Block Processing And Edge Detection For A Dicom Image

D. Nagarajan§, Abitha Gladis N. K#, Nagarajan. V#

§Bolster Trchnologies, chennai-600083, dnrmsu2002@yahoo. com India
#Department of Mathematics, S. T. Hindu College, Nagercoil-629002.
sthcrajan@gmail. com, India

Abstract

The blockprocess function is one of the solutions to processing the large files. It is specify an image, a block size, and a function handle. Edge detection is the initial step in object recognitions. Edge provides a number of derivative estimators, each of which implements one of the definitions. The aim of this paper is load a dicom image thorough block processing techniques by canny, sobel and fuzzy logic edge detection methods. Comparison of canny, Sobel and fuzzy logic image edge detection techniques.

Key: canny, sobel, fuzzy logic

Introduction

Block processing to perform edge detection on a magnetic resonance image. When working with large images, normal image processing techniques can sometimes break down. The images can either be too large to load into memory, or else they can be loaded into memory but then be too large to process. blockprocess then divides the input image into blocks of the specified size, processes them using the function handle one block at a time, and then assembles the results into an output image. blockprocess returns the output to memory or to a new file on disk[1], [3]

Edge detection is frequently the first step in recovering from images. An edge in an image is a significant local change in the image intensity. It is related with the first derivative of the image intensity. The gradient is a measure of change in a function. By analogy significant changes in the gray values in an image. The gradient is the two dimensional equivalent of the first derivative and is defined as the vector[2].
Gradient computation based on intensity values of only two points are susceptible to noise filtering is to improve the performance of edge detection operators. Gaussian edge operator corresponding to smoothing an image with Gaussian function and then computing the gradient. He canny edge detector is the first derivative of a Gaussian and closely approximates the operator that optimizes the product of signal to noise ratio and localization and the smoothed data.

\[
S[I, j] = G[I, j; \Theta] \ I[I, j]
\]

Where \( \Theta \) is the spread of the Gaussian and contrast the degree of smoothing is the first difference approximation to produce two arrays \( P[I, j] \) and \( Q[I, j] \) for \( x \) and \( y \) partial derivatives

\[
P[I, j] = \frac{1}{2} \left( S[I, J+1] - S[I, j] + S[i+1, j+1] - S[i+1, j+1] \right)
\]

\[
Q[I, j] = \frac{1}{2} \left( S[I, J] - S[I, j] + S[i, j+1] - S[i, j+1] \right)
\]

If the image gradient is large then the magnitude have large values. The broad ridges in the magnitude array must be thinned so that only the magnitude at the points of greatest local change remain. Non maxima suppression thins the ridges of gradient magnitude in \( M[I, j] \) by suppressing all values.

\[
P[I, j] = \text{sector}(\Theta[I, j])
\]

\[
N[I, j] = \text{nms}(M[I, j], P[I, j])
\]

\( N[I, j] \) is the amount of contrast at a step change in the image intensity \( N[I, j] \) will contain false edge fragments caused by fine texture and noise to reduce the number of false edge fragment in the nms gradient magnitude is to apply a threshold to \( N[I, j] \) below the threshold values are changed to 0. The threshold to the maximum suppressed magnitude is an array of the edges detected in the image\( I[I, j] \). To estimate the location of an edge is to better than the spacing between pixel[4], [5].
Data base

The skull magnetic resonance image is 677 x 598 pixels and 72 resolution pixel per inch. Using MATLAB PSNR of the image is 8.893 and L2 norm ratio is 56.74%, mean is 65.85, standard deviation of the image is 91.58.

Canny edge detection

Canny edge detection block finds edges by looking for the local maxima of the gradient of the input image. It calculates the gradient using the derivative of the Gaussian filter. The Canny method uses two thresholds to detect strong and weak edges.

\[ I = \text{dicomread('a1.dcm');} \]
\[ \text{normal_edges = edge(I, 'canny');} \]
\[ \text{imshow(normal_edges);} \]
\[ \text{title('canny Edge Detection');} \]

The image is 677 x 598 pixels and 72 resolution pixel per inch.

The blockproc function has built-in support for dicom images, so you do not have to read the file completely into memory using imread. Instead, call the function using the string filename as input. blockproc reads in one block at a time, making this workflow ideal for very large images.

\[ \text{block_size = [50 50];} \]
\[ \text{I2 = blockproc(I, [50 50], edgeFun);} \]
\[ \text{figure;} \]
\[ \text{imshow(I2);} \]
\[ \text{title('Block Processing-canny');} \]
Sobel edge detection
q=dicomread('a1.dcm');
normal_edges = edge(q, 'sobel');
imshow(normal_edges);
title('sobel Edge Detection');

The image is 677 x 598 pixels and 72 resolution pixel per inch
block_size = [50 50];
q2 = blockproc(q, [50 50], edgeFun);
>> figure;
imshow(q2)
title('Block Processing-sobel');
Fuzzy edge detection
An edge is a boundary between two uniform regions. You can detect an edge by comparing the intensity of neighboring pixels. However, because uniform regions are not crisply defined, small intensity differences between two neighboring pixels do not always represent an edge. Instead, the intensity difference might represent a shading effect.

```matlab
Irgb = dicomread('a1.dcm');
I = double(gray);
>> I = double(Irgb);
>> classType = class(Irgb);
scalingFactor = double(intmax(classType));
I = I/scalingFactor;
>> Gx = [-1 1];
Gy = Gx';
Ix = conv2(I, Gx, 'same');
Iy = conv2(I, Gy, 'same');
figure; image(Ix, 'CDataMapping', 'scaled'); colormap('gray'); title('Ix');
figure; image(Iy, 'CDataMapping', 'scaled'); colormap('gray'); title('Iy');
>> edgeFIS = newfis('edgeDetection');
>> edgeFIS = addvar(edgeFIS, 'input', 'Ix', [-1 1]);
edgeFIS = addvar(edgeFIS, 'input', 'Iy', [-1 1]);
>> sx = 0.1; sy = 0.1;
edgeFIS = addmf(edgeFIS, 'input', 1, 'zero', 'gaussmf', [sx 0]);
edgeFIS = addmf(edgeFIS, 'input', 2, 'zero', 'gaussmf', [sy 0]);
>> edgeFIS = addvar(edgeFIS, 'output', 'Iout', [0 1]);
>> wa = 0.1; wb = 1; wc = 1;
ba = 0; bb = 0; bc = 0.7;
edgeFIS = addmf(edgeFIS, 'output', 1, 'white', 'trimf', [wa wb wc]);
edgeFIS = addmf(edgeFIS, 'output', 1, 'black', 'trimf', [ba bb bc]);
>> figure
subplot(2, 2, 1); plotmf(edgeFIS, 'input', 1); title('Ix');
subplot(2, 2, 2); plotmf(edgeFIS, 'input', 2); title('Iy');
subplot(2, 2, [3 4]); plotmf(edgeFIS, 'output', 1); title('Iout')
```
>> r1 = 'If Ix is zero and Iy is zero then Iout is white';
r2 = 'If Ix is not zero or Iy is not zero then Iout is black';
r = char(r1, r2);
edgeFIS = parsrule(edgeFIS, r);
showrule(edgeFIS)
ans =
1. If (Ix is zero) and (Iy is zero) then (Iout is white) (1)
2. If (Ix is not zero) or (Iy is not zero) then (Iout is black) (1)
>> Ieval = zeros(size(I));% Preallocate the output matrix
for ii = 1:size(I, 1)
Ieval(ii, :) = evalfis([[(Ix(ii, :));(Iy(ii, :))]], edgeFIS);
end
>> figure; image(I, 'CDataMapping', 'scaled'); colormap('gray');
title('Original Grayscale Image')
figure; image(Ieval, 'CDataMapping', 'scaled'); colormap('gray');
title('Edge Detection Using Fuzzy Logic')

The image is 561 x 490 pixels and 72 resolution pixel per inch
block_size = [50 50];
w2 = blockproc(I, [50 50], edgeFun);
>> figure;
imshow(w2)
detected the edges in an image using a FIS, comparing the gradient of every pixel in the x and y directions. If the gradient for a pixel is not zero, then the pixel belongs to an edge (black). You defined the gradient as zero using Gaussian membership functions for your FIS inputs.

**Table ;1**

<table>
<thead>
<tr>
<th>Operator</th>
<th>After edge detection</th>
<th>advantages</th>
<th>disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canny</td>
<td>677 X 598</td>
<td>Better detection</td>
<td>Time consuming</td>
</tr>
<tr>
<td>Sobel</td>
<td>677 x 598</td>
<td>Normal Detection of edges</td>
<td>Noise</td>
</tr>
<tr>
<td>Fuzzy logic</td>
<td>561 x 420</td>
<td>Storage of the output memory is quite small</td>
<td>Less accurate</td>
</tr>
</tbody>
</table>

**Conclusion**

The blockprocess function is one of the solution to processing the large files. Using blockprocess, specify an image, a block size, and a function handle. block process only supports zero-padding along the image boundaries. Edge detection is the initial step in object recognitions, The canny edge detector is one of the best edge detector it is heavily on the adjustable parameter. Advantage of canny edge detector is to localization and response, improving signal to noise ratio, better detection specially in noise condition disadvantage of this canny is time consuming. Sobel edge detector is normal one comparing to other detectors. Fuzzy logic is very useful to reduce image memory after edge detection. From the above results in dicom images canny edge detector give the better solution for edge deduction and Fuzzy logic edge detector give the best solution for memory in savingmagnetic resonance images.
References


