VLSI REPORT : HUMAN DETECTION BY HISTOGRAM OF ORIENTED GRADIENTS

Christoporus Deo Putratama (13213008)
Kevin Shidqi Prakoso (13213065)
Bramantio Yuwono (13213128)
Date: 27/12/2016
EL4138- VLSI System Engineering
SEEI - ITB

1. INTRODUCTION

Include the human detection as one of image recognition technology. Human detection is a technique used in the digital camera, the automobile automatic brake system and monitoring system. Therefore, this exercise is aimed at performing a hardware design of the operation unit as a theme of "human detection by Histogram of Oriented Gradients", conscious operations and cost calculation algorithm.

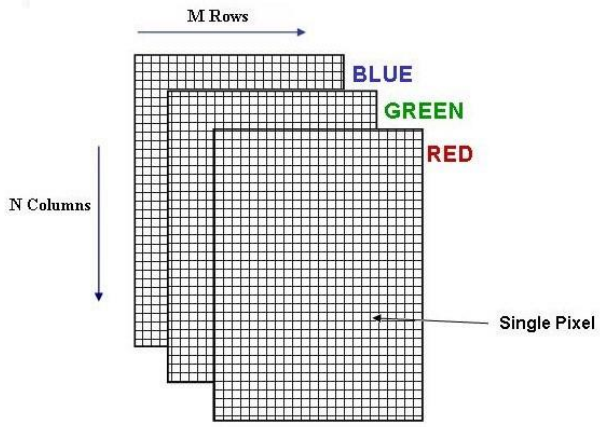
2. SIMULATION RESULT AND ANALYSIS

2.1 GREYSCALING

Human detection on colored images is difficult, so the input image is converted into a greyscale image. A color image has three matrices, consists of R, G, and B matrix, so we must combine each of those matrices into a single matrix by using the following equation.

$$Y = 0.298912 \times R + 0.586611 \times G + 0.114478 \times B$$

With that equation, we get a greyscale matrix that will be used as an input to the human detection system.



Representation of matrices from a color image

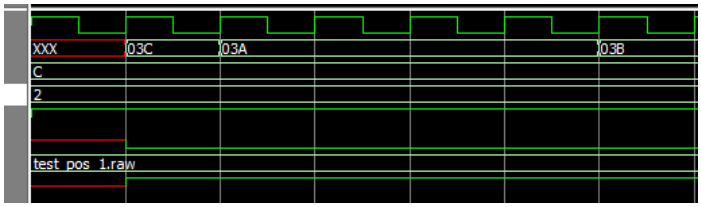
http://archive.cnx.org/resources/089a36ecd6aa8fe1f70b489a6cb1d44950cb5802/graphics1.jpg

The implementation from this greyscaling process is to change all the three matrices into a 24-bit signal. The R matrix will be entered into the input signal from the first bit to the eighth bit. The G matrix will be entered into the ninth bit until to the 16th bit, and the rest were filled by the B matrix. We're using VHDL library called ieee.std_logic_textio to translate a raw image that contain six hexadecimal numbers which represents those matrices. Each matrix were represented by two numbers, with the R matrix in the first two numbers, and then G and B respectively.

A sample of greyscaled image

```
1   003c
2   003a
3   003a
4   003a
5   003a
6   003b
7   003b
8   003a
9   003a
10  003a
11  003a
```

Sample of raw image represented in hex numbers.



Signal created from reading raw file in modelsim.

2.2 RESIZE

Before we scan the image to find objects, the size of image must be recalculated and resized to smaller size so we can detect human in the image faster. First, what we are going to do is to define how many times the image needed to be scaled. We compare the image to a 134 x 70 pixels image and then we can determine the scale from it up to 1.25^n.

After we get the scale number, we will iterate the process of human detection up to n times. With each of iteration, the scaled image will have it's matrix rescaled too. The new matrix's index number are multiplier of the scaler itself.

The implementation of resizing the input image is similar with MATLAB algorithm reference that given in this task, with some change such as in the Scaler module, instead using bilinear interpolation , we use iteration of scaling loop.

```
function pict = Scaler(img, scale)
[rows, cols ] = size(img);
row1 = floor(rows/scale);
col1 = floor(cols/scale);
pict= zeros(row1,col1,'uint8');

    for i=1 : row1
        for j=1 : col1
        a = floor(scale*i);
        b = floor(scale*j);
        pict(i,j) = img(a,b);
        end
    end

end
```

Scaler function that were made with MATLAB

2.3 SCANNING & GET HISTOGRAM

In this section, the input image is scanned to get the gradien from every pixel of the image. The input image will be checked by using a 128 x 64 pixels of virtual rectangle so the scanning process can be emphasized into smaller area and we obtain each gradien faster.

The rectangle is dragged from top left of the image to the right, then followed by next rectangle below and then finally stop at bottom right of the image. We will calculate the gradien in every cell with a block size as an increment inside those rectangles. The conclusion is to calculate the gradien inside a rectangle with a block of 16 x 16 pixels that dragged with an increment of a cell or 8 pixels across the rectangle, the same way as the rectangle does to the image.

While doing scan, we will collect data of the gradien by substracting every matrix of cell with following expression:

$$\begin{cases} f_x(x,y) = L(x+1,y) - L(x-1,y) \\ f_y(x,y) = L(x,y+1) - L(x,y-1) \end{cases}$$

Equation used to calculate gradien

After we get *fx* and *fy*, we can calculate the magnitude an angle of gradien by using following expression:

$$m(x,y) = \sqrt{f_x(x,y)^2 + f_y(x,y)^2}$$

Equation used to calculate magnitude of gradien

$$\theta(x,y) = \tan^{-1}\frac{f_y(x.y)}{f_x(x,y)}$$

Equation used to calculate angle of gradien

But in RTL level, we cannot design a module using multiplier, root and divider easily. So, we will made an approximation to get these values of magnitudes and angles.

```
always@(posedge clk)
begin
  mask = 1;
  factorsum <= val;
  sum = 0;

  while ((val > mask) || (val == mask))
  begin
    if ((val & mask) == mask)
      begin
       sum = sum + factorsum;
      end
    factorsum = factorsum + factorsum;
    mask = mask + mask;
  end

end
```

Verilog Algorithm to approximate a multiplier operation

```
always@(posedge clk)
begin
  m = sqrx + sqry;
  res = 0;
  bit = 1 << 14;

  while (bit > m)
    bit = bit >> 2;

  while (bit != 0)
  begin
    if (m >= res + bit)
    begin
      m = m - (res + bit);
      res = res + (bit << 1);
    end
    else
    begin
      res = res >> 1;
      bit = bit >> 2;
    end
  end
  magnitude <= res;
end
```

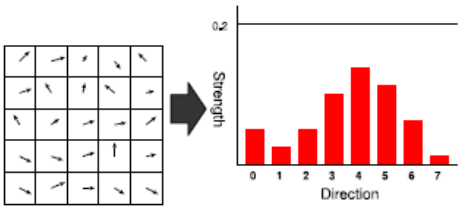
Verilog algorithm to approximate a root operation

On the other hand, the tangent values of the quantization threshold angles Qd(d = 1, 2, . . . , 8), which are described in the next section, can be precomputed since they are constant values. Therefore, we get a quantized gradient orientation by the conditional expression with tan Qd and tan Qd+1 as shown in the equations below.

$$\tan\theta = \frac{\Delta L_y(x,y)}{\Delta L_x(x,y)} \quad (9)$$

$$\tan Q_{d+1} \le \tan\theta < \tan Q_d \ (d = 1, 2, \ldots, 8) \quad (10)$$

Then the gradient strengths m are voted according to the corresponding quantized orientation label for every luminance value in a cell to make the histograms, as shown in the figure below.



Since we quantize the gradient orientation into eight labels, 8-dimension feature vectors are eventually generated. For a (w × h) input image, a total of w/5 × h/5 histograms are obtained since each cell consists of 5×5 luminance values.

Since quantization has been made in the calculation step of a gradient orientation θd as mentioned above, any additional quantization process is not required when histograms are made. We have 320×240 values of luminance gradient, which gives us a cell array with the width of 320/5 = 64 cells and

the height of 240/5 = 48 cells. Since the maximum value of the gradient strength Fi, j is 361×25 = 9,025 for a histogram of a single cell, data size for one cell becomes 14 bits ×8. Therefore, total histograms of luminance gradient for the entire single image are expressed with (14×8)×(64×48) = 344,064 bits.

2.4 NORMALIZATION

The normalization process described in the equation below needs calculation of a square root and division, making compact FPGA implementation difficult.

$$v^m_{k,l} = \frac{f^m_{k,l}}{\sqrt{\|V_{i,j}\|^2 + \varepsilon^2}} \quad (\varepsilon = 1)$$

$$\|V_{i,j}\| = \|F_{i,j}\| + \|F_{i+1,j}\| + \cdots + \|F_{i+2,j+2}\|$$

$$\|F_{i,j}\| = |f^0_{i,j}| + |f^1_{i,j}| + \cdots + |f^7_{i,j}|$$

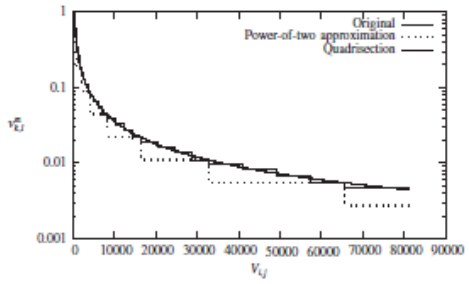
Therefore, we take an approximation approach which expands the method proposed in [4].If the denominator of the equation above (Sqr(||Vi, j||2 + ε2)) is approximated by 2α as 2α−1 <Sqr(||Vi, j||2 + ε2) ≤ 2α, the division for the normalization can be replaced by a shift operation. However, naive approximation to the nearest power-of-two value increases the normalization error. To mitigate the approximation error, we divided the interval between 2α−1 and 2α into four sub-intervals.

The range of 2α−1 to 2α can be divided into four intervals; (2α−1, 2α−1 + 2α−14], (2α−1 + (2^α−1)/4 , 2α−1 + 2 × 2^(α−1)/4], (2α−1 + 2 ×2^(α−1)/4 , 2α−1 + 3 × 2^(α−1)/4], and (2α−1 + 3 × 2^(α−1)/4 , 2α]. Here, ε = 1 and this is significantly smaller than ||Vi, j||2. Therefore, the conditional statements in the normalization equation can be derived when we think:

Sqr(||Vi, j||^2 + ε^2) = ||Vi, j|| .

The figure below shows comparison results of approximation errors for our quadrisection approach and the naive power-of-two approach, in the case of f m k,l = 361. The results show that the normalization errors are effectively reduced with the relatively

simple calculation process.



Since the maximum value of ||Vi, j|| is 81,225, the maximum number of shift operations for the division is 19. Thus, as a fraction part, 19 bits of 0s are appended to the LSB side of f m k,l in advance of shifting, and obtained vm k,l is also expressed with a fixed point arithmetic number with 14 + 19 = 33 bits.

$$
\begin{aligned}
&\textbf{if } (2^{\alpha-1} < \|V_{i,j}\| \le 2^{\alpha-1} + \tfrac{2^{\alpha-1}}{4}) \textbf{ then}\\
&\quad v = \frac{f}{2^{\alpha}} + \frac{f}{2^{\alpha+1}} + \frac{f}{2^{\alpha+2}}\\
&\textbf{else if } (2^{\alpha-1} + \tfrac{2^{\alpha-1}}{4} < \|V_{i,j}\| \le 2^{\alpha-1} + 2\times\tfrac{2^{\alpha-1}}{4}) \textbf{ then}\\
&\quad v = \frac{f}{2^{\alpha}} + \frac{f}{2^{\alpha+1}}\\
&\textbf{else if } (2^{\alpha-1} + 2\times\tfrac{2^{\alpha-1}}{4} < \|V_{i,j}\| \le 2^{\alpha-1} + 3\times\tfrac{2^{\alpha-1}}{4}) \textbf{ then}\\
&\quad v = \frac{f}{2^{\alpha}} + \frac{f}{2^{\alpha+2}}\\
&\textbf{else if } (2^{\alpha-1} + 3\times\tfrac{2^{\alpha-1}}{4} < \|V_{i,j}\| \le 2^{\alpha}) \textbf{ then}\\
&\quad v = \frac{f}{2^{\alpha}}\\
&\textbf{end if}
\end{aligned}
$$

2.5 SUPPORT VECTOR MACHINE

We can use a support vector machine (SVM) when your data has exactly two classes. An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class. The *best* hyperplane for an SVM means the one with the largest *margin* between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points.

The *support vectors* are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab. The following figure illustrates these definitions, with + indicating data points of type 1, and – indicating data points of type –1.

The data for training is a set of points (vectors) x_j along with their categories y_j. For some dimension d, the $x_j \in R^d$, and the $y_j = \pm 1$. The equation of a hyperplane is

$f(x)=x'\beta+b=0$

where $\beta \in R^d$ and b is a real number.

The following problem defines the *best* separating hyperplane (i.e., the decision boundary). Find β and b that minimize $||\beta||$ such that for all data points (x_j,y_j),

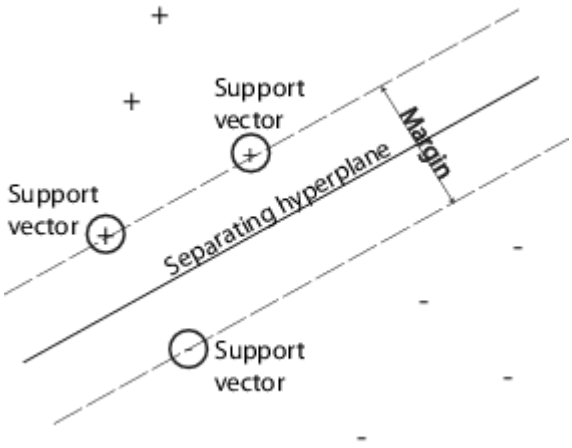
$yjf(xj)\geq 1$.

The support vectors are the x_j on the boundary, those for which yjf(xj)=1.

For mathematical convenience, the problem is usually given as the equivalent problem of minimizing $\leftarrow\beta\leftarrow$. This is a quadratic programming problem. The optimal solution ($\hat{}\beta$,$\hat{}$b) enables classification of a vector z as follows:

class(z)=sign(z'$\hat{}\beta$+$\hat{}$b)=sign($\hat{}$f(z)).

$\hat{}$f(z) is the *classification score* and represents the distance z is from the decision boundary.



As the SVM from the LSI-Contest website still has unpredictable flaws, we decided to train our own SVM classifier using two different datasets, namely MIT pedestrian database and INRIA database. We used Matlab's built-in SVM classifier and constructed a non-linear SVM classifier using a method known as kernel matching. The result of the training process is a new weighing kernel and a new bias kernel to be used as the primary means of human detection.

3. CONCLUSION

The conclusions in this report are:

- We use gradien to compute histogram so we can conclude if there is an object in an image by using SVM.
- We're succeed to approximate multiplier, divider and root operation by using only shifter and iteration.

REFERENCES

[1] http://www.lsi-contest.com/shiyou_1e.html, accessed in 27 December 2016

[2] https://www.mathworks.com/help/fixedpoint/ug/convert-cartesian-to-polar-using-cordic-vectoring-kernel.html, accessed in 27 December 2016Adiono, Trio. *Perancangan Sistem VLSI,* Pusat Mikroelektronika ITB, Bandung, 2012.

[3] Ciletti, M.D. *Advanced Digital Design With the Verilog HDL,* Publishing House of Electonics Industry, Beijing, 2010.