Gamma Knife Treatment Planning

MATTHEW S. LEA MICHAEL T. MARA QIAO ZHANG

January 28, 2010

Abstract

The problem of where to focus radiation during certain types of radiosurgery can be represented as a sphere-packing problem. We implement a method for simulating cancerous tumors on which solutions can be tested and visualized. We then implement an algorithm to approximate the optimal sphere-packing of an irregular shape and consider its performance using our visualization.

1 Introduction

In Gamma Knife Radiosurgery, high intensity gamma radiation is used to kill brain cancer cells while sparing the surrounding healthy cells. Gamma radiation from 201 cobalt-60 sources is delivered to the target cells through collimators on a helmet. The beams intersect at the isocenter, resulting in a spherical dose distribution. We term it a shot. An optimal gamma knife treatment needs to meet the following requirements:

- Shots cannot protrude outside the target.
- Shots cannot overlap.
- Shots cover at least 90% of the target volume.
- Minimal shots used.

Since shots can be represented as spheres of diameters of 4,8,14,18 mm, the task of finding the optimal treatment plan becomes a sphere packing problem. We aim to generate a representation of a tumor cell and fill the cell with spheres of the four diameters available. The constraint is thus to use minimal number of non-overlapping spheres to fill at least 90% of the tumor cell.

2 Tumor Cell Generation

Before we can test any treatment plan we come up with, we first need tumors on which we can test the plans. After some research into tumor growth modeling, we decide that a simple 3D cellular

automaton will do the job nicely. The goal is not to accurately model growth of brain tumors, but rather to come up with shapes that are good approximations of real tumor shapes on which the treatment will be used.

2.1 Single Pressure Bound Model

We start with a 100x100x100 array of zeroes, containing 1 million voxels that each represents $1\mathrm{mm}^3$. At the center of the array, we change one voxel to a 1. Then, for every timestep, we iterate through the entire array. Whenever there is a 1, we check the six empty voxels adjacent to it. If the voxel contains a 0, a formula is used to compute the probability that the tumor cell (represented by a 1) will divide there. In the basic model, this formula is simply:

$$\mathrm{probability} = 1 - (\mathrm{r}/\mathrm{R})) * \mathrm{max\_prob} \tag{1}$$

where R is the maximum radius of the tumor, r is the distance from the first cell to the one that is dividing and max_prob is the maximum probability a division is allowed to have, and was thus a number from 0 to 1. In the simple model, max_prob was used to mitigate the oddities caused by using Euclidean space for organic cell growth. A random number between 0 and 1 was then generated, and the voxel in question was changed to a 1 if the random number was less than the probability. This simple model was basically a model of uniform growth under a single pressure bound (the maximum radius). It led to roughly spherical tumors (spheres are often used to represent "ideal" tumors).

2.2 Multiple Pressure Points Model

In the more advanced model (the one we actually used), we add "pressure points" to the model. Cell growth is inhibited when the cells are closer to the pressure points. The probability that the tumor cell will divide at a point is a factor of the base probability (given by the formula (1)) :

$$\mathrm{probability} = (\mathrm{s}/\mathrm{max\_dist})^{\mathrm{p}} * (1 - (\mathrm{r}/\mathrm{R})) * \mathrm{max\_prob}) \tag{2}$$

where s is the distance from the voxel we are determining the probability of division for to the pressure point, max_dist is the maximum distance it is possible for a voxel to be away from the pressure point, and p is a parameter of the model. The higher the p, the more pressure points hinder growth throughout the model, though with more intensity near said pressure point. We most often use 0.5, 0.8 and 1 for p. These pressure points effectively model the different structures that inhibit tumor growth in the brain. It allows us flexibility to create very odd shapes, and, by randomizing the pressure points, to create more realistic tumors without tediously specifying every pressure point. This model creates a large range of approximations of organic shapes.

3 Sphere-Packing

3.1 Definitions and Overview

We define a *shape* as a connected set of points in a three-dimensional grid, a shape's *boundary* as the set of all points not in the shape but adjacent to at least one point in the shape, and a *sphere-packing* of a shape S with sizes R and coverage c as a set of spheres of radius $r \in R$ such that: the spheres do not share any points, the spheres cover at least c percent of S, the centers of the spheres are points in S, the spheres contain no points in the grid outside of S. An *optimal sphere-packing* is a sphere-packing using a minimal number of spheres for the given parameters. The problem of finding an optimal sphere-packing of an irregular shape is NP-complete [Wang], so we use an approximation. Our method relies on the notion of a medial axis, also called a skeleton. The *medial axis* of an shape is the set of all points have more than one closest point in the boundary of that shape. Wang conjectured that the spheres in at least one optimal packing of a shape have their centers on that shape's skeleton. Thus we can approximate an optimal sphere-packing of S by using an algorithm that for each of the i points in the skeleton of S, approximates the optimal sphere-packing of S_i, the shape formed by removing the largest sphere we can place a the ith skeleton point of S, from S by recursively applying itself to S_i, returning the best of these i packings.

3.2 Skeletonization

To use the algorithm above, we must first have a way to find the skeleton of a shape. We approximate the skeleton using the wildfire method [Bourland and Wu 2000]. We begin with the set of points in the shape, each of which we give a value of infinity, and the boundary of the shape, each point of which we give a value of zero. Starting at $k = 0$ we set all points adjacent to a point in the boundary to a value equal to the minimum of their current value and $k + 1$, then increment k. In the kth step for $k \geq 1$, we set all points adjacent to a point which changed value in the previous step to the minimum of their current value and $k + 1$. Once all points are assigned a new value, we iterate over all points in the shape and put them in the skeleton if they have a value at least as great as all adjacent points.

Each point in our implementation of the algorithm also stores the coordinates of the point in the boundary closest to it. We estimate this value by having each point change its closest point coordinates to the closest point coordinates of the point next to it from which it receives a lower value during the skeletonization algorithm. This aspect of our implementation allows us to calculate whether we can place a given sphere of a given size at a point merely by checking whether it overlaps the closest point rather than checking each point in the boundary of the shape, giving an asymptotic speedup which decreases runtime ten to a hundredfold for the shapes we tested. The cost of this optimization is that our estimated closest point may not always be the closest point, allowing spheres to overlap and protrude from the shape by a small amount.

3.3 Algorithm

The algorithm to approximate the optimal sphere-packing of S given above must check every combination of skeleton points in S and even with dynamic programming is far too slow to run in practice. We follow [Bourland and Wu 1996] by instead checking only endpoints and crosspoints of the skeleton: that is points at the end of the skeleton or points where multiple ridges in the skeleton meet. These points can be detected by iterating over the skeleton and marking points which are adjacent to only one or more than two skeleton points, respectively. Thus the algorithm, in pseudocode, becomes:

Algorithm 1 Optimal Sphere-Packing

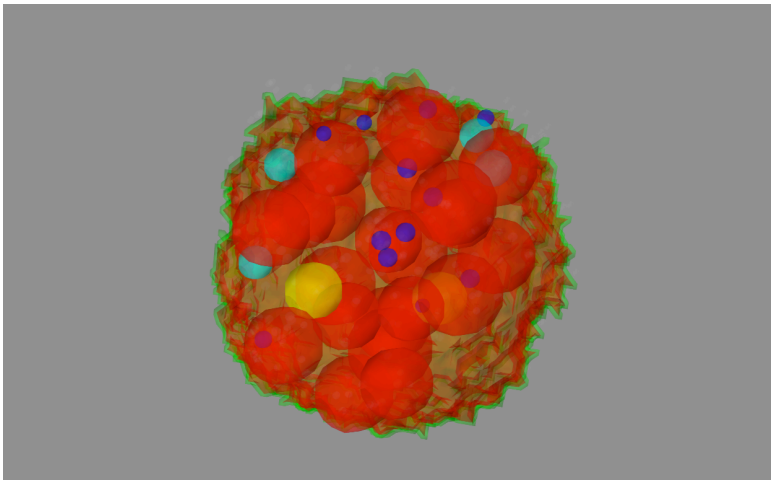
for $S_i \in S$ **do**
  $S_{i,\,spheres} \Leftarrow 0$
  while size $S_i >$ $c\times$size S **do**
    for p in S_i **do**
      $\{\}(p, r)$ denotes the sphere centered at p with radius r
      $p_r \Leftarrow \max\, r \in R \,|\, (p, r)$ in S_i
    end for
    Choose some p such that $p_r = \max\, p_r$ for all $p \in S_i$
    Remove all points within p_r of p from S_i
    Update the boundary and skeleton of S_i
    Increment $S_{i,\,spheres}$
  end while
end for
return $S_j \,|\, S_{j,spheres} = \max\, S_{i,spheres}$ for $S_i \in S$

4 Results

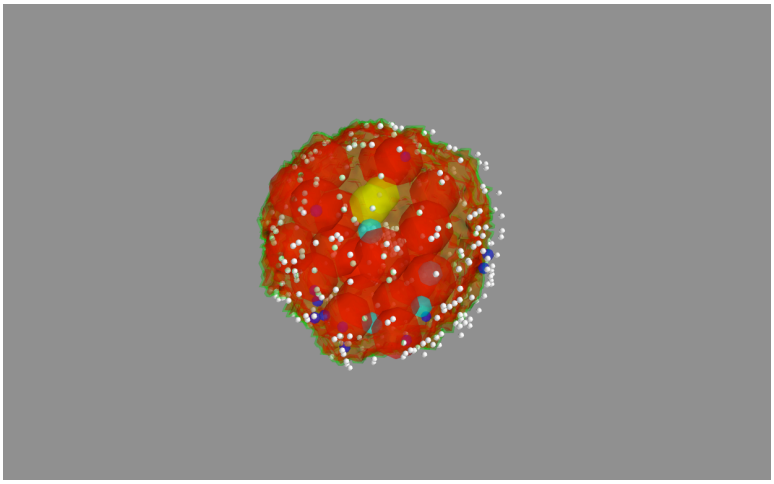
We tested our algorithm on randomly generated tumors each about $8\mathrm{cm}^2$ (8000 points) in volume. An implementation in the Python programming language calculates a sphere-packing on a single core of an Intel 1.83 Ghz Core Duo in 3 to 4 minutes with an additional minute to generate and preprocess the tumor. We were able to achieve at least 90% coverage with between 23 and 30 shots.

Our results are visualized in Mayavi, a Python visualization package, as seen in Fig. 1(a),1(b) and 1(c) below.

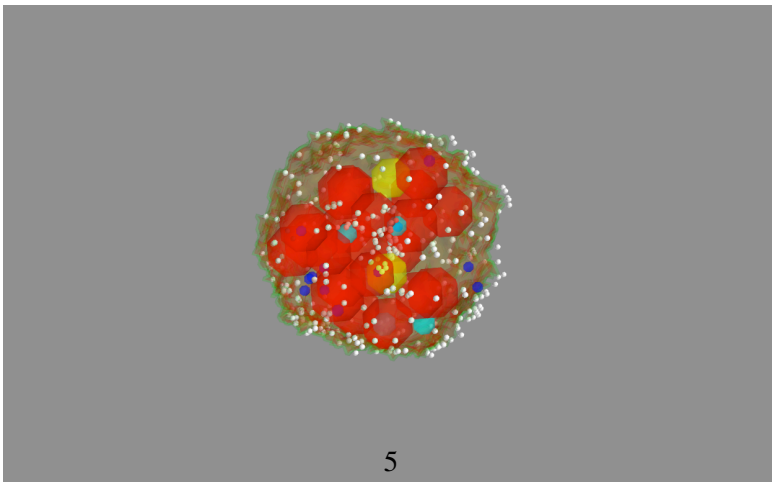
We observe two major flaws in our figures. First, many skeleton points appear at the edge of the tumor. This is a consequence of the algorithm that we use to approximate the distance from the boundary of the tumor to each point inside it when constructing the skeleton and while it creates some unnecessary computation, should not interfere with the quality of the results since our algorithm still recognizes that these points are too close to the boundary of the tumor to place shots at them. Second, some shots protrude outside the tumor. Since we estimate rather than



(a) Tumor with shots (the red shell represents the boundary of the tumor, the red,blue,yellow and blue-green spheres represent the shots of different radii)



(b) Tumor with skeleton and shots



(c) Tumor with skeleton and shots (the skeleton is represented by the small white spheres)

precisely calculate the size of the largest sphere we can place at each point, some protrusion is possible, but the figures show that the spheres protrude farther from the tumors than we expected. We have not yet determined whether this is merely the result of our estimation or a mistake in our implementation.

References

[Bourland and Wu 1996] Bourland, J.D. and Wu, Q. J. "Use of Shape for Automated, Optimized 3D Radiosurgical Treatment Planning." *IEEE Proceedings of International Symposium on Medical Imaging*, pp. 553-558, 1996.

[Bourland and Wu 2000] Bourland, J. D. and Wu, Q. J. "Three-dimensional Skeletonization for Computer-assisted Treatment Planning in Radiosurgery." *Computerized Medical Imaging and Graphics*, v 24, pp. 243-251, 2000.

[Wang] Wang, Jie "Packing of Unequal Spheres and Automated Radiosurgical Treatment Planning" *Journal of Combinatorial Optimization*, vol 3, pp. 453-463, 1999.