FAST LOCAL IMAGE PROCESSING ALGORITHMS USING
RECURSIVE COMPUTATION

R. M. Palenichka and A. Yu. Lutsyk

INTRODUCTION

Most practical computer algorithms are locally homogeneous, i.e., these algorithms produce the result by evaluating at each point the same local function on different data. Classical examples of such algorithms are linear filtering with a finite impulse characteristic, where the local function is linear [1], and median filtering, where the local median is a particular case of an order statistic of the image elements inside a local processing window [2]. Local adaptive algorithms are somewhat more complex, but they use the same operations; in these examples, they are particular cases.

Digital image processing algorithms in their basic form are highly complex and computationally costly. Essential acceleration is achieved by the use of parallel computation, but this requires parallel computer systems, which in turn involve high-cost hardware. It is therefore relevant to consider the reduction of computational effort in standard image processing algorithms and the development of fast algorithms for this purpose. Fast algorithms reduce the image processing time in sequential implementation and the hardware costs (the number of parallel processors) in parallel implementation.

Several approaches are available to the design of fast image processing algorithms [3]. Most approaches focus on efficient implementation of linear filtering [3, 4]. The increasing popularity of median filtering for image processing has stimulated the development of fast image processing algorithms in this category also [5-7]. However, the available algorithms are usually particular algorithms, which are capable of performing only one kind of image processing. This is typical, for instance, of linear and median filtering, where the existing fast algorithms implement only one particular case of filtering. A topical problem is therefore the development of a general design procedure for fast local image processing algorithms.

This paper follows the general approach of [8, 9], which use recursion to obtain fast image processing algorithms. Fast implementation of linear image filtering is considered in [8], and specific cases of image smoothing and edge detection by linear filtering with an exponential impulse characteristic are considered in [9]. The technique of [9] leads to fast preprocessing algorithms in which the number of operations is independent of the size of the local processing window.

ELIMINATION OF REDUNDANT OPERATIONS IN LOCAL IMAGE PROCESSING

Let us consider one of the main approaches to the design of fast local image processing algorithms, i.e., elimination of redundant operations and use of spatial recursion. Suppose that local processing of the image \(g(i, j)\) at the point \((i, j)\) produces some known function \(f(i, j) = F(g(i, j))\), where \(F(\cdot)\) is defined for all \((i, j) \in W(i, j)\), \(W(i, j)\) is the given image processing window. Using the idea of recursive evaluation of a local function, we can represent a whole class of fast local image processing algorithms (more precisely, their computational core) in the form of the recursive relationship

\[
F(g(i, j)) = \Phi(g(m, n), F(g(i, j - 1))),
\]

where \(\Phi(\cdot)\) is a known function of the result \(f(i, j - 1)\) from the previous window \(W(i, j - 1)\) and of the image elements \(\{g(m, n)\}\), where \((m, n) \in W(i, j) \cup W(i, j - 1)\). We can stipulate that the computational complexity of evaluation of \(\Phi(\cdot)\) is at least an order of magnitude lower than the complexity of evaluation of \(F(\cdot)\), i.e., the ratio of their computational complexities is...
Given the diversity and complexity of local image processing algorithms, a general explicit form of representation (1) does not exist. Some considerations may lead to a (constructive) definition of \( \Phi(\cdot) \) in explicit form, which we first consider for the one-dimensional case, i.e., row (column) processing of an image. The current image element is evaluated in an \( N \)-point one-dimensional window (segment) \( W(j) \), which differs from the preceding window \( W(j - 1) \) by two points: \( j - (N + 1)/2 \) and \( j + (N - 1)/2 \), where \( N \) is odd (Fig. 1). A subset of points \( \{(m,n) \mid (m,n) \in W(j) \cap W(j - 1)\} \) is identified in the window \( W(j) \). Then the operations between image elements are reduced to the operations that are used in evaluating the local function \( F(j) \) in the window \( W(j - 1) \). After that the operations associated with the element \( g(j - (N + 1)/2) \) are adjoined, so that the result is equivalent to the evaluation of \( \Phi(\cdot) \) in the window \( W(j - 1) \). Finally, to obtain \( \Phi(\cdot) \) in explicit form, we adjoin the operations that use the element \( g(j + (N - 1)/2) \) in the remaining operations. In certain cases, this procedure may produce a single formula which hopefully consists of arithmetic operations only.

In two-dimensional image processing, the difference between \( W(i, j) \) and \( W(i, j - 1) \) is by whole columns of image elements, assuming an \( L \times L \) square window (Fig. 2). This substantially increases the computational costs compared with the one-dimensional case (sometimes by as much as an order of magnitude). However, with a square (or rectangular) window image processing can be accelerated if we combine the column operations and apply the method of recursive function evaluation to a one-dimensional window \( W(i) \) — a single column. In this case, we can achieve substantial elimination of redundant operations, but additional memory will be required for storing the results of intermediate computations.

Often the local processing window \( W(i, j) \) is not rectangular (Fig. 3 shows disk and cross shaped windows). Then these windows can be (optimally) approximated by a rectangular window. This technique substantially simplifies image processing when it is required to determine maximum likeness with an object that may be rotated through an arbitrary angle. With a small angle of rotation, a window that matches the shape of the object for the given angle is close to the window obtained for the next value of the rotation angle. Therefore, successively computing the measure of likeness for a sequence of rotation angles, we substantially reduce (at least by an order of magnitude) the number of operations (Fig. 4).

The above considerations regarding fast algorithms have been presented for the case of so-called spatial (planar) recursion. However, recursion can be implemented with respect to the time variable. Then the local processing algorithm using (first-order) temporal recursion is represented in the form

\[
\Phi(i, j, k) = \Phi(g(i, j), f(i, j, k - 1)),
\]

not less than \( O(L) \), where \( L \times L \) is the size of the window \( W(i, j) \). Otherwise, there is no point in using this algorithms, because it does not produce a significant cost reduction.