Mining Viewpoint Patterns in Image Databases

Wynne Hsu Jing Dai Mong Li Lee
School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543
{whsu, daijing, leeml}@comp.nus.edu.sg

ABSTRACT

The increasing number of image repositories has made image mining an important task because of its potential in discovering useful image patterns from a large set of images. In this paper, we introduce the notion of viewpoint patterns for image databases. Viewpoint patterns refer to patterns that capture the invariant relationships of one object from the point of view of another object. These patterns are unique and significant in images because the absolute positional information of objects for most images is not important, but rather, it is the relative distance and orientation of the objects from each other that is meaningful. We design a scalable and efficient algorithm to discover such viewpoint patterns. Experiments results on various image sets demonstrate that viewpoint patterns are meaningful and interesting to human users.

1. INTRODUCTION

Advances in digital technologies have dramatically boosted the number of image repositories. Finding meaningful patterns from large sets of images is necessary for automatic indexing, categorizing, retrieving, and analyzing these images. Early image mining research has focused on the extraction of appropriate features from the images and applied traditional data mining algorithms on the extracted features. However, this approach relies heavily on the ability to extract the correct image features so that meaningful patterns can be generated.

Instead of designing tailored image feature extraction algorithms for each new set of images, a more general approach to image mining is to discover the invariant relationships that exist across a set of images. Analyzing large set of images, we find that the absolute positional information of objects do not convey critical perceptual information, but rather, it is the invariant relationships, in the form of the relative spatial relationships among the objects in images, that are important. We term these invariant relationships the viewpoint patterns.

Figure 1 shows three kitchen plan images. Each kitchen plan consists of a subset of {cooktop (C), sink (S), refrigerator (R), microwave (M), dishwasher (D)}. Table 1 lists the objects found in the three kitchen plans and the attributes such as type and location. A quick inspection reveals that while the absolute positions of the objects in each kitchen plans are unique, there exist a fixed relationship among three objects in these plans, namely,

Sink $\xrightarrow{Left,38}$ Dishwasher $\xrightarrow{Left,51}$ Refrigerator.

We call this relationship a viewpoint pattern. Such patterns are insensitive to translational operations, and in some applications, rotational operations too. To the best of our knowledge, this is the first work to discover such relative spatial relationships in images.

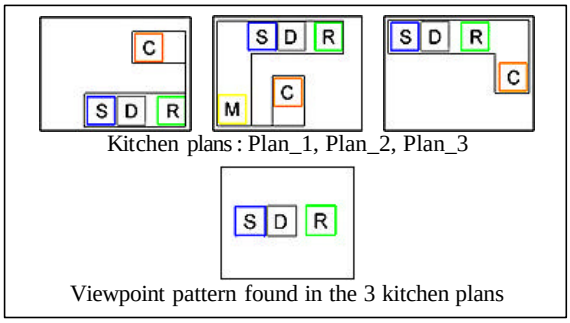

Figure 1. Sample Image Set & Viewpoint Pattern Found.

Table 1. Sample of Object Table for Images in Figure 1.

Image_id	Type	Location_X	Location_Y
Plan_1	C	140	49
Plan_1	S	79	120
Plan_1	D	117	120
Plan_1	R	168	120
Plan_2	S	64	26
…	…	…	…

In this paper, we propose an efficient, scalable, and domain-independent method called ViewpointMiner to discover viewpoint patterns in image databases. We assume that the objects or regions in the images have already been segmented and that in these images, the relative position is much more important than absolute position. The algorithm also allows for rotational-invariant viewpoint patterns to be generated. Consider the kitchen plan Plan_3 in Figure 1. If this plan is rotated 90 degrees, we are still able to discover the same viewpoint pattern shown in Figure 1. Experiments on large image collections demonstrate that viewpoint patterns are meaningful and interesting to human users.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 gives the details of our proposed

SIGKDD '03, August 24-27, 2003, Washington, DC, USA.

algorithm for mining viewpoint patterns. Section 4 analyzes the performance of the algorithm and presents our experiment results on different types of image collections. Finally, we conclude in Section 5.

2. PREVIOUS WORK

In this section, we briefly survey some existing work in image mining and spatial data mining.

2.1 Image Mining

Recent image mining approaches apply data mining techniques after preprocessing the image data to some suitable form for mining. Existing works adapted data mining algorithms such as association rule mining [10] [13], clustering [2] and classification techniques [3] [11] to generate patterns based on pixel level or object level features. While these approaches can discover hidden relationships among appearance features in the images, they ignore the patterns relating to the spatial properties of objects in the images.

The Attributed Relational Graph (ARG) has recently been used to represent frequent patterns in image mining [4]. However, [4] focuses on discovering the adjacent relationships between different regions in images and hence, it is not suitable for finding invariant relationships among disconnected image objects. Further, the patterns represented by ARG are not easy to visualize.

2.2 Spatial Mining

Spatial data mining, on the other hand, mainly focuses on discovering topological relationships from spatial databases. An example of spatial association mining in geographic maps is described in [6]. It has recently been extended to mine topological relations between object pairs from the general category images [12]. However, this approach requires a pre-defined concept hierarchy. In addition, the refinement of the topological relations requires many passes through the object database, rendering the algorithm expensive.

Attempts have also been made to find frequent object classes that typically are close together [5] [8]. In these works, the focus is on neighboring relation, and does not handle orientation information.

Other major spatial mining research include spatial classification methods [7] and spatial clustering methods [9]. These approaches aim to discover patterns in large maps. None of them deal with relative distance and orientation invariant patterns.

3. MINING OF VIEWPOINT PATTERNS

In this section, we first provide an overview of the viewpoint discovery process. This is followed by a formal problem definition and the ViewpointMiner algorithm.

3.1 Overview

We use the example kitchen plan designs in Figure 1 to illustrate the mining process. Figure 2 shows all the possible 2-object pairs in the image Plan_1. The corresponding 2-object table is shown in Table 2. Note that the unit of distance *Dist* is pixel and the unit of orientation *Orient* is radian. This 2-object table is scanned to determine the frequent 2-object patterns each time after we generate candidate patterns with increasing number of constrained attributes. Setting the minimum support to 3, the frequent 2-object patterns generated from the images in Figure 1 are:

$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist\,38, Orient\,1.57} \text{Object}_2(\text{Type=D});$$
$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist\,89, Orient\,1.57} \text{Object}_2(\text{Type=R});$$
$$\text{Object}_1(\text{Type=D}) \xrightarrow{Dist\,51, Orient\,1.57} \text{Object}_2(\text{Type=R}).$$

These 2-object patterns are further generalized to form 2-object interval patterns. For example, the following two 2-object patterns

$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist\,38, Orient\,1.57} \text{Object}_2(\,);\ \text{and}$$
$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist\,39, Orient\,1.57} \text{Object}_2(\,);$$

can be generalized to a single interval pattern:

$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist[38,39], Orient\,1.57} \text{Object}_2(\,).$$

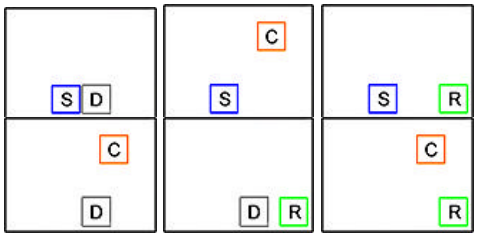

Figure 2. Object Pairs in Image Plan_1.

Table 2. Sample of the 2-Object Table for Plan_1.

Rec #	Obj_1 Type	Obj_2 Type	Dist (Pix)	Orient (Rad)	Image #
0	S	D	38	1.57	Plan_1, _2, _3
1	S	C	93	0.71	Plan_1
2	S	R	89	1.57	Plan_1, _2, _3
3	D	C	74	0.31	Plan_1
4	D	R	51	1.57	Plan_1, _2, _3
…	…	…	…	…	…

Based on the 2-object table, we can build the 3-object table. For each pair of objects in the 2-object table, we add a new object that can be found in the same image of the 2-object pair to form a new 3-object group (Figure 3).

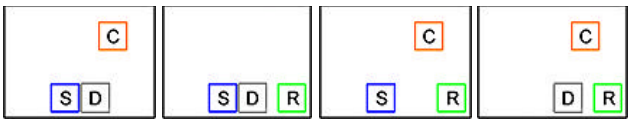

Figure 3. 3-Object Groups in Image Plan_1.

At the same time, we generate the candidate 3-object patterns by concatenating two frequent 2-object patterns. Suppose we have

$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist\,38, Orient\,1.57} \text{Object}_2(\text{Type=D});\ \text{and}$$
$$\text{Object}_1(\text{Type=D}) \xrightarrow{Dist\,97, Orient\ NULL} \text{Object}_2(\,);$$

The candidate pattern generated is:

$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist\,38, Orient\,1.57} \text{Object}_2(\text{Type=D}) \xrightarrow{Dist\,97, Orient\ NULL} \text{Object}_3(\,).$$

When scanning the 3-object table, candidates whose support counts are greater or equal to the minimum support are output as 3-object viewpoint patterns. The process is repeated until no new patterns are found. Referring to Figure 1 again, a frequent 3-object pattern that we discovered is:

$$\text{Object}_1(\text{Type=S}) \xrightarrow{Dist\,38, Orient\,1.57} \text{Object}_2(\text{Type=D}) \xrightarrow{Dist\,51, Orient\,1.57} \text{Object}_3(\text{Type=R}).$$

In order to improve the efficiency of the algorithm, two pruning strategies are used. The first strategy is to prune the k-object patterns generated, so that the number of candidates for (k+1)-object patterns can be reduced. The second strategy is to prune the k-object table after generating the k-object patterns so as to reduce the size of (k+1)-object table. In the following section, we formally define the viewpoint pattern mining problem.

3.2 Problem Definition

As mentioned, viewpoint patterns refer to patterns that demonstrate the invariant relationships of one object from the point of view of another object. Here, we focus on the distance and orientation relationships of objects with respect to another object. Formally, a viewpoint pattern involving N objects, each having K attributes can be described as follows:

$$\text{Object}_1((a_{11}, v_{11}); (a_{12}, v_{12}); \ldots (a_{1K}, v_{1K}))$$
$$\xrightarrow{Dist\ d_1, Orient\ o_1} \text{Object}_2((a_{21}, v_{21}); (a_{22}, v_{22}); \ldots (a_{2K}, v_{2K}))$$
$$\xrightarrow{Dist\ d_2, Orient\ o_2} \ldots$$
$$\xrightarrow{Dist\ d_{N-1}, Oriento_{N-1}} \text{Object}_N((a_{N1}, v_{N1}); \ldots (a_{NK}, v_{NK}))\ (Support)$$

where

1. (a_{ij}, v_{ij}) denote the jth attribute value pair used to describe $Object_i$. Note the object here is an abstract object as defined by the constraints on its attribute values;
2. $\xrightarrow{Dist\ d_i, Orient\ o_i}$ connects $Object_{(i+1)}$ with respect to $Object_i$. Note that v_{ij}, d_i and o_i can be null, a single-value, or an interval. We order these objects by imposing the constraint that $Object_i$ must appear to the left of $Object_{(i+1)}$;
3. *Support* refers to the number of times the pattern appears in the image collection. It is sensitive to reoccurrence in one image;

Our goal is to generate frequent rotational-variant and rotational-invariant viewpoint patterns from image databases. Details of the mining algorithm ViewpointMiner are described in the next subsection.

3.3 ViewpointMiner Algorithm

Let D be the image object database, where all the continuous value attributes have been discretized. Figure 4 gives the Apriori [1] based ViewpointMiner algorithm.

The output SI_k stores all the frequent k-object viewpoint patterns which consist of $S_{k,\,m}$ and I_k. $S_{k,\,m}$ are the frequent k-object patterns with the constraint that there are exactly m values in the set $\{d_{(k-1)}, o_{(k-1)}, v_{kj}$, for all j$\}$ that are single-valued, while others are all null. I_k are the frequent k-object patterns where there are some values in the set $\{d_{(k-1)}, o_{(k-1)}, v_{kj}$, for all j$\}$ that are interval. When k=2, $S_{2,\,m}$ have m single-valued attributes and no interval attributes, while I_2 have some interval attributes.

The preprocessing step in the Figure 4 prunes away the insignificant objects that are either too small or with low contrast to the background. The pattern generation process begins with k = 2. From line 4 to 14, the program generates the candidates, counts the frequent ones and generalizes them to interval patterns. Lines 15 and 16 prune the redundant interval patterns and useless object groups. This process is repeated until no new patterns are generated.

Next, we describe in greater details the six functions that are called from within the loop.

Algorithm ViewpointMiner:
Input: Image object database D;
Output: SI_k patterns for all k;

1) D' = preprocess (D);
2) For (k = 2; $S_{(k-1),\,1}$!= NULL; k++)
3) k-object table = *build (D', k, (k-1)-object table)*;
4) If (k > 2)
5) $CS_{k,\,1}$ = *candidate-gen1* ($SI_{(k-1)}$, $S_{(k-1),\,1}$); // Candidates for $S_{k,\,1}$
6) Else
7) $CS_{2,\,1}$ = set of all attribute value pairs in D';
8) $S_{k,\,1}$ = { c ∈ $CS_{k,\,1}$ | is_frequent (c) };
9) I_k = *generalize* ($S_{k,\,1}$);
10) F = # of object attributes + 2;
11) For (f = 2; f <= F and $S_{k,\,f-1}$!= NULL; f++)
12) $CS_{k,\,f}$ = *candidate-gen2* ($S_{k,\,(f-1)}$); // Candidates for $S_{k,\,f}$
13) $S_{k,\,f}$ = { c ∈ $CS_{k,\,f}$ | is_frequent (c) };
14) I_k = I_k ∪ *generalize* ($S_{k,\,f}$);
15) SI_k = *prune-pattern* ($(\dot{E}_f S_{k,\,f}) \dot{E} I_k$);
16) k-object table = *prune-obj* (*k-object table*, SI_k);
17)End

Figure 4. Algorithm ViewpointMiner

build (D', k, (k-1)-object table): This is the most time consuming part of the algorithm. This function scans the database and records all combinations of k objects that appear in each image into k-object table. The k-object table is indexed by a hash function. The size of the k-object table is controlled by function preprocess (D). In this function, the distance and orientation among the k objects are calculated and stored (Figure 5).

build (D', k, (k-1)-object table):
Input: preprocessed object database D', number of objects k, (k-1)-object table;
Output: k-object table;

1) For each image I in D'
2) S = set of objects occurring in I;
3) IdSet = ids of the (k-1)-object records that occurs in I;
4) For each id in IdSet
5) Find (k-1)-object record *R* via hash function;
6) For each object O in S
7) If O is not contained in R
8) Generate a candidate k-object record C by inserting O to R sorted by location;
9) If all the k-1 object subsets of C are in (k-1)-object table
10) If C is already in the k-object table
11) Increment the count of C;
12) Else add C to the k-object table;
13) Delete the (k-1)-object table;
14) End

Figure 5. Function build (D', k, (k-1)-object table)

candidate-gen1 ($SI_{(k-1)}$, $S_{(k-1),\,1}$): This function generates the candidates of $S_{k,\,1}$ patterns. It is similar to the candidate generation function of the Apriori algorithm [1]. The difference is that each candidate is a sequence of objects where the order of the objects is important. In this function, each candidate for $S_{k,\,1}$ pattern is generated by adding $Object_{(k-1)}$, $o_{(k-2)}$ and $d_{(k-2)}$ of a $S_{(k-1),\,1}$ pattern

to the end of a $SI_{(k-1)}$ pattern, at the same time the sub-pattern formed by the last k-1 objects and k-2 distances and orientations of the candidate must cover the $S_{(k-1),\ 1}$ pattern. Here we have A cover B if and only if all the objects that fit pattern B also fit pattern A. Note that we select a special subset of $SI_{(k-1)}$ and $S_{(k-1),\ 1}$ as the input parameters to the function. This enables us to limit the number of patterns generated (Figure 6).

candidate-gen1 ($SI_{(k-1)}$, $S_{(k-1),\ 1}$):
Input: patterns $SI_{(k-1)}$, single-value patterns $S_{(k-1),\ 1}$;
Output: candidates of $S_{k,\ 1}$: $CS_{k,\ 1}$;

1) For each pattern P_S in $S_{(k-1),\ 1}$
2) For each pattern P_{SI} in $SI_{(k-1)}$
3) If the last k-2 objects and k-3 dist and orient of P_{SI} cover the first k-2 objects and k-3 dist and orient of P_S
4) Generate candidate T by adding the last object and dist and orient of P_S to the end of P_{SI};
5) Add T to $CS_{k,\ 1}$;
6) End

Figure 6. Function candidate-gen1 ($SI_{(k-1)}$, $S_{(k-1),\ 1}$)

candidate-gen2 ($S_{k,\ (f-1)}$): The function generates the $S_{k,f}$ candidate patterns from $S_{k,(f-1)}$ patterns. For each pair of $S_{k,(f-1)}$ patterns with the same first k-1 objects and k-2 dist and orient, if they have only one different attribute value in the last object and last dist and orient, the function generates a temporary candidate by combine them together. If the temporary candidate can be covered by some $S_{k,(f-1)}$ pattern after setting any attribute value of the last object or the last dist or orient as null, it is stored as a candidate pattern (Figure 7).

candidate-gen2 ($S_{k,\ (f-1)}$):
Input: single-value patterns $S_{k,\ (f-1)}$;
Output: candidates of $S_{k,\ f}$: $CS_{k,\ f}$;

1) For each pair of $S_{k,\ (f-1)}$ (P1, P2)
2) If P1, P2 have same first k-1 objects and k-2 dist and orient
3) If P1, P2 have (f-2) same non-null attribute values for $Object_k$, $d_{(k-1)}$ and $o_{(k-1)}$
4) Generate temporary candidate T by adding the attribute that P1 has while P2 has not to P2;
5) If T can be covered by patterns in $S_{k,(f-1)}$ after remove any attribute of $Object_k$ or $d_{(k-1)}$ or $o_{(k-1)}$ from T
6) Add T into $CS_{k,\ f}$;
7) End

Figure 7. Function candidate-gen2 ($S_{k,\ (f-1)}$)

generalize ($S_{k,\ j}$): This function generates I_k by merging the adjacent segments in $S_{k,\ j}$ patterns (Figure 8). It makes use of a sub-generalize function to create the interval range for each selected attribute (see Figure 9).

prune-pattern (SI_k): Since function generalize ($S_{k,\ f}$) produces many interval patterns, there will be some patterns that are already covered by others. This function prunes away the redundant patterns. In so doing, it reduces the number of candidates for the outer loop as well as the final result (Figure 10).

generalize ($S_{k,\ f}$):
Input: single-value patterns $S_{k,\ f}$;
Output: interval-value patterns I_k;

1) For (i = 1; i <= f; i++)
2) $I_{\{ai\}}$ = *sub-generalize*($S_{k,\ f}$, a_i); //a_i, a_{ki} is one of the last f attributes in $S_{k,\ f}$
3) Add $I_{\{ai\}}$ into I_k;
4) For (i = 2; i <= f; i++)
5) For each combination of i attributes $\{a_{k1},\ldots a_{ki}\}$
6) $I_{\{ak1,\ldots aki\}}$ = *sub-generalize* ($I_{\{ak1,\ldots ak(i-1)\}}$, a_{ki});
7) Add $I_{\{ak1,\ldots aki\}}$ into I_k;
8) End

Figure 8. Function generalize ($S_{k,\ f}$)

sub-generalize (SI_k, A):
Input: patterns SI_k, attribute A;
Output: interval-value patterns I_k generalized on A;

1) Group patterns in SI_k among which the only different attribute value is A, and sort each group by A in ascending order;
2) For each group G (p_1, p_2,…, p_N)
3) ind = 2;
4) For (j = 1; j < N; j++);
5) ind --;
6) For (i = window_size; i > ind-1; i--)
7) If (p_j.A == p_{j+i}.A - i)
8) Generate pattern P from p_j by changing value of A to [p_j.A, p_{j+i}.A] and Support to sum of i patterns';
9) Add P into I_k;
10) ind = i;
11) i = 0;
12) End

Figure 9. Function sub-generalize (SI_k, A)

prune-pattern (SI_k):
Input: all the k-object viewpoint patterns SI_k;
Output: k-object patterns SI_k without redundant patterns;

1) For each pair (P1, P2) of patterns in SI_k
2) If P1 is covered by P2
3) If for some i, j, v_{ij} is interval or d_i is interval or o_i is interval in P1
4) If P1.Support is equal to P2.Support
5) Prune P2;
6) Else Prune P1;
7) End

Figure 10. Function prune-pattern (SI_k)

prune-obj (k-object table, SI_k): This function prunes away the records that are not covered by SI_k patterns from D'. If a k-object record is not covered by SI_k patterns, it implies that the record will not be covered by any $SI_{(k+1)}$ patterns. Hence, after the prune-obj function, only the useful k-object records will be kept in the k-object table to generate the (k+1)-object table. This is a simple but important optimization because it decreases the size of the in-memory table (Figure 11).

The ViewpointMiner algorithm incurs low I/O cost, since the number of times to scan the image object database is equal to the

number of objects in the longest pattern minus one. The most time-consuming module is the build function which has an average runtime linear to the total number of images.

prune-obj (k-object table, SI_k):
Input: k-object table, patterns SI_k;
Output: k-object table without useless records;

1) For each record R in k-object table
2) If R is not covered by any pattern in SI_k
3) Delete R from k-object table;
4) End

Figure 11. Function prune-obj (k-object table, SI_k)

3.4 Rotational-Invariant Considerations

The algorithm described above is sensitive to the rotational operations. As a result, images in Figure 12 will have two different viewpoint patterns. However, from an interior designer's point of view, these two patterns essentially describe the same design principle. To discover such rotational-invariant viewpoint patterns, we ignore the orientation information by first mapping each 2-object group into some existing candidate patterns. This mapping is achieved by rotating the object pairs at various angles such as p/2, p, and 3p/2. With this mapping, the "S-D" group from the last set of images of Figure 12 will be mapped to the same viewpoint patterns as the first set of images. The mapping angle is assigned to the "S-D" group so that subsequent 3-object groups can be generated using the mapped relative positions.

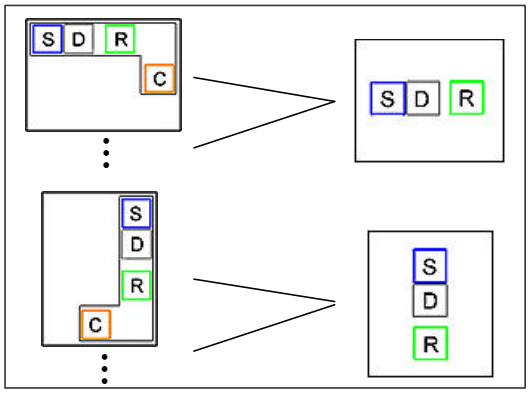

Figure 12. Rotational-Variant Patterns.

The following modifications need to be made to the ViewpointMiner in order to generate rotational-invariant patterns:

1. When generating the candidate pattern $CS_{2,1}$ (Line 7 of Figure 4) where $CS_{2,1}$ involves an orientation attribute, we restrict the orientation values to the interval [0, p/2].
2. When counting the frequencies of the $S_{2,1}$ patterns with orientation attributes (Line 8 of Figure 4), the orientation o_i of each object pair i is expanded to four values: o_i, o_i+p/2, o_i+p, o_i+3p/2. If one of these four orientations can match an existing candidate pattern, then we assign the matched rotation value to the object pair and increase the frequency count of the matched candidate pattern.
3. When generating the candidate pattern $CS_{2,f}$ using the function *candidate-gen2* ($S_{2,(f-1)}$) (Line 12 of Figure 4), we combine two $S_{2,(f-1)}$ patterns to obtain a $S_{2,f}$ candidate pattern. The common rotational angles of these two $S_{2,(f-1)}$ patterns should be kept by the $S_{2,f}$ candidate pattern for subsequent use by *candidate-gen1* (SI_2, $S_{2,1}$).
4. In the function *build (D', k, (k-1)-object table)* for k > 2 (line 3 of Figure 4), when adding a new object to generate a candidate k-object record (line 8 of Figure 5), the relative position of the new object is calculated according to the rotational angle of the (k-1)-object record.
5. In the function *candidate-gen1* (SI_2, $S_{2,1}$) (Line 5 of Figure 4), when combining two 2-object patterns to generate $CS_{3,1}$, the 2-object patterns are instantiated to multiple patterns according to the rotational angles associated with the 2-object patterns. This allows the two 2-object patterns to generate more than one $S_{3,1}$ candidate pattern.

With these modifications, the viewpoint mining algorithm can now find viewpoint patterns from rotation-insensitive images.

4. EXPERIMENTS

In this section, we evaluate the algorithm ViewpointMiner on different types of image databases.

4.1 Experiments on General Category Images

The most expensive function in the ViewpointMiner algorithm is the function *build* that involves scanning the image database to construct the k-object groups. Theoretically, with an appropriate hash function, the time complexity of the function *build* is O(m), where m is the number of images in the database.

We use 5000 images from a general category image collection. We extract over 25,000 objects after restricting the number of objects in each image to be 7. Figure 13 shows a sample of the images and their corresponding extracted objects. The minimum support is set to be 0.15% of the size of the 2-object table. Experiment results show that, in total, we have 57,058 2-object groups and 71,602 3-object groups, and 51,575 4-object groups without object pruning. Figure 14 shows the time needed to build the k-object table as we vary the number of images from 1000 to 5000. We observe that the time needed to build the k-object tables is linear to the number of images.

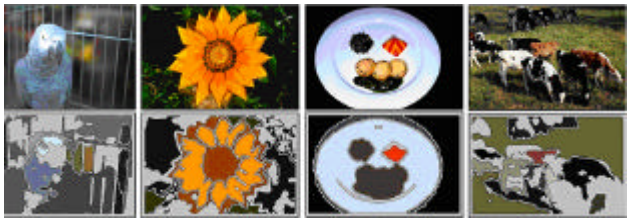

Figure 13. Samples from General Category Image Set.

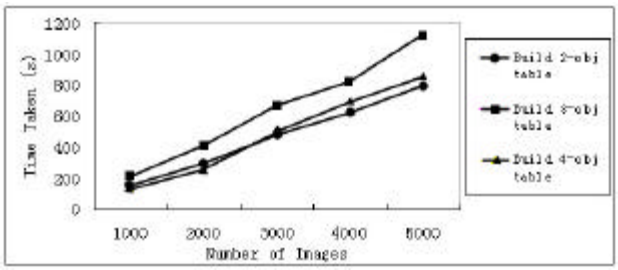

Figure 14. Time Taken to Build k-object Table.

Next, we investigate the effect of pruning on the performance of the algorithm (Figure 15). The results show that after pruning objects using the $S_{2,3}$ patterns, only 14.3% of the 2-object groups remains, and 2,978 3-object groups are generated as compared to the 71,602 groups generated without pruning. This indicates that our viewpoint pattern mining algorithm is scalable and efficient.

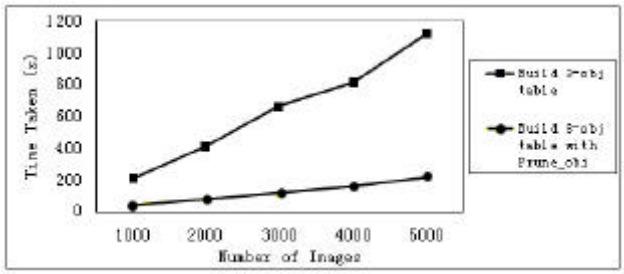

Figure 15. Time Taken by Prune-obj Step

4.2 Experiments on Kitchen Plan Images

In this set of experiment, we collected eleven original kitchen plan images from the Internet. All the plans include a cooktop (C), a sink (S) and a refrigerator (R). Some of the plans also have microwaves (M) and dishwashers (D). Based on these eleven plans, we infer the underlying design principles and generated 40 kitchen plans that adhere to the design principles. In addition, we generated 80 kitchen plan images by randomly placing the five items in each kitchen plan. In total, we have 120 kitchen plan images in which 40 are regarded as meaningful(seeding images).

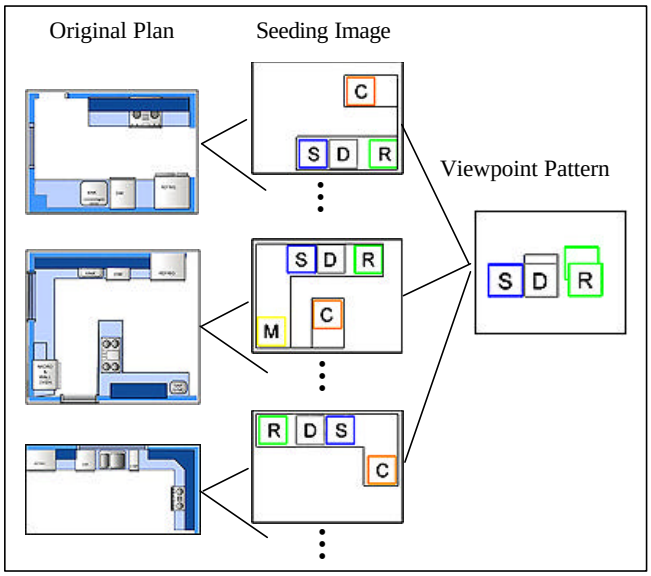

Figure 16. Discovering Pattern from Kitchen Plans

We set the minimum support count to 3, and found 39 2-object patterns, 4 3-object patterns and 1 4-object pattern. Figure 16 shows the process of generating a 3-object pattern. The 2-object patterns covered 150 object pairs, in which 104 pairs are from the seeding images, while the 3-object patterns and 4-object patterns can be traced back to the seeding images. We conclude that the viewpoint patterns generated do capture the underlying design principles in spite of the 67% random noise introduced. Based on the discovered patterns, we can infer some good kitchen design principles such as: arrange the sink next to the dishwasher and place the refrigerator next to the microwave.

5. CONCLUSION

Motivated by the need for automatic indexing, categorizing, retrieving, and analyzing of increasingly large sets of images, this paper explores how image mining can be used to find meaningful patterns from images repositories. We observe that the absolute positional information of objects do not convey critical perceptual information, but rather, it is the invariant relationship, in the form of the relative spatial relationships among the objects in images, that is important.

We introduce the concept of a viewpoint pattern to describe the fixed, invariant relationships among the objects in images. An algorithm called ViewpointMiner has been designed to discover viewpoint patterns from large image collections. Experiments results on general category images and architectural design images demonstrate that ViewpointMiner is efficient and scalable, and is able to discover meaningful patterns from real-world images.

6. REFERENCES

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. *VLDB*, 1994.

[2] E. Chang, C. Li, J. Wang, P. Mork, and G. Wiederhold. Searching Near-Replicas of Images via Clustering. *SPIE Symp. of Voice, Video and Data Communications*, 1999.

[3] B. Fang, W. Hsu, and M. Lee. Tumor Cell Identification Using Feature Rules. *SIGKDD*, 2002.

[4] P. Hong and T. Huang. Mining Inexact Spatial Patterns. *Workshop on Discrete Mathematics and Data Mining*, 2002.

[5] Y. Huang, H. Xiong, S. Shekhar, and J. Pei. Mining Confident Co-location Rules without a Support Threshold. *18th ACM Symp. on Applied Computing*, 2003.

[6] K. Koperski and J. Han. Discovery of Spatial Association Rules in Geographic Information Database. *4th Int. Symp. on Large Spatial Databases*, 1995.

[7] K. Koperski, N. Stefanovic, and J. Han. An Efficient Two-Step Method for Classification of Spatial Data. *Int. Symp. on Spatial Data Handling*, 1998.

[8] Y. Morimoto. Mining Frequent Neighboring Class Sets in Spatial Database. *SIGKDD*, 2001.

[9] R. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. *VLDB*, 1994.

[10] C. Ordonez and E. Omiecinski. Discovering Association Rules based on Image Content. *IEEE Advances in Digital Libraries Conference*, 1999.

[11] A. Vailaya, M. Figueiredo, A. Jain, and H. Zhang. Image Classification for Content-Based Indexing. *IEEE Transaction on Image Processing*, Vol. 10, No. 1, 2001.

[12] O. Zaiane and J. Han. Mining Recurrent Items in Multimedia with Progressive Resolution Refinement. *ICDE*, 2000.

[13] O. Zaiane, J. Han, Z. Li, S. Chee, and J. Chiang. MultiMediaMiner: A System Prototype for MultiMedia Data Mining. *ACM SIGMOD*, 1998.