Constant-time Extraction of Statistical Moments for Object Detection Procedures

Przemysław Klęsk and Aneta Bera
Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, ul. Żołnierska 49, 71-210 Szczecin, Poland

Keywords: Object Detection, Constant-time Feature Extraction, Normalized Central Statistical Moments, Sets of Integral Images.

Abstract: We propose a computational technique, backed with special integral images, allowing for constant-time extraction of statistical moments within detection procedures. The moments under study are formulated in their normalized central version. The set of proposed integral images needs to be prepared prior to the detection procedure. Its size grows quadratically with the imposed maximum order of moments, but the time invested in the preparation is amortized sufficiently well at the scanning stage. We give exact counts of the number of operations involved in extraction according to the proposed algorithm. The main idea is coupled with an auxiliary technique for detection window partitioning. In the experimental part, we demonstrate two examples of detection tasks. Detectors have been trained on the proposed features by the RealBoost learning algorithm and achieve both: satisfactory time performance and accuracy.

1 INTRODUCTION

Dense detection procedures are computationally expensive and therefore require fast algorithms for feature extraction. If a detector is supposed to analyze, for example, 10^5 windows per image for given settings (resolution, number of scales, etc.) then it is typically expected to analyze a single window below the time of $10^{-5}\text{s} = 10\,\mu\text{s}$. If a strict real-time performance is required then the computational expectations can be higher even by two orders of magnitude.

Nowadays, the most common approaches for feature extraction such as Haar-like features (Viola and Jones, 2001; Viola and Jones, 2004) or HOG descriptors (Dalal and Triggs, 2005; Said et al., 2011) owe their popularity to the computational support of *integral images*. In the case of Haar-like features, the integral image is a simple cumulant of pixel intensities. In the case of HOG desciptor, one can prepare a set of integral images, each cumulating so-called *votes* for the gradients within particular angular sections.

In either case, once the cumulants have been prepared, one can take advantage of them during the detection procedure and have a fast — constant-time — method for extraction of each feature. It is achieved by calculating *growths* of integral images in a manner analogical to the calculus, where definite integrals over rectangular domains can be calculated via growths of so called primitive functions (a.k.a. antiderivatives). For two-dimensional domains a growth operation involves two subtractions and one addition.

We dare noticing that apart from the mentioned features (Haar, HOG), other techniques based on the constant-time approach are rather scarce. Frequently, when more advanced features are needed for detection purposes, the researchers perform some sort of preliminary image segmentation and substantially reduce the number of candidate windows. This allows to carry out more demanding computations, often proportional to the number of pixels in each window (Terrillon et al., 2000; Noh et al., 2017).

In this paper we consider statistical moments as features for machine learning and detection. Various applications of statistical moments can be met in literature, many of them in the field of optical and handwritten character recognition, see e.g. (Abandah and Anssari, 2009; Boveiri, 2010). The main contribution of this paper is a computational technique that makes statistical moments 'fit' detection procedures and their time regime (without image segmentation). The technique applies a *set* of special *integral images* that can be regarded as inner products between the image function and suitable power terms. Extraction of a single feature becomes independent of the num-

Klęsk, P. and Bera, A.
Constant-time Extraction of Statistical Moments for Object Detection Procedures.
DOI: 10.5220/0006550000490059
In *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2018)*, pages 49-59
ISBN: 978-989-758-276-9

ber of pixels in the detection window and thereby is an $O(1)$ calculation. We derive a suitable general formula for an arbitrary imposed order of moments and analyze the number of operations involved. The experimental part demonstrates the applicability of the proposed technique.

We explain that the main intention behind our contribution is to extend the existing repertoire of constant-time algorithms for feature extraction backed with integral images. As mentioned before, that reportoire includes nowadays mainly Haar-like features and HOG descriptor. In this paper we demonstrate that statistical moments can be incorporated into that reportoire as well. We would like to emphasize that our intention is *not* to present new detectors that would be competetive against more recent approaches using convolutional neural networks. Please note that despite highly accurate results and the ability to handle multiple classes, CNNs (or R-CNNs) include in their architecture feature extraction stages that are dependent on the number of pixels in each analyzed image fragment. Therefore, constant-time approaches are better from the algorithmic and computational point of view. Time-efficient applications of CNNs in the field of object detection require prescreening to preselect candidate windows and also strongly parallel architectures at disposal (e.g. GPU / CUDA). That is not the scenario we consider in this paper.

2 NORMALIZED CENTRAL STATISTICAL MOMENTS

We start by reminding the mathematical formulation of continuous version of statistical moments. Let f denote any probability density function. A *central statistical moment* of order (p,q) is defined as follows with respect to f:

$$\mu^{p,q} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left(x-\mu^{1,0}\right)^p \left(y-\mu^{0,1}\right)^q f(x,y)\,dx\,dy, \tag{1}$$

where $p+q \geqslant 2$; and the moments of order one are defined as

$$\mu^{1,0} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x f(x,y)\,dx\,dy, \tag{2}$$

$$\mu^{0,1} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} y f(x,y)\,dx\,dy. \tag{3}$$

For image processing tasks, one typically considers zeroth order approximations of statistical moments. The integrals become replaced with sums weighted by pixel intensities rather than a density function. Moreover, in detection tasks, it is useful to define moments in a form independent of the detection window size and position. Therefore, we shall define *normalized central moments*.

Let i represent an image function. Suppose a rectangular detection window spans from a point (x_1,y_1) to (x_2,y_2). Then, its normalized central statistical moments can be calculated as follows:

$$\mu^{p,q}_{\substack{x_1,y_1\\x_2,y_2}} = \sum_{x_1\leqslant x\leqslant x_2}\sum_{y_1\leqslant y\leqslant y_2} \left(\frac{x-x_1}{x_2-x_1} - \mu^{1,0}_{\substack{x_1,y_1\\x_2,y_2}}\right)^p \cdot \left(\frac{y-y_1}{y_2-y_1} - \mu^{0,1}_{\substack{x_1,y_1\\x_2,y_2}}\right)^q \frac{i(x,y)}{D}, \tag{4}$$

where $D = \sum_{x_1\leqslant x\leqslant x_2}\sum_{y_1\leqslant y\leqslant y_2} i(x,y)$, and the moments of order one are

$$\mu^{1,0}_{\substack{x_1,y_1\\x_2,y_2}} = \sum_{x_1\leqslant x\leqslant x_2}\sum_{y_1\leqslant y\leqslant y_2} \frac{x-x_1}{x_2-x_1}\cdot\frac{i(x,y)}{D}, \tag{5}$$

$$\mu^{0,1}_{\substack{x_1,y_1\\x_2,y_2}} = \sum_{x_1\leqslant x\leqslant x_2}\sum_{y_1\leqslant y\leqslant y_2} \frac{y-y_1}{y_2-y_1}\cdot\frac{i(x,y)}{D}. \tag{6}$$

We remark that the moments of order one can be regarded as estimates of *expected* values for x and y variables. The moments of higher orders are called *central* due to the subtractions of those expected values in (4). The fact that moments are *normalized* pertains to the presence of terms $(x-x_1)/(x_2-x_1)$ and $(y-y_1)/(y_2-y_1)$. This maps the variability of coordinates to the $[0,1]$ interval and introduces invariance to scale of the detection window. It can be checked then that the moments themselves take values in the $[-1,1]$ interval.

Let us remind the interpretation of the few first moment orders. The zero-order moment $\mu^{0,0}$ is obviously a unity (integral over a density function). The first-order moments $\mu^{1,0}$, $\mu^{0,1}$ represent expected values (the means) variable-wise. The second-order moments $\mu^{2,0}$, $\mu^{0,2}$ represent variances about the means, whereas the moment $\mu^{1,1}$ expresses the covariance. The third-order moments $\mu^{3,0}$, $\mu^{0,3}$ denote skewness i.e. the degree of deviation from the symmetry about the mean. The fourth-order moments $\mu^{4,0}$, $\mu^{0,4}$ represent kurtosis i.e. the measure of flatness or peakedness of the distribution. Obviously, moments where neither p or q are zero carry a mixed interpretation of the above quantities. Higher order moments are rarely named but do exist uniquely for a given density function and are responsible for the description of finer details. In other words, in the context of recognition and detection tasks, every object (shape) translates uniquely onto a series of moments and vice-versa. It is worth remarking that the information provided by a *finite* number of statistical moments may be useful or not depending on the particular application.

What must be understood is that the statistical information if of different nature than the information carried e.g. by the popular Haar-like features which can be regarded as rough local contours.

3 INTEGRAL IMAGES OF ARBITRARY ORDER AND CONSTANT-TIME CALCULATION OF MOMENTS

We now introduce the following general formula for an *integral image* of order $l+m$:

$$ii^{l,m}(x,y) = \sum_{1 \leqslant j \leqslant x} \sum_{1 \leqslant k \leqslant y} j^l k^m i(j,k). \tag{7}$$

To explain the notation, we remark that the (l,m) pair of indexes is now used rather than (p,q), because to calculate a moment of order (p,q) later on we shall apply the binomial expansion and therefore a double summation over new indexes e.g. (l,m).

Suppose n represents the imposed maximum order of moments variable-wise. Then, the set of all integral images that have to be prepared prior to the detection procedure can be denoted as

$$\left\{ii^{l,m}\right\} = \left\{ii^{l,m} \colon 0 \leqslant l \leqslant n,\ 0 \leqslant m \leqslant n\right\}. \tag{8}$$

Obviously the size of that set is $(n+1)^2$. Is also worth remarking that every integral image within $\left\{ii^{l,m}\right\}$ can be calculated by induction in linear-time with respect to the total number of pixels in the input image. Note, in particular, that $ii^{0,0}$ corresponds to the elementary integral image, such as the one employed in (Viola and Jones, 2001; Viola and Jones, 2004).

For notational convenience, let us now define the *growth operator* Δ. It can be applied to any integral image and any image window, as follows:

$$\begin{aligned}\underset{\substack{x_1,y_1\\x_2,y_2}}{\Delta}(ii) = {} & ii(x_2,y_2) - ii(x_1-1,y_2) \\ & - ii(x_2,y_1-1) + ii(x_1-1,y_1-1),\end{aligned} \tag{9}$$

where *ii* stands for some integral image from set (8).

The following proposition constitutes the main contribution of the paper.

Proposition 1. *Let $n \geqslant 0$ denote the imposed maximum order of moments variable-wise. Suppose the set of integral images $\left\{ii^{l,m}\right\}$, $0 \leqslant l, m \leqslant n$, has been calculated prior to the detection procedure. Then, for any rectangle in the image, spanning from (x_1,y_1) to (x_2,y_2), each of its normalized central statistical moments can be extracted in constant time — $O(1)$ — regardless of the number of pixels, as follows:*

$$\begin{aligned}\mu^{p,q}_{\substack{x_1,y_1\\x_2,y_2}} = {} & 1 \Big/ \left(\underset{\substack{x_1,y_1\\x_2,y_2}}{\Delta}(ii^{0,0})(x_2-x_1)^p(y_2-y_1)^q\right) \\ & \cdot \sum_{l=0}^{p} \binom{p}{l} \left(-x_1 - \mu^{1,0}_{\substack{x_1,y_1\\x_2,y_2}}(x_2-x_1)\right)^{p-l} \\ & \cdot \sum_{m=0}^{q} \binom{q}{m} \left(-y_1 - \mu^{0,1}_{\substack{x_1,y_1\\x_2,y_2}}(y_2-y_1)\right)^{q-m} \cdot \underset{\substack{x_1,y_1\\x_2,y_2}}{\Delta}(ii^{l,m}),\end{aligned} \tag{10}$$

where $p+q \geqslant 2$.

The proof is a straightforward derivation based on binomial expansions and we move it to the appendix.

Note that formula (10) does *not* dependent on the number of pixels but does scale with indexes p,q and the number of operations is roughly proportional to $p \cdot q$. An accurate count of the number of operations involved can be given after Algorithm 1 is presented. It encapsulates Proposition 1 in a more algorithmic style.

In the algorithm we omit the subscripts related to the rectangle coordinates for readability. Apart from parameters already defined, we introduce an additional argument to the procedure — L. It is meant to represent a lookup table storing binomial coefficients. Such a table can be prepared prior to the detection procedure. Let $L[n,k] = \binom{n}{k}$.

Algorithm 1: Constant-time calculation of a statistical moment of order (p,q) under detection procedure.

procedure SM(p, q, x_1, y_1, x_2, y_2, $\left\{ii^{l,m}\right\}$, $\mu^{1,0}$, $\mu^{0,1}$, D, L)
  $a_x := -x_1 - \mu^{1,0} \cdot (x_2 - x_1)$
  $a_y := -y_1 - \mu^{0,1} \cdot (y_2 - y_1)$
  $\mu^{p,q} := 0.0$
  for $l := 0, \ldots, p$ **do**
    $s := 0.0$
    for $m := 0, \ldots, q$ **do**
      $s := s + L[q,m] \cdot a_y^{q-m} \cdot \Delta_{\substack{x_1,y_1\\x_2,y_2}}(ii^{l,m})$
    end for
    $\mu^{p,q} := \mu_{p,q} + L[p,l] \cdot a_x^{p-l} \cdot s$
  end for
  $\mu^{p,q} := \mu_{p,q} \big/ \left(D(x_2-x_1)^p(y_2-y_1)^q\right)$
  return $\mu^{p,q}$
end procedure

One detail requires an explanation. It is assumed that moments of order one $\mu^{1,0}$, $\mu^{0,1}$ and the constant D have been precalculated and can be passed as arguments to the procedure. Note that D can be obtained using the growth on the zeroth order integral image: $D = \Delta_{\substack{x_1,y_1\\x_2,y_2}}(ii^{0,0})$; whereas moments of order one can

be obtained by the following initial calls:

$$\mu^{1,0} = \mathrm{SM}(1,0,x_1,y_1,x_2,y_2,\{ii^{l,m}\},0.0,0.0,D,L),$$
$$\mu^{0,1} = \mathrm{SM}(0,1,x_1,y_1,x_2,y_2,\{ii^{l,m}\},0.0,0.0,D,L).$$

Please note the 0.0 values passed as the eigth and nineth argument in these particular calls.

Let us distinguish three kinds of operations: additions or subtractions — denoted further as $\oplus$, multiplications or divisions — $\odot$, and powers — $\otimes$. It can be checked by a careful count that the number of operations involved in Algorithm 1 is as follows:

$$\begin{aligned} \oplus&: \quad (p+1)(4q+5)+8, \\ \odot&: \quad 2(p+1)(q+2)+4, \\ \otimes&: \quad (p+1)(q+2)+2. \end{aligned} \tag{11}$$

We remark that 3 additions/subtractions were included in the count for each invocation of the growth operator Δ. If one neglects for a while the kind of an operation then the total number of operations becomes:

$$(p+1)(7q+11)+14. \tag{12}$$

Note also that for a fine-tuned implementation the order of loops in Algorithm 1 could be dynamically switched when $q > p$ to make the number of operations beneficial for that case.

To provide the reader with a specific quantitative example, consider a small image fragment of size 48×48 containing 2 304 pixels. Looking at formula (4), one can see that the definition-sytle calculations would approximately require 8 operations per pixel, that is $2\,304 \cdot 8 \approx 1.8 \cdot 10^4$. In contrast, the constant-time calculations from Proposition 1 represented by Algorithm 1 require e.g.: 89, 290, 905 operations, respectively for $p = q = 2,5,10$.

4 PARTITIONING OF DETECTION WINDOW

One of intentions behind Viola and Jones' idea was to generate a great multitude of features, e.g. of order 10^4 or 10^5. The features themselves might be simple, but by having a great number of them at disposal a learning algorithm (e.g. boosting) can usually select a subset of relevant features that in combination form a good description of target objects. That is why when generating Haar-like features according to a certain parameterization (templates, scales, grid of anchoring points) one typically does not care about lack of orthogonality or partial redundancy of information in the features (correlations).

Until now we have introduced only one parameter $n = 0,1,\ldots$ (the maximum order for the statistical moments) which translates directly onto the number of moments that can be extracted. Even for fairly large but reasonable values of n, we can only have $(n+1)^2$ features. For example, $n = 10$ yields 121 features.

In this section we introduce another parameter $N = 1,2,\ldots$ that will be responsible for the *partitioning* of the detection window. Owing to the partitioning one shall be able to generate richer and more numerous feature spaces.

The proposed partitioning scheme works as follows. Suppose the detection window is of resolution $w_x \times w_y$ during the current scan. The window becomes partitioned into a regular grid of $N \times N$ rectangular pieces. We calculate lengths s_x, s_y of the pieces, taken as integer parts of division by N, and memorize the remainders r_x, r_y:

$$\begin{aligned} s_x &= \lfloor w_x/N \rfloor, & r_x &= w_x \bmod N; \\ s_y &= \lfloor w_y/N \rfloor, & r_y &= w_x \bmod N. \end{aligned} \tag{13}$$

The remainders are needed to center the grid of rectangles within the window (with at most 1 pixel deviation). Instead of using (x_1,y_1) as the top-left starting point of the grid, we shall be using its corrected location, namely:

$$\begin{aligned} x'_1 &= x_1 + \lfloor r_x/2 \rfloor, \\ y'_1 &= y_1 + \lfloor r_y/2 \rfloor. \end{aligned} \tag{14}$$

Now, the set of features can be formed by statistical moments extracted from any subset of pieces in the grid under condition that they form a single *rectangle*. In other words we shall consider all possible rectangles spanned between any two pieces (inclusively) of the grid.

Such rectangles can be represented, for example, by quadruples (j_1,k_1,j_2,k_2) indicating indexes of the corner pieces. Following this notation, possible rectangles can span from $(x'_1 + j_1 s_x, y'_1 + k_1 s_y)$ to $(x'_1 + j_2 s_x, y'_1 + k_2 s_y)$, where $0 \leqslant j_1,k_1 < N$ and $j_1 < j_2 \leqslant N$, $k_1 < k_2 \leqslant N$. Fig. 1 illustrates the window partitioning technique.

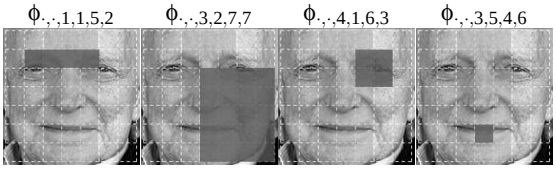

Figure 1: Detection window partitioning for $N = 7$ with examples of rectangles from which statistical moments can be extracted (marked in gray).

We now formally define the set of features as follows:

$$\left\{\phi_{p,q,j_1,k_1,j_2,k_2}\right\} = \left\{\mu^{p,q}_{x'_1+j_1 s_x, y'_1+k_1 s_y}:\ 0 \leqslant p,q \leqslant n,\right.$$
$$\left. 0 \leqslant j_1 < N,\ j_1 < j_2 \leqslant N,\ 0 \leqslant k_1 < N,\ k_1 < k_2 \leqslant N\right\},$$

where each feature ϕ is described by six indexes. It is easy to check that the size of the features set is

$$(n+1)^2 N^2 (N+1)^2/4. \tag{15}$$

For example, by imposing $n = 3$, $N = 7$ one generates 12 544 features.

5 DETECTION EXPERIMENTS

In experiments we apply *RealBoost* (RB) as the main learning algorithm, producing ensembles of weak classifiers with real-valued responses. Various choices are possible as regards the selection of weak classifiers (stumps, shallow decision trees, etc.). We have decided for a variant akin to stumps called RealBost+Bins (RB+B) (Rasolzadeh et al., 2006). In this variant, each weak classifier is based on a single feature selected via minimization of *exponential error*. The range of a feature[1] is divided into a certain number of bins (equally wide) that store classifier's responses calculated as halves of the *logit transform*:

$$\frac{1}{2}\log\left(\widehat{P}(y=+1|\mathbf{x})\Big/\widehat{P}(y=-1|\mathbf{x})\right),$$

where probabilities are estimated using examples' weights from the current round of boosting. Therefore, the weak classifiers work by means of piecewise constant approximations of conditional distributions of classes. For more information on boosting we address the reader to literature, in particular to (Schapire and Singer, 1999; Friedman et al., 2000; Rasolzadeh et al., 2006; Appel et al., 2013).

In the presentation of results, we shall report the imposed sizes of ensembles (denoted by T) and bin counts (denoted by B) to describe classifiers complexity. As regards accuracy results, we shall report the standard measures such as: sensitivity, FAR (false alarm rate), AUC[2], and an overall accuracy.

It is worth mentioning that we apply Jaccard index[3] in two places: (1) to postprocess detected windows and (2) to check positive indications against the ground truth. Typically, a trained detector produces a cluster of many positive windows around each target. At the postprocessing stage, we group such clusters into single indications using Jaccard index. This means that at each step (within a postprocessing loop) two windows with the highest index become averaged. Later, when comparing positive indications against the ground truth, we expect each detected window to have the index of at least 0.5 (with respect to some target position) in order to be counted as a true positive. Otherwise, it becomes a false alarm.

All the software for the presented experiments has been written in C# with key procedures (integral images, feature extraction, detector response) reimplemented for efficiency in C++ as dll libraries.

We remark that cascades of classifiers are not applied in experiments (they are out of the main focus of this paper).

5.1 "Letter A"

For this experiment we have arranged a synthetic data set containing capital letters from the modern English alphabet. Pictures containing the characters of computer fonts were retrieved from the dataset prepared and presented in (de Campos et al., 2009).

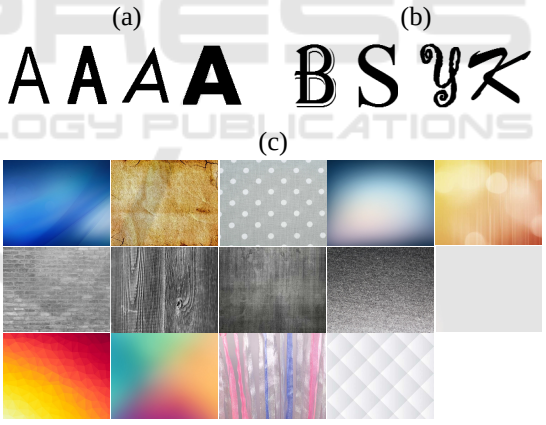

Figure 2: Sample images and backgrounds used to generate the data. Positives: letter 'A' (a), negatives: other letters (b) + elements of backgrounds (c).

We have limited the subset representing the letter 'A' to several fonts with similar characteristics and treated it as our base for creating positive examples. Subsets with other letters were combined in order to prepare the base for negative examples. Fig. 2 depicts the source graphical material used in the experiment (examples of target objects and backgrounds). For testing purposes, we have generated 100 synthetic images by randomly placing letters (without collisions) over random backgrounds. Details on the experimental setup are gathered in Table 1.

[1] Once the outliers are removed. In our case 1% of outliers on each side of an axis has been removed.

[2] area under Receiver Operating Characteristics (ROC) curve

[3] ratio of intersection and union areas

Table 1: "Letter A": experimental setup.

train data		
quantity / parameter	value	additional information
no. of positive examples	1 000	windows with letter 'A'
no. of negative examples	10 000	windows with letters other than 'A' plus random samples of backgrounds
train set size	11 000	positives and negatives in total
test data		
no. of images	100	
no. of positive examples	194	windows with letter 'A' (in all test images)
no. of negative examples	14 035 306	other windows (non-'A') (in all test images)
test set size	14 035 500	positives and negatives in total
no. of negative examples for ROC plot	2 000 000	negative examples sampled on random
detection procedure (scanning with a sliding window)		
image resolution	600×480	imposed resolution
no. of detection scales	8	images scanned with 8 different sizes of window
window growing coefficient	1.2	window widths and heights increase by $\approx 20\%$ per scale
smallest window size	48×48	
largest window size	172×172	
window jumping coefficient	0.05	window jumps equal to $\approx 5\%$ of its width and height

Results. We start reporting results by showing the ROC curves in Fig. 3.

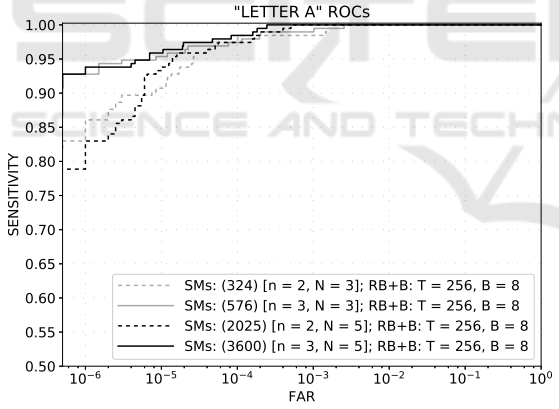

Figure 3: ROC curves for detectors of letter 'A' trained on statistical moments (learning algorithm: RealBoost + bins).

The detectors associated with them have been trained on statistical moments (SMs) obtained for fairly small settings of the feature space parameterization, i.e. $n = 2,3$ as the maximum orders of moments, and $N = 3,5$ as the sizes of partitioning grids. To distinguish better between the curves, logarithmic scale has been imposed on the FAR axis and the sensitivity axis range has been narrowed down. As one can note the parameter N, controlling the grid size, turned out to be of secondary importance in this experiment. ROCs obtained for the higher statistical order, $n = 3$, surpassed the ones for $n = 2$.

Table 2 reports accuracy measures obtained by carrying out the detection procedure on 100 test images.

Table 2: "Letter A": detection results for RB+B algorithm ($B = 8$, $T = 256$).

SMs description (no. of feats.) $[n,N]$	$AUC_{10^{-5}}$	sensitivity	FAR per image	FAR per window $[\cdot 10^{-8}]$	accuracy per window
(324) [2,3]	0.8863	$186/194 \approx 0.9588$	$0/100 = 0.00$	0.000	0.999999430016743
(576) [3,3]	0.9424	$194/194 = 1.0000$	$0/100 = 0.00$	0.000	1.000000000000000
(2025) [2,5]	0.8807	$191/194 \approx 0.9845$	$1/100 = 0.01$	7.125	0.999999715008372
(3600) [3,5]	0.9451	$193/194 \approx 0.9948$	$1/100 = 0.01$	7.125	0.999999857504186

The detector obtained for $n = 3$ and $N = 3$ achieved a perfect score (no false negetatives nor positives), but the remaining detectors also performed very well. Overall results suggest that this data set can be regarded as a fairly easy one. In the table, among other measures, we report $AUC_{10^{-5}}$. It is a normalized AUC measure obtained up to the point 10^{-5} along the FAR axis. In general, one can look at AUC_α values, defined as

$$\frac{1}{\alpha}\int_0^\alpha s(f)\,df, \tag{16}$$

where $s(f)$ represents sensitivity treated as a function of false alarm rate[4]. For example, a sequence of $AUC_{10^{-6}}$, $AUC_{10^{-5}}$, $AUC_{10^{-4}}$ provides a good information about the initial behaviour of the ROC curve, which is of particular importance in detection tasks. On the other hand, the AUC=$AUC_{1.0}$ is of little information, when obtained detectors are comparable and of high accuracy. Please note also that the area under ROC can be interpreted as the average sensitivity.

In Fig. 4 we present some examples of detection outcomes produced by the best detector. Fig. 5 depict the only false alarm raised by the detector with settings $n = 3$, $N = 5$. As one can note it represent the letter 'N', a part of which was mistaken for an 'A'.

Last but not least, in Table 3 we report the time performance[5] registered on our computer with Intel Xeon E5-2699 v4 CPU 2.2 GHz (22/44 cores/threads, 55 MB cache).

The results pertain to the most accurate detector ($n = 3$, $N = 3$). The required preparation of 16 integral images (in this case) took 53 ms of the overall time of 245 ms for the whole procedure. During

[4] technically, ROC is a curve, not a function

[5] median results observed

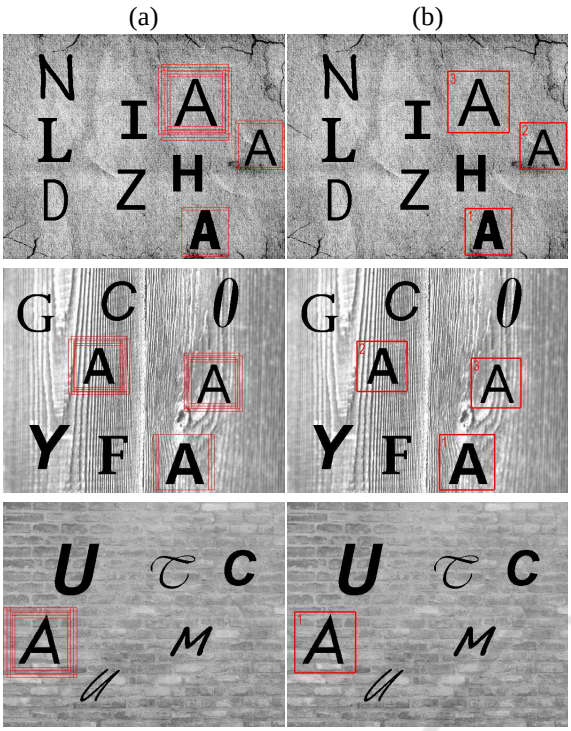

Figure 4: “Letter A”: examples of detections — (a) direct outcomes with all single indications, (b) postprocessed outcomes after grouping clusters of windows.

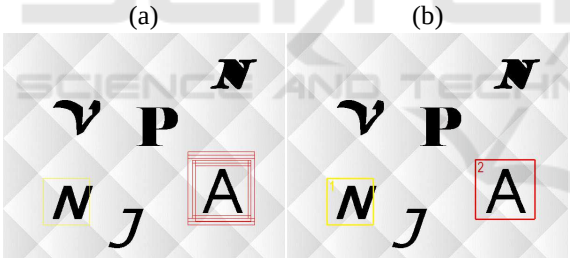

Figure 5: “Letter A”: the single false alarm raised by the detector for $n = 3$, $N = 5$.

Table 3: “Letter A”: time performance for a 600×480 image (parallel computations on: Intel Xeon E5-2699 v4 CPU 2.2 GHz, 22/44 cores/threads, 55 MB cache).

SMs (576) [3,3], $T = 256$, $B = 8$	
quantity (or operations)	time or amount
no. of analyzed windows	140 355
total time of detection procedure	245 ms
no. of prepared integral images	16
preparation time of all integral images	53 ms
preparation time per 1 integral image	3.313 ms
time per 1 window	1.746 μs (amortized: 1.368 μs)
no. of distinct features used by ensemble	120
time per 1 window and 1 feature	14.55 ns (amortized: 11.40 ns)

the procedure 140 355 image windows were analyzed. Please note that even though the learning algorithm was given 576 features at disposal and that the number of boosting rounds was $T = 256$, it finally selected to use only 120 distinct features. The average time of analysis per a single feature was 14.55 ns.

5.2 “Faces”

The learning material for this experiment consisted of images with faces (upright position or close to it), looked up using the *Google Images* search engine for queries such as: ‘person’, ‘people’, ‘family’, ‘children’, ‘students’, etc. The selected results translated onto a train set containing 7 258 face examples coming from 3 000 images. Tab. 4 presents all the details about the experimental setup.

Table 4: “Faces”: experimental setup.

train data		
quantity / parameter	value	additional information
no. of images with faces	3 000	downloaded from *Google Images* for queries: person, people, group of people, family, children, sportsmen, students, etc.;
no. of images without faces	300	as above, queries: view, landscape, street, cars, etc.
no. of positive examples	7 258	face windows marked manually
no. of negative examples	100 000	imposed quantity; examples sampled at random positions and scales within images without faces
train set size	107 258	positives and negatives in total
test data		
no. of images	500	queries as for train data (other images)
no. of positive examples	1 000	windows with faces (in all test images)
no. of negative examples	70 251 859	windows with faces (in all test images)
test set size	70 252 859	positives and negatives in total
no. of negative examples for ROC plot	2 000 000	negative examples sampled on random
detection procedure (scanning with a sliding window)		
image height	480	before detection, images scaled to the height 480, keeping original height : width proportion
no. of detection scales	8	images scanned with 8 different sizes of window
window growing coefficient	1.2	window widths and heights increase by $\approx 20\%$ per scale
smallest window size	48×48	faces smaller than $\approx 10\%$ of image height not to be detected
largest window size	172×172	faces larger than $\approx 36\%$ of image height not to be detected
window jumping coefficient	0.05	window jumps equal to $\approx 5\%$ of its width and height

Results. Again, we have applied RB+B as the learning algorithm. This time we have imposed larger ensembles consisting of 512 weak classifiers and we have been experimenting with both $B = 8$ and $B = 16$

counts for bins. The sizes of generated features sets ranged from 5625 to 19600. Results for the six detectors that we have tested in this experiment are presented in Fig. 6 (ROC curves) and Table 5.

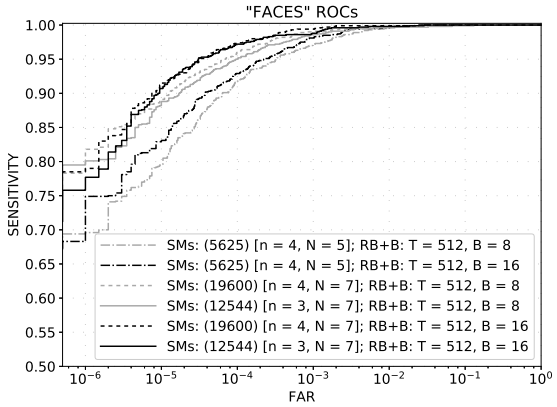

Figure 6: ROC curves for face detectors trained on statistical moments (learning algorithm: RealBoost + bins).

Table 5: "Faces": detection results for RB+B algorithm ($B = 8$ or $B = 16$).

SMs description (no. of feats.) $[n,N]$	$AUC_{10^{-5}}$	sensitivity	FAR per image	FAR per window $[\cdot 10^{-7}]$	accuracy per window
$B = 8$					
(5625) [4,5]	0.7550	752/1000 = 0.752	58/500 = 0.116	8.26	0.999995644363969
(12544) [3,7]	0.8387	852/1000 = 0.852	82/500 = 0.164	11.7	0.999996726155925
(19600) [4,7]	0.8559	780/1000 = 0.780	49/500 = 0.098	6.97	0.999996171025842
$B = 16$					
(5625) [4,5]	0.7862	840/1000 = 0.840	122/500 = 0.244	17.4	0.999995985982481
(12544) [3,7]	0.8495	867/1000 = 0.867	65/500 = 0.130	9.25	0.999997181647274
(19600) [4,7]	0.8651	826/1000 = 0.826	45/500 = 0.090	6.41	0.999996882731076

Looking at accuracy measures in Table 5, one can see that face detection is obviously a more difficult task than detection of simple geometrical patterns (like it was the case in the previous experiment). The best obtained face detector trained on statisitcal moments ($n = 3$, $N = 7$, $B = 16$) exhibits the 86.7% sensitivity and 13% false alarms per image. Obviously, results measured at the 'windows level-of-detail' look more optimistically — the overall accuracy of the best detector was approximately 99.99972%, which could be confronted with the zero-rule classification performance: 99.99953% for our test data.

Examples of correct detections are shown in Fig. 7, whereas Fig. 8 deptics several erroneous outcomes with misdetections or false alarms. Various reasons can be attributed to misdetections, e.g.: unnatural facial expressions, lightening conditions, blurs, non-upright face positions, glasses, fringes, tatoos, etc. As regards false alarms, it is difficult to notice a general rule, but some of them can be explained by bearing certain resemblance to faces.

Figure 7: "Faces": examples of correct detections by the best among detectors trained on statistical moments ($n = 3$, $N = 7$, $B = 16$) — (a) direct outcomes with all single indications, (b) postprocessed outcomes after grouping clusters of windows.

Finally, as in the previous experiment, we report the time performance — see details in Table 6. The results pertain to the most accurate detector ($n = 3$, $N = 7$, $B = 16$). It is worth to remark that in this experiment the learning algorithm 'decided' to select and use more[6] distinct features, namely: 484. This led to a longer overall time of the procedure (722 ms for 150849 analyzed windows), but the average time of computations per feature was roughly at the same level: 9.89 ns.

[6]relatively to the former experiment

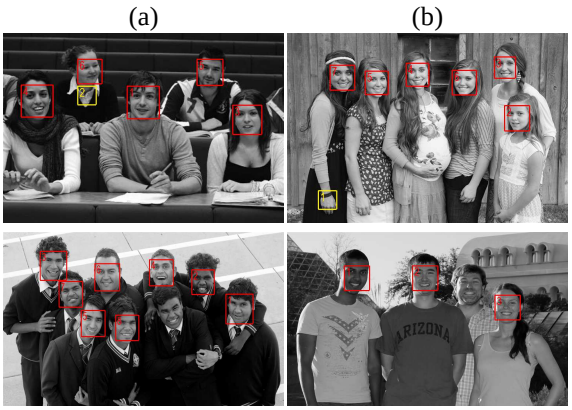

Figure 8: "Faces": examples of faulty detections (false alarms and misdetections) produced by the best among detectors trained on statistical moments ($n = 3$, $N = 7$, $B = 16$).

Table 6: "Faces": time performance for a 640×480 image (parallel computations on: Intel Xeon E5-2699 v4 CPU 2.2 GHz, 22/44 cores/threads, 55 MB cache).

SMs (12544) [3,7], $T = 512$, $B = 16$	
quantity (or operations)	time or amount
no. of analyzed windows	150849
total time of detection procedure	722 ms
no. of prepared integral images	16
preparation time of all integral images	62 ms
preparation time per 1 integral image	3.875 ms
time per 1 window	4.786 μs (amortized: 4.375 μs)
no. of distinct features used by ensemble	484
time per 1 window and 1 feature	9.89 ns (amortized: 9.04 ns)

We also remark that we stick to fairly low orders of moments $n = 2, 3$ for computational reasons. Be reminded that formula (**??**) from Proposition 1 is constant-time regardless of the number of pixels in the analyzed window, but it does depend on the imposed orders of moments. It requires $O(pq)$ operations, hence $O(n^2)$ with respect to n. Therefore, one must consider the trade-off between accuracy and time performance. In the case of our face detection experiments, switching to $n = 4$ would result in detection times higher roughly by the factor of 1.5 taking into account formula (12).

5.3 "Faces": Statistical Moments vs Haar-like Features

In this experiment we compare face detectors trained on statistical moments (SMs) and Haar-like features (HFs). More precisely, we take under considerations four detectors:

(A) the best detector from the previous section trained on SMs (12 544 features),

(B) the worst detector from the previous section trained on SMs (5 625 features),

(C) a detector trained on HFs (6 125 features),

(D) a detector trained on a combination of features from cases C and B: HFs (6 125 features) and SMs (5 625 features).

The above numbers given in parentheses indicate the counts of features available at the learning stage, not the counts of distinct features finally applied. It should be explained that the number of Haar-like features is implied by the following parameterization of the related feature space: 5 Haar templates[7], 25 scaled versions of a feature per a template, 7×7 grid of anchoring points ($5 \cdot 25 \cdot 7^2 = 6\,125$).

Results. ROC curves obtained for the four considered detectors are plotted in Fig. 9.

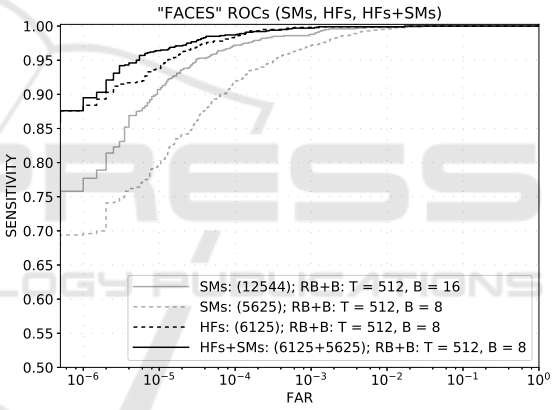

Figure 9: ROC curves for face detectors trained on: statistical moments (SMs), Haar-like features (HFs), and a mixture (HFs+SMs).

The following two observations can be formulated immediately by looking at the ROC characteristics: (1) Haar-like features appear to be better suited for face detection than statistical moments (for our conditions of the experiment); (2) operating characteristics obtained for the detector using the combined feature space surpass all others. A closer insight into the particular features selected by the combined detector revealed that there were 155 statistical moments within the total of 486 distinct features in the ensemble ($\approx 32\%$). In other words, the combined detector benefited from both the information residing in Haar-like features and statistical moments.

[7] 2 vertical templates (double and tirple rectangle), 2 horizontal templates (double and tirple rectangle), 1 diagonal template

Table 7 presents detailed accuracy measures obtained by the detection procedure carried out on 500 test images. The combined detector (HFs+SMs) turned out to be superior than all others, however, the difference from the detector based on HFs alone is very small and may be data dependent (within statisitcal error).

Table 7: "Faces": detection results for detectors trained on: statistical moments (SMs), Haar-like features (HFs), and a mixture (HFs+SMs).

detector description (no. of feats.)	$AUC_{10^{-5}}$	sensitivity	FAR per image	FAR per window $[\cdot 10^{-7}]$	accuracy per window
$B = 8$					
SMs (12544)	0.8495	$867/1000 = 0.867$	$65/500 = 0.130$	9.25	0.999997181647274
SMs (5625)	0.7550	$752/1000 = 0.752$	$58/500 = 0.116$	8.26	0.999995644363969
HFs (6125)	0.9151	$953/1000 = 0.953$	$28/500 = 0.056$	3.99	0.999998932442149
HFs+SMs (6125+5625)	0.9381	$947/1000 = 0.947$	$21/500 = 0.042$	2.99	0.999998946676254

Finally, Table 8 presents a juxtaposition of time measurements obtained for the two most interesting detectors based on: Haar-like features alone and a combination of Haar-like features and statistical moments. The reader is also encouraged to confront the numbers in the table with results reported in the former experiment (Table 8).

Table 8: "Faces": time performance for a 640×480 image for detectors based on HFs and HFs+SMs (parallel computations on: Intel Xeon E5-2699 v4 CPU 2.2 GHz, 22/44 cores/threads, 55 MB cache).

HFs (6125), $B = 8$, $T = 512$	
quantity (or operations)	time or amount
no. of analyzed windows	150849
total time of detection procedure	289 ms
no. of prepared integral images	1
preparation time of all integral images	8 ms
preparation time per 1 integral image	8 ms
time per 1 window	1.916 μs (amortized: 1.863 μs)
no. of distinct features used by ensemble	486
time per 1 window and 1 feature	3.94 ns (amortized: 3.83 ns)
HFs+SMs (6125+5625), $B = 8$, $T = 512$	
quantity (or operations)	time or amount
no. of analyzed windows	150849
total time of detection procedure	522 ms
no. of prepared integral images	26
preparation time of all integral images	87 ms
preparation time per 1 integral image	3.346 ms
time per 1 window	3.460 μs (amortized: 2.884 μs)
no. of distinct features used by ensemble	486
time per 1 window and 1 feature	7.12 ns (amortized: 5.93 ns)

6 CONCLUSIONS

We have proposed a compuational technique which allows to extract statistical moments (normalized, central) of arbitrary order with a support of special integral images. The computation time does not depend on the number of pixels in the detection window, and therfore is a constant-time computation in this respect, but does depend on the particular order of each moment. As an auxiliary technique, we have proposed a partitioning scheme for the detection window. By applying it, one can generate large feature spaces while keeping the maximum order of moments reasonably small. We belief our proposition can extend the repertoire of existing approaches dedicated for dense detection procedures.

The presented idea can be regarded as both competitive and complementary to the commonly met Haar-like features. In particular, the experiment from Section 5.3 demonstrates that a detector can benefit from having different kinds of features at disposal when learning, e.g. both: Haar-like features and statistical moments. Obviously, from the point of view of computational complexity, the constant-time feature extraction is a prerequisite for such a favourable scenario to take place.

As regards other work within our research project, both ongoing and future, it pertains to various types of features used in detection tasks and attempts at representing them via special integral images. In particular, we have already achieved some results for constant-time extraction of Fourier moments of low orders (Klęsk, 2017). We still plan to 'attack' Zernike and Fourier–Mellin moments and back their extraction with integral images.

ACKNOWLEDGEMENTS

This work was financed by the National Science Centre, Poland. Research project no.: 2016/21/B/ST6/01495.

REFERENCES

Abandah, G. and Anssari, N. (2009). Novel Moment Features Extraction for Recognizing Handwritten Arabic Letters. *Journal of Computer Science*, 5(3):226–232.

Appel, R. et al. (2013). Quickly Boosting Decision Trees — Pruning Underachieving Features Early. In *Proc. of the 30th Int. Conference on Machine Learning (ICML-13)*, volume 28, pages 594–602. JMLR Workshop and Conference Proceedings.

Boveiri, H. (2010). On Pattern Classification Using Statistical Moments. *International Journal of Signal Processing and Pattern Recognition*, 3(4):15–24.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA. IEEE Computer Society.

de Campos, T. E., Babu, B. R., and Varma, M. (2009). Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407.

Klęsk, P. (2017). Constant-time fourier moments for face detection — can accuracy of haar-like features be beaten? In *Artificial Intelligence and Soft Computing: 16th International Conference, ICAISC 2017, Zakopane, Poland, June 11-15, 2017, Proceedings, Part I*, pages 530–543. Springer International Publishing.

Noh, Y., Koo, D., Kang, Y., et al. (2017). Automatic crack detection on concrete images using segmentation via fuzzy c-means clustering. In *International Conference on Applied System Innovation (ICASI 2017)*, pages 877–880. IEEE.

Rasolzadeh, B. et al. (2006). Response Binning: Improved Weak Classifiers for Boosting. In *IEEE Intelligent Vehicles Symposium*, pages 344–349.

Said, Y., Atri, M., and Tourki, R. (2011). Human detection based on integral Histograms of Oriented Gradients and SVM. In *Communications, Computing and Control Applications (CCCA 2011)*, pages 1–5. IEEE.

Schapire, R. and Singer, Y. (1999). Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336.

Terrillon, J.-C., McReynolds, D., Sadek, et al. (2000). Invariant neural-network based face detection with orthogonal Fourier-Mellin moments. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 2, pages 993–1000.

Viola, P. and Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. In *Conference on Computer Vision and Pattern Recognition (CVPR'2001)*, pages 511–518. IEEE.

Viola, P. and Jones, M. (2004). Robust Real-time Face Detection. *International Journal of Computer Vision*, 57(2):137–154.

APPENDIX

Proof of Proposition 1

Proof. The derivation is started from formula (4). First, the means (moments of order one), that are present under powers, should be multiplied by suitable unity terms: $\mu_{\cdot}^{1,0}\cdot\frac{x_2-x_1}{x_2-x_1}$, $\mu_{\cdot}^{0,1}\cdot\frac{y_2-y_1}{y_2-y_1}$. This allows to extract the denominators and form the normalizing constant $1/\left(D(x_2-x_1)^p(y_2-y_1)^q\right)$ in front of the summation. Then, the powers are expanded by means of the binomial theorem, grouping the terms into the ones dependent on the current pixel index (x,y) and the ones independent of it. This yields:

$$\frac{1}{D(x_2-x_1)^p(y_2-y_1)^q}\sum_{x_1\leqslant x\leqslant x_2}\sum_{y_1\leqslant y\leqslant y_2}\left(\sum_{l=0}^{p}\binom{p}{l}x^l\left(-x_1-\mu^{1,0}_{\substack{x_1,y_1\\x_2,y_2}}(x_2-x_1)\right)^{p-l}\cdot\sum_{m=0}^{q}\binom{q}{m}y^m\left(-y_1-\mu^{0,1}_{\substack{x_1,y_1\\x_2,y_2}}(y_2-y_1)\right)^{q-m}\cdot i(x,y)\right). \quad (17)$$

Finally, by changing the order of summations one arrives at:

$$\frac{1}{D(x_2-x_1)^p(y_2-y_1)^q}\cdot\sum_{l=0}^{p}\binom{p}{l}\left(-x_1-\mu^{1,0}_{\substack{x_1,y_1\\x_2,y_2}}(x_2-x_1)\right)^{p-l}\cdot\sum_{m=0}^{q}\binom{q}{m}\left(-y_1-\mu^{0,1}_{\substack{x_1,y_1\\x_2,y_2}}(y_2-y_1)\right)^{q-m}\cdot\underbrace{\sum_{x_1\leqslant x\leqslant x_2}\sum_{y_1\leqslant y\leqslant y_2}x^l y^m i(x,y)}_{\Delta^{x_1,y_1}_{x_2,y_2}(ii^{l,m})}. \quad (18)$$

The underbrace indicates how the expensive summation over all pixels in the rectangle gets replaced by the cheap constant-time computation of the growth of a suitable integral image. Note also that the required normalizer D is calculated by the growth of the zeroth order integral image $D=\Delta^{x_1,y_1}_{x_2,y_2}(ii^{0,0})$. □