International Journal of Multimedia and Ubiquitous Engineering
Vol.9, No.8 (2014), pp.335-346
http://dx.doi.org/10.14257/ijmue.2014.9.8.31

Block-Based Sparse Integral Histogram with Interpolation

JinYong Park, Hyunki Hong and TaeYong Kim

GSAIM, Chung-Ang University, Seoul, Republic of Korea
jusias@nate.com, honghk@cau.ac.kr, kimty@cau.ac.kr

Abstract

Integral Histogram is widely used among various real-time algorithms due to its computationally efficient facility to compute histograms from an image. However, its advantage is greatly diminished for algorithms requiring only a few histograms since required initialization operation takes up majority of total running time. In this paper we propose a block-based integral histogram that drastically reduces initialization time, a major weakness of integral histograms, and suggest interpolation methods to generate more reliable histogram for the location whose histogram is not available. Experimental results show that the proposed method is fast and accurate to apply real-time applications.

Keywords: *Integral histogram, interpolation, quad-tree structure, block-based histogram.*

1. Introduction

Wide varieties of algorithms in image processing field make use of histograms. Many approaches in computer vision require multiple comparisons of histograms for rectangular patches of an input image. In such approaches, we dispose a reference histogram and try to find the region of the current image whose histogram is the most similar, that can be very time consuming, and may also need a lot of information to store. Obviously a big deficit in systems utilizing exhaustive histogram retrieval is that the cost for the intersection is dependent on the size of the patch and grows proportionally to the area. A fast way to compute histograms in terms of time computation is the Integral Histogram [1], which is now used in many applications needing massive histogram computations. The advantage of the integral histogram approach is the dramatic improvement on the cost of retrieval of histograms for rectangular regions. Once Integral Histogram (IH) has been computed over all pixels, we also can derive any histogram of a sub-region only using four elementary operations.

IH is the best in the sense that it requires low computation times and is flexible enough to adapt many applications, especially in recent tracking algorithms. Recognizing human actions in video sequences is attracting much attention, and Yi and Lin [10] aims to deal with the problem of action recognition with salient trajectories. To efficiently obtain the kernel histogram of a rectangle area, the integral histogram technique is adopted. To detect pedestrians on a movement feature space the orientation histograms can also be computed quickly using the integral histogram [11], which offers the possibility of accelerating the detection stage. Chai [2] applied such technique to Particle filter to implement real-time user interface, and Zhang [5] improved computing time of stereo matching using joint integral histograms. It is also used other areas such as Adaboost [4] or Bilateral Filtering [6].

Because IH needs computationally expensive initialization operation, it is efficient only if more than certain numbers of histograms are extracted. So, algorithms that need small number of histograms such as particle filter are not able to gain much from such technique. Many researchers have been carried out to

ISSN: 1975-0080 IJMUE

reduce initialization time and improve algorithms that uses only limited amount of histograms. Thomas Muller [3] presented an initialization method by grouping pixels into blocks. However, the algorithm introduces substantial amount of errors because of approximation during initialization. Furthermore, although errors during histogram generation are inevitable since the algorithm neglects data from some pixels, errors are amplified further by inexact calculation employed in the algorithm.

In this paper we propose a method of block-based sparse integral histogram (BSIH) to reduce the initialization time and suggest the interpolation method for the BSIH, which includes expansion of the key ideas and supporting experimental results over previous work [14]. Pixels with similar color values are grouped and integrated by a representative value in sparse IH, which is much faster than the initialization of the conventional IH that compute histograms for all pixels in an image. To reduce the artifact by blocking we use the quad-tree data structure for an image [7], which is efficient to manage the histogram data. Since there is no histogram data for a pixel inside a block, we propose interpolation methods to estimate the IH of a pixel using the nearest nodes that have already computed IH. Experimental results show that the initialization of proposed BSIH is fast and its interpolation is accurate enough to apply real-time applications like the particle filter or the stereo matching.

The rest of paper is organized as follows. In Section 2, we describe the overview of integral histogram and its application. In Section 3, we present the block-based sparse integral histogram with interpolation methods and section 4 shows the experimental results for the performance and the accuracy of the proposed method. Conclusions are shown in Section 6.

2. Integral Histogram and Application

Integral Histogram is a method proposed by F. Porikli[1] in 2005. The idea of Integral Histogram will be more easily graspable from Integral Image. Integral Image creates a table with sums of all pixels above and to the left of each pixel in the original image, which allows retrieval of specific template information just through addition and subtraction at relevant location. In case of Image Histogram, it instead stores and uses frequency data rather than pixel values as in Integral Image case.

The value of Integral Histogram image at each pixel is:

$$H(x_1, y_1, b) = H(x_1 - 1, y_1, b) + H(x_1, y_1 - 1, b) - H(x_1 - 1, y_1 - 1, b) + Q(f(x_1, y_1)) \quad (1)$$

where x_1, y_1 specifies a pixel, b represents bin. f is the value of corresponding pixel, and Q is indicator function that is defined to be 1 if pixel is applicable to bin in question. To compute value of integral histogram H for a pixel, one can add value of pixel immediately above and pixel to the left; subtract the data from upper-left pixel; and add output of indicating function for the pixel in question. The Integral Histogram image is populated with each pixel's integral histogram value. Then, histogram can be obtained from the image by following:

$$h(x_1, y_1, x_2, y_2, b) = H(x_2, y_2, b) - H(x_1 - 1, y_2, b) - H(x_2, y_1 - 1, b) + H(x_1 - 1, y_1 - 1, b) \quad (2)$$

where x_1, y_1, x_2, y_2, refers to left, upper, right, lower edges of template, respectively. After the computation of the integral histogram, no further image access is needed.

To obtain template's histogram *h*, one can subtract integral histogram value of pixel immediately above upper-right corner of template and value of pixel immediately to the left of lower left corner from the value of lower-right pixel, which is performed in constant time independent from the patch size. Due to its ability to extract histograms very quickly, it is frequently used in time critical real-time algorithms that need frequent access of histograms. But still there is one big drawback that the computation of the initial integral histogram is indeed rather time and memory consuming.

Particle filtering is a general sampling method for performing inference in state-space models where the state of a system evolves in time and information about the state is obtained via noisy measurements, which has become famous in the area of vision-based tracking [12]. The particle filter estimates probability density by evaluating observation likelihood based on a fixed number of particle samples. When the particle filter is used in the histogram environment, observation likelihood of each particle sample should be evaluated in the area base. The integral histogram can be used to search and localize objects of interest within a given image. Chai, *et al.* [2] combined the integral histogram with the existing particle filter to apply the particle filter to real-time applications.

Depth estimation from a stereo image pair has been one of the most fundamental tasks in the field of computer vision. It aims at estimating a pair of corresponding points between two consecutive images taken from different viewpoints. Stereo matching can be classified into global and local categories according to the strategies used for estimation. Global approaches generally define an energy model with various constraints and solve it using global optimization techniques. Local approaches obtain a disparity map by measuring correlation of color patterns in local neighboring windows. It has been generally known that the local approaches are much faster and more suitable for a practical implementation than global approaches. However, the complexity of the leading local approaches which provide high-quality disparity maps is still huge [9].

Zhang *et al.* [13] presents a local stereo matching approach using the joint integral histograms, and they achieve a speedup factor of about two orders of magnitude.

3. Block-based Sparse Integral Histogram with Interpolation

Block-based sparse integral histogram (BSIH) is a method that focuses on reducing data for initialization by grouping pixels with similar values into blocks. We use the quad-tree method [7] to segment image into blocks, a method which not only is convenient to manage, but which also reduces amount of nodes created.

3.1. Block-based Sparse Integral Histogram

Figure 1. Pixel Grouping by Quad-tree, an Example Image, and its Partitioned Blocks

To create the SBIH image, a two stage process is necessary. First, an input image is segmented using quad-tree method. Then, the integral histogram can be computed by determining missing data for some pixels using the method outlined later in this section. Detail method for the segmentation of an image using quad-tree is found in Oliver [8]. Figure 1 shows the example of blocks by quad-tree and a real image with partitioned blocks.

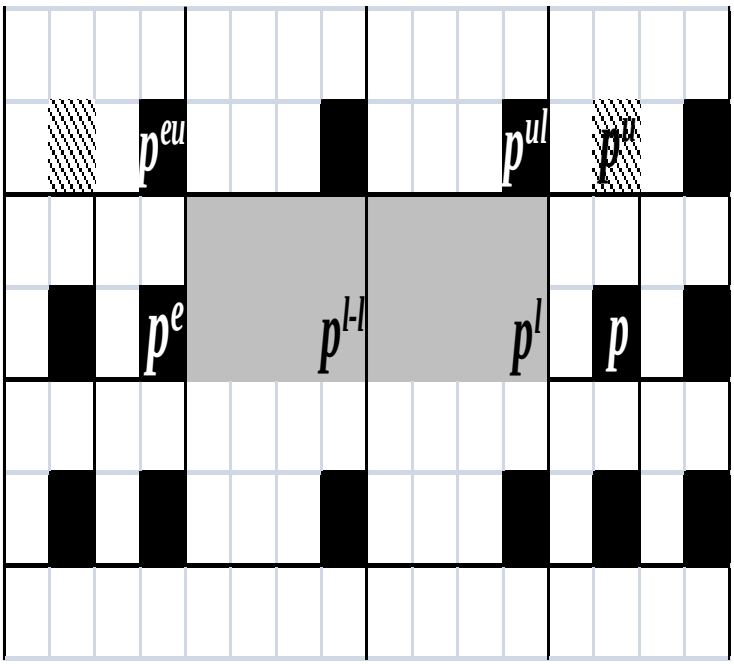

Figure 2. Reference Points p, pl, pl-l ,pe ,pul ,and pu in Eqs. (4) and (5)

The value of block-based integral histogram is:

$$H'(p,b) = H'(x_p, y_p - \mathrm{h_p}, b) + H'(x_p - w_p,\ y_p, b) - H'(x_p - w_p,\ y_p - \mathrm{h_p}, b) + Q\big(f(x_1, y_1)\big) \cdot w_p \cdot \mathrm{h_p} \quad (3)$$

where x_p, y_p refers to x and y coordinates of location p, w_p, h_p refers to width and height of the block that contains location p. The only difference is that original integral histogram uses value of pixel immediately above the pixel while block-based IH uses the pixel immediately above the block. Data of each block is stored at lower-right corner designated as reference point of the block.

In cases where no reference points above and left of the block exist, exact value of these points can be determined by following:

$$e = \{p^l, p^{l-l}, \ldots, p^{e-1}\}$$
$$H'(p,b) = h_p \cdot \textstyle\sum_{m \in e}\{Q\big(f(m)\big) \cdot w_m\} + H(p^e, b) - H(p^{eu}, b) + H(p^{u,l}, b). \quad (4)$$

Equation (4) is used when no reference point exists immediately left of the block. p^l refers to a point located block width away to the left of p, p^{l-l} refers to a point located block width away to the left of p^l. p^e refers to first other reference point located left of p when similar method as with p^l is continuously applied. p^{e-1} refers to the reference point immediately left of p^e. Similarly, p^u refers to a point located block height above p, and p^{ul} refers to upper-left location. Exact histogram can be obtained by computing grey area in Figure 2.

The node of the above is determined by following:

$$e' = \{p^u, p^{u-u}, \dots, p^{e'-1}\}$$
$$H'(p,b) = w_p \cdot \sum_{m \in e'} \{Q(f(m)) \cdot h_m\} + H(p^{e'}, b) - H\left(p^{e'^u}, b\right) + H(p^{u,l}, b). \quad (5)$$

This is computed similarly to (4). p^u refers to a point located block height above reference point p, and $p^{e'}$ refers to first other reference point located above p, as with p^e.

3.2. Histogram Interpolation

Unlike original integral histogram methods, interpolation must be used in the block-based IH since histogram value is not available for all locations. Interpolation methods include simple normalization method that uses the location of q and the reference point of q's block. First order interpolation method or second order interpolation method is used for pixels inside a block.

Basic histogram can be calculated as following:

$$h(x_1, y_1, x_2, y_2, b) = H'(x_2, y_2, b) - H'(x_2, y_1 - 1, b) - H'^{(x_1-1, y_2, b)} + H'(x_1 - 1, y_1 - 1, b). \quad (6)$$

Integral Histogram for a rectangle patch can be computed with estimated histogram H' in Eq. (6). As mentioned before, if there is no reference point at the corresponding location, the value must be computed using Eq. (4) or Eq. (5). The value can be determined easily without any interpolation if normalization method introduced in Muller [3] is employed:

$$H'(x_1, y_1, b) = \frac{x_1 \cdot y_1}{x_p \cdot y_p} \cdot H'(p, b). \quad (7)$$

This method normalizes the histogram value at reference point p using the area from origin to x_1, y_1. Because of its simplicity, this method provides high speed but lower accuracy. Generally, the level of error increases as the distance between the location in question and the reference point increases. To reduce this kind of errors, we propose the first order interpolation method:

$$H'^{(x_1, y_1, b)} = \left(H'^{(p,b)} - H'^{(p^{ul}, b)}\right) \frac{x_1 \cdot y_1 - x_{p^{ul}} \cdot y_{p^{ul}}}{x_p \cdot y_p - x_{p^{ul}} \cdot y_{p^{ul}}} + H'(p^{ul}, b). \quad (8)$$

This method interpolates only at locations between p^{ul}, and p where error exists, and has benefit of being able to use exact value of p^{ul}. The method is commonly used because of its computational feasibility and high accuracy. However, since x-axis and y-axis is interpolated at the same time, the errors are not completely eliminated. The errors can further be reduced by using the second order interpolation method:

$$H'(x_1, y_1, b) = \left(H'(p^u, b) - H'(p^{ul}, b)\right) \cdot \frac{x_{p^u} - x_1}{x_{p^u} - x_{p^{ul}}} + \left(H'(p^l, b) - H'(p^{ul}, b)\right) \cdot \frac{y_{p^l} - y_1}{y_{p^l} - y_{p^{ul}}} + Q(f(x_1, y_1) \cdot (x_p - x_1) \cdot (y_p - y_1) + H'(p^{ul}, b). \quad (9)$$

Although second order interpolation method can produce very accurate results, due to its computational intensive nature, it is advised to be used only at the edges of very large block. Generally, this algorithm supplements the areas with large errors. Additionally, a flawless result can be computed using a method similar to the tree-structure outlined in

[3], but such method are generally unsuitable due to overwhelming amount of calculation needed.

It is believed that because interpolated area for integral histogram value *H'* increases as we approach the bottom-right part of an image, the level of interpolation errors must also be escalated. However, since all four points used in histogram calculation have similar levels of interpolation errors already, they actually offset each other and ultimately maintain the overall accuracy of the histogram.

4. Experiment Result and Discussion

By grouping image into blocks, we were able to drastically improve the initialization time. The improvement is more explicit as image size increases and image complexity decreases.

Table 1. Initialization Times for Various Image Sizes (ms), Minimum block size is 2x2

	256 x 256	512 x 512	1024 x 1024
Original	28.5	112.7	452.6
Muller [3]	4.0	12.4	45.7
Proposed	4.7	13.5	49.9

As shown in Table 1, the proposed method improves the initialization time by factor of 6 to 9 when compared with original algorithm. Considering that the initialization time takes up about one third of total computation time when full search is performed, this boosts the overall computing time significantly. The proposed algorithm is slightly slower than Muller's algorithm, because the quad-tree method generates nodes dynamically during the initialization. Here, the minimum block size is limited to 2x2; initialization time is twice faster than cases where minimum block size limit is 1x1 without losing much accuracy. If minimum block size is 4x4 block, then only one eighth of blocks will be created compared to the case where minimum block size is 2x2 block.

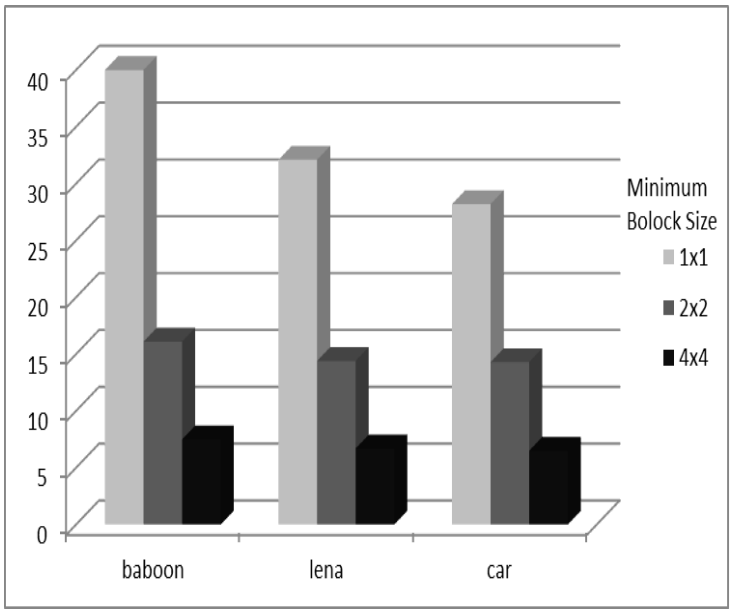

Figure 2. Initialization Time According to the Sizes of Minimum Blocks in 512x512 Images

The processing time of BSIH initialization depends on the minimum block size that is defined according to the complexity of a given image. If an image has many plain regions, then the size of minimum block can be increased without deteriorating the interpolation error. Figure 3 shows the initialization times according to the size of minimum block for sample images. Baboon image is complex and Car image is relatively simple. Since a complex image needs more blocks to be divided, thus the initialization time of a complex image is longer than that of a simple image, and the difference of initialization times between 1x1 and 4x4 minimum blocks are smaller in the simple image.

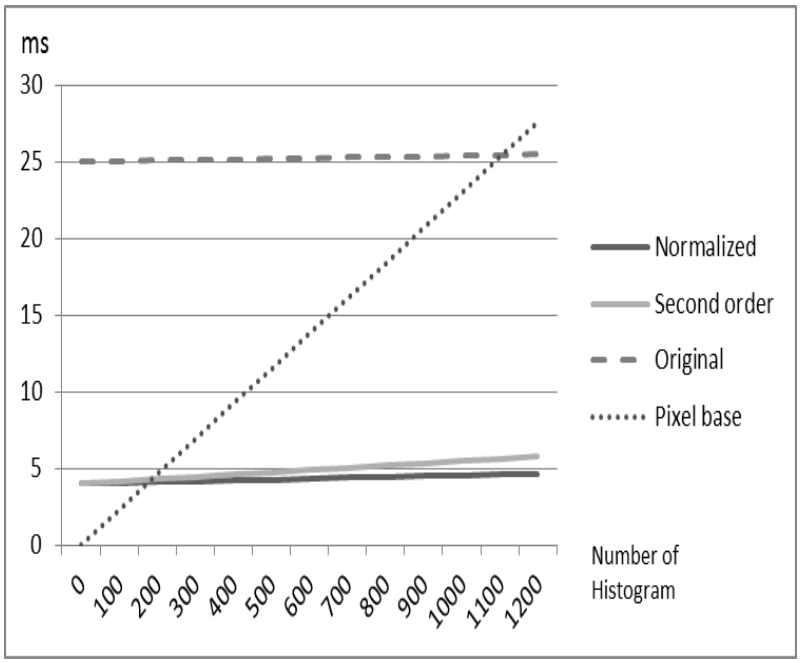

Figure 3. Computation Times with the Number of Histograms

Figure 4 presents the computation time for varying number of histograms generated from a 256x256 sized image. Since the BSIH has small initialization time compared to that of original algorithm, it is more efficient even when only small numbers of histograms are calculated. As shown in Figure 4 the BSIH is always efficient than the original IH and better than pixel-based histogram methods if needed histograms are more than 150. So it is feasible to be used in algorithms requiring only limited number of histograms like object tracking using the particle filter or Kalman filter. Also, since it is faster than original IH even in the case of global matching, this technique can be utilized in the method like stereo matching.

Table 2. Processing Time to Calculate a Histogram for One Location According to Interpolation Methods (ms)

No interpolation, Eq. (6)	Thomas Muller, Eq. (7)	First order interpolation, Eq. (8)	Second order interpolation, Eq. (9)
0.00055	0.00070	0.00102	0.00153

Table 2 shows the elapsed time to calculate a histogram for a location according to interpolation methods. Histogram calculation using the first order interpolation in Eq. (8) and the second order interpolation in Eq. (9) take 0.00102ms and 0.00153ms, respectively, which are two or three times more than that of original method without depending on the size of a rectangle patch. Since the initialization time of the BSIH is much faster than

original IH, the calculation time of each location is negligible if amount of locations to be calculated is not huge.

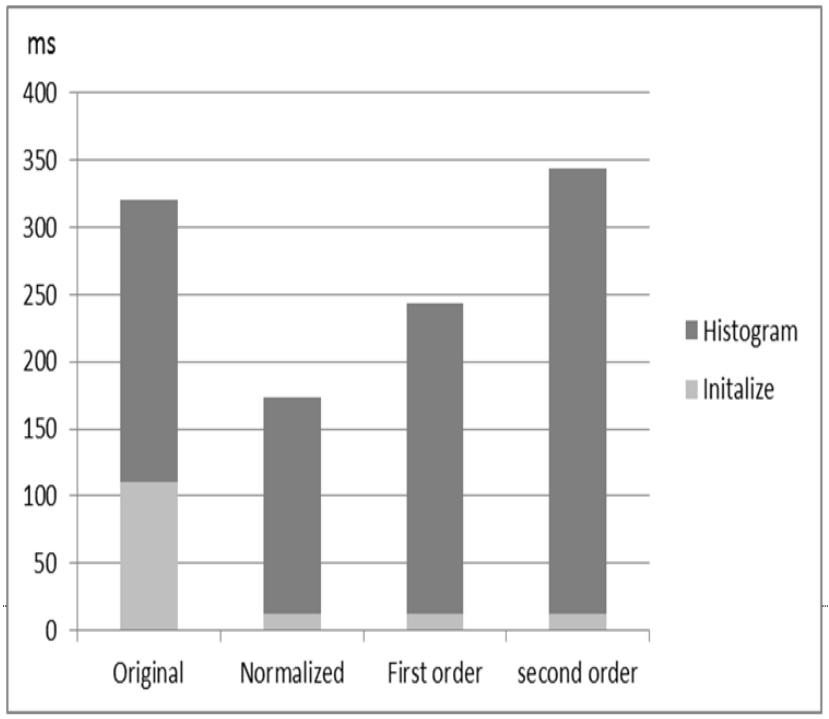

Figure 4. Computation Time to Retrieve Histogram for Entire Region of Image

The method proposed in this paper has faster initialization time compared to original method, but increases actual computation time for generating histograms. Fig. 5 shows the computation times to retrieve all histograms for a 512x512 sized image. It shows that overall computing time with the first order interpolation method is shorter than the original method, although it takes longer to generate histograms. We can achieve both speed and accuracy at the same time by using the second order interpolation adaptively for larger blocks and using the first order interpolation to rest of the image.

It is impossible to make error-free interpolation in BSIH since it needs to approximate histogram to some extent. Especially, the sections divided into larger blocks are more error-prone because more values have to be interpolated. Level of errors in Muller's algorithm is higher due to inaccurate initialization and the lack of interpolation method. Our proposed method is able to retrieve more reliable histogram information because of more accurate initialization and through the use of interpolation methods.

Table 3. Interpolation Errors for Images with Minimum Block Size of 2x2 (Average Histogram Difference per Pixel)

	Baboon	Lena	Airplane	Car
Original	0	0	0	0
Muller's	0.0142	0.0837	0.0989	0.0796
Normalized	0.0079	0.0340	0.0283	0.0348
First order	0.0028	0.0142	0.0147	0.0188
Second order	0.0007	0.0026	0.0027	0.0027

Histogram errors are measured by averaging differences for all locations in an image. The proposed BSIH with the interpolations increases accuracy up to 35 times compared to Muller's method. In Table 3, Baboon has highest complexity while Airplane has lowest complexity. Level of errors is lower in the image with higher complexity because BSIH uses little amount of large blocks in such images.

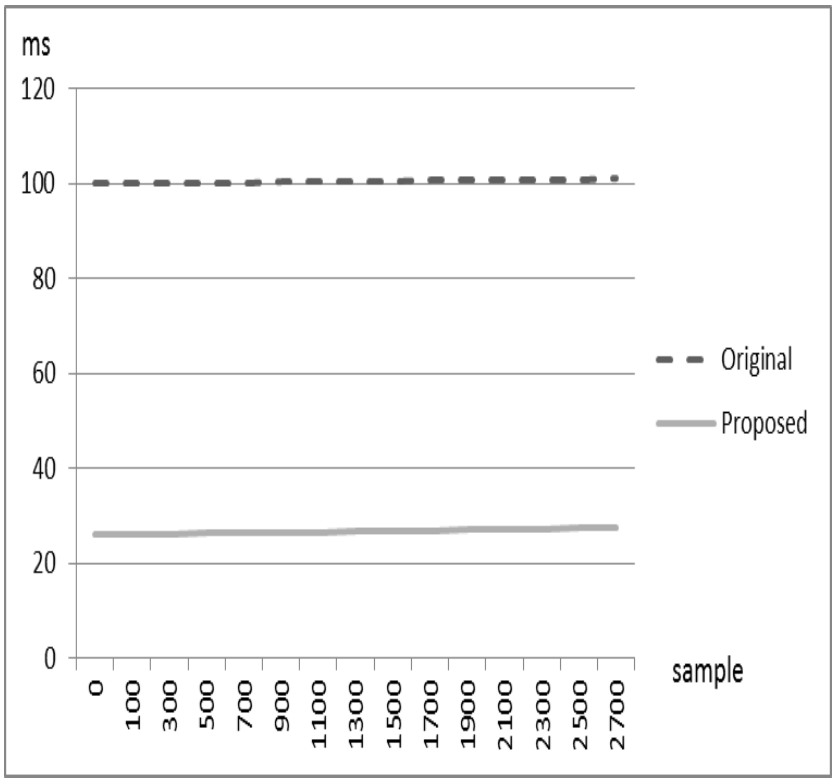

Figure 5. Computation Times when Applied to Particle Filter

Proposed BSIH provides better performance in the method using fewer histograms such as the particle filter because of reduced initialization time. As can be seen in Figure 6, the proposed BSIH behaves twice as fast compared to original method. It is able to perform well even with large amount of samples in the particle filter.

5. Conclusion

Even the Integral Histogram requires low computation time and is flexible enough to adapt many applications, because the method needs relatively expensive initialization, its application is restricted to the areas of quite numbers of histograms are extracted. In this paper we propose a method of block-based sparse integral histogram to enhance the performance of the initialization. To reduce the artifact by blocking we use the quad-tree data structure for an image, and propose the first order and the second order interpolation methods to estimate the Integral Histogram of a pixel using the nearest nodes. The main advantages of our approach are that it is not strongly dependent on the histogram quantization, it is fast to compute initial IH and flexible to accuracy by interpolations.

Experimental results show that the initialization of BSIH is fast up to ten times against original Integral Histogram and its interpolation is accurate enough to apply real-time applications like the particle filter or the stereo matching. As future works of our research, we are developing an analysis algorithm which extracts the

complexity of an image using the texture and the entropy to decide the size of minimum block.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2013-009166) and by Chung-Ang University's Cross Functional Team (CFT) Program under Brain Korea 21 PLUS Project in 2014.

References

[1] F. Porikli, "Integral Histogram: A fast way to extract Histograms in Cartesian spaces", IEEE Conf. on Computer Vision and Pattern recognition, vol. 1, (**2005**), pp.829-836.
[2] Y. Chai, S. Shin, K. Chang and T. Kim, "Real-time User Interface using Particle Filter with Integral Histogram", IEEE Transactions on Consumer Electronics, vol. 56, no. 2, (**2010**), pp. 510-515.
[3] T. Muller, C. Lenz, S. Barner and A. Knoll, "Accelerating Integral Histogram Using an Adaptive Approach", LNCS, (**2008**), pp. 209-217.
[4] A. Y. Jammoussi, "Joint Integral Histogram based Adaboost for Face Detection System", International Journal of computer Applications, vol. 23, no. 5, (**2011**).
[5] K. Zhang, G. Lafruit, R. Lauwereins and L. Van Gool, "Joint Integral Histograms and Its Application in Stereo Matching", IEEE conf. Image Processing, (**2010**), pp. 817-820.
[6] P.-H. Hsu, Y.-C. Tseng and T.-S. Chang, "Low Memory Cost Bilateral Filtering Using Stripe-based Sliding Integral Histogram", IEEE International Symposium on ISCAS, (**2010**) May 30-June 2, pp. 3120-3123.
[7] R. John and S.-F. Chang, "Quad-Tree Segmentation for Texture-based Image Query", ACM 2nd International Conference on Multimedia, (**1994**), pp. 279-286.
[8] M. A. Oliver and N. E. Wiseman, "Operations on Quad tree Encoded Images", The Computer Journal, vol. 26, no.1, (**1983**).
[9] A. Rao, R. K. Srihari, L. Zhu and A. Zhang, "A Method for Measuring the Complexity of Image Databases", IEEE Transactions on multimedia, vol. 4, no. 2, (**2002**), pp. 160-173.
[10] Y. Yi and Y. Lin, "Human action recognition with salient trajectories", Signal Processing, no. 93, (**2013**), pp. 2932-2941.
[11] P. Negri, N. Goussies and P. Lotito, "Detecting pedestrians on a Movement Feature Space", Pattern Recognition, vol. 47, no. 1, (**2014**), pp. 56-71.
[12] M. Isard and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density," European Conference Computer Vision, (**1996**), pp. 343–356.
[13] K. Zhang, G. Lafruit, R. Lauwereins and L. V. Gool, "Joint Integral Histograms and its application in stereo matching", IEEE 17th International Conference on Image Processing, (**2010**) September 26-29, Hong Kong, pp. 817-820.
[14] J. Park, J. Park and T.-Y. Kim, "Block-Based Fast Integral Histogram", World Congress on Engineering and Technology, (**2012**), May 27-30, China, pp. 1-4.

Authors

Jin-Yong Park received a BS degree in Computer Science from An-Yang University in 2010 and a MS degree in Advanced Imaging Engineering at Chung-Ang University in 2012.

Hyunki Hong received his BS, MS, and PhD degrees in electronic engineering from Chung-Ang University, Seoul, Korea, in 1993, 1995, and 1998, respectively. From 1998 to 1999 he worked as a researcher in the Automatic Control Research Center, Seoul National University, Korea. From 2000 to 2014, he was a professor in the

Department of Imaging Science and Arts, Graduate School of Advanced Imaging Science, Multimedia and Film at Chung-Ang University. Since 2014, he has been a professor in the School of Integrative Engineering, Chung-Ang University. His research interests include computer vision, augmented reality, and multimedia application.

TaeYong Kim (Corresponding Author) received a BS degree in Electrical Engineering and MS degree in Communication Engineering from Han-Yang University, Seoul, Korea, in 1986 and 1988, respectively, and a PhD in Computer Science and Engineering from Pohang University of Science & Technology, Korea, in 1998. He has been a professor at the Graduate School of Advanced Imaging Science, Multimedia and Film, Chung-Ang University, Seoul, Korea since 2003. His research interests are in Computer Vision, Image Processing, and A.I.