

software, and (2) the TPS implemented on 3D Slicer. The design of the overall system is to make the SARRP software and TPS run as server and client, respectively. Figure 2 shows one possible configuration where both programs run on the SARRP computer (Windows operating system).

It was also desirable to create a standalone treatment planning system. The SARRP software, however, introduces many hardware dependencies because it contains interfaces to the robotic system (animal positioning system and gantry rotation), and to the many image sources (x-ray flat panel detector and several video sources). Presently, some of these hardware drivers restrict the software to run on the Windows operating system (Windows XP or Windows 7). For this reason, a *SARRP Server* program was created to provide the services required by the TPS module, without the hardware dependencies, as illustrated in Fig. 3. The SARRP and Server software are based on the open source *cisst* package and use wxWidgets for the GUI, and are therefore portable between operating systems. Because 3D Slicer is available on multiple operating systems, and the TPS module is written in Python, the Standalone TPS can run on any operating system. Finally, the client/server design also allows the TPS to run remotely from the SARRP, as shown in Figure 4.

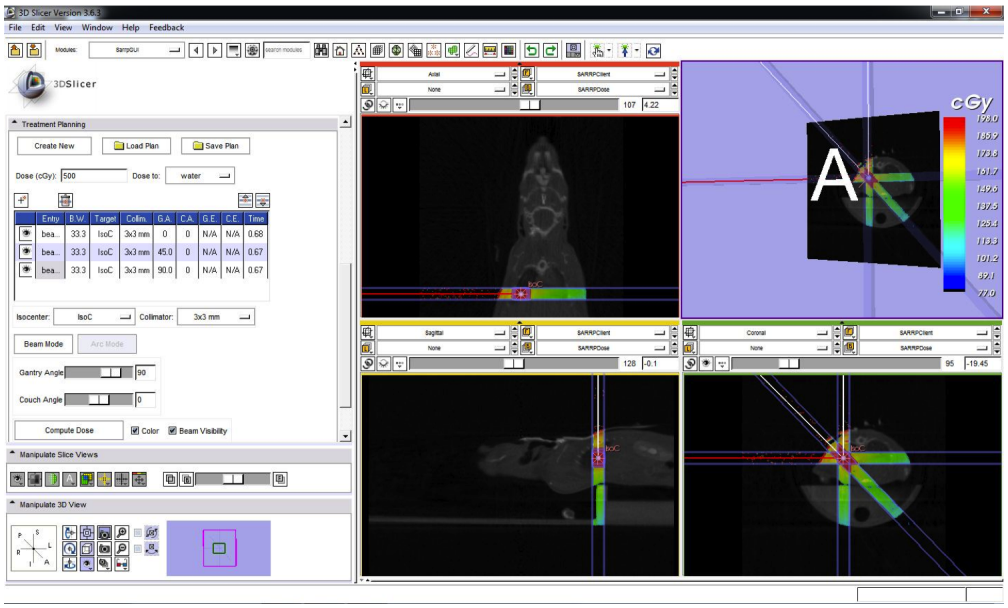


Figure 5: SARRP TPS with 1 isocenter and 3 beams

As shown in Fig. 1, there are two TCP/IP socket connections between the TPS and SARRP software. One is the OpenIGTLink socket, which is used for image transfers (i.e., responses to IMAGE and DOSE requests) and for sending the current beam position (transform) to 3D Slicer. To simplify the implementation, rather than developing custom extensions to OpenIGTLink, we chose to use another TCP/IP socket, with our own protocol, for commands not currently supported by OpenIGTLink. The general strategy is for the client (TPS module in 3D Slicer) to send a string request, possibly with parameters, to the server (SARRP or Server software) via the general TCP/IP socket (see Fig. 1). The server responds with the requested data (if any) and with an acknowledgment. For the IMAGE or DOSE commands, the desired volume is sent via OpenIGTLink. The SARRP software also sends the current beam position (as a transform) via OpenIGTLink whenever it differs from the previously transmitted beam position by a specified threshold.

## 3 TPS User Interface

The TPS module is shown in Fig. 5. It consists of four frames (in addition to *Help & Acknowledgement*): *Experiment Info*, *Target Selection*, *Treatment Planning*, and *Treatment Plan Execution* (Fig. 6). This GUI is designed to support the workflow of radiation oncologists.

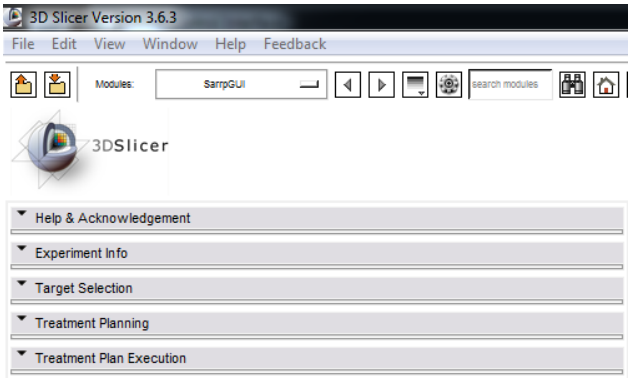


Figure 6: 4 Frames of SARRP TPS

In the *Experiment Info* frame (Fig. 7), two tabs are provided. In the first tab, CT volumes stored on the SARRP computer (whether local or remote) can be retrieved via OpenIGTLink. It is also possible to load volumes acquired from other imaging modalities for image fusion, using the volume loading functionality provided by 3D Slicer (for convenience, a "Load Volume" button is provided). The CBCT ac-

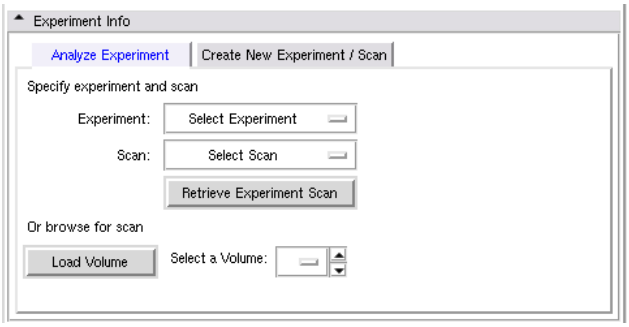


Figure 7: Frame1: Experiment Info

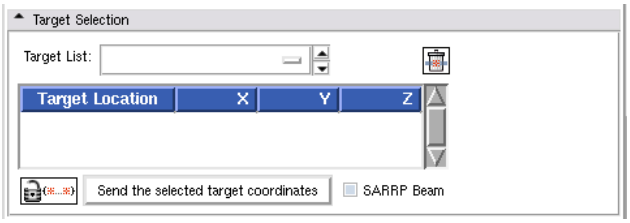


Figure 8: Frame2: Target Selection

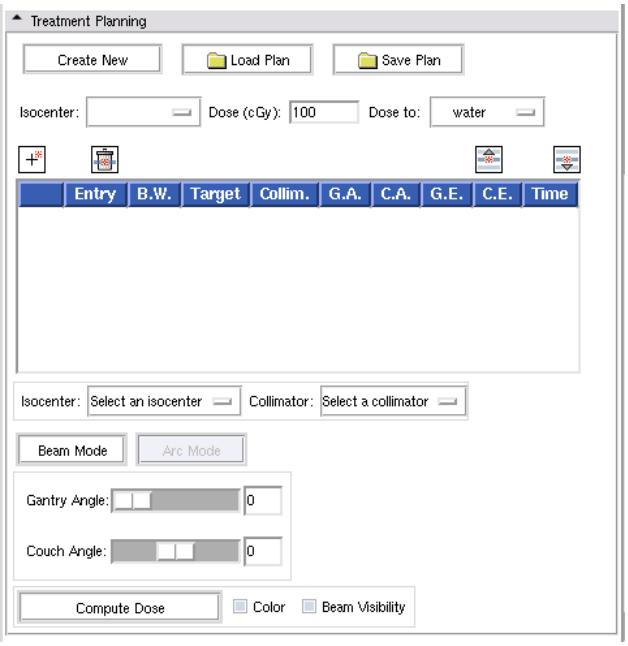


Figure 9: Frame3: Treatment Planning

quisition for a new specimen and/or a new scan and reconstruction with several options are provided in the *Create New Experiment/Scan* tab (this tab is not present for a Standalone TPS).

The coordinates of isocenters are displayed on the second frame, *Target Selection* (Fig. 8), where any of the selected isocenters can be sent to the SARRP software. This feature uses the fiducial list provided by 3D Slicer (i.e., the isocenters are actually Slicer fiducials). A checkbox is also provided to toggle on/off the display of the SARRP beam on the CBCT image. The SARRP beam relies on the SARRP software to send the beam position (as a transform matrix) via OpenIGTLink. Essentially, this is a navigation system, where the "tracked instrument" (locator) is actually the x-ray beam. The ball tip of the locator denotes the isocenter and the shaft of the locator corresponds to the beam axis. In the future, a custom locator should be defined because the x-ray beam should continue past the isocenter.

The *Treatment Planning* frame (Fig. 9) contains buttons to create, load, and save a plan. JavaScript Object Notation (JSON) format is used to facilitate the implementation of enconding/decoding a plan and allow users to read a saved plan without running a TPS. Users are allowed to determine a prescribed dose for each target and a type of dose computation. Researchers can create a number of entries (e.g. beams and/or arcs) for each plan, specify properties such as the isocenter, collimator type, couch and gantry angles, and switch a mode between beam (default) and arc. Once a valid plan is created, it can be sent to the dose engine in the SARRP software for execution; in this case, we serialize the plan using the `dumps` method of the Python `json` package and transmit it via the general TCP/IP socket. Checkboxes are provided for the color visualization of dose distribution and the beam's nominal outline (i.e., not including the minor effects of beam divergence).


Figure 10: Frame4: Treatment Plan Execution

Currently, users can stop the SARRP system and generate a report in the bottom frame, *Treatment Plan Execution* (Fig. 10). In the future, users will be able to directly execute plans from here.

# 4 TPS Implementation

The SARRP software maintains a directory structure of CT scans, which is organized by *experiment name* (analogous to *patient name* in a clinical system) and *scan name*. To load a CT volume in 3D Slicer, a user must specify the experiment and scan information; the list of experiments is obtained from the SARRP software by sending the `GET_PATIENTLIST` command via the socket, and the list of scans for a particular experiment is obtained via the `GET_SCANLIST` command. This implementation is somewhat inefficient when Slicer and SARRP are running on the same computer, since Slicer could directly scan the hard drive for the list of patients and scans, but allows us to easily support a networked configuration, as shown in Fig. 4. Once the experiment and scan are selected, the TPS sends the `IMAGE` request, with the experiment and scan names as parameters, via the socket and SARRP sends the corresponding volume to the TPS through OpenIGTLink. When the volume is received, the TPS locates it on the background to allow users to easily adjust the window/level.

Once a planning volume is loaded, the next step is to set isocenters (targets). After one or more isocenters are created, the coordinates of a selected isocenter can be sent to the SARRP software through the socket. This can be used to move the SARRP so that it positions the selected point at the machine isocenter. The location of the X-ray beam can be represented and updated in real time by the locator of the OpenIGTLink IF module.

In the *Treatment Planning* frame, multiple plans can be created using the isocenters defined in the previous frame. First, the amount of prescribed dose (in centigray, cGy) for each isocenter must be entered, followed by selecting a type of dose computation (i.e., either "dose to water" or "dose to medium"). As shown in Fig. 9, we provide icons for adding, deleting, and moving entries up or down in the list. The ability to move the entry is useful to sort based on collimator, thereby minimizing the amount of time needed to switch collimators.

When the user clicks on the icon to add a new entry, the TPS software automatically sets some default values for user convenience. For example, all new entries are set as beams (rather than arcs). Also, if only one isocenter was defined, that isocenter is automatically selected for the new entry. If, however, the user identified more than one possible isocenter, the TPS leaves that field empty so that the user can later select it from the "Isocenter" drop-down menu. Once the isocenter is known, the TPS software creates a *measurements node*, which is represented by red-colored lines on the viewers. For beams, the node is an instance of `vtkMRMLMeasurementsRulerNode`, whereas for arcs it is an instance of `vtkMRMLMeasurementsAngleNode`. To prevent confusion, only the node corresponding to the current (selected) plan entry is shown in red; the others are depicted as white-colored lines (see Fig. 5). The TPS also provides a toggle button in the first column of the plan table to allow users to turn on/off the visualization of its node.

For the initialization, we had to make one of the two (in the case of beam) or three (in the case of arc) points fixed to, and placed on, its selected isocenter. We provide several options to change the beam: moving the slider bar, typing a new value in the text box, or grabbing a point of the node on the viewers and moving it. The visualization of the beam's nominal outline, depending on the shape of collimator, is achieved using MRML nodes such as `vtkMRMLModelDisplayNode` and `vtkMRMLModelNode` and VTK objects such as `vtkTransformPolyDataFilter`, `vtkCylinderSource` (for round collimators), and `vtkCubeSource` (for rectangular collimators).

Figure 11 shows an example of a JSON file consisting of prescribed doses and target coordinates for each isocenter, the type of dose computation, a file version and several entries (beams or arcs) with their associated properties. Each plan entry has a name, type (i.e., either beam or arc), isocenter name, dose weight in percentage, and collimator name. Beams have just a single gantry and couch angle, whereas arcs have starting and ending angles for the gantry and couch. Once all the properties of plans are properly set, the dose can

```
{
    "FileVersion": 2,
    "NumPlans": 2,
    "DoseType": "water",
    "IsoC": {
        "dose": 100,
        "isocenter_coordinates": [
            -3.9980087280273437,
            -8.8361988067626953,
            3.1830594539642334
        ]
    },
```
(1)

```
"offset1": {
    "dose": 100,
    "isocenter_coordinates":
        -4.0,
        -22.217441558837891,
        1.8007192611694336
    ]
},
```
(2)

```
"0": {
    "collimator": "3x3 mm",
    "couch": 0,
    "gantry": 15.0,
    "isocenter": "IsoC",
    "name": "beam0",
    "time": 0.43407499999999999,
    "type": "beam",
    "weight": 100.0
},
```
(3)

```
    "1": {
        "collimator": "5x5 mm",
        "couch": 10,
        "gantry": 90.0,
        "isocenter": "offset1",
        "name": "beam1",
        "time": 0.45163900000000001,
        "type": "beam",
        "weight": 100.0
    }
}
```
(4)


Figure 11: Example of JSON

be computed by clicking the "Compute Dose" button, which sends the `DOSE` command, with the dose plan (JSON string) as a parameter, to the SARRP software. The SARRP software parses the JSON string and invokes the dose engine on the GPU. After the computation is completed, the SARRP software returns the dose volume to 3D Slicer throught the OpenIGTLink connection. It also returns an updated dose plan (JSON string) that includes the computed exposure time for each beam and the measured source-to-surface distance (SSD).

Once the output volume is received, one of the event handlers from `vtkMRMLVolumeNode`, `ImageDataModifiedEvent`, detects the incoming image data and the TPS moves the existing planning volume from the background to the foregroud layer and locates the dose volume on the background layer in the three 2D viewers with the fade scale set to 50 percent. Placing it on the background layer enables users to adjust the color visualization by using vertical and/or horizontal mouse motions (e.g., to change the window/level). Those adjustments are caught by the `DisplayModifiedEvent` handler of the `vtkMRMLDisplayableNode` class, which reads the new window/level values and updates the range of the scalar bar in the 3D viewer so that the correct dose value is shown for each color.

## 5 TPS Validation

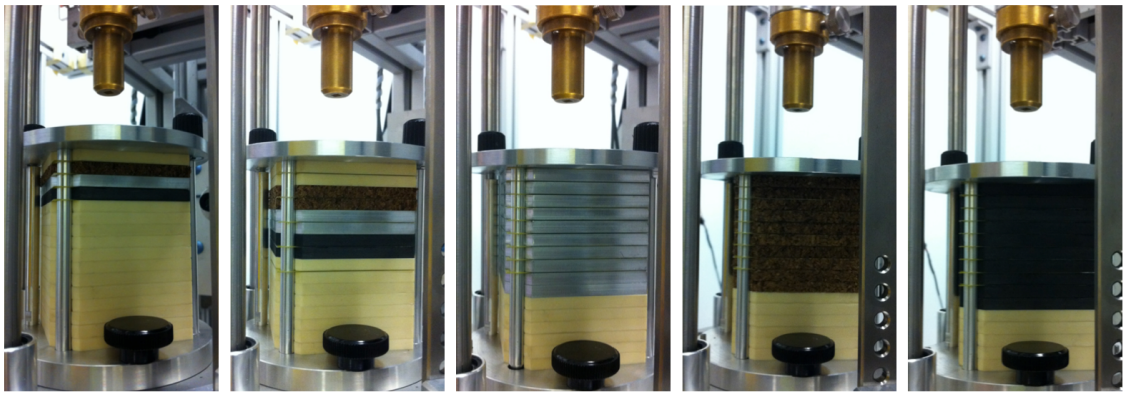


Figure 12: Five Phantom Setups

We validated the TPS dose computation with 5 different phantom setups. For each setup, a stack of up to 4 different materials (e.g., cork, water, graphite, and aluminum, with densities of 0.25, 1.0, 1.7, and 2.69,

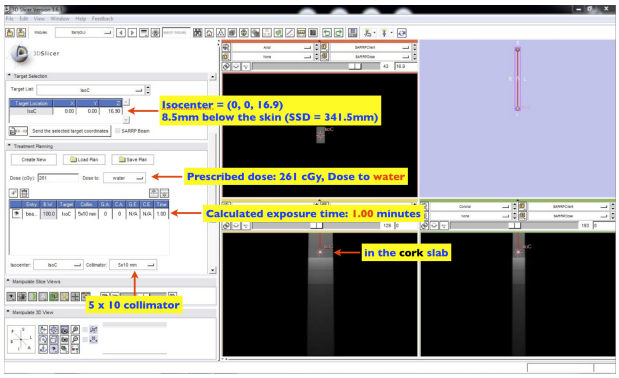


Figure 13: Validation of the TPS with SET1

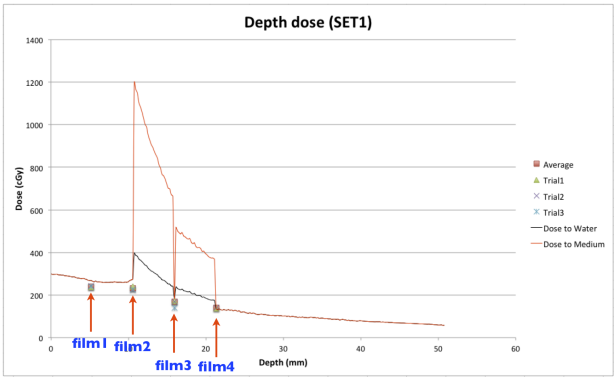


Figure 14: Validation Result of SET1

respectively) were used. These materials encompass the range of densities of mouse organs (e.g., lung, fat, and bone, which have densities of 0.25, 0.95, and 1.92, respectively). As shown in Fig 12, for SET1, 4 radiation sensitive GAFCHROMIC EBT films whose density is close to water were used between each heterogeneous material of 5 mm layers. Slabs of water, cork, aluminum, and cork were stacked from top to bottom. In SET2, each material with the same order of SET1 was double-stacked and 5 films were placed in between. The other sets consist of a stack of aluminum (SET 3), cork (SET 4), and graphite (SET 5), with a film placed between each layer. For all the phantoms, the remaining volume was filled with water equivalent plastic slabs. The films in the phantoms were exposed to radiation with different settings of source to surface distance (SSD), exposure time (i.e., either 1.5 or 3 minutes), and collimator type (i.e., either 5x9 or 5x5 mm).

For the planning volumes, we created synthetic CBCT images in `nrrd` format, based on the known density values of each material. We loaded each image into the TPS and specified the isocenter, prescribed dose, and collimator to match the experimental values, as shown in Fig. 13 for SET1. We then computed the dose and compared it to the dose measured from the films. Figure 14 shows a comparison of the depth dose (i.e., dose measured along the central beam axis) produced by the dose engine to the actual film measurements for SET1. The results for the other phantoms were similar.

## 6 Conclusions and Future Work

We have successfully implemented and validated the TPS for the SARRP system based on 3D Slicer, using the OpenIGTLink protocol to transfer image and transform data between the two programs. We chose to create a separate socket for exchanging other data objects, rather than extending the OpenIGTLink protocol, for simplicity of implementation. Overall, we had a positive experience using Python to create a custom module for 3D Slicer, even though we used Slicer 3.6, which does not natively use Python (rather, it has a Python "wrapper" over the native TCL code). We expect that it will be even easier to use Python with Slicer 4, though it will be necessary for us to replace the KWWidgets GUI elements with their Qt counterparts.

Our future work includes the integration of a segmentation module, so that the user can segment the different types of tissue, including lung, soft tissue, fat, and bone, and assign known density values, rather than relying on the CBCT pixel values to determine density.

## Acknowledgments

The SARRP was developed with support from NIH R01 CA108449, under the leadership of John Wong. Robert Jacques and Todd McNutt developed the GPU dose engine. Ian McMahon and Hongliang Ren assisted with the initial integration of 3D Slicer. Esteban Velarde and Michael Armour performed the dose validation experiments. Development of the TPS was supported by funding from Xstrahl Limited.

## References

[1] R. Jacques, R. Taylor, J. Wong, and T. McNutt. Towards real-time radiation therapy: GPU accelerated superposition/convolution. *Computer Methods and Programs in Biomedicine*, 98(3):285 – 292, 2010. 1

[2] S. Pieper, M. Halle, and R. Kikinis. 3d slicer. pages 632–635, 04 2004. 1

[3] J. Tokuda, G. S. Fischer, X. Papademetris, Z. Yaniv, L. Ibanez, P. Cheng, H. Liu, J. Blevins, J. Arata, A. J. Golby, T. Kapur, S. Pieper, E. C. Burdette, G. Fichtinger, C. M. Tempany, and N. Hata. Openigtlink: an open network protocol for image-guided therapy environment. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 5(4):423–434, 2009. 1

[4] J. Wong, E. Armour, P. Kazanzides, I. Iordachita, E. Tryggestad, H. Deng, M. Matinfar, C. Kennedy, Z. Liu, T. Chan, O. Gray, F. Verhaegen, T. McNutt, E. Ford, and T. L. DeWeese. High-resolution, small animal radiation research platform with x-ray tomographic guidance capabilities. *International Journal of Radiation Oncology Biology Physics*, 71(5):1591 – 1599, 2008. 1