Refined Morphological Methods of Moment Computation

Tomáš Suk *and* Jan Flusser
Institute of Information Theory and Automation of the ASCR
Pod vodárenskou věží 4, 182 08 Praha 8, Czech Republic
E-mail: suk@utia.cas.cz, flusser@utia.cas.cz

Abstract

A new method of moment computation based on decomposition of the object into rectangular blocks is presented. The decomposition is accomplished by means of distance transform. The method is compared with earlier morphological methods, namely with erosion decomposition to squares. All the methods are also compared with direct computation by definition.

1. Introduction

Moments are scalar quantities which have been used to characterize an image and to capture its significant features. From the mathematical point of view, moments are projections of a graylevel function onto a polynomial basis. Functions of moments, insensitive to certain group of transformations, are called *moment invariants*. Moment invariants have become one of the most important and most frequently used tools for object description and recognition. Hundreds of successful applications of moment invariants have been reported in literature (see [3], Chapter 8, for a survey). Even though they suffer from certain intrinsic limitations, moment invariants frequently serve as a reference method for evaluating the performance of other shape descriptors.

Geometric moment of a continuous image $f(x, y)$ is defined as

$$m_{pq}^{(f)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x, y)\,\mathrm{d}x\,\mathrm{d}y, \qquad (1)$$

where $p + q$ is the order of the moment. If the image $f(x, y)$ is a discrete one of the size $M \times N$, then we can estimate its moment as

$$\bar{m}_{pq}^{(f)} = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} x^p y^q f(x, y). \qquad (2)$$

Another way is using higher-order approximation of the integral and/or exact integration of of $x^p y^q$ over rectangular regions [2].

When applying moment invariants in practice, we face two important computational issues – stability and complexity. Stability of moment calculation, particularly avoiding floating-point under/overflow, is ensured by a choice of proper polynomial (usually orthogonal) basis, as has been discussed in many papers. Computing complexity of all invariants is determined by the complexity of moment computation. Having the moments, we can calculate any invariant in $\mathcal{O}(1)$ time. This is why the algorithms for computing image moments (both of binary and graylevel images) have attracted such attention (see [3], Chapter 7, for an overview).

In this paper we deal with *binary* images because of their importance in practical pattern recognition applications. Since any binary object is fully determined by its boundary, which is supposed to consist of much less than $\mathcal{O}(MN)$ pixels (this assumption may not be necessarily true, see a chessboard), there is a big potential for an improvement. The methods for fast computation of the moments of the binary images can be divided into two groups referred as *decomposition methods* and *boundary-based methods*.

While the boundary-based methods mostly employ Green's theorem [4, 6, 10], the decomposition methods use the following idea. Having a binary object Ω, we decompose it into $K \geq 1$ disjoint blocks $B_1, B_2, \ldots, B_K$ such that $\Omega = \bigcup_{k=1}^{K} B_k$. Then

$$m_{pq}^{(\Omega)} = \sum_{k=1}^{K} m_{pq}^{(B_k)}. \qquad (3)$$

If we can calculate the moment of each block in $\mathcal{O}(1)$ time (as we can for rectangular blocks for instance) then the overall complexity of $m_{pq}^{(\Omega)}$ is $\mathcal{O}(K)$. If $K \ll MN$ the speed-up may be significant.

The power of any decomposition method depends on our ability to decompose the object into a small number

1051-4651/10 $26.00 © 2010 IEEE
DOI 10.1109/ICPR.2010.242

of blocks in a reasonable time. Individual decomposition methods differ from one another namely by the decomposition algorithms. Simple algorithms produce relatively high number of components but perform fast, while more sophisticated decomposition methods end up with small number of blocks but require more time. The complexity of the decomposition must be always considered as a part of the whole algorithm. Even if the decomposition is performed only once and can be used for calculation of all moments, the time needed for decomposing the image is often so long that it substantially influences the efficiency of the whole method. Clearly, the efficiency of the decomposition methods becomes apparent only if a high number of moments is required to calculate but this is not the only factor. Good performance can be achieved only on "compact" objects (an example of a compact object is in Fig. 1), while some images cannot be efficiently decomposed by any algorithm (a chessboard is an extreme example).

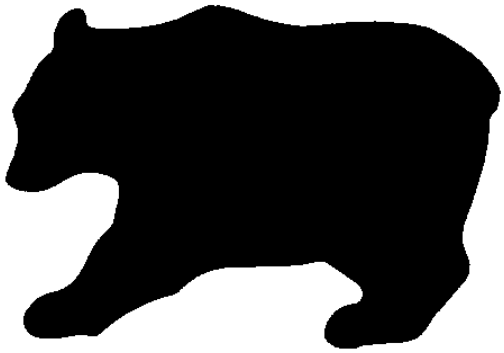


Figure 1. The bear image.

The first method of this kind performed decomposition into individual rows [16]. Later on, it was generalized to row segments [5] and to block of row segments [14]. Hierarchical quadtree decomposition, proposed in [15], yields better results but still far from being optimal. Morphological decomposition into squares [12] results in a low number of blocks but performs much slower than the previous methods.

The goal of this paper is to improve the morphological decomposition in two ways. We replace time-consuming erosion by distance transform and generalize the method in such a way that it may produce not only squares but arbitrary rectangles. Thanks to this, we obtain less final blocks in a shorter time.

2. Morphological Decomposition

The morphological method as was proposed by Sossa in [12] is based on erosions. The erosion is an operation, where a small structural element (here 3×3 square is used) moves over the image and when the whole element lies in the object, then the central pixel of the window is preserved in the object, otherwise it is assigned to the background. So, each erosion shrinks the object by one-pixel boundary layer. We repeat it until the whole object disappears and count the number of erosions s. Then a $(2s-1) \times (2s-1)$ square can be inscribed into the object and it forms one block of the decomposition. This block is removed from the object and the algorithm is repeated until the whole object is decomposed into square blocks.

The pixels of the object before the last disappearing erosion are potential centers of the inscribed square. Theoretically, we can choose one of them randomly, but the corner pixels provide better odds to more compact rest of the object. If we find a square 2×2 among them, a square with size equaling an even number of pixels can be inscribed into the object. If the potential square centers create a line segment (with a single or double-pixel width), then the corresponding squares can be unified into one rectangle.

It is possible, especially in the last steps of the method, that several of the identically sized squares (overlapping as well as non-overlapping) can be inscribed into different places of the object. Of course, it is possible to inscribe and remove one of them, repeat the erosions, inscribe and remove another one etc. A better approach is, after inscribing and removing one of them, to remove the centers of the squares that would overlap this one and search a center of another block of this size without repeating the erosions.

Although this method yields good decompositions of most compact shapes, it is still only sub-optimal even on simple shapes (see Fig. 2)[1].

3. Distance Transform

If the objects is sufficiently compact, i.e. the biggest inscribed square is bigger than a certain minimum size, we can speed up the previous algorithm by using distance transform. In morphological method, we must repeat the erosions s-times for finding $(2s-1) \times (2s-1)$ inscribed square, while the distance transform with a suitable metric can be calculated only once. The distance transform of a binary image is an image, where each pixel of an object shows the distance to the nearest boundary pixel, the background pixels are zero [1].

The distance transform strongly depends on the metric used for the distance measurement. We use a simplified version of the Seaidoun's algorithm [11] for the

[1] Thanks to Mirko Navara from the Center for Machine Perception, Czech Technical University, Prague, for this example.

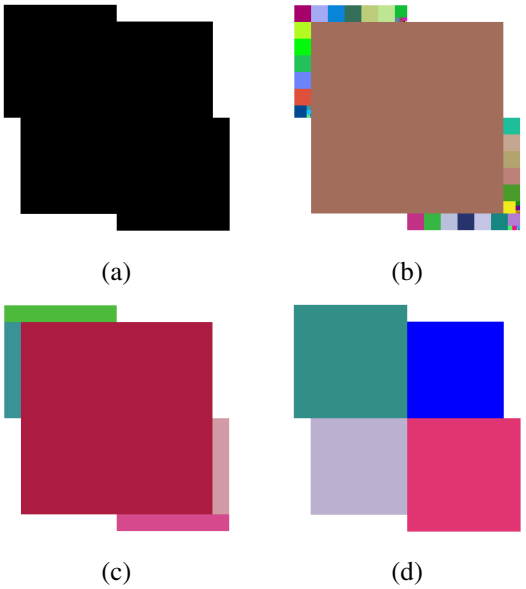


Figure 2. (a) Original binary object, (b) morphological decomposition into squares – 51 blocks, (c) morphological decomposition into rectangles – 5 blocks, and (d) optimal manual decomposition into squares – 4 blocks

chessboard metric

$$d(a,b) = \max\{|a_x - b_x|, |a_y - b_y|\}. \qquad (4)$$

We successively search the image from left, right, top and bottom, count distances from the last boundary pixel and calculate the minimum from the four directions. The result is the distance transform, the maximum of the result equals s for the inscribed square $(2s-1) \times (2s-1)$ and the pixels with this maximum value are possible centers of the inscribed squares.

The computing complexity of this algorithm is $\mathcal{O}(MN)$ in comparison with $\mathcal{O}(sMN)$ of the repeated erosions. However, if s is small (i.e. if the object is not very compact) the distance transform may be slower than the erosion. The distance transform with another metric could be used for decomposition of an object into blocks of different form.

4. Even-sized Squares and Rectangles

In this Section we explain why even-sized and odd-sized square blocks must be treated differently. The original paper [12] did not consider even-sized squares at all, it only worked with odd-sized ones. If we followed this approach here and inscribed an odd-sized square $(2s-1)\times(2s-1)$ whenever an even-sized square $2s \times 2s$ can be inscribed into the object, we would slow down the algorithm substantially. A better approach is first to search small squares 2×2 among the potential centers of the squares and only when there are no such squares, search individual pixels as centers of the odd-sized squares.

Another solution can be using the erosions by a structural element 2×2 instead of 3×3. If we used e.g. a top-left pixel of the element as the "central" pixel, we would obtain top-left pixel of the inscribed square in the last non-zero layer instead of its center. If we needed s erosions for zeroing the object, then the size of the square would be $s \times s$ and there would be no need to differentiate between odd-sized and even-sized squares. We would need a double number of erosions for the zeroing but with a smaller structural element. Slightly longer erosions are partially compensated by an easier searching of the square centers.

If the potential square centers create a line segment (with single-pixel or double-pixel width), then the corresponding squares can be unified into a rectangle. The number of blocks is then significantly less, compare Figs. 2b and 2c. It is a further reduction of the computation time.

Sometimes, the potential square centers create a form, when a rectangle with both even and odd width can be inscribed, see Fig. 3. We can use either an even-sized rectangle $(2s + a - 2) \times 2s$ or a more elongated odd-sized rectangle $(2s + a + b - 2) \times (2s - 1)$. We should choose the bigger one of them. The optimization in this sense is difficult and we use only a sub-optimal algorithm for this decision. If we use the erosions 2×2, this problem cannot occur.

Figure 3. The potential centers of the squares.

5. Experiments

The kernel function $x^p y^q$ of the geometric moments may take very high values for large images and the moments suffer from the loss of precision. To overcome this, we use a system of orthogonal polynomials $\{\tau_p(x)\}$ instead of x^p in the kernel functions. We employ discrete Chebyshev polynomials [7] in our experiments, because we are able to preserve their precision

even for very high orders of the moments. The particular choice of polynomial basis does influence neither the decomposition algorithm itself nor its comparison to other methods.

We took the image of the size 465×465 pixels from Fig. 1 and the chessboard image of the same size and successively computed all discrete Chebyshev moments up to certain upper bound r of both indices. The limit r increased from zero to 464. The moments were computed by a direct calculation from definition, by decomposition to squares via erosions by 3×3 element (E3×3 – S), by decomposition to rectangles via erosions by 2×2 element (E2×2 – R), by decomposition to squares via distance transform (DT – S) and by decomposition to rectangles via distance transform (DT – R). The computation times[2] of the bear image to the limit $r = 200$ are in the graph in Fig. 4. The maximum moment index r is on horizontal axis, the computation time t in seconds is on vertical axis. We can see a big expense of the de-

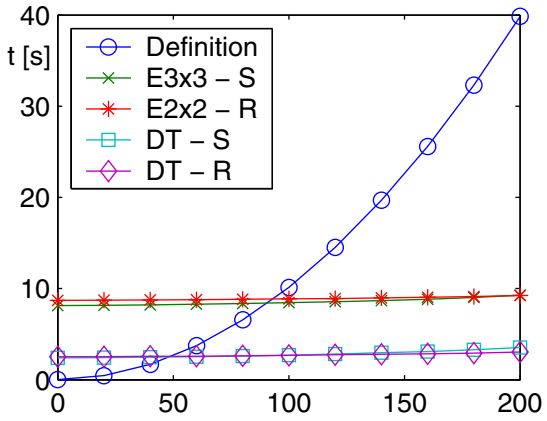


Figure 4. The computation times of the bear image.

compositions (check $t(0)$), but as the requested moment order increases, all decomposition methods outperform the definition. According to our expectation, the best results were achieved by distance transform decomposition into rectangles. It is also apparent from the graph that for any r the distance transform is faster than erosions.

The time graph changes dramatically when we compute moments of the chessboard image, see Fig. 5. The chessboard image is the worst possible case for decomposition algorithms. Now the shape of all the curves is similar, there is almost no difference between the erosions and the distance transform. Direct calculation

[2] All the times in this section are related to the code in C++ language on a PC with Intel Pentium III 2.5GHz CPU and Windows XP.

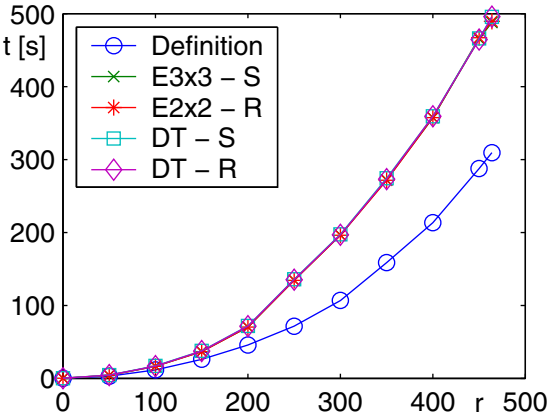


Figure 5. The computation times of the chessboard image.

from the definition exhibits the best performance (note that we did not summed over zero pixels).

The order $r = 464$ is required for precise reconstruction of the image from its moments, where the same number of moments as pixels is needed. The computation times in seconds of the same methods for the order $r = 10$ (which is usually enough for object recognition) are in Tab. 1. The time savings in the case of the bear image and $r = 464$ is about 50 times comparing the computation by definition with the rectangles from the distance transform. The moment values from the different methods are not absolutely identical, but their accuracy is sufficient for precise reconstruction. The decom-

Table 1. The computation times [s].

method	bear		chessboard	
	$r=10$	$r=464$	$r=10$	$r=464$
definition	0.13	270	0.28	309
E3×3 – S	8.2	15	0.23	488
E2×2 – R	8.7	11.4	0.23	489
DT – S	2.4	9.2	0.5	495
DT – R	2.5	5.2	0.45	496

positions of the bear image into (a) 1404 squares and (b) 487 rectangles can be seen in Fig. 6. For a comparison, the chessboard image was decomposed into 108 112 one-pixel blocks.

6. Conclusion

We proposed a new method of decomposition of binary objects into rectangles. This method was developed primarily for the fast moment computation but can

find utilization also in other tasks such as lossless compression and object approximation. The method, based on the distance transform, performs faster than concurrent morphological methods. The method, however, does not provide an optimal decomposition in terms of the number of rectangles. This decomposition algorithm can be also used, without any modifications, to calculate orthogonal moments (more precisely, moments orthogonal on a rectangular region) [9] and moments of graylevel images. Graylevel image can be expressed as a union of disjoint binary images, which can be obtained either as intensity slices [8] or bit planes [13].

Similarly to other decomposition method, the proposed technique is advantageous only when the object is compact and the required number of moments is high. Otherwise the direct calculation from the definition is at least comparable if not faster.

7. Acknowledgement

This work has been supported by the grant No. 102/08/1593 of the Czech Science Foundation.

References

[1] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics, and Image Processing*, 34(3):344–371, 1986.

[2] J. Flusser. Refined moment calculation using image block representation. *IEEE Transactions on Image Processing*, 9(11):1977–1978, 2000.

[3] J. Flusser, T. Suk, and B. Zitová. *Moments and Moment Invariants in Pattern Recognition*. Wiley, Chichester, 2009.

[4] X. Y. Jiang and H. Bunke. Simple and fast computation of moments. *Pattern Recognition*, 24(8):801–806, 1991.

[5] B. C. Li. A new computation of geometric moments. *Pattern Recognition*, 26(1):109–113, 1993.

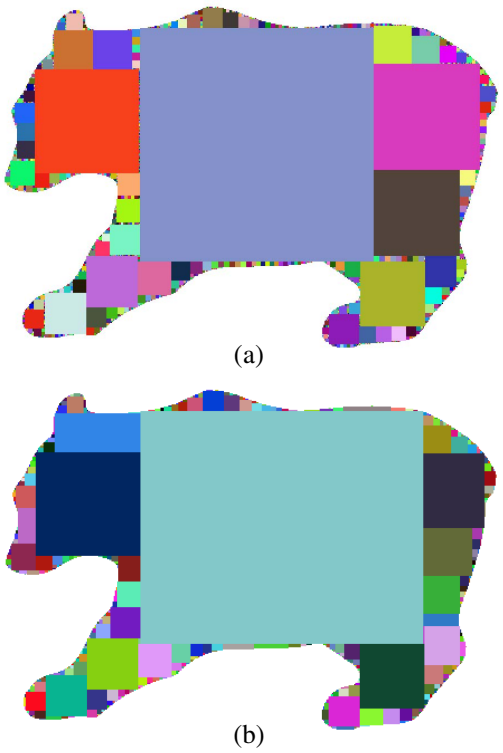
[6] B.-C. Li and J. Shen. Fast computation of moment invariants. *Pattern Recognition*, 24(8):807–813, 1991.

[7] R. Mukundan. Some computational aspects of discrete orthonormal moments. *IEEE Transactions on Image Processing*, 13(8):1055–1059, 2004.

[8] G. A. Papakostas, E. G. Karakasis, and D. E. Koulouriotis. Efficient and accurate computation of geometric moments on gray–scale images. *Pattern Recognition*, 41(6):1895–1904, 2008.

[9] G. A. Papakostas, D. E. Koulouriotis, and E. G. Karakasis. A unified methodology for the efficient computation of discrete orthogonal image moments. *Information Sciences*, 179(20):3619–3633, 2009.

[10] W. Philips. A new fast algorithm for moment computation. *Pattern Recognition*, 26(11):1619–1621, 1993.



(a) (b)

Figure 6. The decomposition into squares resulting in 1404 blocks (a) and into rectangles resulting in 487 blocks (b).

[11] M. Seaidoun. *A fast exact euclidean distance transform with application to computer vision and digital image processing*. PhD thesis, Northeastern University, Boston, USA, September 1993. Advisor John Gauch.

[12] J. H. Sossa-Azuela, C. Yáñez-Márquez, and J. L. Díaz de León Santiago. Computing geometric moments using morphological erosions. *Pattern Recognition*, 34(2):271–276, 2001.

[13] I. M. Spiliotis and Y. S. Boutalis. Parameterized real-time moment computation on gray images using block techniques. *Journal of Real-Time Image Processing*, 2009. 11 pages, DOI 10.1007/s11554-009-0142-0.

[14] I. M. Spiliotis and B. G. Mertzios. Real-time computation of two-dimensional moments on binary images using image block representation. *IEEE Transactions on Image Processing*, 7(11):1609–1615, 1998.

[15] C.-H. Wu, S.-J. Horng, and P.-Z. Lee. A new computation of shape moments via quadtree decomposition. *Pattern Recognition*, 34(7):1319–1330, 2001.

[16] M. F. Zakaria, L. J. Vroomen, P. Zsombor-Murray, and J. M. van Kessel. Fast algorithm for the computation of moment invariants. *Pattern Recognition*, 20(6):639–643, 1987.