Fast Template Matching Based on Normalized Cross Correlation with Adaptive Multilevel Winner Update

Shou-Der Wei and Shang-Hong Lai
Department of Computer Science
National Tsing Hua University
Hsinchu 300, Taiwan

Corresponding Author: Dr. Shang-Hong Lai
Department of Computer Science
National Tsing Hua University
Hsinchu 300, Taiwan
Email: lai@cs.nthu.edu.tw
Phone: +886-3-574-2958
Fax: +886-3-572-3694
Abstract

In this paper we propose a fast pattern matching algorithm based on the normalized cross correlation (NCC) criterion by combining adaptive multilevel partition with the winner update scheme to achieve very efficient search. This winner update scheme is applied in conjunction with an upper bound for the cross correlation derived from Cauchy-Schwarz inequality. To apply the winner update scheme in an efficient way, we partition the summation of cross correlation into different levels with the partition order determined by the gradient energies of the partitioned regions in the template. Thus, this winner update scheme in conjunction with the upper bound for NCC can be employed to skip unnecessary calculation. Experimental results show the proposed algorithm is very efficient for image matching under different lighting conditions.

Keywords: pattern matching, normalized cross correlation, winner update strategy, multi-level successive elimination, fast algorithms

1. Introduction

Pattern matching is widely used in many applications related to computer vision and image processing, such as object tracking, object detection, pattern recognition and video compression, etc. The pattern matching problem can be formulated as follows: Given a source image $I$ and a template image $T$ of size $M$-by-$N$, the pattern matching problem is to find the best match of template $T$ from the source image $I$ with minimum distortion or maximum correlation. The most popular similarity measures are the sum of absolute differences (SAD), the sum of squared differences (SSD) and the normalized cross correlation (NCC).

For some applications, such as the block motion estimation in video compression, the SAD and SSD measures have been widely used. For practical applications, a number of approximate block
matching methods have been proposed [1][2][3] and some optimal block matching solutions have been proposed [4][5][6], which have the same solution as that of full search but with less operations by using the early termination in the computation of SAD, given by

\[ SAD(x, y) = \sum_{i=1}^{M} \sum_{j=1}^{N} |T(i, j) - I(x + i, y + j)| \]  

(1)

In [7], a coarse-to-fine pruning algorithm with the pruning threshold determined from the lower resolution search space was presented. This search algorithm can be proved to provide the global solution. Hel-Or and Hel-Or [8] proposed a fast template matching method based on accumulating the distortion on the Walsh-Hadamard domain in the order of the associated frequency. In general, a small number of the first few projections can capture most of the distortion energy. By using a predefined threshold, they can early reject most of the impossible candidates very efficient.

Chen et al. [9] proposed a fast block matching algorithm based on the winner-update strategy, which can significantly reduce the computation and guarantee to find the globally optimal solution. In their algorithm, only the current winner location with the minimal accumulated distortion is considered for updating the accumulated distortion. This updating process is repeated until the winner has gone through all levels in the pyramids that are constructed from the template and the candidate windows for the distortion calculation.

In addition to SAD and SSD, NCC is also a popular similarity measure. The definition of NCC is given as follows:

\[ NCC(x, y) = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} I(x + i, y + j) \cdot T(i, j)}{\sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} I^2(x + i, y + j)} \cdot \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} T^2(i, j)}} \]  

(2)
The NCC measure is more robust than SAD and SSD under uniform illumination changes, so the NCC measure has been widely used in object recognition and industrial inspection. The correlation-like approach is very popular for image registration [15]. The traditional NCC needs to compute the numerator and denominator, which is very time-consuming. Lewis [12] employed the sum table scheme [13][14] to reduce the computation in the denominator. After building the sum table for the source image, the block squared intensity sum for a candidate at the position (x,y) in the source image can be calculated very efficiently with 4 simple operations. Although the sum table scheme can reduce the computation of the denominator in NCC, it is strongly demanded to simplify the computation involved in the numerator of NCC. The FFT-based method has been employed to calculate the cross correlation in the frequency domain [16]. It is very effective especially when the size of the template is comparable to the size of the search image. Stefano and Mattoccia [10][11] derived upper bounds for the cross correlation based on Jensen's and Cauchy-Schwarz inequalities to early terminate some search points. They partitioned the search window into two blocks and compute the partial cross correlation for the first block with the other blocks bounded by the upper bound. Then, the successive elimination algorithm (SEA) was applied to reject the impossible candidates successively.

In this paper, we propose an efficient NCC-based pattern search algorithm based on the winner update procedure combined with the adaptive multilevel partition scheme, which determines the elimination order based on the sum of the gradient magnitudes in the template. The winner update scheme is employed with the bound value derived from the Cauchy-Schwarz inequality to skip unnecessary calculation. The rest of this paper is organized as follow: we first briefly review some previous related works. Then, we present the proposed efficient NCC algorithm in section 3. The implementation details and experimental results are given in section 4 and 5, respectively. Finally, we conclude this paper in the last section.
2. Previous Works

In this section we introduce several previous works that are related to the proposed algorithm, including the winner update scheme, the multi-level uniform partition and the upper bound for cross correlation derived from Cauchy-Schwarz inequality.

2.1. The Winner Update Scheme

The winner update scheme [9] was employed to compare the SAD values in a hierarchical order of low bounds (LB) with the pyramid structure [6] and Minkowski's inequality as $LB_0 \leq LB_1 \leq \ldots \leq LB_k$. They globally chose the temporary winner with the minimal bound value from the candidate pool and update the winner level and low bound value, until the temporary winner reaches the final level. To sum up, the winner update algorithm is given as follows:

Algorithm 1: The Winner Update Scheme

Step 1: Calculate the $LB_i$ of all candidates and initialize the Hash Table

Repeat

Step 2: Select the candidate with minimal $LB$ in hash table as temporary winner

Step 3: Update the level and $LB$ of the temporary winner

Step 4: Push temporary winner into Hash Table.

Until the winner reaches the maximal level.

2.2. The SEA and Multi-level Uniform Partition

The Successive Elimination Algorithm (SEA) [4] used an upper bound for a block sum difference as the criterion to eliminate the impossible candidate blocks to reduce the computation of motion
estimation based on the SAD criterion. It can be easily shown that the following relation holds:

$$\left| T_{\text{sum}} - C(u, v) \right| \leq SAD(u, v)$$

(3)

where $T_{\text{sum}}$ and $C(u, v)$ represent the sums of the image intensities for the template and the window in the source image $I$ at position $(u, v)$, respectively, and $SAD(u, v)$ denotes the corresponding SAD value computed at the same window. Suppose $(\hat{u}, \hat{v})$ is the current optimal motion vector in the previous search process, and the corresponding SAD value is denoted by $SAD_{\text{min}}$. If $SAD(u, v)$ is less than $SAD_{\text{min}}$, then $SAD_{\text{min}}$, and $(\hat{u}, \hat{v})$ are updated accordingly. The boundary value $BV = \left| T_{\text{sum}} - C(u, v) \right|$ is used as the elimination criterion. For each candidate block, we can repeat this procedure to prune out a large portion of candidates. At the end, we can still find the best motion vector.

Later, Gao et al. extended the SEA to a multilevel successive elimination algorithm (MSEA) [5]. They partition the template uniformly to provide tighter and tighter boundary values from the lowest level to the highest level, as depicted in Figure 1. At level 0, the boundary value is equal to that for SEA and the boundary value at the final level is the same as the SAD value. The relation of boundary values at different levels is $BV_0 \leq BV_1 \leq \cdots \leq BV_{\log_2 N} = SAD$. MSEA builds an image pyramid structure of the current and reference blocks with $L = \log_2 N$ levels.

Fig 1: The levels of elimination order for a template in MSEA. Using only level 0 as the elimination criterion is the same as SEA, and the value at the final level is the same as SAD.

2.3. The Upper Bound of Multi-level NCC
Although the NCC measure is more robust than SAD, the computational cost of NCC is very high. The technique of sum table [12][13][14] can be used to reduce the computation involved in the denominator in NCC. To reduce the computational cost in the numerator, Stefano and Mattoccia [10][11] derived upper bounds for the cross correlation based on Jensen's and Cauchy-Schwarz inequalities to early terminate some impossible search points. Because the bound is not very tight, they partitioned the search window into two blocks and compute the partial cross correlation for the first block with the other blocks bounded by the upper bound. Then they used the SEA scheme to reject the impossible candidates successively. Based on the Cauchy-Schwarz inequality [11] given below

\[ \sum_{i=1}^{N} a_i \cdot b_i \leq \left( \sum_{i=1}^{N} a_i^2 \right)^{\frac{1}{2}} \cdot \left( \sum_{i=1}^{N} b_i^2 \right)^{\frac{1}{2}} \]

the upper bound (UB) of the cross correlation can be derived as follows:

\[ UB(x, y) = \sum_{i=1}^{M} \sum_{j=1}^{k} I(x+i, y+j) \cdot T(i, j) + \sum_{i=1}^{M} \sum_{j=k+1}^{N} I(x+i, y+j) \cdot T(i, j) \]

\[ \geq \sum_{i=1}^{M} \sum_{j=1}^{k} I(x+i, y+j) \cdot T(i, j) + \sum_{i=1}^{M} \sum_{j=k+1}^{N} I(x+i, y+j) \cdot T(i, j) \]

\[ = \sum_{i=1}^{M} \sum_{j=1}^{N} I(x+i, y+j) \cdot T(i, j) \]

Thus, the boundary value of NCC is given by

\[ BV(x, y) = \frac{UB(x, y)}{\left( \sum_{(i,j) \in W} I^2(x+i, y+j) \right)^{\frac{1}{2}} \cdot \left( \sum_{(i,j) \in W} T^2(i, j) \right)^{\frac{1}{2}}} \]
where \( W \) is the window for the template. Similar to the SEA scheme, the candidate at the position \((x,y)\) of image \( I \) is rejected if \( BV(x,y) < NCC_{\text{max}} \), and \( NCC_{\text{max}} \) is updated by \( NCC(x,y) \) if \( NCC(x,y) > NCC_{\text{max}} \).

3. The Proposed Fast NCC-Based Image Matching Algorithm

The cross correlation (CC) can be bounded by the Cauchy-Schwarz inequality as described above, but the bound is not very tight. As shown in the following equation, if we can divide the block into many sub-blocks and calculate the summation of the upper bounds of all sub-blocks, we can obtain a tighter bound.

\[
\sum_{i=1}^{N} a_i^2 \cdot \sum_{i=1}^{N} b_i^2 \geq \sqrt{\sum_{i=1}^{k} a_i^2} \cdot \sqrt{\sum_{i=1}^{k} b_i^2} + \sqrt{\sum_{i=k+1}^{N} a_i^2} \cdot \sqrt{\sum_{i=k+1}^{N} b_i^2} \geq \sum_{i=1}^{N} a_i \cdot b_i
\]  

(7)

Following the partitioning scheme of MSEA, we have many upper bounds for different partitioning levels and the relation between the upper bounds for different levels are given in the equation below.

\[
UB_0 \geq UB_1 \geq \cdots \geq UB_{L = \log_2 N} = CC
\]  

(8)

where the upper bound at the \( l \)-th level, \( UB_l \), is given by

\[
UB_l(x,y) = \sum_{k=1}^{l} \left( \sum_{(i,j) \in B_k} I(x+i,y+j) \cdot \sum_{(i,j) \in B_k} T^2(i,j) \right)
\]  

(9)

where \((B_1, B_2, \ldots, B_l)\) forms a partition of the template. Note that the upper bound at the final level is equal to the cross correlation. The MSEA scheme can provide tighter and tighter upper bounds as the partitioning level increases, but there are at most \( L = \log_2 N \) levels. With more partitioning levels, we have better chance to early reject most candidates. The upper bound of a
block is determined by the summation of the squared pixel values in the block. If the block is homogeneous, the upper bound is close to cross correlation value. Thus, the partitioning within homogeneous area has less chance to reject non-optimal candidates. However, they increase operation counts for measuring the similarity. These unnecessary operations should be avoided.

In other words, the blocks with large intensity variances normally contain more details. This means the block sum cannot represent the details of block, so partitioning a block with larger variance may produce a larger decrease in the upper bound. Consequently, in order to obtain a tighter bound in the early stage, it is reasonable to partition the blocks with large intensity variances into sub-blocks first.

The proposed NCC-based template matching algorithm consists of applying the winner update scheme with the successive upper bounds for the normalized correlation to achieve efficient search of the location with maximal NCC. The order of upper bound update for each template is very crucial to the efficiency of the winner update search. In this work, the block partition order is determined by using the adaptive block partitioning algorithm [17]. In order to obtain a tight bound in the early stage, it is reasonable to partition the blocks with large variances into sub-blocks first. From the consideration of efficient computation, we determine the elimination order by the sums of horizontal gradient magnitudes instead of the variances for the subblocks in the template. The block with the currently largest sum of gradient magnitudes is divided into 2x2 sub-blocks for consideration of further partitioning. Note that a block will not be further partitioned into 2x2 sub-blocks if its sum of gradient magnitudes is less than a given threshold $T$.

This partitioning process is repeated until the gradient magnitude sums of all subblocks are less than this threshold. In our implementation, we set the threshold $T$ to zero to find the optimal solution that is the same as that obtained by the full search NCC. Figure 2 depicts an example of block partitioning by using the proposed algorithm. After the above procedure, we obtain a
sequence of partition order that is stored in the elimination order queue. The adaptive algorithm of determining the block partitioning order is given in Algorithm 2.

![Block partitioning order example](image)

**Fig 2: An example of the block partitioning order.**

<table>
<thead>
<tr>
<th>Level 1</th>
<th>Level 2</th>
<th>Level 3</th>
<th>Level 4</th>
<th>Level 5</th>
<th>Level 6</th>
<th>Level 7</th>
</tr>
</thead>
</table>

**Algorithm 2: Algorithm for determining the adaptive block partitioning order**

1. Push the template image block into the queue
2. **REPEAT**
   1. Select the block with the largest sum of gradient magnitudes from the queue.
   2. Divide the selected block into four sub-blocks and calculate their sums of gradient magnitudes.
   3. Check the four sub-blocks and push each sub-block into the queue if its sum of gradient magnitudes is greater than a given threshold $T$.
3. **UNTIL** the queue is empty
4. Store the partitioning order in the elimination order queue

With the block partitioning order obtained by using the above algorithm, we have the relation of upper bounds for different levels as $UB_0 \geq UB_1 \geq \cdots \geq UB_{max.} \geq CC$. We can calculate the boundary values from equation (6) and have the relation of boundary values of different levels as $BV_0 \geq BV_1 \geq \cdots \geq BV_{max.} \geq NCC$. If $T$ is set to 0, then $BV_{max.}$ is the same as the true NCC value.

The value $BV_i$ is closer to NCC as the level increases. Based on the above relation for the boundary values, we can apply the winner update scheme on it. At first, we calculate $BV_i$ for all candidates, and then at each iteration we choose the candidate with the current maximal $BV_i$ as the winner to update its level and calculate its new boundary value $BV_{i+1}$. This procedure is repeated until the chosen winner reaches the maximal level, thus the corresponding $BV_{max.}$ is the
same as the maximal NCC value. The following is the algorithm of applying winner update scheme with the adaptive block partition for NCC.

**Algorithm 3: The proposed fast NCC-based pattern matching algorithm**

1. Build the template pyramid and source image pyramid
2. Determine the elimination order by Algorithm 2.
3. Calculate the $BV_1$ of all candidates and initialize the Hash Table
4. Apply the winner update scheme

**Repeat**

4.1: Select the candidate $I(x, y)$ with maximal $BV$ in hash table as the temporary winner
4.2: Update the level and $BV$ of the temporary winner
   4.2.1. Retrieve the next partitioning in the next level $l$.
   4.2.2. Calculate $UB_l$ for level $l$. Compute $BV_l = UB_l / (||T|| \cdot ||I(x, y)||)$
   4.2.3. Push the temporary winner into Hash Table.

**Until** the winner reaches the maximal level.

We can also apply the proposed NCC-based pattern matching algorithm on the zero-mean normalized cross correlation (ZNCC) by rewriting it in the following form

$$
ZNCC(x, y) = \frac{\sum_{i=1}^{M} \sum_{j=1}^{N} (I(x + i, y + j) - \bar{I}(x, y)) \cdot (T(i, j) - \bar{T})}{\sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} (I(x + i, y + j) - \bar{I}(x, y))^2} \cdot \sqrt{\sum_{i=1}^{M} \sum_{j=1}^{N} (T(i, j) - \bar{T})^2}} \sum_{i=1}^{M} \sum_{j=1}^{N} I(x + i, y + j) \cdot T(i, j) - MN\bar{I}(x, y)\bar{T})
$$

where

$$
\bar{I}(x, y) = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} I(x + i, y + j), \quad \bar{T} = \sum_{i=1}^{M} \sum_{j=1}^{N} T(i, j)
$$
Note that the summations of the image intensities and the squared intensities in the local window at each location in the image can be computed very efficiently by using the corresponding integral images.

4. Implementation Details

Similar to the method proposed in [9], we also used a hash table to find the temporary winner. We first initialize the hash table by calculating the $BV_i$ of all candidates and determining the bucket number for all candidates by their $BV_i$ values, the candidates with higher $BV_i$ values will be located in the front buckets. Since the $BV$ value is in the range of 0 to 1, we determine the bucket number by the following equation

$$BucketNo = MaxBucketNo - BV * MaxBucketNo$$

At each iteration, we retrieve the candidate with the current maximal $BV_i$ from the first non-empty bucket. After updating the level and the new $BV_{i+1}$ for the winner, push the winner back to the hash table and determine its new bucket number by the vale $BV_{i+1}$. Figure 3 depicts an example of hash table with 10000 buckets. The candidate with BV value in the range of 0.9999 to 0.9998 will be located in bucket 1, and so on. We deal with the problem of multiple candidates in one bucket with the linking list.
In section 3, we bound the cross correlation by the square sum in the subblocks. For efficient implementation of the proposed algorithm, we can use the integral image [13][14] to calculate the intensity square sum for subblocks. In our implementation, we use block sum pyramid [6][9] to calculate the intensity square sum for pixels in the subblocks for square template very efficiently. We can retrieve the intensity square sum for any subblock in the partition order from the source image pyramid and the template pyramid.

After building the source image pyramid, we can retrieve the intensity square sums of the subblocks based on the partition order for all candidates. Figure 4 shows an example of calculating the block sum of candidate \( C(2,3) \) by the partition order. The sizes of the source image and the template in Figure 4 are 8-by-8 and 4-by-4, respectively, and the maximal level of pyramid is 2. The right side of Figure 4 is an example of partition order, and following the arrows we can obtain the block sums of the subblocks in the corresponding layers and positions in the source pyramid.
Fig 4: An example of obtaining the block sum from the source image pyramid for the candidate $C(2,3)$ based on a given block partition order.

5. Experimental Results

In this section, we show the efficiency improvement of the proposed algorithm for NCC-based pattern matching. The proposed algorithm adaptively partitioned the image block into many subblocks to obtain successively tighter upper bounds for the cross correlation. To compare the efficiency of the proposed algorithm, we also implement the full-search (NCC), BPC[11] with the correlation ratio $Cr=50\%$ and the winner update scheme (WUS) [9] for fast and optimal SAD-based template matching. Note that all the above NCC-based methods used the sum table to reduce the computation in the denominator of NCC. The experiments were performed on a PC with an Intel P4 2.6G CPU and 512M RAM. In our experiment, we use the sailboat image of size 512-by-512 and its noisy version as the source images and six template images of size 64x64 selected from the original sailboat image as shown in Figure 5 and Figure 6, respectively. The templates in Figure 6(d), (e), and (f) are the brighter version (increase 40% brightness) of the original templates in Figure 6(a), (b), and (c), respectively. To compare the robustness and the
efficiency of the proposed algorithms, we add random Gaussian noises with \( \sigma = 8 \) onto the search image as shown in Figure 5(b) and compare the performance of the pattern search on the noisy image. The full search, BPC and the proposed algorithm are guaranteed to find the optimal NCC solution from the search image, so we only focus on the comparison of search time required for different algorithms. The execution time required for the full search, BPC and the proposed algorithm is shown in Table 1 and 2. All these three algorithms used the sum table to reduce the computation of denominator in equation (2). For efficiently calculating the bound of the numerator, we also used the approach of BSPA [6] to build two block square sum pyramids for both the intensity image and the gradient map. For a fair comparison, the execution time shown in this section includes the time of memory allocation for sum table and pyramids, and building sum table, pyramids and the gradient map. Table 3 and Table 4 show the experimental results of applying different templates of size 128-by-128 as shown in Figure 7. The templates in Figure 7(d), (e), and (f) are the brighter versions (40% brightness increase) of the original templates in Figure 7(a), (b), and (c), respectively. All of these experimental results show the significantly improved efficiency of the proposed algorithm for the NCC-based pattern matching compared to the full search and BPC methods. In addition, the computational efficiency of the proposed algorithm is similar to that of the SAD-based winner update method by Chen et al. [9], but their method failed to find the optimal solutions when there are illumination changes in this experiment.
Fig 5: (a) The original sailboat image and (b) the noisy sailboat image added with Gaussian noise with $\sigma = 8$.

Fig 6: (a), (b) & (c) are three template images of size 64-by-64, and (d), (e) & (f) are their corresponding brighter versions.

Fig 7: (a), (b) & (c) are three template images of size 128-by-128, and (d), (e) & (f) are their corresponding brighter versions.
Table 1: The execution time (in milliseconds) of applying the full-search (NCC), BPC, WUS (SAD), and the proposal algorithm to the NCC-based pattern matching with the six templates shown in Figure 6(a)~(f) and the source image shown in Figure 5(a).

<table>
<thead>
<tr>
<th>ms</th>
<th>T(a)</th>
<th>T(b)</th>
<th>T(c)</th>
<th>T(d)</th>
<th>T(e)</th>
<th>T(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-Search (NCC)</td>
<td>4782</td>
<td>4672</td>
<td>4672</td>
<td>4703</td>
<td>4672</td>
<td>4688</td>
</tr>
<tr>
<td>BPC</td>
<td>2547</td>
<td>2593</td>
<td>3000</td>
<td>2532</td>
<td>2657</td>
<td>2906</td>
</tr>
<tr>
<td>WUS (SAD)</td>
<td>42</td>
<td>43</td>
<td>42</td>
<td>*149</td>
<td>*117</td>
<td>*90</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>71</td>
<td>69</td>
<td>69</td>
<td>70</td>
<td>70</td>
<td>73</td>
</tr>
</tbody>
</table>

* indicates that method finds an incorrect matching result.

Table 2: The execution time (in milliseconds) of applying the full search NCC, BPC, WUS (SAD), and the proposed algorithm to the NCC-based pattern matching with the six templates shown in Figure 6(a)~(f) and the noisy source image shown in Figure 5(b).

<table>
<thead>
<tr>
<th>ms</th>
<th>T(a)</th>
<th>T(b)</th>
<th>T(c)</th>
<th>T(d)</th>
<th>T(e)</th>
<th>T(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-Search (NCC)</td>
<td>4672</td>
<td>4672</td>
<td>4656</td>
<td>4656</td>
<td>4812</td>
<td>4735</td>
</tr>
<tr>
<td>BPC</td>
<td>2531</td>
<td>2656</td>
<td>3000</td>
<td>2531</td>
<td>2797</td>
<td>2907</td>
</tr>
<tr>
<td>WUS (SAD)</td>
<td>53</td>
<td>54</td>
<td>57</td>
<td>*168</td>
<td>*117</td>
<td>*96</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>80</td>
<td>71</td>
<td>78</td>
<td>120</td>
<td>77</td>
<td>125</td>
</tr>
</tbody>
</table>

* indicates that method finds an incorrect matching result.

Table 3: The execution time (in milliseconds) of applying the full-search NCC, BPC, WUS (SAD), and the proposed algorithm to the NCC-based pattern matching with the six templates of size 128-by-128 shown in Figure 7(a)~(f) and the source image shown in Figure 5(a).

<table>
<thead>
<tr>
<th>ms</th>
<th>T(a)</th>
<th>T(b)</th>
<th>T(c)</th>
<th>T(d)</th>
<th>T(e)</th>
<th>T(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-Search (NCC)</td>
<td>13422</td>
<td>13407</td>
<td>13407</td>
<td>13376</td>
<td>13376</td>
<td>13376</td>
</tr>
<tr>
<td>BPC</td>
<td>7281</td>
<td>8063</td>
<td>9110</td>
<td>7219</td>
<td>7703</td>
<td>8406</td>
</tr>
<tr>
<td>WUS (SAD)</td>
<td>88</td>
<td>45</td>
<td>47</td>
<td>*160</td>
<td>*89</td>
<td>*111</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>103</td>
<td>94</td>
<td>94</td>
<td>105</td>
<td>92</td>
<td>90</td>
</tr>
</tbody>
</table>

* indicates that method finds an incorrect matching result.
Table 4: The execution time (in milliseconds) of applying the full-search NCC, BPC, WUS (SAD), and the proposed algorithm to the NCC-based pattern matching with the six templates of size 128-by-128 shown in Figure 7(a)–(f) and the noisy source image shown in Figure 5(b).

<table>
<thead>
<tr>
<th></th>
<th>T(a)</th>
<th>T(b)</th>
<th>T(c)</th>
<th>T(d)</th>
<th>T(e)</th>
<th>T(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-Search (NCC)</td>
<td>13360</td>
<td>13391</td>
<td>13391</td>
<td>13313</td>
<td>13282</td>
<td>13329</td>
</tr>
<tr>
<td>BPC</td>
<td>7282</td>
<td>8125</td>
<td>9187</td>
<td>7219</td>
<td>7828</td>
<td>8656</td>
</tr>
<tr>
<td>WUS (SAD)</td>
<td>101</td>
<td>104</td>
<td>58</td>
<td>*246</td>
<td>*91</td>
<td>*117</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>172</td>
<td>121</td>
<td>114</td>
<td>215</td>
<td>145</td>
<td>127</td>
</tr>
</tbody>
</table>

* indicates that method finds an incorrect matching result.

To see the influence of noise corruption on the efficiency of winner update in the proposed algorithm, we show in Figure 8 the total numbers of winner updates at all locations in the search image to find the template shown in Figure 6(a) from the original sailboat image and its noisy version shown in Figure 5(a) and Figure 5(b), respectively. The total number of candidates is 201601 (449x449) and the average winner update counts are 1.0072 and 1.1495 for the clean and noisy images, respectively. Note that the peaks of winner update counts shown in the figures correspond to the final searched location. It is evident that the proposed algorithm is very efficient since all the locations other than the peak solution have very small number of winner updates. We can also see that the winner updates in the noisy search image as shown in Figure 8(b) are a little bit more than those in the clean image as shown in Figure 8(a).
Fig. 8: The total numbers of winner updates by applying the proposed method to find the template in Figure 6(a) from the source sailboat image with (a) no noise (Figure 5(a)) and (b) additive Gaussian noises (Figure 5(b)).
The execution of the proposed algorithm can be divided to 4 stages: stage 1 consists of memory allocation and pyramid construction, stage 2 is to determine the partition order, stage 3 involves calculating the boundary values $BV_1$ of all candidates and initializing the hash table, and stage 4 is the winner updating process. The executing time of these four stages when applying the proposed algorithm to four templates shown in Figure 6(a), 6(d), 7(a) and 7(d) and the noisy source image in Figure 5(b) is shown in Table 5. We can see from the first row of this table that the time of allocating memory and building pyramids occupies almost half of the total execution time. The proposed algorithm can be even more efficient if stage 1 can be done off-line for some applications. The execution time in stage 2 depends on the size of the template. For some applications that the elimination order can be determined off-line, the execution time in the second stage can be saved. The execution time in stage 4 heavily depends on the content in the template. From Table 5, we can see the bright version of the templates (Figure 6(d) and 7(d)) need more execution time than that required for the original templates.

Table 5: The execution time (in milliseconds) for all the four stages by applying the proposed algorithm to four templates in Figure 6(a), 6(d), 7(a) and 7(d) and the noisy source image in Figure 5(b).

<table>
<thead>
<tr>
<th>template</th>
<th>source</th>
<th>Stage1</th>
<th>Stage2</th>
<th>Stage3</th>
<th>Stage4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 6(a)</td>
<td>Figure 5(b)</td>
<td>30</td>
<td>3</td>
<td>33</td>
<td>4</td>
</tr>
<tr>
<td>Figure 6(d)</td>
<td>Figure 5(b)</td>
<td>30</td>
<td>3</td>
<td>33</td>
<td>54</td>
</tr>
<tr>
<td>Figure 7(a)</td>
<td>Figure 5(b)</td>
<td>38</td>
<td>37,</td>
<td>23</td>
<td>7</td>
</tr>
<tr>
<td>Figure 7(d)</td>
<td>Figure 5(b)</td>
<td>38</td>
<td>37,</td>
<td>23</td>
<td>117</td>
</tr>
</tbody>
</table>

To show the performance of the proposed algorithm for different template matching tasks, we randomly selected 100 different templates of sizes 64-by-64 and 128-by-128 from the sailboat
and airplane images. The histograms of the execution time are depicted in Figure 9 and 10. From these figures we can see the proposed method takes about 60-140 milliseconds to find most of these randomly selected templates. The execution time required for the proposed template matching algorithm depends on the image content in the template and the search images. The more unique content in the template, the faster the proposed algorithm finds the template from the image. In Figure 10(a), there is a special case with execution time more than 700 milliseconds. For this case, the template in the airplane image is shown in Figure 11(a), and the winner update numbers of applying the proposed algorithm on this homogeneous template are depicted in Figure 11(b). Note that there is lack of any edge, texture or image features in this template, thus making the search of this template from the image very inefficient. Even for this particular template, the proposed algorithm takes about one sixth of the execution time of the full-search NCC method.

![Histograms](image)

Figure 9. The histograms of the execution time (in milliseconds) required for applying the proposed algorithm to find 100 randomly selected templates of size (a) 64-by-64 and (b) 128-by-128 from the sailboat image.
Figure 10. The histograms of executing time (in milliseconds) required for applying the proposed algorithm to find 100 randomly selected templates of size (a) 64x64 and (b) 128x128 from the airplane image.

Figure 11. (a) The homogeneous template window randomly selected from the airplane image, and (b) the winner update numbers of applying the proposed algorithm to find this template from the original image.
In the following experiments, we use the templates that are not cropped from the source images. Figure 12(a) and 12(b) are different images taken by the same camera under quite different illumination conditions. The region inside the white rectangle in Figure 12(a) was selected as the template and the image in Figure 12(b) is the source image. The results of applying the proposed algorithm and the SAD-based WUS method [9] to search the template from the source image are shown in Figure 12(b) and Figure 12(c), respectively. From the results, we can see the proposed NCC-based pattern matching algorithm is robust against lighting change, while the SAD-based WUS method failed to find the true solution in this example. The sizes of template and source image are 128x128 and 480x480, respectively. The execution time for the proposed algorithm and the WUS method is 160 ms and 141 ms, respectively. Figure 13 shows another experiment for finding the object from images acquired under different lighting condition and added with random noises. We add zero-mean random Gaussian noises with $\sigma=8$ to the real source image as shown in Figure 13(b). From Figure 13, we can see the proposed algorithm can accurately find the template under different lighting conditions, but the WUS method still cannot find the correct location. In this experiment, the sizes of the template and source images are 64x64 and 480x480, respectively. The execution time for the proposed algorithm and the WUS method is 282 ms and 238 ms, respectively.
Figure 12. (a) The region inside the white rectangle is the template. The results of applying (b) the proposed algorithm and (c) the SAD-based WUS method on a search image acquired under a different lighting condition are shown with the white rectangles.
Figure 13. (a) The region in the white rectangle is the template. The results of applying (b) the proposed algorithm and (c) the SAD-based WUS method to a source image, acquired under different lighting condition and added with random noises, are shown with the white rectangles.

6. Conclusions
In this paper, we proposed a very efficient algorithm for NCC-based pattern search from an image. To achieve very efficient computation, we partition the summation of cross correlation into different levels and apply the winner update scheme to find the location with maximal NCC. The block partition order is adaptively determined by the sum of gradient magnitudes for each partitioned regions in the template. For the NCC pattern search, the winner update scheme is applied in conjunction with the upper bound for the cross correlation derived from Cauchy-Schwarz inequality. The experimental results show the proposed algorithm is very efficient and robust for pattern matching under uniform illumination change and noisy environments.

References


