Point, Harris Corner, and SIFT Point Coordinate Encoding Algorithms

Jian-Jiun Ding [a*], Szu-Wei Fu [b], and Pin-Xuan Lee [c]
Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan [a, b, c]

***Abstract* —In this paper, we discuss a special data compression problem, i.e., how to compress the coordinates of points in an image. For the applications of template matching, object tracing, and object-oriented image processing, it is required to record the corners and the scale-invariant feature transform (SIFT) points of an object. The simplest way to encode these points is to record their coordinates directly. However, this method is inefficient. In this paper, we try a variety of methods to encode point coordinates. We find that the algorithm which adopts the techniques of shortest distance re-ordering, maximal axis distance, adaptive arithmetic coding, and dense-based context generation has the best performance to encode Harris' corners and SIFT points, the proposed algorithm is also useful for encoding other sparsely distributed two dimensional data.** [1]

I. Introduction

Corners [1-4] and SIFT points [5-8] are significant features in images and can be used for object description, template matching, pattern recognition, object tracing, and other object-oriented image processing applications.

Since corners or SIFT points are invariant to scaling and using corners or SIFT points to represent an object is much more efficient than using the pixels or the edges of the object, many advanced image processing techniques are based on corners and SIFT points. For example, for the application of object tracing, instead of tracing all of the pixels or the boundary of the object, it will be much more efficient to trace the corners or the SIFT points of the object.

In this paper, we discuss the problem of how to encode the corners and the SIFT points efficiently. Although one can use 1 and 0 to denote whether a pixel is a corner or encode the coordinates of corners directly, these two ways are not efficient enough. In this paper, we propose an algorithm which is the combination of point distance encoding, adaptive arithmetic coding, dense-based context, and distance-based re-ordering. With the proposed algorithm, the corners and SIFT points of images can be recorded in a very efficient way. Especially, when the order of corners on the boundary contour is considered, the proposed algorithm is much more efficient than other methods.

In Section 2, we describe three direct ways to encode corners and SIFT points: the binary method, the coordinate method, and the adaptive arithmetic coding method. In Sections 3 and 4, the proposed method is introduced. In Section 5, several simulations are performed to compare the proposed method and other methods for encoding Harris' corners and SIFT points. A conclusion is made in Section 6.

[1] This work was supported in part by the NSC under Grant No. 102-2221-E-002-054-MY2.

Corresponding author:djj@cc.ee.ntu.edu.tw
DOI:10.6159/IJSE.2014.(4-2).61

II. Direct and Arithmetic Coding Methods for Point Coordinates

There are several possible ways to encode the corners or the SIFT points. For example, one can change an image into the binary form and use '0' and '1' to denote the non-corner (or non-SIFT) pixels and corner (or SIFT) pixels, respectively. We call it the *binary* method. Suppose that the size of the image is $M\times N$. Then the data size required for the *binary* method is

$$MN \tag{1}$$

bits. However, this method is obviously not efficient when the corner (or SIFT) pixels are sparsely distributed.

Another method is to encode the coordinates of corners or SIFT points. We call it the *coordinate* method. If the image size is $M\times N$, first, we use $\log_2(MN)$ bits to encode how many corners or SIFT points in the image. Then, we use $\log_2(MN)$ bits to encode the coordinate of each corner or SIFT point. Therefore, if the number of corners or SIFT points is L, then the number of bits is

$$(L+1)\log_2(MN). \tag{2}$$

Compare with the binary method, one can see that if

$$L+1 < MN/\log_2(MN), \tag{3}$$

which is always the case, then the coordinate method is more efficient than the binary method. However, its efficiency is not enough, especially for the image that has many corners and SIFT points.

Adaptive arithmetic coding [9-11] is an entropy coding method that is more efficient than Huffman coding and static arithmetic coding. The flowchart of adaptive arithmetic coding for encoding corners and SIFT points is shown as in Fig. 1. We call it the *adaptive arithmetic coding* method. First, as the binary method, we use '0' and '1' to denote whether a point is a corner (or a SIFT point). Then, we use the procedure in Fig. 1 to modify the values $T(0)$ and $T(1)$ of the frequency table and the range $[L, H]$ iteratively. When applying this method, the number of bits is a little higher than

$$MN\cdot E(x)/\log 2 \tag{4}$$

where $E(x)$ is the entropy of x. Since corners and SIFT points always sparsely distributed, $E(x)$ is usually very small, the adaptive arithmetic coding method usually requires much less number of bits than the binary method and the coordinate method for corner and SIFT point encoding.

However, we think that the adaptive arithmetic coding method is not efficient enough. In Section 3, we propose another algorithm to further reduce the number of bits required for encoding the corners or SIFT points of an image.

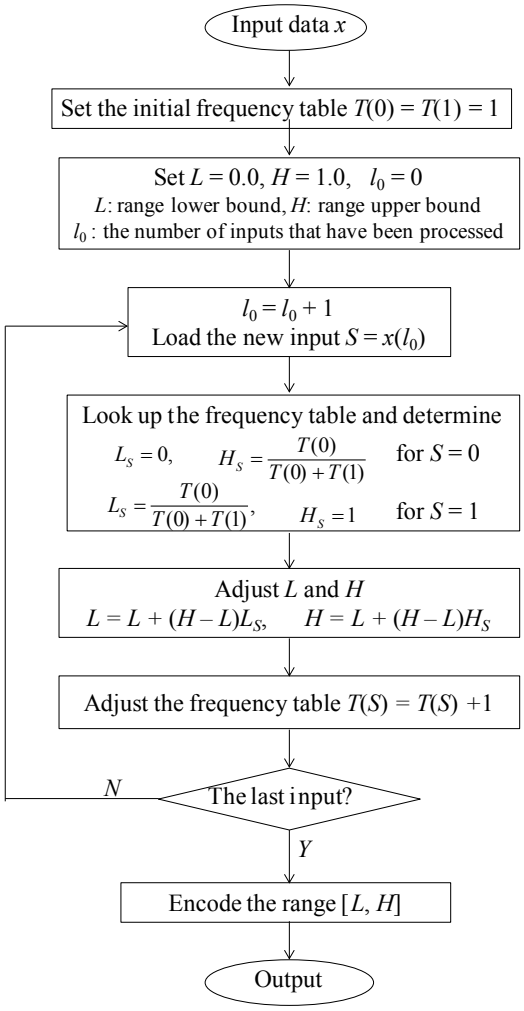

Fig. 1 Flowchart of the adaptive arithmetic coding method for encoding corners or SIFT points.

III. Proposed Algorithm for Corner or SIFT Point Coordinate Encoding

In fact, the distributions of corners and SIFT points are always not uniform. For example, for Barbara image in Fig. 2, one can see that most corners of Barbara image concentrate on some certain regions (the stripes of clothes, the table covers, and the boundary of each region). Therefore, we suggest that it is better to use *different probability models for different regions* of an image. As the method in Fig. 1, our algorithm is also applies adaptive arithmetic coding. However, there are several differences

(1) Instead of encoding whether a point is a corner, we encode the coordinate difference between two corners or two SIFT points. However, the coordinate difference used here is neither of the Euclidian distance form nor of the Manhattan distance form. We use the following way to define the coordinate difference:

$$d_{i-1,i} = \max\left(\left|x_i - x_{i-1}\right|, \left|y_i - y_{i-1}\right|\right) \quad (5)$$

where $(x_{i\text{-}1}, y_{i\text{-}1})$ and (x_i, y_i) are the coordinates of the $(i-1)^{\text{th}}$ and the i^{th} corners (or SIFT points), respectively.

Fig. 2 Harris points of Barbara image (marked by white dots)

We call (5) the *maximal axis distance*. The advantages of the maximal axis distance are: (i) It is always an integer value, which is more suitable for arithmetic coding. (ii) It is easy to compute. (iii) If the image size is $M \times N$ and the number of pixels whose distance to the point (x_i, y_i) is d is denoted by C_d, then, when using the Manhattan distance, C_d may decrease with d if $d > \max[\max(M\text{-}x_i, x_i\text{-}1), \max(N\text{-}y_i, y_i\text{-}1)]$. When using the maximal axis distance, C_d increases with d in most cases.

(2) Before encoding the coordinate difference, the *shortest distance re-ordering* is applied. Instead of sorting according to x-coordinates and y-coordinates, to reduce the distance between adjacent corners, we apply the following algorithm to sort corners (or SIFT points):
   (i) First, find the corner (or SIFT point) whose "maximal axis distance" to (0, 0) is minimal. Denote its coordinate by (x_1, y_1). Set $i = 1$.
   (ii) Then, find the corner (or SIFT point) that has not been numbered whose "maximal axis distance" to (x_i, y_i) is minimal. Denote it by (x_{i+1}, y_{i+1}) and set $i = i+1$.
   (iii) Repeat (ii) until all of the corners (or SIFT points) in the image have been numbered.

(3) Different probability models are used for a dense region and a sparse region. That is, we also use the adaptive arithmetic coding, but the context is generated by the previous "maximal axis distance". We call it the *dense-based context*. If

$$d_{i-2,i-1} \le 7, \quad (6)$$

then we conclude that $(x_{i\text{-}1}, y_{i\text{-}1})$ is in a region where the corners (or SIFT points) is densely distributed. In this case, we set the context to 1 and use the probability model corresponding to context 1 to encode (x_i, y_i). By contrast, if $d_{i-2,i-1} > 7$, then we conclude that $(x_{i\text{-}1}, y_{i\text{-}1})$ is in a region where the corners (or SIFT points) is sparsely distributed and the probability model corresponding to context 2 is used to encode (x_i, y_i).

(4) The proposed algorithm also uses the adaptive arithmetic coding. However, the coding process is quite different from that in Fig. 1. In addition to that the context is adopted, the frequency table is assigned according to the maximal axis distance between $(x_{i\text{-}1}, y_{i\text{-}1})$ and (x_i, y_i).

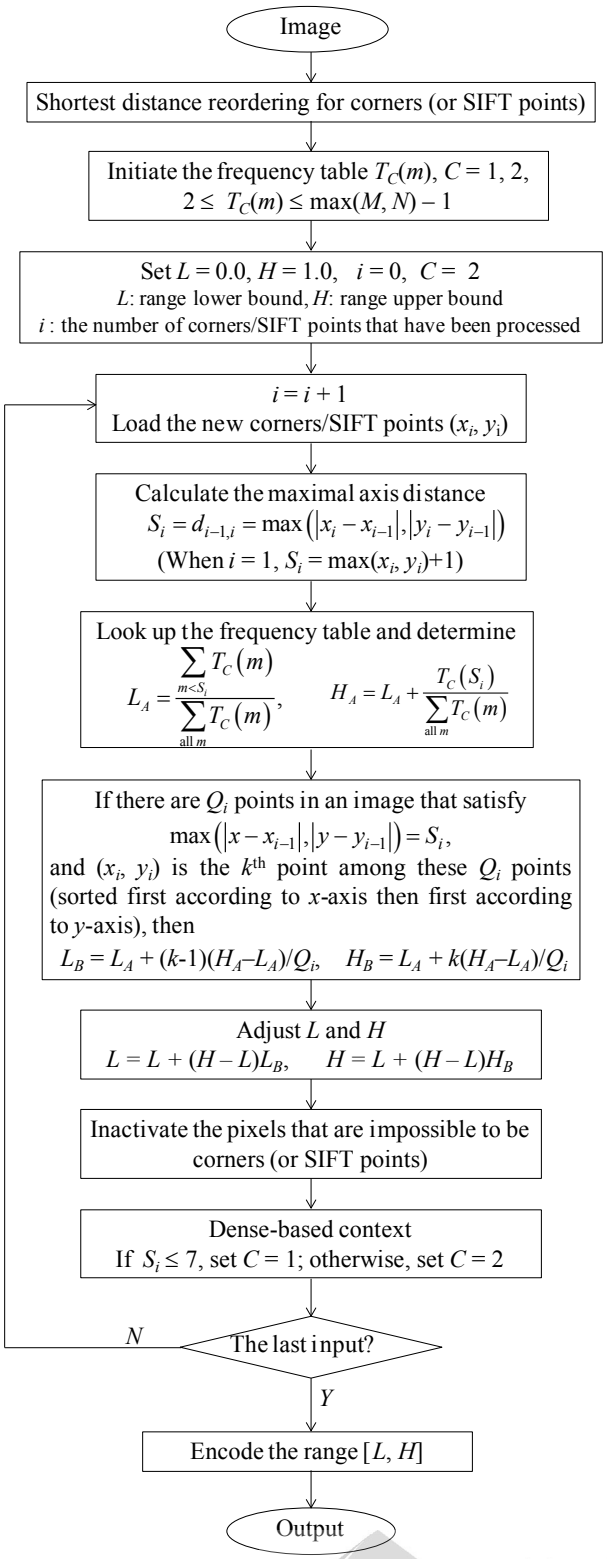

Fig. 3 Flowchart of the proposed method for encoding corners or SIFT points.

Furthermore, in addition to the maximal axis distance, the explicit values of x_i and y_i should also be encoded.

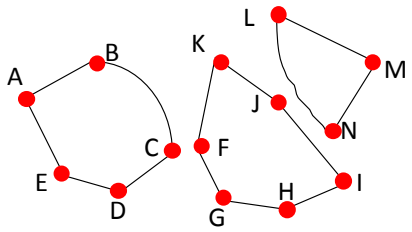

Fig. 4 For the applications of boundary description, the orders of corners on boundary curves should still be considered.

According to the techniques described above, the flowchart of the proposed algorithm is shown as in Fig. 3.

There are some details in Fig. 3 that should be noticed. First, the parameter C, which is the context of adaptive arithmetic coding, is initiated as 2 because due to the shortest distance re-ordering algorithm, the first corner (or SIFT point) of an image will be near to (1, 1) and the corners (or SIFT points) always sparsely distributed in this region. Second, when $i = 1$, we set S to $\max(x_i, y_i) + 1$ instead of $\max(x_i, y_i)$ because we avoid the value of S being less than 2. Moreover, there is a step of "inactivate that are impossible to be corners (or SIFT points) in Fig. 3. The goal of this step is to improve the coding efficiency. Since *shortest distance reordering* is applied in our algorithm, if the maximal axis distance between the $(i\text{-}1)^{\text{th}}$ corner (x_{i-1}, y_{i-1}) and the i^{th} corner (x_i, y_i) is $d_{i-1,i}$, then it means that the pixel satisfying (7) or (8) is impossible to be the j^{th} corner (SIFT point) where $j > i$:

$$\max(|x - x_{i-1}|, |y - y_{i-1}|) < d_{i-1,i} \quad (7)$$

or

$$\max(|x - x_{i-1}|, |y - y_{i-1}|) = d_{i-1,i}, \quad x < x_i \quad \text{or} \quad x = x_i, \quad y < y_i. \quad (8)$$

Therefore, the pixels satisfying (7) or (8) can be inactivated. That is, in the future iteration, when determining Q_j (the number of points that satisfy $\max(|x - x_{j-1}|, |y - y_{j-1}|) = S_j$) where $j > i$, the points that satisfy (7) or (8) can be ignored. It will decrease the value of Q_j, widen the range of $[L_B, H_B]$, and hence improve the coding efficiency.

IV. Proposed Algorithm for Corner Encoding When the Orders of Corners on the Boundary Curve is Considered

In some applications, such as boundary description and shape analysis [12], the order of corners on the boundary curve should be considered. As the example in Fig. 4, the orders of corners should be A, B, C, D, E, F, G, H, I, J, K, L, M, and N and the points A, F, and L should be denoted as the starts of curves. To retrieve the original curves, the orders of corners should be valid. For example, if one applies the shortest distance re-ordering algorithm in Section 3 directly, the orders of corners becomes A, B, C, F, G, H, I, N, M, L, K, J, D, and E and the original curves cannot be reconstructed. Therefore, for the applications such as boundary description and shape analysis, one should modify the re-ordering algorithm in Section 3 to order and encode the corners of images. The modified algorithm is as follows.

TABLE I
COMPARISON OF CODING EFFICIENCY FOR HARRIS CORNERS. THE TESTED DATA CONSISTS OF 49 512×512 IMAGES

Compression Methods	Binary method	Coordinate method	Adaptive arithmetic coding	Proposed method
Sum of the Numbers of Bits	12845056	2263644	1001699	**959672**

TABLE II
COMPARISON OF CODING EFFICIENCY FOR SIFT POINTS. THE TESTED DATA CONSISTS OF 49 512×512 IMAGES

Compression Methods	Binary method	Coordinate method	Adaptive arithmetic coding	Proposed method
Sum of the Numbers of Bits	12845056	1789362	829419	**812216**

(i) Find the corner whose "maximal axis distance" to (0, 0) is minimal and denote its coordinate by (x_1, y_1). Set $i = 1$.
(ii) Check which curve the corner (x_i, y_i) belongs to.
(iii) Then, sort the corners on this curve in the clockwise order or in the counterclockwise order, dependent on using which method the sum of the maximal axis distances is smaller. If there are k corners on the curve, denote the coordinates of these corners by (x_i, y_i), (x_{i+1}, y_{i+1}), (x_{i+2}, y_{i+2}), …, and (x_{i+k-1}, y_{i+k-1}).
(iv) Then, find the corner that has not been numbered whose "maximal axis distance" to (x_{i+k-1}, y_{i+k-1}) is minimal. Denote it by (x_{i+k}, y_{i+k}) and set $i = i+k$.
(v) Repeat (ii)-(iv) until all of the corners in the image have been numbered.

V. SIMULATIONS

In this section, several simulations are performed to compare the performances of the binary method, the coordinate method, the adaptive arithmetic coding method, and the proposed method for corner and SIFT point compressions.

In Table I, first, Harris' algorithm is applied to find the corners of 49 test images. Each image has a size of 512×512. In sum, there are 125709 corners in these images. Then, a variety of methods are applied to encode them. Simulations show that the proposed method indeed has the best performance to encode the corners of images. When using the proposed method, the average number of bits require for each corner is 959672/125706 = 7.6341. By contrast, if one encodes the coordinates of each corner directly, the number of bits for each corner is $2\times\log_2(512) = 18$.

In Table II, we test the performances of SIFT point compression. As Table I, there are 49 images and each image has a size of 512×512. In sum, there are 99360 SIFT points in these images. The results in Table II again shows that the proposed method has the best performance to encode SIFT points of images.

The simulation in Table III is also about corner compression. However, the orders of corners on the boundary are considered.

TABLE III
COMPARISON OF CODING EFFICIENCY FOR HARRIS CORNERS WHEN THE ORDERS OF CORNERS ON OBJECT BOUNDARIES ARE CONSIDERED. THE TESTED DATA CONSISTS OF 50 SIMPLE IMAGES

Compression Methods	Binary method	Coordinate method	Adaptive arithmetic coding	Proposed method
Sum of Numbers of Bits	8876464	29666	29154	**21197**

We test 50 images and use a variety of algorithms to encode their corners and preserve the orders of corners on the boundary curve. The simulation results in Table III show that the proposed method much outperforms other methods in this case. This may due to that, when using other methods, extra memory is required to encode the order of corners. It is not required for the proposed method.

VI. CONCLUSION

In this paper, the problem of how to compress the corners and the SIFT points of an image efficiently is discussed. We propose the techniques of (i) shortest distance re-ordering, (ii) the maximal axis difference, and (iii) dense-based context generation and find that the algorithm that is a combination of these techniques has the best performance for encoding corners and Harris points. In addition to corners and SIFT points, the proposed algorithm can also be used for encoding the speeded up robust feature (SURF) points [13], the centroids of objects, and any other sparsely and non-uniformly distributed 2-D data in an efficient way.

REFERENCES

[1] C. G. Harris and M. Stephens, "A combined corner and edge detection", *Proc. 4th Alvey Vision Conf.*, pp. 189-192, 1988.
[2] S. C. Pei and J. J. Ding, "Improved Harris' algorithm for corner and edge detections," *ICIP*, San Antonio, USA, vol. 3, pp. 57-60, Sept. 2007.
[3] J. B. Ryu, C. G. Lee, and H. H. Park, "Formula for Harris corner detector," *Electron. Lett.*, vol. 47, pp. 180-181, 2011.
[4] W. B. Zhao and Y. N. Zhang, "Survey on corner detection," *Application Research of Computers*, vol. 10, pp. 17-19, 2006.
[5] D. G. Lowe, "Object recognition from local scale-invariant features." *IEEE International Conference on Computer Vision*, vol. 2, pp. 1150-1157, 1999.
[6] S. Stephen, D. Lowe, and J. Little. "Vision-based mobile robot localization and mapping using scale-invariant features," *IEEE International Conference on Robotics and Automation*, vol. 2, pp. 2051-2058, 2001.
[7] Q. Li, G. Wang, J. Liu, and S. Chen, "Robust scale-invariant feature matching for remote sensing image registration," *IEEE Geoscience and Remote Sensing Letters*, vol. 6, pp. 287-291, 2009.
[8] W. Cheung and G. Hamarneh, "*N*-SIFT: *N*-dimensional scale invariant feature transform," *IEEE Trans. Image Processing*, vol. 18, pp. 2012-2021, 2009.
[9] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, vol. 30, pp. 520-540, 1987.
[10] P. G. Howard and J. S. Vitter, "Arithmetic coding for data compression," *Proceedings of the IEEE*, vol. 82, pp. 857-865, 1994.
[11] P. G. Howard and J. S. Vitter, "Analysis of arithmetic coding for data compression," *Inf. Process. Manage.*, vol. 28, pp. 749-763, Dec. 1992.
[12] D. Zhang and G. Lu. "Review of shape representation and description techniques," *Pattern Recognition*, vol. 37, pp. 1-19, 2004.
[13] J. Luo, and O.g Gwun. "A comparison of SIFT, PCA-SIFT and SURF," *International Journal of Image Processing*, vol. 3, pp. 143-152, 2009.