Exact Integral Images at Generic Angles for 2D Barcode Detection

Tat-Jun Chin, Hanlin Goh and Ngan-Meng Tan
Computer Vision and Image Understanding Department
Institute for Infocomm Research, Singapore
{*tjchin, hlgoh, nmtan*}*@i2r.a-star.edu.sg*

Abstract

Using integral images for fast computation of sums of rectangular areas is very popular in computer vision. However the method does not extend naturally to rotations at arbitrary angles. We propose a novel solution to elegantly compute integral images at generic angles. Our method is exact in the sense that no approximations are used to derive it and it is vulnerable only to the unavoidable aliasing effects of discretization. Detailed experiments show that our method is more accurate than previously proposed ideas. We also demonstrate its usefulness by detecting 2D barcodes embedded in images.

1. Introduction

The usage of integral images was popularized by the seminal work of Viola and Jones [3] on rapid object detection, where Haar-like features are computed efficiently from sums of rectangular areas. The method is also effective for 2D barcode detection [5, 4] since the target object comprises of black and white rectangles only. Unfortunately *rotated* integral images cannot be easily computed, rendering the method ineffective against non-upright target objects unless rotations are specifically accounted for during training or design.

Algorithms for computing integral images at $45°$ and $26.5°$ (only) later emerged [2, 1], and the quest to extend the method to generic angles continued in [1] where a pair of integral images at $0°$ and $45°/26.5°$ were used to *approximate* Haar-like feature *responses* between those two angles, thus seemingly mitigating the need to explicitly compute rotated integral images.

However accounting for $45°$ and $26.5°$ only is still too restrictive, and there is no theoretical basis for the approximation technique of [1], thus its efficacy cannot be confirmed. We reveal in §4 that the accuracy of [1] is actually very poor. We overcome these shortcomings with a novel solution for computing integral images at arbitrary angles, and using the results to realize sums of rectangles at generic rotations. Our method is based on decomposing a rotated integral image into two complementary "half" integral images; see Fig. 1.

We call our method "exact" to emphasize that we explicitly obtain integral images at generic angles and our sums of rectangles are *not* approximated like in [1]. Realistically and inevitably it is only exact insofar as aliasing effects due to image discretization allow it. An obvious example is the phenomenon where the straight lines defining the edges of a rotated rectangle can only be materialized as jagged lines on a 2D intensity grid.

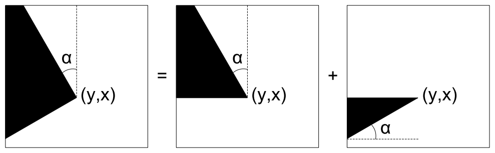


Figure 1. Obtaining rotated integral images as the sum of 2 half integral images.

2. Rectangular Sums from Integral Images

Let image $\mathbf{I}(y,x)$ have the upright integral image $\mathbf{II}^0(y,x)$, where the superscript indicates the angle of rotation which is zero in this case. Thus at a particular coordinate (y_1,x_1) the integral image takes the value

$$\mathbf{II}^0(y_1,x_1) = \sum_{y=1}^{y_1}\sum_{x=1}^{x_1}\mathbf{I}(y,x). \tag{1}$$

We can compute the sum of pixel intensities within rectangles in the image rapidly with just four look-ups to $\mathbf{II}^0(x,y)$ *regardless* of the size of the rectangle. For example, the sum of intensities in the box in Fig. 2(a) is

$$\mathbf{II}^0(y_d,x_d) - \mathbf{II}^0(y_c,x_c) - \mathbf{II}^0(y_b,x_b) + \mathbf{II}^0(y_a,x_a). \tag{2}$$

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

When a Haar-like feature like those in Fig. 2(b) is convolved with an image, the response at a location is obtained as the sum of intensities under the white area minus the sum of intensities under the black area. Clearly this can be implemented efficiently with Eq. (2).

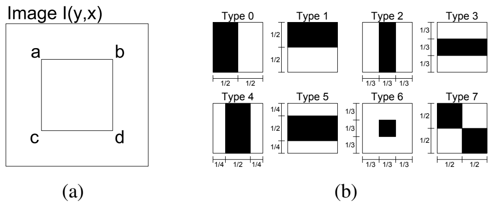


Figure 2. Computing rectangular pixel sums efficiently with integral images.

The power of this approach is also due to having a very efficient procedure to compute $\mathbf{II}^0(y,x)$ quickly:

$$\begin{aligned} s(y,x) &= s(y,x-1)+\mathbf{I}(y,x), && (3)\\ \mathbf{II}^0(y,x) &= s(y,x)+\mathbf{II}^0(y-1,x), && (4) \end{aligned}$$

where $s(y,x)$ is the cumulative row sum at the y-th row and x-th column. Naturally to make rotated integral images attractive we require a fast algorithm to obtain $\mathbf{II}^\alpha(y,x)$ for $\alpha > 0$. The sum of pixels under the black area in the leftmost image of Fig. 1 gives $\mathbf{II}^\alpha(y,x)$.

3. Integral Images at Generic Angles

Our method begins with the notion of decomposing a rotated integral image into two components as in Fig. 1. The two components are then joined to produce the desired rotated integral image. Similar to [2, 1] the required computational effort is maintained at $\mathcal{O}(n)$, where n is the number of pixels in the image.

3.1 Ladder Vectors and Half Integral Images

We produce "ladder vectors" to define the inclinations of lines in a discrete 2D grid. A ladder vector $\mathbf{L}^\alpha$ for angle α is obtained as shown in Table 1. The contents of $\mathbf{L}^\alpha$ represent column displacement values when following the slope of a line downwards.

Ladder vectors play a crucial role in building half integral images. They provide a global (i.e. image wide) reference for adding the appropriate integrated values of previous rows to the current cumulative row sum. The operation to produce half integral images for angle α is

$$\mathbf{hII}^\alpha(y,x) = s(y,x) + \mathbf{hII}^\alpha(y-1, x-\mathbf{L}^\alpha(y)), \quad (5)$$

```
Input: Desired angle $\alpha$, dimensions of input image.
Initialize: $g = \tan(\pi/2-\alpha)$, $c = g$, $f = 0$, $r = 0$,
$\mathbf{L}^\alpha = [\,]$ (null vector).
 1. while $r \neq \max(\text{rows},\text{cols})$ do
 2.     if $c < g/2$
 3.         $c = c + g$
 4.         $f = f + 1$
 5.     else
 6.         $c = c - 1$
 7.         $\mathbf{L}^\alpha \leftarrow f$ (append)
 8.         $r = r + 1$
 9.         $f = 0$
10.     end if
11. end while
Output: Vector $\mathbf{L}^\alpha$ of length $\max(\text{rows},\text{cols})$.
```

Table 1. Algorithm for ladder vector $\mathbf{L}^\alpha$.

where $s(y,x)$ is obtained as in Eq. (3) and the prefix $\mathbf{h}$ indicates half integral image. The possibility of sampling from $\mathbf{L}^\alpha(y)$ columns to the left is the difference between Eqs. (4) and (5). Note that a 0 value is always used in place of an out-of-bounds reference to $\mathbf{hII}^\alpha$. Fig. 3 illustrates $\mathbf{hII}^\alpha$'s built using ladder vectors.

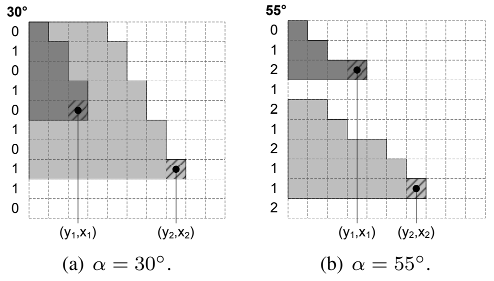


Figure 3. The darker and lighter areas respectively indicate pixels summed by $\mathbf{hII}^\alpha(\mathbf{y_1},\mathbf{x_1})$ and $\mathbf{hII}^\alpha(\mathbf{y_2},\mathbf{x_2})$. In (a) $\mathbf{hII}^\alpha(\mathbf{y_2},\mathbf{x_2})$ encompasses $\mathbf{hII}^\alpha(\mathbf{y_1},\mathbf{x_1})$. The numbers on the left are contents of ladder vectors $\mathbf{L}^{30^\circ}$ and $\mathbf{L}^{55^\circ}$.

It is evident that the 2nd half integral image in Fig. 1 cannot be obtained using Eq. (5), thus rotated integral images cannot be produced as trivially as suggested. We invent an elegant solution, depicted in Fig. 4, to circumvent this difficulty: The pixels summed by $\mathbf{II}^\alpha(y,x)$ belong to areas Q and R. We can produce the following equation with some algebraic manipulation:

$$\text{Q} + \text{R} = (\text{P} + \text{Q} + \text{R} + \text{S}) - (\text{P} + \text{Q}) + \text{Q} - \text{S}. \quad (6)$$

The four components on the RHS can be readily obtained with our results thus far:

$$\begin{aligned} \text{P + Q + R + S} &= \mathbf{II}^0(\text{rows}, x), & (7)\\ \text{P + Q} &= \mathbf{II}^0(y, x), & (8)\\ \text{Q} &= \mathbf{hII}^\alpha(y, x), & (9)\\ \text{S} &= \overline{\mathbf{hII}}^\alpha(y+1, x). & (10) \end{aligned}$$

Item $\overline{\mathbf{hII}}^\alpha$ is the outcome of rotating the image clockwise by $90°$, computing $\mathbf{hII}^\alpha$ and reversing the rotation. In practice this is easily achieved by changing the order of iteration from left-to-right-then-top-to-bottom to bottom-to-top-then-left-to-right for Eqs. (3) and (5). Finally the rotated integral image is obtained as

$$\begin{aligned} \mathbf{II}^\alpha(y, x) = {} & \mathbf{II}^0(rows, x) - \mathbf{II}^0(y, x) + \\ & \mathbf{hII}^\alpha(y, x) - \overline{\mathbf{hII}}^\alpha(y+1, x). \quad (11) \end{aligned}$$

Most applications like object detection [3] and 2D barcode scanning [5, 4] need $\mathbf{II}^0(y, x)$ for upright targets anyway so obtaining it is not an extra burden. We also emphasize that the two-pass method of [2] can be used in conjunction with the ladder vectors but this would require a clumsier method to extend to generic angles.

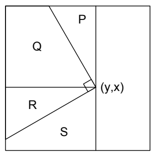


Figure 4. (Q + R) correspond to $\mathbf{II}^\alpha(y, x)$.

3.2 Rectangular Sums at Generic Angles

Care has to be taken when obtaining coordinates of vertexes of rotated rectangular shapes. A corner (y, x) must occur at the intersection of two orthogonal rectangle edges at α and $(\pi/2 + \alpha)$ in a manner that the area subtended by the lines is equivalent to the area summed by $\mathbf{II}^\alpha(y, x)$. This is to ensure that the "loop is closed" when applying Eq. (2) with $\mathbf{II}^\alpha(y, x)$ such that no pixels from undesired areas are inadvertently included.

We rely on the ladder vector to compute valid coordinates for the rectangle corners. Based on Fig. 5, given point (y_a, x_a) as reference a point horizontally displaced by length L_h (e.g. point b) is obtained as

$$M_h = round(L_h \cos\alpha), \quad (12)$$

$$y_b = y_a - \sum_{i=x_a+1}^{x_a+M_h} \mathbf{L}^\alpha(i), \ x_b = x_a + M_h. \quad (13)$$

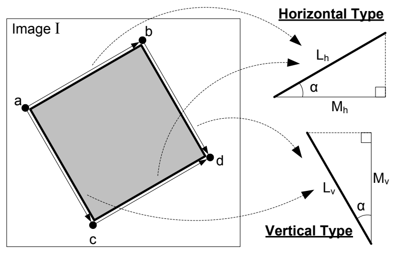


Figure 5. Obtaining reference coordinates of vertexes of rotated rectangles.

On the other hand, a point vertically displaced by length L_v (e.g. point c) is computed as

$$M_v = round(L_v \cos\alpha), \quad (14)$$

$$y_c = y_a + M_v, \ x_c = x_a + \sum_{i=y_a+1}^{y_a+M_v} \mathbf{L}^\alpha(i). \quad (15)$$

Point d can then be obtained from either points c or d. These four corners are then used to lookup $\mathbf{II}^\alpha(y, x)$ to obtain the rectangular sum via Eq. (2). Such a process is also required for defining the Haar-wavelets in Fig. 2(b).

4. Experiments

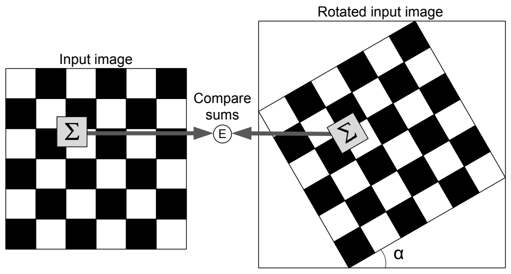
We first examine the accuracy of computing rotated rectangular pixel sums using the proposed method. We use a chessboard image with well-defined horizontal and vertical edges since the rectangular sums derived would be very sensitive to errors in positioning, interpolation and approximation [1], thus giving us a clear indication of accuracy. Ground truth values are first obtained by computing sums of pixels under squares at multiple positions and scales using the *upright* integral image on the *unrotated* input image. The input image is then rotated at angle α and $\mathbf{II}^\alpha(y, x)$ is obtained. At positions and scales corresponding to the upright case, sums of pixels under rotated squares are obtained using the proposed method and compared against the ground truth values. Fig. 6(a) illustrates the experiment.

We follow the error function of [1] to carry out the comparison between ground truth and rotated sums:

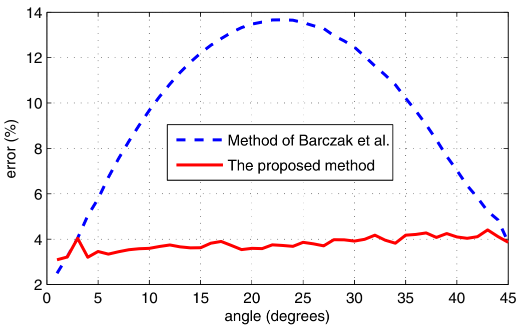
$$E = (|sum_\alpha - sum_0|/max) \times 100\%, \quad (16)$$

where max is the maximum possible pixel sum for the particular scale. We also implemented the method of [1] to approximate the sums of rotated squares by linearly weighting the sums of squares at $0°$ and $45°$ computed using $\mathbf{II}^0(y, x)$ and $\mathbf{II}^{45°}(y, x)$, i.e.

$$sum_\alpha = sum_0 \cdot (45° - \alpha)/45° + sum_{45°} \cdot \alpha/45°. \quad (17)$$

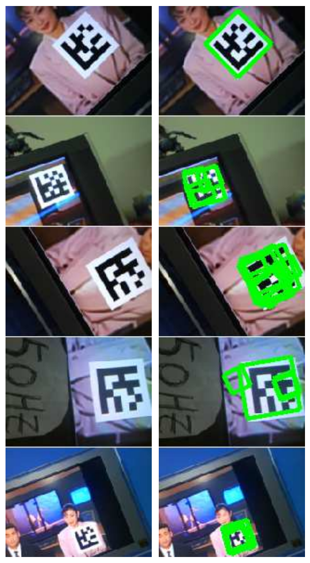


(a) The experimental setting.



(b) Error comparison of two methods.

Figure 6. Experimental results.



(a) Input image. (b) Detections.

Figure 7. 2D barcode detection at generic angles with rotated integral images. Note that input image rotations are not needed.

Fig. 6(b) depicts the error of two methods for $\alpha = [1°, 2°, \ldots, 45°]$ and averaged across scales of $[12, 18, 24, \ldots, 48]$. As was observed in [1], the error of the approximation technique peaks halfway in between 0° and 45° when the underlying sum of pixels is the farthest from sum_0 and $sum_{45°}$. Recall that only $\mathbf{II}^0(y,x)$ and $\mathbf{II}^{45°}(y,x)$ are explicitly computed by [1]. In distinct contrast, the error of our method stays consistently low across the whole range of rotation since we compute rotated integral images at specific angles. In fact, the persistent error stems from discretization effects only such as intensity interpolation of image rotations and jagged edges of the square templates.

We demonstrate a practical application with the detection of *rotated* 2D L-shaped barcodes embedded in input images; see Fig. 7. This is achieved by scanning a template at multiple *scales* and *angles* on an input image. The response of placing a template on a position is obtained by seeking the difference between the average intensity of pixels under the white area (i.e. the barcode boundary) and the average intensity of pixels under the black area (i.e. the L-shape). The pixels under the data portion are ignored. The response will be high if the pattern under the template is the barcode. We compute this response using rotated integral images since only rectangular summing is required, thus avoiding the costly operation of rotating the input image.

5. Conclusion

We propose a method to compute integral images at generic angles which provides an efficient means for summing pixels under rotated rectangular shapes. Since our method obtains the rotated integral images explicitly the rectangular sums obtained are more accurate that approximated values from methods like [1]. Experimental results support this argument. We also demonstrate a practical application by detecting 2D barcodes embedded in cluttered input images.

References

[1] A. L. C. Barczak, M. J. Johnson, and C. H. Messom. Real-time computation of Haar-like features at generic angles for detection algorithms. *Res. Lett. Inf. Math. Sci.*, 9:98–111, 2006.

[2] R. Lienhart and J. Maydt. An extended set of Haar-like features for rapid object detection. In *ICIP*, 2002.

[3] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.

[4] X.-W. Xu, Z.-Y. Wang, Y.-Q. Zhang, and Y.-H. Liang. A skew distortion correction method for 2d bar code images based on vanishing points. In *Int. Conf. on Machine Learning and Cybernetics*, 2007.

[5] H. Yan, Z. Wang, and S. Guo. A method for 2d bar code recognition by using rectangle features to allocate vertexes. In *Int. Workshop on Graphics Recognition*, 2005.