Journal of VLSI Signal Processing 25, 5–17, 2000.

Computing Moments by Prefix Sums

FENG ZHOU†
Department of Information and Electronic Eng., Zhejiang University, People's Republic of China

PETER KORNERUP
Department of Math. and Computer Science, SDU/Odense University, Denmark

Abstract. Moments of images are widely used in pattern recognition, because in suitable form they can be made invariant to variations in translation, rotation and size. However the computation of discrete moments by their definition requires many multiplications which limits the speed of computation. In this paper we express the moments as a linear combination of higher order prefix sums, obtained by iterating the prefix sum computation on previous prefix sums, starting with the original function values. Thus the p'th moment $m_p = \sum_{x=1}^{N} x^p f(x)$ can be computed by $O(N \cdot p)$ additions followed by p multiply-adds. The prefix summations can be realized in time $O(N)$ using $p+1$ simple adders, and in time $O(p \log N)$ using parallel prefix computation and $O(N)$ adders. The prefix sums can also be used in the computation of two-dimensional moments for any intensity function $f(x, y)$. Using a simple bit-serial addition architecture, it is sufficient with 13 full adders and some shift registers to realize the 10 order 3 image moment computations ($m_{00}, m_{01}, m_{10}, m_{02}, m_{20}, m_{12}, m_{21}, m_{03}, m_{30}$) for a 512×512 size image at the TV rate. In 1986 Hatamian published a computationally equivalent algorithm, based on a cascade of filters performing the summations. Our recursive derivation allows for explicit expressions and recursive equations for the coefficients used in the final moment calculation. Thus a number of alternative forms for the moment computation can be derived, based on different sets of prefix sums. It is also shown that similar expressions can be obtained for the moments introduced by Liao and Pawlak in 1996, forming better approximations to the exact geometric moments, at no extra computational cost.

1. Introduction

The final goal of many image processing tasks is to recognize objects, to match scenes and to classify images. The shape, gray or color features of objects are mainly used to describe objects. The most efficient method to describe the shape, gray or color features of an object is to use moments, which was proposed by Hu [1] in 1962. After moments of an object have been computed, they can be converted to various other kinds of moments, e.g., standard moments or moment invariants. Some are independent of scale, translation, rotation and reflexion, so these are now widely used in pattern recognition. However, the computation of moments requires a large number of additions and multiplications. In many real-time applications, the speed of computation is of crucial importance. Various algorithms [2–11] have been proposed to reduce the number of multiplications or to implement fast multiplications. Parallel algorithms [2, 3, 6, 7] are also used for the computation of moment invariants. For binary images which contain only the information about the shape of an abject, the two dimensional moment computation (a function of the area) can be converted to a one dimensional computation (a function of the boundary), so the computation of moments can be simplified [6–11].

In this paper we derive a fast algorithm for the computation of discrete moments, which uses only some additions for every picture element. Computationally

*Supported by the Danish Research Councils, grant number 5.21.08.02.
†Work performed during a stay at Odense University.

this algorithm turns out to be equivalent to the algorithm by Hatamian [4]. The one-dimensional moment $m_p = \sum_{x=1}^{N} x^p f(x)$ is rewritten into a linear combination $m_p = \sum_{n=1}^{p} B_n^p S_1^{n+1}$, where the B_n^p are simple integer coefficients which can be obtained in advance, and the values S_1^n are obtained by iteratively computing prefix sums over previous prefix sums, starting with the values of $f(x)$. Using the definition, computing all moments up to some order p requires $O(N \cdot p)$ multiplications and $O(N \cdot p)$ additions, whereas this method only requires $O(N \cdot p)$ additions, followed by $O(p^2)$ multiply-add operations. Since N is generally much larger than p, basically all multiplications are eliminated. The algorithm can be applied to two-dimensional moment computation. Since for image processing the data rate is exclusively determined by the speed of the adders, and the order of moments and size of image influence only the width of the adders, it is possible to realize the real-time computation of moments for any intensity function $f(x, y)$. It is also shown that the improved approximations to the geometric moments introduced by Liao and Pawlak [12] can be computed from the prefix sums in a completely analogous manner.

Hatamian [4] derived a computational equivalent algorithm by observing that the output of a digital filter with impulse response $h_p(n) = n^p u(n)$ ($u(n)$ being the unit function) is precisely the p'th moment $_p = \sum_0^N (N - n)^p f(n)$, when $f(n)$ is input to the filter. He then decomposed the all-pole version of the filter into a cascade of a single-pole filters, and observed that these generate linear combinations of the moments through the order p. Thus the moments can be obtained by solving a (triangular) system of equations.

Our derivation is based on the recursive nature of prefix sum computations, applied to a binomial expression of the definition of the moments. This way we obtain explicit expressions and recursive equations for the coefficients for a variety of linear forms for the moments. Because the image size is usually much larger than the moment order $(p+q)$ for image moments, if the ratio of the time between two adjacent picture elements to the delay time of an adder is , only $\lceil \frac{q+1}{} \rceil + 1$ or $\lceil \frac{p+1}{} \rceil + 1$ adders are required for the computation of moments of the order $(p + q)$. Using a bit-serial addition method, it is sufficient with 13 full adders and some shift registers to realize the order 3 image moment computation ($m_{00}, m_{01}, m_{10}, m_{02}, m_{20}, m_{12}, m_{21}, m_{03}, m_{30}$).

In Section 2 we express the computation of moments as linear combinations of prefix sums over the picture elements. In Section 3 some other representations of the moment computation using the prefix sums are derived, together with the computation of moments in hardware. The improved approximations to the geometrical moments are then introduced in Section 4, and in Section 5 two-dimensional moments are computed using two-dimensional prefix sums, and the 10 order 3 moments are implemented using bit serial addition. Section 6 concludes the paper.

2. A New Expression for the Discrete Moments

Given a two-dimensional digital image intensity function $f(x, y)$, the *discrete moment* of order $(p + q)$ of an area A is defined as

$$m_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{M} x^p y^q f(x, y).$$

This is often used as an approximation to the *geometric moment* defined as the integral:

$$\tilde{m}_{pq} = \int_0^N \int_0^M x^p y^q f(x, y)\, dy\, dx$$

but as demonstrated by Liao and Pawlak [12] better approximations exists. As we shall show later, it is sufficient initially to consider the one-dimensional discrete moment of order p:

$$m_p = \sum_{x=1}^{N} x^p f(x),$$

from which we will derive a better approximation $\hat{m}_p$ of $\tilde{m}_p = \int_0^N f(x)\, dx$, following Liao and Pawlak.

For x integral and $p > 0$, x^p can be expressed as

$$\begin{aligned} x^p &= \sum_{x_1=1}^{x} \left[x_1^p - (x_1 - 1)^p \right] \\ &= \sum_{x_1=1}^{x} \sum_{p_1=0}^{p-1} \binom{p}{p_1} (-1)^{(p+1-p_1)} x_1^{p_1} \\ &= \sum_{x_1=1}^{x} \sum_{p_1=0}^{p-1} A_{p_1}^p x_1^{p_1} \end{aligned} \tag{1}$$

where[1]

$$A^p_{p_1} = \binom{p}{p_1}(-1)^{(p+1-p_1)}.$$

Now define *higher order prefix sums*, $S^n_{x_n}$, by initialization:

$$S^0_{x_0} = f(x_0), \quad x_0 = 1, 2, 3, \ldots, N,$$

and recursively calculating prefix sums over prefix sums:

$$S^n_{x_n} = \sum_{x_{n-1}=x_n}^{N} S^{n-1}_{x_{n-1}} \tag{2}$$

for $n = 1, 2, 3, \ldots$ and $x_n = 1, 2, 3, \ldots, N$. Observe that this recursive computation is easily realized, noting that $S^n_x = S^{n-1}_x + S^n_{x+1}$, by computing S^n_x in decreasing order of $x = N-1, N-2, \ldots, 2, 1$. Also define

$$R^i_{p_i} = \sum_{x_i=1}^{N} x_i^{p_i} S^i_{x_i},$$

so $R^0_p = m_p$, and by (1) we can now rewrite $R^i_{p_i}$

$$\begin{aligned}
R^i_{p_i} &= \sum_{x_i=1}^{N} x_i^{p_i} S^i_{x_i} \\
&= \sum_{x_i=1}^{N} \sum_{x_{i+1}=1}^{x_i} \sum_{p_{i+1}=0}^{p_i-1} A^{p_i}_{p_{i+1}} x_{i+1}^{p_{i+1}} S^i_{x_i} \\
&= \sum_{x_{i+1}=1}^{N} \sum_{x_i=x_{i+1}}^{N} \sum_{p_{i+1}=0}^{p_i-1} A^{p_i}_{p_{i+1}} x_{i+1}^{p_{i+1}} S^i_{x_i} \\
&= \sum_{x_{i+1}=1}^{N} \sum_{p_{i+1}=0}^{p_i-1} A^{p_i}_{p_{i+1}} x_{i+1}^{p_{i+1}} \sum_{x_i=x_{i+1}}^{N} S^i_{x_i} \\
&= \sum_{p_{i+1}=0}^{p_i-1} \sum_{x_{i+1}=1}^{N} A^{p_i}_{p_{i+1}} x_{i+1}^{p_{i+1}} S^{i+1}_{x_{i+1}} \\
&= A^{p_i}_0 \sum_{x_{i+1}=1}^{N} x^0_{i+1} S^{i+1}_{x_{i+1}} \\
&\quad + \sum_{p_{i+1}=1}^{p_i-1} A^{p_i}_{p_{i+1}} \sum_{x_{i+1}=1}^{N} x_{i+1}^{p_{i+1}} S^{i+1}_{x_{i+1}} \\
&= A^{p_i}_0 S^{i+2}_1 + \sum_{p_{i+1}=1}^{p_i-1} A^{p_i}_{p_{i+1}} R^{i+1}_{p_{i+1}}.
\end{aligned} \tag{3}$$

Let $p_0 = p$ and using (3), then for $p > 0$, m_p can be expressed as

$$\begin{aligned}
m_p = R^0_{p_0} &= A^{p_0}_0 S^2_1 + \sum_{p_1=1}^{p_0-1} A^{p_0}_{p_1} R^1_{p_1} \\
&= A^{p_0}_0 S^2_1 + \sum_{p_1=1}^{p_0-1} A^{p_0}_{p_1} A^{p_1}_0 S^3_1 + \sum_{p_1=1}^{p_0-1} A^{p_0}_{p_1} \times \sum_{p_2=1}^{p_1-1} A^{p_1}_{p_2} R^2_{p_2} \\
&= A^{p_0}_0 S^2_1 + \sum_{p_1=1}^{p_0-1} A^{p_0}_{p_1} A^{p_1}_0 S^3_1 \\
&\quad + \sum_{p_2=1}^{p_0-2} \sum_{p_1=p_2+1}^{p_0-1} A^{p_0}_{p_1} A^{p_1}_{p_2} R^2_{p_2}.
\end{aligned}$$

The third term here can be split and becomes

$$\begin{aligned}
&\sum_{p_2=1}^{p-2} \sum_{p_1=p_2+1}^{p-1} A^{p_0}_{p_1} A^{p_1}_{p_2} A^{p_2}_0 S^4_1 \\
&\quad + \sum_{p_3=1}^{p-3} \sum_{p_2=p_3+1}^{p-2} \sum_{p_1=p_2+1}^{p-1} A^{p_0}_{p_1} A^{p_1}_{p_2} A^{p_2}_{p_3} R^3_{p_3}.
\end{aligned}$$

Continuing to split the last term, after n steps the n'th term is split into

$$\begin{aligned}
&\sum_{p_{n-1}=1}^{p-n+1} \sum_{p_{n-2}=p_n+1}^{p-n+2} \cdots \sum_{p_1=p_2+1}^{p-1} \prod_{i=1}^{n-1} A^{p_{i-1}}_{p_i} A^{p_{n-1}}_0 S^{n+1}_1 \\
&\quad + \sum_{p_n=1}^{p-n} \sum_{p_{n-1}=p_n+1}^{p-n+1} \cdots \sum_{p_1=p_2+1}^{p-1} \prod_{i=1}^{n} A^{p_{i-1}}_{p_i} R^n_{p_n},
\end{aligned}$$

and with $n = p_0 = p$ the last term is split into

$$\begin{aligned}
&\sum_{p_{p-1}=1}^{1} \sum_{p_{p-2}=p_{p-1}+1}^{2} \cdots \sum_{p_1=p_2+1}^{p-1} \prod_{i=1}^{p-1} A^{p_{i-1}}_{p_i} A^{p_{p-1}}_0 S^{p+1}_1 \\
&\quad + \sum_{p_p=1}^{0} \sum_{p_{p-1}=p_p}^{1} \sum_{p_{p-2}=p_{p-1}+1}^{2} \cdots \sum_{p_1=p_2+1}^{p-1} \prod_{i=1}^{p-1} A^{p_{i-1}}_{p_i} R^{p-1}_{p_{p-1}}
\end{aligned}$$

The last sum in the above expression is zero, so m_p can be represented as

$$m_0 = S^1_1, \quad m_p = \sum_{n=1}^{p} B^p_n S^{n+1}_1 \tag{4}$$

where

$$\begin{aligned} B_n^p &= \sum_{p_{n-1}=1}^{p_0-n+1} \sum_{p_{n-2}=p_{n-1}+1}^{p_0-n+2} \cdots \\ &\quad \cdots \sum_{p_2=p_3+1}^{p_0-2} \sum_{p_1=p_2+1}^{p_0-1} A_{p_1}^{p_0} A_{p_2}^{p_1} \cdots A_{p_{n-1}}^{p_{n-2}} A_0^{p_{n-1}} \\ &= (-1)^p \sum_{k=0}^{n} \binom{n}{k} (k)^p (-1)^k \\ &= (-1)^{p+n} n! \left\{ \begin{matrix} p \\ n \end{matrix} \right\}. \end{aligned}$$

The last factor in the above formula is the Stirling number of the second kind. It is easy to see that B_n^p satisfies the following recursion:

$$B_n^p = n\big(B_{n-1}^{p-1} - B_n^{p-1}\big)$$

with $B_0^0 = 1$, $B_0^p = 0$ for $p > 0$, $B_1^p = (-1)^{p-1}$ and $B_n^p = 0$ for $n > p$. Note that with B_0^p defined, all m_p in (4) can be computed by

$$m_p = \sum_{n=0}^{p} B_n^p S_1^{n+1} \quad \text{for } p \geq 0.$$

According to the above, we can list the coefficients for the first 6 moments in Table 1.

Let us conclude this section by stating an algorithm for computing the moments $m_0, m_1, m_2, \ldots, m_p$, from the values of the function $f(N)$, $f(N-1), \ldots, f(1)$, obtained by observing that $S_i^n = S_i^{n-1} + S_{i+1}^n$ implies $S_i^n = f(i) + \sum_{k=1}^{n} S_{i+1}^k$ with $S_{N+1}^k = 0$:

Algorithm PSM (Prefix Sum Moments)
for $k := 1$ **to** $p+1$ **do** $S[k] := 0$;
for $i := N$ **downto** 1 **do**
begin
  $S[0] := f[i]$;
  for $k := 1$ **to** $p+1$ **do** $S[i] := S[i] + S[i-1]$;
end;
$m[0] := S[1]$;
for $k := 1$ **to** p **do**
begin
  $h := 0$;
  for $n := 1$ **to** k **do** $h := h + B[n,k] \times S[n+1]$;
  $m[k] := h$;
end;

Observe that we have here interchanged the order of summation for the recursive prefix summation over prefix summation, compared to the definition (2). Also note that this is again (suitably interpreted) prefix summation over prefix sums. This happens to be the organization of summation used by Hatamian [4], except that he input data in the reverse order, and thus computes moments with respect to the endpoint. Obviously, computing the first p moments $m_0, m_1, \ldots, m_p$ requires $N(p+1)$ additions for the calculations of the prefix sums, followed by $p(p+1)/2$ multiply-adds for the final formation of the moments. Using the definition $m_p = \sum_1^N x^p f(x)$ requires pN multiplications and $p(N-1)$ additions. Since in practice N is much larger than p, the new method essentially saves the pN multiplications. Using the definition (2) of S_x^n, all the prefix sums can be computed in time $O(p \log N)$ using parallel prefix computation and $O(N)$ adders. However, for image processing, data is normally input sequentially so processing the data sequentially is feasible. Organizing the summation as in the algorithm above, the inner loop again performs prefix summations which can be realized in time $O(\log p)$ additions per row, for a total of $O(N \log p)$ addition times. But a simple pipelining of $p+1$ adders implies that data can be input at a rate corresponding to a single adder cycle.

Table 1. The coefficients B_n^p for the first 6 moments.

p	B_0^p	B_1^p	B_2^p	B_3^p	B_4^p	B_5^p	B_6^p
0	1						
1	0	1					
2	0	−1	2				
3	0	1	−6	6			
4	0	−1	14	−36	24		
5	0	1	−30	150	−240	120	
6	0	−1	62	−540	1560	−1800	720

In the previous expressions for moment computation, data is input in the order from the last to the first in order to compute the needed prefix sums. We can also input data from the the first to the last. In this case we can obtain another representation m'_q of moments as follows:

$$\begin{aligned} m'_0 &= \sum_{x=1}^{N} f(x) = \ddot{S}_1^1 \\ m'_p &= \sum_{x=1}^{N} (N+1-x)^p f(x) \\ &= \sum_{n=1}^{p} B_n^p \ddot{S}_1^{(n+1)} \end{aligned}$$

where the new prefix sums are

$$\ddot{S}^0_{x_0} = f(x_0), \quad \ddot{S}^{n+1}_{x_{n+1}} = \sum_{x_n=1}^{x_{n+1}} \ddot{S}^n_{x_n}.$$

Here m'_p can also be converted into other moments, e.g. into standard moments or moment invariants. If we need moment m_p, it can be obtained from m'_p as follows:

$$\begin{aligned} m_p &= \sum_{x=1}^{N} x^p f(x) \\ &= \sum_{x=1}^{N} [(N+1) - (N+1-x)]^p f(x) \\ &= \sum_{p_1=0}^{p} \binom{p}{p_1} (N+1)^{(p-p_1)} (-1)^{p_1} m'_{p_1}. \end{aligned}$$

3. Some Alternative Expressions for the Moments

Observe from the Table 1 that some coefficients B^p_n have negative values, some are larger than $n!$. In this section we will derive some alternative expression for m_p, using other coefficients and other sets of prefix sums.

We may then define some new coefficients $C^p_{n,i}$, $i = 0, 1, 2, 3, \ldots$, which again are prefix sums over prefix sums:

$$\begin{aligned} C^p_{n,0} &= B^p_n \\ C^p_{n,i} &= \sum_{k=n}^{p} C^p_{k,i-1} = \sum_{k=0}^{n} \binom{n}{k} (i-k)^p (-1)^k \end{aligned}$$

for $i = 1, 2, 3, \ldots$.

It is also easy to prove that $C^p_{n,i}$ satisfies the following recursion:

$$C^{p+1}_{n,i} = nC^p_{n-1,i} + (i-n)C^p_{n,i}$$

with $C^p_{1,i} = i^p - (i-1)^p$. Now using

$$\begin{aligned} S^{k+1}_n &= S^k_n + S^{k+1}_{n+1} \\ &= S^{k-1}_n + S^k_{n+1} + S^{k+1}_{n+1} \\ &\vdots \\ &= S^{n+1}_n + \sum_{i=n+1}^{k} S^{i+1}_{n+1}, \end{aligned}$$

let

$$\begin{aligned} T^p_n &= \sum_{k=n}^{p} C^p_{k,n-1} S^{k+1}_n \\ &= \sum_{k=n}^{p} C^p_{k,n-1} \left(S^{n+1}_n + \sum_{i=n+1}^{k} S^{i+1}_{n+1} \right) \\ &= \left(\sum_{k=n}^{p} C^p_{k,n-1} \right) S^n_{n+1} + \sum_{k=n+1}^{p} C^p_{k,n-1} \sum_{i=n+1}^{k} S^{i+1}_{n+1} \\ &= C^p_{n,n} S^{n+1}_n + \sum_{i=n+1}^{p} S^{i+1}_{n+1} \sum_{k=i}^{p} C^p_{k,n-1} \\ &= C^p_{n,n} S^{n+1}_n + \sum_{i=n+1}^{p} C^p_{i,n} S^{i+1}_{n+1} \\ &= C^p_{n,n} S^{n+1}_n + T^p_{n+1}. \end{aligned}$$

But $m_p = T^p_1$, so for $p \geq 1$

$$m_p = C^p_{1,1} S^2_1 + T^p_2 = \sum_{n=1}^{p} C^p_{n,n} S^{n+1}_n, \tag{5}$$

where the coefficients $C^p_{n,n}$ can be shown to be $C^p_{n,n} = |B^p_n|$. All coefficients are now non-negative. But we can also express m_p in another form using the C coefficients:

$$\begin{aligned} m_p &= \sum_{n=1}^{p} B^p_n S^{n+1}_1 = \sum_{n=1}^{p} B^p_n \left(S^2_1 + \sum_{k=2}^{n} S^{k+1}_2 \right) \\ &= \sum_{n=1}^{p} B^p_n S^2_1 + \sum_{n=1}^{p} B^p_n \sum_{k=2}^{n} S^{k+1}_2 \\ &= C^p_{1,1} S^2_1 + \sum_{n=2}^{p} C^p_{n,1} S^{n+1}_2 \\ &= C^p_{1,1} S^2_1 + C^p_{2,1} S^3_2 + \sum_{n=3}^{p} C^p_{n,2} S^{n+1}_2 \\ &= C^p_{1,1} S^2_1 + C^p_{2,1} S^3_2 + C^p_{3,2} S^4_2 + C^p_{4,2} S^5_3 + \\ &\quad \cdots + \sum_{n=k}^{p} C^p_{n,k-1} S^{n+1}_k \\ &= \sum_{n=1}^{p} C^p_{n,\lceil \frac{n}{2} \rceil} S^{n+1}_{\lceil \frac{n+1}{2} \rceil}, \end{aligned}$$

valid for $p \geq 1$. With $C^0_{0,0} = 1$ and $C^p_{0,0} = 0$ for $p \geq 1$, it is again possible to use the general form for $p \geq 0$

$$m_p = \sum_{n=0}^{p} C^p_{n,\lceil \frac{n}{2} \rceil} S^{n+1}_{\lceil \frac{n+1}{2} \rceil}.$$

Table 2. The coefficients $C^p_{n,\lceil \frac{n}{2} \rceil}$ for the first 6 moments.

p	$C^p_{0,0}$	$C^p_{1,1}$	$C^p_{2,1}$	$C^p_{3,2}$	$C^p_{4,2}$	$C^p_{5,3}$	$C^p_{6,3}$
0	1	1					
1	0	1					
2	0	1	2				
3	0	1	0	6			
4	0	1	2	12	24		
5	0	1	0	30	0	120	
6	0	1	2	60	120	360	720

We can then prove that $C^p_{n,\lceil \frac{n}{2} \rceil}$ has the following properties:

$$
\begin{aligned}
p! \geq C^p_{n,\lceil \frac{n}{2} \rceil} &\geq 0 \\
C^p_{1,1} = \sum_{k=1}^{p} B^p_k &= 1 \\
C^p_{p,\lceil \frac{p}{2} \rceil} = B^p_p &= p! \\
C^p_{n,\lceil \frac{n}{2} \rceil} &= \begin{cases} 0 & p = 1, 3, 5, 7, \ldots \\ 2C^p_{n-1,\lceil \frac{n}{2} \rceil} & p = 2, 4, 6, 8, \ldots, \end{cases}
\end{aligned}
$$

for $n = 2, 4, 6, 8, \ldots$.

The coefficients $C^p_{n,\lceil \frac{n}{2} \rceil}$ for the first 6 moments are shown in Table 2. Comparing the above coefficients with the previous two kinds of coefficients we can see that now all coefficients $C^p_{n,\lceil \frac{n}{2} \rceil}$, are non-negative and equal to or less than $p!$, and almost half of the coefficients for odd valued p are zero.

For yet another form we note that from $S^{n+1}_x = S^n_x + S^{n+1}_{x+1}$ we can also obtain:

$$S^{k+1}_n = S^{p+1}_n - \sum_{j=k+1}^{p} S^{j+1}_{n+1}$$

Now define a new set of coefficients $D^p_{n,i}$

$$
\begin{aligned}
D^p_{n,0} &= -B^p_n \\
D^p_{n,i} &= -\sum_{k=1}^{n-1} D^p_{k,i-1} = -\sum_{k=i}^{n} \binom{n}{k} (i-k)^p (-1)^k
\end{aligned}
$$

for $i = 1, 2, 3, \ldots$.

Similarly to the coefficients $C^p_{n,i}$, $D^p_{n,i}$ also satisfies a recursion:

$$D^{p+1}_{n,i} = -nD^p_{n-1,i} - (i-n)D^p_{n,i}$$

with $D^p_{n,i} = 0$ for $n = 1, 2, 3, \ldots, i$ and $D^p_{i+1,i} = (-1)^i$. Also let

$$U^p_n = -\sum_{k=1}^{p} D^p_{k,n-1} S^{k+1}_n,$$

and in analogy with the derivation of T^p_n we obtain

$$U^p_n = D^p_{p+1,n} S^{p+1}_n + U^p_{n+1},$$

hence we have

$$m_p = U^p_1 = \sum_{n=1}^{p} D^p_{p+1,n} S^{p+1}_n.$$

Since we only need the values of $D^p_{p+1,n}$ we may drop the lower index $p+1$, so that

$$D^p_n = D^p_{p+1,n},$$

which can be found to satisfy

$$
\begin{aligned}
D^p_n &= \sum_{k=0}^{n-1} \binom{p+1}{k} (n-k)^p (-1)^k \\
D^p_n &= D^p_{p+1-n} > 0
\end{aligned}
$$

for $n = 1, 2, 3, \ldots, p$. From Table 3 listing the coefficients D^p_n for the first 6 moments we also see that these coefficients are smaller than the former coefficients, and exhibits a certain symmetry.

Hence for moment computation we now have four different forms, using various sets of prefix sums, as summarized below:

1) $$m_p = \sum_{n=0}^{p} B^p_n S^{n+1}_1$$

2) $$m_p = \sum_{n=0}^{p} \left| B^p_n \right| S^{n+1}_n$$

Table 3. The coefficients D^p_n for the first 6 moments.

p	D^p_0	D^p_1	D^p_2	D^p_3	D^p_4	D^p_5	D^p_6
0	1						
1	0	1					
2	0	1	1				
3	0	1	4	1			
4	0	1	11	11	1		
5	0	1	26	66	26	1	
6	0	1	57	302	302	57	1

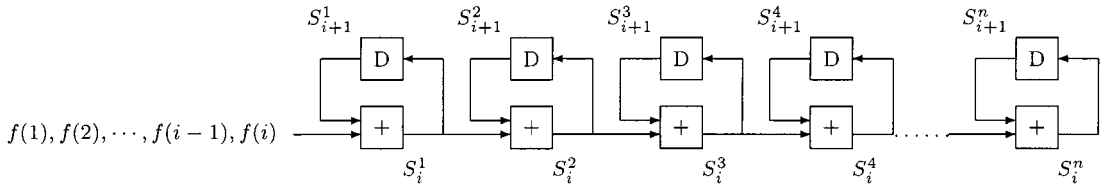


Figure 1. An adder structure for the computation of S_i^n.

3) $$m_p = \sum_{n=0}^{p} C^p_{n,\lceil \frac{n}{2} \rceil} S^{n+1}_{\lceil \frac{n+1}{2} \rceil}$$

4) $$m_p = \sum_{n=0}^{p} D^p_n S^{p+1}_n.$$

where the expressions for m_p have been extended to apply for $p = 0$ (summation is started at $n = 0$), by appropriate definitions of the values of the coefficients for $n = 0$ and defining $S_0^1 = S_1^1$. Besides the different coefficients in the four forms, we note that different sets of prefix sums are used. Thus $p + 1$ different prefix sums (including S_1^1) are needed for each form in forms 1)–3) whereas $p(p + 1)/2 + 1$ different prefix sums are required for form 4).

The prefix summations can easily be realized by adders as shown in Fig. 1. Assuming that it takes time t to compute the set $\{S_i^n\}$ from the values of S_{i+1}^n in this structure, then the set of values $\{S_i^n\}$ is available at time $(N + 1 - i)t$.

Fig. 2 shows a diagram of the particular sets of values $\{S_i^n\}$ needed for the computation of the moments $m_0, m_1, m_2, \ldots, m_p$, using forms 1) to 4), together with the time of their availability. For form 1) the bottom row is needed, and it is available at time $N \cdot t$. Form 2) requires the values $\{S_n^{n+1}\}$ along the diagonal, with S_n^{n+1} available at time $(N - n + 1)t$. For form 3) two useful prefix sums $S^{n+1}_{\frac{n}{2}+1}$ and $S^{n+2}_{\frac{n}{2}+1}$ are available at time $(N - \frac{n}{2})t$. Finally, form 4) requires all the sums shown in the diagram (m_k requires column $k + 1$).

Since all four forms require $p(p+1)/2$ multiply-add operation, a design of a hardware structure for the final computation of the moments $m_0, m_1, m_2, \ldots, m_p$ will depend on the order in which the needed prefix sums becomes available. If in Fig. 1 the structure is made into a pipeline as shown in Fig. 3, then the whole diagram in Fig. 2 becomes skewed, such that the diagonal becomes available at time $N \cdot$, where now is the delay of one adder stage of the pipeline. The whole triangle then becomes delayed (pushed down), e.g., so that the prefix sum S_1^n needed in form 1) becomes available at time $(N + n - 1)$.

$$
\begin{array}{cccccccc}
 & & & & & & S_p^{p+1} & (N+1-p)*t \\
 & & & & & S_{p-1}^{p} & S_{p-1}^{p+1} & (N+2-p)*t \\
 & & & & \cdots\cdots & & & \\
 & & & S_i^{i+1} & \cdots & S_i^{p} & S_i^{p+1} & (N+1-i)*t \\
 & & & & \cdots\cdots & & & \\
 & & S_3^4 & S_3^5 & S_3^6 & \cdots & S_3^{p} & S_3^{p+1} & (N-2)*t \\
 & S_2^3 & S_2^4 & S_2^5 & S_2^6 & \cdots & S_2^{p} & S_2^{p+1} & (N-1)*t \\
S_1^1 & S_1^2 & S_1^3 & S_1^4 & S_1^5 & S_1^6 & \cdots & S_1^{p} & S_1^{p+1} & N*t
\end{array}
$$

Figure 2. Delay times for the different prefix sums needed.

4. Improved Approximations to the Geometric Moments

As pointed out by Liao and Pawlak [12] the discrete moments are not the best possible approximations to the geometric moments. Again considering the one-dimensional case, let

$$\tilde{m}_p = \int_0^N x^p f(x)\, dx$$

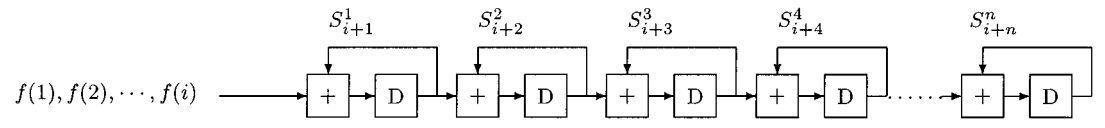


Figure 3. A pipelined structure for the implementation of S_i^n.

and define

$$\hat{m}_p = \sum_{i=1}^{N} f_i \int_{i-1}^{i} x^p \, dx$$

where $f_i = f(i - \frac{1}{2})$. If we assume (as in the case where $f(x)$ represents pixel values) that the function f is piecewise constant on intervals of the form $[i-1; i]$, then $\hat{m}_p = \tilde{m}_p$, and in general $\hat{m}_p$ will be a better approximation to $\tilde{m}_p$ than m_p is.

We shall now show that $\hat{m}_p$ can similarly be expressed as linear combinations of prefix sums. First we express $\hat{m}_p$ in terms of m_k, $k = 0, 1, \ldots, p$.

$$\begin{aligned}
\hat{m}_p &= \sum_{i=1}^{N} f_i \int_{i-1}^{i} x^p \, dx \\
&= \sum_{i=1}^{N} f_i \frac{i^{p+1} - (i-1)^{p+1}}{p+1} \\
&= \frac{1}{p+1} \sum_{i=1}^{N} f_i \left[i^{p+1} + \sum_{k=0}^{p+1} \binom{p+1}{k} i^k (-1)^{p-k} \right] \\
&= \frac{1}{p+1} \sum_{i=1}^{N} f_i \sum_{k=0}^{p} \binom{p+1}{k} i^k (-1)^{p-k} \\
&= \frac{1}{p+1} \sum_{k=0}^{p} \binom{p+1}{k} (-1)^{p-k} \sum_{i=1}^{N} i^k f_i \\
&= \frac{1}{p+1} \sum_{k=0}^{p} \binom{p+1}{k} (-1)^{p-k} m_k \qquad (6)
\end{aligned}$$

Expressing m_k in terms of prefix sums, using any of the formulas found in the previous sections, e.g., using the B_n^k-coefficients, we have:

$$\begin{aligned}
\hat{m}_p &= \frac{1}{p+1} \sum_{k=0}^{p} \binom{p+1}{k} (-1)^{p-k} \sum_{n=0}^{k} B_n^k S_1^{n+1} \\
&= \sum_{n=0}^{p} \hat{B}_n^p S_1^{n+1}, \qquad (7)
\end{aligned}$$

where the new coefficients $\hat{B}_n^p$ are expressed by

$$\hat{B}_n^p = \frac{1}{p+1} \sum_{k=n}^{p} \binom{p+1}{k} (-1)^{p-k} B_n^k. \qquad (8)$$

It is thus possible to compute the improved approximation $\hat{m}_p$ in the very same way as m_p in the form of linear combinations of prefix sums, just using different coefficients $\hat{B}_n^p$. Evaluating (8) we find the first few coefficients as listed in Table 4.

Table 4. The coefficients $\hat{B}_n^p$ for the first 6 moments.

p	$\hat{B}_0^p$	$\hat{B}_1^p$	$\hat{B}_2^p$	$\hat{B}_3^p$	$\hat{B}_4^p$	$\hat{B}_5^p$	$\hat{B}_6^p$
0	1						
1	$-\frac{1}{2}$	1					
2	$\frac{1}{3}$	-2	2				
3	$-\frac{1}{4}$	$\frac{7}{2}$	-9	6			
4	$\frac{1}{5}$	-6	30	-48	24		
5	$-\frac{1}{6}$	$\frac{31}{3}$	-90	260	-300	120	
6	$\frac{1}{7}$	-18	258	-1200	2400	-2160	720

Table 5. The coefficients $\hat{C}_{n,n}^p$ for the first 6 moments.

p	$\hat{C}_{0,0}^p$	$\hat{C}_{1,1}^p$	$\hat{C}_{2,2}^p$	$\hat{C}_{3,3}^p$	$\hat{C}_{4,4}^p$	$\hat{C}_{5,5}^p$	$\hat{C}_{6,6}^p$
0	1						
1	$-\frac{1}{2}$	1					
2	$\frac{1}{3}$	0	2				
3	$-\frac{1}{4}$	$\frac{1}{2}$	3	6			
4	$\frac{1}{5}$	0	6	24	24		
5	$-\frac{1}{6}$	$\frac{1}{3}$	10	80	180	120	
6	$\frac{1}{7}$	0	18	240	960	1440	720

Observe that the coefficients have alternating signs, which is of concern for image processing, where the data and hence the prefix sums are non-negative. Thus there is a danger of severe cancellations in the computation of the inner products (7) of $\hat{m}_p$, and since the prefix sums grow very large, it may not be possible to perform all the computations exact in integer arithmetic.

For the discrete moments it was found that the C and D coefficients were non-negative, and using (6) we can find equivalent $\hat{C}$ and $\hat{D}$ coefficients to use for the better approximations $\hat{m}_p$. Recall that two forms were found using C-coefficients, the first one using $C_{n,n}^p = |B_n^p|$, for which Table 5 can be computed using (5) and (6).

Here S_0^1 is needed, which may be defined as before by $S_0^1 = S_1^1$, to allow the general expression:

$$\hat{m}_p = \sum_{n=0}^{p} \hat{C}_{n,n}^p S_n^{n+1} \quad \text{for } p \geq 0.$$

Note that S_0^1 and its coefficients are numerically small, so there will not be any severe cancellation in the computation here.

Table 6. The coefficients $\hat{C}^p_{n,\lceil \frac{n}{2} \rceil}$ for the first 6 moments.

p	$\hat{C}^p_{0,0}$	$\hat{C}^p_{1,1}$	$\hat{C}^p_{2,1}$	$\hat{C}^p_{3,2}$	$\hat{C}^p_{4,2}$	$\hat{C}^p_{5,3}$	$\hat{C}^p_{6,3}$
0	1						
1	$-\frac{1}{2}$	1					
2	$\frac{1}{3}$	0	2				
3	$-\frac{1}{4}$	$\frac{1}{2}$	-3	6			
4	$\frac{1}{5}$	0	6	0	24		
5	$-\frac{1}{6}$	$\frac{1}{3}$	-10	20	-60	120	
6	$\frac{1}{7}$	0	18	0	240	0	720

Table 7. The coefficients $\hat{D}^p_n$ for the first 6 moments.

p	$\hat{D}^p_0$	$\hat{D}^p_1$	$\hat{D}^p_2$	$\hat{D}^p_3$	$\hat{D}^p_4$	$\hat{D}^p_5$	$\hat{D}^p_6$
0	1						
1	$-\frac{1}{2}$	1					
2	$\frac{1}{3}$	0	1				
3	$-\frac{1}{4}$	$\frac{1}{2}$	$\frac{5}{2}$	1			
4	$\frac{1}{5}$	0	5	9	1		
5	$-\frac{1}{6}$	$\frac{1}{3}$	$\frac{28}{3}$	$\frac{251}{6}$	$\frac{47}{2}$	1	
6	$\frac{1}{7}$	0	17	154	229	54	1

Similarly we find coefficients for the alternative expression using $\hat{C}$-coefficients:

$$\hat{m}_p = \sum_{n=0}^{p} \hat{C}^p_{n,\lceil \frac{n}{2} \rceil} S^{n+1}_{\lceil \frac{n+1}{2} \rceil} \quad \text{for } p \geq 0.$$

as listed in Table 6, now of smaller magnitudes but unfortunately also of alternating signs for odd values of p.

Finally, for form 4) the modified coefficients $\hat{D}^p_n$ may be found as listed in Table 7.

We may thus conclude that for all four forms it is possible to obtain the improved approximations to the geometric moments, by simply changing the coefficients in the expressions, using the very same prefix sums. In the following we shall hence not be concerned with which set of coefficients are used.

5. Two-Dimensional Moment Computation

Applying the prefix summation method to the two-dimensional space, we can realize the computation of two-dimensional moments using two methods. In both methods the following prefix sums for $n = 0, 1, 2, \ldots, q$, $y_{n+1} = 1, 2, 3, \ldots$ in the first dimensional direction are computed initially:

$$S^0_{y_0}(x) = f(x, y_0) \quad \text{and} \quad S^{n+1}_{y_{n+1}}(x) = \sum_{y_n = y_{n+1}}^{N} S^n_{y_n}(x).$$

The first method then continues by computing prefix sums $S^{n+1,m+1}_{x_{n+1},y_{m+1}}$ for $m = 0, 1, 2, 3, \ldots, p$, $x_{m+1} = 1, 2, 3, \ldots$ in the second dimensional direction:

$$S^{0,n+1}_{x_0,y_{n+1}} = S^{n+1}_{y_{n+1}}(x_0),$$

and

$$S^{m+1,n+1}_{x_{m+1},y_{n+1}} = \sum_{x_m = x_{m+1}}^{N} S^{m,n+1}_{x_m,y_{n+1}},$$

thus the moments

$$m_{pq} = \sum_{x=1}^{N} \sum_{y=1}^{N} x^p y^q f(x, y)$$

can be obtained for the various forms as:

1) $$m_{pq} = \sum_{m=0}^{p} \sum_{n=0}^{q} B^p_m B^q_n S^{m+1,n+1}_{1,1},$$

2) $$m_{pq} = \sum_{m=0}^{p} \sum_{n=0}^{q} \left| B^p_m \right\| B^q_n \left| S^{m+1,n+1}_{m,n} \right.,$$

3) $$m_{pq} = \sum_{m=0}^{p} \sum_{n=0}^{q} C^p_{m,\lceil \frac{m}{2} \rceil} C^q_{n,\lceil \frac{n}{2} \rceil} S^{m+1,n+1}_{\lceil \frac{n+1}{2} \rceil, \lceil \frac{n+1}{2} \rceil},$$

4) $$m_{pq} = \sum_{m=0}^{p} \sum_{n=0}^{q} D^p_m D^q_n S^{p+1,q+1}_{m,n},$$

Using this first method, the number of multiplications needed for computing the moment m_{pq} of order $p + q$ is thus pq.

The second method reduces the number of multiplications by beginning with performing one of the four kinds of the first dimensional moment computations $m_q(x) = \sum_{y=1}^{N} y^p f(x, y)$:

1) $$m_q(x) = \sum_{n=0}^{q} B^q_n S^{n+1}_1(x)$$

2) $$m_q(x) = \sum_{n=0}^{q} \left| B^q_n \right| S^{n+1}_n(x)$$

3) $$m_q(x) = \sum_{n=0}^{q} C^q_{n,\lceil \frac{n}{2} \rceil} S^{n+1}_{\lceil \frac{n+1}{2} \rceil}(x)$$

4) $$m_q(x) = \sum_{n=0}^{q} D^q_{n,n} S^{p+1}_n(x)$$

Computing the first dimensional moment $m_q(x)$ of order q thus needs q multiply-adds. Then the following prefix sums for $m = 0, 1, 2, 3, \ldots, p$, $x_{m+1} = 1, 2, 3, \ldots$ are computed

$$\hat{S}^{0,q}_{x_0} = m_q(x_0), \quad \hat{S}^{m+1,q}_{x_{m+1}} = \sum_{x_m=0}^{N} \hat{S}^{m,q}_{x_m}.$$

Finally, using p multiply-adds, the two dimensional moments m_{pq} are found as

$$1) \qquad m_{pq} = \sum_{m=0}^{p} B^p_m \hat{S}^{m+1,q}_1$$

$$2) \qquad m_{pq} = \sum_{m=0}^{p} \left| B^p_m \right| \hat{S}^{m+1,q}_m$$

$$3) \qquad m_{pq} = \sum_{m=0}^{p} C^p_{m,\lceil \frac{m}{2} \rceil} \hat{S}^{m+1,q}_{\lceil \frac{m+1}{2} \rceil}$$

$$4) \qquad m_{pq} = \sum_{m=0}^{p} D^p_m \hat{S}^{p+1,q}_m,$$

so only $p+q$ multiply-adds are required for the second method.

Therefore for two-dimensional $(n = p+q)'$th moment computation, using the prefix sum method, only pq or $p+q$ multiplication-additions are needed, in addition to computing the prefix sums $S^{n,m}$ or $\ddot{S}^{n,m}$ using $(n+1)(n+4)/2$ adders (each is used to compute one prefix sum). An implementation of the computation of the two-dimensional prefix sums $S^{n,m}(j)$ needed for method 1 is shown in Fig. 4. Only additions are needed to obtain the prefix sums, so a high speed can be obtained for two-dimensional computations.

When the speed of an adder is higher than the data input rate, the number of adders for the computation of the first dimensional sums S^n can be reduced to $\lceil \frac{q+1}{} \rceil$ (is the ratio of the time between two data and the delay of the adder) by time-multiplexing a single adder. Since usually the order p of moments in the second dimension is much smaller than the number of data in one of the dimensions, only a single adder is needed for the computation of all second dimensional sums $S^{m,n}$. The whole structure of this prefix sum computation for two dimensions is shown in Fig. 5.

In image processing the following 10 moments $m_{00}, m_{01}, m_{10}, m_{11}, m_{02}, m_{20}, m_{12}, m_{21}, m_{03}, m_{30}$ are often used. We can use form 3) to implement the computation of these, and for simplicity in this example we use the discrete moments as defined in Section 3,

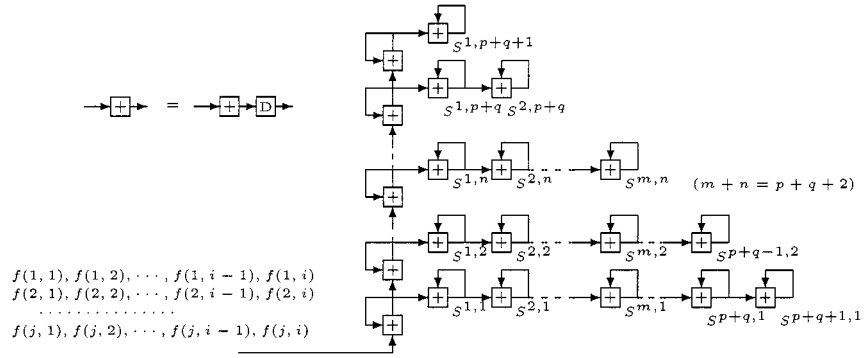


Figure 4. Implementation for $S^{n,m}$.

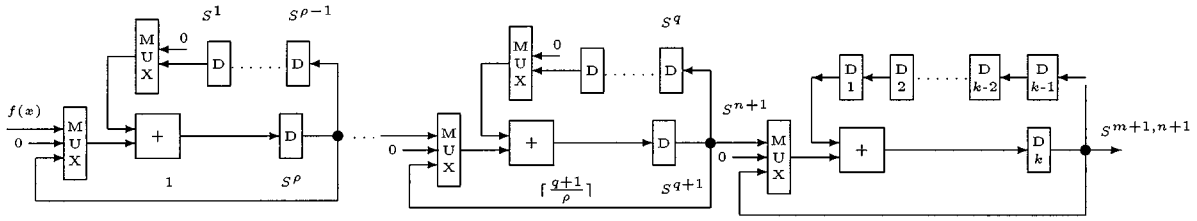


Figure 5. An alternative implementation for $S^{n,m}$.

employing the coefficients B_n^p:

$$
\begin{aligned}
m_{00} &= S_{1,1}^{1,1}; & m_{10} &= S_{1,1}^{2,1} \\
m_{20} &= S_{1,1}^{2,1} + 2S_{2,1}^{3,1}; & m_{30} &= S_{1,1}^{2,1} + 6S_{2,1}^{4,1} \\
m_{01} &= S_{1,1}^{1,2} & m_{02} &= S_{1,1}^{1,2} + 2S_{1,2}^{1,3} \\
m_{03} &= S_{1,1}^{1,2} + 6S_{1,2}^{1,4}; & m_{11} &= S_{1,1}^{2,2} \\
m_{12} &= S_{1,1}^{2,2} + 2S_{1,2}^{2,3}; & m_{21} &= S_{1,1}^{2,2} + 2S_{2,1}^{3,2}
\end{aligned}
$$

where the 10 two-dimensional prefix sums

$$S_{1,1}^{1,1}, S_{1,1}^{2,1}, S_{2,1}^{3,1}, S_{2,1}^{4,1}, S_{1,1}^{1,2}, S_{1,1}^{2,2}, S_{2,1}^{3,2}, S_{1,2}^{1,3}, S_{1,2}^{2,3}, S_{1,2}^{1,4}$$

are obtained from 4 one-dimensional sums (S_1^1, S_1^2, S_2^3, S_2^4). Using form 3), all the coefficients are non-negative and simple (0, 1, 2 and 6), so there is at most one multiply-add in the final calculation. For a 512×512 size image with 8 bit pixel values, the maximum values for S_1^1, S_1^2, S_2^3, S_2^4 have respectively 17, 25, 33 and 40 bits, so 26 bits are used for $S_{1,1}^{1,1}$, 34 bits are needed for $S_{1,1}^{2,1}$, $S_{1,1}^{1,2}$, 42 bits are required for $S_{2,1}^{3,1}$, $S_{1,1}^{2,2}$, $S_{1,2}^{1,3}$ and 49 bits are used for $S_{2,1}^{4,1}$, $S_{2,1}^{3,2}$, $S_{1,2}^{2,3}$, $S_{1,2}^{1,4}$. We can use bit-serial adders and shift registers to realize the real-time moment computation, as shown in Fig. 6.

We also employ the first method for this two-dimensional moment computation. For the first dimensional prefix sums $S^n, n = 1, 2, 3, 4$, since the maximum number of bits required is 40, we split the data into two parts, low-order part (20-bits) and high-order part (20-bits), using 7 units (4 are used for the low-order part, 3 are needed for the high-order part). Note that S^1 needs only 17 bits, so the high-order part is unnecessary here. The structure of the two units which performs low- and high-order part prefix summations is shown in Fig. 7(A).

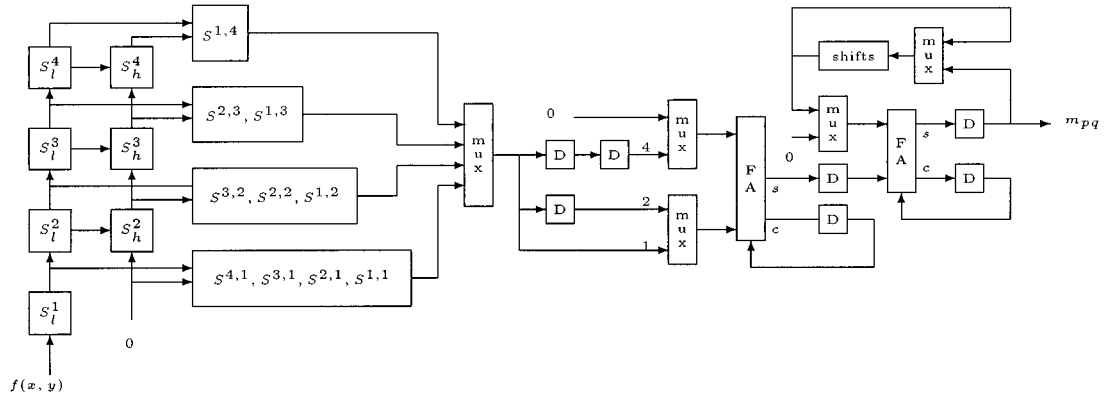


Figure 6. 2-D moment computation using a bit-serial method.

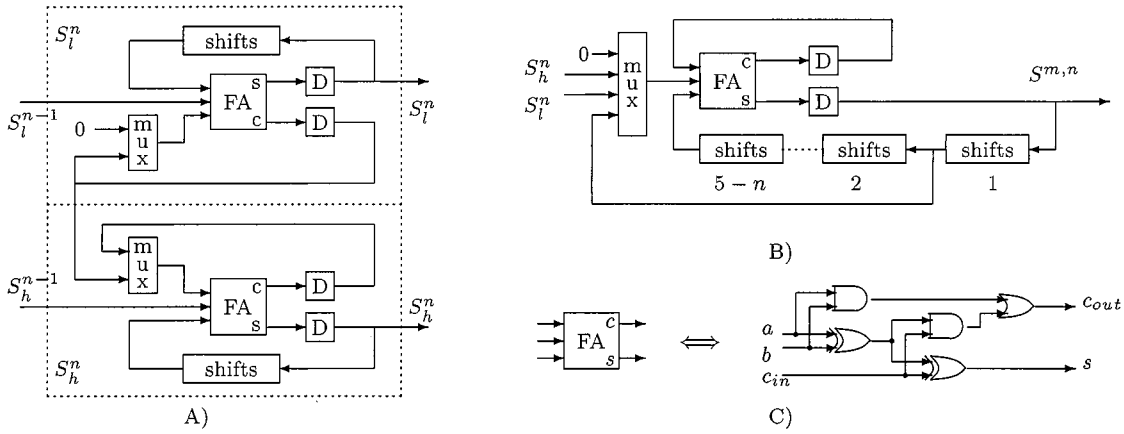


Figure 7. A) A structure for the first dimensional computation of S^n ($n = 1, 2, 3, 4$). B) A structure for the second dimensional computation of $S^{m,n}$ ($m = 0, 2, \ldots, 5 - n$) computation. C) A full adder circuit.

Note that two prefix sums S_2^3, S_2^4 are obtained at $(N-1) \times t$, two other prefix sums S_1^1, S_1^2 are ready at time $N \times t$ (t is the time between two pixels). Also note that t is larger than 100 ns for a 512×512 image at the TV rate. It is thus possible to perform 20 bit-serial additions in 100 ns, since the delay of a full adder (FA) can be less than 5 ns if an internal clock of the order of 200 MHz can be supplied.

Four computing units are used for the second dimensional prefix summations, each consisting of one full adder and $5-n$ ($n = 1, 2, 3, 4$) shift registers, as shown in in more detail in Fig. 7(B). Because the maximum number of bits is 49 for the two-dimensional prefix sums and four prefix sum computations for $n = 1$ are required, it takes about 1 s ($4 \times 49 \times 5$) to perform the second dimensional prefix summations for one line of data. After all data have been input, all the 2-D prefix sums needed are available and the multiplication-additions can be performed to obtain the 10 moments. Since the coefficients for form 3) are simple, a full adder is sufficient to multiply, and another is used to accumulate as shown in the rightmost part of Fig. 6. According to the output order $S_{1,1}^{1,1}, S_{1,1}^{2,1}, S_{2,1}^{3,1}, S_{2,1}^{4,1}, S_{1,1}^{1,2}, S_{1,2}^{1,3}, S_{1,2}^{1,4} S_{1,1}^{2,2}, S_{2,1}^{3,2}, S_{1,2}^{2,3}$, the 10 wanted moments are thus obtained. Here shifters are used to store the values of $S_{1,1}^{2,1}$, $S_{1,1}^{1,2}$ and $S_{1,1}^{2,2}$. Besides some shifters only 13 full adders are needed. Compared with previous implementations for 2-D moment computations, the bit-serial method can save many adders and does not need any multipliers.

Conclusions

The computation of discrete moments has been rewritten in terms of iterated prefix summations, where the prefix sums are easily computed using only adders. Applying the method of prefix summations to the moment computation, we eliminate almost all multiplications required by the definition of the moments. Only a few multiplications with some constant coefficients are needed after the computation of prefix sums for all data has been completed. We derived several alternative forms for the moment computation, expressed in terms of various sets of prefix sums. Moment computations can thus be realized at very high rates, where the computing speed is almost exclusively determined by the speed of the adders. Two-dimensional moments can be thus easily be realized from one-dimensional moment computations. Either by first computing prefix sums or moments along one of the dimensions, and then combining along the second dimension. The hardware requirements for real time image processing then become very minimal, as demonstrated by a bit-serial design of an implementation. Here we need only 13 full adders and some shift registers to implement the 10 third order moment computations ($m_{00}, m_{01}, m_{10}, m_{02}, m_{20}, m_{12}, m_{21}, m_{03}, m_{30}$) for a 512×512 size image at the TV rate.

Our method is computationally equivalent to the algorithm developed by Hatamian [4], but our derivation allows us to obtain several alternative forms for the final moment calculation. For an implementation this allows us to eliminate some delay circuitry needed in Hatamians VLSI realization, where we further note that for two dimensional moment computation the time-critical row prefix summation can also be performed in a redundant representation (e.g., carry-save).

Finally, following Liao and Pawlak, it is possible to reduce the discretization error in approximating the geometric moments, by simply using some alternative coefficients in the computation of the moments from the prefix sums, at no additional computational cost. For image analysis where the data are non-negative pixel values, the coefficients can be chosen such that the computation is numerically stable.

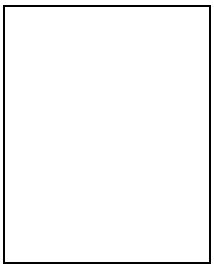
Note

1. Note that here and in the following superscripts will often be used for additional indices in connection with capital letter symbols.

References

1. M.K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Inf. Theory*, vol. IT-8, 1962, pp. 179–187.
2. A.P. Reeves, "A Parallel Mesh Moment Computer," in *Proc. 6th ICPR*, Oct. 1982, pp. 465–467.
3. A.P. Reeve, "Parallel Algorithms for Real-Time Image Processing," in *Multicomputers and Image Processing, Algorithms and Programs*, New York: Academic Press, 1982, pp. 7–18.
4. M. Hatamian, "A Real Time Two-Dimensional Moment Generation Algorithm and its Single Chip Implementation," *IEEE Trans. ASSP*, vol. 34, no. 3, 1986, pp. 546–553.
5. M.F. Zakaria, L.J. Vroomen, P.J.A. Zoombar-Murray, and J.M.H.M. van Kessel, "Fast Algorithm for Computation of Moment Invariants," *Pattern Recognition*, vol. 20, 1987, pp. 634–643.
6. K. Chen, "Efficient Parallel Algorithms for the Computation of Two-Dimensional image Moments," *Pattern Recognition*, vol. 23, 1990, pp. 109–119.
7. W.E. Batchelor, W. Liu, R. Cavin, and S. Chen, "A Bit Level Systolic Array for Real-Time Two-Dimensional Moment

Generation," in *Systolic Array Processor*, J. McCanny et al. (Eds)., 1989, pp. 449–458.
8. B. Bamieh and R.J.P. De Figueiredo, "A General Moment-Invariants/Attributed-Graph Method for Three Dimensional Object Recognition from a Single Image," *IEEE J. Robotics Automation*, vol. 2, 1986, pp. 31–41.
9. N.J.C. Strachan, P. Nesvadba, and A.R. Allen, "A Method for Working Out the Moments of a Polygon Using an Integration Technique," *Pattern Recognition Lett.*, vol. 11, 1990, pp. 351–354.
10. B.-C. Li and J. Shen, "Fast Computation of Moment Invariants," *Pattern Recognition*, vol. 24, no. 8, 1991, pp. 807–813.
11. X.Y. Jiang and H. Bunke, "Simple and Fast Computation of Moments," *Pattern Recognition*, vol. 24, no. 8, 1991, pp. 801–806.
12. S.X. Liao and M. Pawlak, "On Image Analysis by Moments," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 3, 1996, pp. 252–265.



Feng Zhou was born in Hangzhou, Zhejiang, PRC, in 1961, and received the Ph.D. degree in 1991 from Zheijiang University, Hangzhou. While serving with the Department of Information and Electronic Engineering at the same university, he spent a leave in 1994 and 1995 as a post.doc. with Odense University, Denmark. On return from this leave he was designated Director of the Signal Processing Lab. His research interests are in image compression, coding, signal and image processing, computer vision and information theory.

Peter Kornerup was born in Aarhus, Denmark, 1939. He received the mag.scient. degree in mathematics from Aarhus University, Denmark, in 1967. After a period with the University Computing Center, from 1969 involved in establishing the computer science curriculum at Aarhus University, he helped found the Computer Science Department in 1971, and through most of the 70's and 80's he served as chairman of the department. He spent a leave during 1975/76 with the University of Southwestern Louisiana, Lafayette, LA, and another in 1979 with Southern Methodist University, Dallas, TX. Since 1988 he has been Professor of Computer Science at Odense University, Odense, Denmark, where he has also served as the chairman of its Department of Mathematics and Computer Science. His research interests include compiler construction, computer networks and computer architecture, but in particular computer arithmetic and number representations. Prof. Kornerup has served on the program committees for a number of IEEE and ACM sponsored meetings, in particular he has been on the Program Committee of the 4th through the 14th IEEE Symposium on Computer Arithmetic, and served as Program Chairman for these symposia in 1983, 1991 and for the 14th in 1999. He was an associate editor of the IEEE Transactions on Computers during 1991–95, and is a member of the IEEE.