Simulation of The Detection of Noxious Breast Cancer

M. E.Wahed

Computer Science department, Suez Canal University, Egypt
M.Abdallah

Computer Science department, Suez Canal University, Egypt
Mohamed Soliman Elkomy

Computer Science department, Suez Canal University, Egypt

Abstract
We present a method to help diagnose breast cancer. Test data is generated with the help of the Laplacian Framework. Then the test data is used to train a neural network so that it is able to classify patients that have or do not have breast cancer.

1. Introduction:

BreastCancerCare.org.uk states that a small cancer may grow very quickly or a larger cancer may have been growing slowly over a longer time. Due to the sensitivity of the breast area, women are reluctant in going through the examination process. In this paper we present a method based on a 3D scan of the breast area without human intervention, where a patient can follow up on tumor size change in the breast area over time.
Two of the Signs and Symptoms that Doctors Hospital at Renaissance (http://www.dhr-rgv.com/):

» A lump, mass, or thickening in the breast
» Change in the size or shape of a breast

Our method will be very effective in identifying such changes that happen in the breast area.

2. Laplacian Framework:

Surface representation and processing is one key topic in computer aided design. The surface representation of a 3D object may affect the information that we can perceive about that object. For example the triangular mesh representation can be used to: display the surface, deduce some topological information about the object, in addition to knowing the differential properties of the object that the model represents.
In this paper we build on work done on mesh processing and modeling that is based on the Laplacian framework and differential representations pointed out by [Olg05] and [ATOM06]. As opposing to dealing with Cartesian coordinates, the differential representation utilized in the Laplacian framework results in detail-preserving operations.
Laplacian operator and differential surface representation and surface reconstruction:
To understand the work that this paper presents we will first talk about the Laplacian differential surface representation.
If " M " is a mesh representing an object with " V " vertices and "E " edges and " F " faces. For each vertex " $\mathbf{v_i}$ " we have three Cartesian coordinated associated with each vertex: x_i, y_i , and z_i.

The differential or δ-coordinates of $\mathbf{v_i}$ is defined to be the difference between the absolute coordinates of $\mathbf{v_i}$ and the center of mass of its immediate neighbors in the mesh

$$\delta_i = \left(\delta_i^{(x)} + \delta_i^{(y)} + \delta_i^{(z)} \right) = \mathbf{v_i} - \frac{1}{d_i} \sum_{j \in N(i)} \mathbf{v_i}$$

Where N(i) = { j |(i, j) ∈ E} and di = |N(i)| is the number of immediate neighbors of i (1-ring of the vertex). The transformation of the vector of absolute Cartesian coordinates to the vector of δ -coordinates can be represented in matrix form. Starting from the adjacency matrix A which is a square matrix that has 1 in the cell if both the row "i" and column "j" of the cell in the matrix represent an edge between the two vertices i, and j.

Also we have the diagonal matrix D that has $D_{ii} = d_i$, hence we can write the L matrix
$L = I - D^{-1}A$
And the symmetric version L_s
$L_s = DL = D - A$
Then we can write:

$$L\,x = \delta^x$$
$$L\,y = \delta^y$$
$$L\,z = \delta^z$$

Manuscript received September 5, 2014
Manuscript revised September 20, 2014

We cannot restore the Cartesian coordinates starting from δ –coordinates; because L is singular. In order to restore the Cartesian coordinates we need to specify the Cartesian coordinates of one vertex to resolve the translational degree of freedom. Substituting the coordinates of vertex i is equivalent to dropping both the *i*th row and column from L, which makes the matrix invertible [Olg05]. Usually how this is done is by placing more than one special constraint of the mesh vertices. We have therefore |C| additional constraints (called the positional constraints) of the form:

$$\mathbf{v_j} = \mathbf{c_j}\ , \mathbf{j} \in \mathrm{C}$$

If the vertices are ordered from 1 to m, then the linear system looks like:

$$\begin{pmatrix} L \\ \omega I_{m*m} | \mathbf{0} \end{pmatrix} x = \begin{pmatrix} \delta^x \\ \omega c_{1:m} \end{pmatrix} \qquad (1)$$

The additional constraints make the linear system over-determined (more equations than unknowns) and in general no exact solution may exist. However, the system is full-rank and thus has a unique solution in the least-squares sense:

$$\check{x} = argmin_x \left(\|Lx - \delta^x\|^2 + \sum_{j \in C} \omega^2 |x_j - c_j| \right)$$

(1) Can be written on the following form

$$Ax = b \qquad (2)$$

3. Least squares Perturbation and Laplacain Mesh Processing:

I. The Effect of delta Changes on Relative Error

1. The Effect of Changes in b on Relative Error

In this section we consider what if questions applied to equation (2). For example what if want to make changes on the model under investigation, the question that arises do we need to solve the new problem from the beginning. [Fas13] states that the liner system $Ax = b$ may be perturbed as $A(x + \delta x) = (b + \delta b)$ this implies that $A\delta x = \delta b$, and hence we can solve this linear equation to get δx and add to solution of the original problem and get $x + \delta x$, i.e the solution of the perturbed problem.

2. The Effect of Changes in A, and b on Relative Error

If we solve (1) without including all the positional constraints we will have a deformed object of the original object. So what if we want to add more constraints to our problem latter. Again the question arises: Do we need to solve a new problem from the beginning? The answer is no, we can work on the perturbed system $(A + \delta A)(x + \delta x) = (b + \delta b)$ to find:

$$\delta x = (A + \delta A)^{-1}(\delta b - \delta A x)$$

II. Implementation details and Results

We have used Matlab, and Graph tool box created by "Gabriel Peyre". To calculate the least squares solution "mldivide" command was used. Due to the nature of the problem the solver used is Sparse QR Factorization via "SuiteSparseQR".

1. The Effect of Changes in b on Relative Error

- The changes in b can result from multiple factors: aging that occurred on an object, or simply because we want to make a change on the object replacing a piece by another piece.
- In figure (1) we see the effect of modifying part of the model
- The Implementation details go as follows:

(1) Solve the original problem $Ax = b\ to\ get\ x$
(2) Introduce a delta change on the b part in a limited number of vertices
(3) Calculate δb
(4) Calculate δx from $\delta x = A^{-1}\delta b$
(5) Add $x + \delta x$

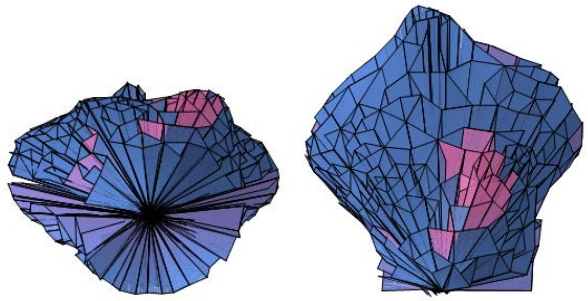


Figure (1), on the top a bottom view to show the incase in size of the breast, on the bottom the red area shows where the tumor exits

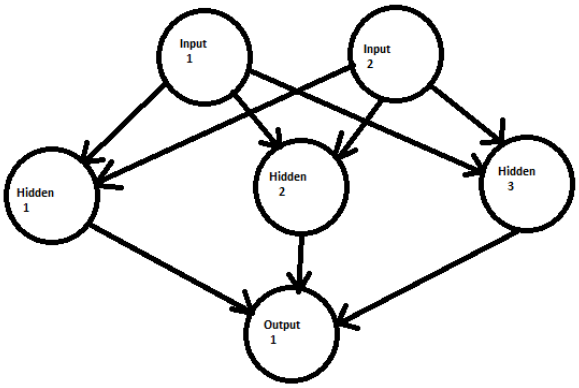
As indicated in the following table, the difference between values before applying the changes to the model and after Appling the changes to the model. The changes appear in the z component of the breast affected area of the model.

4. Feed Forward Back propagation Neural Network:

I. Introduction:

In “feed forward”, neurons are connected foreword. Each layer contains connections to the next layer. The term back propagation describes how the neural network is trained. It is a supervised training method, the network must be provided with both inputs and anticipated outputs, and the back propagation training algorithm then takes the calculated error and adjusts the weights of various layers backwards from the output layer to the input layer.

Values with delta changes	Values without delta changes	Values with delta changes	Values without delta changes	Values with delta changes	Values without delta changes
0.60824	0.60711	0.14036	0.14	0.14029	0.13966
0.54745	0.54623	0.070222	0.069529	0.31594	0.31536
0.51604	0.51516	-0.017041	-0.017617	0.44149	0.44075
0.48689	0.48571	0.30394	0.30427	0.56147	0.56086
0.35449	0.35398	0.39946	0.39955	0.48003	0.47956
0.41426	0.41339	0.47032	0.46925	0.29299	0.29268
0.27682	0.2761	0.50747	0.50657	0.64017	0.63964

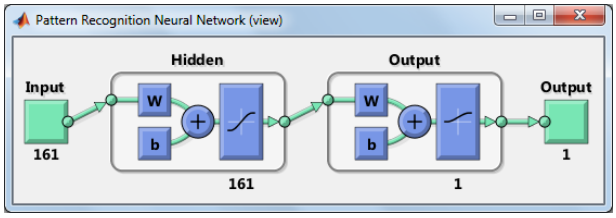


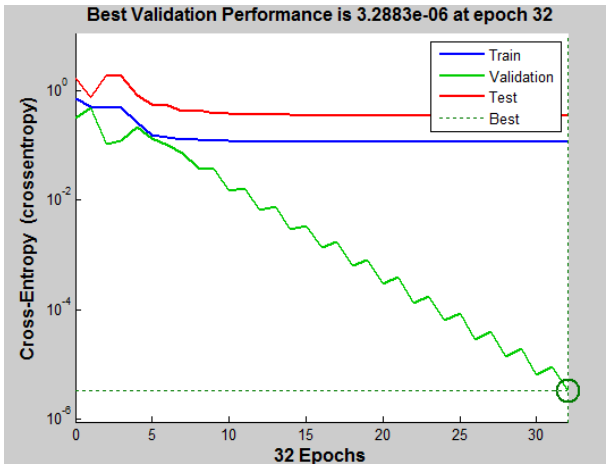
A feed forward neural network with one hidden layer

II. Choosing The Network Structure:

A two layer feed forward neural network can be trained to react to a given input pattern with a prescribed output response. The hidden layer maps an input vector onto one of the vertices of a unit hyper cube. The output neuron realizes a hyper plane to separate vertices according to the different class that they belong to [TK09].

We used Matlab as a pattern recognition tool to train a two layer neural network.
The input is the model with a tumor represented in a 30% increase factor of the natural body in one of four parts of the breast area. The input layer consists of 161 inputs. We apply the laplacian framework to the model with delta changes. We divide the breast area into four parts. We also use four values of ω from equation(1) which produces 16 models. The models that don’t have cancer are produced using the laplacian framework without delta changes, again with ω having four values. The hidden layer contains 161 elements. The network output is a layer composed of one output. The output one if there is tumor, zero otherwise.





III. Results:

The training set was divided into two halves. One half is used for training, the other half is used for testing and validation.
After training the network it was able to identify the models that have cancer from the others that don’t have cancer.

5. Conclusion:

We have shown how to generate patient and healthy models of the breast area using the Laplacian framework. We then used a two layer neural network to classify sample patients as having cancer. The network produces output proportional to the deformation caused by the tumor which helps the patient to keep track of deformation introduced by time that may be caused by cancer.

References:

[BGAA12] Baerentzen J., Gravesen J., Anton F., Aanaes H.: Guide to Computational Geometry Processing. Springer (2012)
[BKPAL 10] Botsch M., Kobbelt L., Pauly M., Alliez P., Levy B.: Polygon Mesh Processing. AK Peters,Wellesley (2010)
[Fas13] Fasshauer G.: online notes for course MATH 477/577 at IIT
[Hea97] Heath M.: Scientific Computing An Introductory Survey. McGraw-Hill (1997)
[HJ08] Heaton J.: Introduction to Neural Network for Java. Heaton Research (2008)
[HL95] Hillier F., Lieberman G.: Introduction to Operations Research. McGraw hill (1995)
[Mey00] Meyer C. Matrix Analysis and Applied Linear Algebra. SIAM (2000)
[NISA06] Nealen A., Igarashi T., Sorkine O., Alexa M.: Laplacian mesh optimization. Proceedings of ACM GRAPHITE (2006)
[Pey04] Peyre G.: Toolbox Graph Mathworks (2004)
[R02] Raida Z. : Modeling EM structures in the neural network toolbox of MATLAB, 2002
[Sor05] Sorkine O.: Laplacian Mesh Processing. Proceedings of EUROGRAPHICS 2005, STAR Volume.
[TB97] Trefethen L., Bau D. : Numerical Linear Algebra. SIAM (1997)
[TK09] Theodoridis S., Koutroumbas K. : Pattern Recognition. Elsevier Inc. 2009
[WAE14] Wahed M., Abdallah M., Elkomy M. : Non Superfluous Time Solutions of the Laplacian Framework. ijcsns 2014

Mohamed Elkomy received his B.Sc. in Electrical Engineering from Ain Shams University in 2001. After working as a teaching Assistant (from 2006 to 2007) he has received his Master in Computing from the American university in Cairo in 2009. His research interest includes Computational Geometry, Geometric Processing, and Electrical CAD design.