VIBE

Visualization of Intrafraction Behavior from Electromagnetic Tracking

Tamar Nevo

Presentation Overview

- Define the problem & medical background
- Identify goals of the project
- Outline my process
- Demo the final animation
- Discuss medical contributions
- Suggest future directions

Problem Definition

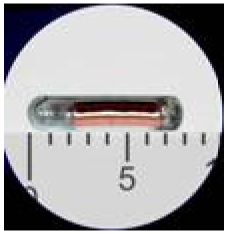
- Organs naturally move during the course of radiation treatments
  - Sidenote: Intrafraction motion = organ movement within one treatment on a given day
  - Interfraction motion = organ movement between treatments on different days
- Though more obvious in organs like the lungs, which expand and contract with breathing, other organs, like the prostate, also demonstrate movement
- Radiation that reaches non-target (non-cancerous) tissue may result in various side effects
- There has been an effort in radiation therapy to improve the precision of cancer treatments, reducing side effects and better controlling tumors

Background

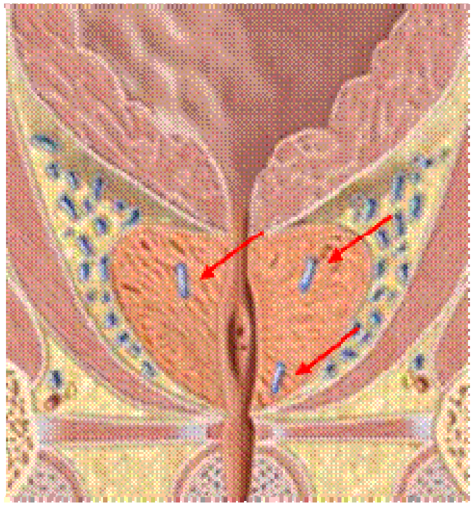
- Accuracy of radiation is challenging due to natural organ movement during treatments
- Radiology Oncologists are working to improve the precision of cancer treatments – to better target tumor-bearing tissue and to reduce the unintentional doses reaching normal tissue – by tracking the motion and deformation of the cancerous organ
- The Calypso® 4D Localization System uses electromagnetic sensors to track the exact position and motion of the organ in real-time
- For this, it is sometimes described as "GPS for the Body®"

How does Calypso® work?

- Prior to treatment, three Beacon® transponders are implanted into the target tissue, in this case, the prostate gland
- Beacon® transponders are tiny electromagnetic sensors, which monitor the position and motion of the organ
- Through safe radiofrequency waves, the Calypso System tracks and records the location of each transponder
- Can be thought of as “motion capture for organs”



Beacon® transponder
(8 mm in length)



Project Goals

- Understand intrafraction prostate motion by visualizing and reanimating organ contours
- Provide results for Radiology Oncologists, allowing them to interpret the outcomes and determine their practical application and significance
- Presenting data with which to identify potential motion patterns

Tools & Languages

- Calypso 4D Localization System
- Beacon® Transponders
- MATLAB
- Excel (VBA)

Resampling the Data

- The Calypso machine records and outputs the location of each transponder sequentially so the original excel document looks like the following:

Transponder	X	Y	Z	PositionTimeString
1	0.768007685	-0.670575507	1.237602205	2008-02-12T15:14:24.9445000-05:00
2	0.543598218	0.305644489	-1.031243917	2008-02-12T15:14:25.0355000-05:00
3	-1.595396957	0.522831979	-0.420960678	2008-02-12T15:14:25.1165000-05:00
1	0.773501245	-0.685332166	1.253303285	2008-02-12T15:14:25.2385000-05:00
2	0.530271027	0.278862085	-1.017448201	2008-02-12T15:14:25.3195000-05:00
3	-1.606955031	0.507740164	-0.432733083	2008-02-12T15:14:25.4005000-05:00

- Want location of transponders at the **same instance** to track movement and deformation of three-transponder triangle over a period of time

- In order to obtain this triangle, the data must be resampled

Resampling the Data

- Generate b-spline curves to explain the position and movement of each transponder independently in order to retrieve its position at any time during the 18 minute interval

- Using VBA code, create separate sheets that isolate each transponder's x,y,z position, like so:

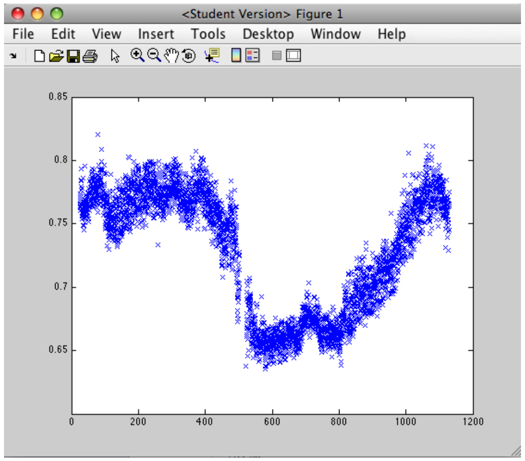
Transponder	X	Y	Z	PositionTimeString
1	0.768007685	-0.670575507	1.237602205	2008-02-12T15:14:24.9445000-05:00
1	0.773501245	-0.685332166	1.253303285	2008-02-12T15:14:25.2385000-05:00
1	0.765768046	-0.703897066	1.261517722	2008-02-12T15:14:25.5525000-05:00
1	0.770603624	-0.703192297	1.257850022	2008-02-12T15:14:25.8365000-05:00
1	0.771952138	-0.705367329	1.250430385	2008-02-12T15:14:26.1505000-05:00

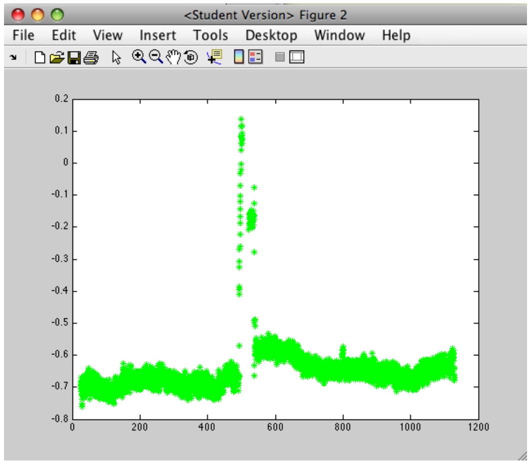
- Format allows me to copy an entire column (about 3650 times) into an array and graph the movement along one of the axes over time

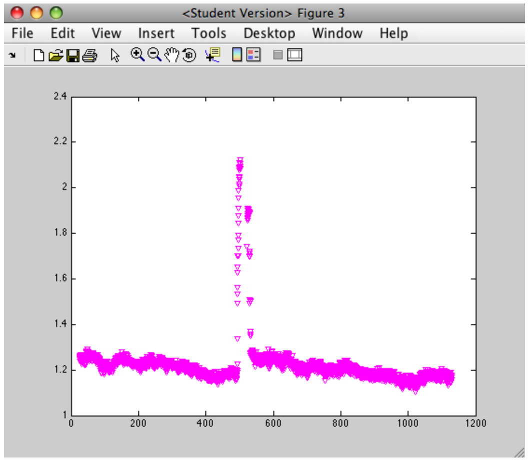
Resampling the Data

- Convert timestamps from 2008-02-12T15:14:24.9445000-05:00 to seconds using another VBA parsing function
- Remove spurious data (specifically zeros, where the machine might have faulted)
- Ready to graph

Resulting Graphs

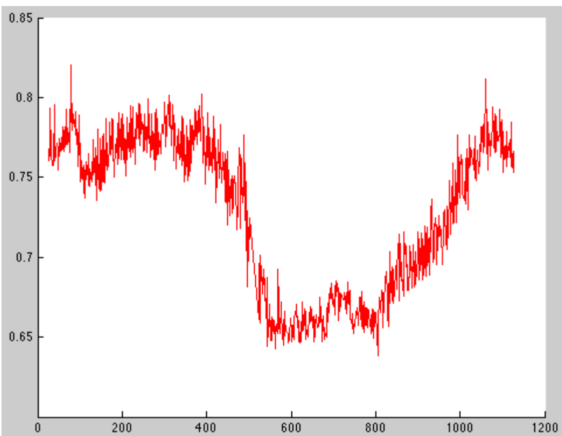


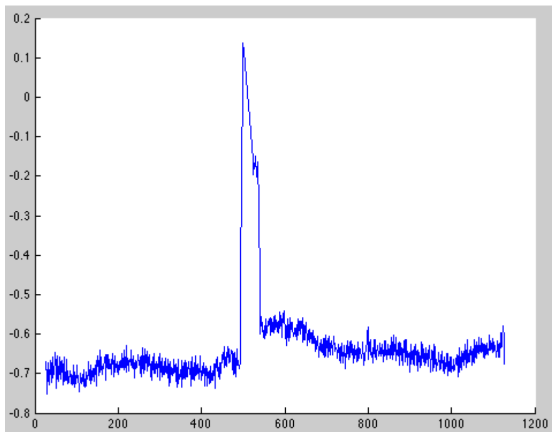


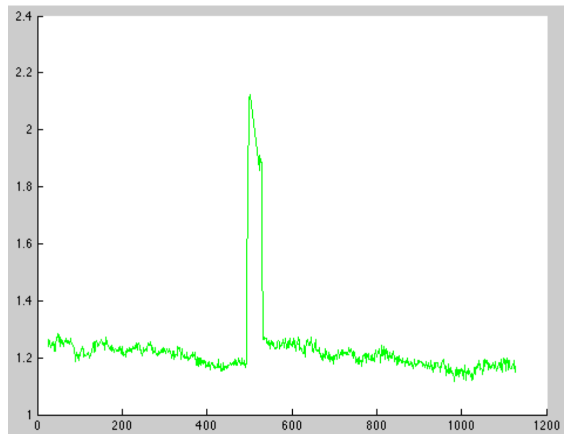


- X, Y, and Z position graphs for Transponder 1 demonstrate some shift around 500 seconds
- Similar graphs are generated for T2 and T3 (9 total)

Resulting Graphs







- Transponder 1 b-spline curves, which can later be interpolated from to find positions at any time

Interpolation

- Interpolate from each of the 9 graphs to obtain the three-dimensional locations of each transponder at every second, generating three points to create a triangle

T1			T2			T3		
x	y	z	x	y	z	x	y	z
0.7677	-0.6713	1.2385 ;	0.5427	0.3005	-1.0336 ;	-1.6027	0.5202	-0.4254 ;
0.7625	-0.6838	1.2551 ;	0.5427	0.3005	-1.0336 ;	-1.6027	0.5202	-0.4254 ;
0.7623	-0.6899	1.2590 ;	0.5409	0.2947	-1.0305 ;	-1.6073	0.5170	-0.4247 ;
0.7669	-0.6938	1.2542 ;	0.5382	0.2886	-1.0218 ;	-1.6066	0.5136	-0.4160 ;
0.7857	-0.7359	1.2553 ;	0.5484	0.2296	-1.0264 ;	-1.5884	0.4618	-0.4088 ;
0.7824	-0.7274	1.2477 ;	0.5505	0.2276	-1.0324 ;	-1.5750	0.4530	-0.4146 ;
0.7660	-0.6884	1.2363 ;	0.5445	0.2795	-1.0389 ;	-1.5701	0.5065	-0.4369 ;
0.7635	-0.6767	1.2409 ;	0.5350	0.3304	-1.0342 ;	-1.5994	0.5270	-0.4460 ;
0.7610	-0.6877	1.2461 ;	0.5269	0.3151	-1.0342 ;	-1.6127	0.5200	-0.4471 ;
0.7586	-0.7098	1.2502 ;	0.5199	0.2539	-1.0372 ;	-1.6070	0.4856	-0.4406 ;
0.7579	-0.7243	1.2456 ;	0.5131	0.2594	-1.0337 ;	-1.6037	0.4787	-0.4362 ;
0.7578	-0.7055	1.2492 ;	0.5193	0.2828	-1.0385 ;	-1.6053	0.4977	-0.4388 ;
0.7581	-0.6752	1.2542 ;	0.5338	0.3153	-1.0473 ;	-1.6117	0.5411	-0.4481 ;
0.7594	-0.6686	1.2465 ;	0.5388	0.3118	-1.0380 ;	-1.6074	0.5301	-0.4453 ;
0.7765	-0.6923	1.2463 ;	0.5399	0.2912	-1.0419 ;	-1.6055	0.5122	-0.4496 ;
0.7906	-0.7162	1.2451 ;	0.5380	0.2669	-1.0523 ;	-1.6057	0.4897	-0.4599 ;
0.7664	-0.6899	1.2273 ;	0.5321	0.2940	-1.0499 ;	-1.6040	0.5033	-0.4703 ;

Filtering Data

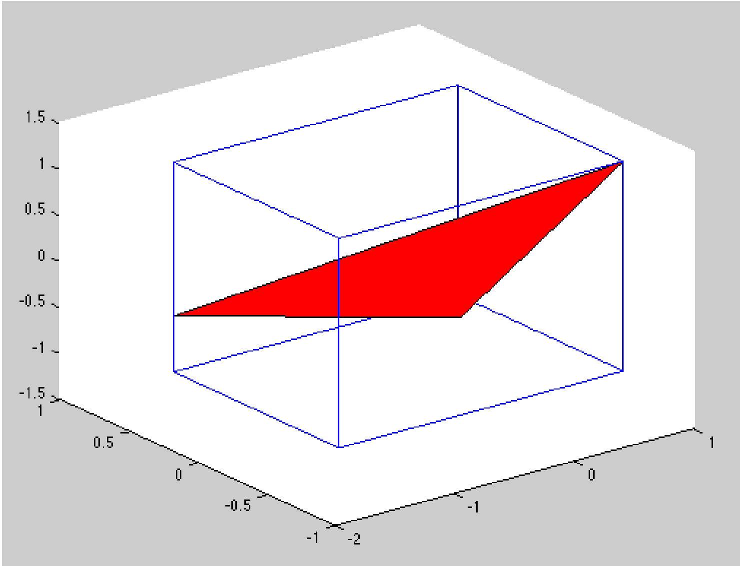
- Because the accuracy of the machine is .1 mm, much of the perceived movement is machine noise
- Format excel columns to display only 1 decimal place

T1			T2			T3		
x	y	z	x	y	z	x	y	z
0.8	-0.7	1.2 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.3 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.3 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.3 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.3 ;	0.5	0.2	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.2 ;	0.6	0.2	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.2 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.2 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.2 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.3 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.2 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.2 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;
0.8	-0.7	1.3 ;	0.5	0.3	-1.0 ;	-1.6	0.5	-0.4 ;

Animating Triangle

- With three points in space, can plot triangle
- Calculate mins and maxs and from this, compute bounding box

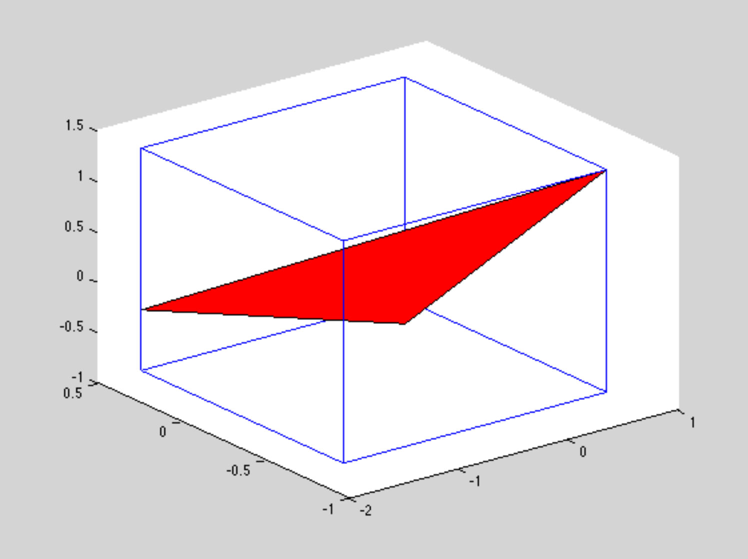
```
function box = boundingBox(P1, P2, P3)

    %triangle vectors
    x = [P1(1) P2(1) P3(1)];
    y = [P1(2) P2(2) P3(2)];
    z = [P1(3) P2(3) P3(3)];

    %calculate all 6 mins and maxs
    mins = calcMins(P1, P2, P3);
    minX = mins(1); minY = mins(2); minZ = mins(3);
    maxs = calcMaxs(P1, P2, P3);
    maxX = maxs(1); maxY = maxs(2); maxZ = maxs(3);

    %square vectors
    xbox1 = [minX maxX maxX minX minX];
    ybox1 = [minY minY minY minY minY];
    zbox1 = [maxZ maxZ minZ minZ maxZ];

    ybox2 = [maxY maxY maxY maxY maxY];

    xbox3 = [maxX maxX maxX maxX maxX];
    ybox3 = [minY maxY maxY minY minY];

    xbox4 = [minX minX minX minX minX];

    %plot all four squares to create cube and plot triangle
    figure(1);
    axis('square');

    fill3(x,y,z,'r');
    hold on
    plot3(xbox1,ybox1,zbox1,'b',xbox1,ybox2,zbox1,'b',xbox3,ybox3,zbox1,'b',xbox4,ybox3,zbox1,'b');
    hold off

box = 0;

end
```



Animating Triangle

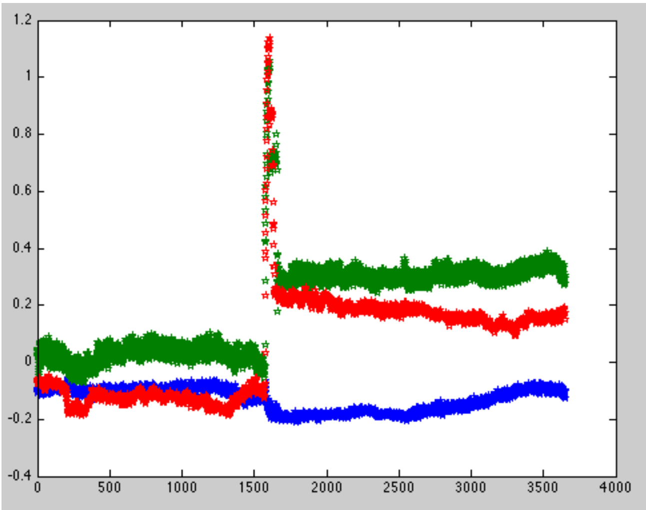
- To see the changes over time, I wrote a function called “animate,” which creates an .avi file of a sequence of all of the frames

```
function animation = animate(P1, P2, P3)

[length, three] = size(P1);
aviobj=avifile('test2.avi');
hf= figure('visible','off');

for i = 1:length

    p1 = [P1(i,1), P1(i,2), P1(i,3)];
    p2 = [P2(i,1), P2(i,2), P2(i,3)];
    p3 = [P3(i,1), P3(i,2), P3(i,3)];
    boundingBox(p1,p2,p3);

    aviobj=addframe(aviobj,hf);

end

%movie(M);

aviobj=close(aviobj);

end
```

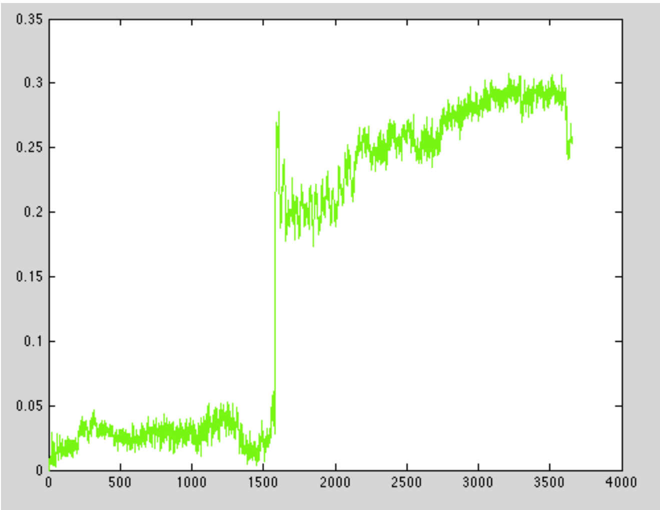


Finding Transformation Matrices

- First plot graphs for 1) Centroid movement in the X, Y, and Z directions, 2) Rotation between each frame and the first, and 3) scale change in each direction
- Because the scale change is mainly constant in all dimensions, Dr. Badler and I chose only to consider translation and rotation



Centroid Movement



Rotation

Finding Transformation Matrices

- Next step: for each triangle, compute 1) translation and 2) rotation matrix between first and current frame
- Use tMatrices, rMatrices functions to output an array of matrices for each frame, which can then be applied in a for-loop to animate the contour

Finding Transformation Matrices

```
function matrices = rMatrices(P1, P2, P3)

[length, three] = size(P1);
matrices = zeros(4,4,length-1);

normals = calcNormals(P1, P2, P3);
angles = plotAngles(P1, P2, P3);
centroid = calcCentroids(P1, P2, P3);

for i = 1:length-1
    v1 = [normals(i,1) normals(i,2) normals(i,3)];
    v2 = [normals(i+1,1) normals(i+1,2) normals(i+1,3)];

    %find rotation axis by taking cross product of consecutive frames'
    %normals
    w = cross(v1, v2);
    wLength = sqrt(w(1)^2 + w(2)^2 + w(3)^2);
    if wLength ~= 0
    w = w/wLength;
    end

    a = w(1);
    b = w(2);
    c = w(3);
    theta = angles(i);

    %use values from the rotation axis vector (w = [a,b,c]) and the angle
    %between the two normals to plug into rotation matrix; combine values
    %for translation
    m = [a^2+(1-a^2)*cos(theta), a*b*(1-cos(theta))-c*sin(theta), a*c*(1-cos(theta))+b*sin(theta), 0;
         a*b*(1-cos(theta))+c*sin(theta), b^2+(1-b^2)*cos(theta), b*c*(1-cos(theta))-a*sin(theta), 0;
         a*c*(1-cos(theta))-b*sin(theta), b*c*(1-cos(theta))+a*sin(theta), c^2+(1-c^2)*cos(theta), 0;
         0, 0, 0, 1];
    matrices(:,:,i) = m;
end
end
```

```
function matrices = tMatrices(P1, P2, P3)

[length, three] = size(P1);
matrices = zeros(4,4,length-1);

centroid = calcCentroids(P1, P2, P3);

for i = 1:length-1
    Tx = centroid(i+1,1) - centroid(1,1);
    Ty = centroid(i+1,2) - centroid(1,2);
    Tz = centroid(i+1,3) - centroid(1,3);

    m = [1 0 0 Tx; 0 1 0 Ty; 0 0 1 Tz; 0 0 0 1];

    matrices(:,:,i) = m;
end
end
```

Displaying Contours

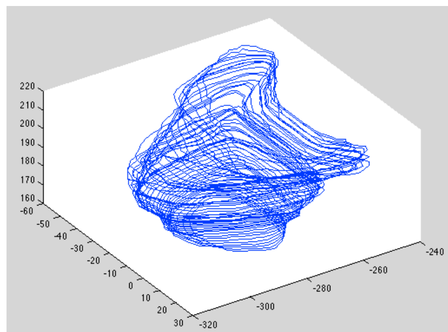
Data provided:

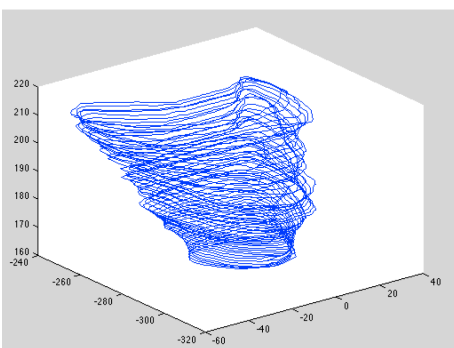
X	Y	Z
-10.205	-295.748	164.664
-8.057	-295.761	164.664
-5.908	-295.569	164.664
-3.959	-295.221	164.664
-3.76	-295.179	164.664
-1.611	-294.674	164.664
0.537	-294.105	164.664
2.686	-293.528	164.664
4.441	-293.072	164.664

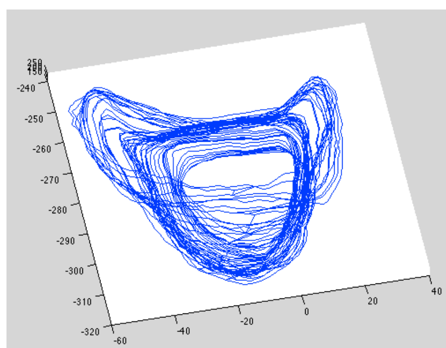
- Each row represents a point along one contour loop, and each loop is separated by a blank row
- MATLAB does not recognize blank rows, so I replaced blanks with an indicator (arbitrarily 5000000) so that a function could recognize the end of a contour loop
- Before animating, I tested the 3-D contour display with the function drawContour

Displaying Contours

drawContour generates a 3-D wireframe of the prostate, which can be rotated in MATLAB to display different perspectives







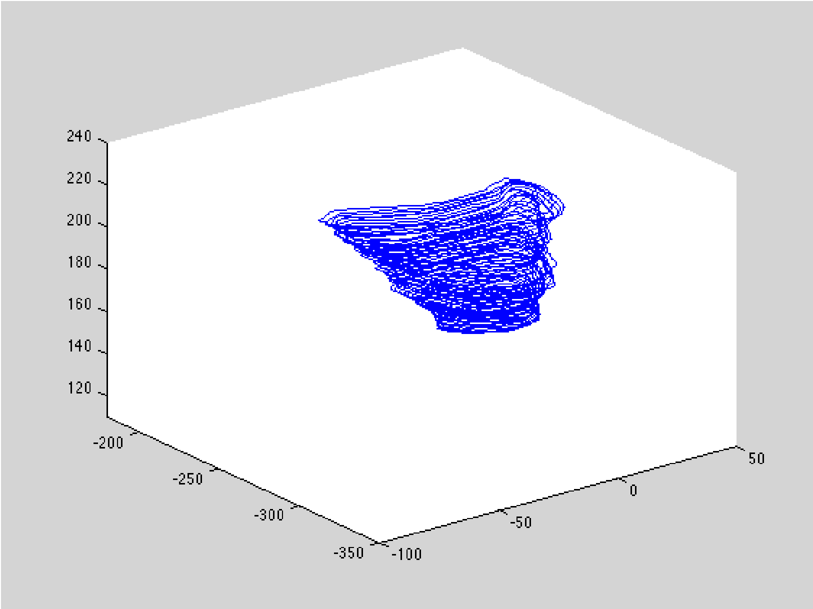
```
function contour = drawContours(X,Y,Z)
[length, one] = size(X);
temp = 0;
firstX = X(1);
firstY = Y(1);
firstZ = Z(1);

%make each contour line form a full circle
%by filling in the blank with the first
%element of the loop
for i = 1:length

    if X(i) == 5000000
        X(i) = firstX;
        Y(i) = firstY;
        Z(i) = firstZ;
        if i ~= length
        firstX = X(i+1);
        firstY = Y(i+1);
        firstZ = Z(i+1);
        end
        %draw the contour lines
        for j = 1:i-temp
            a(j) = X(j+temp);
            b(j) = Y(j+temp);
            c(j) = Z(j+temp);
        end
        plot3(a,b,c);
        hold on;

        temp = i;
    end
end
hold off;
end
```

Animating Contours

- Ran into several problems
  - "Frozen screen" effect because of overriding initial array that contains indicators
  - Correctly displaying wireframe and holding axes constant
  - Jumping around – incorrect rotation matrix

Animating Contours

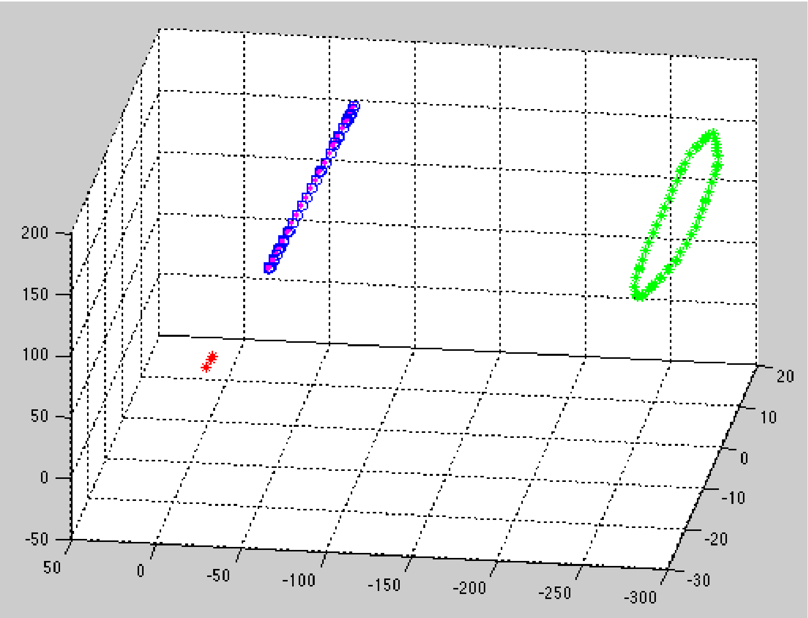
```
for i = 1:time-1
    temp = 0;
    firstX = origX(1);
    firstY = origY(1);
    firstZ = origZ(1);
    m = tm(:,:,i)*rm(:,:,i)*tim(:,:,i);
    for j = 1:numPoints
        if origX(j) == 5000000
            X(j) = firstX;
            Y(j) = firstY;
            Z(j) = firstZ;
            if j ~= numPoints
                firstX = origX(j+1);
                firstY = origY(j+1);
                firstZ = origZ(j+1);
            end

            %multiply by matrix
            newX(j) = m(1,1)*X(j) + m(1,2)*Y(j) + m(1,3)*Z(j) + m(1,4);
            newY(j) = m(2,1)*X(j) + m(2,2)*Y(j) + m(2,3)*Z(j) + m(2,4);
            newZ(j) = m(3,1)*X(j) + m(3,2)*Y(j) + m(3,3)*Z(j) + m(3,4);

            %draw the contour lines
            for k = 1:j-temp
                xLoop(k) = newX(k+temp);
                yLoop(k) = newY(k+temp);
                zLoop(k) = newZ(k+temp);
            end
            axis([-100 50 -350 -180 110 240]);
            plot3(xLoop,yLoop,zLoop);
            hold on;
            clear xLoop yLoop zLoop

            temp = j;

         else
            %multiply by matrix
            newX(j) = m(1,1)*X(j) + m(1,2)*Y(j) + m(1,3)*Z(j) + m(1,4);
            newY(j) = m(2,1)*X(j) + m(2,2)*Y(j) + m(2,3)*Z(j) + m(2,4);
            newZ(j) = m(3,1)*X(j) + m(3,2)*Y(j) + m(3,3)*Z(j) + m(3,4);
         end
    end
    hold off;
    aviobj=addframe(aviobj,hf);

end
```

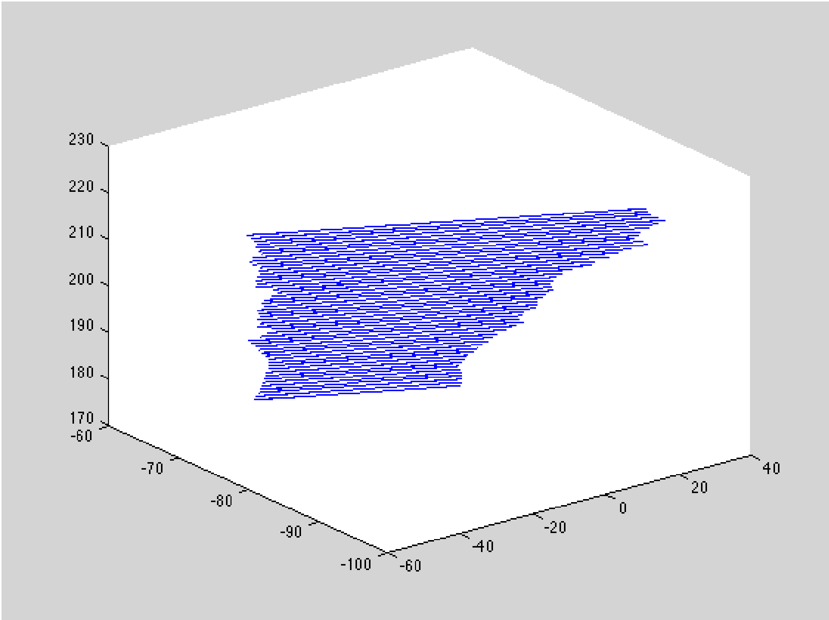


Issue with Transformation

- Forced fitting
  - Neglecting scale change/deformation
  - Causes forced rotation
  - Working on computing ONE transformation matrix that takes into account shearing/scaling/compression, in addition to rotation and translation
- Scale disparity
  - Small triangle controlling large contour
  - Tiny shift in triangle translates into magnified shift in contour

Issue with Transformation





Contributions

- Ability to visualize organ motion from spaciotemporal data, providing a better understanding of intrafraction prostate motion
- Computing clinically useful measures
  - centroid movement, rotation angles, min/max displacement
- Opportunity for identifying patterns of behavior and improving treatment accuracy
- Published in an abstract submitted to ASTRO, the American Society for Therapeutic Radiology and Oncology and another anticipated publication in the near future

Development of a Novel System for Visualizing Prostate Motion in Patients Undergoing Radiotherapy with Electromagnetic Target Localization and Tracking

Author Block R. R. Rajendran, T. Nevo, E. Rubin, A. Kassaee, N. Badler, N. Vapiwala

University of Pennsylvania Medical Center, Philadelphia, PA

Future Directions

- Fixing "jumpiness" of current animation
- Future application by Radiology Oncologists in effort to reduce error and improve accuracy/effectiveness of therapy
- Ability to import data files and run program directly, essentially reducing number of steps
- Build an interface that displays useful outcomes
- Real-time animation and ultimately automated target monitoring and radiation beam adjustment during treatment

Acknowledgements

- **Dr. Norm Badler**, project advisor, who attended all hospital meetings with me and kept me on track throughout the semester with weekly conferences and daily encouragements
- **Professor Jianbo Shi**, master of MATLAB, who made himself beyond available, patiently and relentlessly sitting with me for hours debugging code
- The team at HUP – **Ramji Rajendran, Ali Kassaee, and Neha Vapiwala** – for being such a pleasure to work with, guiding me and patiently explaining all of the medical terminology
- My friends and family for their support