Gamma Knife Radiosurgery
Project / Assignment for CISC-330

Contents

Problem Statement

We will treat a small metastatic brain tumor with Gamma Knife radiosurgery (**Figure 1**). Our Gamma Knife helmet provides Cobalt pencil beams over a nearly full hemisphere, the beams are separated by a uniform angle in longitude and latitude (**Figure 2**). The centerlines of the pencil beams intersect in the center of the hemispherical helmet. This point is called the isocenter point or shortly isocenter. The shape of each pencil beam is considered as a cylinder with a given radius.

Fig. 1: Gamma Knife Radiosurgery

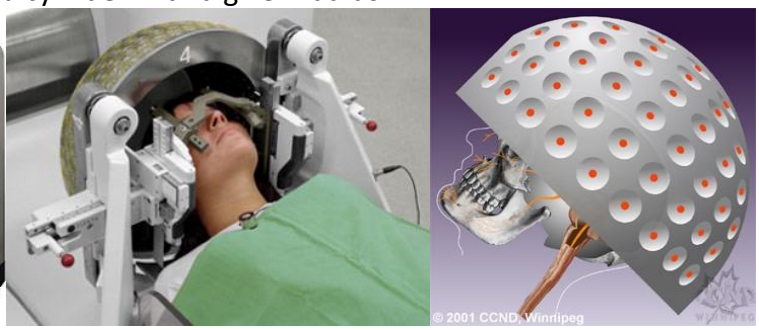


Fig. 2: Full hemispherical Gamma Knife helmet

The tumor was contoured in cross sectional images. The prescribed target volume (PTV) was contoured by leaving some margin around the tumor, the yellow contour in **Figure 3**. The organ at risk (OAR) was also contoured, red contour in **Figure 3**. The PTV and OAR were reconstructed in 3D, and approximated as spheres. We will irradiate the PTV with a helmet that provides pencil beams to irradiate the PTV. With the use of the head frame bolted into the skull, we will position the patient in the helmet so that the isocenter coincides with the center of the PTV. Since the PTV is spherical and expectedly not too large, we will treat this patient with a single isocenter.

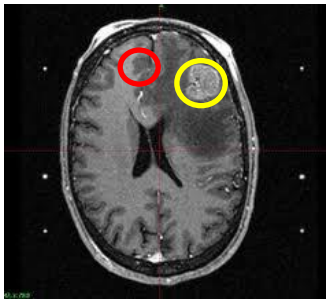


Fig. 3: The PTV (yellow) and OAR (red) contoured in a cross sectional image.

The radiation dose is computed inside a dose box made around the OAR and PTV. The dose box encompasses these structures as tightly as possible without margin. The dose box is distributed into a uniform grid of voxels of a given voxel size. The radiation dose is computed in each voxel. The dose is also computed on the surface of the PTV and OAR. The distribution of the dose is analyzed for covering the PTV and sparing the OAR, using dose volume histogram (DVH) and dose surface histogram (DSH).

The patient's head has been contoured in the cross sectional images and reconstructed approximately as an ellipsoid. The ellipsoid has no rotation, i.e. the principal axes of the ellipsoid are aligned with the axes of the image coordinate system, like in **Figure 4**, where a, b, and c mark the half length of the principal axes of the ellipsoid.

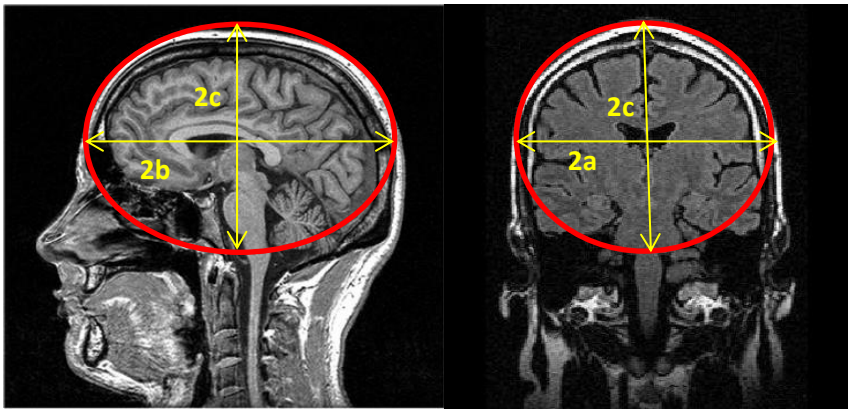
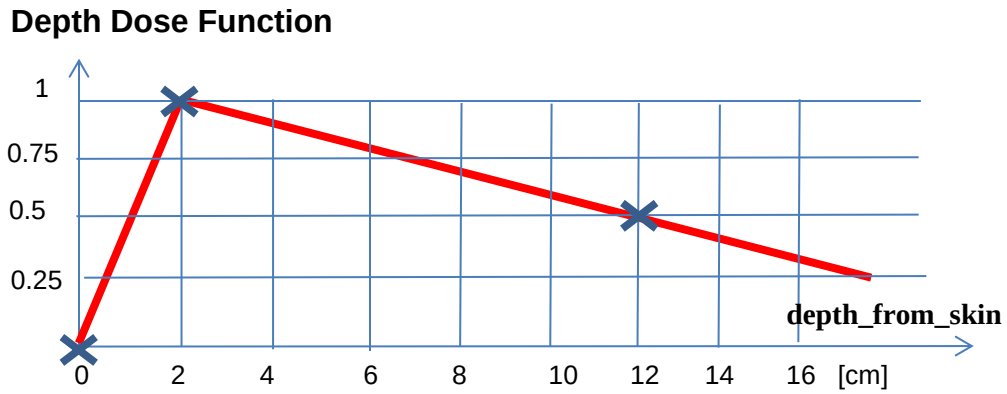


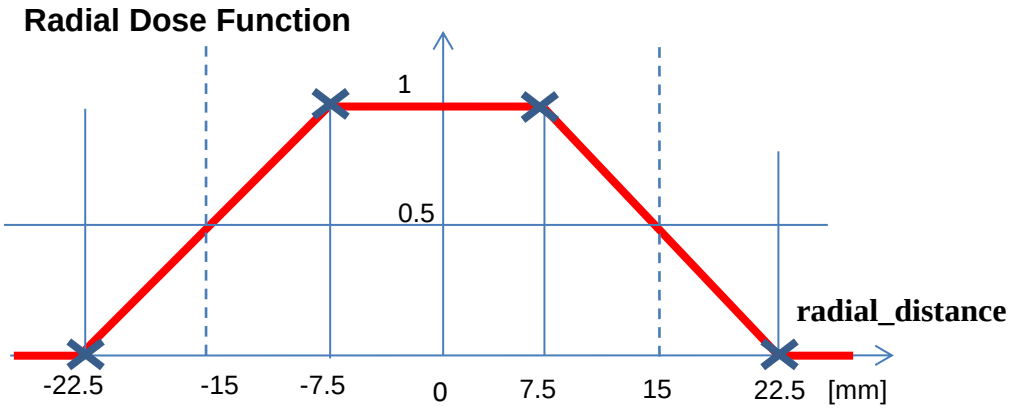
Fig. 4: Ellipsoid approximating the patient's head

The depth and the radial dose functions for the given beam_radius were measured and approximated as shown below.

Depth Dose Function: Each beam, the build-up is linear, it delivers a 1.0 unit dose at 2cm depth at the beam's centerline, then the dose attenuates linearly according to the figure below:



Radial Dose Function: Each beam is radially "leaking" some dose outside the collimator, according to the figure below:



Variables and Numerical Values for Testing

Helmet: [beam_separation_angle = 10 degrees, beam_radius = 15 mm]
PTV: [ptv_radius = 15 mm, ptv_center = (30, 0, 15) mm]
OAR: [oar_radius = 15 mm, oar_center = (0, 30, 45) mm]
Head: [head_a = 80 mm, head_b =100 mm, head_c=80 mm, head_center = (0, 0, 0) mm

Functions and Tests

Do not hardcode variables and parameters in the functions. Communicate all parameters through proper input and/or apply global variables, as needed.

Implement a debugging flags in the code that you can turn on for module testing, when you also check the validity and range of input and check for mathematical singularities, etc. In the final dose computations, you should turn off the debug mode, in order to achieve acceptable running time.

You will need to implement and test the following functions:

1. Compute Dose Box 5 pts

Description: Write a function to compute the Dose Box around the PTV and OAR. Make the box as tight as possible around these anatomical structures, without margin. Define the box with the lower left and upper right corner points. The box should have no rotation relative to the coordinate axes.

Name: Compute_Dose_Box
Input: PTV, OAR
Output: dose_box
Test: Print the dose_box; examine the result by ground truth hand calculation for correctness. (**2 pts)**

2. Draw 3D Scene 5 pts

Description: Draw the 3D scene with Head, PTV, OAR, isocenter, dose_box, and the coordinate axes. Choose the appropriate surface representations, colors, etc. for the most meaningful visual presentation.

Name: Draw_3D_Scene
Input: Head, PTV, OAR, isocenter_point, dose_box
Test: Run the function, plot some views to show that the dose box properly covers the PTV and OAR. (**2 pts)**

3. Compute Linear Function 5 pts

Description: Write a function to compute the value of ***y***=m***x*** +b linear function, given by P1=(x1, y1) and P2=(x2,y2) points of the line, for a value of ***x***.

Name: Compute_Linear_Function
Input: P1, P2, x
Output: y
Test: Run the function for P1(1,1), P2(5,5) x = {0, 1, … 6 }; Print & examine the result. (**2 pts)**

4. Compute Radial Distance 5 pts

Description: Write a function to compute the radial distance between the center line of a beam and an arbitrary point in space (**3 pts**)

Name: Compute_Radial_Distance
Input: point, line [point_on_line, line_direction_vector]
Output: distance
Test: Run the function with two examples of known ground truth **(2 pts)**

5. Compute Depth Dose 5 pts

Description: Write a function to compute depth dose value according to the Depth Dose Function graph. Write the results into a global look up table (LUT) with appropriate resolution, such as 1.0mm increments of depth. You will use this table for quick access of depth dose during dose computation. Use your Compute_Linear_Function that you developed earlier above. (**3 pts**).

Name: Compute_Depth_Dose
Input: max_depth_from_skin, d0, depth_from_skin_resolution
Output: depth_dose_ LUT
Test: Run the function, print the LUT; show that values in the marked points of the depth dose curve are correct. (**2 pts).**

6. Compute Radial Dose 5 pts

Description: Write a function to compute radial dose value according to the Radial Dose Function graph. Write the results into a global look up table (LUT) with appropriate resolution, such as 1mm increments of radial distance. You will use this table for quick access of depth dose during dose computation. Use your Compute_Linear_Function that you developed earlier above. (**3 pts)**.

Name: Compute_Radial_Dose
Input: max_radial_distance, beam_radius, radial_distance_resolution
Output: radial_dose_ LUT
Test: Run the function, print the LUT; show that values in the marked points of the radial dose curve are correct. (**2 pts).**

7. Compute Beam Direction Vector 5 pts

Description: Write a function to compute the unit direction vector for the pencil beam's centerline from beam longitude and beam latitude, in Cartesian coordinates. (**3 pts).**

Name: Compute_Beam_ Direction_Vector
Input: beam_longitude, beam_latitude
Output: beam_direction_vector
Test: Run the function with beam_longitude= {0, 90, 180, 270} degrees and beam_latitude={0, 90} degrees. Print the results in table format; check for correctness by hand calculation (**2 pts).**

8. Compute Skin Entry Point 5 pts

Description: Write a function to compute the skin entry point of a pencil beam. Write up the equations of the beam and the head, and solve for the intersections. Derive the solution/formula analytically (**3pts**). Implement the formula in MATLAB (**2 pts**).

Name: Compute_Skin_Entry_Point
Input: Head, beam_longitude, beam_latitude, isocenter_point
Output: skin_entry_point
Test: none here

9. Compute Skin Entry Point Table 5 pts

Description: Write a function to compute the skin entry point for each beam and write the result into a global table. (**2 pts**)

Name: Compute_Skin_Entry_Point_Table
Input: Head, isocenter_point
Output: skin_entry_point_table
Test: Run the function; draw a 3D scene with the head and the centerline of each pencil beam. The plot should show where the beam centerlines intersect the head. Add a marker to each computed skin entry point. Check visually if the markers drawn coincide with the beams exiting the head (**3 pts**).

10. Compute Beam Safety 5 pts

Description: Write a function to compute if a pencil beam is safe. The beam is unsafe when the beam intersects with the OAR. In comment, explain how you compute beam safety. Unsafe beam holes in the Gamma Knife helmet will be plugged by the technicians; these beams will not contribute any dose.

Name: Compute_Beam_Safety
Input: beam_radius, beam_longitude, beam_latitude, isocenter_point, OAR
Output: 0 = unsafe (intersection) 1 = otherwise
Test: none here

11. Compute Beam Safety Table 5 pts

Description: Write a function to compute beam safety for each beam and write the result into a global table.

Name: Compute_All_Beam_Safety
Input: beam_radius, isocenter_point, OAR
Output: beam_safety_table
Test: Print the table by beam longitude & latitude**.**

12. Compute Point Dose from Beam 10 pts

Description: Write a function to compute the dose at a point of interest from a beam. (Tips: compute the beam direction vector, compute the skin entry point, compute the radial dose value, compute the depth dose value, and finally compute the dose value in the given point as the product of depth dose value and radial dose value.)

Name: Compute_Point_Dose_from_Beam
Input: Head, beam_radius, beam_longitude, beam_latitude, isocenter_point, point_of_interest
Output: point_dose_value
Test: Full debugging of this function is not easy, but we can get by with a quick and dirty "sanity check". Turn on debug the debug mode. Run the function with beam_longitude {0, 90} and beam_latitude = {0, 90} values and with a fake isocenter_point that coincides with the center of the head. Note that for these beams the skin depth is the same as the appropriate half-diameter of the head. Then, for each of these beams, compute the point_dose_value in the (fake) isocenter point – here the radial_dose_value is 1.0 and therefore the point_dose_value should be equal to the depth_dose_value. (**3 pts)**

13. Compute Point Dose 5 pts

Description: Write a function to compute the dose in a given point of interest from all beams.

Name: Compute_Point_Dose_from_All_Beams
Input: Head, Helmet, OAR, isocenter_point, point_of_interest
Output: point_dose_value
Test: none here; will be called later.

14. Compute Dose Volume Histogram 10 pts

Description: Write a function to compute the dose inside the dose_box with a given dose voxel size and save the dose values in a 3D dose matrix**.**

Compute dose volume histograms in terms absolute dose units on the horizontal axis of the graph, with setting D_o =1.0 dose unit in your code, determine the range on the horizontal axis sensibly. (Remember, D_o is the amount of dose that one beam delivers along the beam's center line at d_o depth.)

Implement two helper functions named Is_Point_Inside_PTV and Is_Point_Inside_OAR, to mark each dose voxel for subsequent DVH computation**.**

Without some optimization, the DVH computations can run for a long time. You have already implemented some: you pre-computed the depth dose, radial dose, skin entry points, and beam safety into global tables before iterating through the dose voxels. A further speed-up can be obtained from computing dose only at voxels inside the PTV or OAR; call your helper functions and mark up each dose voxel for subsequent DVH computation. With these tricks, the DVH with dose voxel size of 1mm should compute in a few minutes on most computers. Report the running time of your program. For quick testing you can use larger voxel size. Make sure to turn off debug mode.

Compute and plot the dose volume histograms for the PTV and OAR **(7 pts).**

Examine the DVH and explain what the two curves suggest about the dose coverage of the PTV and OAR (**3 pts).**

Name: Compute_ Dose_Volume_Histogram

Input: Head, Helmet, PTV, OAR, isocenter_point
Test: None beyond the plots and prints required in the function's body.

15. Compute Surface Dose 10 pts

Description: Write a function to compute the dose on the surfaces of the PTV and OAR. (Tipp: subdivide the PTV and OAR surfaces to quadrilateral patches in spherical coordinates and compute the dose in each vertex (**5pts).**

Plot the PTV and OAR surface dose as a colored surface, including a color bar (**3 pts).**

Compute the hottest and coldest dose and locations on the PTV and OAR surfaces. Print the locations and the associated dose values. Mark these locations in the surface plots (**2 pts).**

Name: Compute_Surface_ Dose
Input: Head, Helmet, isocenter_point, PTV, OAR
Test: None beyond the plots and prints required in the function's body

16. Compute Dose Surface Histogram 10 pts

Description: Write a function to compute the dose surface histogram (DSH) for the PTV.

Compute the dose surface histograms in terms absolute dose units on the horizontal axis of the graph, with setting D_o =1.0 dose unit in your code, determine the range on the horizontal axis sensibly. (Remember, D_o is the amount of dose that one beam delivers along the beam's center line at d_o depth.)

Plot the histogram curve (title, with axis labels, etc.) and briefly analyze/explain the curve. Tips: Subdivide the PTV surface to small patches; compute the dose in each vertex of each patch; compute the average dose belonging to each patch; compute the surface of each patch; compute the total surface area as a sum of all patches; compute the coldest patch; compute the hottest patch; compute the histogram curve between these values. (**10 pts**)

Name: Compute_ Dose_Surface_Histogram
Input: Head, Helmet, isocenter_point, PTV
Test: None beyond the plots and prints required in the function's body

Testing and Reporting

Write a routine to test each function you developed. If the name of the function you want to test is Whatever_Function, then the test routine should be named Test_Whatever_Function and it should be placed the Whatever_Function.m file. Write a main.m function that executes the tests. Save all output and plots in a file and provide the required analyses and explanations.

General Rules

- Read the online syllabus carefully for general instructions on the submission of assignments.
- Always explain how you solve a problem. Use drawings, math formulas, text, block diagram, pseudo code - anything that you find them appropriate to convey your ideas. I must know that you understand what you are doing and I must be able to follow your reasoning. Depending on the quality and depth of your reasoning and discussion or results you may pick (or lose) lots of points.
- Write proper header and richly comment your code. There is no such thing as too much comment. Good style and neatness will earn you valuable points. The lack of these will cause reduction.

- Always test the validity and deformity of the input data; lack of such testing will lead to deduction.
- Test each module fully and construct several test cases with known ground-truth answer.
- Use decimal digits sensibly and consider what is precision is practical for the given problem. Generally, resolution finer than 1 mm for radiosurgery is not practically achievable. Use integer or decimal point format in your outputs. No exponential number format!
- Write a testing m file(s) for each module or problem.
- Capture the output, to show that your program does what it is supposed to do. Make plots whenever it is requested or makes sense. Add explanation text as you see it useful.
- Use MATLAB routines for recurring tasks.
- Submit the m files and the captured output file, as well as any drawing, or supplemental information you feel relevant.
- **Have fun!**