Particle Edge Detection Using Machine Learning Techniques

Michail Alifierakis

Abstract

Edge detection is the area of computer vision that deals with the recognition of edges and boundaries in images. Since the introduction of the Canny edge detection algorithm many similar approaches with improved accuracy of edge detection have been proposed. Here we are focusing on creating a faster edge detection method by combining supervised learning algorithms and existing edge detection methods. More specifically we recommend the use of common edge detectors for the creation of a training set that will be used to train algorithms with a small number of features. The consequent reduction of computational time required by this new algorithm will make it more suitable for analysis of large data sets. Even though this method sacrifices accuracy, it can still provide a useful tool for researchers to extract data from a large number of microscopy images and subsequently help in the development of more robust theoretical models.

Introduction

One of the traditional challenges in microscopy imaging for scientific purposes is image analysis. Images obtained with different techniques (Optical Microscopy, Transmission Electron Microscopy, Scanning Electron Microscopy etc.) have to be interpreted in order for observations to be reported numerically and for trends to be identified. For instance, we want to recognize accurately and with little computational effort objects such as cells, nuclei, bacteria and metal or polymer nanoparticles and calculate their number, size, volume, concentration, etc. This particle recognition can be done with edge detection techniques.

Edge detection is the area of computer vision that identifies points of discontinuity in the brightness value of images. These discontinuities are usually caused by:

- Depth discontinuities
- Surface orientation discontinuities
- Change in material properties
- Variations in scene illumination

Many algorithms have been developed over the years with the purpose of recognizing the change in the above properties [1–3]. The goal is to recognize material property-changes (i.e. object boundaries) while ignoring the noise in an effort to identify the objects captured in an image. The detection of edges is a critical preprocessing step for a variety of tasks, including object recognition, segmentation and active contours.

Perhaps the most widely used edge detector is the Canny edge detector developed by John F. Canny in 1986 [4]. In this work a multi-stage algorithm was presented along with a computational theory of edge detection that explains why this technique works (for more details see the Background section). His theory gave rise to improved edge detectors, Gaussian [5] and non-Gaussian [6–8] (like the Deriche edge detector [6], which uses recursive filters for image smoothing and reduces the

required computations per pixel). Canny's edge detector is still considered state-of-the-art although it was created in the early days of computer vision [9,2].

Here we present an algorithm based on machine learning techniques that will recognize particles faster than the Canny edge detector. Potential application areas are: automatic particle detection [10], particle size monitoring [11] and particle velocimetry [12]. Since for the cases of particle recognition a Canny edge detector is usually sufficient we will train machine learning algorithms based on the data generated by the Canny edge detector. In other words, we can generate a label for each pixel (edge or non-edge) using the Canny edge detector and then transform our edge detection problem to a classification problem.

Machine learning techniques have been used in the past as part of corner detection algorithms [13,14]. They have also been used for very specific applications in image processing like facial expression classification [15] and text detection in images [16], tissue classification in Magnetic Resonance Images [17] or boundary detection in images of natural scenery [18]. To the best of our knowledge there is no machine learning approach for the edge detection of particles and in particular there is no supervised learning algorithm that uses existing edge detectors to produce the labels for the training set.

We do not intend to produce a general-purpose operator as most previously published algorithms intended to. We are, instead, proposing a methodology to tackle specific edge recognition problems. In the Proposed Research section we are illustrating that our approach works for the case of particle edge detection from Transmission Electron Microscopy (TEM) images and in the Future Work section we state our future plans for improvement of the presented algorithm.

Specific Goal

Our goal is to produce a particle edge detection algorithm that operates faster than the Canny edge detection algorithm. Our approach sacrifices accuracy comparing to the Canny algorithm but will be more appropriate for processing of a large number of similar images and for live particle monitoring.

Background

Sobel Operator

The Sobel operator, developed by Irwin Sobel and Gary Feldman [19] is a discrete differentiation operator. It essentially computes an approximation of the gradient of the image intensity function.

The 3x3 form of the Sobel kernels for the horizontal ($\boldsymbol{G}_x$) and the vertical directions ($\boldsymbol{G}_y$) are defined as follows:

$$\boldsymbol{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \boldsymbol{I} \quad (1)$$

$$\boldsymbol{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \boldsymbol{I}, \quad (2)$$

where * is the convolution operator and $\boldsymbol{I}$ is the intensity matrix of the source image. The result of this convolution operation between a 3x3 kernel $\mathbf{A}$ (with elements $a_{i,j}$) and a pixel $p_{i,j}$ is equal to: $a_{1,1}p_{i+1,j+1} + a_{1,2}p_{i,j+1} + a_{1,3}p_{i-1,j+1} + a_{2,1}p_{i+1,j} + a_{2,2}p_{i,j} + a_{2,3}p_{i-1,j} + a_{3,1}p_{i+1,j-1} + a_{3,2}p_{i,j-1} + a_{3,3}p_{i-1,j-1}$. This means that the calculation of each convolution matrix requires 9 multiplications and 8 additions per pixel.

The gradient approximation ($\boldsymbol{G}$) used in the calculations below is defined as:

$$G = \sqrt{{G_x}^2 + {G_y}^2} \quad (3)$$

This gradient approximation brings the total **operations per pixel to the number of 38**.

Canny Edge Detector

The Canny edge detector [4] is essentially a series of steps that are designed to yield good edge detection results. More specifically the steps are:

1) Image smoothening.
2) Calculation of intensity gradients.
3) Non-maximum suppression
4) Double threshold implementation.
5) Hysteresis threshold implementation.

Image Smoothening

This step has been criticized in the literature since it can oppress the signal of weak edges. In his paper Canny suggests image smoothening through a two-dimensional Gaussian. In our calculations we used a 5x5 Gaussian operator, which requires 25 multiplications and 24 additions, i.e. **49 operations per pixel**.

Intensity Gradients

In step 2, the intensity gradient for each pixel is calculated using a Sobel operator and we define the angle of the direction of the gradient as:

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (4)$$

This calculation is then rounded to one of four angles representing vertical, horizontal, and two diagonal directions. The total number of operations per pixel for the calculation of the angle of gradient direction is 38 (including two operations on average for rounding the angle), while the gradient calculation using the Sobel operator requires 38 operations as we stated earlier. 34 of those operations are

common for the angle and the gradient, bringing the total to **42 operations per pixel** for this step.

Non-Maximum Suppression

At step three, we find the potential edges by suppressing the points that are not local maxima. Given the direction of the gradient and the gradient values that we found on the previous step we compare the gradient values of the current pixel with the two surrounding pixels in the direction of the angle of the gradient. If the current pixel has higher gradient values than its surrounding pixels then it is considered an edge, else it is considered a non-edge. This step requires about **2 operations per pixel**.

Double Threshold Implementation

In this step we take the pixels identified as potential edges in the previous step and we compare their gradient values to two thresholds. If the gradient value exceeds the value of the high threshold then the pixel is characterized as an edge. If the gradient of the pixel is lower than the value of the low threshold then the pixel is characterized as a non-edge. This step requires an average of **2 value comparisons** per pixel.

Hysteresis Threshold Implementation

For the pixels with intensity values in-between the low and high threshold values of the previous step, we only qualify as edges the ones that are neighbors of pixels that are already qualified as edges (i.e. their gradient exceeds the high threshold value). This step requires looking up the gradient values of the intensity of 8 surrounding pixels for each pixel with gradient value between the two thresholds but we do not take into account looking up values as a per pixel mathematical operation and thus this step does not contribute to the total number of operations per pixel of the Canny algorithm.

The Canny algorithm requires about **95 operations per pixel** in total. The choice of the low and high threshold values is manual and depends on the image properties.

In figure 1 we see a TEM image of Janus particles and the application of the Canny algorithm to find the edges of those particles.

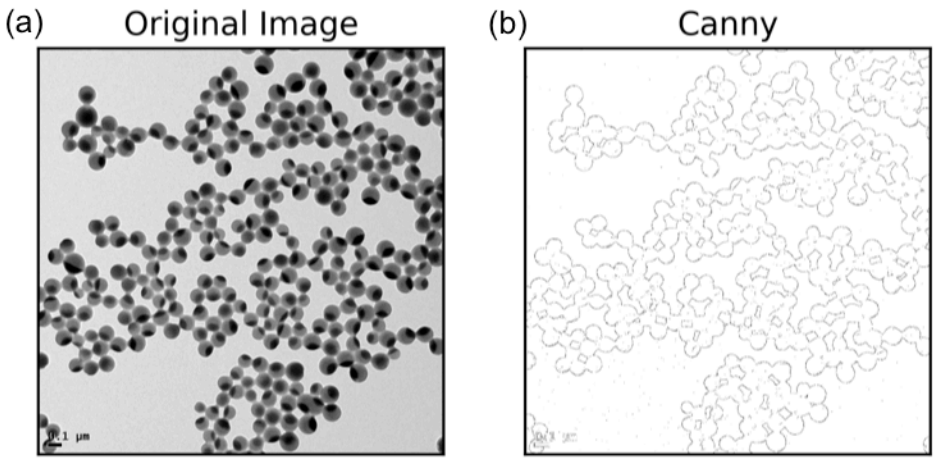

Figure 1. (a) The original TEM image of janus polymer nanoparticles courtesy of Chris Sosa, (b) the edges of image (a) generated by the Canny edge detection algorithm. The edges are shown in black and all the white pixels are non-edges.

Proposed Research

Our goal is to create a faster edge detection algorithm more appropriate for a large number of similar images using machine learning techniques. We are tackling the problem of identifying the edges as a supervised classification problem. As we mentioned earlier, the Canny edge detector is still considered state-of-the-art and it performs very well on microscopy images with a good intensity contrast. For this, we have used Canny to produce our training data set. The trained algorithm can then be applied to other images.

Feature Selection

Since the image edges correspond to points of sudden change of intensity it seems that the gradient should be an important feature for edge detection. For this reason the first feature we will extract from each data point (pixel) in the image is the Sobel gradient ($\boldsymbol{G}$). One more feature that we take into account in our analysis is the intensity value of each pixel.

The Transmission Electron Microscopy (TEM) image presented in figure 1(a) contains approximately 4 million pixels. This is a very large feature set for most machine learning algorithms and for this we will reduce it to 20,480 pixels. This corresponds to 10 rows of pixels. Our training set is essentially the region within the black lines shown in figure 2(a) and magnified in figure 2(c).

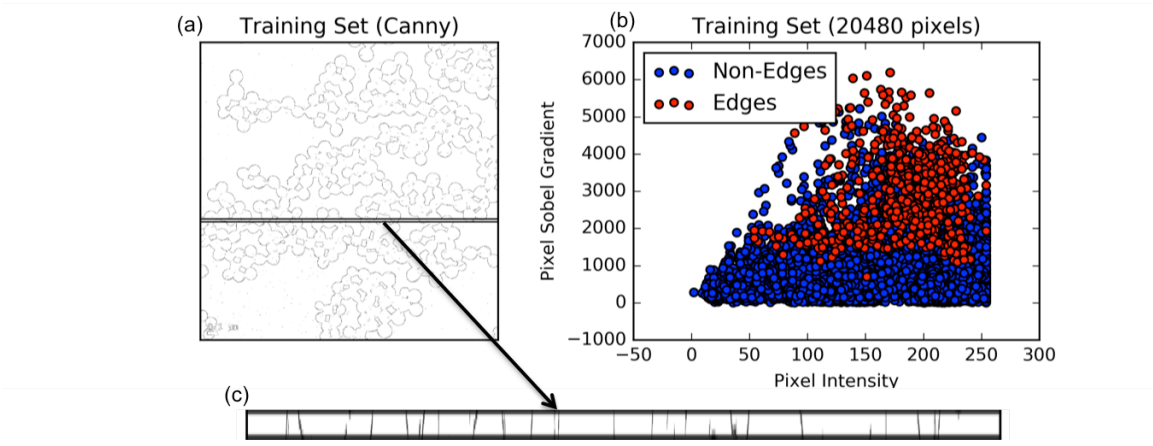

Figure 2. (a) The edge image as identified by the Canny operator. The region between the white lines contains the pixels that are used as our training set. (b) The data points that are used as our training set appear here. Red points correspond to edges and blue points correspond to non-edges (as identified by the Canny algorithm). (c) Our magnified training set. The height of this image only contains 10 pixels and might appear distorted.

The red points in figure 2(b) correspond to edges while the blue correspond to non-edges. As we can see there is a lot of overlap between red and blue points, which raises the need for the introduction of more features to describe our data or to improve the already existing features.

Fitting

The use of a simple logistic regression model with linear boundaries roughly separates the data in edges and non-edges (brown and blue respectively in figure 3). The background is separated in a brown and a blue area by a linear boundary based on the results of logistic regression while the color of the data points shows the results from the Canny algorithm.

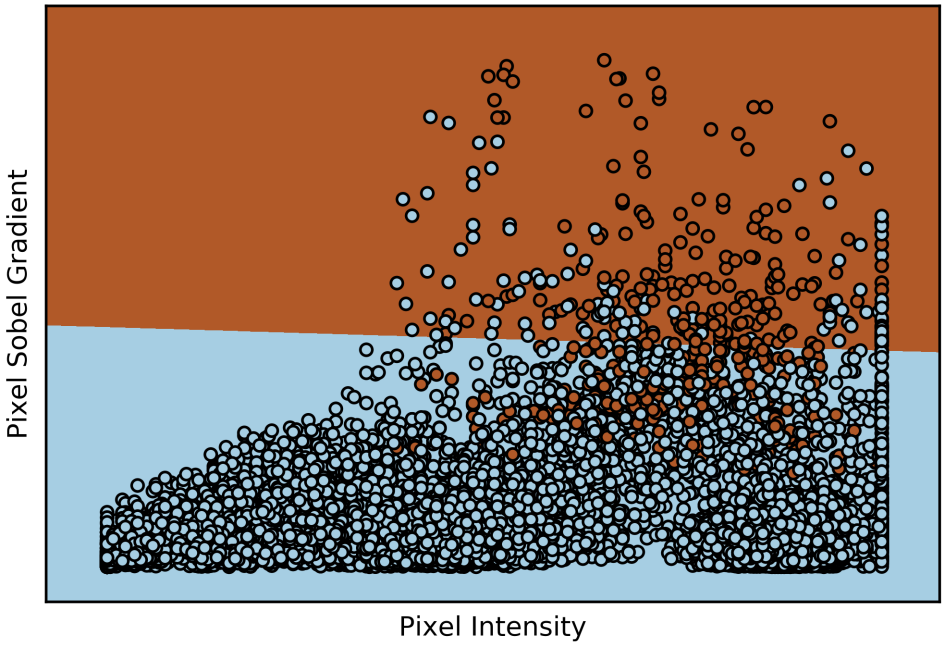

Figure 3. The top 30,720 pixels (15 rows of pixels) of our dataset. The edges appear as brown points and the non-edges as blue points, as identified by the Canny algorithm. The brown and blue regions behind the points show the result of logistic regression, i.e. all the data points in the blue region are characterized as non-edges and the data points in the brown region as edges.

Better fitting can be achieved with the use of a polynomial equation or other machine learning algorithms (such as neural networks) but the improvements were small and not very important for illustration purposes.

Evaluation of Results

The results in practice do not look as bad as figure 3 indicates. As shown in figure 4(b), most particles seem to have been identified while for many of them the boundaries seem thicker than in the results produced by the Canny edge detection (figure 4(a)). This is a product of the lack of the step (3) of the Canny algorithm, which suppresses all the possible edges except the local maxima.

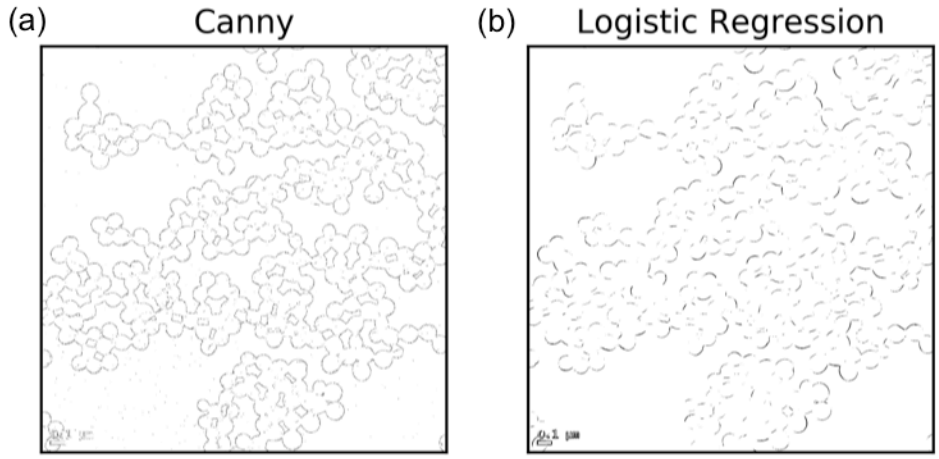

Figure 4. (a) The edge image generated by the Canny edge detection algorithm. (b) The result produced by our method.

Depending on the specific application thicker edges do not necessarily constitute a problem. 98.2 % of pixels were classified correctly as edges and non-edges using this logistic regression equation that we trained using 0.5 % of the data as our training set and 36 % of the pixels that were characterized as edges by Canny were also characterized as edges by our method. Then we used the same logistic regression function that we trained from this image to classify the pixels of two different images. The results are shown below.

Figure 5 is a very successful example of the implementation of our algorithm. In figure 5(c) all the particle boundaries can be seen clearly and they look continuous and thin. In this case 98.7 % of the pixels was classified correctly as edges and non-edges and 96.7 % of the edges identified by Canny were also identified by our method.

Figure 6 on the other hand is an example where our method did not perform very well. In figure 6(c) we can see that the boundaries are thick in certain areas and they are not continuous. 97.8 % of the pixels are classified correctly but only 28.7 % of the edges identified by Canny were also characterized as edges by our method.

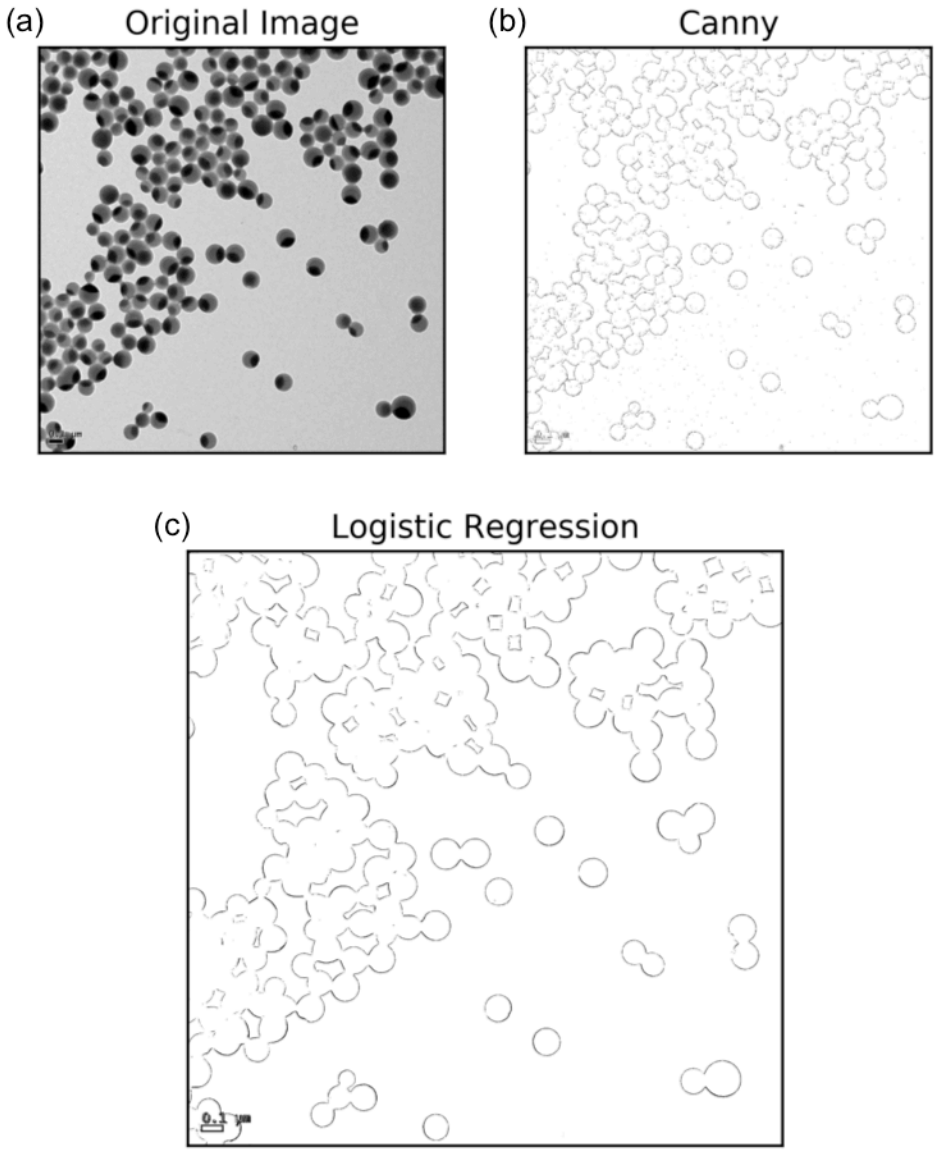

Figure 5. (a) A new TEM image of the same materials as in figure 1(a). (b) The edges as identified by the Canny algorithm and (c) the result from our algorithm using the logistic regression function trained from the pixels shown in figure 2(c).

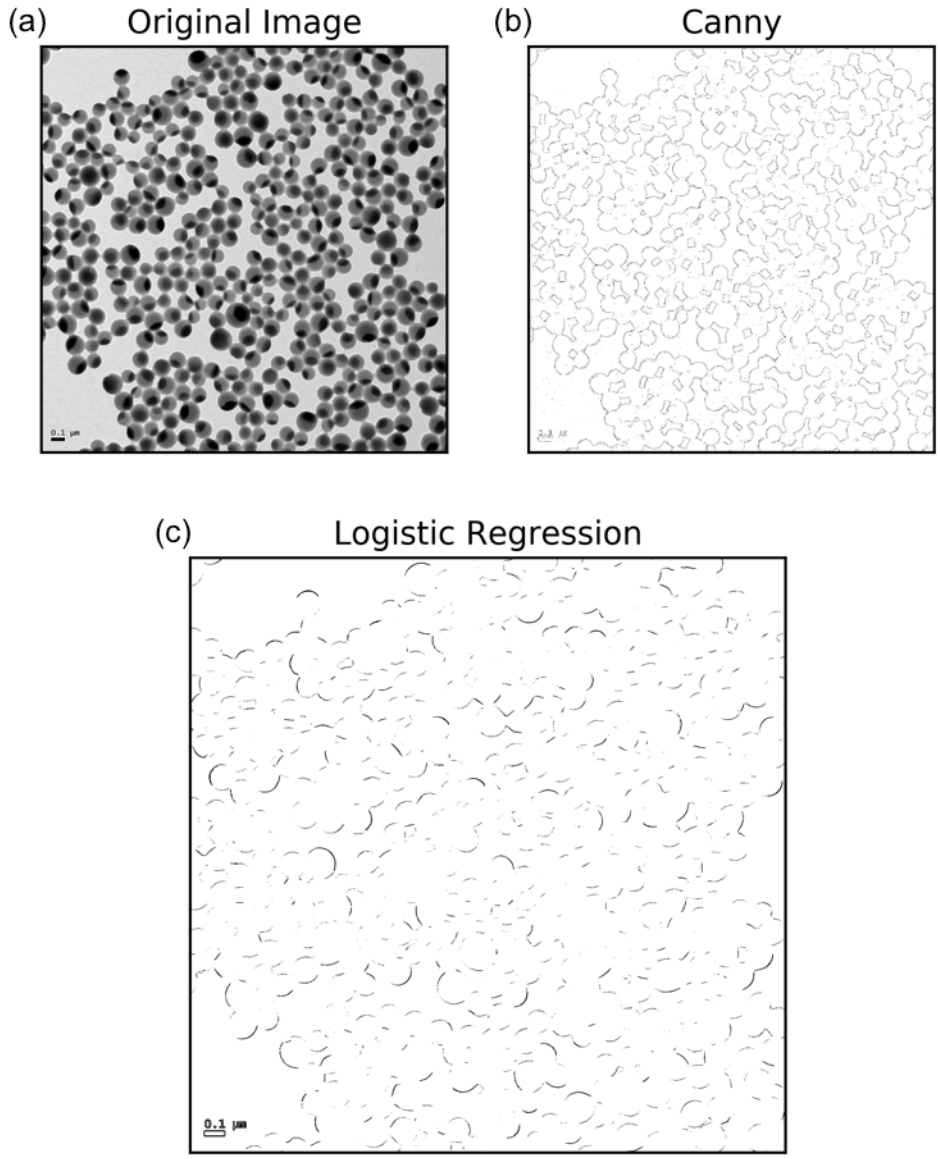

Figure 6. (a) A new TEM image of the same materials as in figure 1(a). (b) The edges as identified by the Canny algorithm and (c) the result from our algorithm using the logistic regression function trained from the pixels shown in figure 2(c).

Qualitatively we can see that figure 6(a) has a higher concentration of particles relatively to figures 1(a) and 5(a). A higher concentration of particles can create lower Sobel gradients in the particle boundaries in the concentrated regions. This is

something that affects our results when our training set does not include pixels that belong to particles in concentrated regions.

Another reason the results of figure 6 are not as good as the previous two examples is a difference in pixel intensity values. We trained our logistic regression algorithm using figure 1(a). Figure 1(a) has an average pixel intensity of 170.7 and figure 5(a) has an average intensity of 171.2. Figure 6(a), in contrast, has an average pixel intensity of 150.9. This difference could be the reason of the poor performance of our algorithm on figure 6(a).

Speed Improvement

It is hard to calculate the exact reduction in computational time achieved by our method as the performance of the Canny algorithm depends on the specific implementation. In our case, we used libraries that already implement the Sobel operator and the Canny algorithm and logistic regression. These parts were compiled and optimized but the rest of our code is neither compiled nor optimized.

Our edge detection algorithm is constituted of two basic steps: training and implementation. The training step of our algorithm can be very time-consuming depending on the number of pixels we use as our training set and the type of machine learning algorithm we use for classification. Our goal is to save time on the implementation part and given a large enough number of images the time saving in implementation should surpass the time lost from training. For a small number of images the use of this algorithm is not recommended since it reduces accuracy and it could potentially also increase processing time due to the additional training step.

Given a trained linear logistic regression equation in order to classify a pixel as edge or non-edge all we have to do is calculate the value of this sigmoid function and then round it to 1 or 0:

$$h = \frac{1}{1 + e^{-(a+bx_1+cx_2)}} \tag{5}$$

Where x_1 and x_2 are our two features and a, b and c are the constants calculated using our training set.

The operations per pixel required for the calculation of equation (5) are 8 (when we take into account the rounding of the result). To this number we need to add the number of operations required for the calculation of the Sobel gradient, which is one of our two features. Therefore, the total number of operations required by our method is **46 operations per pixel**. **This is a little less that half of the operations required by the Canny algorithm (48.4 % of the total 95 operations per pixel).**

This means that a good algorithmic implementation of our suggested method is expected to decrease the running time by half, for problems that include a significantly large number of images.

Future Work

We have presented here a promising algorithm that has potential for improvement in speed and in edge recognition capabilities. We are planning to work on three main areas for this improvement.

First, improving quality of current feature extraction. For instance, a Scharr [20] operator can be used instead of the Sobel operator.

Second, we can extract better features from our images through utilization of information about the proximity of the pixels. These relations are introduced in the Canny algorithm when at step 3 of non-maximum suppression and step 5 of hysteretic threshold usage.

Third, better fitting algorithms can be used. The use of neural networks can be promising since they can capture more complicated non-linear functions for the classification boundaries and they can deal with a large number of training samples.

Conclusions

We presented a general strategy to edge detection through the use of machine learning classification algorithms based on the labels generated by a Canny edge detection algorithm. We managed to train a linear logistic regression algorithm to classify pixels as edges and non-edges in TEM images of polymer nanoparticles with a 98 % accuracy using 0.5 % of the information contained in one of the images. The algorithm performs better in terms of quality of classification when all the images have good contrast and similar intensity values.

We also suggested ways to improve the accuracy of this algorithm through better feature extraction and better classification algorithms. The speed improvement achieved by our approach, comparing to the Canny algorithm, was not measured but we calculated that we can decrease running time by more than 50 % when it is implemented in a large enough set of images.

References

[1] D. Ziou and S. Tabbone, Pattern Recognit. Image Anal. C/C Raspoznavaniye Obraz. I Anal. Izobr. **8**, 537 (1998).
[2] E. Nadernejad, S. Sharifzadeh, and H. Hassanpour, Appl. Math. Sci. **2**, 1507 (2008).
[3] R. Maini and H. Aggarwal, Int. J. Image Process. **3**, 1 (2009).
[4] J. Canny, IEEE Trans. Pattern Anal. Mach. Intell. **PAMI-8**, 679 (1986).
[5] M. Basu, IEEE Trans. Syst. Man Cybern. Part C (Applications Rev. **32**, 252 (2002).
[6] R. Deriche, Int. J. Comput. Vis. **1**, 167 (1987).
[7] P. Perona and J. Malik, IEEE Trans. Pattern Anal. Mach. Intell. **12**, 629 (1990).
[8] N. R. Pal and S. K. Pal, Pattern Recognit. **26**, 1277 (1993).
[9] M. Roushdy, GVIP J. **6**, 17 (2006).
[10] W. V Nicholson and R. M. Glaeser, J. Struct. Biol. **133**, 90 (2001).
[11] S. Maaß, J. Rojahn, R. Hänsch, and M. Kraume, Comput. Chem. Eng. **45**, 27 (2012).
[12] S.-J. Lee and G.-B. Kim, J. Appl. Phys. **94**, 3620 (2003).
[13] E. Rosten and T. Drummond, in *Comput. Vision–ECCV 2006* (Springer, 2006), pp. 430–443.
[14] E. Rosten, R. Porter, and T. Drummond, IEEE Trans. Pattern Anal. Mach. Intell. **32**, 105 (2010).
[15] L. H. Thai, N. D. T. Nguyen, and T. S. Hai, 6 (2011).
[16] Chunmei Liu, Chunheng Wang, and Ruwei Dai, in *Eighth Int. Conf. Doc. Anal. Recognit.* (IEEE, 2005), pp. 610–614 Vol. 2.
[17] D. W. Shattuck, S. R. Sandor-Leahy, K. A. Schaper, D. A. Rottenberg, and R. M. Leahy, Neuroimage **13**, 856 (2001).
[18] D. R. Martin, C. C. Fowlkes, and J. Malik, IEEE Trans. Pattern Anal. Mach. Intell. **26**, 530 (2004).
[19] I. Sobel and G. Feldman, A Talk Stanford Artif. Proj. 271 (1968).
[20] B. Jähne, H. Scharr, and S. Körkel, Handb. Comput. Vis. Appl. **2**, 125 (1999).