Edge Detection Evaluation of HEp-2 Cells in Fluorescent Images

Ossama Edbali - ID: 1466610

December 9, 2015

1 Aim

This study aims to analyse the results and efficacy of several edge detection algorithms on HEp-2 cells.
The edge detection/segmentation algorithms used are: Otsu, Sobel, Roberts, Canny, Canny using anisotropic diffusion, Laplacian of Gaussian, Difference of Gaussians, dilate-erode method and localised histogram edge detection (the last one is my development).
The evaluation of each method is carried by ROC curves, sensitivity-specificity variation graph and correspondence analysis.
All the Matlab code and full data can be found here: `https://github.com/UoBCS/hep2-edge-detection`.

2 Method

Given the three images (9343 AM, 10905 JL and 43590 AM) it can be seen that they are the outcome of different fluorescence patterns on the HEp-2 cells. First of all the experiments where conducted using three main scripts: `x9343AMtask`, `x10905JLtask` and `x43590AMtask`.
Each script loads the relative image and true edge image (1). Then converts the original image to a grey-scale image using the `rgb2gray` Matlab function (2). Finally, it performs all the edge detection algorithms described above (delegating to the class `EdgeDetection.m`) (3).
A pre-analysis (qualitative) of the input has been carried out using the background approximation image as a surface to see where illumination varies (using `show_background` function). It can be seen that the various input images use different fluorescence patterns on the HEp-2 cells.
For all edge detectors there is a method inside the `EdgeDetection` class as well as methods for generating the ROC space and specificity and sensitivity in function of some parameters (e.g. threshold, sigma, number of iterations).
For Roberts and Sobel a function `detect_edges(img, filterX, filterY)` was developed which convolves `img` with the two filters and uses `magnitude` to produce the final result (before thresholding).
In regards to LoG, Gaussian smoothing was performed using `gaussian_smoothing(image, sigma)`. Then convolve the smoothed image with the laplacian operator and finally find the zero crossings: `edge(res, 'zerocross')`.

Otsu's method

One of the algorithms used in this study is the Otsu's algorithm. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum. First of all I calculate the weighted mean and variance for both the background and foreground (μ_b, σ_b^2 and μ_f, σ_f^2). Then I compute the **within-class variance** (two variances multiplied by their associated weights): $\sigma_W^2 = W_b\sigma_b^2 + W_f\sigma_f^2$. I do the same calculation for candidate thresholds and the one that has the lowest within-class variance will be chosen. This algorithm is implemented in the Matlab's `graythresh`. This will produce a binary image which edges' can be detected easily by any of the edge detectors (Sobel in the experiment).

Dilate-erode method

The dilate-erode method is a 4-step algorithm that uses a combination of morphological operators before using Sobel edge detector. The reason why this method has been used is that when Sobel (or Roberts) is performed on the grey-scale image, there are some lines of high contrast surrounding the actual edge (like noise). The gaps between these small lines and the actual edge can be filled using a **dilate morphological operator** (or non-linear neighbourhood operator) with a structuring element (from the `strel` function). Then `imfill` has been used to fill the holes inside the cell. The next step is to reduce to get the cell back to its state before dilation using the **erosion** operator. Finally the Sobel filter has been used to detect edges. There are various ways to create the morphological operators. Here is the one used in this study (erode example using a 5x5 mask): `erode = @(x) min(x(:)); out = nlfilter(in,[5 5],erode)`.

Canny with anisotropic diffusion (CAD)

Anisotropic diffusion is a non-linear diffusion filtering for avoiding the blurring problems of Gaussian smoothing (and others). It is an iterative process to perform edge-preserving smoothing on the cells. The algorithm is described mathematically as follows:

$$\frac{\partial}{\partial t} I(x,t) = \nabla \bullet (c(x,t) \nabla I(x,t))$$

Here t is the iteration step, $c(x,t)$ is a decreasing diffusion function of the image gradient magnitude.
The `anisodiff2D(im, num_iter, delta_t, kappa, option)` function in the codebase has been implemented to accomplish anisotropic diffusion. Here `im` is the input image, `num_iter` is the number of iterations, `delta_t` is the integration constant, `kappa` is the diffusion constant and `option` defines which diffusion function to use.

I adopted three evaluation methods in order to assess the accuracy/efficacy of each edge detector technique:

ROC curve

Here `ID` is the image with detected edges and `IT` is the image with true edges.

```
function [ sensitivity, specificity ] = compute_roc( ID, IT )
     TP = ID & IT; TP = sum(TP(TP == 1));
     FP = ID & ~IT; FP = sum(FP(FP == 1));
     FN = ~ID & IT; FN = sum(FN(FN == 1));
     TN = ~ID & ~IT; TN = sum(TN(TN == 1));
     sensitivity = TP / (TP + FN); specificity = TN / (TN + FP);
end
```

An optimal/ideal edge detector would produce a value of (0, 1) or $spec = sens = 1$. This is the primary evaluation method that I used to compare the various edge detectors as well as how they perform differently on each input image.

Sensitivity/Specificity variation

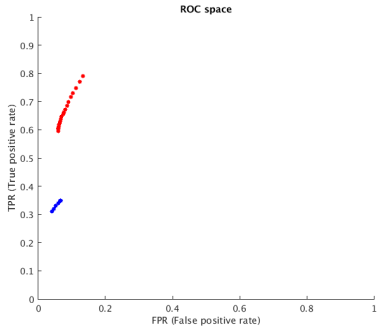
This evaluation method simply uses the sensitivity and specificity parameters in function of a variable (threshold, sigma, kappa or number of iterations). From this we can see how different algorithms respond to variation of algorithm's parameters. This is implemented by the `roc_params_comparison` static method in the `EdgeDetection` class.

Correspondence analysis (intra-method analysis)

From N edge detection results (using the same edge detector) test for correspondence as follows: a pixel location identified as an edge by all N detector configurations (i.e. changing parameter) will have the highest correspondence (N), and a location identified as an edge by only one detector setup will have the lowest.
This is computed by the `correspondence_analysis` function in the codebase which adds all the input image matrices and, using Matlab's `bar`, plots a bar diagram with frequencies for each correspondence level (using `histc`).

3 Results

A very interesting result is the comparison between Canny and Canny with anisotropic diffusion (using 9343 AM as a reference):



(a) ROC space for CAD (red) and Canny (blue)

It can be noted that Canny does not perform very well (sensitivity under 0.4). However if we apply anisotropic diffusion, sensitivity increases drastically (over 0.8). This happens because the input image is very dark at the borders and Canny looses information in the smoothing step as well as in the non-maximum suppression step. Supporting this statement are the results of applying Canny to 43590 AM (which is very dark compared to the other two images and therefore a high smoothing).
A summary of the whole data using the optimal parameters (**for the FULL evaluation graphs and tables please visit this site:**):

Table 1: Overall results (1: 9343 AM, 2: 10905JL, 3: 43590AM)

Edge detector	Image	Sensitivity	Specificity
Otsu	1	0.9225	0.9974
	2	0.9279	0.9957
	3	0.8813	0.9958
Sobel	1	0.8498	0.8638
	2	0.8953	0.8852
	3	0.8195	0.8237
Roberts	1	0.8477	0.8400
	2	0.8646	0.8779
	3	0.7540	0.8215
Canny	1	0.3404	0.9408
	2	0.3057	0.9863
	3	0.2396	0.9808
CAD	1	0.7910	0.86651
	2	0.8139	0.9204
	3	0.6734	0.8921
Dilate-Erode	1	0.8299	0.8058
	2	0.8003	0.8119
	3	0.7446	0.7863
LoG	1	0.2250	0.9419
	2	0.2135	0.9324
	3	0.1996	0.9207
DoG	1	0.2137	0.9857
	2	0.2446	0.9941
	3	0.17301	0.99094

4 Conclusion

From the evaluation Otsu's method performed the best in all input images:

Table 2: Otsu evaluation

Image	Sensitivity	Specificity	Threshold
9343 AM	0.9225	0.9974	0.0902
10905 JL	0.9279	0.9957	0.1333
43590 AM	0.8813	0.9958	0.0510

It can be noted that Canny does not perform very well (sensitivity under 0.4). However if we apply anisotropic diffusion, sensitivity increases drastically (over 0.8). This happens because the input image is very dark at the borders and Canny looses information in the smoothing step as well as in the non-maximum suppression step. Supporting this statement are the results of applying Canny to 43590 AM (which is very dark compared to the other two images and therefore a high smoothing).