An Edge Detection Algorithm Using Statistical Nonstationarity

John DiCecco
Department of Electrical and Computer Engineering, University of Rhode Island, Kingston, RI 02881, USA

Abstract- **Images are often nonstationary. In certain cases, this nonstationarity is used by operations to distinguish regions of interest in an image. Edge detection schemes, such as Robert's Cross or the Sobel operator, rely on the difference of weighted values, i.e. the difference in means, to distinguish one homogenous region from another. In this way an edge can be identified and marked. There are instances, however, when this type of edge detection can yield inaccurate results such as when the regions are changing with a slow gradient or when one mean calculation is offset by a single (or several) pixel value(s). To compensate for this possibility, a similar algorithm using variance instead of mean as the statistic to be analyzed is proposed.**

***Index Terms*— Image processing, nonstationary, edge detection**

I. Introduction

An image can be arbitrarily defined as nonstationary if its mean or variance changes by a threshold amount over the extent of the image. That is, a stationary image is only defined as one where the statistics do not change in space, regardless of any arbitrary threshold [1]. An algorithm has been developed which accurately determines when the variance in a predetermined subimage of the original image shows a statistically significant change. This change will be determined using analysis of variance, or the so called ANOVA. This analysis will focus on techniques involving the "one-way" model.

II. Methods

The algorithm begins by defining a 5 by 5 square pixel region in an image. In a similar manner to the Sobel operator, the square is divided into horizontal edge and vertical edge detectors (Fig.1). The sample variance is approximated by (1), where n is the number of pixels in each sample, k is the number of groups, $\bar{X}$ is the sample mean and X_i is the pixel value [2].

$$s^2 = \frac{n\sum_{i=1}^{n}(X_i - \bar{X})^2}{k-1} \tag{1}$$

Once this inter sample value is calculated, the intra sample value (pooled variance) is calculated in a similar fashion by (2)

$$s^2{}_p = \frac{(n_1 - 1)s^2{}_1 + (n_2 - 1)s^2{}_2}{(n_1 - 1) + (n_2 - 1)} \tag{2}$$

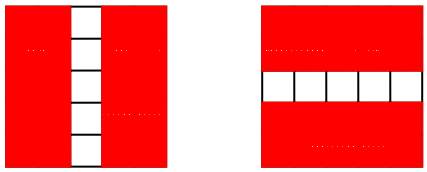

Fig. 1. A 5 by 5 pixel mask establishes a sample space to calculate the variance within the mask for vertical edge detection (l) and horizontal edge detection (r). Shaded regions are the sample spaces.

III. Calculations

In order to provide justification for the assertion of any difference in the statistics, the F distribution, which describes the distribution of estimates of variance, is used [2]. In simple terms, it is the ratio of the sample variance over the pooled variance, or

$$F = \frac{(s^2)}{(s^2{}_p)} \tag{3}$$

The shape of the distribution, like many other distributions, is dependent on the degrees of freedom of the sample space. There are, however, two distinct calculations; one for the numerator and one for the denominator. The numerator calculation is simply the number of sample spaces k, minus one, or (k-1). The denominator is similarly calculated as the total number of samples in the sample space N, minus one, or (N-1). This is referred to as an F((k-1),(N-1)) distribution. For this analysis, any F value above 4.413 indicates a statistically significant difference in variance [2].

IV. Results

A result of running the ANOVA algorithm is shown in Fig. 2. The top image is the original image. The cluster of images below the original are lexicographically the ANOVA algorithm, the Sobel operator, Robert's cross, and the Canny iterative method of edge detection. It is clear that the ANOVA algorithm is more effective in identifying edges than either the Sobel operator or Robert's cross, and its performance is comparable to that of Canny.

While these results are promising, they are somewhat sensitive to the type of image and the threshold for the pooled variance. Additionally, the algorithm returns a raw image with thicker than desired edges. This is a simple matter to

resolve by thinning the edges [3]. Nonetheless, this is an issue that must be addressed before assertions can be made regarding its usefulness.

Fig. 3 illustrates the issue of sensitivity to threshold. In the bottom image, the threshold is set arbitrarily high and very few of the differences between the variances are able to cross this threshold. This results in very poor edge detection. By decreasing the threshold by a factor of 10, the edges in the image are more appropriately resolved (top).

The results suggest an alternate method of edge detection. The assertion is not that it is a better method of edge detection but rather an alternate method that has very appealing results for specific applications. These applications include rapidly changing images and textural images, i.e. images with similar means but differing variances in sub regions of the image.

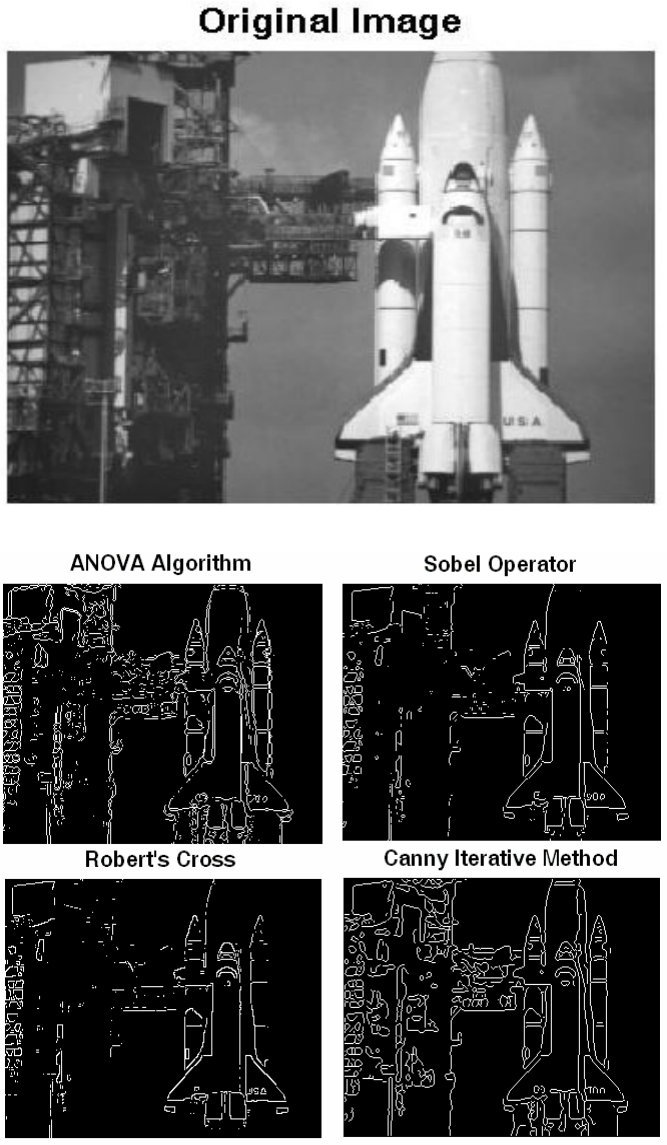

Fig. 2. The top image is the original photograph of the space shuttle. Lexicographically from the top left of the lower cluster is the ANOVA edge detection algorithm, the Sobel operator, Robert's Cross and the Canny iterative method. Notice the fine details in the ANOVA edge detection that are lost in Sobel and Robert's. While Canny performs better in some areas of the image, the ANOVA algorithm is comparable and in some regions provides better resolution, such as the scaffold region in left portion of the image.

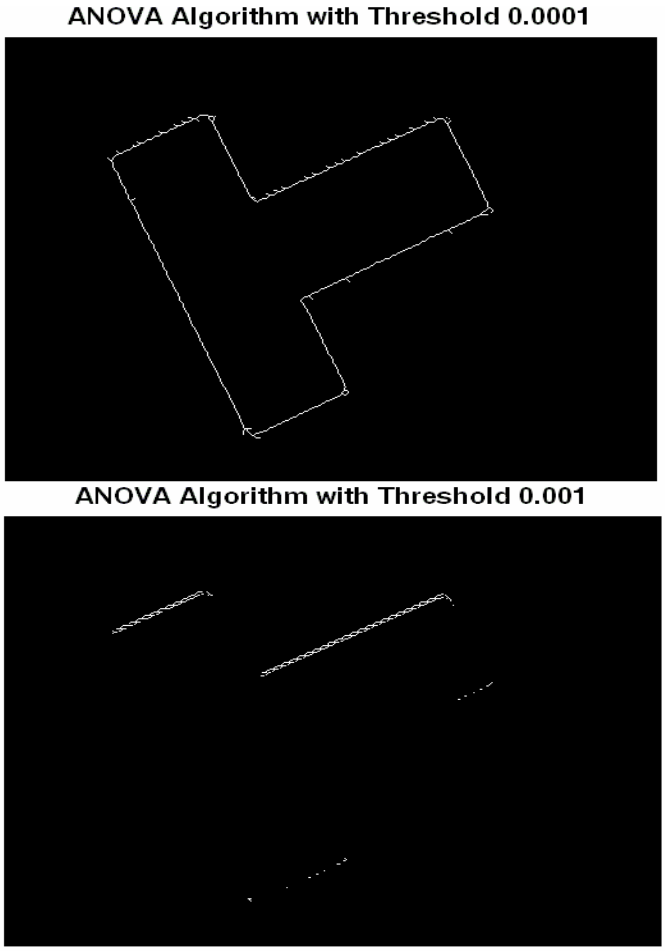

Fig. 3. The top image is the outline of a noisy t with a low threshold. With just a 10 fold increase in the threshold, the algorithm cannot accurately resolve the edges. The performance is clearly dependent on this threshold value.

V. Discussion

An algorithm for edge detection using statistical nonstationarity has been proposed. The performance of the algorithm has been shown to perform as effectively as several of the most popular methods.

There are two important issues with the algorithm in its current form. One is the inefficiency in the code which slows the algorithm down for larger images. Second is the threshold for the pooled variance. If the pooled variance is too small, it will generate very large F values and skew the distribution. This parameter has to be adjusted for different types of images. Future works on this algorithm will attempt to correct this automatically.

References

[1] Leon-Garcia, Alberto. *Probability and Random Processes for the Electrical Engineer*. Addison-Wesley Publishing, 1994.

[2] Bahn, Anita K. *Basic Medical Statistics*. Grune & Stratton, Inc. 1972

[3] Gonzalez, R, et al. *Digital Image Processing Using Matlab*. Pearson Education Ltd. 2004

```
clear all
%definitions -
%mu1 - average of the first horizontal sample space
%mu2 - average of the second horizontal sample space
%mu3 - average of the first vertical sample space
%mu4 - average of the second horizontal sample space
%mu_av - the average of the averages
%varsam - the variance between sample spaces
%samvar - the variance within the sample space
%samvarpool - the pooled variance
%F - the F distribution calculation

d = imread('shuttle.gif');
%d = imread('ele585_1.gif');
%d = imread('ele585_2.gif');
%d = imread('grey_mon.jpg');
%d = imread('T_shaped_object.gif');
%d = imread('testim6.jpg');
d = im2double(d);
%resize is added as a temporary solution to large images
%d = imresize(d,.5);
[m,n] = size(d);
sampoolthresh = 0.001;
Fnum = 10;
for i = 3:m-2
    for k = 3:n-2

        %edge detection using variance in the horizontal detection
        mu1(i-2,k-2) = (d(i-2,k-2) + d(i-2,k-1) + d(i-2,k) + d(i-2,k+1) + d(i-2,k+2) + d↙
(i-1,k-2) +...
            d(i-1,k-1) + d(i-1,k) + d(i-1,k+1) + d(i-1,k+2))/10;
        mu2(i-2,k-2) = (d(i+1,k-2) + d(i+1,k-1) + d(i+1,k) + ...
            d(i+1,k+1) + d(i+1,k+2) + d(i+2,k-2) + d(i+2,k-1) + d(i+2,k) + d(i+2,k+1) +...
            d(i+2,k+2))/10;
        mu_av(i-2,k-2) = (mu1(i-2,k-2)+mu2(i-2,k-2))/2;
        varsam(i-2,k-2) = 10*((mu1(i-2,k-2)-mu_av(i-2,k-2))^2 + (mu2(i-2,k-2)-mu_av(i-2,k-↙
2))^2);
        samvar1(i-2,k-2) = ((d(i-2,k-2) - mu1(i-2,k-2))^2 + (d(i-2,k-1) - mu1(i-2,k-2))^2↙
+ ...
            (d(i-2,k) - mu1(i-2,k-2))^2 + (d(i-2,k+1) - mu1(i-2,k-2))^2 + ...
            (d(i-2,k+2) - mu1(i-2,k-2))^2 + (d(i-1,k-2) - mu1(i-2,k-2))^2 + ...
            (d(i-1,k-1) - mu1(i-2,k-2))^2 + (d(i-1,k) - mu1(i-2,k-2))^2 + ...
            (d(i-1,k+1) - mu1(i-2,k-2))^2 + (d(i-1,k+2) - mu1(i-2,k-2))^2)/9;
        samvar2(i-2,k-2) = ((d(i+1,k-2) - mu2(i-2,k-2))^2 + (d(i+1,k-1) - mu2(i-2,k-2))^2↙
+ ...
            (d(i+1,k) - mu2(i-2,k-2))^2 + (d(i+1,k+1) - mu2(i-2,k-2))^2 + ...
            (d(i+1,k+2) - mu2(i-2,k-2))^2 + (d(i+2,k-2) - mu2(i-2,k-2))^2 + ...
            (d(i+2,k-1) - mu2(i-2,k-2))^2 + (d(i+2,k) - mu2(i-2,k-2))^2 + ...
            (d(i+2,k+1) - mu2(i-2,k-2))^2  + (d(i+2,k+2) - mu2(i-2,k-2))^2)/9;

        %edge detection using variance in the vertical detection
        mu3(i-2,k-2) = (d(i-2,k-2) + d(i-2,k-1) + d(i-1,k-2) + d(i-1,k-1) +...
```

```
            d(i,k-2) + d(i,k-1) + d(i+1,k-2) + d(i+1,k-1) +  d(i+2,k-2) + d(i+2,k-1))/10;
        mu4(i-2,k-2) = (d(i-2,k+1) + d(i-2,k+2) + d(i-1,k+1) + d(i-1,k+2) + ...
            d(i,k+1) + d(i,k+2) + d(i+1,k+1) + d(i+1,k+2) + d(i+2,k+1) + d(i+2,k+2))/10;
        mu_avh(i-2,k-2) = (mu3(i-2,k-2)+mu4(i-2,k-2))/2;
        varsamh(i-2,k-2) = 10*((mu3(i-2,k-2)-mu_avh(i-2,k-2))^2 + (mu4(i-2,k-2)-mu_avh(i-2,k-2))^2);
        samvar3(i-2,k-2) = ((d(i-2,k-2) - mu3(i-2,k-2))^2 + (d(i-2,k-1) - mu3(i-2,k-2))^2 + ...
            (d(i-1,k-2) - mu3(i-2,k-2))^2 + (d(i-1,k-1) - mu3(i-2,k-2))^2 +...
            (d(i,k-2) - mu3(i-2,k-2))^2 + (d(i,k-1) - mu3(i-2,k-2))^2 + ...
            (d(i+1,k-2) - mu3(i-2,k-2))^2 + (d(i+1,k-1) - mu3(i-2,k-2))^2 + ...
            (d(i+2,k-2) - mu3(i-2,k-2))^2 + (d(i+2,k-1) - mu3(i-2,k-2))^2)/9;
        samvar4(i-2,k-2) = ((d(i-2,k+1) - mu4(i-2,k-2))^2 + (d(i-2,k+2) - mu4(i-2,k-2))^2 + ...
            (d(i-1,k+1) - mu4(i-2,k-2))^2 + (d(i-1,k+2) - mu4(i-2,k-2))^2 +...
            (d(i,k+1) - mu4(i-2,k-2))^2 + (d(i,k+2)  - mu4(i-2,k-2))^2 + ...
            (d(i+1,k+1) - mu4(i-2,k-2))^2 + (d(i+1,k+2) - mu4(i-2,k-2))^2 + ...
            (d(i+2,k+1)- mu4(i-2,k-2))^2 + (d(i+2,k+2) - mu4(i-2,k-2))^2)/9;

        %check to see if the horizontal pooled variance is too small or close to zero
        samvarpool(i-2,k-2) = (9*samvar1(i-2,k-2) + 9*samvar2(i-2,k-2))/(2*9);
        %if its large enough ...
        if samvarpool(i-2,k-2)>sampoolthresh
           F(i-2,k-2) = varsam(i-2,k-2)/samvarpool(i-2,k-2);
        else
           %if not, the variance was small so set the F number to zero
           F(i-2,k-2) = 0;
        end
        %check to see if the vertical pooled variance is too small or close to zero
        samvarpoolh(i-2,k-2) = (9*samvar3(i-2,k-2) + 9*samvar4(i-2,k-2))/(2*9);
        %if its large enough ...
        if samvarpoolh(i-2,k-2)>sampoolthresh
           Fh(i-2,k-2) = varsamh(i-2,k-2)/samvarpoolh(i-2,k-2);
        else
           %if not, the variance was small so set the F number to zero
           Fh(i-2,k-2) = 0;
        end


    end
end
[m1,n1]=size(samvar1);
%create holding image
hol = zeros(m1,n1);
ind1 = find(F>Fnum);
ind2 = find(Fh>Fnum);
hol(ind1)=1;
hol(ind2)=1;
%thin out the edge
hol = bwmorph(hol,'thin',10);
%make egde image the same size as the original
final = zeros(m,n);
```

```
final(3:m-2,3:n-2) = hol;
% ed1 = edge(d,'sobel');
% ed2 = edge(d,'roberts');
% ed3 = edge(d,'canny');
imshow(final),title(['ANOVA Algorithm with Threshold ',num2str(sampoolthresh)],'FontSize',↙
16,'FontWeight','bold')
% imshow(d),title('Original Image','FontSize',16,'FontWeight','bold')
%figure,imshow(ed1),figure,imshow(ed2),figure,imshow(ed3)
```