EN161 – LAB3: Enhancement and Warping

October 6, 2004

To run the solution to a problem n, simply type pn.m.

1) The answers are shown in Figures 1 and 2.

![Figure 1: Some types of edges in Image A.](image.png)

2) Results for Image A are shown in Figures 3 and 4, while the results for Image B are shown in Figures 5 and 6.

Before comparing the methods, it should be noted that the images were generated by Matlab’s edge function, which does automatic parameter selection and some post-processing such as image thinning. With that in mind, we can start a reasonably fair comparison of the images we obtained.

Canny performed the best. For example, it was the only method that detected the computer and the desk, and the corners were quite sharp. All the other methods missed the desk and computer part of Image A, because it is in low contrast. Canny was the best for preservation of corners also for Image B. This can be seen if we compare the corners in the windows in each edge image; most methods failed to capture that corner. Canny was
not totally A to noise, but it was the only one to detect reflectance edges due to folding of on the paper background of Figure A. Another plus in Canny was that it yielded almost no isolated points; all small edge segments that it gave was due to some significant feature in the image; rarely these small features were noise. On the other hand, many Laplacian, Prewitt and Sobel edges were small and spurious, although for the Laplacian case it is more due to thresholding than to the method itself.

Laplacian usually gives more closed edges and smoother contours than the others. This is not always desirable. Generally it closes contours that should not be closed. For example, on Figure 5(b) and 5(b) we can see that the hands of the “humanoids” are “melted” with features that they are in touch with, making them less recognizable. It must be noted that Laplacian should not give so many “broken” edges as in Figures 5(b) and 5(b). We note that this is certainly due to thresholding, because the corresponding contours at zero threshold are rarely broken, as shown in Figures 5(c) and 5(c).

Shadow edges in Image A were captured by all the methods, as long as they were contrasting enough. On the other hand, shadow edges in Image B were not captured, because they were too faint. Canny detected reflectance edge in Image A (the wedge in the computer monitor), but the
other methods didn’t. All the methods also failed to detect the wedge corresponding to the chimney of the house in Figure B.

Sobel and Prewitt performed very closely on all the cases. I could not find a specific case where one was different than the other in any meaningful way. Sobel should be better immune to noise than Prewitt, but there was no evidence of this on my tests.

Of the edges that occur in Image A and B, the most difficult to detect was of the reflectance type and texture. Texture edges usually are detected at least by Canny, but they are usually mixed with many small edges representing the small texture features. Therefore, texture edges were detected to some extent by all methods, but they are somehow indistinguishable from noise and many nearby features. High contrast texture edges were detected by all methods. See for example the frontier of the light grass with dark bushes in Image B.

In summary, Canny performed the best, although it does give too much detail that is mostly significant but may not be desirable in the application. Sobel and Prewitt both gave clear major outlines, but lacked capturing of significant details, giving mostly isolated and broken edges in these cases. Laplacian stood between Sobel/Prewitt and Canny; it gave more closed edges and better capturing of image details, but not as well as Canny. This last point can be seen by comparing the endings of the tools carried by the men in Image A, as well as the interface between their hair and their face. The cable of these tools and the hair regions detected by Sobel and Prewitt were mostly not closed contours, while Laplacian closed many of these contours and Canny correctly closed almost all of them.

Scripts: p2.m, p2_run.m.

3) Results for Image A are shown in Figures 7 to 9 while the results for Image B are shown in Figures 10 and 12.

The direction of the gradient is roughly orthogonal to the edges, as shown in Figures 8 and 11. The overall lateral non-max suppression algorithm works surprisingly well (Figures 9 and 12). For Image A, I used a gradient threshold of 0.35 and a smoothing with sigma 1, while for Image B I used a 0.3 threshold and sigma 2.

Scripts: p3.m, p3_run.m, edge_thin.m.

4) Results are shown in Figures 13 and 14. They are not very animating, unfortunately. The border of the windows in Image B were not captured properly. However, for the simpler Image A the result is quite clean. The edges were a lot disconnected in this method.

Scripts: p4.m, myedge.m.
Figure 3: Result of Matlab's Canny- and Laplacian-based edge detectors on Image A.
Figure 4: Result of Matlab’s Prewitt- and Sobel-based edge detectors on Image A.
Figure 5: Result of Matlab’s Canny- and Laplacian-based edge detectors on Image B.
Figure 6: Result of Matlab’s Prewitt- and Sobel-based edge detectors on Image B.
Figure 7: Smoothed Image A (a) and its gradient magnitude image (b).
Figure 8: Gradient vectors for a small window of Image A.
Figure 9: Edges given by lateral non-max suppression on Image A.
Figure 10: Smoothed Image B (a) and its gradient magnitude image (b).
Figure 11: Gradient vectors for a small window of Image B.
Figure 12: Edges given by lateral non-max suppression on Image B.
Figure 13: Edges given by the bicubic interpolation algorithm.

Figure 14: Edges given by the bicubic interpolation algorithm.