Fast Calculation of Histogram of Oriented Gradient Feature by Removing Redundancy in Overlapping Block

SOOJIN KIM AND KYEONGSOON CHO
Department of Electronics Engineering
Hankuk University of Foreign Studies
Gyeonggi-do, 449-791 Korea
E-mail: {ksjsky9888; kscho}@hufs.ac.kr

In order to improve pedestrian detection accuracy, histogram of oriented gradient (HOG) feature is widely used in many applications. Although HOG feature can provide high detection accuracy, fast detection time is hardly achieved due to its computational complexity. Therefore, this paper describes a novel algorithm for fast calculation of HOG feature. In the proposed algorithm, HOG feature is calculated based on cells instead of overlapping blocks to avoid redundancy. Furthermore, by identifying key rules and sharing common operations in trilinear interpolation, the number of required operations in HOG feature calculation is reduced up to 60.5% while detection accuracy is not degraded at all. Therefore, the proposed method is applicable to many applications such as intelligent vehicles, robots, and surveillance systems in which both high detection rate and fast detection time are strongly required.

Keywords: pedestrian detection, histogram of oriented gradient, trilinear interpolation, high detection rate, fast detection time

1. INTRODUCTION

A robust and discriminative feature is strongly required for pedestrian detection because of a wide range of pedestrian’s appearance, illumination, and cluttered background. Since histogram of oriented gradient (HOG) [1] feature has achieved great success on pedestrian detection, it is widely used in many applications such as intelligent vehicles, robots, and surveillance systems. In these applications, detection speed is essential as well as detection accuracy [2]. However, its detection speed is relatively slow due to the high computational complexity of HOG feature calculation. Especially trilinear interpolation, one of the most effective techniques to improve detection accuracy in HOG feature calculation, degrades detection speed significantly. In order to accelerate detection speed, therefore, we present a novel method to significantly reduce the number of required operations in HOG feature calculation. By replacing block-based operation with cell-based one and by identifying key rules in trilinear interpolation, each cell in overlapping blocks is considered only once and redundant operations are totally removed. Furthermore, the number of required operations is reduced up to 60.5% by sharing the common operations in trilinear interpolation.

The rest of this paper is organized as follows. Section 2 introduces the related works to pedestrian detection using HOG feature. Section 3 briefly overviews the algorithm of
HOG feature calculation, and Section 4 describes the proposed algorithm to accelerate the speed of HOG feature calculation. The experimental results are given in Section 5. Finally, Section 6 concludes this paper.

2. RELATED WORK

In recent years, many researches have been conducted for pedestrian detection. Symmetry, corners, texture, shadow, color, edges, and gradients can be used to represent the characteristics of the pedestrian. Among the various types of feature, HOG feature that is based on gradient information is widely used since it is considered to be the most discriminative feature for pedestrian detection. The performances of 16 state-of-the-art pedestrian detectors across six data sets are evaluated in [2]. They reported that nearly all modern detectors employ some form of gradient histograms. In [3], a diverse set of pedestrian detectors is evaluated with identical test criteria and data sets. According to their experiments, an approach using HOG feature clearly outperforms others. In order to improve detection accuracy, several features are often combined with HOG feature. A combination of several features with HOG feature outperforms any individual feature in [4]. By combining HOG feature with local binary pattern (LBP) [5], the detection rate is increased by 7% in [6]. However, the increase in detection accuracy has been paid for with increased computational costs [7]. In many applications, including automotive safety, surveillance, and robotics, fast detection time is as important as high detection rate.

Many researches have been proposed to improve detection speed for pedestrian detection. Several methods to simplify the computation of HOG feature extraction are proposed in [8-10]. In order to reduce the amount of calculations, the interpolation technique is discarded in [8], a simplified linear interpolation technique is applied in [9], and an approximated trilinear interpolation is applied in [10]. Consequently, detection rates are significantly degraded. Even though trilinear interpolation is one of the most effective techniques to improve detection accuracy, it is not easy to apply trilinear interpolation technique in its original form to provide real-time detection due to its high computational complexity. Nonetheless, trilinear interpolation technique cannot be discarded, simplified or approximated in HOG feature calculation since high detection rate is also important in many applications. Instead the method to accelerate the computation of trilinear interpolation is strongly required.

3. ALGORITHM OF HOG FEATURE CALCULATION

3.1 Brief Review on Overall Algorithm

HOG feature is presented by N. Dalal and B. Triggs to improve accuracy of human detection. It is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid [1]. Given pre-defined parameters in Table 1, HOG feature is calculated as follows. As shown in Fig. 1, the first step of HOG feature calculation is to calculate image gradient for each pixel in $S_i \times S_j$ detection window. Gradient for $x$-direction is calculated by Eq. (1) and $y$-direction is calculated by Eq. (2). In these equations,
\( f(x, y) \) represents a pixel value for \((x, y)\) position in an image \(I\). Then the gradients are used to calculate weighted votes for gradient magnitude \(M\) and orientation \(\theta\) by Eqs. (3) and (4). The third step is to accumulate weighted votes for gradient magnitude into \(N\) orientation bins over \(p \times p\) spatial cells. When inter-bin distance (\(\theta_{\text{dist}}\)) is 20° over 0°–180°, \(N\) is determined as 9. In order to avoid aliasing, block-based interpolation is applied to interpolate weighted votes bilinearly between the neighboring bins in both orientation and position. The next step is to normalize contrast within each block, and two representative normalization schemes are presented in Eqs. (5) and (6). In these equations, \(B_k\) represents 36-dimensional vector for a block, \(c\) represents each element in the vector, and \(\varepsilon\) is a small constant used to avoid division by zero. The dimension of each block is determined by the number of orientation bins in the block. Finally, the last step is to collect histogram vectors for all overlapping blocks over detection window. When \(S_X = 48, S_Y = 96, p = 6, c = 2, L = 6,\) and \(N = 9\), as shown in Fig. 1, the dimension of the final HOG feature for each detection window is 3,780 since the total number of 36-dimensional overlapping blocks in a window is 105.

**Table 1. Parameters for HOG feature calculation.**

<table>
<thead>
<tr>
<th>window size (pixels)</th>
<th>(S_X \times S_Y)</th>
<th>cell size (pixels)</th>
<th>(p \times p)</th>
</tr>
</thead>
<tbody>
<tr>
<td># of cells in block</td>
<td>(c \times c)</td>
<td>block size (pixels)(=pc \times pc)</td>
<td></td>
</tr>
<tr>
<td>block stride (pixels)</td>
<td>((L_x, L_y))</td>
<td># of bins per cell</td>
<td>(N)</td>
</tr>
<tr>
<td># of cells in window</td>
<td>(C_x \times C_y)</td>
<td>((C_x = S_X/p, C_y = S_Y/p))</td>
<td></td>
</tr>
<tr>
<td># of blocks in window</td>
<td>(B_x \times B_y)</td>
<td>((B_x = (S_X - pc + L_x)/L, B_y = (S_Y - pc + L_y)/L))</td>
<td></td>
</tr>
</tbody>
</table>

**Step 1) calculate image gradient for each pixel**

![Image gradient calculation](image)

\[
\begin{align*}
\text{for } x\text{-direction: } & gx = \partial_x f(x,y) = f(x+1,y) - f(x-1,y) \\
\text{for } y\text{-direction: } & gy = \partial_y f(x,y) = f(x,y+1) - f(x,y-1)
\end{align*}
\]

**Step 2) calculate gradient magnitude and orientation**

gradient magnitude: \(M(x,y) = (gx^2 + gy^2)^{1/2}\) \(\text{(3)}\)

gradient orientation: \(\theta(x,y) = \tan^{-1}(gy/gx)\) \(\text{(4)}\)

**Step 3) accumulate weighted votes into orientation bins over spatial cells**

![Gradinet Caculation](image)

**Step 4) normalize contrasts within overlapping blocks of cells**

![Normalization](image)

block: 2x2 cells
orientations: 9 for each cell
\(L_1\)-norm: \(|c|/(\|B_k\|+\varepsilon)\) \(\text{(5)}\)

\(L_2\)-norm: \(|c|/(\|B_k\|^2+\varepsilon^2)^{1/2}\) \(\text{(6)}\)

**Step 5) collect histograms for all overlapping blocks over detection window**

![Histogram](image)

detection window: 48x96 pixels
block stride: \((6, 6)\)

# of blocks in detection window: \(7 \times 15 = 105\)

\(\rightarrow\) dimension of the final HOG feature: 36x105 = 3,780

Fig. 1. HOG feature calculation.
3.2 Interpolation Technique in HOG Feature Calculation

As mentioned in Section 3.1, an interpolation technique is applied at the third step of HOG feature calculation in order to avoid aliasing and to improve detection accuracy. Fig. 2 shows a simple example of linear and trilinear interpolations. When \( N = 9 \) over \( 0-180^\circ \) and a gradient orientation for a pixel in \( Cell_0 \) is \( 32^\circ \), the two nearest neighboring bins are determined as 1 and 2 as shown in the figure. Since only orientation of a pixel is considered in linear interpolation, the weighted votes are calculated by multiplying magnitude \( (M) \), Gaussian coefficient \( (G) \), and weight for orientation \( (W_\theta) \). Then, the weighted votes are distributed into only two bins (bin 1 and bin 2 for \( Cell_0 \) only). On the other hand, orientation and the position of a pixel in block are all considered in trilinear interpolation. Therefore, the weighted votes are calculated by multiplying magnitude \( (M) \), Gaussian coefficient \( (G) \), weight for orientation \( (W_\theta) \), and weights for pixel position \( (W_x \) and \( W_y \)), and distributed into eight surrounding bins (bin 1 and bin 2 for \( Cell_0-Cell_3 \)) as shown in Fig. 2.

![Fig. 2. Example of linear and trilinear interpolations.](image)

Even though trilinear interpolation improves detection rate considerably, fast detection time is hardly achieved due to its high computational complexity. Since HOG feature is calculated by considering all overlapping blocks in detection window, the amount of computations is further increased. Table 2 shows the number of required multiplications per detection window including interpolation and \( L^2 \)-norm normalization. As shown in the table, the number of required multiplications for trilinear interpolation is much larger than linear interpolation. In order to accelerate detection speed, therefore, most researches usually discard trilinear interpolation and often apply linear interpolation instead.

<table>
<thead>
<tr>
<th>Interpolation</th>
<th># of multiplications per detection window</th>
</tr>
</thead>
<tbody>
<tr>
<td>linear</td>
<td>( {4 \times p^2 \times c^2} + (c^2 \times N) + 1) ( \times B_x \times B_y )</td>
</tr>
<tr>
<td>trilinear</td>
<td>( {32 \times p^2 \times (c-1)^2} + (32 \times p^2 \times (c-1) + (8 \times p^2 \times (c^2 \times N) + 1) ( \times B_x \times B_y )</td>
</tr>
</tbody>
</table>

Table 2. Number of required operations in interpolations.
4. PROPOSED ALGORITHM

Trilinear interpolation is applied in HOG feature calculation to avoid aliasing, but it increases computational complexity significantly due to the overlapping-block-based operation. However, there have been no attempts to remove the redundant operations in trilinear interpolation. Moreover, trilinear interpolation technique is not applied in many researches to achieve high detection speed, resulting in degraded detection accuracy. In this paper, therefore, we address the problem of HOG feature calculation and propose a novel algorithm to totally remove the redundant operations to accelerate detection speed without any loss of detection accuracy.

As described in the previous section, redundant operations are inherently involved in the original algorithm due to the overlapping-block-based operation. When $S_x=48$, $S_y=96$, $c=2$, and $L=p=6$, for example, 40 cells are overlapped twice and 84 cells are overlapped four times as shown in Fig. 3. Although the total number of cells in detection window is 128, the number of required cells for HOG feature calculation is 420 due to the overlapping blocks. In this paper, therefore, we propose a novel algorithm of HOG feature calculation to totally remove the redundant operations. Since the overlapping operations for each cell are considered in advance, each cell is required only once in the proposed algorithm. From now on, we will describe the proposed algorithm in case that $c=2$ and $L=p=6$.

Fig. 3. Number of required cells in HOG feature calculation.

Fig. 4. Pre-defined regions in block.
Trilinear interpolation requires the largest computational efforts in HOG feature calculation. Since the amount of operations in trilinear interpolation depends on the pixel position in a block, we define region $x_R$ and $y_R$ in a block and divide the block into four regions ($SB_1$ to $SB_4$) as shown in Fig. 4. An example of trilinear interpolation for the pixel position is shown in Fig. 5. When a pixel is in $x_R \cap y_R$ as indicated (a) in Fig. 5, weighted votes are distributed into eight surrounding bins (two nearest bins to $\theta$ for each of four cells). In case that a pixel is in $x_R \cup y_R - x_R \cap y_R$ as indicated (b), weighted votes are distributed into four bins (two nearest bins to $\theta$ for each two nearest cells that the pixel is positioned). Otherwise, weighted votes are distributed into only two bins (two nearest bins to $\theta$ for the cell that the pixel is positioned). The last case is indicated as (c) in Fig. 5.

Fig. 5. Trilinear interpolation for pixel position.

Since interval of each overlapping block is one cell in each direction, most cells in detection window have up to four types, depending on the position in overlapping block. Therefore, we divide a cell into four regions ($SC_1$ to $SC_4$) and define four cell types (type #1 to type #4) as shown in Fig. 6. For example, since cell_a in Fig. 6 is overlapped by four blocks (block_a to block_d), cell_a is type #4 for block_a, type #3 for block_b, type #2 for block_c, and type #1 for block_d. As shown in the figure, only Gaussian coefficient among the operands in trilinear interpolation differs for each type ($G_1$ to $G_4$ represent Gaussian coefficients in region $SB_1$ to $SB_4$). Therefore, we modified the equation of trilinear interpolation to share the common operations for each cell type as shown in Table 3. In this table, $A$ to $D$ represent weights for pixel position ($Wx \times Wy$). Table 4 shows the
equations for $A-D$. Note that $G1$–$G4$ and $A-D$ are pre-computed in the proposed algorithm since they are already determined by $p$ and $c$. The modified equations of trilinear interpolation are applied in Fig. 7 where the proposed algorithm of HOG feature calculation is presented.

Table 3. Proposed trilinear interpolation for each cell type.

<table>
<thead>
<tr>
<th>cell type</th>
<th>trilinear interpolation</th>
<th>cell type</th>
<th>trilinear interpolation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$SC_1$</td>
<td>$[Nk+a] \leftarrow M \times A \times G_1 \times \beta$</td>
<td>$SC_1$</td>
<td>$[N(k-4)+a] \leftarrow M \times C \times G_2 \times \beta$</td>
</tr>
<tr>
<td>$SC_2$</td>
<td>$[Nk+a] \leftarrow M \times A \times G_1 \times \beta$</td>
<td>$SC_2$</td>
<td>$[N(k-3)+a] \leftarrow M \times A \times G_2 \times \beta$</td>
</tr>
<tr>
<td>$SC_3$</td>
<td>$[Nk+a] \leftarrow M \times A \times G_1 \times \beta$</td>
<td>$SC_3$</td>
<td>$[N(k-3)+a] \leftarrow M \times A \times G_2 \times \beta$</td>
</tr>
<tr>
<td>$SC_4$</td>
<td>$[Nk+a] \leftarrow M \times A \times G_1 \times \beta$</td>
<td>$SC_4$</td>
<td>$[N(k-3)+a] \leftarrow M \times A \times G_2 \times \beta$</td>
</tr>
</tbody>
</table>

Table 4. Pre-computed weights for pixel position.

<table>
<thead>
<tr>
<th>$(x, y)$ position for a pixel in cell</th>
<th>weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x'= (x+0.5)/p-0.5$</td>
<td>$A \leftarrow X_{x'} \times Y_{y'}$</td>
</tr>
<tr>
<td>$y'= (y+0.5)/p-0.5$</td>
<td>$B \leftarrow X_{y'} \times (1-Y_{y'})$</td>
</tr>
<tr>
<td>$x''=\text{floor}(x')$</td>
<td>$y''=\text{floor}(y')$</td>
</tr>
<tr>
<td>$y_1 = y'-y''$, $y_2 = 1-y_1$</td>
<td>$C \leftarrow (1-X_{x}) \times Y_{y}$</td>
</tr>
<tr>
<td>$X_{x} = \text{max}(x_1, x_2)$</td>
<td>$Y_{y} = \text{max}(y_1, y_2)$</td>
</tr>
<tr>
<td>$D = -(1-X_{x}) \times (1-Y_{y})$</td>
<td></td>
</tr>
</tbody>
</table>

In order to totally remove redundant operations, HOG feature is calculated by cell-based operation in the proposed algorithm. Fig. 7 shows the proposed algorithm for fast HOG feature calculation. As shown in Fig. 7, each cell is considered only once by identifying key rules and sharing common operations in trilinear interpolation. Unlike the original algorithm, block normalization is conducted after the interpolation for all cells in detection window is finished in our algorithm. In Fig. 7, $i$ represents an index of cell row in detection window, $j$ represents an index of cell number, $k$ represents an index of group of $N$ bins for cell, $n_1$ and $n_2$ represent the two nearest bins to $\theta$, and $n_i$ and $n_j$ represent the corresponding weights. In order to further reduce the redundant operations in the proposed trilinear interpolation in Table 3, we shared common operations and found the best order as shown in Fig. 7. By multiplying the operands in trilinear interpolation with the specific order, the number of required multiplications in trilinear interpo-
ulation is reduced up to 60.5% (from 272,160 to 107,496). When the proposed algorithm is applied with \( c=3 \), the number of required operations is further reduced up to 68% (from 604,800 to 193,896).

![Image of proposed algorithm for fast HOG feature calculation](image)

Fig. 8 shows the examples of HOG feature calculation using the proposed algorithm. When the current pixel is at \((0, 0)\) in the 0th cell, only type #1 is considered since the 0th cell belongs to only one block as shown in the figure. Since the pixel is in SC1 of type #1, the operation of trilinear interpolation is required for two bins as shown in Eq. (7). When the current pixel is at \((4, 0)\) in the 79th cell, type #2 and type #4 are considered due to the two overlapping blocks as shown in Fig. 8. In this case, the operation of trilinear interpolation is required for six bins since the pixel is in SC2 of type #2 and type #4. As shown in Eqs. (8) and (9), a total of 18 multiplications are required for two cell types.
However, it is reduced to 12 by sharing the common operations in the specific order as presented in step 5 of Fig. 7. By applying the proposed method in HOG feature calculation, the total number of required multiplications for each detection window is reduced up to 60.5%.

We identified key rules and presented the revised equations of trilinear interpolation. By using the proposed algorithm, detection speed can be significantly accelerated without any loss of detection accuracy since the redundant operations in trilinear interpolation are totally removed and HOG feature is calculated by cell-based operation. Therefore, the proposed algorithm is considerably useful in many applications in which both high detection rate and fast detection speed are strongly required.

5. EXPERIMENTAL RESULTS

We have conducted experiments in order to demonstrate that trilinear interpolation is superior to linear interpolation in terms of detection rate. In order to evaluate detection rate, we tested the pedestrian detectors on Daimler [11], INRIA [1], and MIT [12] pedestrian datasets using linear support vector machine (SVM) [13]. For Daimler datasets, 5,000 positive and 5,000 negative samples are used to train the detectors, and 10,000 positive and 12,870 negative samples are randomly selected to test them. For INRIA and MIT datasets, 2,878 positive and 12,180 negative samples are used to train the detectors, and 1,594 positive and 10,560 negative samples are randomly selected for testing. Detection rates on Daimler and INRIA+MIT datasets are shown in Fig. 9. As shown in the figure, the detection rate of the detector using trilinear interpolation is 78% at 10^-4 false positive per window (FPPW) on Daimler dataset and 84% on INRIA+MIT dataset. In comparison, the detection rate of the detector using linear interpolation is 65% on Daimler dataset and 82.5% on INRIA+MIT dataset. Therefore, trilinear interpolation technique should be applied in HOG feature calculation for high detection accuracy.
In the previous section, we proposed a novel algorithm to remove redundant operations totally in trilinear interpolation. In order to further reduce the redundant operations in the proposed trilinear interpolation, we shared common operations and found the best order. Table 5 shows the comparison results of the number of required multiplications in trilinear interpolation per detection window. As shown in the table, the number of required multiplications is reduced for each order of six kinds. By multiplying the operands in trilinear interpolation with the specific order (order #2), the number of required multiplications in trilinear interpolation is reduced up to 60.5% (from 272,160 to 107,496) in the proposed method.

<table>
<thead>
<tr>
<th>Table 5. Number of required multiplications in trilinear interpolation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>conventional method</td>
</tr>
</tbody>
</table>
| order #1 | 1) Q← M× β  
          2) R← Q× (A~D)  
          3) S← R× (G₁~G₄) | 110,736 | 59.3% ↓ |
| order #2 | 1) Q← M× β  
          2) R← Q× (G₁~G₄)  
          3) S← R× (A~D) | 107,496 | 60.5% ↓ |
| order #3 | 1) Q← M× (G₁~G₄)  
          2) R← Q× (A~D)  
          3) S← R× β | 117,180 | 56.9% ↓ |
| order #4 | 1) Q← M× (G₁~G₄)  
          2) R← Q× β  
          3) S← R× (A~D) | 113,400 | 58.3% ↓ |
| order #5 | 1) Q← M× (A~D)  
          2) R← Q× β  
          3) S← R× (G₁~G₄) | 117,540 | 56.8% ↓ |
| order #6 | 1) Q← M× (A~D)  
          2) R← Q× (G₁~G₄)  
          3) S← R× β | 118,800 | 56.4% ↓ |
In order to evaluate the actual processing time, we implemented our algorithm using C/C++ programming language and OpenCV2.0 library [14]. Table 6 shows the comparison results of CPU processing time per detection window. Five test images from Daimler dataset and an Intel Core i7-2600@3.40GHz with 16GB RAM are used in the evaluation. By comparing CPU processing time, we found that our algorithm reduces the processing time up to 58.6% for these test images.

<table>
<thead>
<tr>
<th></th>
<th>CPU processing time (ms)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conventional method</td>
<td>proposed method</td>
<td>comparison</td>
</tr>
<tr>
<td>test image 1</td>
<td>8.61</td>
<td>3.58</td>
<td>58.4% ↓</td>
</tr>
<tr>
<td>test image 2</td>
<td>8.51</td>
<td>3.56</td>
<td>58.2% ↓</td>
</tr>
<tr>
<td>test image 3</td>
<td>8.64</td>
<td>3.58</td>
<td>58.6% ↓</td>
</tr>
<tr>
<td>test image 4</td>
<td>8.37</td>
<td>3.54</td>
<td>57.7% ↓</td>
</tr>
<tr>
<td>test image 5</td>
<td>8.48</td>
<td>3.62</td>
<td>57.3% ↓</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

Since HOG feature is calculated by collecting histograms for all overlapping blocks in detection window, most cells in the overlapping blocks are considered redundantly in trilinear interpolation. Since trilinear interpolation requires the largest computational efforts in HOG feature calculation, it significantly increases the entire processing time of pedestrian detection. In this paper, therefore, we proposed a novel algorithm of fast HOG feature calculation to remove the redundant operations totally in trilinear interpolation. By identifying key rules and sharing common operations in trilinear interpolation, high detection rate is still achieved but the number of required multiplications is reduced up to 60.5%. Since trilinear interpolation is applied without any loss of accuracy and the number of required operations in HOG feature calculation is significantly reduced, the proposed algorithm can be applied to accelerate detection speed in many applications in which both high detection speed and accuracy are crucial.

REFERENCES


Soojin Kim was born in 1983 at Seoul, Korea. She received her B.S. and M.S. degrees in Electronics Engineering from Hankuk University of Foreign Studies, Korea, in 2007 and 2009, respectively. She received her Ph.D. degree from the Department of Electronics Engineering at Hankuk University of Foreign Studies, Korea, in 2014. From 2010 to 2013, she was a Researcher at the SoC Platform Research Center at Korea Electronics Technology Institute, Korea. Her research interests are the SoC architecture and design for multimedia and communications, pattern recognition and their application to vision systems.

Kyeongsoon Cho was born in 1959 at Seoul, Korea. He received his B.S. and M.S. degrees in Electronics Engineering from Seoul National University, Korea, in 1982 and 1984, respectively. He received his Ph.D. degree from the Department of Electrical and Computer Engineering at Carnegie Mellon University, U.S.A, in 1988. From 1988 to 1994, he was a Senior Researcher at the
Semiconductor ASIC Division of the Samsung Electronics Company. He was responsible for the research and development of the ASIC cell library and design automation. Since 1994, he has been a Professor at the Department of Electronics Engineering at Hankuk University of Foreign Studies. From 1999 to 2003, he was a Senior Director at Enhanced Chip Technology. From 2003 to 2004, he was a head of the CoAsia Korea Research and Development Center and a technical advisor of Dongu HiTek from 2005 to 2012. From 2005 to 2011, he was a Vice Director of the Collaborative Project for Excellence in System IC Technology sponsored by the Ministry of Knowledge Economy, Korea. Since 2012, he has been a Technical Advisor of DawinTech. His current research activities include the SoC architecture and design for multimedia and communications, SoC design and verification methodology.