Min-Space Integral Histogram

Séverine Dubuisson and Christophe Gonzales
Laboratoire d’Informatique de Paris 6,
Université Pierre et Marie Curie
{severine.dubuisson,christophe.gonzales}@lip6.fr

Abstract. In this paper, we present a new approach for quickly computing the histograms of a set of unrotating rectangular regions. Although it is related to the well-known Integral Histogram (IH), our approach significantly outperforms it, both in terms of memory requirements and of response times. By preprocessing the region of interest (ROI) computing and storing a temporary histogram for each of its pixels, IH is effective only when a large amount of histograms located in a small ROI need be computed by the user. Unlike IH, our approach, called Min-Space Integral Histogram, only computes and stores those temporary histograms that are strictly necessary (less than 4 times the number of regions). Comparative tests highlight its efficiency, which can be up to 75 times faster than IH. In particular, we show that our approach is much less sensitive than IH to histogram quantization and to the size of the ROI.

Keywords: Integral histogram, multiple histogram computation.

1 Introduction

The complex nature of images implies that a large amount of data needs to be stored in histograms (colors, edges, etc.), hence making their computations time consuming. In many applications, large sets of histograms need be computed frequently, hence making it a computational bottleneck. This explains why fast histogram computation has received some attention in the literature [1284].

An image yields a distribution over a color space by mapping each of its pixels into its color. The histogram $H$ of an $N \times M$ image $\mathcal{I}$ is defined by $H(k) = \sum_{x=1}^{N} \sum_{y=1}^{M} \{ \mathcal{I}(x,y) = k \}$, where $\mathcal{I}(x,y)$ is a pixel, $k = 0, \ldots, K - 1$ is its value (in this paper, this is a color value, but gray intensities, gradient orientations, etc., could also be considered). Binning the probability distribution induced by $H$ is a way to summarize it, and the applied quantization (bin size) controls the rate of summarization. In such a case, histogram $H$ is divided into $B$ bins $b = 0, \ldots, B - 1$, and is defined by:

$$H(b) = \sum_{x=1}^{N} \sum_{y=1}^{M} \left\{ \mathcal{I}(x,y) \in \left[ \frac{bK}{B}, \frac{(b+1)K}{B} \right] \right\}.$$ 

The classical approach to compute an histogram of a $w \times h$ region $R$ consists of browsing all its pixels, hence yielding a time complexity of $O(wh)$.
In this paper, we propose a novel approach to speed-up multiple histogram computation by reducing computational redundancies. We will show that this new method outperforms the well-known Integral Histogram (IH) both in terms of memory requirements and of response times.

Our approach requires that the ROI in which the histograms need be computed is known in advance, that it is still applicable to some major problems of interest in computer vision, especially in visual object tracking methods that generate multiple tracking hypotheses for each frame, where an hypothesis corresponds to an unrotated rectangular patch in the frame. This includes, for instance, grid-based localization methods as well as particle filters [5], and also some recent approaches that use sets of fragments to model objects [6,7,8,9]. Actually, our motivation for developing Min Space Histograms (MSIH) was particle filtering-based tracking.

The paper is organized as follows. Section 2 first presents a short overview of the main approaches that have been proposed in the literature to speed-up histogram computation. In particular, it recalls IH’s approach. Section 3 then describes our new approach: we first present the overview of the method and, then, we detail it, including its time and space complexity. Section 4 gives some comparative results for the computation of a set of histograms, both in terms of response times and memory requirements. Three approaches are compared; the classical one, IH and our method. Finally, concluding remarks are given in Section 5.

2 Fast Histograms Computations

The classical approach would certainly be sufficient for practical applications did the latter need computing only very few histograms. Unfortunately, in practice, applications often have to repeatedly compute large sets of histograms and, in those cases, a more efficient approach is compulsory to get admissible response times. To achieve this, it can be observed that the rectangular regions where histograms are computed often overlap, thus inducing some redundancies. Exploiting the latter is the key idea underlying fast histograms computation.

One of the first works in this direction was proposed in [1]. In the context of image filtering (median filter). Considering the histogram $H_R$ of a region $R$, that of another region $Q$ is computed by removing from $H_R$ the histogram of the pixels that belong to $R$ but not to $Q$ and adding that of the pixels that belong to $Q$ but not to $R$. This approach can be very efficient when the two regions considered have a large intersection. Similar in spirit, the method proposed in [2] breaks up region $R$ into the union of its columns in the image, and all the column histograms are kept up to date in constant time using a two-step approach. In [3], the authors propose the distributive histogram based on a distributive property of disjoint regions combined with a per-column histogram maintenance and a row-based update of these column histograms. This approach can be easily extended to cope with non-rectangular regions and multi-scale processing.

When massive amounts of histograms need be computed, IH [4] proves to be particularly effective and, actually, it is now used in many practical appli-