Image Contour Extraction

Alexandros-Panagiotis Oikonomou

December 6, 2012

1 Introduction

For the assignement of the Intelligent Algorithms course I was asked to implement an Image Contour Extractor algorithm. With the use of MATLAB the principle of Dynamic Programming was adopted for the implementation. Throughout this report the design of the algorithm is documented along with findings made by testing the algorithm.

2 Program Design

This section will describe the design used in order to implement the algorithm.

2.1 Search Space

The design to extract the search space has been kept into a fairly simple level. There exists a function search_space which takes as arguments the image matrix returned by the imread command and the contour initialisations. It returns a 3D Matrix of type (rows(image),M,3) called points. M is the number of rows the user wishes to have inside the search space. In the first index of the third dimension (i,j,1) the x value of the image is kept, in the second index (i,j,2) the y value of the image is stored and in the third index (i,j,3) the intensity of the of (x,y) point is stored. That way each (i,j) point in the points matrix correspons to an (x,y) value of the image. Before the matrix is returned the values of the intensities are negated.

2.2 Dynamic Programming

In order to apply the algorithm supplied in the assignement there exists a function get_energies which takes as arguments the image matrix, the two initialisation contours and the lambda that the user wishes to choose. The first command is to call the search space function mentioned above. As output parameters it returns an energy matrix, a position matrix and the matrix returned by the search space called intensity. The intensity matrix is returned in order to be used for backtracking. The energy matrix is a 3D matrix of the form (rows(intensity),columns(intensity),rows(intensity)). The first two terms are there to correspond to the same point as in the intensity matrix. For all the points in the previous column, depending on the point the algorithm is currenlty at, the minimum energy of all the points in the previous column is stored in the last term of the energy matrix.

$$E_i\left(v_{i-1}, v_i, v_{i+1}\right) = \lambda \frac{\left|v_{i+1} - 2\,v_i + v_{i-1}\right|^2}{\left|v_{i+1} - v_{i-1}\right|^2} + (1-\lambda)\, I\left(v_i\right), \quad (1)$$

The reason it is a vector is that the above action is done for all the points of the next column of the point the algorithm is currently at. The energy is computed according to (1). The position matrix has the same first two terms, but the difference is in the third term, where the row of the minimum energy was found is stored, instead of the value itself. The result of computing energies of all these points is that a quadruple for-loop is used

2.3 Backtracking

The backtracking is completed in a function called get_contour which returns the points of the contourṪhe first calculations of this function locates the minimum energy along with its row and depth index in all the vectors of the next to last column. The reason this column is picked is that in the last column the computation is not completed in its entirety, due to the fact that there is no next column to do the computation correctly see (1). The algorithm now jumps to the node with the row index returned and the depth index in the position matrix and reads the value. Then the optimal path is chosen, by going to the next column by selecting the row, from the value returned before. The depth is found from the row index of that value.

2.4 Negating Images

It is also noteworthy to mention that the intensities of the points are negated before passed on to the produce_energies function. The reason for that is that the algorithm searches and stores the minimum energy. That means that lower intensities are desired. The problem with the returned image, when applying the imread function, is that pixels which seem to be blacker have lower intensities and pixels which seem to be whiter seem to have higher intensities. By negating the intensities, this property is reversed and hence the algorithm searches for pixels wich seem whiter. Negating the image is done by the function negating_intensities.

3 Algorithm Testing

This section will report the findings of applying the algorithm in different situations.

3.1 Initial Results

The initial test of the algorithm was to be done in a picture supplied by the assignement. The picture was of a X-ray of a tongue. The testing configuration asked was to be with λ=0.5. However M was bot specified so I chose M=100. The result contour displayed all but one points of the contour to lie perfectly on the outline of the tongue(5.1 Appendix).

3.2 Scalability with changes in M

3.3 Robustness with changes in Lambda

Changing λ seemed to make the contour behave differently on certain occasions. It was observed that for 0.1 < λ <0.7 the contour did not seem to change significantly. It remained the same as it was for λ=0.5, with having most of the points lying in the tongue outline, but having one point in a completely different point. The picture resembled the one in 5.1(Appendix). On the contrary for 0.8 < λ <0.9 it was observed that this point would now lie in the tongue, adjacent with the rest of the contour points. This made the contour more precise and accurate(5.2.1 Appendix). Finally, a λ'=0.00001 was tested in order to identify any irregularities that may be caused for small values of λ's. Again most of the points lied on the tongue, but this time in addition to the one irregular point that existed in previous λ's there were some additional irregular points in the initial x's(5.2.2 Appendix). This can be attributed to the fact that with low λ's emphasis is given in the intensity of the points and the intensity is found high at these points.

3.4 Different contour initialisations

To identify the strength of the algorithm, different initialisations of contours were tested.Initially, the first change was to move the top and bottom contours relative to the y-axis. Initial contour with higher top contour was tested(5.3.1 Appendix) and it was verified that the final contour would still be correctly drawn. The reason for that was that the intensities of the added search space were more or less the same as before, which resulted into having a precise contour. Next a lower bottom initial contour was chosen(Appendix 5.3.2). This time it is apparent that the contour breaks badly, even though some points still lie in the tongue outline. The reason for that is that points above the bottom contour have lower intensities than those that lie on the tongue. That makes the backtracking algorithm to choose them instead of the one on the tongue. Finally

3.5 Other tested images

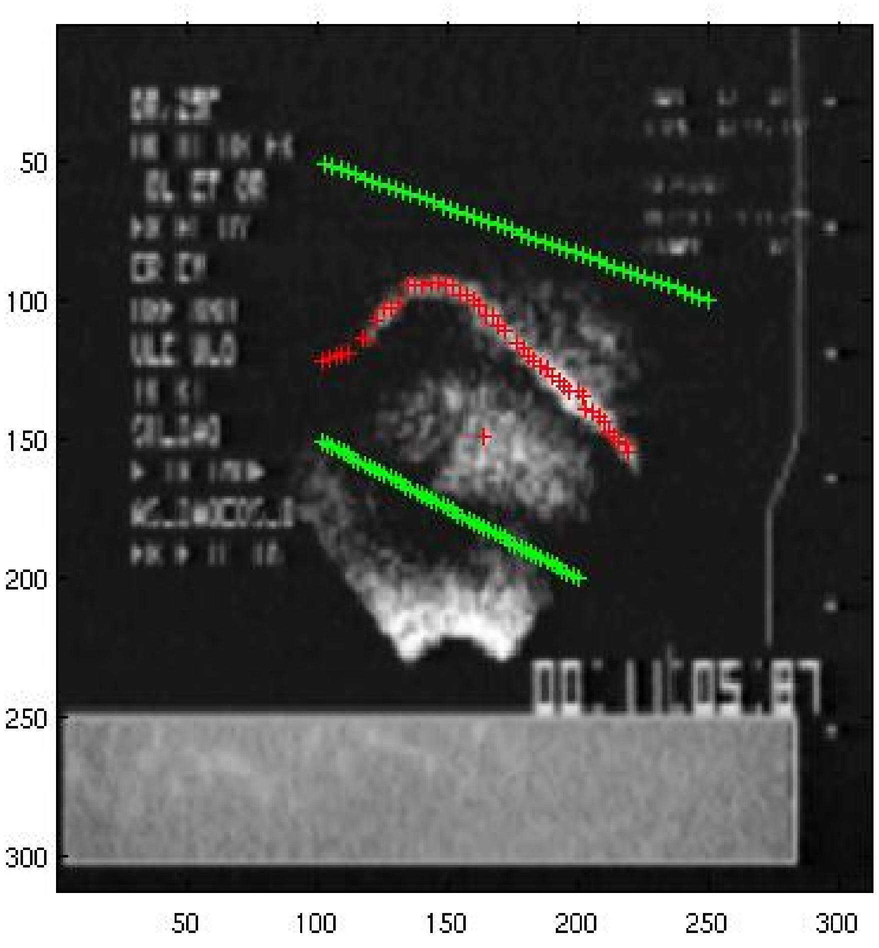
4 Conclusion

as

5 Appendix

5.1 Initial result

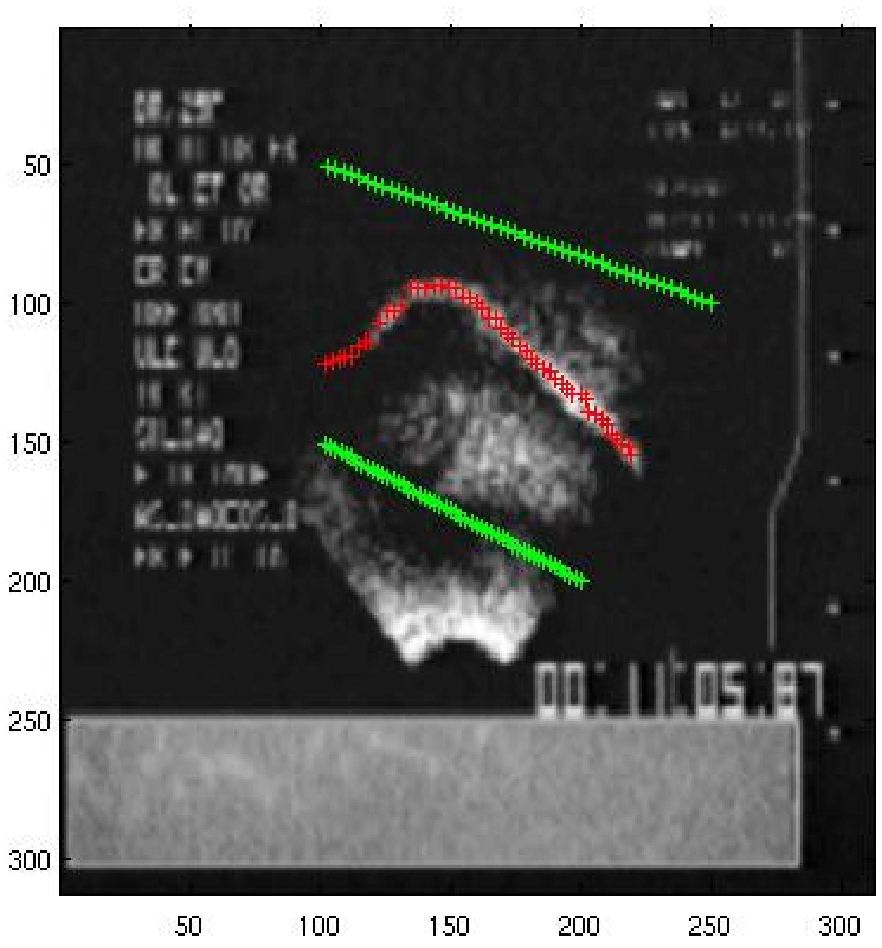
$\lambda = 0.5$ and M=100



5.2 Tweaking λ's

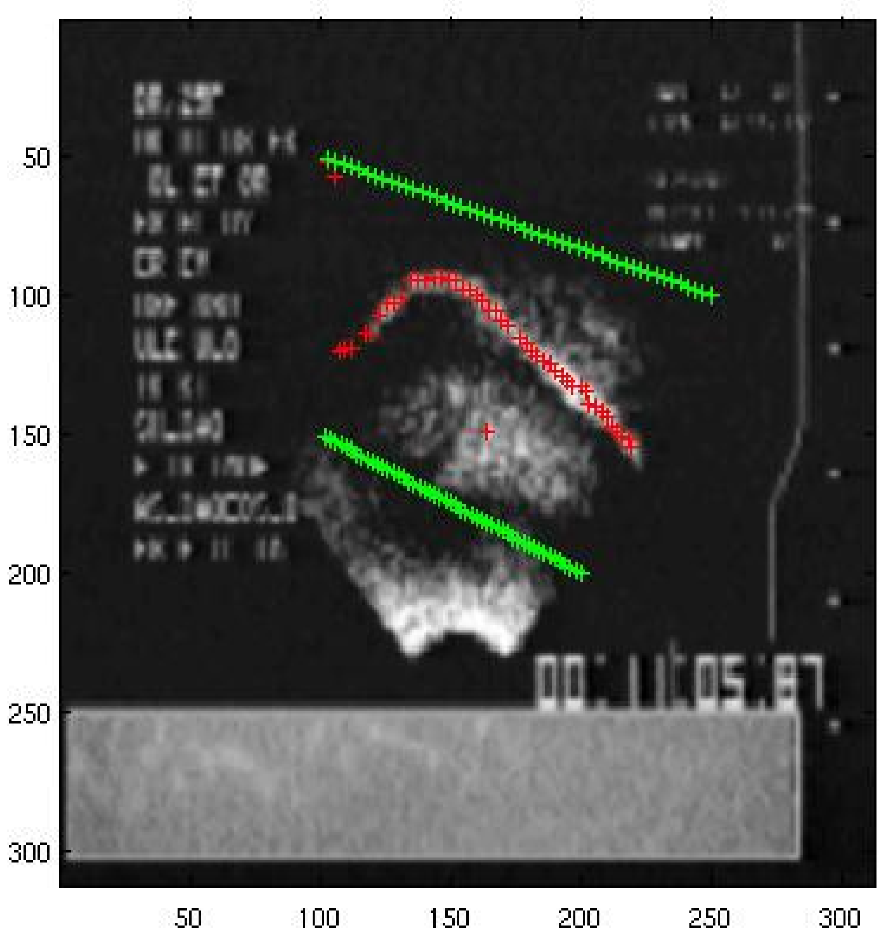
5.2.1 High λ's

$0.8 < \lambda < 0.9$ and $M=100$



5.2.2 Low λ's

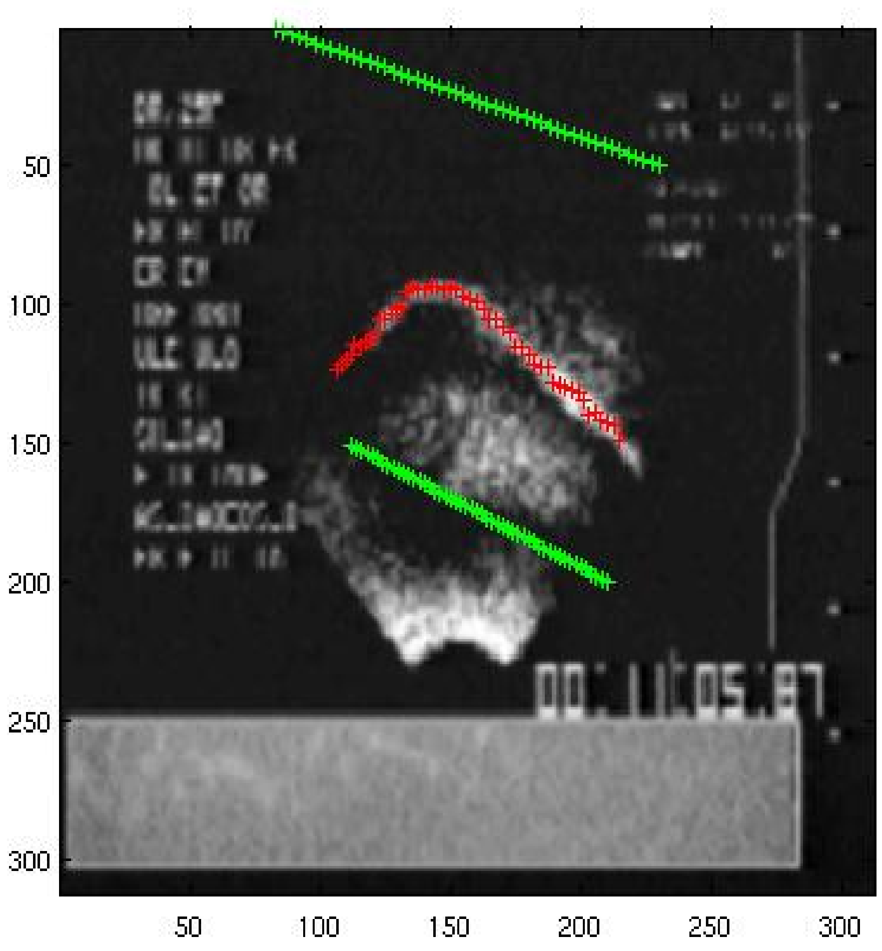
λ=0.00001 and M=100



5.3 Contour initialisations

5.3.1 Higher Top Initial Contour

The y of the top contour is decreased by 50. λ=0.8 and M=100



5.3.2 Lower Bottom Initial Contour

The y of the top contour is decreased by 50. λ=0.8 and M=100