マルチフェーズハッシュを利用した部品ベースオブジェクト発見手法

ヒブラン フェンテス ピネダ　古賀久志　渡辺俊典
電気通信大学 大学院 情報システム学研究科

あらまし　領域分割された画像から，例を教示することなくオブジェクトを自動発見する手法を提案する．提案手法ではオブジェクトを部品の集合体と想定して，さらに各部品は同色の近隣画素群から構成されていると考え，以下の 4 フェーズを経て画像からオブジェクトを自動的に抽出する．(1) 近くの同色画素同士をクラスタにして部品を決定する．(2) 抽出された部品を属性値によって分類してラベル付けする．(3) 近接する部品群をクラスタにしてオブジェクト候補を抽出する．(4)　複数個出現したオブジェクト候補を意味のあるオブジェクトとみなして抽出する．提案システムでは，上記の 4 ステップをすべてハッシュ関数を用いて実現する．特に，最初の 3 フェーズは LSH(locality-sensitive hashing) のようなユークリッド空間におけるハッシュによって実現され，第 4 フェーズは通常のハッシュによって実現される．このようにオブジェクト発見における基本オペレーションがハッシュ関数のみで実現できる可能性を示した点が提案手法の特徴である．本手法はハッシュ技術しか利用しないため，簡単に実装できる．さらに，第にコンポーネント間の厳密な位置関係を見ないので，第 4 フェーズでは回転、平行移動に対してロバストに同種オブジェクトを発見できる．実験により提案手法の有効性を示す．

Component-Based Automatic Object Discovery Using Multiple Phase Hashing

Gibran Fuentes Pineda, Hisashi Koga, and Toshinori Watanabe
Graduate School of Information Systems
University of Electro-Communications

Abstract　This paper proposes a component-based method to discover objects automatically without examples from segmented images. Our approach deems an object as combination of components, where each component consists of near pixels with the same color. The object discovery is realized in four phases: (1) discovery of components by gathering close pixels with the same color, (2) labeling of components by gathering components with similar attribute values, (3) discovery of object candidates by gathering close components, and (4) determination of valid objects among candidates, such that if the same kind of object candidates appear multiple times, they are regarded as meaningful objects. The primary contribution of this approach is to demonstrate that several essential functions in object discovery can be implemented only by hashing techniques. Especially, the first three phases rely on a hashing on Euclidean space like locality-sensitive hashing. The final fourth phase uses standard hashing technique. Since the algorithm only uses hashing techniques, it is easy to implement. Our system is robust against various parameters (rotation, translation, etc). The experimental results under different scenes and patterns present the validness of the method.

1 Introduction

In order for computers to discover objects, the objects should be modeled by some means so that computers can handle them. For example, an object may be represented as a graph which describes the interrelation between its components and the graph matching is used to detect objects. Instead, feature vectors are often computed from images and the similar objects are identified by the distances between their feature vectors. While this example associates the models in computers with the actual objects explicitly, there are some examples in which the association is rather implicit. In the Bayesian inference method, a statistical hypothesis is constructed from examples given a priori. In the neural network, a computer maintains the knowledge acquired by learning image database. In the object recognition systems, the models are created manually in the sense that the target objects are teached from examples, which seems not likely to be scalable in the future when we are forced to deal with extremely large image databases. Hence, the purpose of our research is the automatic acquisition of models from given images.

1.1 Outline of the paper

In this paper, we present a component-based system to discover objects from segmented images by only using hashing techniques. Our approach deems an object as combination of components, where each component contains different attributes (color, size, etc) and consists of near pixels with the same color. The object discovery method is divided in four phases: (1) discovery of components by gathering close pixels with the same color, (2) labeling of components according to their attributes, (3) discovery of object candidates by gathering close components, and (4) determination of valid objects among candidates, such that if the same kind of object candidates appear multiple times, they are regarded as meaningful objects. The primary contribution of this approach is to demonstrate that several essential functions in object discovery can be implemented only by hashing techniques. Especially, we introduce a hashing on Euclidean space like locality-sensitive hashing. This technique is utilized in the first and third phases to gather near points and components respectively whereas in the second phase to gather similar components under a single class. The final fourth phase is carried out by normal hashing. Throughout this work, we assume that segmented images are given as input to the system, possibly as result of a preprocessing phase composed by segmentation and quantization.

We strongly believe that the elemental functions of pattern recognition can be consistently implemented by hashing techniques, which indicates that it is possible to adapt it to different pattern recognition problems. Because our approach is a kind of component based object recognition and does not examine the rigorous location relation between components, the same kind of objects are identified as such robustly against rotation and slide operations.

In the remaining of Sect. 1, a brief review of the related work is presented. In Sect. 2, the hashing technique for finding approximated near neighbors that motivates our work is introduced. Sect. 3 explains how to cluster near neighbor points obtained by hashing. After that, we describe our automatic object discovery system in detail in Sect. 4. In Sect. 5 the evaluation result of our system is reported. Sect. 6 mentions the conclusion and the future work.

1.2 Related Work

About the automatic acquisition of models, finding generating grammars from texts are known as the grammatical inference in literature [2]. However the extension of this approach to images has not succeeded yet due to the wide variety of objects. With respect to the component-based object recognition, some previous results are known in which the recognized objectives are human faces [7] and body [4]. In both cases, positive and negative examples are given to train SVM's (support vector machine). Similarly, some probabilistic approaches [8][6] attempt to learn object models by employing unsupervised training of given examples.

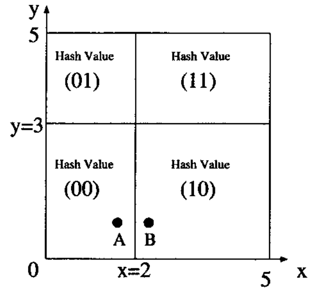

Figure 1: Space partition by a hash function in LSH

2 Locality-Sensitive Hashing

LSH [5] is a randomized algorithm for searching approximated nearest neighbor points for a given query point q from a set of points P in a high-dimensional Euclidean space. LSH uses a hash function satisfying the property that near points are stored into the same entry (bucket) with high probability and narrows the range of search for the nearest neighbors. In addition, LSH prepares multiple hash functions to reduce probabilistic failure. We explain this mechanism in detail from now on.

Let $p = (x_1, x_2, \ldots, x_d)$ be a point in a d-dimensional space. where the maximal coordinate value of any point is limited to be less than a constant C. We can transform p to a Cd-dimensional vector $v(p) = \text{Unary}(x_1)\text{Unary}(x_2)\cdots\text{Unary}(x_d)$ by concatenating unary expressions for every coordinate. Here Unary(x) is a sequence of x ones followed by $C - x$ zeros. In LSH, the hash value is computed by picking up k bits randomly from these Cd bits and concatenating them, where k is a parameter of this algorithm. In a word, this process corresponds to cutting the d-dimensional space into cells by k hyperplanes. Fig. 1 illustrates a two-dimensional case such that $C = 5$ and $k = 2$. The hash value of each point is shown, when the second bit (i.e. the line $x = 2$) and the eighth bit (i.e. the line $y = 3$) are selected from the total $2 \times 5 = 10$bits. This figure shows that near points tend to take the same hash value. As k becomes large, remote points are less likely to take the same hash value because the sizes of generated cells grow small. We refer to k as the number of *sample bits* hereafter.

By contrast, depending on the result of space division, near points may take different hash values (e.g. point A and point B in Fig. 1). To exclude this failure, multiple l hash values $h_1, h_2, \cdots h_l$ are prepared in LSH, expecting that two points close to each other will take the same hash value at least for one hash function. However, when l becomes large, the increase of overhead in computing hash functions and memory consumption should be cared about.

After all, the procedure to find the nearest point to q from the set of points P is summarized as follows.

1. For each point in P, l hash values are computed. As the result l hash tables are built.
2. Similarly, l hash values are computed for q.
3. Let P_i be the set of points in P classified into the same bucket as q on the hash table for h_i. In LSH, the points in $P_1 \cup P_2 \cup \cdots \cup P_l$ become the candidates for the nearest point to q. Among them, the nearest point to q is determined by calculating the distance to q actually.

3 Hash-based Clustering

This section explains our clustering algorithm that makes use of the hash tables. This algorithm plays an important role in our system for object discovery. We remark that the description in this section is only conceptual and not necessarily accurate. This is because we add some modification to LSH later, though the explanation assumes the original LSH here.

Our algorithm is motivated the subsequent observation about LSH: On one hand, by increasing the number of sample bits, points stored in the same hash bucket will become close to one another. On the other hand, by exploiting multiple hash functions, near points will be stored in the same hash bucket at least on one hash table. Supported by this idea, our clustering algorithm adopts the next rule.

Rule 1 *Points stored in the same hash bucket at least on one hash table are classified into the same cluster.*

Let us illustrate this with Fig. 2. Lets assume we have two hash functions denoted by h_1 and h_2. Suppose that the points a and c enter the same bucket on the hash table of h_1 and that the points b and c enter the same bucket on the hash table of h_2. Then, even if a and b are distributed to different hash buckets by h_1, a, b and c form a single cluster altogether. Therefore, like the single-linkage clustering algorithm [1], our algorithm has the ability to discover clusters in various shapes.

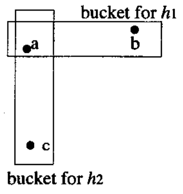

Figure 2: Our clustering algorithm

4 Automatic Object Discovery Using Hashing

One successful tool for automatic knowledge discovery is clustering which can classify data without prerequisite information. For the above reasons, we rely on clustering to achieve the discovery of objects from segmented images. In designing this scheme, we take notice of the next natural principle that:

- An object is formed from multiple components that are close to one another.
- A component consists of multiple adjacent pixels.

Motivated by these principles, we designed the discovery system to be performed into four fundamental phases described below.

1. Discovery of components by gathering near pixels with the same color.
2. Classification of components according to their attribute values. Each class is symbolized by a label ID.
3. Discovery of object candidates by gathering close components.
4. Determination of valid objects among candidates such that, if the same kind of object candidates appear multiple times, they are regarded as meaningful objects.

In Fig. 3 we illustrate the block diagram of our system. The remarkable feature of our system is to use the hashing technique through all phase, aiming to be easily implemented. More precisely, the first three phases utilize of the locality-sensitive hashing that categorizes near points on the feature space into the same cluster, and then the fourth phase by standard hashing technique.

4.1 Discovery of Components

For each color, by applying our hash-based clustering to the (x,y) coordinates of pixels of that color,

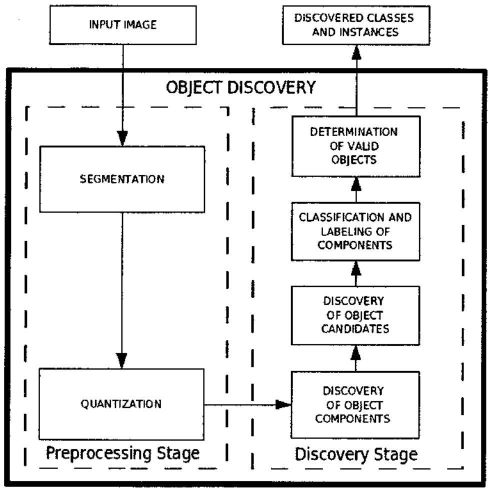
INPUT IMAGE
DISCOVERED CLASSES AND INSTANCES
OBJECT DISCOVERY
SEGMENTATION
QUANTIZATION
DETERMINATION OF VALID OBJECTS
CLASSIFICATION AND LABELING OF COMPONENTS
DISCOVERY OF OBJECT CANDIDATES
DISCOVERY OF OBJECT COMPONENTS
Preprocessing Stage
Discovery Stage

Figure 3: Overview of the Object Discovery

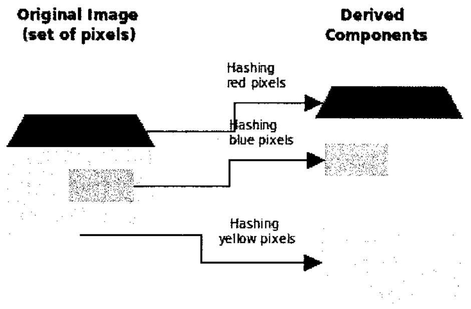

Figure 4: Extraction of object candidates

we can get near pixels of the same color as a single cluster. As the result, components of objects are derived. See Fig. 4. In this example, three components corresponding to "a roof of house", "a wall of house" and "a window of house" respectively are derived from the source image deemed as a simple set of pixels.

We define deterministic hash functions in this phase by selecting the sample bits at equal intervals of a parameter I. Let X_{max} (Y_{max}) denote the number of columns (rows respectively) of the given image. Then $\frac{X_{max}+Y_{max}}{I}$ sample bits are chosen in total for a single hash function and we can prepare I hash functions so that the sample bits of the I hash functions do not coincide one another at all in the following way.

	Locations of sample bits
1st hash function:	$1, I+1, 2I+1, \cdots$
2nd hash function:	$2, I+2, 2I+2, \cdots$
$\cdots\cdots$	
I-th hash function:	$I, 2I, 3I, \cdots$

Note that, since the locations of the sample bits are slided by one dot both in the direction of the x axis and the y axis, the resulting cells are slided in the direction of a vector (1,1).

Now we discuss the accuracy of our strategy in discovering components from the next two viewpoints, that is, (I) two separate components are not extracted as a single component falsely and (II) a single component is extracted without separated into pieces.

About the first viewpoint, as the cells generated by our hash functions always become squares with edges of length I, the next theorem holds. This theorem claims that the parameter I specifies the exactness of our algorithm.

Theorem 1 *If the minimum distance between two separate components of the same color exceeds $\sqrt{2}I$, these components are not extracted as a single connected component falsely by our algorithm.*

As for the second viewpoint, for any point (X,Y), all of the four points $(X+1,Y)$, $(X-1,Y)$, $(X,Y+1)$ and $(X,Y-1)$ are guaranteed to be stored in the same cells as (X,Y) for some hash functions. Therefore,

Theorem 2 *A single component is never divided into multiple components falsely by our algorithm.*

4.2 Classification and Labeling of Components

Components found in the first phase are classified and labeled according to their attributes (color, size, etc). Thereby, similar components are gathered into the same class by applying our hash-based clustering to the attributes (size) of components. To handle the relative difference between attributes values, we select the sample bits in the LSH algorithm at intervals proportional to its size in such a way that we can group components with a difference smaller than a threshold fixed in relation to their sizes as shown in Fig. 5. For example, $a_i = 100$ and $a_j = 110$ are identically clustered as $a_i = 1000$ and $a_j = 1100$ despite of the difference between one an another case.

Instead of the clustering technique used in the previous phases, the one presented in [3] is utilized. This technique defines a cluster as a graph representing the relation between similar pairs. Thus, all graphs are partitioned in such a way that in each

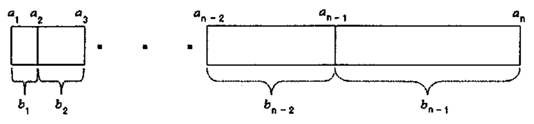

Figure 5: Selection of sample bits.

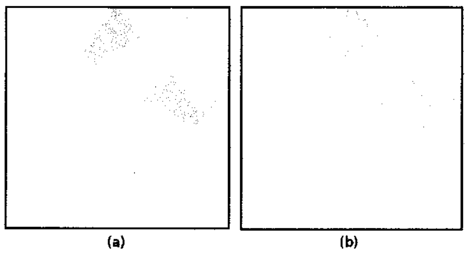

Figure 6: Green components found and their contour.

Figure 7: Object candidates.

cluster there is a center and all the other nodes in the cluster have an edge to the center.

After the labeling process, we define an order of priority $l_1, l_2, \ldots, l_M$ among labels according to their sizes, such that l_1 corresponds to the largest size and l_M corresponds to the smallest size.

4.3 Discovery of Object Candidates

To discover object candidates, our system first computes the contour of each component by the next rule.

Definition 1 *Let $P = p_1, p_2, \ldots, p_N$ be the set of pixels in a component C. The contour of C is defined by the subset of pixels in P with at least one pixel in the 4-neighbor different to P.*

Fig. 6 illustrates the contour extraction from the components in the left image. The contours are shown in the right image. Then, by applying the hash-based clustering described in Sect. 3 to the coordinates (x, y) of all points on the contour of each component, object candidates are found.

In Fig. 7 is shown an example of the extraction of object candidates by gathering components with near pixels surrounding their contours.

After all, an object candidate is depicted as a list of classes of components.

4.4 Determination of Valid Objects

Among object candidates there can exist meaningless objects created erroneously. Therefore, we investigate how many times the same kind of object appear in the image and " it is considered as a valid object if this kind of object is placed in the image multiple times " .

Suppose that an object candidate O consists of components, each of which is labeled among the set of labels $l_1, l_2, \ldots, l_M$. Thereby, we gather the same kind of objects by using the next hashing function. Let $G(l_i)$ be the number of components labelled as l_i in O, the hash value of O is formed by concatenating each $G(l_i)$ expressed by k bits, therefore

$$h(O) = G(l_1)G(l_2)\cdots G(l_M) \qquad (1)$$

Aiming to invalidate insignificant noise, the comparison for equality between object candidates utilizes large components only, such that the sum of the area of components used in comparison exceeds 80% of the total area of the object candidate they belong to. The remaining extremely small components are not used in comparison, if any.

4.5 Advantages and Characteristics

Owing to its architecture, our approach has some interesting characteristics and advantages mentioned below.

Robustness against rotation and slide: Since the method represents a component-based approach and does not check the strict location relation, the same kind of objects are ascertained to be identical robustly against rotation and slide operations.

Robustness against noise: By ignoring small components in comparing object candidates, our system is able to see through the same kind of components against small noises.

Flexibility: Since the method works in four independent phases and manage the object attributes separately, other approaches and/or attributes can be easily incorporated to it.

In addition, our method is also applicable to the object recognition problem in which already-known

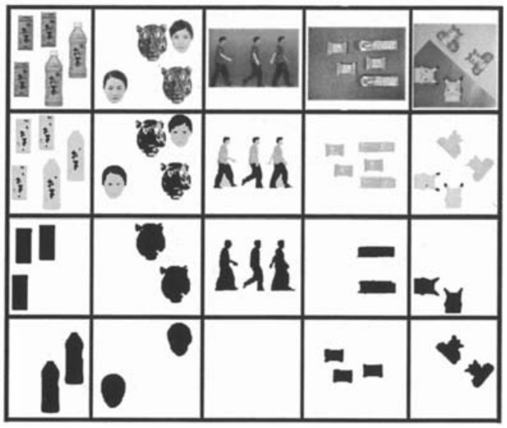

Figure 8: Experimental results: original and segmented images (rows 1 and 2) and discovered objects class 1 and 2 (rows 3 and 4).

objects are searched from a given image. All we have to do additionally is to compute the hash value of the known objects by evaluating the hash function used in the fourth phase. Then, by examining the hash bucket storing it, we can find the same object from the image as the known objects. Thus, our approach enables the unification of the knowledge acquisition process and the pattern recognition process.

5 Experimental Results

The input images were first preprocessed (segmented and quantized) by using external programs. In the first and second rows of Fig. 8 the original image and the segmented image are presented respectively. Third and fourth rows show the classes and discovered object instances. The image consist of five examples with different classes of objects and instances, in the examples (a), (b), and (c) the images are artificial ((c) was created from a image sequence) whereas (d) and (e) are natural. As we can appreciate, in all examples our system succeeded discovering objects and classes against slides and small noises. In Fig. 8 (c), we can clearly notice the robustness of our method against rotation. Example (b) is an interesting case which shows that the system can be successfully and easily extended to image sequences.

6 Conclusion

Essential functions required in the object discovery can be easily implemented by only using multiple phase hashing. The characteristics of the algorithm due to its architecture based on components and hashing techniques, make it very flexible, robust to various parameters (rotation, translation, etc) and easy to implement.

One of the future directions of this work is to use attributes except the size. Another direction is the automatic disposal of the background, and it is also very important to research how to obtain segmented images from real original images.

References

[1] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, 1972.

[2] C. M. Cook, A. Rosenfield, and A. R. Aronson. Grammatical inference by hill climbing. *Informational Sciences*, 10:59–80, 1976.

[3] T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable techniques for clustering the web. In *Proc. of the Third International Workshop on the Web and Databases (Informal Proceedings)*, pages 129–134, 2000.

[4] B. Heisele, T. Serre, M. Pontil, and T. Poggio. Component-based face detection. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 657–662, 2001.

[5] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. of 30th ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[6] D. Liu, D. Chen, and T. Chen. Latent layout analysis for discovering objects in images. In *Proc. of the IEEE International Conference on Pattern Recognition (ICPR'06)*, pages 468–471, 2006.

[7] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(4):349–361, 2001.

[8] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 2, page 2101, 2000.