Computationally Efficient Histogram Extraction for Rectangular Image Regions

Fatih Porikli

TR2005-014 June 2005

Abstract

We present a novel method, which we refer as an integral histogram, to compute the histograms of all possible target regions in an image. Our method is computationally superior and makes it possible to employ even an exhaustive search process in real-time, which was impractical before. Furthermore, it enables the description of higher level histogram features. To accomplish fast extraction, we exploit the spatial arrangement of image points, and recursively propagate an aggregated histogram by starting from an origin and traversing through the remaining images along a given scan-line. At each step, we update a single histogram bin using the values of integral histogram at the previously visited neighboring points. After integral histogram is propagated, histogram of any target region can be computed easily by using simple arithmetic operations. Our numerical analysis proves that the integral histogram method drastically decreases the amount of the required computations.

Proc. of Real-Time Image Processing

Computationally Efficient Histogram Extraction for Rectangular Image Regions

Fatih Porikli*,

Mitsubishi Electric Research Laboratories, Cambridge, USA

ABSTRACT

We present a novel method, which we refer as an integral histogram, to compute the histograms of all possible target regions in an image. Our method is computationally superior and makes it possible to employ even an exhaustive search process in real-time, which was impractical before. Furthermore, it enables the description of higher level histogram features. To accomplish fast extraction, we exploit the spatial arrangement of image points, and recursively propagate an aggregated histogram by starting from an origin and traversing through the remaining images along a given scan-line. At each step, we update a single histogram bin using the values of integral histogram at the previously visited neighboring points. After integral histogram is propagated, histogram of any target region can be computed easily by using simple arithmetic operations. Our numerical analysis proves that the integral histogram method drastically decreases the amount of the required computations.

Keywords: Histogram Computation, Complexity Reduction

1. INTRODUCTION

A histogram is an array of numbers in which each element, bin, counts the number of pixels having the same color values. Thus, a histogram is a mapping from the set of image color values to the set of non-negative real numbers. From a probabilistic point of view, the normalization of an histogram results in a function that is most akin to the probability density function of the image. Histograms are among the most common features used in many computer vision tasks from object based retrieval,[9],[8] to segmentation,[13] to detection[2] to tracking.[5]

One bottleneck of the existing approaches is the speed of the histogram extraction and search processes. It is obvious that only an exhaustive search can provide the global optimum, although such a search is computationally very expensive. Still, several sub-optimal techniques that are powered by gradient descent methods and application specific constraints have been developed to deliver accelerated alternatives to the basic exhaustive search. However, computer vision tasks that rely on the optimal solutions, such as detection and tracking, still demand a theoretical breakthrough in histogram extraction as much as an powerful computers to crunch numbers.

It is possible to calculate the sum of the values within rectangular windows in linear time without repeating the summation operator for each possible window.[4] A constant number of operation for each rectangular sum is needed to compute such sums over distinct rectangles many times. A cumulative image function is defined such that each element of this function holds the sum of all values to the left and above of the pixel including the value of the pixel itself. The cumulative image can be computed for all pixels with four arithmetic operations per pixel. We start in the top left corner, keep going first to the right and then down, and use the formula that the value of the cumulative image at the current pixel equal to the addition of the left and the up pixel and subtraction of the upper left pixel's cumulative values. After the cumulative image is computed, the sum of image function in a rectangle can be computed with another four arithmetic operations with appropriate modifications at the border. Thus with a linear amount of computation, the sum of image function over any rectangle can be computed in linear time.

However, this method is restricted only to the sum of the intensity values and it fails to provide more descriptive features for detection process.

To address the computational requirements of detection tasks, we develop a fast method to compute histograms of all possible target regions in a given image. We take advantage of the spatial positioning of points in the image, and propagate an aggregated function, which we refer as the integral histogram, starting from an origin point and traversing through

*fatih@merl.com, phone: 1.617.621.7586

the remaining points along a scan-line. We iterate an integral histogram at the current point using the histograms of the previously processed neighboring points. At each step, we increase the value of the bin that the current point fits into the bin's range. After the integral histogram is obtained for each image point, histograms of target regions can be computed easily by using the integral histogram at the corner points of those regions without reconstruction of separate histograms for every single regions. The histogram of a rectangular region is computed by intersecting the integral histogram at the four corner points.

Integral histogram method has several advantages: First, it is computationally superior than the existing approaches. Integral histogram makes it possible to execute even an exhaustive histogram search process in the image, which was infeasible with conventional approaches. It enables description of advanced histogram features.

We give a throughout analysis of the different number of bins and bin structures. Our experiments confirm that the integral histogram method drastically decreases the amount of computations needed to obtain a multitude of histograms, thus, it significantly improves the speed of search algorithms based on histogram comparison.

In the next section, we summarize the previous work. In section 3, we introduce the integral histogram formulation in detail. In section 4, we give a computational complexity analysis by considering different scenarios.

2. BACKGROUND

A conventional approach of measuring distances between a given histogram and histograms of all possible target regions is an exhaustive search. This process requires generation of histograms for the regions centered at every possible points. In case the search should be done at different scales, i.e. different target region sizes, the whole process should be repeated as many times as the number of scales. We give a pseudo-code of the conventional histogram search using below:

Algorithm 2.1: HISTOGRAMCONVENTIONAL(T)

```
for each $possible\ scale$
  do
    for each $possible\ point$
      do
        for each $target\ point$
          do
            $GetCurrentValue$
            $FindBin$
            $GetBinValue$
            $IncreaseBinValue$
        for each $bin$
          do $Normalize$
        $ComputeHistogramDistance$
```

To our knowledge, the conventional approach is the only solution (other than the presented integral histogram method) that guarantees to find the global optimum in histogram based search.

3. INTEGRAL HISTOGRAM FORMULATION

Integral histogram is a recursive propagation method works in an image. It is a superset of the cumulative image formulation mentioned in the previous section.

Suppose our image function f is a defined such that $\mathbf{x} \rightarrow f(\mathbf{x})$ where $\mathbf{x} = [x_1, x_2]$ is an image point. Our function maps each image point to a color value, i.e. $f(\mathbf{x}) = [g_1, .., g_k]$. Let assume the image to be bounded within the range N_1, N_2, i.e. $0 \leq x_i \leq N_i$.

We define an integral histogram $H(\mathbf{x}, b)$ along a scan-line of image points $\mathbf{x}_0, \mathbf{x}_1, ..$ such as;

$$H(\mathbf{x}, b) = \bigcup_{\mathbf{p}=0}^{\mathbf{x}} Q(f(\mathbf{p})) \tag{1}$$

where $Q(.)$ gives the corresponding bin of the current point, and $\cup$ is the union operator that is defined as follows: the value of the bin b of $H(\mathbf{x}, b)$ is equal to the sum of the previously visited points's histogram bin values i.e. sum of all $Q(f(\mathbf{p}))$ while $\mathbf{p} < \mathbf{x}$. In other words, $H(\mathbf{x}, b)$ is the histogram of the larger region between the origin and current point; $0 \leq p_1 \leq x_1$, $0 \leq p_2 \leq x_2$, ..., etc. Note that, $H(\mathbf{N}, b)$ is equal to the histogram of all points in our image since $\mathbf{N} = [N_1, N_2]$ is our boundary. Therefore, the integral histogram can be obtained recursively as

$$H(\mathbf{x}) = H(\mathbf{x} - 1) \cup Q(f(\mathbf{x})) \tag{2}$$

using the initial condition $H(0) = 0$, i.e. all bins are empty at the origin.

Then, the histogram of a target region $T = [\mathbf{p}^-, \mathbf{p}^+]$ where $\mathbf{p}^- < \mathbf{p}^+$ can be computed using the propagated integral histogram values at the bounding points of the region as;

$$h(T, b) \quad = \quad H(\mathbf{p}^+, b) - \sum_{i \neq j}^{2} H([p_i^-, p_j^+], b) + H(\mathbf{p}^-, b) \tag{3}$$

which becomes $h(T, b) = H(p_1^+, p_2^+, b) - H(p_1^-, p_2^+, b) - H(p_1^+, p_2^-, b) + H(p_1^-, p_2^-, b)$ for an image. Note that the region is bounded within $p_1^- \leq x_1 \leq p_1^+, p_2^- \leq x_2 \leq p_2^+$,

As opposed to the conventional histogram computation, the integral histogram method does not repeat the histogram extraction for each possible region as given in the pseudo-code below:

Algorithm 3.1: HISTOGRAMINTEGRAL(T)

```
for each $possible\ point$
  do
    for each $target\ point$
      do $PropagateIntegral$
    $GetCurrentValue$
    $FindBin$
    $GetBinValue$
    $IncreaseBinValue$
for each $possible\ scale$
  do
    for each $possible\ point\ p \in Image$
      do
        for each $bin$
          do
            $ComputeIntersection$
            $Normalize$
        $ComputeHistogramDistance$
```

In case of a $N_1 \times N_2$ gray level image, the scan-line can be assigned as top to bottom and left to right order, and the recursion can be written as

$$H(x_1, x_2, b) \quad = \quad H(x_1 - 1, x_2, b) + H(x_1, x_2 - 1, b) - H(x_1 - 1, x_2 - 1, b) + Q(f(x_1, x_2)) \tag{4}$$

for all $b = 1, ..B$. This propagation assigns the histogram bins of the current point by using the intersection of the bins of the three previous histograms, with an increment of the value of the bin that the current image point belongs to as illustrated in Fig. 1.

4. COMPLEXITY ANALYSIS

We analyze both conventional and integral histogram methods in terms of the relative cost of processor operations, which is usually measured against the cost of an integer addition operation. Relative costs of several operations reported in the literature as well as our own observations are presented in the Table 1.

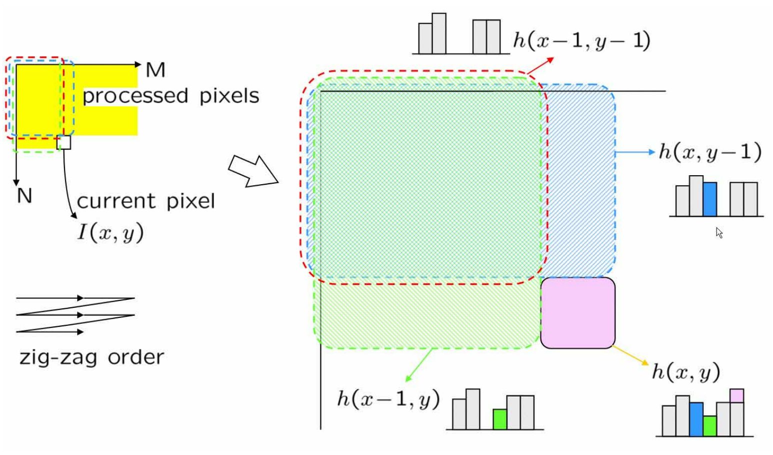


Figure 1. Propagation of integral histogram. Yellow indicates already traversed points. At each recursion along the zig-zag order, the current integral histogram is obtained from the integral histogram values of the three neighbors, and the bin that the corresponds to current points value is increased by one.

	A	B	C	D	E
Integer addition	1	1	1	1	1
Integer multiply	4	4	1.2	24	4
Integer divide	6	36	4.4	-	75
Floating-point addition	20	3	1	4.2	4
Floating-point multiply	20	5	1.2	113	4
Floating-point divide	20	38	1.2	-	100
Type conversion	20	-	-	-	105
Bit-wise shift	1	-	-	-	2

Table 1. Column-A is the relative cost of the basic processor operators as given in Qualline.[6] Column-B is the cost of the operators executed on a P4 processor that uses streaming SIMD extensions and Prescott processor arithmetic operations.[10] Column-C is the corresponding numbers of P3 MMX arithmetic operations.[11] Column-D is the relative costs on a P4 working running C++ compiler.[7] We also did our experiments to determine the relative costs (column-E).

Since the cost of the array indexing becomes comparable, we also make an assessment of indexing operators. In Qualline,[6] it is explained that an ordinary indexing for an 2-dimensional array requires 2 additions, 1 multiplication, and 2 logical operators, which has a total relative cost of 8. By using a look-up table of pointers, the multiplication can be replaced by 1 pointer referencing. However, we found that the cost of an 2-dimensional array indexing is approximately 11 in our experiments.

The type of the input data, i.e. whether it is integer or floating point, affects the computational load. Thus, we present a detailed analysis of both methods using these data types. Our analysis can be extended to fixed point operations as well.

4.1. Integer Data Type

Suppose the input image is a 2-dimensional array where the range of values at each dimension is N_1, N_2 with associated k-dimensional tensors, the histograms are k-dimensional with B identical bins dedicated for each dimension, and bin size is an integer number. Furthermore, assume our target window (where we compute a histogram, i.e. our target object size) is $M_1 \times M_2$. The conventional histogram matching algorithm requires these main tasks:

- Get current values: 1 array indexing (2-dimensional) and k additions,
- Find bin: k integer divisions (or floating point multiplication and float-to-integer conversion),
- Get bin value: k-dimensional array indexing,

- Increase bin value: 1 integer addition,
- Normalize: B^k floating point multiplication.

In terms of the relative cost, the conventional algorithm requires $11 + k$ operations for getting the current values in the input tensor, $75k$ operations to compute the corresponding bin indices, 1 operation (for 1 addition) to increase the bin value. Computing bin indices can be done by a floating-point multiplication and then float-to-integer conversion, however the cost of this option ($109k$) is higher than the division itself ($75k$). After all the $M_1 \times M_2$ points in the target window are processed, the histogram bins are normalized with the number of points, which requires B^k floating point multiplications, thus $4B^k$ operations in terms of the relative cost. Note that the previous computations are repeated for each of the $N_1 \times N_2$ histograms matches. Then, the total number of operations needed for all candidates becomes

$$\left[(12 + 76k) \prod_j^2 M_j + 4B^k\right] \prod_i^2 N_i \tag{5}$$

Note that, for different window size combinations $M_s = 1, .., S_s$, where S_s represents the maximum size of the range for the dimension s, the above algorithm is repeated over again;

$$\left[(12 + 76k) \prod_j^2 M_j + 4B^k\right] \prod_i^2 N_i \prod_s^2 S_s \tag{6}$$

On the other hand, the integral histogram method only needs

- Propagate integral: 3 k-dimensional array indexing and $2k$ integer additions,
- Get current values: 1 array indexing (2-dimensional) and k additions,
- Find bin: k integer divisions (or floating point multiplication and float-to-integer conversion),
- Get bin value: k-dimensional array indexing,
- Increase bin value: 1 integer addition,
- Compute intersection: 4 k-dimensional array indexing and $3k$ integer additions,
- Normalize: B^k floating point multiplications.

Thus, the propagation takes $3(7k - 3) + 2k = 23k - 9$ operations in addition to the cost of getting the current value of the tensor values $(11 + k)$, finding the indices of the corresponding bin $(75k)$, and accumulating the obtained bin value (1), which is repeated for all points in the data space. Then, we find that $(3 + 99k) \prod_i^2 N_i$ operations are required to construct the integral histogram. We compute the histogram intersection using $4(7k-3)+3k = 31k-12$ operations, and normalize the result using B^k floating point divisions ($4B^k$ operations) for each histogram. Then, the cost of all $N_1 \times N_2$ histograms and all possible search window dimension matches is only

$$\left[3 + 99k + (31k - 12 + 4B^k) \prod_s^2 S_s\right] \prod_i^2 N_i \tag{7}$$

Of course, both methods compute histogram distances using the given metric in addition to the above costs.

We define a ratio of the computational load of the conventional approach versus the integral histogram method;

$$r = \frac{[(12 + 76k) \prod_j^2 M_j + 4B^k] \prod_s^2 S_s}{3 + 99k + (31k - 12 + 4B^k) \prod_s^2 S_s} \tag{8}$$

4.2. Floating Point Data:

Use of floating point data increases the cost of the divisions in the computation of the bin indices. The cost increases from $75k$ for each point to $100k$. The bin value increment cost becomes 4, which was 1 before. The total cost for the conventional approach becomes;

$$\left[(15+101k)\prod_{j}^{2} M_j + 4B^k\right]\prod_{i}^{2} N_i \prod_{s}^{2} S_s \tag{9}$$

For the integral histogram method, the complexity of the step for finding bin indices increases to $100k$. In the propagation stage, the cost of additions rises from $2k$ to $8k$. In the intersection computation, the cost becomes $4(7k-3)+12k = 40k-12$. The total cost becomes;

$$\left[3+130k(40k-12+4B^k)\prod_{s}^{2} S_s\right]\prod_{i}^{2} N_i \tag{10}$$

Note that, in addition to above costs, the conventional approach has another important disadvantage. After each computation, it needs the histogram array values to be destroyed, which creates additional overhead.

4.3. Gray Level Images

For a $M_1 \times M_2$ gray level image and a search window size range S_1, S_2, the parameters of the above analysis becomes $k = 1$, and the ratio is

$$r_2 = \frac{[88M_1M_2 + 4B]S_1S_2}{102 + (50+4B)S_1S_2} \tag{11}$$

2-D data is very common in most vision problems from gray-level surveillance video to mono-chrome aerial imagery. For instance, our problem may involve finding a 64 × 64 target pattern in 3 different hierarchical resolutions (e.g. 64 × 64, 32 × 32, 16 × 16) using a 16-bins histogram. Our method hunts for the pattern $2,435$ times faster. In Fig. 2-2nd row, we show the comparison results, which can speed up the process up to 6 × 10^4 times with respect to the prementioned conventional approach.

4.4. Color Images

For a color image with a 3D histogram (assuming each point has 3 color values in a tensor form), the parameters become $d = 2$ and $k = 3$. Assuming we are searching for a template window size up to S_1, S_2 in image dimensions the ratio is

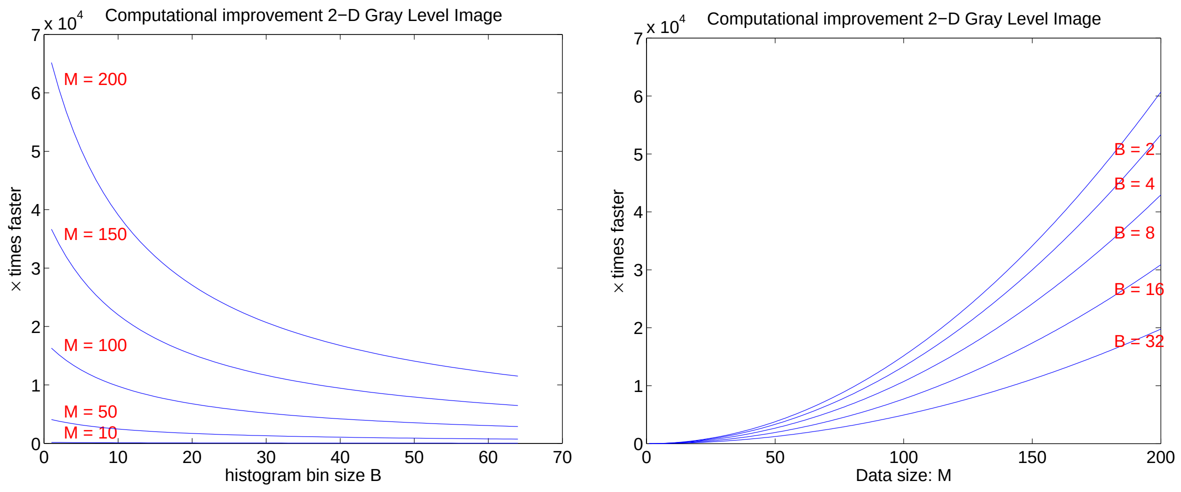
$$r_3 = \frac{[240M_1M_2 + 4B^3]S_1S_2}{300 + (81+4B^3)S_1S_2} \tag{12}$$

In Fig. 2-3rd row, we present the computational savings for a color image search. Even for a regular model matching task that searches a 100 × 100 object model in 20 scales using histograms for each color channel coded in 4-bits (16-bins), the process is accelerated 146 times. As shown in the graphs, the savings can go up to 7 × 10^5 depending on the number of bins and target size.

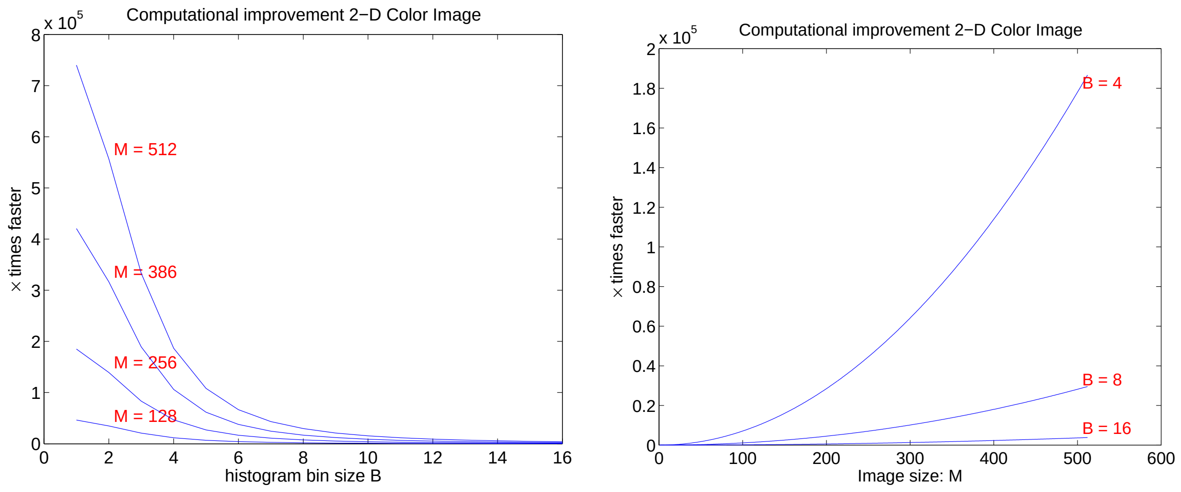
5. OBJECT DETECTION RESULTS

Figure 3 shows detection of given patterns using histogram features. In the image pattern search example, we search for the target object, i.e. ball and balloon, using a 2^{15}-bins color histogram and 320×240 color images. Although the conventional approach and integral histogram give the very same similarity map, the integral histogram method runs in 63msecs, however, the conventional approach requires 2 minutes on a 3.2Ghz P4.

In these examples, pixel color value itself may not be sufficient to find the target object. For instance, in the balloon example, balloon contains same colors of sky and mountains, thus a simple color filtering may be erroneous. Similar analogy is valid for the ball example. It is straightforward to see how histograms improve detection performance in comparison to using only pixel color values or sum of color values.



Computational saving for gray-level image



Computational saving for color image

Figure 2. Achieved computational reduction in comparison to conventional method. The integral histogram method is *as many times as* faster than the existing approach.

6. DISCUSSION

We present a novel and computationally very fast method to compute the histograms of all possible regions in an image. The integral histogram provides not a sub-optimal or partial, but a optimum and complete solution for histogram based search problems.

Our intensive simulations prove that the integral histogram method can expedite the search process more than thousands of times in comparison to the existing conventional approaches. In addition, it enables construction of advanced histogram features for further feature selection and classification purposes.

REFERENCES

1. D. A. Forsyth and J. Ponce. “Computer Vision: A Modern Approach”, *Prentice Hall*, 2002.
2. C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection, *In Proceedings of ICCV*, 1998.
3. S. Ruiz-Correa, L. G. Shapiro, and M. Meila, “A new paradigm for recognizing 3-D object shapes from range data”, *In Proceedings of CVPR*, 2003.

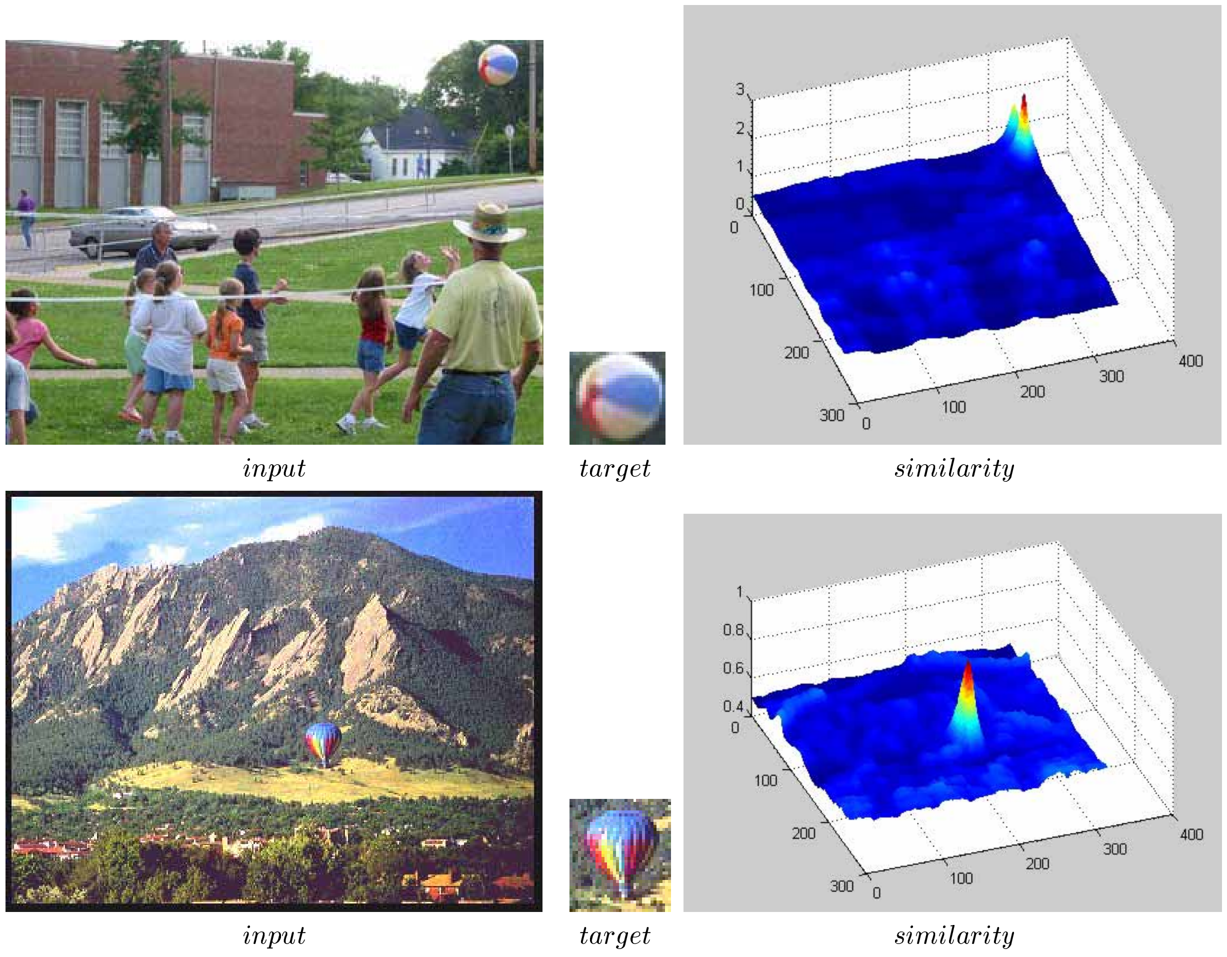


Figure 3. Object detection using a 2^{15}-bins color histogram. The computed similarity map is same as the conventional approach; however the integral histogram method runs in 63msecs although the conventional exhaustive search takes approximately 2 minutes for 100 scales on a 3.2Ghz P4.

4. P. Viola and M. Jones, “Robust real-time face detection”, *In Proceedings of ICCV*, page II: 747, 2001.
5. D. Comaniciu, V. Ramesh, and P. Meer, “Real-time tracking of non-rigid objects using mean shift”, *In Proceedings of CVPR*, 2000.
6. Steve Oualline, “Practical C++ programming”, *O’Reilly & Associates*, ISBN: 1-56592-139-9, 1995.
7. J. Mathew, P. Coddington and K. Hawick, “Analysis and development of java grande benchmarks”, *In Proceedings of ACM*, 1999.
8. C. Carson, M. Thomas, S. Belongie, J.M. Hellerstein, and J. Malik, “Blobworld: A system for region-based image indexing and retrieval”, *In Proceedings of ICVS*, 1999.
9. J. Huang, S. Kumar, M. Mitra, W.J. Zhu, and R. Zabih, “Image indexing using color correlograms”, *In Proceedings of CVPR*, 1997.
10. R. Bryant and D. O’Hallaron, “Computer systems: a programmer’s perspective”, *Prentice Hall*, ISBN 0-13-034074-1, 2003.
11. Y. Moon, F. Luk, H.C. Ho, T.Y. Tang, K. Chan, C. Leung, “Fixed-point arithmetic for mobile devices; a fingerprint verification case study”, *In Proceedings of SPIE*, 2002.