Neurocomputing 118 (2013) 141–149

Contents lists available at SciVerse ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Efficient computation of histograms on densely overlapped polygonal regions

Yuting Zhang [a], Yueming Wang [b,*], Gang Pan [a], Zhaohui Wu [a]

[a] Department of Computer Science, Zhejiang University, Hangzhou 310027, PR China
[b] Qiushi Academy for Advanced Studies, Zhejiang University, Hangzhou 310027, PR China

ARTICLE INFO

Article history:
Received 18 September 2012
Received in revised form
31 January 2013
Accepted 20 February 2013
Communicated by Qingshan Liu
Available online 28 March 2013

Keywords:
Polygon
Efficient computation
Histogram
Incremental computation

ABSTRACT

This paper proposes a novel algorithm to efficiently compute the histograms in densely overlapped polygonal regions. An incremental scheme is used to reduce the computational complexity. By this scheme, only a few entries in an existing histogram need to be updated to obtain a new histogram. The updating procedure makes use of a few histograms attached to the polygon's edges, which can be efficiently pre-computed in a similar incremental manner. Thus, the overall process can achieve higher computational efficiency. Further, we extend our method to efficiently evaluate objective functions on the histograms in polygonal regions. The experiments on natural images demonstrate the high efficiency of our method.

1. Introduction

Histogram is one of the most frequently used techniques to encode image features, e.g., histograms of oriented gradient (HoG) [1], histograms of color intensity [2], local binary pattern (LBP) [3], scale-invariant feature transform (SIFT) [4], and bag of visual words [5]. These features are used for detection [1,6], recognition [7], image classification [8], tracking [9], flow estimation [10], image retrieval [11], face liveliness detection [12], and many other vision tasks. In some of these applications, histograms are computed in the neighborhood regions around keypoints [4] that are sparsely located on images. In other cases, histograms are computed in densely overlapped regions [1,6] that are organized in regular grids. The keypoint-based paradigm of feature extraction is widely used for images [4] and even for time series [13] due to its high efficiency that attributes to the sparsity of the keypoints. In contrast, the grid-based paradigm can produce richer image features with a much higher computational cost. Generally, the feature richness can significantly benefit the solution of vision tasks [10], such as improving the accuracy rate of the detection and recognition applications. To make practical use of the richer image features produced by the grid-based paradigm, improving the efficiency of histogram computation in densely overlapped regions is very important.

There are several algorithms in the literature to efficiently compute the image histograms in densely overlapped regions. Porikli et al. [14] applied the integral image [15] to histogram computation. When the number of histogram bins is much smaller than the number of pixels in the region, better efficiency was obtained with a high memory cost. Otherwise, it is even worse than the simplest method. Sizintsev et al. [16] utilized the overlap between regions to reduce redundant computation. Wei et al. [17] presented a similar idea and extend it to the function evaluation on histograms, making the evaluation more scalable to the bin number and the region size, which are usually large in computer vision problems.

Most existing algorithms compute histograms in rectangular regions. However, many real-world objects show diverse shapes and poses. For example, the whole shape of an airplane is irregular. The wings are triangular, the engines are oval, and the windows appear to be parallelograms in most poses. Thus, the image features often include noisy background or lose necessary foreground information if we merely use rectangular regions. Hence, features extracted in rectangular regions are insufficient to describe images well in many circumstances.

In order to obtain more discriminative features, it is necessary to compute the histograms in non-rectangular regions. Pham et al. [18] used the triangular integral image to efficiently compute polygonal Haar-like features, and demonstrated that features in suitable non-rectangular local regions can better describe the visual characteristics of images. Compared with the Haar-like feature, the histogram in the region with a suitable shape can be more powerful and discriminative in describing visual information. The triangular integral image can be straightforwardly generalized for histogram computation, nevertheless it does not scale well on the histogram bin number as [14] did on rectangular regions. So far, to efficiently

* Corresponding author. Tel.: +86 571 87952141.
E-mail addresses: zyt@zju.edu.cn (Y. Zhang), ymingwang@gmail.com (Y. Wang), gpan@zju.edu.cn (G. Pan), wzh@zju.edu.cn (Z. Wu).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.02.027

compute histograms in non-rectangular regions is still an unsolved problem, and there is no work to integrate efficient function evaluation into histogram computation for non-rectangular regions.

This paper proposes a novel algorithm to address the problem. We focus on the case of computing histograms in polygonal regions, and consider the typical situation in object detection and image retrieval as well as in many cases of other tasks, in which each polygon is required to slide along the rows and columns of images, thus generating a large number of densely overlapped polygonal regions. Given two overlapped regions, if the histogram in one region is computed, we update it according to two regions' non-overlapped areas to obtain the histogram in the other one. The non-overlapped areas is broken into smaller ones. The histogram in each smaller area can be pre-computed efficiently. In addition, we extend this incremental scheme to the problem of evaluating objective functions on the histograms in polygonal regions.

Wei et al.'s method [17] for rectangular regions can be taken as a special case of ours. Compared with it, our method is designed with more decent efforts on constructing the incremental computation strategy, handling the computational procedure, and devising more efficient data structure. The proposed method has no strong connection with Pham et al.'s triangular integral image [18]. However, their idea of decomposing computation according to the edges of a polygon gives us inspiration.

We compare the proposed algorithm with several other methods, including the most straightforward method, the generalization of the triangular integral image, and a simplified variation of our method. The experimental results show that the proposed algorithm is computationally more efficient.

2. Basic idea

Given a $W \times H$ index bitmap with B indices, we represent it as a function $A(x,y)\in\{1,2,\ldots,B\}$ for $x\in\{1,2,\ldots,W\}$ and $y\in\{1,2,\ldots,H\}$. By locating the left-top corner of the index bitmap at the origin of $\mathbb{R}^2$ (y-axis pointing down), A is generalized to the continuous domain as

$$a(x,y)=\begin{cases} A(\lceil x\rceil,\lceil y\rceil), & 0<x\le W, 0<y\le H,\\ 0 & \text{otherwise,}\end{cases} \tag{1}$$

where $(x,y)\in\mathbb{R}^2$, $\lceil\cdot\rceil$ is the ceiling function, and the pixel (x,y) on A is the unit square region $(x-1,x]\times(y-1,y]$ on $\mathbb{R}^2$. Given a region $\Phi\in\mathbb{R}^2$, we denote the histogram of a in it as

$$\mathbf{c}(\Phi)=(c_1(\Phi),c_2(\Phi),\ldots,c_B(\Phi)), \tag{2}$$

where,

$$c_b(\Phi)=\int\int_{\Phi}\iota_b(a(x,y))\,dx\,dy, \tag{3}$$

$\iota_b(t)=1$ if $t=b$, and $\iota_b(t)=0$ if $t\neq b$.

For two regions Φ and Φ', the following holds,

$$\mathbf{c}(\Phi')=\mathbf{c}\Phi+\delta\mathbf{c}\Phi',\Phi, \tag{4}$$

where,

$$\delta\mathbf{c}(\Phi',\Phi)=\mathbf{c}(\Phi'\backslash\Phi)-\mathbf{c}(\Phi\backslash\Phi') \tag{5}$$

is the *overall residual* histogram, and "\" denotes set subtraction. As shown in Fig. 1a, if the region Φ' is obtained by moving Φ a little rightward, Φ and Φ' have much overlap. $\Phi'\backslash\Phi$ is the newly emerged region in Φ' which is called the *positive residual* in this paper. Similarly, $\Phi\backslash\Phi'$ is the *negative residual*. According to (4), if $\mathbf{c}(\Phi)$ is known, $\mathbf{c}(\Phi')$ can be computed from $\mathbf{c}(\Phi)$ by adding only the non-zero entries of $\delta\mathbf{c}(\Phi',\Phi)$. Indeed, the number of these non-zero entries is usually very small compared with the bin number of $\delta\mathbf{c}(\Phi',\Phi)$. This is true if the number of pixels inside $\Phi'\backslash\Phi$ and $\Phi\backslash\Phi'$ is smaller than the bin number. More importantly, we observed that nearby pixels in index bitmaps used for visual feature extractions usually have the same value with high probability. This phenomenon attributes to the local smoothness of the natural images, from which the index bitmap is obtained. As a result, no matter how large $\Phi'\backslash\Phi$ and $\Phi\backslash\Phi'$ are, $\delta\mathbf{c}(\Phi',\Phi)$ is usually very sparse when the movement between Φ and Φ' are not too large. Hence, the incremental computation scheme in (4) helps to improve the efficiency in computing $\mathbf{c}(\Phi')$ and further improve the overall computational efficiency of all histograms on an index bitmap.

It is worth pointing out that, in many cases (e.g. the computation of HoG [1]), different pixels on the index bitmap are associated with different weights. Let $\gamma(x,y)$ be the bitmap of pixel-wise weights. Eq. (3) then becomes $c_b(\Phi)=\int\int_{\Phi}\gamma(x,y)\,d_b(a(x,y))\,dx\,dy$. In this case, (4) still holds, so our algorithm also works with index bitmaps of weighted pixels.

The idea of incremental filtering has a long history. It appeared early in [19], and was surveyed by [16] for histogram computation in rectangular regions. In [17], the importance of local smoothness on incremental histogram computation was demonstrated. Here, we extend these idea to the case of non-rectangular regions.

3. Incremental histogram computation with residual segmentation

The computation efficiency of (4) is determined by both the nature of index bitmaps and how efficiently the overall residual histogram $\delta\mathbf{c}(\Phi',\Phi)$ can be computed for overlapped regions Φ and Φ'. In this section, we present an efficient algorithm for the case where Φ and Φ' are of the same polygonal shape. We show that the residuals between two polygonal regions of the same shape, P and P', can be broken into several parts. Each part is attached to one edge and called the edge residual. The computation of $\delta\mathbf{c}(P',P)$ then reduces to the computation of the histogram in every edge residual, which can be pre-computed by a similar incremental scheme as that in (4).

3.1. Edge residuals

Given a polygonal region P represented by a vertex list $v_1v_2\ldots v_N$, the ith edge v_iv_{i+1} ($v_{N+1}=v_1$) is *left-side/right-side* if its left/right side is P's exterior, or *horizontal* if it has a zero slope. Let $v_i=(x_i,y_i)$, and $v_1v_2\ldots v_N$ be clockwise ordered, v_iv_{i+1} is left-side if and only if $y_i>y_{i+1}$, and is right-side if and only if $y_i<y_{i+1}$.

As shown in Fig. 1b, we move P horizontally a little rightward with a given stride and obtain a new polygonal region $P'=u_1u_2\ldots u_N$. For each non-horizontal edge v_iv_{i+1}, an edge residual $\Delta(v_i,v_{i+1})$ is formed by $v_iv_{i+1}u_{i+1}u_i$. The edge residuals attached to right-side edges are parts of the positive residual $P'\backslash P$, and those attached to left-side edges are parts of the negative residual $P\backslash P'$. These edge residuals constitute the overall residual between P and P'. Thus, the over residual histogram $\delta\mathbf{c}(P',P)$ can be computed by

$$\delta\mathbf{c}(P',P)=\sum_{i=1}^{N}(-1)^{\omega_i}\mathbf{c}(\Delta(v_i,v_{i+1})), \tag{6}$$

where $\omega_i=1$ if v_iv_{i+1} is a left-side edge, otherwise $\omega_i=0$. Note that the overlap between edge residuals always happens in a left-side edge and a right-side one, so it is neutralized in the computation of $\delta\mathbf{c}(P',P)$.

The decomposition scheme used in (6) has a loosely analogical relationship with that of image integration in [18], where the integration in a polygonal region is decomposed into the summation of those in several triangular regions attached with edges. Nevertheless, while the decomposition scheme in [18] is tailored