Implementation of Ultrasound Elastography: A Dynamic Programming Approach

Hassan Rivaz*, Emad Boctor, Pezhman Foroughi, Richard Zellars, Gabor Fichtinger, and Gregory Hager

Cesar A. P. Cipelli ID: 40070751, Jamil Reza Chowdhury ID: 40101116

***Abstract*—**We tried to accomplish a 2-D strain imaging technique based on calculating a cost function using dynamic programming (DP) method. For this, we first started by the 1-D DP approach. We used the sets of pre-compression and post-compression images. We considered the tissue deformation to be in the axial direction only for a single dimension. We then calculated the cost function as described in original paper [1]. We finally calculated a displacement map of the post-compression image with respect to the pre-compression image. After that, we wrote a strain function to calculate the gradient of the displacement map. This generated the strain image. We could clearly see the tumor**.** Then, 2-D Dynamic Programming was done to improve the results.

***Index Terms*—axial displacement, lateral displacement, axial strain, lateral strain, dynamic programming.**

I. Introduction

IN this work, elastography was implemented using ultrasound images. This method utilizes elastic properties of the soft tissues and quasi-static forces to calculate displacement map and strain images. Displacement images or maps show the most probable location of a pixel on another image. For example, if the pixel in (417,140) has a displacement value of -50, the most probable location of the same pixel on the other image was inferred at (367,140). With the displacement, it is possible to calculate strain derivative in the *y-direction*. This was done using a vertical window in order to smooth the strain image.

First, it was collected real ultrasound images with an object of interests underneath it, a tumor for example. Second, these images were processed using the 1D Dynamic Programming approach explained in [1], where each pixel is compared with its neighbors on the same column (A-lines) from a minimum displacement, d_{min}, up to a max displacement, d_{max}. After, the algorithm is generalized to include lateral displacements, which results in an improvement of the output images with stimulating results.

II. Review

This paper is an attempted reproduction of the method exposed in [1]. Because of that, most of the references here are the same as presented on the original paper.

It is important for the reader to realize that the method implemented here is not the most up to date and new implementations have already been developed for the same problem, such as [2] and [3]. However, it is a good starting point to understand the field. An in-detail review of the commercial Elastography state of the art methods in Ultrasound Images can be found in [4].

III. Theory

Dynamic programming is an approach which refers to the simplification of a complex problem by dividing it into simpler sub-problems in a repetitive manner. It is an optimization algorithm that optimizes a cost function in a non-greedy manner. This is possible because the only greedy decision is taken at the last step and all other steps are connected in a "chain".

Elastography is a process of using tissues stiffness to infer some diagnostic. For example, it can be applied to tumor detection, because tumors are stiffer than the tissue around it. To apply this method, first, it is necessary to obtain an image pre-compression. Then, take a second image using an ultrasound probe applying a static (or quasi-static) compression. Because of the compression, the different tissues are going to deform axially and laterally with different strains. As the tumor is stiffer, it will deform less than the soft healthy tissue. Consequently, its displacement will vary less, and its strain will be lower than background tissue, i.e. the tissue around it.

Combining these two concepts, a short explanation of the method demonstrated in [1] will be provided. The algorithm analyses one pixel at a time in an A-line (column in an ultrasound image) and calculates a cost function for each allowed axial displacement applied to this pixel. So, if the allowed axial displacement is between -100 and 0 and pixel at position 366 in A-line 2 of image 1 (pre-compression) is being analyzed, this pixel will be compared with pixels 366+0=366 up to 366-100=266 in A-line 2 on the post-compression image and the corresponding cost function will be calculated. Equation (1) and (2) are a reproduction of (4) and (5) in [1].

$$C(i,d_i) = \min_{d_{i-1}}\{C(i-1,d_{i-1}) + \omega S(d_i,d_{i-1})\} + \Delta(i,d_i) \tag{1}$$

$$M(i,d_i) = \operatorname*{argmin}_{d_{i-1}}\{C(i-1,d_{i-1}) + \omega S(d_i,d_{i-1})\} \tag{2}$$

where i is the current row, d_i is displacement of current row, ω regularization weight, S is Smoothness of displacements and Δ is the difference between two signals, all defined in [1]. Equation (1) means that for each current row and displacement being evaluated, all the displacement values of the previous row (d_{i-1}) must be attempted for the calculation, and the minimum should be selected. After that, the delta function is calculated, and the cost function is updated for matrix C. The matching d_{i-1} that

resulted in the minimum value in (1), is stored on matrix M, as shown in (2).

The distinctiveness of this method is that it is not greedy. When it needs to select the optimal displacements to generate the displacement map it does not take the minimum displacement for each row but the minimum for the last row and then traces back the minimum displacement using the memoization matrix *M*. That can be seen in [1, eq. (6)].

For one dimension, displacement is always calculated for axial direction. In the case of also considering lateral displacement, referred as the two dimensions (2D) approach, the method includes lateral displacements and generalizes the concepts applied to the 1D version, as demonstrated in [1].

The algorithm makes use of the restricted search range for the axial displacement of the previous row as stated in [1], for the 1D as for the 2D case.

IV. METHODOLOGY

The algorithm was implemented using MATLAB R2018b. Hyperparameters are regularization weight, fixed at 0.15, maximal and minimum displacement in the axial direction and, in 2D, lateral direction. After the displacement map was generated using the method described in [1], the strain image was calculated using a custom function. This function uses a window determined by the user, in this paper 43 and 73, to approximate a linear polynomial and retrieve its slope. Basically, it was attempted to get a column-wise smoothed derivative of the displacement since acquiring the derivative of the displacement directly (as of the definition of strain) was too noisy.

With the strain image calculated, in order to compare the results achieved by this work, some metrics must be defined. First, SNR (Signal to Noise Ratio): this is measured by choosing a window on the strain image and calculating (3) where *s* is the average and σ is the standard deviation from the pixels of the chosen window. Also, CNR (Contrast to Noise Ratio) is used to compare the target (tumor) and background values using (4), where $\bar{s_b}$ and $\bar{s_t}$ are the spatial average strain value for the background and target, and, σ_b^2 and σ_b^2 are the spatial variances for background and target, respectively. Another measurement utilized was the strain ratio, defined in (5). The lesser, the better. The results were further compared using strain plots, where a row and a column crossing through the target are selected so it is easily seen the different strain levels on background and target.

$$\text{SNR} = \text{s}/\sigma \tag{3}$$

$$\text{CNR} = \sqrt{\frac{2(\bar{s_b}-\bar{s_t})^2}{\sigma_b^2+\sigma_b^2}} \tag{4}$$

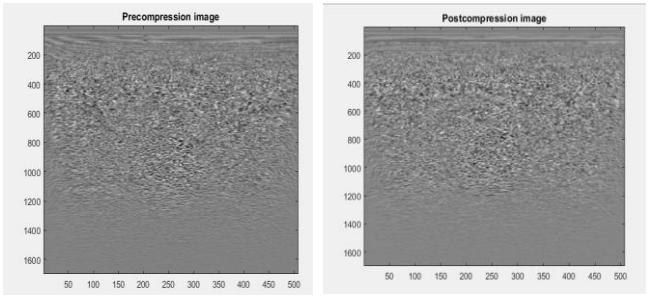
$$\text{Strain Ratio (SR)} = \bar{s_t}/\bar{s_b} \tag{5}$$

The images used on this work were the same used on [1] and can be accessed at [5].

V. RESULTS AND DISCUSSION

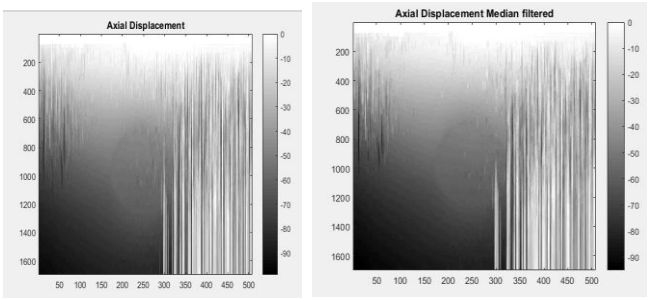
A. One Dimension (1D)

The algorithm in one dimension was run for a couple of sample ultrasound images shown in Fig. 1. The displacement map created by the code and a median filtered version is shown in Fig. 2. Strain image was calculated based on the algorithm described before and it is shown in Fig. 3. The strain was calculated with a window size of 43 and 73. The performance results are shown below on Table I and the background and target window specifications are documented as well. Strain plots are shown in Fig. 4.



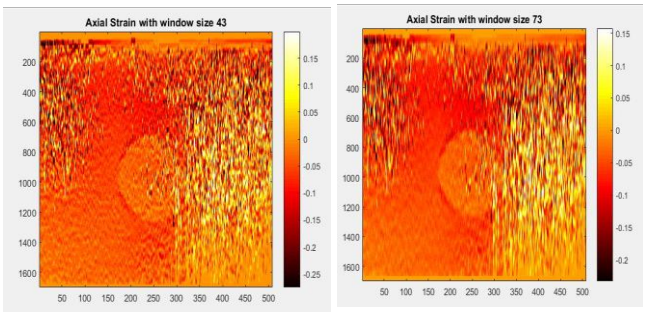
(a) (b)

Fig. 1 Input ultrasound image showing (a) Pre-compression image. (b)Post-compression image



(a) (b)

Fig. 2 (a) Displacement map generating by 1D method. (b) Same image median filtered



(a) (b)

Fig. 3 Strain images calculated with (a)window size of 43 (b)window size of 73

TABLE I
COORDINATES OF CHOSEN WINDOWS

	(initial row, initial column)	(final row, final column)
Background	(565,151)	(658,193)
Target	(850,255)	(100,310)
Signal to Noise ratio (SNR)	CNR	Strain ratio (SR)
13.2310	5.8486	0.4735

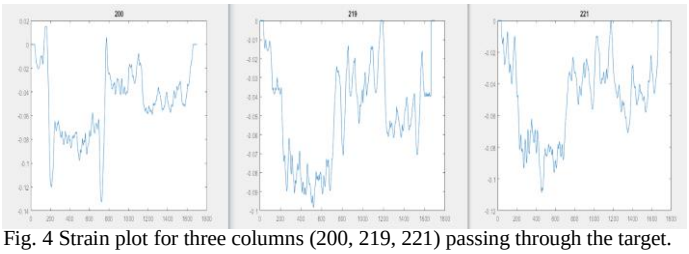


Fig. 4 Strain plot for three columns (200, 219, 221) passing through the target. Strain image calculated with the window size of 73

Firstly, displacement map shows a favorable image with a stiffer darker tumor on the center of the image. However, the image was not clear especially on the right portion, showing noisy stripes. This is believed to happen because of lateral displacement that was applied during the image acquisition. Consequently, no axial displacement could describe correctly the shift between the two images.

Secondly, the median filter effectively reduced noise, however it added some unwanted blurriness on the image.

Thirdly, the calculated strain images outputted as expected, showing a tumor in the middle of the image. However, it did not show precise information neither on the image right and left ends due to the lack of a reliable displacement on these regions.

SNR calculated for a window with relative smoothness on background, reached almost 9. CNR is consistent with the frequency distribution shown in Fig. 3(d) of [1]. SR shows that background strain is twice as big as target strain.

Lastly, strain plot from columns 200, 219 and 221 shows that three different regions can be specified. Left and right-most regions are backgrounds and middle region is the location of the tumor. The absolute values of the center region are closer to zero, indicating a low rate of change in displacement, i.e. low strain. Strain plot for rows was not shown due to the noisy stripes on the right portion of the image making hard to spot 3 distinct regions.

B. Two Dimensions(2D)

The algorithm in two dimensions was run for the same sample ultrasound images shown in Figure 1. The displacement map created by the code and a median filtered version are shown in Fig. 5. Strain image was calculated based on the algorithm previously described and it is shown in Fig. 6. The performance results are shown below in Table II and the background and target window specifications are documented as well. Strain plots are shown in Fig. 7 and 8.

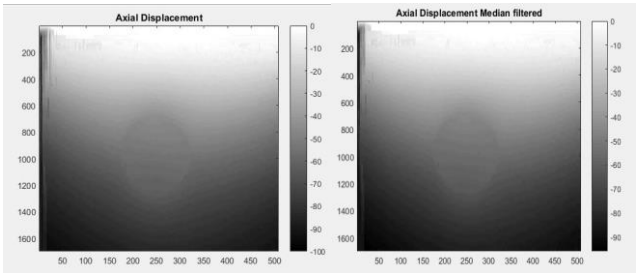


Figure 5. Axial Displacement Map generated by the 2D method (left) and the same image median filtered (right)

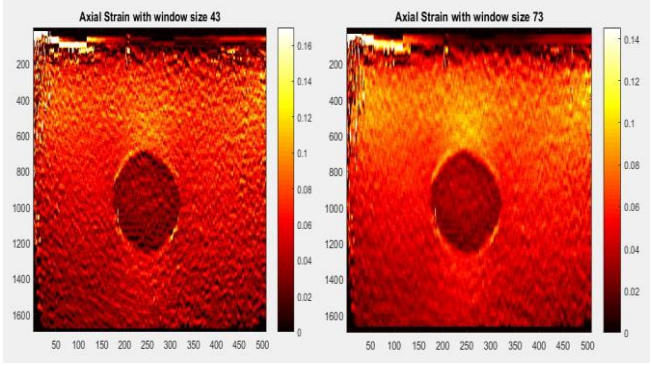


Figure 6. Strain images calculated with window size 43 and 73 respectively

TABLE II
COORDINATES OF CHOSEN WINDOWS AND SNR, CNR, SR VALUES

	(initial row, initial column)	(final row, final column)
Background	(400,56)	(464,68)
Target	(853,185)	(928,215)
Signal to Noise ratio (SNR)	CNR	Strain ratio (SR)
16.3346	9.0793	0.4032

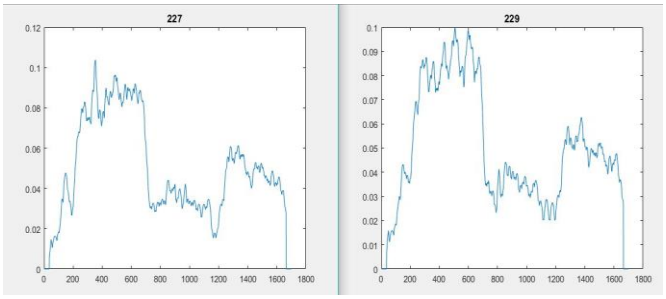


Figure 7. Strain plot for two columns (227, 229) passing through the target. Strain image calculated with the window size of 73.

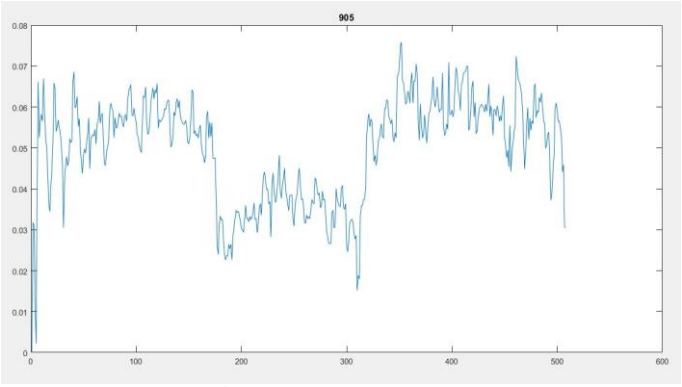


Figure 8. Strain plot for row 905 passing through the target. Strain image calculated with a window size of 73.

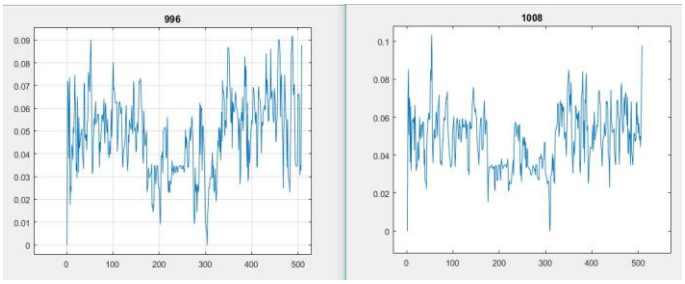


Figure 9. Strain plot for row 996 and 1008 passing through the target. Strain image calculated with a window size of 43.

Displacement map is improved, when compared to Fig. 2. It is possible to easily spot the tumor due to higher contrast rate, confirmed by the higher CNR. The noisy stripes are not present anymore. This output corroborates that there were lateral displacements of the right and left portion of the images. Image is so clear that median filtered has not proved useful in this case.

Strain images outline a much better picture this time, exposing exactly the tumor location and boundaries. Still, there was still a little bit of noise on the background, which does not obscure the target itself. Moreover, the absolute values for the target are on a lesser range when compared to the noisier version on Fig. 3.

For the performance metrics, improvements were made on all of them. SNR jumped from 13 to 16, CNR from around 6 to 9, which is also consistent with Fig. 3(d) shown in [1]. Finally, SR went from 0.47 to 0.40 which means the background strain is 2.5 greater than target strain, making the result even stronger.

Strain plots for columns now show a noticeable difference on three distinct regions that separate background and target. It can also be seen from the background regions, left and right sides of each plot, that strain is almost linearly decaying, which is expected since tissues obey Hooke's law. On strain plot for row, the three regions are still apparent exposing the algorithm effectiveness. Background regions of the row plot do not vary as the columns' do because it is a transversal view, so it is not expected to decay.

VI. Conclusion

From the results, it can be concluded that the implementation was successfully done using the Dynamic Programming approach described in [1].

The 1D case generated noisy results, since it was only considered the axial direction. On the other hand, the 2D resulted in better outcomes because of its sophistication, considering not only axial but also lateral displacement.

Performance metrics also showed that in terms of CNR, SNR and SR the 2D approach outperformed 1D.

For future work, the team would recommend coding in C or an equivalent lower level programming language in order to speed up the runtime.

It is also recommended that after understanding the dynamic programming approach, readers further research papers [2] and [3], since these implement newer approaches to the same problem.

Source code for this work is available at: https://github.com/cesarau41/elastography-dynamic-programming.

References

[1] Rivaz, H., Boctor, E., Foroughi, P., Zellars, R., Fichtinger, G., Hager, G., Ultrasound Elastography: a Dynamic Programming Approach, IEEE Trans. Medical Imaging, Oct. 2008, vol. 27 pp 1373-1377

[2] H. Rivaz, Boctor, E., Choti, M., Hager, G., Real-Time Regularized Ultrasound Elastography, IEEE Trans. Medical Imaging, April 2011, vol. 30 pp 928-945

[3] H. S. Hashemi and H. Rivaz, "Global Time-Delay Estimation in Ultrasound Elastography," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 64, no. 10, pp. 1625–1636, 2017.

[4] "EFSUMB Guidelines and Recommendations on the Clinical Use of Liver Ultrasound Elastography, update 2017 (long and short version)," *Ultraschall in der Medizin - European Journal of Ultrasound*, vol. 38, no. 03, pp. 327–329, 2017.

[5] *users.encs.concordia.ca*. [Online] Available: https://users.encs.concordia.ca/~hrivaz/Ultrasound_Elastography/.