Accelerating Integral Histograms Using an Adaptive Approach

Thomas Müller, Claus Lenz, Simon Barner, and Alois Knoll

Technische Universität München,
Dept. of Informatics VI, Robotics and Embedded Systems,
Boltzmannstr. 3, DE-85748 Garching, Germany
{muelleth,lenz,barner,knoll}@cs.tum.edu

Abstract. Many approaches in computer vision require multiple retrievals of histograms for rectangular patches of an input image. In 2005 an algorithm to accelerate these retrievals was presented. The data structure utilized is called *Integral Histogram*, which was based on the well known *Integral Image*.

In this paper we propose a novel approximating method to obtain these integral histograms that outperforms the original algorithm and reduces computational cost to more than a tenth. Alongside we will show that our adaptive approach still provides reasonable accuracy – which allows dramatic performance improvements for real-time applications while still being well suited for numerous computer vision tasks.

Keywords: Computer Vision, Object Recognition, Tracking, Early Processing, Integral Histogram, Adaptive Approximation.

1 Introduction

In statistics one certainly comes along histogram computation, as a histogram is a statistical description of a set of observations. Each observation is categorized and the number of observations matching each category are summed up. If the counters on the categories are each divided by the total number of observations, the total sum of all values is 1.0 and we call the histogram *normalized*. A normalized histogram can be seen as an approximation of the probability density function from the observation data given.

If more than one aspect defines a category, observations have to be examined regarding any of the relevant aspects and thus the resulting histogram has multiple dimensions. Normalized histograms with multiple dimensions are sometimes also called *Joint Histograms* [1].

Besides visualization of statistics, one of the most relevant applications for histograms is within computer vision (CV). Here a histogram is a very important tool that can be used for various tasks like image understanding and tracking (e.g. applying *Particle Filters* [2]). For the reason of further analysis numerous different types of histograms can be computed on image data, for example color histograms [3], gradient histograms [4], color cooccurrence histograms [5], local

A. Elmoataz et al. (Eds.): ICISP 2008, LNCS 5099, pp. 209–217, 2008.

feature histograms [6,7] or histograms of coefficients from wavelet transforms on localized object parts [8].

In this paper we confine ourselves to the computation of color histograms, but in fact the approach presented here can be extended to any other type of histogram. When computing a color histogram the observations mentioned above refer to the pixels of the image and the categories are determined by the gray values (one-dimensional histogram) or color values (multidimensional histogram). In order to obtain an image histogram typically the number of possible values for a color is limited, for example, a three-channel color value with 8-bit color-depth (256 different values per channel) – like standard RGB color space – could be quantized into $N = 4$ bins per channel. The histogram computed for such an input image then has three dimensions, which results in a total number of $N^3 = 64$ categories.

The approach presented here proposes a new method for retrieval of *Integral Histograms* (see Section 2). Our approach is considerably faster compared to the original algorithm, because it approximates the integral histogram using an adaptive stop criterion (see Section 3). In this way, we overcome the problem of expensive initialization, a mayor drawback of the original algorithm, while still exploiting the benefit of extremely fast retrieval times (see Section 4).

2 The Integral Histogram

For sophisticated tasks in computer vision often histogram retrievals of rectangular regions are necessary [9]. A model-based object recognition approach could for example define a set of spatially connected histograms as a model describing the object to search. Within the recognition algorithm a magnitude of a view dozens to several thousand hypothesis (considering different rotations and scaling) have to be evaluated in order to detect an object with high accuracy. Each of the evaluations requires the retrieval of histogram(s) of a rectangular sub-image. Porikli [10] recently presented a method to speed up these retrievals dramatically. He introduced the concept of *Integral Histograms*, which is based on the *Integral Image* data structure earlier described by Viola and Jones [11].

Considering a linear sequence of observations, an integral histogram in principle specifies a bundle of histograms each summarizing observations from the beginning to a certain observation. The algorithm for one-dimensional data thus can be defined recursively by:

$$H(x^i, b) = H(x^{i-1}, b) \cup Q\left(f(x^i)\right) \tag{1}$$

For image data, which is a two-dimensional data-plane rather than an observation sequence, the computation of subsequent histograms can be specified as shown below, according to [10]:

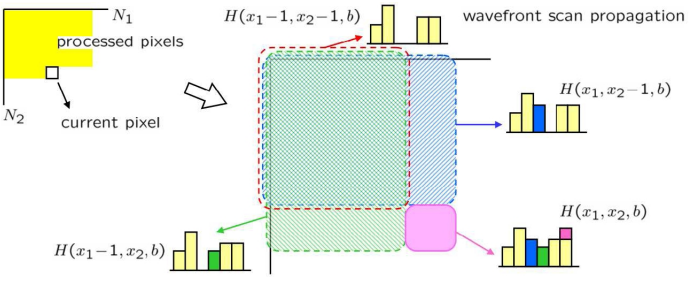

Fig. 1. Propagation of integral histogram by wavefront scan[1]

$$\begin{aligned} \forall b : H(x_1, x_2, b) = \quad & H(x_1 - 1, x_2, b) \\ & + H(x_1, x_2 - 1, b) \\ & - H(x_1 - 1, x_2 - 1, b) \\ & + Q(f(x_1, x_2)) \end{aligned} \tag{2}$$

In the above equations b denotes a bin (or category) within the histogram and $Q(f(\boldsymbol{x}))$ an empty histogram except for one entry that refers to the categorized value of the pixel at position $\boldsymbol{x}$. Figure 1 illustrates the flow of the algorithm.

The big advantage of the integral histogram approach is the dramatic improvement on the cost of retrieval of histograms for rectangular regions. After the first step, the computation of the integral histogram, no further image access is needed. A histogram for an arbitrary sub-image with left-upper coordinate $(x_1|x_2)$ and right-lower coordinate $(x_3|x_4)$, relevant for the evaluation of a hypothesis, can be retrieved in constant time independent from the patch size. The scheme below describes the retrieval operation from the integral data structure:

$$\begin{aligned} \forall b : H(x_1, x_2, x_3, x_4, b) = \quad & H(x_3, x_4, b) \\ & - H(x_3 - x_1, x_4, b) \\ & - H(x_3, x_4 - x_2, b) \\ & + H(x_3 - x_1, x_4 - x_2, b) \end{aligned} \tag{3}$$

As one can see from (3), only one addition and two subtractions (in addition to array-value retrievals) are needed per bin b in order to create the new histogram. In contrast, the naive implementation needs $(x_3 - x_1) \cdot (x_4 - x_2)$ pixel queries and extra operations for computing the corresponding bin and summing up.

Obviously a big deficit in systems utilizing exhaustive histogram retrieval is, that the cost for the intersection is dependent on the size of the patch and grows proportionally to the area. Now, with the use of an integral histogram, the cost for the retrieval remains constant for arbitrary sub-images.

In a nutshell, integral histograms have great computational advantages compared to a naive approach in scenarios, where a multitude of histograms for

[1] Taken from [10]. Copyright © Mitsubishi Electric Research Laboratories, Inc., 2005

image-patches have to be retrieved, because the retrieval operation is very efficient. But still there is one big drawback which is addressed in this paper: the first step, the computation of the integral histogram, is in deed rather time and memory consuming.

3 Advanced Integral Histogram Computation

In Porikli's original work on integral histograms two methods for propagation were proposed: the wavefront scan (see Figure 1) and a string scan method. Both methods and various variations require the analysis of every single pixel of the input image, so the larger images get, the more effort is needed for computation of the integral histogram (proportional to the area).

This exact solution is not optimal by means of performance considering natural images, as in such images often uni-colored areas occur. We exploit this finding and introduce a novel method based on iterative approximation applying an adaptive stop criterion.

3.1 Adaptive Refinement

Within the iterative algorithm the input image I is equally divided into four rectangles and with every subsequent level these rectangles are divided the same way. We call this *refinement*. Intuitively recursive refinement stops at a maximum depth of

$$d_{max} = \lfloor log_2 \left(\text{argmin}_I \left(width_I, height_I \right) \right) \rfloor . \tag{4}$$

A more sophisticated approach considering the presence of uni-colored areas defines an adaptive abort criterion. Here refinement stops, when either d_{max} is reached or the criterion is matched. The criterion $c(\boldsymbol{p})$ comparing the n binned values $b(p)$ of a set of pixels $\boldsymbol{p} = \{p_1, p_2, \ldots, p_n\}$ can be defined as follows:

$$c(\boldsymbol{p}) = \begin{cases} true & \text{if } \forall i, j \leq n : b(p_i) = b(p_j) \\ false & \text{otherwise} \end{cases} \tag{5}$$

From (5) one can deduce that refinement is aborted, if the values of all pixels used for determination (random or equally distributed pixels inside the rectangle) are categorized into the same bin. Thus uni-colored areas are not refined any further (Figure 2).

3.2 Histogram Retrieval

As we always refine by dividing a rectangle into four sub-rectangles successively, a *Quad-Tree* seems to be a good choice for data-maintenance, because we do not loose information about the accuracy level of an approximation. Also spatial information for the histogram origins are stored efficiently. Thus from a leaf position inside the quad-tree we are able to reconstruct the position of the corresponding pixel in the original image and vice versa. Moreover, downwards from

Fig. 2. Using an adaptive stop criterion preferentially refines where strong gradients occur

the root-node we can access any histogram at a maximum of d_{max} steps, which is the maximum depth of the tree.

In order to obtain a histogram stored in a node of the quad-tree, we define a method for retrieval of a node corresponding to a position p as follows:

```
function getNode(Position p, Node root) : Node
  Node n := root
  int d  := 0
  do
    if p = n.pos or n.noChildren then break
    else
      Node lu := n.child(left, upper)
      HVal h := p.x <= lu.pos.x ? left  : right
      VVal v := p.y <= lu.pos.y ? upper : lower
      n := n.child(h, v)
    fi
    d := d + 1
  while d < dmax
  return n
```

One can see from the above listing, that this method always delivers the node containing the best approximation for a position if the exact position is not represented in the tree.

3.3 Recursive Approximation

On the first level of the recursive approximation, each of the histograms the integral histogram consists of is initialized with the bin value of the lower-rightmost pixel of the input image.

Within the next iterations, we recursively compute better approximations by refining and successive updating steps. Supposing we refined to the full extent of d_{max} without adaptive abortion, the recursive update can be define according to Equation (2). But considering the use of an abortion criterion, we first have to determine the best approximation for the four sub-rectangular histograms we need for the update according to the method described in Section 3.2, before we can actually update. The positions corresponding to those are determined relative to the position of the parent histogram p^d and according to the level (the depth) d we process. For an image I we can calculate these positions ($l$eft-$u$pper to $r$ight-$l$ower) as follows:

$$\begin{aligned} p_{l,u}^{d+1} &= \left(p^d.x - 2^{-d} width_I, \quad p^d.y - 2^{-d} height_I\right) \\ p_{l,l}^{d+1} &= \left(p^d.x - 2^{-d} width_I, \quad p^d.y\right) \\ p_{r,u}^{d+1} &= \left(p^d.x, \quad p^d.y - 2^{-d} height_I\right) \\ p_{r,l}^{d+1} &= p^d \end{aligned} \tag{6}$$

Furthermore the histograms used for the update have to be weighted, again considering depth d and their position p:

$$w_I^d(p) = \frac{2^{2d} \cdot p.x \cdot p.y}{width_I \cdot height_I} \tag{7}$$

Utilizing Equations (6) and (7) and having retrieved the best approximating histograms $H'(p)$ as described in the last section, the update method for a bin b can be formalized as follows:

$$\begin{aligned} H(p^d, b, d) = \quad & Q\left(f(p_{r,l}^{d+1})\right) \\ & - w(p_{l,u}^{d+1}) \cdot H'\left(p_{l,u}^{d+1}, b\right) \\ & + w(p_{l,l}^{d+1}) \cdot H'\left(p_{l,l}^{d+1}, b\right) \\ & + w(p_{r,u}^{d+1}) \cdot H'\left(p_{r,u}^{d+1}, b\right) \end{aligned} \tag{8}$$

Note that since $Q(f(p_{r,l}^{d+1})) = Q(f(p^d))$ refers to a single entry, its weight has to be set to 1.0 accordingly. As a final step, to keep all contributions in balance for the updates in $d-1$, the resulting histogram has to be normalized according to the weights from (8):

$$\bar{H}\left(p^d, b, d\right) = \frac{H\left(p^d, b, d\right)}{1 - w(p_{l,u}^{d+1}) + w(p_{l,l}^{d+1}) + w(p_{r,u}^{d+1})} \tag{9}$$

4 Experimental Results and Conclusion

Figure 3 depicts mean retrieval times for a single histogram. In case of the integral histograms the values shown are computed with respect to the relatively

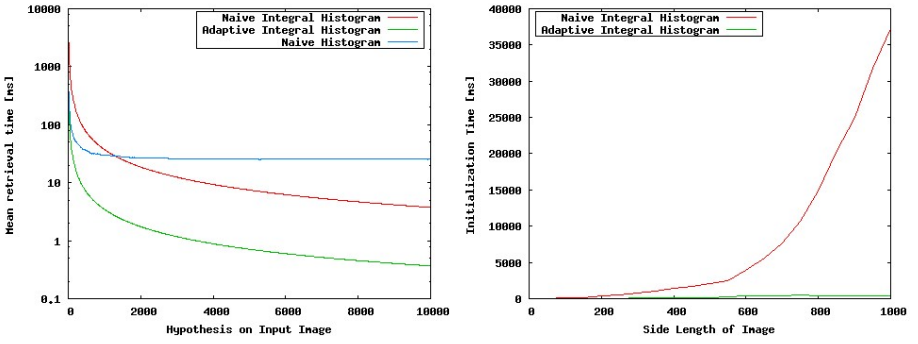

Fig. 3. Left: Mean retrieval times (note the logarithmic scale), Right: Initialization times subject to side length of the input image

expensive initialization of the integral histogram (right graph). Note that the left graph is drawn in logarithmic scale, which clarifies the enormous speed-up applying the presented algorithm. As one can see, the more hypothesis are evaluated, the less the initialization plays a role.

An important finding is, that applying the adaptive approach results in a computational benefit with a mere magnitude of about 50 hypothesis on a $\sim$2.3 M pixel input image. The naive integral approach [10] reaches this break even point with above 1800 hypothesis. Mean values for retrieval of a *single* random histogram in an integral histogram structure fluctuate between 0.01 to 0.03 ms, while retrieval using a traditional approach is highly dependent on the size of the patch and takes a mean of 24.9 ms in our evaluation scenario.

Utilizing our approach the problem of expensive creation of the integral histogram can be overcome. In fact mostly initialization can be executed in around 150 ms and for smaller images $\sim 250^2$ px, considering 8-bit color depth and 8 bins per channel, the approach easily reaches the real-time limit of 25 Hz. The presented algorithm on the one hand benefits from very short retrieval times

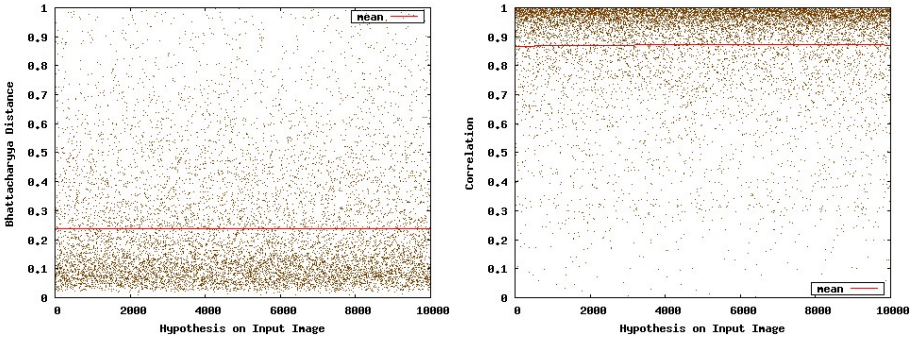

Fig. 4. Accuracy using the adaptive integral histogram approach compared to exact solutions (left: Bhattacharyya Distance, right: Correlation) on random patches

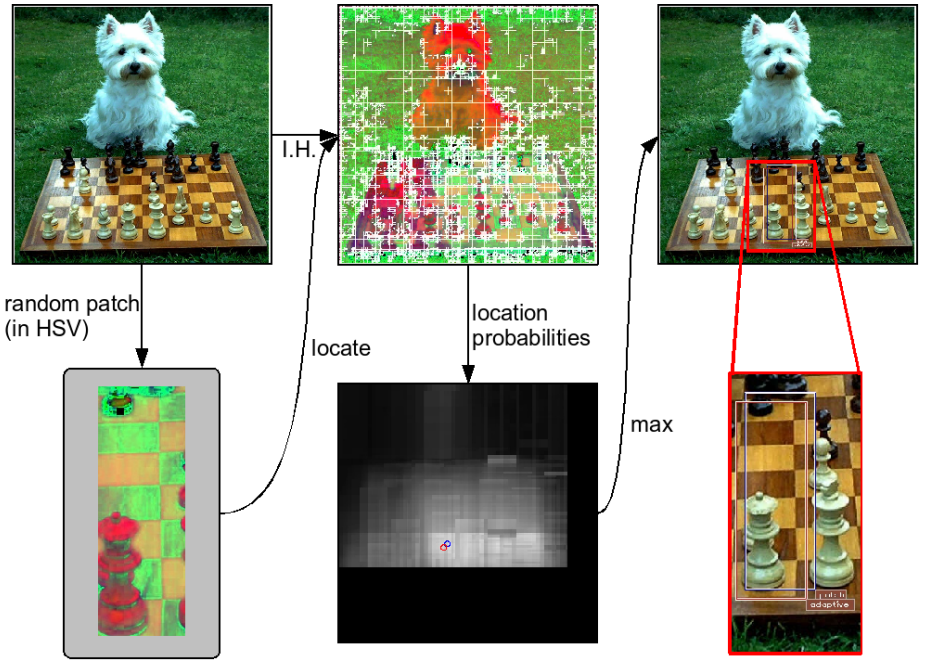

Fig. 5. Demo application with exhaustive search of a random patch

provided by the integral histogram data structure and on the other reduces the cost of structure initialization dramatically (at 2.3 M pixel by 26.27) compared to [10]. We can thus reach a speed-up proportional to the number of hypothesis compared to a conventional approach. This speed-up grew up to ∼250 when retrieving 10^4 random histograms in our experiments.

A deficit of approximating solutions in real world applications always is the cost of achieving a desired accuracy. Concerning this topic, Figure 4 shows two common measures, the Bhattacharyya distance [12] and the correlation. Values in the figure have been compared to the exact solution obtained with the conventional approach. It becomes clear, that the solutions are well suited for real-world applications like particle filters. Moreover, due to extremely fast retrieval times, one can easily compensate for occasional outliers by simply evaluating a few more histograms.

Figure 5 shows a possible application of our approach. In this test, the adaptive integral histogram approach was used to locate a random patch in an image. The demo application uses exhaustive search and so evaluates every possible region (considering size) of the original image in order to detect the patch. The figure also shows a map of saliency - brighter areas refer to higher , darker regions to lower similarity. On the left, the extracted integral histogram using hue and saturation of the HSV color space is shown.

In a nutshell the proposed approach already outperforms the conventional approach with ∼50 hypothesis to evaluate on an input image, while the original algorithm [10] does not until ∼1800 on natural color images with 8 bit color-depth – 30 times faster. Yet there is some potential for improvements. Accuracy can be tuned through adjustments on the adaptive stop criterion and the number of bins. Performance on the other hand can be optimized by limiting d_{max} to a lower depth, resulting in further acceleration up to a factor 10 in experiments – a factor 300 (!) in total. Finally, due to the tree data structure, the algorithm can easily be parallelized in order to gain even more on the performance side.

Acknowledgement

This work was partly supported by the DFG excellence initiative research cluster *Cognition for Technical Systems (CoTeSys)* and the EU integrating project of Framework Program 6, *Joint-Action Science and Technology (JAST)* (FP6-003747-IP).

References

1. Pass, G., Zabih, R.: Comparing images using joint histograms. Multimedia Systems 7(3) (1999)
2. Arulampalam, S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing 50(2), 174–188 (2002)
3. Swain, M.J., Ballard, D.H.: Color indexing. International Journal of Computer Vision 7(1), 11–32 (1991)
4. Schiele, B., Crowley, J.L.: Recognition without correspondence using multidimensional receptive field histograms. International Journal of Computer Vision 36(1), 31–50 (2000)
5. Chang, P., Krumm, J.: Object recognition with color cooccurrence histogram. In: IEEE CVPR (1999)
6. Hetzel, G., Leibe, B., Levi, P., Schiele, B.: 3d object recognition from range images using local feature histograms. In: IEEE CVPR, vol. 2 II, pp. 394–399 (2001)
7. Laptev, I.: Improvements of object detection using boosted histograms. In: BMVC 2006, vol. III, p. 949 (2006)
8. Schneiderman, H., Kanade, T.: A statistical model for 3d object detection applied to faces and cars. In: IEEE CVPR (2000)
9. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: IEEE CVPR, vol. I, pp. 798–805 (2006)
10. Porikli, F.M.: Integral histogram: A fast way to extract histograms in cartesian spaces. In: IEEE CVPR, vol. I, pp. 829–836 (2005)
11. Viola, P., Jones, M.: Robust real-time object detection. In: IEEE ICCV Workshop on Statistical and Computational Theories of Vision (2001)
12. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bull. Calcutta Math. Soc. 35, 99–109 (1943)