Image Processing Summative Assignment

jrvh15: 000280744

January 4, 2017

1 System Design

1.1 Image Segmentation

On images with a slightly illuminated background, separating the worms from the background proved to be a challenge, due to the gradient caused by the illumination. It was not possible to build a template of the background either to perform background subtraction, as each image was illuminated differently.

Initially, the Laplacian filter was applied to try to separate the worms from the background, but due to the gradient, this resulted in large patches of noise that made the worms indistinguishable. Contrast Limited Adaptive Histogram Equalization (CLAHE) was applied to attempt to make the worms more distinct but increasing the contrast of the image created more noise and patches of missing information. Ultimately, to get an approximation of the background, Gaussian blur (large kernel) was applied to a copy of the image, then subtracted from the image, allowing the illuminated background to be removed (Fig. 1). After applying median blur to remove as much noise as possible, the edges in the image were extracted with the Canny filter; with any remaining large noise blobs flood filled with 0s.

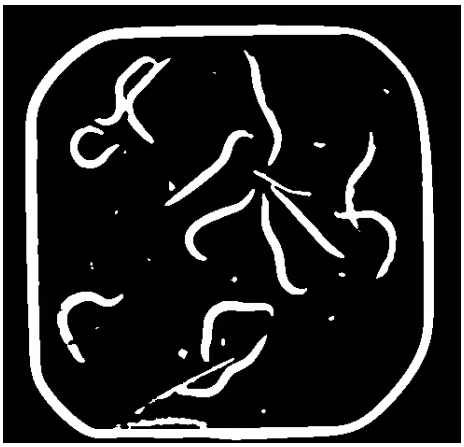

Figure 1: After the approximation of the background has been subtracted from the original image.

On black backgrounds, however, the gradient on the worms' bodies supplied more information (refer to Section 1.2); thus, histogram equalization was applied (Fig. 2) instead of processing the image in binary. The image was processed in binary only to identify clusters, following thresholding and Otsu's binarizaton, as binarization made the worms more clearly defined.

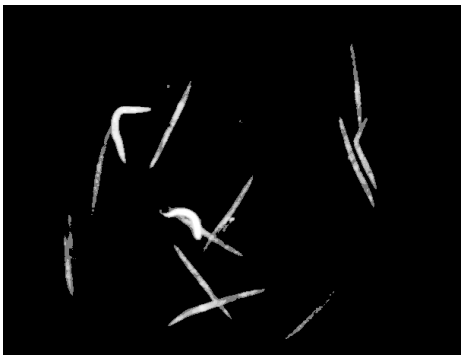

Figure 2: Histogram equalization applied to images with black backgrounds to make worms more prominent.

Figure 3: Applying CLAHE to the image with a black background displays less success in making worms more prominent.

1.2 Worms Detection

Although the watershed algorithm would have provided a better solution, especially with clusters, it requires Distance Transform, which would produce a high error margin due to the objects being too narrow.[1]

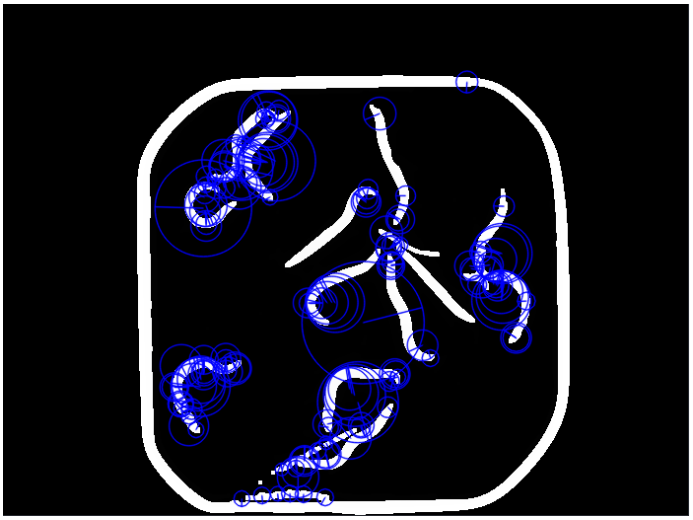

Figure 4: SURF algorithm detecting ROI. The prominence of the ROI detected is illustrated by the radius of the circles drawn. ROI are ranked according to prominence, and the algorithm returns the exact number of ROI specified; thus, it cannot be used to detect worms, which are of arbitrary numbers and shapes.

[1]Confirmed in *Image Segmentation with the Exact Watershed Transform* (H. Meine, U. Kothe): `hciweb.iwr.uni-heidelberg.de/system/files/private/downloads/867911548/meine_05_subpixel-watershed-algorithm.pdf`

While experimenting, the Speeded-Up Robust Features (SURF) algorithm was used to try to detect worms, and although regions of interest (ROI) were successfully extracted, this information was not particularly useful. As such, a simple blob detector was set up with custom parameters instead: allowing the worms to be detected based on their convexity, inertia, and area. Due to the fact that the blob detector detects worms based on convexity, both the image and its inverse (using bitwise-not) have to be used for all the worms to be identified. This method, however, has allowed for overlapping worms to be identified, as worms are detected so long as they have a curved border (Fig. 5 & Fig. 6).

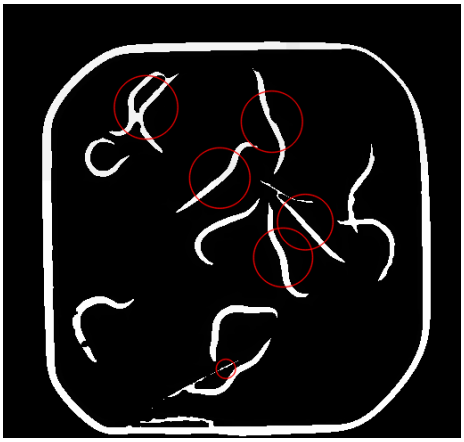

Figure 5: Worm detection based on convexity, inertia, and area of blob. The red circles indicate worms detected. The remaining worms are identified in Fig. 6.

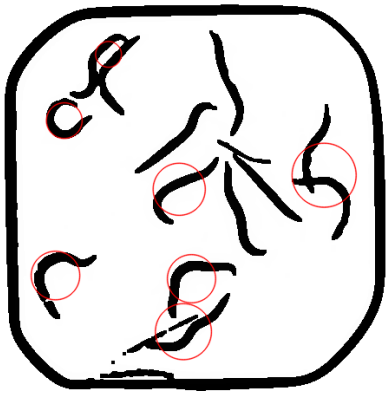

Figure 6: Worm detection on the inverse of the image in Fig. 5.

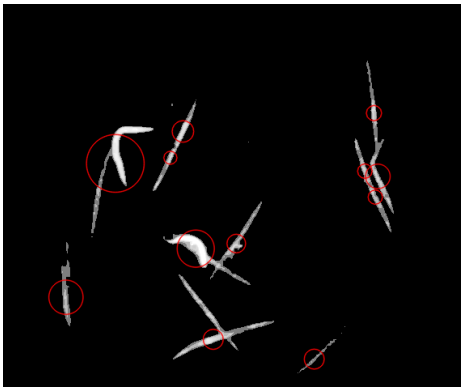

Figure 7: Identifying worms on a black background after histogram equalization, without binarization. However, because it is affected by pixel intensity, the blob detector may inaccurately detect worms that are slightly split in half.

Figure 8: Inverse of worms after histogram equalization.

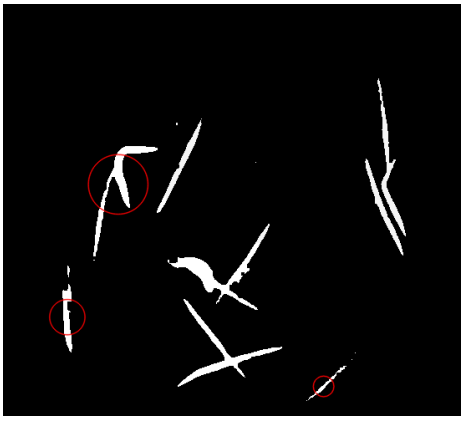

Figure 9: Comparison of worm detection accuracy in images with black backgrounds, after binary thresholding and Otsu's binarization, against histogram equalization.

Figure 10: Worm detection on the inverse of Fig. 9. This shows that preprocessing with histogram equalization yields more accurate results.

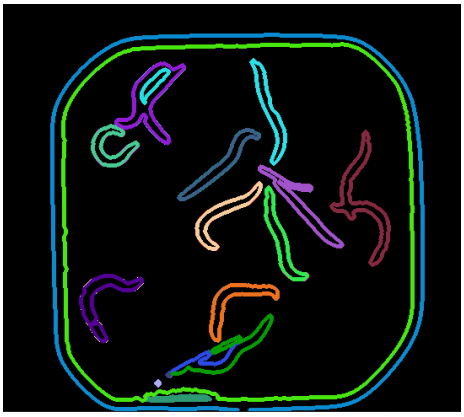

Figure 11: Segmenting and counting based on contouring returns less accurate results, due to errors caused by overlap; this leads back to the same problem when two worms share an edge. Blob detection outperforms this method.

1.3 Cluster Detection

Erosion (2 iterations) was used to try to separate clustered worms; however, in order for the blob detection to work more accurately, the worms had to be dilated (3 iterations), and this negated the effects of the erosion. However, the step was still carried out, as eroding removed noise blobs; but, as much as possible, the opening and closing transformations were used to try to separate touching borders.

Although the algorithm is able to detect worms that are connected at a single point separately at most times, it is unable to detect them as separate objects when they share an edge.

In a further attempt to identify intersections, the Shi-Tomasi algorithm was applied to detect corners (Fig. 12). The distances between each of the corners were calculated, as there is a high probability that they show an intersection where there are multiple corners within a small radius of one another (Fig. 13).

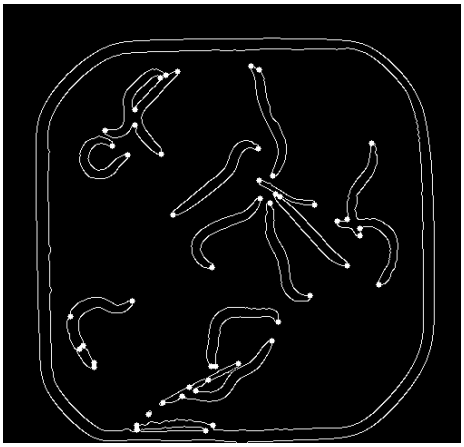

Figure 12: Corners detected by the Shi-Tomasi algorithm.

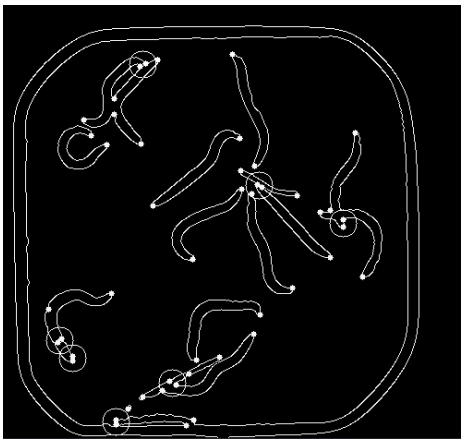

Figure 13: Intersections detected based on the distance between the corners identified by the Shi-Tomasi algorithm. The distance between each point is calculated and if it falls below a specified threshold, the region is circled, as it suggests a possible intersection.

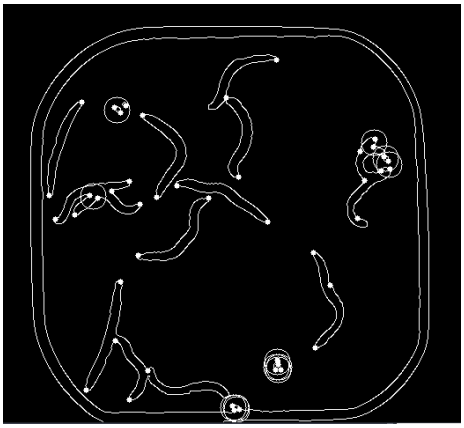

Figure 14: Where noise is not properly removed; this intersection detection method displays a high margin of error. Corners detected are ranked by "strength"; noise blobs with its more prominent corners are identified before the contours of the worms. Thus, the optimal conditions for this to work would be if the worms were intersecting/overlapping (not just touching), there is a lack of noise blobs in the image, and the worms are not curled up.

1.4 Live/Dead Classification

Hough Transform was initially used to try to classify the worms as either living or dead based on the curve of the borders (Fig. 15), as dead worms exhibit rod-like features. However, the uneven texture of the dead worms and the arbitrary shapes that the worms can take on (e.g. curved, straight, curled up) lead to inaccurate results.

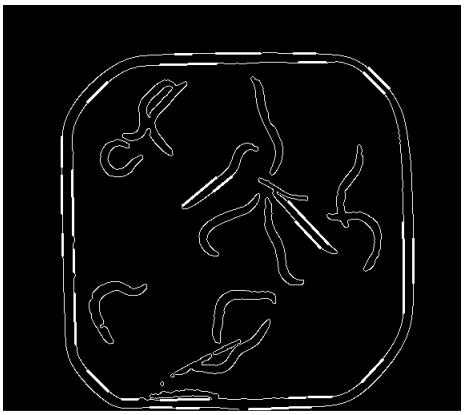

Figure 15: Hough Transform detects any straight edges within the image. However, it returns inaccurate results when worms are straight but alive, and when dead worms have rough edges.

Thus, the uneven texture of dead worms was used to classify them instead. Live worms exhibit well-defined, straight borders while dead worms show rough edges. Using the fit ellipse function to annotate small patches allowed for dead worms to be identified (Fig. 16), but this may produce a margin of error where noise is not completely removed from the image (Fig. 17).

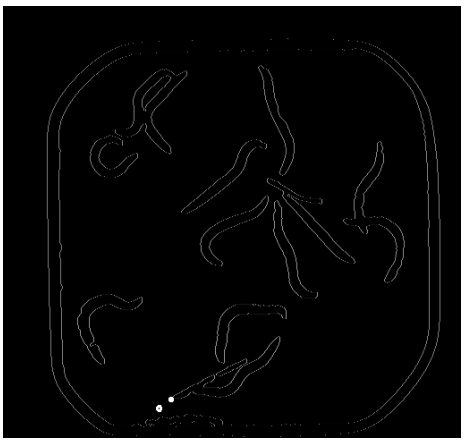

Figure 16: Detecting worms based on their texture, i.e. borders on dead worms are far less defined, and instead appear as small blobs, which is highlighted in the picture. Ellipses are drawn around these rough or broken edges, identifying the worm as dead.

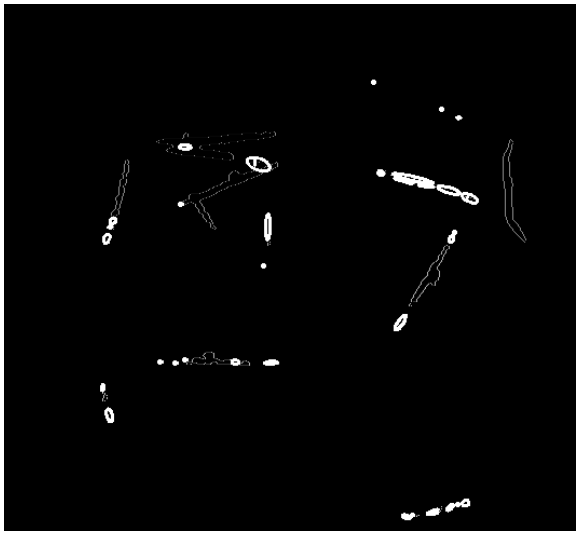

Figure 17: Small margin of error due to noise blobs. The live worm on the right with its clearly defined, continuous border is left unannotated.