
Algorithm 1 haar_featlist($window_y = 24$, $window_x = 24$, double *rectangle_patterns [$10 \times no_{rectangles}$], $no_{rectangles}$)

```

index_features = 0
index_rectangle = 0
{ $no_{rectangles}$  = the TOTAL number of rectangles regardless of the pattern}
for  $r = 0$ ,  $r < no_{rectangles}$  do
    temp  $\leftarrow$  (id of current pattern) {as they wrote it: rect_param[0 + index_rectangle] and is initially 0}
    if id_current_feature  $\neq$  temp then
        id_current_feature  $\leftarrow$  temp {id_current_feature is initially 0}
        W  $\leftarrow$  (width of current pattern) {as they wrote it: rect_param[1 + index_rectangle]}
        H  $\leftarrow$  (height of current pattern) {as they wrote it: rect_param[2 + index_rectangle]}
        { $24 \times 24$  is the size of the sub-window – so I guess we don't have to slice anything}
        {loop over the image trying to fit the current pattern}
        for  $w = W$ ,  $w < 24+1$ ,  $w = w+W$  do
            for  $h = H$ ,  $h < 24+1$ ,  $h = h+H$  do
                for  $y = 0$ ,  $y+h < 24+1$ ,  $y++$  do
                    for  $x = 0$ ,  $x+w < 24+1$ ,  $x = ++$  do
                        Features[0 + index_features]  $\leftarrow$  id_current_feature
                        {store the top-left coordinates of the pattern in the sub-window}
                        Features[1 + index_features]  $\leftarrow$  x
                        Features[2 + index_features]  $\leftarrow$  y
                        {store the width and height of the pattern in the sub-window}
                        Features[3 + index_features]  $\leftarrow$  w
                        Features[4 + index_features]  $\leftarrow$  h
                        Features[5 + index_features]  $\leftarrow$  index_rectangle
                        {it is stored as an array instead of matrix – because they don't know the future size of it, maybe? – and so the size of one feature is 6}
                        index_features  $\leftarrow$  index_features + 6
                    end for
                end for
            end for
        end for
    end if
    {the rectangles are stored as an array instead of matrix – I can't see the reason here – and so the size of one rectangle is 10 ..look at the top of haar.c for more info}

    index_rectangle  $\leftarrow$  index_rectangle + 10
    return Features
end for

```

Algorithm 2 haar(Image, rectangle_patterns, Features, $subwindow_y=24$, $subwindow_x = 24$, P, standardize)

```

window_size  $\leftarrow subwindow_x \times subwindow_y = 24*24$ 
last  $\leftarrow window\_size-1$ 
if standardize then
    {P = 10 but I do not know what it means, the final features have the size P  $\times$  number_of_features}
    for p=0 to P do
        [IntegralImage]  $\leftarrow$  MakeIntegralImage((Image + index), 24, 24){index is 0 in the beginning}
        for i=0 to window_size do
            temp_Image  $\leftarrow$  Image[i+index] {index is initially equal to 0}
            variation  $\leftarrow$  temp_Image*temp_Image
        end for
        variation  $\leftarrow \frac{variation}{window\_size}$ 
        mean  $\leftarrow \frac{IntegralImage[last]}{window\_size}$ 
        standard_deviation  $\leftarrow \frac{1}{\sqrt{variation-mean^2}}$ 
        for f=0 to sizeof(Features) do
            x  $\leftarrow$  (top coordinate of the feature){Features[1 + index_features]}
            y  $\leftarrow$  (left coordinate of the feature){Features[2 + index_features]}
            w  $\leftarrow$  (width of the feature) {Features[3 + index_features]}
            h  $\leftarrow$  (height of the feature) {Features[4 + index_features]}
            index_rectangle  $\leftarrow$  (index of the corresponding rectangle){Features[5 + index_features]}
            R  $\leftarrow$  (number of rectangles in the pattern){rectangle_patterns[3 + index_rectangle]}
            value  $\leftarrow$  0
            {loop over all rectangles in the pattern of the current feature}
            for r to R do
                x_rectangle  $\leftarrow$  x *  $\frac{w}{width\_of\_current\_pattern}$  * (top coordinate of the current rectangle)
                y_rectangle  $\leftarrow$  y *  $\frac{h}{height\_of\_current\_pattern}$  * (top coordinate of the current rectangle)
                width_rectangle  $\leftarrow$   $\frac{w}{width\_of\_current\_pattern}$  * (width of the current rectangle)
                height_rectangle  $\leftarrow$   $\frac{h}{height\_of\_current\_pattern}$  * (height of the current rectangle)
                value  $\leftarrow$  (weight of current rectangle) * Area(IntegralImage, x_rectangle, y_rectangle, width_rectangle, height_rectangle)
                index_rectangle  $\leftarrow$  index_rectangle + 10
            end for
            final_features[f + index_feature]  $\leftarrow$  value * standard_deviation
        end for
        index  $\leftarrow$  index + window_size
        index_feature  $\leftarrow$  index_feature + 6
    end for
else
    the same as above but without computing the "standard_deviation"
    final_features[f + index_feature]  $\leftarrow$  value
end if
return final_features

```
