### Hash-Based Linked-List Histogram Construction

**Yan-Tsung PENG†, Student Member, Fan-Chieh CHENG†, Shanq-Jang RUAN†, and Chang-Hong LIN†, Members**

**SUMMARY** A histogram is a common graphical descriptor to represent features of distribution of pixels in an image. However, for most of the applications that apply histograms, the time complexity of histogram construction is much higher than that of the other parts of the applications. Hence, column histograms had been presented to construct the local histogram in constant time. In order to increase its performance, this letter proposes a linked-list histogram to avoid generating empty bins, further using hash tables with bin entries to map pixels. Experimental results demonstrate the effectiveness of the proposed method and its superiority to the state-of-the-art method.

**key words:** image histogram, linked list, hash-based

1. **Introduction**

In image processing and computer vision applications, the image histogram is such a key descriptor that many efforts had been extensively studied [1]–[3]. There are three histogram construction methods, including histogram edge modification (HEM) [1], integral histogram computation (IHC) [2], and column histograms junction (CHJ) [3] methods.

Let \( w \) be the width of a square image and \( r \) be the radius of local histogram, and Table 1 shows a comparison of the complexities of histogram construction methods. The HEM method [1] allocated \( S(r^2) \) space for the intersection area between two consecutive pixels to construct the next histogram. Based on the integral image, an integral histogram was generated per pixel to compute local histogram in constant time [2]. To reduce the memory size, the histograms of neighbours are separated to several disjoint column histograms for updating [3].

Figure 1 snapshots the use of distributive column histograms [3]. Let \( P \) be the path of all neighbours of the visited pixel. The corresponding histogram \( T \) can be disjointed as

\[
T(P) = T(p_1 \cup p_2 \cup p_3 \cdots \cup p_d) = \sum_{i=1}^{d} T(p_i),
\]

where \( d \) is path diameter and \( p_i \) is a disjointed column shown in Fig. 1 (a). For updating \( T(p_i) \) per row, the new pixel is added and the old pixel is removed. Similarly, local histogram can be easily modified with adding the next of the rightmost column histogram and removing the leftmost column histogram shown in Fig. 1 (b). Let \( f \) be the input image of size \( w \times h \), \( K \) be the local histogram per pixel, \( D_i \) be the \( i_{th} \) column histogram, and \((x,y)\) be the pixel index. After initializing \( K \) and \( D_i \), the CHJ method is shown below:

1. \( \text{for } y \leftarrow 1 \text{ to } h \) do
2. \( \text{for } x \leftarrow 1 \text{ to } w \) do
3. \( \text{Add } I(x, y + r) \) to \( D_x \)
4. \( \text{Subtract } I(x, y - r - 1) \) from \( D_x \)
5. \( \text{Add } D_{x+r} \) to \( K \)
6. \( \text{Subtract } D_{x-r-1} \) from \( K \)
7. \( \text{end for} \)
8. \( \text{end for} \)

For updating \( D_x \), one addition and one subtraction are needed. To update \( K \), 256 additions and 256 subtractions are needed. It is easily observed that updating \( K \) can be further optimized. Hence, the proposed method not only avoids visiting empty bins in the histogram construction by utilizing the linked-list structure but expedites the indexing of the linked list by using the hash table.

2. **Proposed Method**

Figure 2 (b) shows the main idea of the proposed method. The proposed data structure for a histogram basically includes one hash table and one linked list. Figure 2 (a) shows that the bins should be visited \( L \) times, while the statistical gray levels can be counted and stored to reduce the times of visiting bins (\( \leq L \)) by using the proposed method.

---

**Table 1** Comparison of computation and memory costs.

| Method      | HEM [1] | IHC [2] | CHJ [3] |
|-------------|---------|---------|---------|
| Time complexity | \( O(r) \) | \( O(1) \) | \( O(1) \) |
| Space complexity  | \( S(r^2) \) | \( S(w^2) \) | \( S(w) \) |

---

*Manuscript revised February 4, 2013.

†The authors are with the Department of Electronic and Computer Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan.

a) E-mail: sjruan@mail.ntust.edu.tw

DOI: 10.1587/transinf.E96.D.1204

Copyright © 2013 The Institute of Electronics, Information and Communication Engineers
Let $DL$ be column linked lists and $H$ be hash tables. After initializing $K$, $DL$, and $H$, the pseudo code of the proposed method is shown below:

1: for $y ← 1$ to $h$ do  
2:     for $x ← 1$ to $w$ do  
3:         if $I(x, y + r) \neq I(x, y - r - 1)$ then  
4:             Add $I(x, y + r)$ to $DL_x$ by indexing $H_x$  
5:             Subtract $I(x, y - r - 1)$ from $DL_x$ by indexing $H_x$  
6:         end if  
7:     end for  
8:     Add $DL_{x+r}$ to $K$  
9:     Subtract $DL_{x-r-1}$ from $K$  
10: end for

As the new and past pixels are identical, the column histogram does not need to be updated so that the column linked list can be selectively adjusted.

For each column pixel, a hash table containing 256 slots to store node pointers is allocated. As the various bins become zero, the elements of the hash table are set to NULL. With perfect hashing used, the hash function maps each possible gray level to a unique slot where a corresponding node stores in each linked list. The reduced computation cost of updating $K$ for each pixel is

$$K_{\text{cost}} = 2C(x), \quad (2)$$

where $C(x) \leq 256$ is the number of nodes in $DL_x$.

3. Experimental Results

This letter uses six standard gray-scale images to test the performance of the proposed linked-list CHJ (LCHJ) method. Note that all test images have $256 \times 256$ pixels with different numbers of gray-levels. The programs are written in C with the compiler, “gcc 4.7.0” in a PC with a 2.66 GHz processor and a 2 GB memory for testing.

In our experiment, the radius is set to 31 without losing generality. Let $\#\{C(x)\}$ be the average number nodes per column in the LCHJ method and $(\neq\%)$ be the percentage of $I(x, y + r) \neq I(x, y - r - 1)$ in the whole image. Table 2 lists execution time (ms) for the CHJ [3] and LCHJ methods. For all test images, CHJ method uses 0.25 MB memory in total, while LCHJ method uses a larger memory, from 0.33 to 0.37 MB. In average, the LCHJ method almost reduces 60% computation cost for histogram construction with only 40% additional storage needed.

4. Conclusion

In this letter, we review the distributive column histogram construction, and proposed the hash-based linked-list data structure to accelerate it. Without visiting empty bins, the proposed method improves the performance of the histogram construction. In summary, the use of linked lists significantly reduces the executive time since the sparse distribution of the gray levels usually exists in nature images. Experimental results show that time consumption of the proposed method is much lower than the compared method, while the memory used is only moderately increased. In the future work, the proposed method can be further improved using SIMD or other parallel implementations as the column histogram structure is still maintained.

Acknowledgement

This work was supported by the National Science Council of ROC under Contract NSC101-2221-E-011-164-MY3.

References

[1] T.S. Huang, G.J. Yang, and G.Y. Tang, “A fast two-dimensional median filtering algorithm,” IEEE Trans. Acoust. Speech Signal Process., vol.ASSP-27, no.1, pp.13–18, Feb. 1979.
[2] F. Porikli, “Integral histogram: A fast way to extract histograms in Cartesian spaces,” Proc. IEEE 2005 Int. Conf. Comput. Vis. Pattern Recognit. (CVPR), San Diego, CA, USA, pp.829–837, 2005.
[3] S. Perreault and P. Hebert, “Median filtering in constant time,” IEEE Trans. Image Process., vol.16, no.9, pp.2389–2394, Sept. 2007.