

Motion Adapted Reconstruction

Stuart Rowland
Sherif Gadoue
Keith Schubert
Reinhard Schulte
Yair Censor

- Try to reconstruct an object from projections taken while the object is moving.
- As a first effort, we will use a simple moving ellipsoid phantom and straight line projection data.

A solid ellipsoid can be specified with 10 parameters: c_x , c_y , c_z , u , v , w , α , β , γ and density, where

- c_x , c_y , c_z is the translated position of the center of the ellipsoid
- u , v , w are the lengths of the semi axes of the unrotated ellipsoid in the X, Y, and Z directions, respectively
- α , β , γ are the Euler rotation angles in ZYZ order
- density is the density of the object

A simple moving ellipsoid phantom

The moving ellipsoid is arbitrarily defined as

$$cx = -3.6 \cos(2\pi t)$$

$$cy = -1.8 \sin(2\pi t)$$

$$cz = -3 \cos(2\pi t)$$

$$u = 5.5 + 0.5 \sin(2\pi t)$$

$$v = 6.5 + 0.5 \sin(2\pi t + \pi)$$

$$w = 11 + \sin(2\pi t + \pi/2)$$

$$\alpha = \frac{15\pi}{180} \sin(2\pi t + \pi/4)$$

$$\beta = \frac{15\pi}{180} \sin(2\pi t + \pi)$$

$$\gamma = \frac{20\pi}{180} \sin(2\pi t)$$

$$\text{density} = 1$$

The reference image is arbitrarily defined as

$$cx = 0$$

$$cy = 0$$

$$cz = 0$$

$$u = 5.5$$

$$v = 6.5$$

$$w = 11$$

$$\alpha = 0$$

$$\beta = 0$$



















$$\gamma = 0$$

$$\text{density} = 1$$

Note that the moving ellipsoid has a period of 1.

5 instances of the moving ellipsoid are generated at $t=0.0, 0.2, 0.4, 0.6$ and 0.8 .

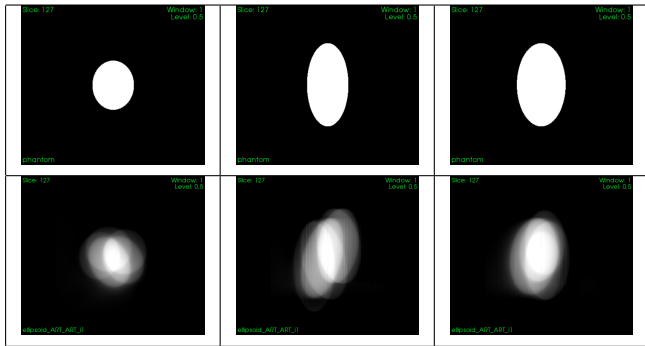
Images of the phantom at the 5 time intervals and the reference image

	t=0.0	t=0.2	t=0.4	t=0.6	t=0.8	reference
Z=0						
X=0						
Y=0						

Projection data collection

We generated noiseless parallel straight line projection data using a circular orbit at 1° intervals over 360 degrees. The projection at angle n was taken with $t = 0.2 \times n$.

Reconstruction from moving data projections



The top row is a repeat of the reference image.

The bottom row is a reconstruction using the moving projection data where the motion was ignored.

Can we do better?

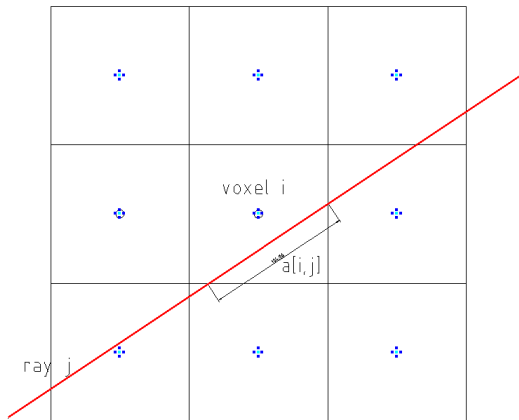
Discrete reconstruction problem

- Let x be a 3D digitized image consisting of I voxels. $x_i \in \mathbb{R}$ is the value of the i th voxel where $i \in [1..I]$.
- Let d be the vector of J ray sums. $d_j \in \mathbb{R}$ is the value of the j th ray, r_j , where $j \in [1..J]$.
- The path of the ray, r_j , that generated ray sum d_j is known.
- Let A be the $I \times J$ projection matrix. $a_{i,j} \in \mathbb{R}$ is the value of the i,j th component of A .
- The discrete reconstruction problem is to find an approximate solution to

$$Ax = d.$$

- There are many algorithms that solve this problem. ART is one of them.

Calculation of elements of A in the normal case



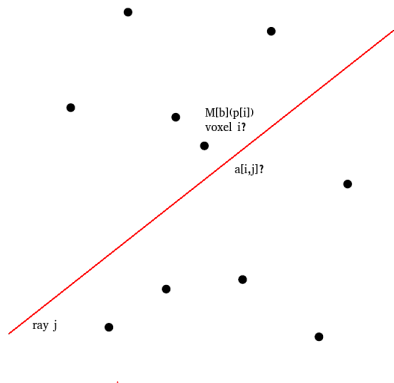
The elements of the matrix A , $a_{i,j}$ are the lengths of the intersection of ray j with voxel i .

For simplicity we use 2D images. 3D is implied.

We assume that:

- there exists a motion model that relates an independent variable or variables to the motion states. In our case, the independent variable is time. Another alternative is volume and flow.
- the motion can be approximated by a finite number, B , of motion bins, M_b , for $b \in [1..B]$ and there is a function, f , that maps the independent variable to a bin number.
- if \vec{p}_i is the position of the center of voxel i in the reference image the motion model gives us $\vec{M}_b(\vec{p}_i)$, the position of \vec{p}_i in motion bin b .
- for every ray, r , the value of the independent variable is known. I.e., there is a function, g , that maps rays to the time when the ray was measured and thus $f(g(r))$ is the corresponding motion bin for ray r .

Transformed locations of voxel centers



An illustration of possible displacement of voxel centers at motion bin b .

Problems: How to define voxel i and the intersection of ray j with that voxel?

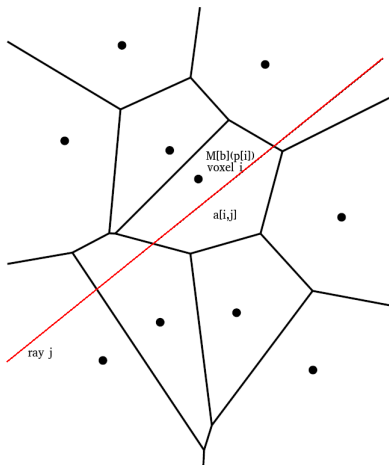
Voronoi diagrams

From wikipedia:

We are given a finite set of points $\{p_1, \dots, p_n\}$ in the Euclidean plane. In this case each site p_k is simply a point, and its corresponding Voronoi cell R_k consists of every point in the Euclidean plane whose distance to p_k is less than or equal to its distance to any other p_k . Each such cell is obtained from the intersection of half-spaces, and hence it is a convex polygon. The line segments of the Voronoi diagram are all the points in the plane that are equidistant to the two nearest sites. The Voronoi vertices (nodes) are the points equidistant to three (or more) sites.

- This definition easily extends to an n -dimensional Euclidean space.
- In 3D, each Voronoi cell is a convex polyhedron.

Voronoi diagram of transformed voxel centers



The Voronoi cells for the previous distribution of points.

Problem: How to find the boundaries of the Voronoi cells?

The Fortune algorithm can generate the Voronoi diagram in 2D. It does not extend to 3D.

Delaunay triangulation

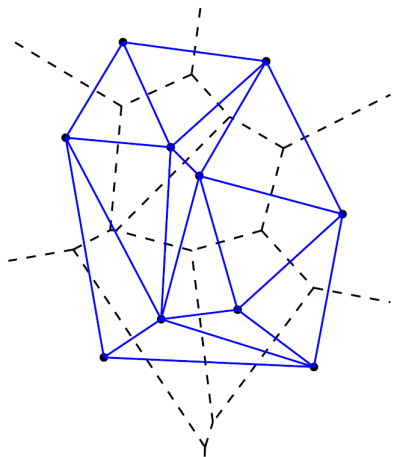
From wikipedia:

In mathematics and computational geometry, a Delaunay triangulation (also known as a Delone triangulation) for a given set P of discrete points in a plane is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$.

- The triangulation is unique unless 4 points lie on a circumcircle.
- The triangulation extends to 3D with triangles replaced by tetrahedrons and circles with spheres.
- In 3D the Delaunay tetrahedralization is unique unless 5 points lie on a circumsphere.
- For our purposes, uniqueness is unimportant. When multiple solutions exist, any one of them is acceptable.

Two problems are duals when the solution of one provides the solution to the other.

Voronoi diagrams and Delaunay triangulation are duals

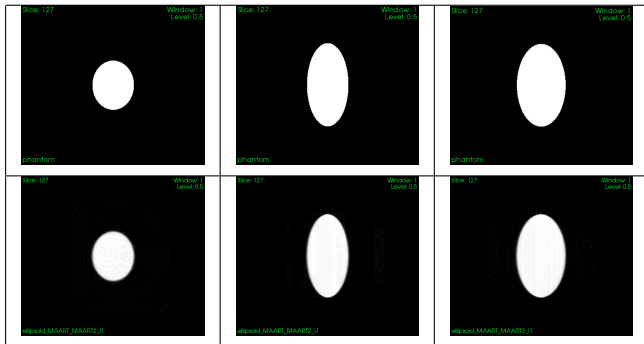


The Computational Geometry Algorithms Library (www.cgal.org) includes a C++ implementation of an algorithm to compute the Delaunay tetrahedralization.

Intersection of a ray with a Voronoi cell

- Intersect the ray with each of the planes that define the sides of the Voronoi cell.
- Test each intersection to see if it is in the Voronoi cell.
- There will be zero or two such points.
- When there are two points, save the neighboring cells on the other side of the two planes.
- $a_{i,j}$ is the distance between the the two points.
- The saved cells are the previous and next cells intercepted by the ray.

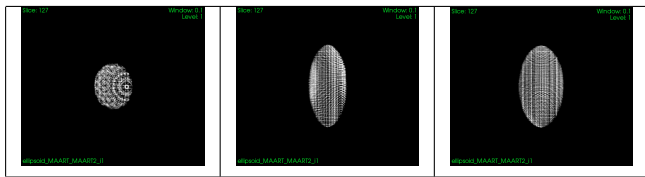
Motion Adapted Reconstruction



The top row is a repeat of the reference image.

The bottom row is a reconstruction from the moving data using the modified projection matrix.

But it is not perfect



Previous image, but with level=1 and window=0.1.

- Given a motion model, we have shown that using Voronoi cells we can remove the effect of motion at the cost of other artifacts.
- Since the only change was to modify the projection matrix A , the method can be used with any iterative reconstruction algorithm.

- Identify the cause of the artifacts and try to remove them.
- Apply these concepts to more realistic pCT data and reconstruct using MLP.