LOW-COMPLEXITY HOG FOR EFFICIENT VIDEO SALIENCY

Teahyung Lee, Myung Hwangbo, Tanfer Alan⋆, Omesh Tickoo, Ravishankar Iyer

Intel Labs, Hillsboro, Oregon, USA
⋆Technische Universitt Darmstadt, Computer Systems Group, Darmstadt, Germany

ABSTRACT

In this paper, we propose a low-complexity histogram of oriented gradients (HOG) implementation for efficient video saliency framework. After showing how original HOG calculations present significant computation bottleneck for visual understanding pipes, we present the optimized HOG flow and algorithm for video saliency framework, which can reduce computational requirements without losing algorithmic performance. Furthermore, simplification for light-weight computations and data-reusable scanning for optimal memory usage are explained for improving system efficiency. Based on our testing and analysis, the proposed HOG implementation optimizes computational complexity and performance while maintaining the video saliency algorithm capability.

Index Terms— low-complexity HOG, video saliency

1. INTRODUCTION

A widespread availability of inexpensive video cameras and rise of social networking platforms has led to creation of huge amount of video data. Effective consumption of such large video data sizes is leading to the need for efficient video summarization applications that can filter out salient parts of a video or group of images [1, 2]. Video summarization is generally composed of three steps as shown in Fig. 1. Firstly feature extraction acquires spatial or temporal cues as saliency. Then similarity measure is performed with attributes or attention models using the extracted features/saliency. Finally key frames are selected for a summary based on scene analysis information and decision policy. One typical example can be found in [2]. To perform video summarization on mobile/wearable environments, its computational complexity is one of bottlenecks for real-time implementation. Our experiments show that feature extraction puts high demands from per-pixel processing as compared to the other steps. Fig. 1 shows the feature extraction is a dominant share of total computation.

HOG (Histogram of Oriented Gradients) is one of most popular and fundamental image features and a basis for other higher-level and more complex feature sets [3, 4, 5]. Many hardware accelerators for HOG have been developed for real-time object detection. Cao proposed a variant of the HOG feature set and efficient integral map for stop-sign detection application [6]. Kadota *et al.* proposed several methods to simplify computation, such as conversion of the division and square root [7]. Most of previous architecture study on HOG has been focused on real-time up to 1080p format case especially for object detection application [8, 9]. In our proposed scheme, algorithm computation is explored for low complexity and efficient system requirement. This is achieved by reducing or removing expensive operations and functional units such as arctangent, division, square-root, and normalization steps in the design flow. Being one of popular feature extraction modules, some variations of HOG also have been studied for dynamic application areas in [10, 11, 12].

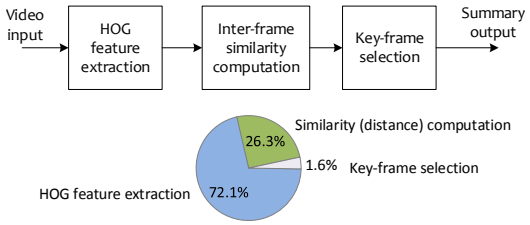

Fig. 1. Video summarization pipeline and their workload analysis. The HOG feature extraction is a main bottleneck.

In this paper we propose a low-complexity HOG for efficient video saliency framework, which shows equivalent saliency results with optimal computational complexity and performance for real-time system.

2. HOG-LX

HOG is an image feature [13] that consists of local 1-D histograms of gradient directions when an image is partitioned into small rectangular regions called cells. Computationally it involves image gradient, orientation histogram per cell, and contrast normalization over blocks. To reduce computations of the original HOG while minimizing performance degradation as a scene descriptor, we propose a low-complexity HOG (named HOG-LX) as an efficient front-end to many computer vision applications. It reorganizes the order of computations and creates operations per histogram channel. The design of HOG-LX scheme in Fig. 2 is focused on hardware acceleration, which can be leveraged for low-power/cost products including wearable and mobile platforms.

2.1. Accelerated histogram binning

Each cell in the HOG computes an orientation histogram from image gradients within a $n \times n$ cell region. Histogram

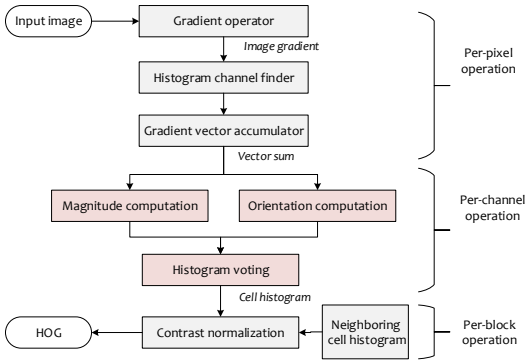

Fig. 2. HOG-LX computation flow that embeds new per-channel operations in the histogram binning.

channels are evenly spread over 0 to 180 degrees while the sign of gradient directions is commonly ignored. Every pixel in the cell contributes a weighted vote for the histogram channels based on its gradient magnitude. This original binning method [13] involves expensive computations to be executed as per-pixel operations. Every pixel requires a squared or square root operation of gradient values for the voting, arctangent operation for the gradient orientation, and bilinear operation for a split vote with neighboring bins. To address this, we propose a new scheme that converts the per pixel binning into operations per histogram channel. For example, in case of the cell size n = 8 and the number of histogram channel m = 9, $\times$64 operations (equal to n^2) can be reduced to $\times$9 operations (equal to m).

Per-channel computation: In Fig. 2 the histogram channel finder and accumulator are newly introduced for per-channel operations. Prior to the histogram voting, the channel finder decides which channel each gradient belongs to and the accumulator computes a sum of all gradient vectors associated with the same channel. Suppose two gradients in a cell lie on the same channel as depicted in Fig. 3. Rather than dealing with the individual gradients $\mathbf{g}_1$ and $\mathbf{g}_2$, we first take a vector sum, $\mathbf{g}_{sum} = \mathbf{g}_1 + \mathbf{g}_2$. The remaining expensive operations are then performed only on a single vector $\mathbf{g}_{sum}$. A vector sum $\mathbf{g}_{\mathcal{B}_i} = \sum_{k \in \mathcal{B}_i} \mathbf{g}_k$ combines all the pixel gradient falling on a channel $\mathcal{B}_{i=1,\ldots,9}$. Given $\mathbf{g}_{\mathcal{B}i}$ the per-channel operation in Fig. 2 computes a squared root and arctan only once for each channel. Also the bilinear interpolation for the vote split does not need to be computed for every pixel gradient.

Histogram channel finding: The per-channel computation needs to know which channel each pixel gradient belongs to. We develop a fast channel finder that avoids arctan operation. It minimizes the number of comparisons using the symmetry of tangent. The channel boundary condition $\theta_i \leq \tan^{-1}(I_y/I_x) \leq \theta_{i+1}$ can be rewritten as $I_x \tan\theta_i \leq I_y \leq I_x \tan\theta_{i+1}$, which involves a series of comparisons between the y-gradient I_y with a product of the x-gradient I_x and tangent values at channel borders. The symmetric property of tangent is employed to make the architecture design more

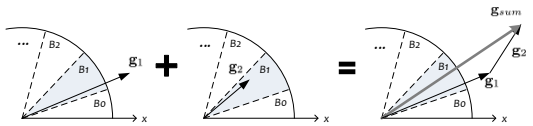

Fig. 3. A vector sum used in HOG-LX. All pixel gradients falling in the same histogram channel are combined into a single vector $\mathbf{g}_{sum} = \mathbf{g}_1 + \mathbf{g}_2$.

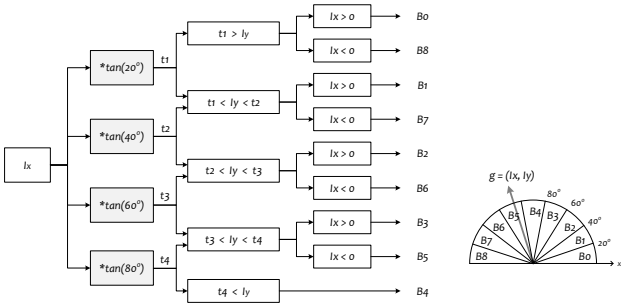

Fig. 4. A histogram channel finder data flow that can be hardware-implemented as a series of comparisons given $\mathbf{g} = (I_x, I_y)$.

simple; $\tan\theta = -\tan(\pi - \theta), \theta = [0, \pi)$. This symmetry can decrease the number of multiplications and memory requirement for boundary tangent values. The channel finding process is followed by accumulating the current gradient on $\mathbf{g}_{\mathcal{B}i}$ at the corresponding channel. Simultaneous operation in the multiplication and accumulation would form a critical pipeline path in the hardware architecture design.

2.2. Simplification for light-weight computations

The original HOG uses many sophisticated operations that permit to capture a wide appearance variance of visual objects. From various experimental case studies we found that the level of sophistication can be adjusted depending on applications, especially for scene classification or scene saliency in which HOG is used as a global scene descriptor [2]. Given a trade-off between HOGs discriminative power and algorithm simplification, we observed that the performance degradation with lowering complexity can be acceptable in the video saliency framework. HOG-LX chooses the following simplifications which can eliminate non-hardware-friendly operations and achieve low-complexity while maintaining its role as a global scene descriptor:

- Skip the bilinear voting in the histogram binning
- No arctangent computation in the histogram binning
- Use L_1 norm of a gradient vector sum in the voting
- Use L_∞ norm in the contrast normalization

Skipping the bilinear operation enables the coarse-level channel finder in Fig. 4 where arctangent is no longer required. The contrast normalization using L_∞ norm only needs to keep maximums in cell histograms. These simplifications can decrease a required chip area and functional units, which also decreases power consumption significantly for HW implementation.

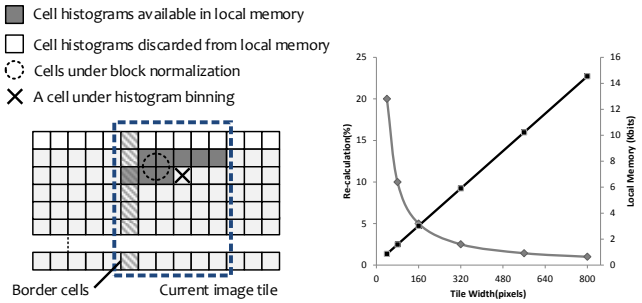

Fig. 5. (Left) tiled horizontal image scanning and (right) recalculation-rate vs. tile size.

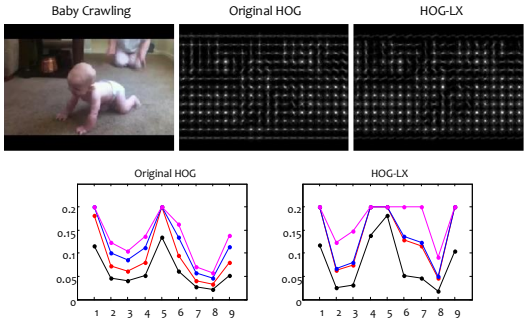

Fig. 6. Comparison between the original HOG and HOG-LX as a scene descriptor.

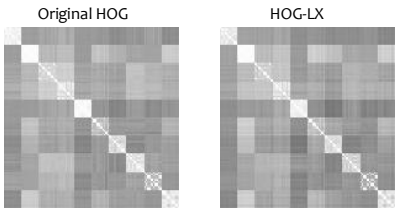

Fig. 7. Inter-frame similarity matrix of the original and HOG-LX between 200 images from 10 actions in UCF-101.

2.3. Data-reusable scanning for efficeint memory usage

The gray region in Fig. 5 indicates a portion of the cell histograms that should be kept in the memory while the cell histogram and block normalization are concurrently running. The required size M to hold this memory region can be expressed as $M = (w/n + 1) \times m \times d$, an image width w, cell width n , the number of channels m, and a bit-resolution d of the cell histogram. To reduce M we divide the input image into vertical tiles with overlapping border cells.

The tiled memory scanning in Fig. 5 keeps histograms of horizontally adjacent cells of a single tile instead of keeping an entire image in the local memory. The tile width can adjust the local memory size as much as needed under memory-constrained conditions. Nonetheless cell histograms at the left and right border of a tile is subject to recalculation when a new tile of the image is fetched. As the tile width becomes bigger the local memory size linearly increases and the recalculation rate exponentially decreases (see Fig. 5). When the tile width is 20 cells (*i.e.*, 160 pixels), a 432B local memory would be sufficient at the cost of 5% re-calculation rate of cell histogram, which enables a low-cost solution for the memory-constrained platform approach.

3. ANALYSIS AND EVALUATIONS

We used Verilog-based tool, ModelSim [14], for simulation to evaluate the efficiency of HOG-LX. Across the different range of tested formats HOG-LX showed better performance. For example, for 1080p with 30 fps video using 20-cell width tile, HOG-LX can achieve 1.17 GOPs and the local memory size for cell histogram is 432B at 0.85 Mbps memory bandwidth. The original HOG (without a sliding window scheme used for detection) requires 15.12 GOPs and 3.07 Gbps memory bandwidth for 1080p with 30 fps. It demonstrates that our low complexity solution can achieve a lighter system requirement under a memory size constraint, which is beneficial for low power/cost system design. Since HOG-LX skips complex functional units of arctan and division module, this architecture is well-suited for low cost and low complexity designs, which can be leveraged for wearable or mobile platform areas.

To compare the video saliency performance between the original HOG and HOG-LX, we used UCF-101 action dataset [15]. From 10 different action videos a total of 1000 images are randomly selected for the input $\mathbf{V}$. Fig. 6 illustrates one example of the HOG-LX extracted from an entire image and compares it with the original HOG [13]. At the top cell histograms are visualized with bars aligned to corresponding orientations. The bottom plots compare cell histograms at a select cell, which normalized by four different blocks. Since the histogram patterns share a similar trend, both the original HOG and HOG-LX are quite close in terms of overall gradient distributions. Fig. 7 compares the similarity w_{ij} between an image i and j computed by the original HOG and HOG-LX, respectively. Though both HOG are different to some degree as shown in Fig. 6, their similarity matrices are very close. It means that HOG-LX has an equivalent scene discrimination power to the original HOG in the video saliency framework where w_{ij} is a key input. This result also demonstrates that HOG-LX is suitable for applications that use image distance computations as a core functionality.

4. CONCLUSION

We proposed a low-complexity HOG as part of an efficient video saliency framework implementation. From the workload performance analysis, which identifies HOG as a major computational bottleneck, its original computational complexity is reduced by using the vector sum and flow reordering. Additional algorithm simplifications and data-reusable memory scanning were employed to improve system efficiency. The proposed HOG-LX showed an equivalent discrimination power as a scene descriptor in the video saliency framework.

5. REFERENCES

[1] A. Kamoji, R. Mankame, A. Masekar, and A. Naik, "Key frame extraction for video summarization using motion activity descriptors," *IJRET*, vol. 62, pp. 291–294, January 2014.

[2] S. Chakraborty, O. Tickoo, and R. Iyer, "Adaptive keyframe selection for video summarization," in *WACV*. IEEE, 2015.

[3] L.-J.Li, H. Su, Y. Lim, and L. Fei-Fei, "Object bank: an object-level image representation for high-level visual recognition," *IJCV*, Sept 2013.

[4] P. Dollr, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE PAMI*, Aug 2014.

[5] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE PAMI*, Sept 2010.

[6] T. P. Cao and G. Deng, "Real-time vision-based stop sign detection system on FPGA," in *PDICTA*, 2008.

[7] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura, "Hardware architecture for HOG feature extraction," in *IIH-MSP*, 2009.

[8] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Architecture study of HOG feature extraction processor for real-time object detection," in *SIPS*, 2012.

[9] K. Takagi, K. Mizuno, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "A sub-100-milliwatt dual-core HOG accelerator VLSI for real-time multiple object detection," in *ICASSP*, 2013.

[10] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod, "CHoG: compression histogram of gradients a low bit rate descriptor," in *CVPR*, 2009.

[11] N. Buch, J. Orwell, and S. Velastin, "3d extended histogram of oriented gradients (3DHOG) for classification of road users in urban scenes," in *BMVC*, 2009.

[12] A. Bosch, A. Zisserman, and X. Muno, "Image classification using POIs and multiple kernel learning," *IJCV*, vol. 62, pp. 291–294, January 2008.

[13] N. Dalal and B. Triggs, "Histogram of oriented gradients for human detection," in *CVPR*. IEEE, 2005.

[14] Mentor Graphics, "ModelSim," `http://www.mentor.com/products/fv/modelsim/`.

[15] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: a dataset of 101 human action classes from videos in the wild," in *CRCV-TR-12-01*, 2012.