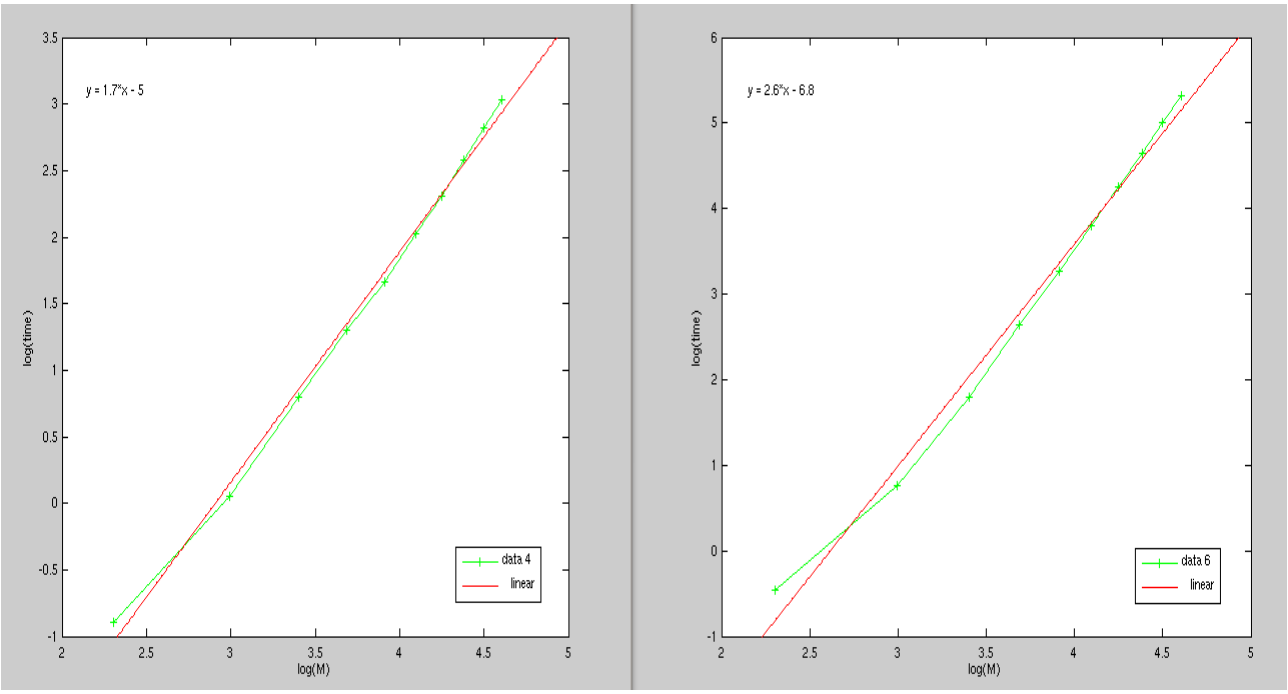
COMP3032 Image Contour Extraction Report
Marinos Mavrommatis (mm1g10)
06 December 2012

1 Scaling of the algorithm

As the dynamic programming paradigm suggests, the backtracking part of application has constant time complexity.
Arithmetic operations and comparisons of vectors (points) appear to be the most expensive, even though points are only two dimentional.
Theoretically the algorithm is of Theta(N*M^3) but since vector operations are highly optimised in Matlab compared to for-loops, eliminating a for-loop by vectorisation produces significant performance gains. In particular the log(time) against log(M) graph has a gradient of almost three in the case of the three for-loops implementation and almost two in the case of the two for-loop implementation. This implies that the practical complexity of the algorithm is reduced from Theta(N*M^3) to Theta(N*M^2) when removing a loop by vectorisation. The actual slopes are 1.7 and 2.6 respectively but this is justified by the fact that from M=30, the cost of unoptimised parts of the algorithm is amortised, whereas the approximate line is fitted for all values of M. All the time measurements were averaged over three runs.



2 Robustness w.r.t. lambda

From a minimum of lambda=0.01, the minimal contour chosen by the algorithm stays

within the outline of the tongue but doesn't appear as a straight line since different points within the outline have slightly different intensities. Up to a maximum of lambda=0.9 the result stays within the tongue outline almost entirely. The mentioned observations were based on M=50. [also put lambda=0.7 in pics]
Since picking pixels to construct the search space is done by averaging out the result of a division, areas with very few high-intensity (desired) pixels are not represented correctly in the serach space. That is, the high-intensity pixels might just be missed and thus not taken into acount when constructing the contour. To address this issue, a bigger M is selected. In particular, for M=100 the produced contour is much smoother for all values of lambda.
It is worth mentioning that there is a "black spot" withing the outline, that is big enough to be picked for both M=50 and M=100. Only with M=100 and a minimum lambda of 0.5 the resulting contour passes through the black area. For smaller values of lambda the continuity factor of the Energy function is not high enough and the contour skips the black area by "jumping" over or below it.

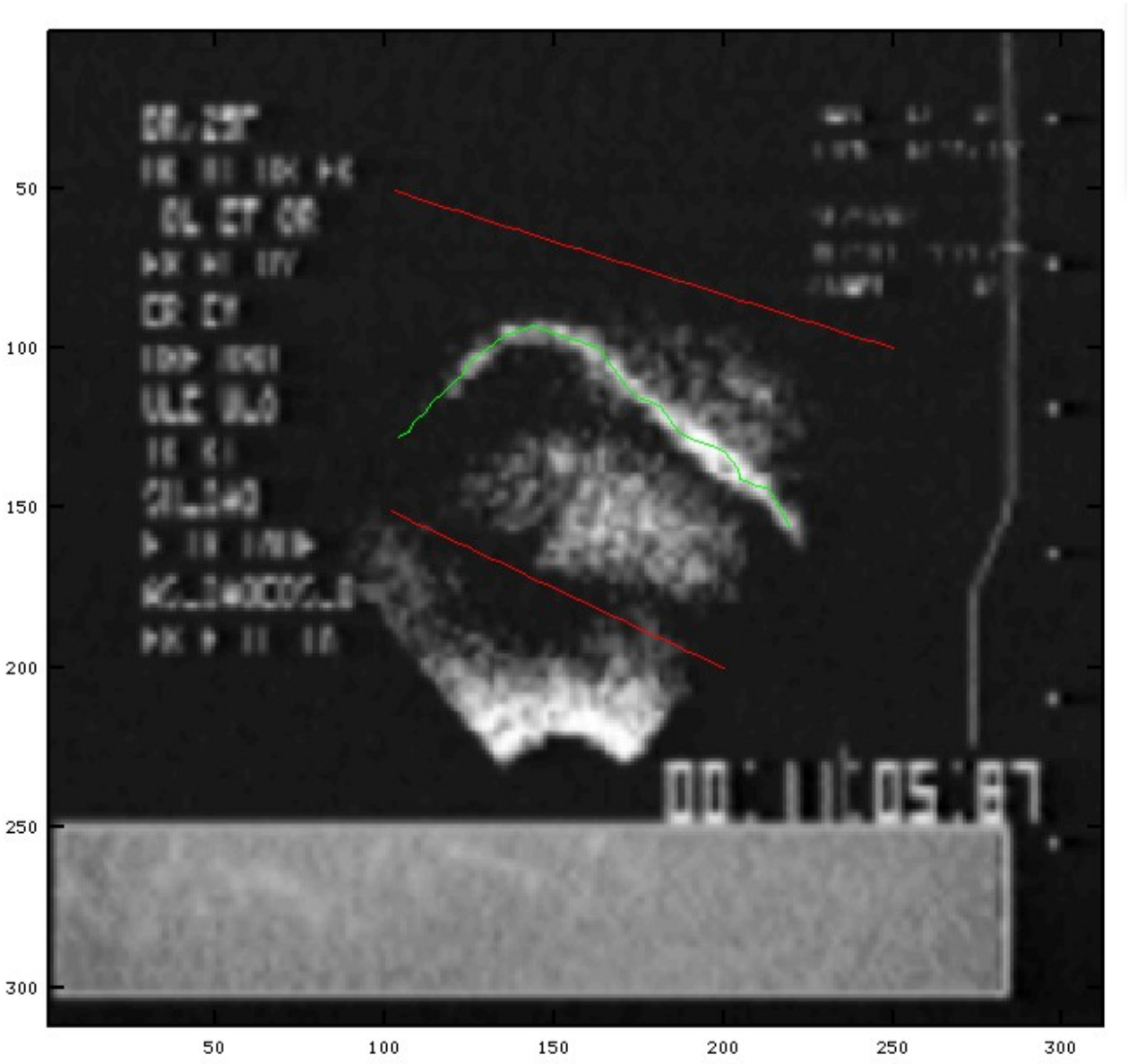


Illustration 1: M=50 lambda=0.5

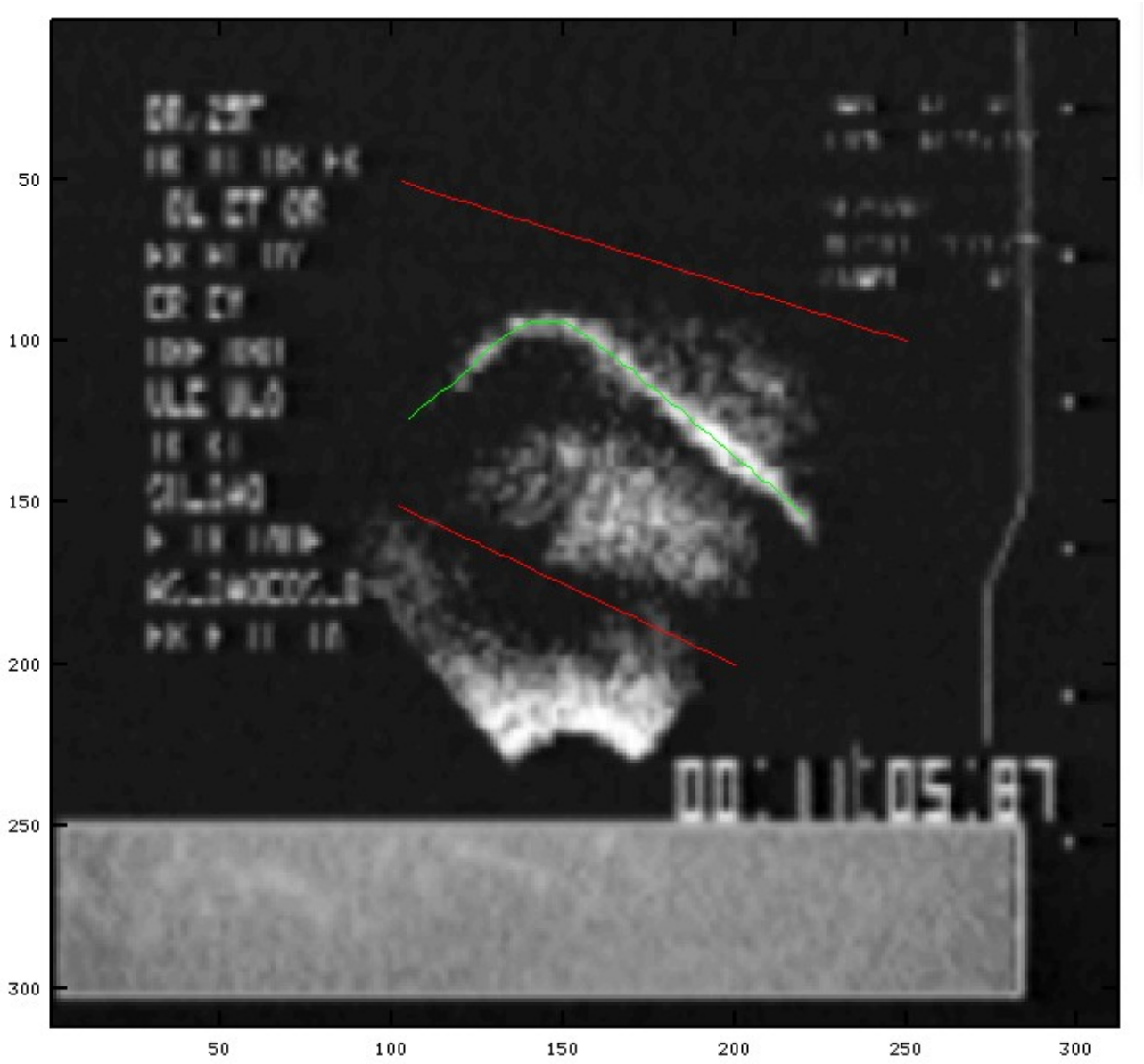
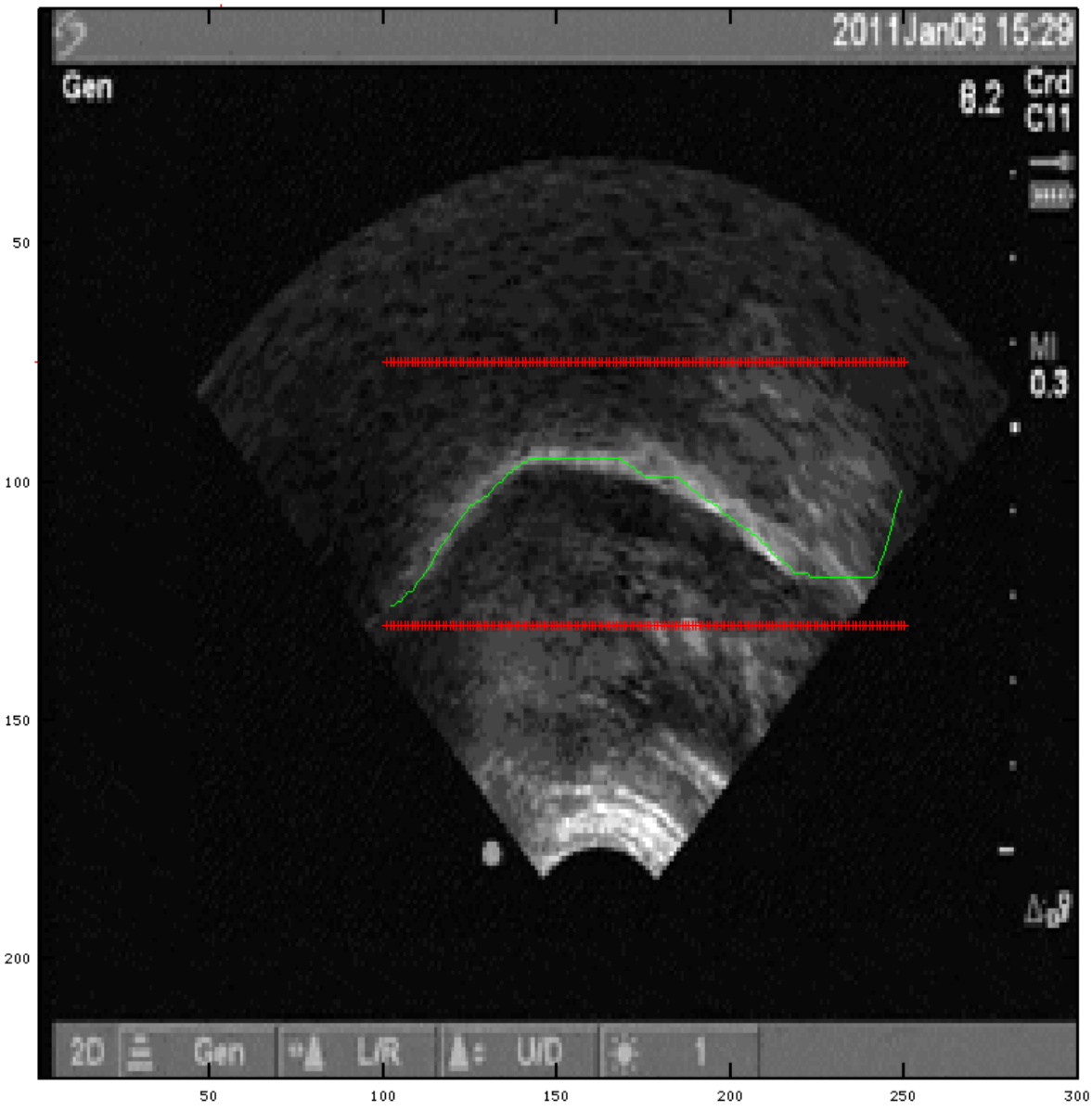
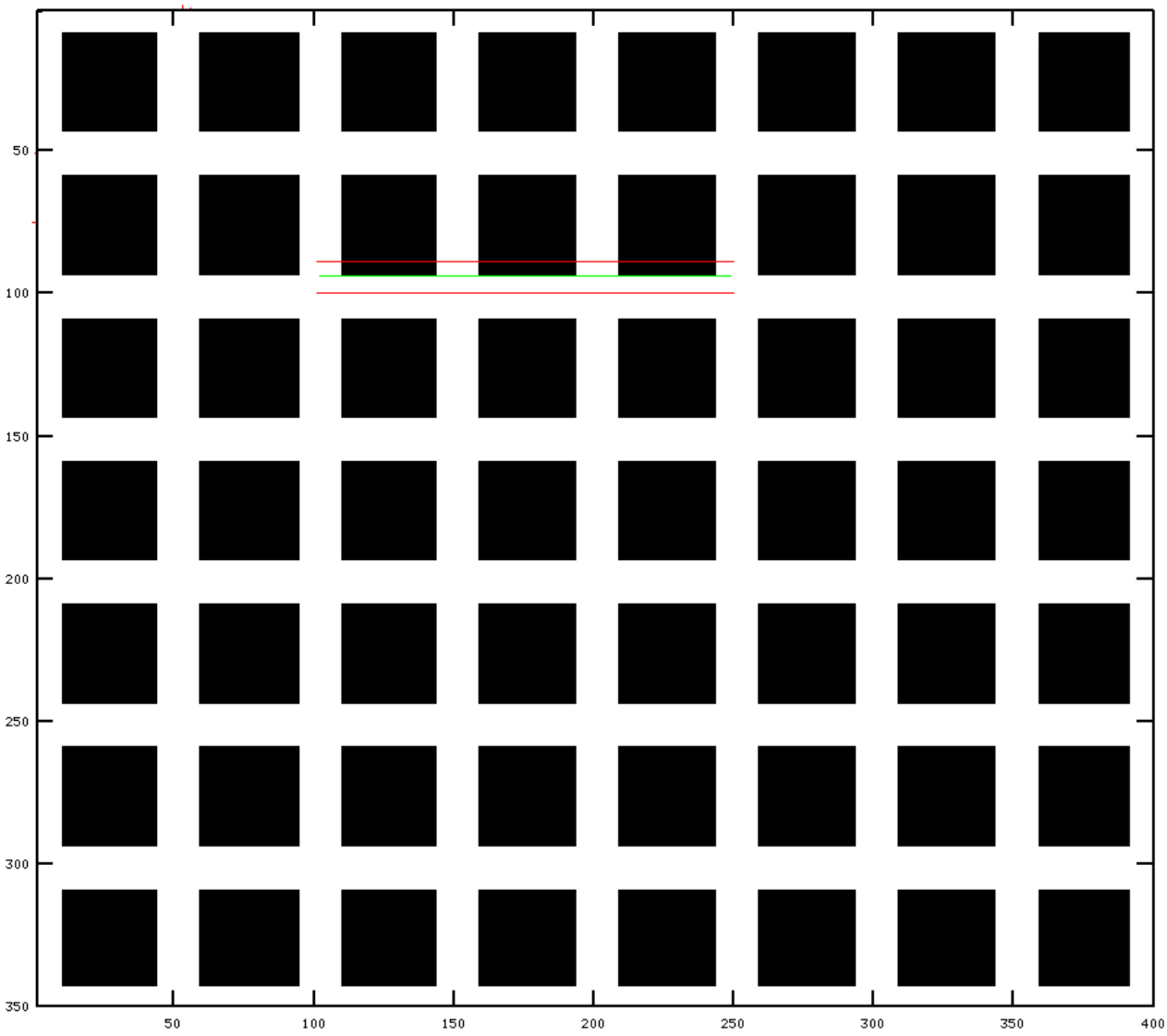


Illustration 2: M=100 lambda=0.5

3 Applying the algorithm to other images and contours

The previous sections describe and explain the behaviour of the algorithm when varying lambda and M, but only used examples from a single image. The algorithm behaves in an identical manner on different initial contours and different images, as shown below.


2011Jan06 15:29
Gen
8.2
Crd
C11
MI
0.3
2D
Gen
L/R
U/D
1
50
100
150
200
250
300


50
100
150
200
250
300
350
50
100
150
200
250
300
350
400