COMP3032: Intelligent Algorithms
Image Contour Extraction

Ashley J. Robinson

School of Electronics and Computer Science
University of Southampton

December 3, 2012

Abstract

Starting with developing a search space extraction method and then a basic dynamic programming algorithm it was possible to obtain the desired contour from the provided ultrasound image. Tuning the algorithm and extending it to minimise contour angle as well as image intensity provided a more natural contour. This algortihm was further optimised to acheive the same results in significantly less time. The approach was then applied to closed contours by defining circular extraction boundaries then running the extraction algorithm twice out of phase around the search space. Finally the closed contour algorithm was then tested on both artificial and natural images.

1 Search Space Extraction

As discussed in the brief (Gunn, 2012) the search space is extracted from the area between two contours. The data is stored in an $M \times N$ matrix where N is length of the search space contours and M can be chosen to suit resolution requirements. The search space doesn't have to be defined as a rectangle which means data can't be taken directly from overall image matrix.

The matrix is then formed by taking a direct line from the two corresponding contour points then dividing this line into M sections. Figure 1 shows how this done by considering the differences in the image matrix indexes then dividing these into M sections.

The search space extraction using the example contours and image provided is shown in figure 2. The extracted search space image is upside down but this really makes no difference and is just due to indexing choices during extraction. During the extraction process a mapping matrix is also created so that when an optimal contour has been found to fit the search space the coordinates can be mapped back the original image therefore making more sense when seen in context

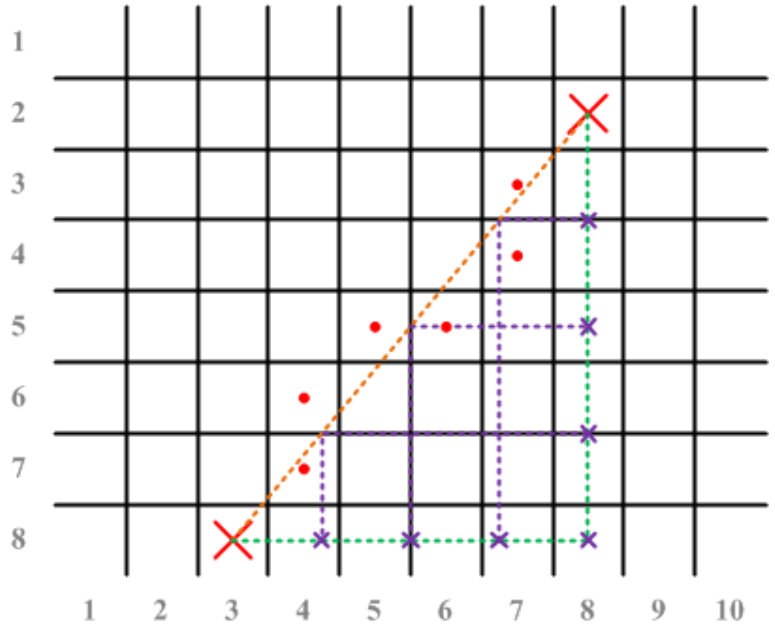

FIGURE 1: Creating a search space matrix.

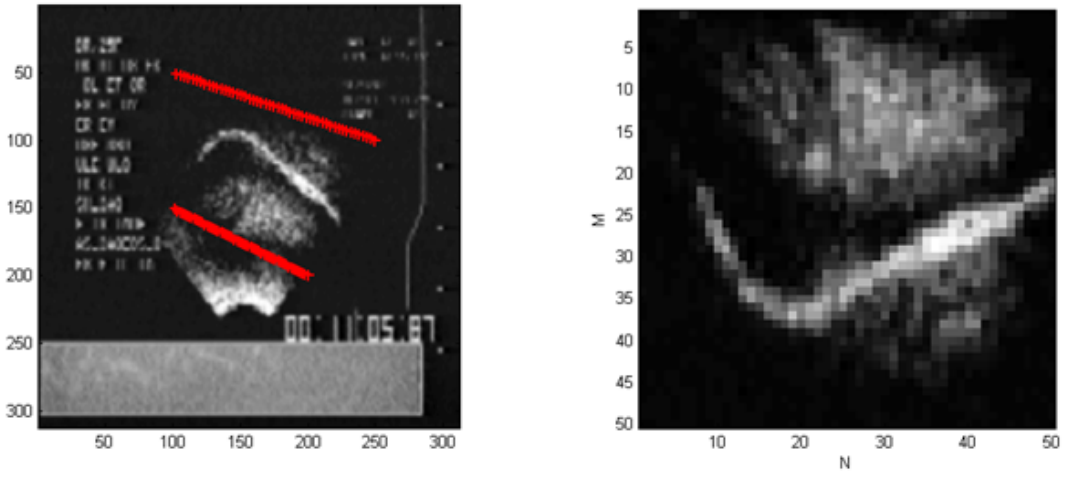

FIGURE 2: Extracting the search space.

2 Optimal Contour Search

2.1 Basic extraction

The goal is to get from one side of the search space to the other while expending minimal energy. Energy is expended by both moving between pixels and to low intensity pixels. This can be expressed using equation 1 where function I is the intensity of an element of the matrix and λ is weighting function that can bias the equation towards distance or intensity. This basic approach is capable of extracting the contour from the ultrasound image.

$$E(v_i) = \lambda(|index_{next} - index_{current}|) + (1 - \lambda)(1 - I(index_{current})) \tag{1}$$

2.2 Improved extraction

To favour natural contours the angle between point transitions must also be considered. To create an angle three possible matrix indexes are used so unlike the previous method the energy function now takes three inputs. The method used in Gunn and Nixon (1998) is to try an minimise the distance between the current and next matrix indexes but increase the distance between the last and next; therefore the angle tends towards π but also attempts to minimise the overall path; equation 2.

$$E(v_{i-1}, v_i, v_{i+1}) = \lambda \frac{|index_{next} - index_{current}|}{|index_{next} - index_{last}|} + (1 - \lambda)(1 - I(index_{current})) \tag{2}$$

Figure 3 is the best extraction and was found by tuning λ, section 2.3, for this particular image. The curves shown at the bottom and far left indicate a natural extraction. The surface plot in figure 4 is of the energy matrix used to fill the position matrix by walking along the gully. The flat surface at the far end would create issues if the algorithm was run in the opposite direction through the search space as initialisation would be an issue. The idea discussed in section 3 would improve initialisation.

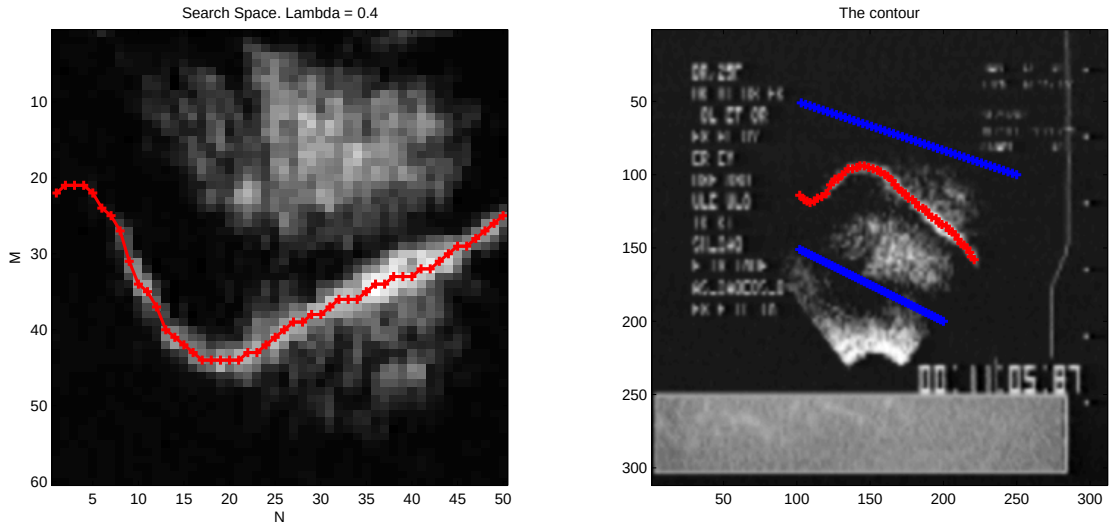

FIGURE 3: The search space contour extraction and the final image contour with boundaries (blue). M = 60, N = 50, $\lambda = 0.4$

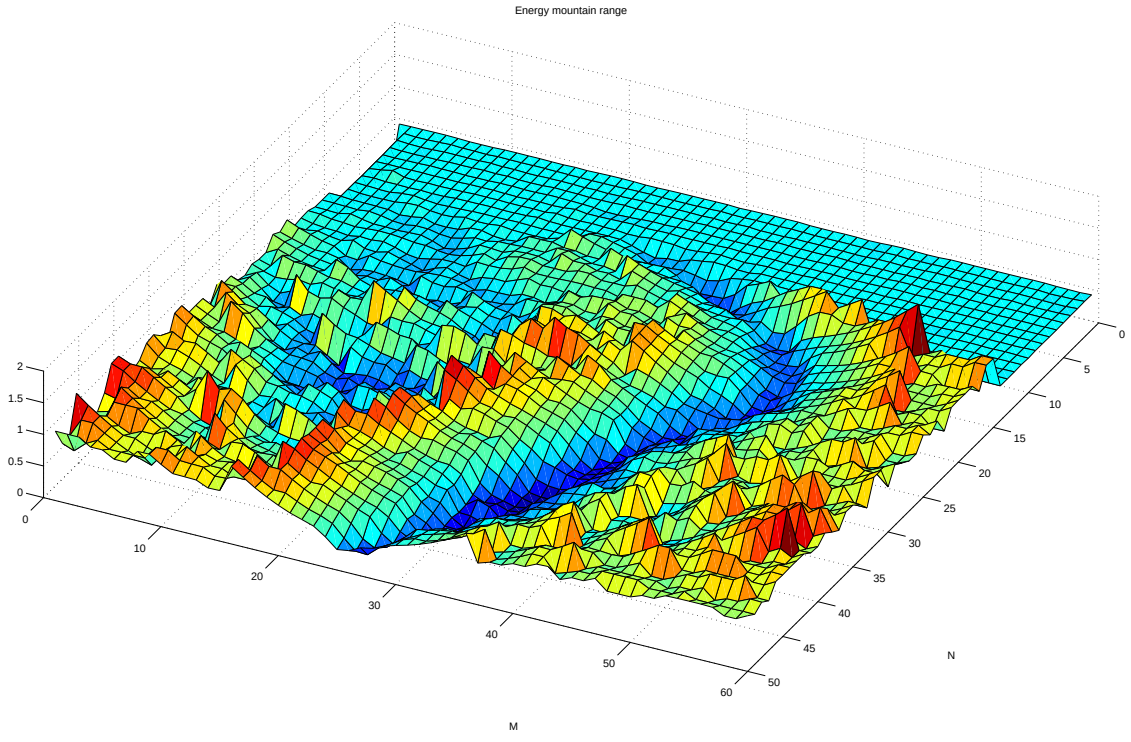

FIGURE 4: The energy mountain range of figure 3. M = 60, N = 50, $\lambda = 0.4$

2.3 Tuning λ

The weighting function λ is chosen to either favour the straight paths or high intensity paths. Figure 5 shows how varying the value of λ between 0 and 1 can effect the extraction. Acceptable contours are produced when lambda is between 0.1 and 0.9 but the best fitting contours are seen when lambda is between 0.1 and 0.4. Overall quite a robust function which will work for many different values but further testing would be required if this function was expected to work on other images. The naive approach of picking high intensity pixels only is show when λ is 0 and the extraction produced when λ is 1 is complete nonsense.

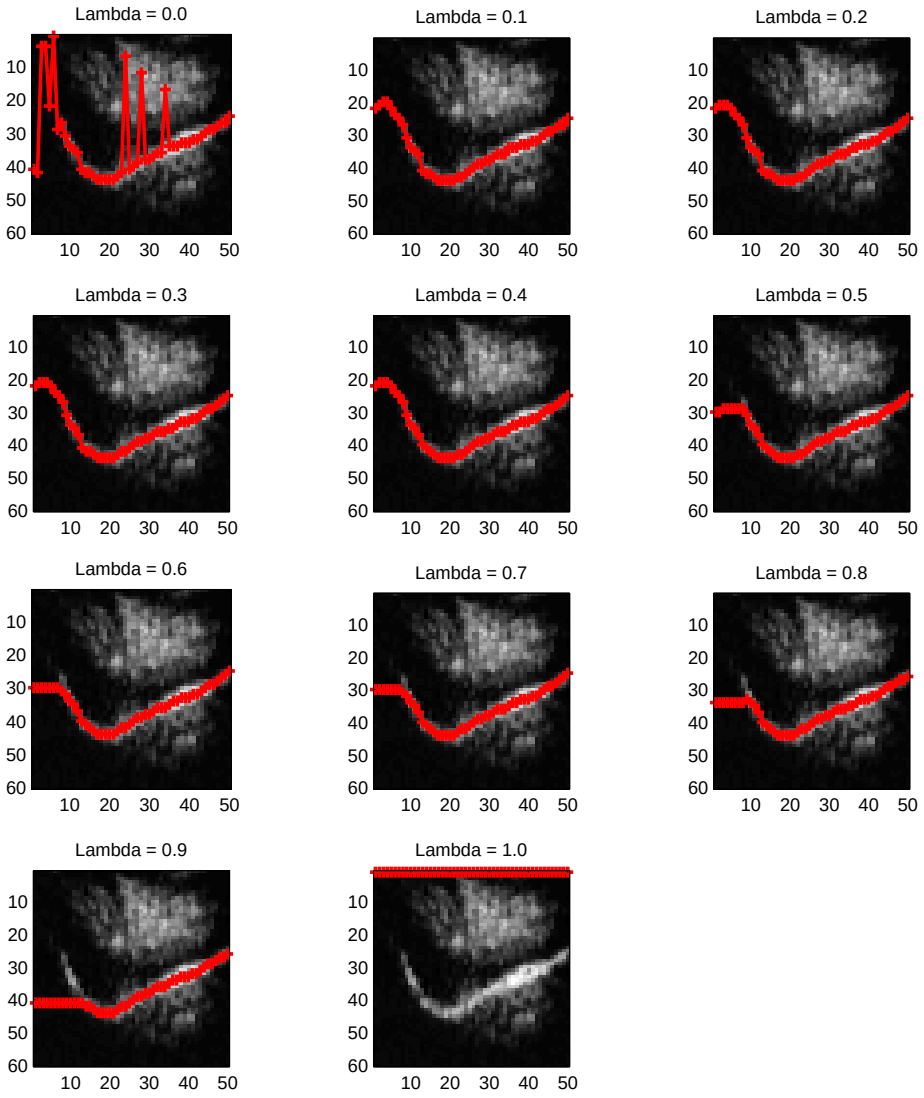

FIGURE 5: Tuning lambda

2.4 Efficiency and optimisation (Cormen et al., 2009)

The most time consuming part of the algorithm is to fill the position matrix which will later be backtracked to find the best path through the search space. This has to access every element of the search space matrix but also possible transitions to and from the current element and therefore contains nested loops (Appendix A). Equation 3 shows the asymptotic upper bound of the algorithm.

$$f(n) = O(n^4) \tag{3}$$

Accessing every possible transition is counter productive as the energy function applied in the nested loop kernel most probably won't yield better results for elements at great distances. Figure 7 shows how limiting the transition search no longer makes makes inner iterations dependant on n. The implementation of this approach is shown in appendix B where running time is now described as equation 4.

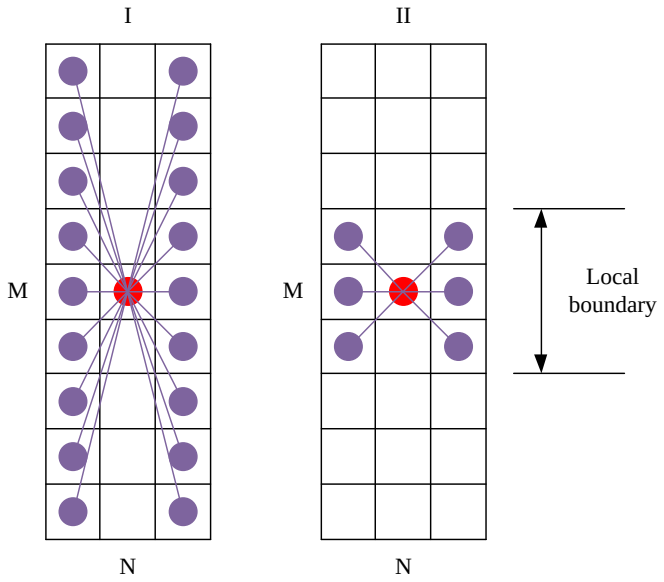

FIGURE 6: (I) Unbounded scan (II) Bounded scan

$$f(n) = O(n^2) \tag{4}$$

The boundary variable allows the algorithm to test at a given distance from its current M index. Testing for large transitions is pointless as the running time will tend back to equation 3 but for smaller distances running time is greatly reduced. Figure 7 shows how varying the distance effects execution time. Acceptable contours are extracted for boundaries greater than or equal to two.

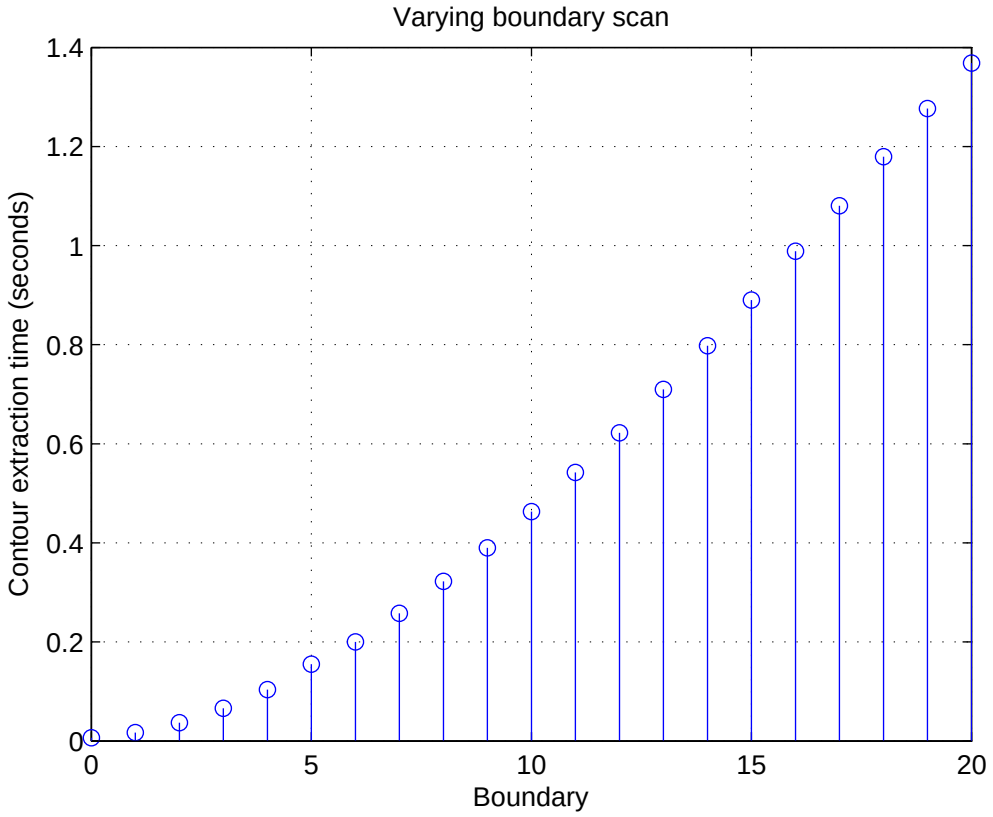

FIGURE 7: Graph. Varying the boundary scan range

Figure 7 is the algorithm executed using the code in appendix B where the boundary value is held in the variable *jump*. The code in in appendix A took 5.7 seconds to run.

3 Closed Contour Extraction (Gunn and Nixon, 1998)

The key is an effective extraction of the search space. Two circles are used as boundary contours where the difference in radius defines M and the circumference defines N. This obviously produces a much larger search space than previously considered and because the algorithm is run twice the computational overheads are much greater. The artificial image used in figure 8 shows how the extraction takes place. There are two search space contours, the first initialises on the highest intensity matrix element while the second uses the mid-point of the first contour. They both proceed in a contour clockwise direction around the image.

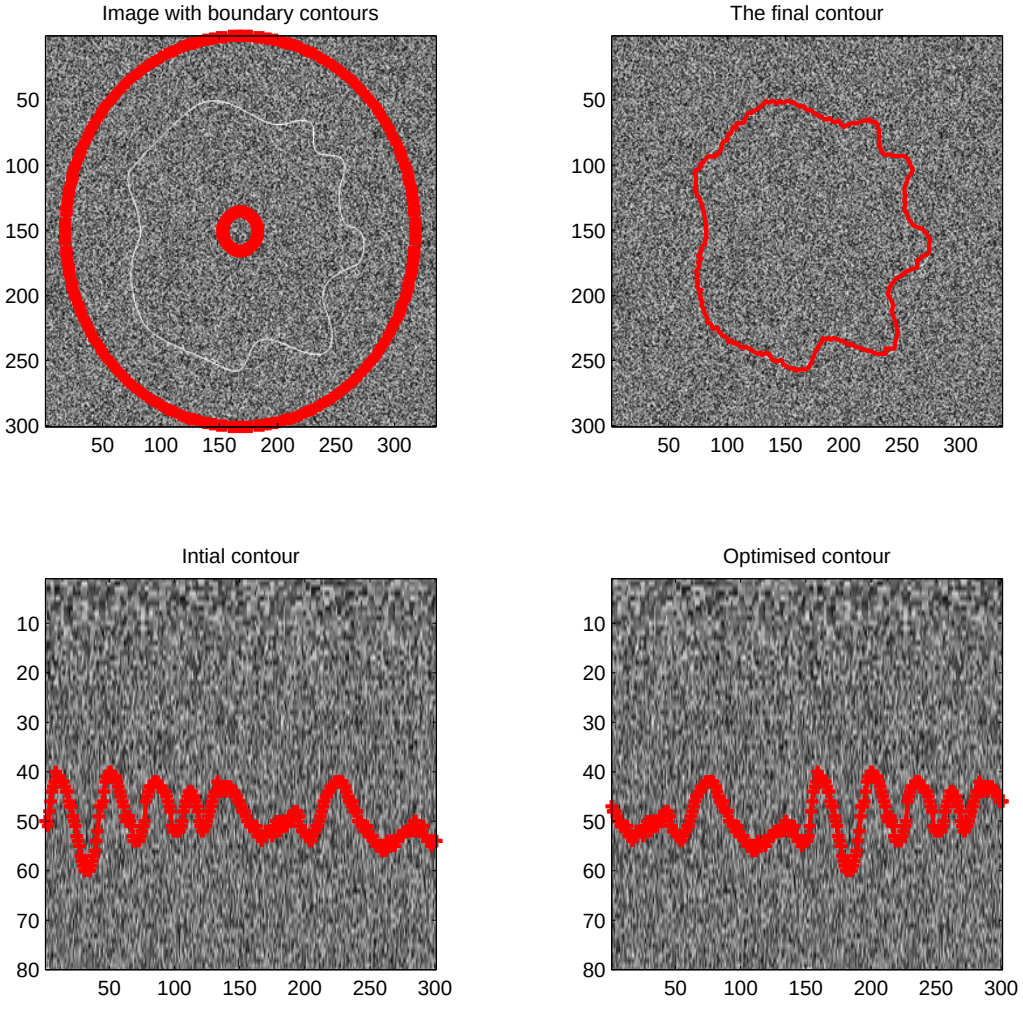

FIGURE 8: Extraction of a white line against a static background. M = 80, N = 300, $\lambda = 0.7$

Applying the algorithm on more natural images produces an interesting output. The pint glass in figure 9 has a metallic rim and therefore produces a high intensity closed contour because of the light incident on the surface. The contour is extracted successfully but areas of high reflection on the rim cause glares where the algorithm will take the path favouring the smaller contour; this is the inside of the rim because the distance is less in the search space matrix. Appendix C contains more test images. Some of which are easily extracted and others which cannot be extracted using this method.

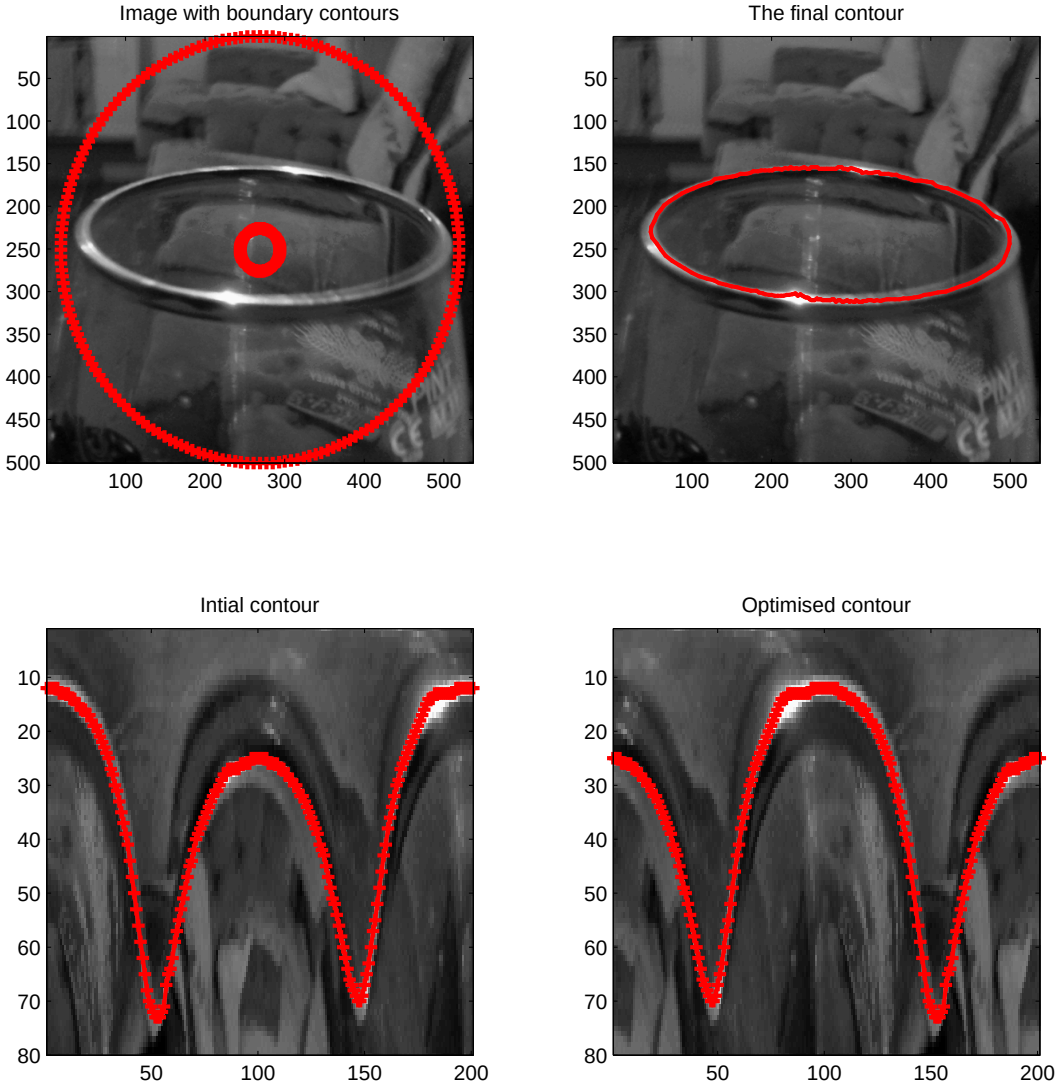

FIGURE 9: Extraction of the metallic rim of a pint glass. M = 80, N = 200, $\lambda = 0.7$

References

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. The MIT Press, third edition, 2009.

Steve R. Gunn. COMP3032 assignment: Image contour extraction, 2012. University of Southampton, ECS, COMP3032.

Steve R. Gunn and Mark S. Nixon. Active contours for head boundary extraction by global and local energy minimisation. *Int. Journal of Computer Vision*, 30(1):43–54, 1998.

Appendices

A Initial contour search

```
for i=2:1:N %Move along the contour
    for j=1:1:M %Move between the contours
       %It could be right first time?
       energy(j,i)= energy(1,i-1) + Energy(lambda,1,j,1,M,im_matrix(j,i));
       for v0=1:1:M  %Last
          for v2=1:1:M  %Next
          check = energy(v0,i-1) + Energy(lambda,v0,j,v2,M,im_matrix(j,i));
             if(check < energy(j,i))
                energy(j,i) = check;
                position(j,i) = v0;%Where you have been
             end
          end
       end
    end
end
```

B Optimised contour search

```
for i=2:1:N %Move along the contour
   %LOW
    for j=1:1:jump %Move between the contours, up until jump
       energy(j,i) = energy(1,i-1) + Energy(lambda,1,j,1,M,im_matrix(j,i));
       for v0=1:1:(j + jump)  %Last
          for v2=1:1:(j + jump)  %Possible
          check = energy(v0,i-1) + Energy(lambda,v0,j,v2,M,im_matrix(j,i));
              if(check < energy(j,i))
                energy(j,i) = check;
                position(j,i) = v0;
             end
          end
       end
    end
    %MID
    for j=(jump + 1):1:(M - jump) %Move between the contours, bound by jump
       energy(j,i) = energy(1,i-1) + Energy(lambda,1,j,1,M,im_matrix(j,i));
       for v0=(j - jump):1:(j + jump)  %Last
          for v2=(j - jump):1:(j + jump)  %Possible
          check = energy(v0,i-1) + Energy(lambda,v0,j,v2,M,im_matrix(j,i));
             if(check < energy(j,i))
                energy(j,i) = check;
                position(j,i) = v0;
             end
          end
       end
    end
    %HIGH
    for j=(M - jump + 1):1:M %Move between the contours, jump below M
       energy(j,i) = energy(1,i-1) + Energy(lambda,1,j,1,M,im_matrix(j,i));
       for v0=(j - jump):1:M  %Last
          for v2=(j - jump):1:M  %Possible
          check = energy(v0,i-1) + Energy(lambda,v0,j,v2,M,im_matrix(j,i));
             if(check < energy(j,i))
                energy(j,i) = check;
                position(j,i) = v0;
             end
          end
       end
    end
end
```

C Closed contour test images

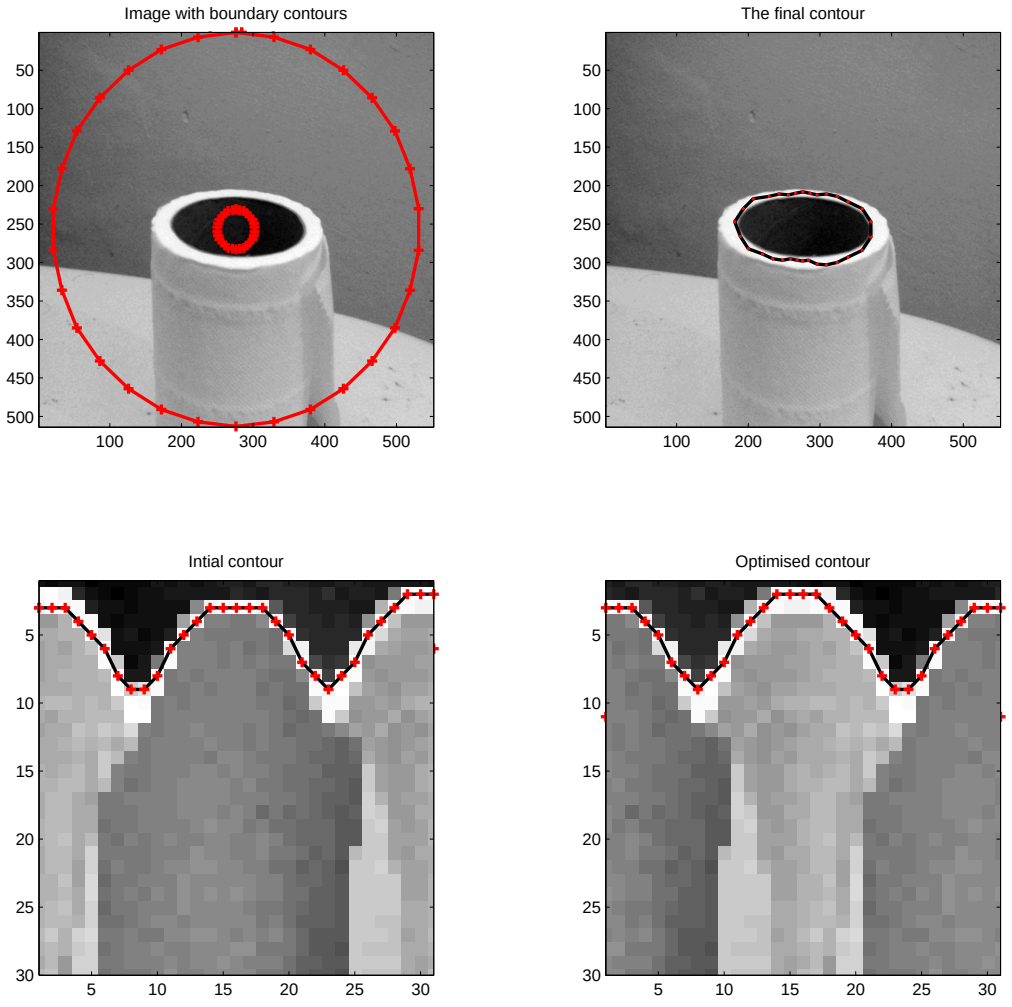

FIGURE 10: Extraction of a bog roll with a very small search space matrix. M = 30, N = 30, $\lambda = 0.7$

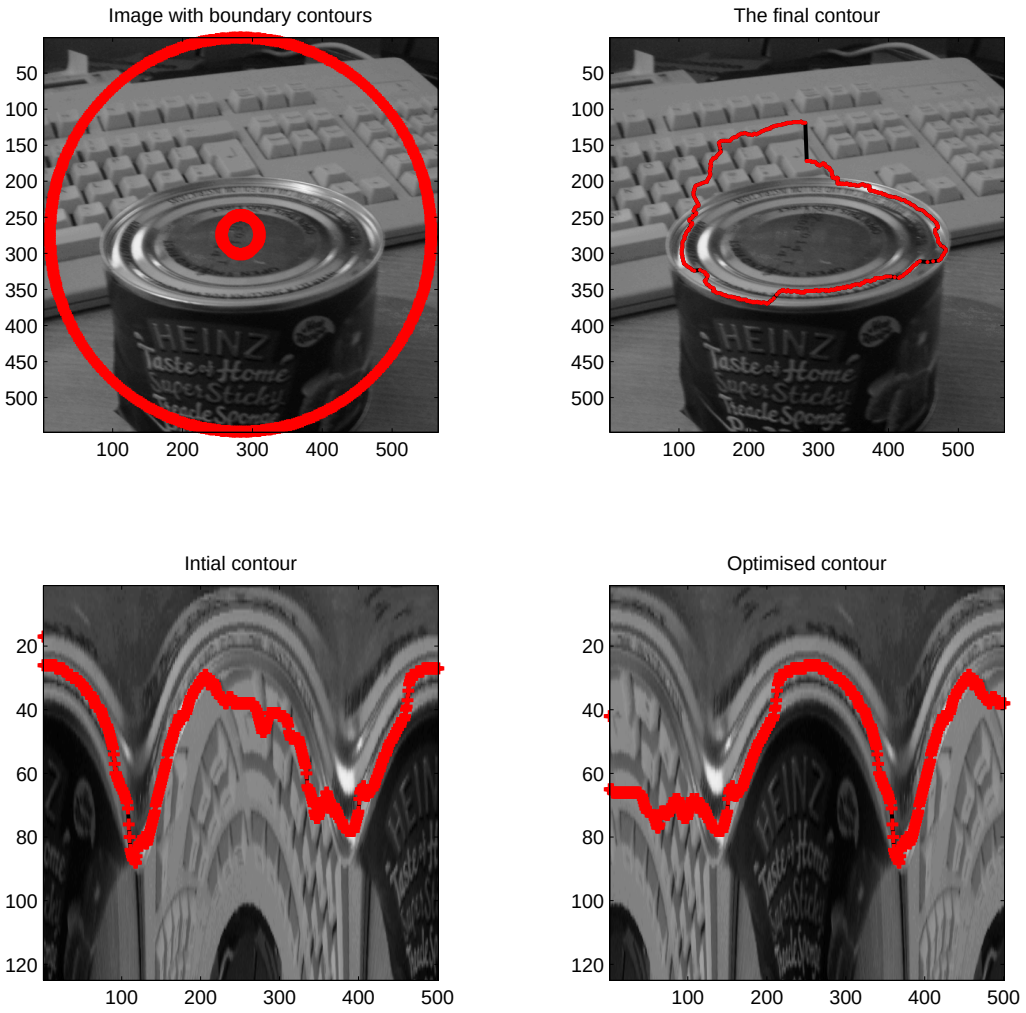

FIGURE 11: Failed extraction of a tin. M = 125, N = 500, $\lambda = 0.7$

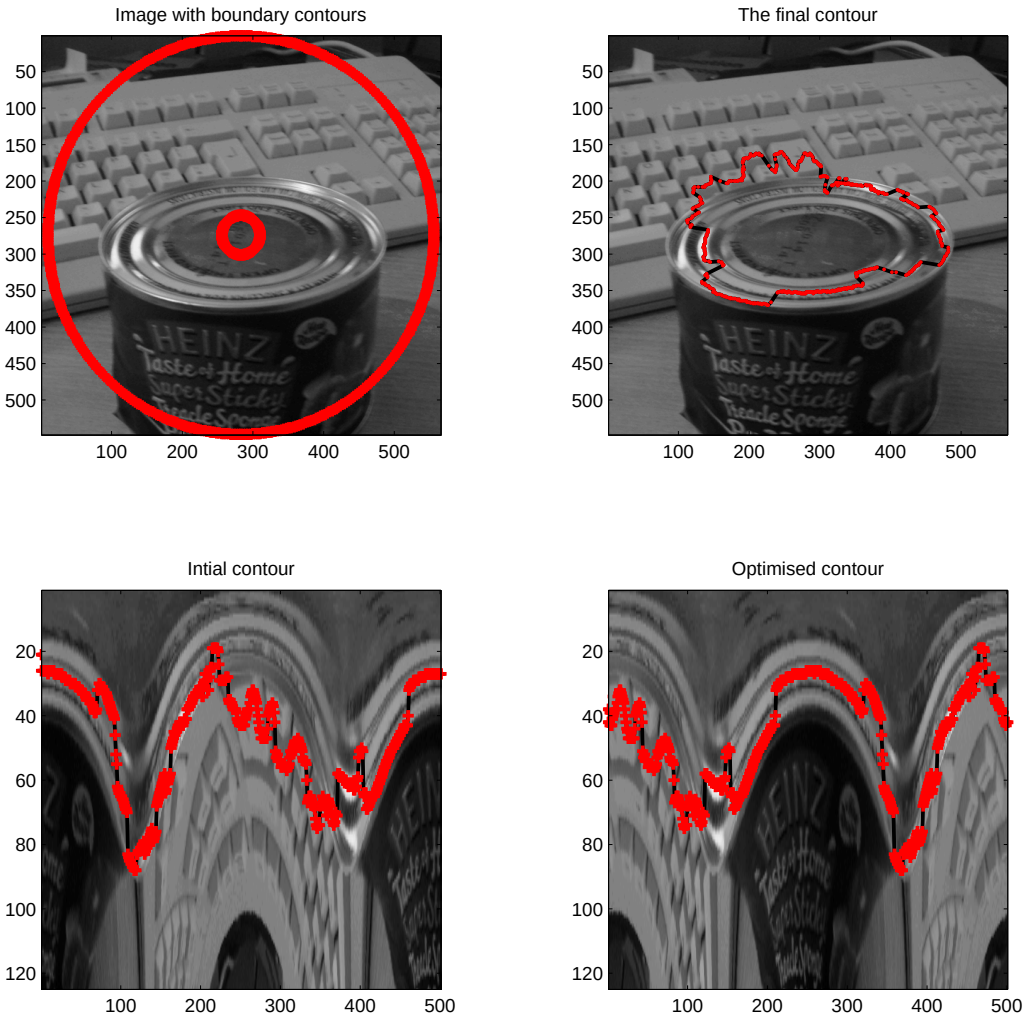

FIGURE 12: Another failed extraction attempt of a tin. M = 125, N = 500, $\lambda = 0.2$