Real-time Linear Projection Speed Sign Detection System in Low Light Conditions

Tam P. Cao            Darrell M. Elton           John Devlin
Department of Electronics Engineering, Latrobe University, Australia
t.cao@latrobe.edu.au  d.elton@latrobe.edu.au  j.devlin@latrobe.edu.au

Abstract

In this paper, a real-time algorithm for detecting speed signs in low light conditions is presented. This application oriented algorithm starts with performing the popular Difference of Gaussian (DoG) filter as a pre-processing step. After that pixels are thresholded and classified into predefined classes. The paper introduce an efficient voting method called linear projection, in which each pixel that belongs to any predefined classes continuously votes on an incremental image along its gradient vector to identify centre of the speed sign candidates. This algorithm is fast, reliable, suitable for real-time hardware (FPGA) implementation. A Matlab simulation model is built to verify the algorithm’s operation in nighttime driving conditions. A FPGA initial prototype of this algorithm has been implemented and showed promising results.

1. Introduction

Vision-based systems that are able to detect and classify speed signs have been studied for a long time, back in 1980s [1]. Many techniques have been explored including: Artificial Neural Network (ANN)[2],[3]; Support Vector Machine (SVM)[4]; gradient based [5],[6],[7]; geometric labeling [8] and signature (FFT) matching [9]. Most of the algorithms using those techniques work well, but in a limited range of lighting conditions, normally in daytime.

A popular gradient-based radial symmetry speed sign detection algorithm is proposed in [5]. In brief, the algorithm identifies the center of the speed sign circle in the image by a voting mechanism. A pixel will vote for 2 potential circle’s centers at a distance (potential radius) away from the current pixel’s location along its gradient (one in positive direction and one in negative direction). An associated symmetry image is generated for each radius being considered. The voting is repeated for the next radius in the searching range. Finally, a final symmetry image is generated to identify candidates’ centers with their corresponding radii. The detection rate of 90% was reported and the frame rate was reported to be at 20 frames per second at the frame resolution of 320x240. However, no information concerning performance of this algorithm at nighttime is reported.

Detecting speed signs in nighttime driving conditions is a challenging task, and has not been thoroughly addressed. There are many reasons behind this difficulty, but mostly due the low contrast of objects in low illumination and various kinds of distortions introduced by the camera used. Generally, in low light conditions, camera reduces its shutter speed and/or increases its sensitivity (ISO) number to record images. As the result, the noise level in these images increases significantly. The noise here refers to the graininess of the image due to high ISO number, motion blur due to slow shutter speed, low contrast due to under-illuminated objects, saturation due to direct head lights and colour distortions of the image sensor (if the camera in use is colour one).

An algorithm that attempted to address speed sign detection task in nighttime conditions is presented in [7]. This algorithm is similar to radial symmetry algorithm except that colour images are used instead of grayscale images as in [5]. In this algorithm, the image is first colour thresholded in L*a*b colour space, then the resultant binary image will be processed via voting mechanism as in the radial symmetry algorithm [5]. The L*a*b colour filter used in this algorithm, like many other colour filters, fails to correctly retrieve colour information in nighttime conditions. As the result, the detection performance is significantly reduced.

Infrared cameras were used in [10] to detect human on the road in nighttime conditions. The same technique can not be used for speed sign detection because speed signs are not high temperature objects (“cold” or passive objects).

Lowe [11] has shown the use of DoG in object detection in normal lighting conditions, in which DoG
is used repeatedly over the sampled and re-sampled image to identify the key locations. These key locations, Scale Invariant Feature Transform (SIFT) keys, are used as the starting points of the object-searching task over several scale of the same image.

In this paper, DoG filter is used for processing nighttime recorded grayscale images. The DoG filter is applied on only one image scale (without performing non-maximal suppress) and then the resulting filtered image is further processed with a new detection method called linear projection. The combination of using DoG filter and linear projection results in a fast detection algorithm that both increase detection rate and reduce computation time for images recorded in low light conditions. Our approach is not to define a night vision system, but a system that works on artificial light sources available at night and other low light conditions.

This paper is organized in 5 sections. This introduction is followed by the algorithm description in section II, in which the use of DoG and different steps of linear projection are described. Section III presents the Matlab simulation and test results of the proposed algorithm. In section IV, an incomplete FPGA implementation of this algorithm together with experimental results is discussed. Finally, section V outlines a short conclusion and some future work.

2. Algorithm Description

2.1. Difference of Gaussian (DoG) filter

The DoG filtered image is obtained as the difference between the first and the second order of convolving the input image with the Gaussian smoothing kernel, such as the follow:

\[
IG\text{IG}_G \ast - \ast = \sigma \sigma \sigma \sigma (1)
\]

For edge detection, an image is DoG filtered, then processed with non-maxima suppress to reveal thin edges (1 pixel wide) in the image. However, we use the DoG differently. For our purpose, we keep the output of DoG without performing non-maxima suppress. The DoG filtered image is thresholded to obtain a binary image. The result is an ‘edge-like’ image with thick trails representing objects’ edges (instead of thin edges). An example of the DoG used for this application is shown on Figure 1. The original image shown in Figure 1.a experienced some motion blurred (due to slow shuttle speed of the camera and the car is in motion), the cropped out region of the speed sign shows that top and bottom parts of the circle is blurred. Figure 1.b shows the DoG filtered image after filtering the image with DoG kernel (7x7, \( \sigma = \sqrt{2} \)). Figure 1.c shows the same image edge filtered with Sobel masks as used by radial symmetry algorithm [5].

The motion blur type of edge suggests that using first order edge information in this type of image is not very reliable in this kind of lightning conditions.
There are two factors that affect the thickness of edge trails: mask size and standard deviation. Firstly, the size of the Gaussian mask is chosen so that it is normally larger than half of the width of the speed signs’ ring. Typically the width of the circle’s border is not exceeding 10 pixels; hence the mask size of 5x5, 7x7 or 9x9 is adequate. Larger the mask size, the thicker the trail is and the less details of the image retained. Secondly, the larger the standard deviation is, the thicker the trail in the DoG filtered image and the more distortions to the object’s are introduced because the output of a pixel is affected by far-away pixels, which are not necessary of the same structure. Typically, \( \sigma \) is chosen between 1 and 2. We chose \( \sigma = \sqrt{2} \) as used by Lowe [11].

2.2. Classifying pixels to classes

In this step, the binary image from DoG filtering in section 2.1 is processed. For every pixel in the binary image, we compare each center pixel and its 8 surrounding pixels against pre-defined 3x3 templates or classes. These classes are generated based on the geometry of the circle (i.e. the speed sign’s ring). If there is a match, the center pixel will be assigned a corresponding angle \( \beta \) to represent the gradient vector. Some of the representative classes and corresponding \( \beta \) values are shown in Figure.2.a. Only 5 classes are shown in Figure.2.a, however, the total number of classes used is 20 with the corresponding angle values \( -180^\circ \leq \beta \leq +180^\circ \) are: \( \pm 161.55, \pm 153.44, \pm 135^\circ, \pm 116.57^\circ, \pm 108.45^\circ, \pm 71.55^\circ, \pm 63.43^\circ, \pm 45^\circ, \pm 26.56^\circ \) and \( \pm 18.45^\circ \).

Positive angle is defined as counter clockwise relative to horizontal axis, as marked on Figure.3.a. As describe in the later step, \( \tan \beta \) is used for further processing (not \( \beta \) itself). The equivalent values of \( \tan \beta \) are \( \pm 3, \pm 2, \pm 1, \pm \frac{1}{2} and \pm \frac{1}{3} \) (for all 20 values of \( \beta \) mentioned earlier). Both inner edge and outer edge of the circle are treated in the same way, hence in theory, the total number of points on the circle selected is 40 (20 on the outer edge and 20 on the inner edge); however, quantization errors (pixel is square makes the actual number of points increase. As the result, partial occlusion will not affect the detection stage. We can choose to use of either or both inner and outer edges of the circle. The relationship between pixel classes and the circle’s geometry is illustrated in Figure.2.b (shown 12 of 20 classes on the outer edge).

By sorting pixels into different classes, the proposed algorithm only selects pixels that are likely to belong to the circle. Of course, in this step there are still many pixels selected but do not lie on the circle (noise-pixels), such as pixels on the digits and other objects. The number of noise-pixels depends on the content and sizes of the image being processed. However, vast majority of the pixels in the image are not selected; hence computation requirement is reduced compare to radial symmetry and related algorithms ([5],[7]).

2.3. Linear Projection

We define an eligible pixel is a pixel that belongs to one of the 20 classes defined in previous section, section 2.2. Each eligible pixel will vote for a range of pixels, called affected pixels, along its gradient vector. After the pixel is processed, it can be disregard if desirable. This voting process repeats for all eligible pixels in the image. As shown in Figure.3.a, a point \( P(x_0, y_0) \) (with \( x_0, y_0 \) are coordinate of pixel P) is an eligible pixel on the circle. P’s gradient vector g is defined by the angle \( \beta \) associated with the class of P. All affected pixels voted by P lie on the gradient line \( L(g) \), shown as dash-line on Figure.3.a, and have coordinates \( x, y \) that satisfy the straight line equation (2).

\[ \frac{x - x_0}{ \cos \beta } = \frac{y - y_0}{ \sin \beta } \]

Figure 2. a. Representatives of different pixel classes with corresponding angle values. b. Pixel classes with relationship to the circle’s geometry (not all classes shown).
\[ y = \tan \beta \ast (x + x_0) - y_0 \quad (2) \]

where \( \beta \) is the angle factor associated with the class of \( P \).

After all eligible pixels that locate on the circle are processed; all corresponding gradient lines are expected to intersect at the centre of the circle, shown on Figure 3.b (not showing the speed limit number).

In practice, depending on the camera and lenses used, the circle appears in an image has a limited range of sizes or range of radii. In Figure 3.a, we limit the affected pixels to the highlighted portion of the gradient line \( L(g) \). As shown, \( r_{\text{min}} \) and \( r_{\text{max}} \) is the minimum and maximum of the range considered, referred as the searching range. The true radius \( r \) of the circle in the image being processed is within the searching range \( r_{\text{min}} \) and \( r_{\text{max}} \). Hence, taking into account the size of circle size being targeted, the \( x, y \) coordinates of the affected pixels voted by \( P \) must satisfy (2) and the following condition,

\[ r_{\text{min}} \leq \sqrt{x^2 + y^2} \leq r_{\text{max}} \quad (3) \]

All affected pixels are accumulated onto an incremental image of the same size with the image being processed. For every affected pixel, the corresponding pixel in the incremental image is incremented by 1. After the whole image is processed, this incremental image is used to identify circle’s centre, which is one of the local maxima within the resultant incremental image. Figure 4. shows an example of an image being processed and its corresponding incremental image.

The difference between the proposed voting method, called linear projection, and the voting process in radial symmetry algorithm [5] is that in radial symmetry algorithm, every pixel in the image votes for 2 affected pixels per radius and all pixels are processed again for the next radius being considered. On the other hand, in linear projection method, each eligible pixel (not all pixels in the image are eligible pixels) votes for all affected pixels within the searching range at the same time, as the highlighted portion shown in Figure 3.a. Every eligible pixel is processed once only which save computational overheads. Great majority of the pixels in the image are ineligible to vote and each of these ineligible pixels is examined and ignored (for all radii being considered). Hence, ineligible pixels require less computation.

The combined effects of pixel classification and linear projection offer 2 major advantages over radial symmetry algorithm:

1. Number of pixels in the image being processed is significantly reduced by classifying pixels into classes based on circle’s geometry.
2. Eligible and ineligible pixels are processed efficiently. More processing is used for eligible pixels while minimum processing
power is required for ineligible pixels (ineligible pixels are read from system memory only, no further processing).

As discussed earlier, gradient lines are expected to intersect at the circle’s center. This only true when the speed sign’s ring is a circle. In practice, due to viewing angles, the speed sign’s ring normally appears as an ellipse in the image. In these cases, the gradient lines are expected to intersect at either of the 2 ellipse’s foci. The middle point between these 2 foci will be selected as the representative circle (ellipse) center. Due to the symmetry properties of the circle, rotations have no affect on the process of detecting the circle in the image. However, excessive rotation will cause error in recognizing the number written on the speed sign.

2.4. Verification

After local maxima are detected, a verification process started to examine if the local maxima is truly centre of the circle or just randomly voted by non-circle objects (noise). This is done by performing another linear projection on a limited region within ± rmax pixels around each local maximum that are above a threshold. A radius profile is created for each local maximum. This profile keeps record of what radius contributes to the local maximum, i.e. the radius whose one of the affected pixels is the local maximum. This verification process help to reduce false positive, especially when there is no speed sign in the image. If the local maximum is truly the circle’s centre, its radius profile will indicate a clustered number of radius contributions around the true centre and small number of contribution caused by other radii radius (i.e. majority of the votes received at the local maximum is made up by a small group of radii).

After identifying centres and corresponding radii of speed sign candidates within the image, appropriate portions of the image will be extracted to determine if there is speed limit written on the candidates. This number classification task requires much less computing than the task of detecting candidates.

3. Matlab Simulation Results

3.1. Test Data

A Matlab model implementing linear projection algorithm with the DoG filter has been built and compared to the equivalent Matlab model of radial symmetry algorithm. Test images are extracted from 30 video sequences taken at night at various locations including roads in resident, city and highway areas. Each video sequence contains from 130 to 1049 frames, at the frame resolution of 640x480 pixels. Only representative portion(s) of each video sequence (including non-speed sign portions) are used for the testing purposes. Each video sequence contributes different number of frames (images) to the test set depending on the content. In the test images, we consider a valid speed sign is a speed sign satisfies the following two conditions:

1. A human can comfortably tell what the content of the sign is without reasoning (the inferring process in the human brain).
2. Has object(s) enclosed in a circle and the circle’s radius is between 12 and 40 pixels (inclusive), including prohibition signs enclosed in a red circle such as “no-right turn” sign.

The experimental result is shown on Table.1. The total number of images tested with both algorithms is 735 images (converted to grayscale), in which 395 images contain valid speed sign and 340 images do not contain valid speed sign. An image without speed sign is said to be successful detected if none of the objects appear in the image is considered as a candidate. In contrast, an image with speed sign candidate is said to be successful detected if the candidate is selected by highlighting the relevant area.

The overall performance of the radial symmetry algorithm (tested with night time images described above) is about 48.84% and the performance of linear projection with DoG algorithm is 75.65%. Of the total 395 images with valid candidate(s) tested, 136 (34.43%) are detected by radial symmetry while 327 (82.78%) of them are detected by linear projection with DoG. The reason causing radial symmetry to fail in this test condition is radial symmetry algorithm is highly sensitive to light sources and low contrast of the
sign in the image. Hence whenever the candidate is presented together with a strong light source such as from car’s headlights or street lights, the light source will be selected as the best candidate. Light sources do cause some confusion to the linear projection algorithm but only when no speed sign candidate appears in the image. When the speed sign candidate appears in the image, the DoG will normally make speed sign become a stronger candidate than light sources. The detection rate of both radial symmetry and linear projection will be increase significantly if a classification stage (classifying the number written on the speed sign if any) is integrated. This stage will reject most of the false candidate detected when no speed sign appear in the image.

Radial symmetry algorithm requires an average of 21.56 seconds (radius range from 12 to 40 pixels) to process while linear projection with DoG algorithm requires only 3.93 seconds to process (with the same radius range). The reason that makes the difference is the number of pixels being processed. On average, only 2258 pixels are fully processed by linear projection algorithm. These are eligible pixels that has been qualified by the process describe in section II.B. The number of eligible pixels in the image greatly varies with image content. In contrast, all the pixels (307200) in the image are fully process by the radial symmetry algorithm.

3.3. Discussion

Applying DoG to daytime image is possible; however, in daytime conditions, there are much more eligible pixels in the DoG filtered image. As the result, this will increase the computational requirement for the algorithm. Moreover, in daytime conditions, images are expected to have better contrast and insignificant motion blur distortions hence other filtering method such as colour filter seems to give better result.

4. FPGA Implementation

A Field Programmable Gate Arrays (FPGA) prototype implementation was built to verify the real-time operation of the linear projection algorithm. The hardware platform used in this implementation is the Xilinx ML402 board equipped with the Virtex-4 sx35 FPGA and other peripherals.

The prototype is able to process image at the resolution of 640x425 in which valid speed sign candidate has radius between 12 and 35 pixels (inclusive). Two typical images are selected from each video sequence (from the test set used in section 3) and then resized to the 640x425 resolution. These resized images are DoG filtered on the PC then tested with the hardware model. The test result from processing an image on the FPGA is shown on the onboard LCD display and the monitor attached to the VGA port on the development board. The detection rate of this hardware model is about the same as its Matlab model.

On average, the system needs 1 clock cycle to process each non-eligible pixel. Each eligible pixel requires 498 clock cycles to process, which includes:
--- 15 clock cycles to read the pixel and it neighbours
--- 3 clock cycles for overhead
--- 480 clock cycles to update the incremental image (for radii from 12 to 35 inclusive)

These calculations are based on the 100 MHz system clock available on the development board and the current synchronous SRAM memory interface, which is able to complete 1 read instruction in 5 clock cycles and 1 write in 1 clock cycle (with buffer). The read instructions could be pipelined to reduce read access if required.

In addition to the linear projection model, an initial filtering model is implemented on hardware. This model takes image (at 640x425 resolution) as input, and display it smoothed image on the VGA and the processing time on the LCD. The filtering module estimates the computational cost of the DoG filter process. In our initial implementation, an image is filtered with a 5x5 mask, which is an equivalent to the 5x5 Gaussian mask with $\sigma = \sqrt{2}$. It is expected that the 2nd order of Gaussian filter can be computed easily without increase in computational time due to the separable property of the Gaussian filter [12], i.e.

<table>
<thead>
<tr>
<th>TABLE 1</th>
<th>MATLAB EXPERIMENTAL RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radial Symmetry</td>
<td>Linear Projection</td>
</tr>
<tr>
<td>Total number of images</td>
<td>735</td>
</tr>
<tr>
<td>Total image detected</td>
<td>359</td>
</tr>
<tr>
<td>Images with speed sign</td>
<td>395</td>
</tr>
<tr>
<td>Images with speed sign detected</td>
<td>136</td>
</tr>
<tr>
<td>Eligible pixels per image</td>
<td>307200</td>
</tr>
<tr>
<td>Average processing time</td>
<td>21.56 sec</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>LINEAR PROJECTION FPGA EXPERIMENTAL RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of image tested</td>
<td>60</td>
</tr>
<tr>
<td>Time require for filtering</td>
<td>14.4 ms</td>
</tr>
<tr>
<td>Time require for detection</td>
<td>24 ms</td>
</tr>
<tr>
<td>Average eligible pixels</td>
<td>2779</td>
</tr>
<tr>
<td>Total (expected) processing time</td>
<td>38.4 ms</td>
</tr>
</tbody>
</table>
where $K$ is the DoG filtered image of $I$ as shown in (1).

The average computation requirement for this filtering process is about 1 read instruction (5 clock cycles) per pixel and all pixels are treated in the same way.

The performance of the prototype implementing linear projection and the filtering modules are shown on Table.2. The total time required to process an image is about 38.4 ms or equivalent to 26 frames per seconds (fps). This frame rate can be increase significantly if we use on board block RAM (BRAM) available on the FPGA. BRAM is dual port RAM that can complete 1 read and 1 write instructions in 1 clock cycle. However, lower series of FPGA may not have enough BRAM to support the whole system.

5. Conclusion and future work

In this paper a method for real-time detection of speed sign in nighttime conditions has been proposed. There are 2 major factors that make this algorithm suitable for real-time applications: the Difference of Gaussian operation that overcomes the low contrast and motion blur distortions; the efficiency in processing eligible and ineligible pixels using pixel classification and linear projection. Experimental results from large variation of images have shown a significant improvement over the popular radial symmetry algorithm. A prototype FPGA implementation of the algorithm shows that this algorithm is capable of working in real-time on hardware (FPGA) at the average frame rate of about 26 fps at the frame resolution of 640x425. The implementation of this algorithm can be improved in various ways such as the number classification stage can be included to increase the overall detection rate of the algorithm or the computation time required for the FPGA system can be reduced by increasing read throughput of the SRAM interface or using only FPGA’s onboard block RAM. A completed implementation of the algorithm is expected to be integrated in the near future.

Acknowledgement

The authors wish to acknowledge the supports of Automotive Co-operative Research Center (AutoCRC) Australia, Latrobe University Australia, and Seeing Machines Pty. Ltd, for the work described.

10. References