"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XVIII – 2015 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in:
PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

MINING APPROACHES IN VISUAL DATA SETS

P.V. MARINOV[1]
[1]Technical University-Sofia/Department of Computer Science, Sofia, Bulgaria

***Abstract**: Quality and the speed of query reply in internet visual search is a problem with increasing importance due to the expanding of the shearing visual data. Searching advanced solutions of this problem, mining methods are recently widely used. The paper is dedicated to the applications of data mining approaches to frequent item sets in visual data. The description of the problem of the item set mining is given, firstly. Further, some efficient algorithms for solving the problem are discussed. In the end, several measures for evaluating the quality of the item sets and the effectiveness of the association rules are considered.*
***Keywords:** Data mining, Frequent itemsets mining, Association rules.*

FREQUENT ITEMSET MINING

Recently enormous amount of images and videos are shared on Internet. Many retrieval and mining algorithms for visual data are developing to advance this process. Frequent itemset mining is a popular family of methods to detect the combined appearance of items from a large body of data. They come from the market basket analysis, where large databases of customer transactions have to be investigated in order to understand the buying habits of shoppers. [1]. Similar problems appear in web mining [2], fraud detection in on-line advertising [3], document analysis and massive recommendation systems for related search queries [4]. Very efficient are algorithms laying on powerful local image features methods.

In this paper a formulation of the itemset mining problem is given and algorithms solving the problem are discussed. Some quality measures for the mining results are given. The paper considers itemset mining algorithms in the domain of visual data adapting them to work with local visual features. The resulting algorithms successfully identify frequent feature configurations as representatives of object classes and support the mining of specific objects in video data.

Frequent Itemset and Association Rules

An *itemset I* is a set $I = \{i_1 \ldots i_p\}$ of *p* items.

Let *A* be a subset of *I* with *l* ($l \leq p$) items, *i.e.* $A \in I,\ |A| = l$. Then we call *A* a *l-itemset*.

A *transaction* is an itemset $T \subseteq I$ with a transaction identifier *tid*(*T*).

A *transaction database D* is a set of transactions with unique identifiers

Definition 1. The *support of an itemset* $A \in D$ is

$$support(A) := \frac{|\{T \in D \mid A \subseteq T\}|}{|D|} \in [0,1] \quad (1.1)$$

Conversely, for each item set the transactions can be find, which support it.

Definition 2. The cover of an itemset *A* in *D* consists of the set of transaction identifiers of transactions in *D* that support A:

$$cover(A, D) := tid(T) \mid (T \in D, A \subseteq T) \quad (1.2)$$

In item set mining process, the sets that occur frequently in the database are interesting:

$$D = \{tid(T_1) \ldots tid(T_n)\},\ tid(T_i) \neq tid(T_j)\ \ \forall \{i, j\} \in I \mid i \neq j \quad (1.3)$$

Definition 3. An itemset *A* is called *frequent* in *D* if $support(A) \geq s$, where s is a threshold for the minimal support defined by an expert.

Two special types of frequent itemsets are often considered in the literature. These are: *Closed itemset* that is a frequent item set A, for which no superset has the same support; and *Maximal itemset* that is a frequent item set A for which no superset is frequent.

The statistical dependence between the items (subsets) of the set in mining frequent itemsets is an important measure given in the form of association rules.

Definition 4. An *association rule* is an expression $A \to B$ where *A* and *B* are itemsets (of any length) and $A \cap B = 0$.

The quality of an association rule is given by its support confidence [1].

Definition 5. The *support of an association rule* $A \to B$ is

$$supp(A \to B) := supp(A \cup B) = \frac{|\{T \in D \mid (A \cup B) \subseteq T\}|}{|D|} \quad (1.4)$$

the support of the common itemsets, which make up the rule. It measures its statistical significance.

Definition 6. The *confidence of an association rule* $A \to B$ is

$$supp(A \to B) := supp(A \cup B) = \frac{|\{T \in D \mid (A \cup B) \subseteq T\}|}{|D|} \quad (1.5)$$

The left-hand side of a rule is called antecedent, the right-hand side is the consequent.

The confidence is a measure of the strength of the implication $A \to B$. It can be considered as a maximum possible estimate of the conditional probability that B is true given that A is true [5].

The tasks that itemset mining algorithms have to solve are:

1) given a minimal support threshold s, to detect all frequent itemsets (i.e. A | support(A) > s) in a database D in an efficient manner.

2) to create association rules from the mined itemsets.

Some algorithms which fulfill this task are described below.

Frequent Itemset Mining Algorithms

The earliest and the most widespread itemset mining algorithm is APriori algorithm [1]. Many improvements of its first version algorithms are proposed. The most notably is FP-Growth [6].

The efficient frequent itemset search in APriori algorithms is based on monotony property, i.e. the downward closure property:

Downward closure of support: Let *D* be a database and let *A* and *B* be two itemsets. Then

$$A \subseteq B \rightarrow supp(B) \leq supp(A) \qquad (1.6)$$

This follows immediately from $cover(B) \subseteq cover(A)$. In other words all *l*-subsets of frequent (*l*+1)-sets must also be frequent.

APriori

The APriori algorithm performs a breadth-first search through the search space of all itemsets. It generates iteratively candidate itemsets C_{l+1} of size $l + 1$. It works in two phases: 1) a database pass phase, where the support of the itemsets in C_l is calculated and checked if it exceeds the frequency threshold *s*, and 2) a candidate formation for *l*+1 itemsets. The code could be shown as follows:

1: $l \leftarrow 1,\ L \leftarrow 0$
2: $C_l \leftarrow \{\{A\}\} \mid A\ item\ of\ size\ 1, A \in D\}$
3: while $C_l \neq 0$ do
4: $L_l \leftarrow 0$
5: database pass:
6: for $A \in C_l$ do
7: if *A* is frequent then
8: $L_l \leftarrow L_l \bigcup A$
9: end if
10: end for
11: candidate formation:
12: $C_{l+1} \leftarrow$ sets of size *l*+1, whose all subsets are frequent
13: $C_l \leftarrow C_{l+1}$
14: $L \leftarrow L \bigcup L_l$
15: end while
16: return *L*

The number of iterations of the algorithm depends on the data, on the number of items in the largest frequent set.

The computational complexity of the algorithm can be divided by the two phases, the database pass phase and the candidate generation phase. The worst-case complexity for each iteration of the database pass phase is approximately square in the number of frequent *l*-itemsets L_l, used to generate the $l + 1$ sets. In practice this part of the algorithm usually runs linear in L_l. Checking a set C_l for frequency requires testing its presence in all transactions of the database *D*, in each iteration the complexity is thus $O(|C_l|np)$, where C_l is the number of candidate *l*-itemsets, *n* is the number of transactions and *p* is the number of items.

The main disadvantage of the APriori algorithm is that it requires multiple passes over the database during the support counting procedure.

An improved version is AprioriTid algorithm that reduces the time needed for the support counting procedure. At every iteration *l* repeatedly it replaces every transaction in the database by the set of candidate itemsets that occur in that transaction [1]. AprioriTID algorithm has the additional property that the database is not used at all for counting the support of candidate itemset after the first pass. An encoding of the candidate itemsets used in the previous pass is employed for this purpose.

This property allows to be found frequent itemsets very quickly.

FP-Growth

FP-Growth (Frequent Pattern Growth) algorithm applies a depth-first search [6]. It uses two additional conditions besides downward closure property:

• Consider the *cover*(*A*) of an itemset *A* Transactions containing *A* can be selected to form a conditional database (CDB), and to find patterns containing *A* from that conditional database *{a, b}, {a, c}, {a}* → *{a, b, c}*.

• All itemsets should be sorted by the same manner (e.g., by descending support) to prevent the same pattern from being found in multiple conditional databases.

From the first condition, it follows that the conditional databases are smaller sub-problems to be solved. FP-growth uses a tree structure to store the database in a compressed form. Firstly, the algorithm removes infrequent items and sorts the transactions based on the remaining items. Then it compresses these cleaned transactions into a prefix Frequent Pattern tree (the FP-tree), the root of which is the most frequent item. Each path on the tree represents a set of transactions that share the same prefix; each node corresponds to one item. Each level of the tree corresponds to one item, and an item list is formed to link all transactions that possess that item. Storing transactions in the FP-tree in support descending order helps keeping the database small, since in general the more frequently occurring items are arranged closer to the root of the FP-tree and thus are more expected to be shared.

FP-Growth then starts to mine the FP-tree for each item whose support is larger than *s* by recursively building its conditional FP-tree. With this, the problem of finding frequent itemsets is converted to searching and constructing trees recursively.

Algorithm: FP-Growth
Data: Database D, minimal support *s*
Result: Frequent itemsets

1: Define and clear F-list : *F*[]
2: foreach *Transaction* $T_i \in D$ do
3: foreach *Item* $a_j \in T_i$ do
4: $F[a_j]++$;
5: end
6: end
7: Sort *F*[];
8: Define and clear the root of FP-tree: *r*;
9: foreach *Transaction* $T_i \in D$ do
10: Make T_i ordered according to *F*[];
11: Construct Tree (T_i,*r*);
12: end
13: foreach *Item* $a_i \in I$ do
14: call Growth (*r*,a_i,*s*);

15: end
Then it recursively runs the procedure *Growth*
Procedure Growth
Data: *r*,*a*,*s*
1: if *r contains path Z* then
2: foreach *combination γ of the nodes in Z* do
3: Generate pattern $\beta = \gamma \cup \alpha$ with support = minimum
support of nodes in γ
if *support*(*β*) > *s* than
4: Output (*β*)
5: end
6: end
7: end
8: else
9: foreach b_i in *r* do
10: Generate pattern $\beta = \beta_i \cup \alpha$ with support = support(b_i)
if *support*(*β*) > s than
which builds and searches the conditional trees [4].

Interestingness Measures for Itemsets and Rules

Presented above algorithms give measures for support and confidence to assess the quality of frequent itemsets and corresponding association rules. Depending on the application one can use other appropriate filters.

Organizing the mining of itemsets, interesting are the sets whose items show either strong dependence or weak independence. The expected value for the support of an itemset is computed from the product of the supports of the individual items. The difference to 1 of the ratio of actual and expected support of an itemset can be good quality measure:

$$dep(A) = 1 - \frac{\prod_{i=1}^{l} support(A[i])}{support(A)} \qquad (1.7)$$

where *A* is an itemset of length *l* and *A*[*i*] is the *i*-th item of the itemset *A*. Only itemsets for which this difference is above a given threshold are then retained as interesting. That is why, this is especially interesting.

1.4. Graph Mining

Graph mining is structured method of data mining, which is used for mining XML data. The approaches to graph-based data mining could be divided into five groups based on: 1) greedy search; 2) inductive logic programming; 3) inductive database; 4) mathematical graph theory; 5) kernel function.

The mathematical graph theory based approaches are conceptually close to itemset mining and use practically the same terminology and the same search concepts. As in itemset mining, the frequent subgraphs have a support higher than a threshold *s.* The key measure is the (minimal) support for frequent subgraphs. Mining is based on candidate generation using the downward closure property.

Algorithms from this class include *e.g*. AGM (Apriori-based Graph Mining) [7], FSG (Frequent SubGraph Discovery) [8], gSpan (graph-based Substructure pattern mining) [9].

AGM

AGM (Apriori-based Graph Mining) similarly to the APriori algorithm starts from frequent one-vertex graphs and generates candidate graphs of larger sizes by pairwise joining of frequent subgraphs that satisfy the following two conditions:

1) The frequent sub-graphs $G(X_k)$ and $G(Y_k)$ to be joined must consist of *k* vertices with same elements except for those in the *k*-th row and *k*-th columns of their matrices X_k and Y_k:

$$\mathbf{X_k} = \begin{pmatrix} X_{k-1} & x_1 \\ x_2^T & 0 \end{pmatrix}, \qquad \mathbf{Y_k} = \begin{pmatrix} X_{k-1} & y_1 \\ y_2^T & 0 \end{pmatrix} \quad (1.8)$$

If this is happens, the graphs are added and a new graph $G(Z_{k+1})$ is organized with matrix

$$\mathbf{Z_{k+1}} = \begin{pmatrix} X_{k-1} & x_1 & y_1 \\ x_2^T & 0 & z_{k,k+1} \\ y_2^T & z_{k+1,k} & 0 \end{pmatrix} \qquad (1.9)$$

where $z_{k,k+1}$ and $z_{k+1,k}$ represent an edge label between the *k*-th vertices of X_k and Y_k. One and the same graph can be obtained by changing the order of X_k and Y_k. T In order to avoid redundancy, a second condition is required.

2) The two graphs may be added if and only if

code (first matrix) $\leq$ *code*(second matrix)

where *code*(*g*) is invariant representation of the graph *g*.

AGM algorithm implements a complete search and finds all frequent subgraphs. It as well processes labeled vertices and edges, that requires insertion of a special node in the place of each edge label. For a dense graph, this necessity results in a graph much larger than the original one.

FSG

FSG (Frequent SubGraph Discovery) is similar to the AGM algorithm. The difference here is that FSG uses graph vertex invariants (degree of the vertices and the labels of the vertices and edges), and similarly to the AprioriTID algorithm, keeps transaction id-s (TIDs).

Using TIDs, FSG keeps for every frequent subgraph a list of the transaction identifiers that support it. When the intersection of the TID lists of $G(X_k)$ and $G(Y_k)$ is shorter than the minimal support, the new candidate graph created by joining $G(X_k)$ and $G(Y_k)$ couldn't be frequent and can be deleted from the list of candidate graphs to be checked for frequency. The last happens before calculating the costly subgraph-graph isomorphism: Otherwise its frequency is computed by using a subgraph isomorphism algorithm on the limited search space determined by the intersection of the TID-lists. Using this strategy, in contrast to AGM, FSG handles labeled vertices and edges without a modification of the graph, achieving in this way higher efficiency.

FSG fulfils a complete search and finds all frequent subgraphs.

gSpan

gSpan (Graph-based substructure pattern mining) is based on a depth-first search (DFS) and canonical labeling. In order to generate an invariant code for the graphs, gSpan uses a tree representation instead of a matrix one as in AGM and FSG. The frequent subgraphs are searched beginning with the frequent one-edge graphs and expanding them by one vertex at each step. gSpan relies on the coding technique - DFS codes to

encode and search the graphs. By applying DFS coding and DFS search, gSpan can derive complete sets of frequent subgraphs over a given minimal support *s* in a very efficient manner in both computational time and memory consumption.
The graph mining is applied, where frequent subgraphs are mined as part of an object-class recognition.

FREQUENT ITEMSET MINING IN VISUAL DATA

Detection of patterns in data is probably the most important task in computer vision. It first appears at the lowest stages of a recognition pipeline (feature extraction), and reappears at every higher stage of the system. Detection of *repeating* patterns allows valuable insights from the data to be gained.
A mining algorithm simply digs through data without a previously defined goal on what to look for (*e.g*. find the most important objects in a video sequence). Data mining puts emphasis on descriptive models and patterns for existing data, and on handling large datasets. The latter is the main motivation to borrow techniques from data mining and apply them to visual data. Itemset mining has been used successfully in a variety of domains to detect repeating combinations of items efficiently. Also, a bag of visual words (quantized local appearance features) can be described as a set.
These algorithms are used for finding repeating patterns in existing data and as predictor for unseen data.

Mining Specific Objects in Video

Mining interesting objects and scenes from video data means detecting frequently occurring objects automatically. It is useful as first step for retrieval and browsing and as a basis for video summarization.
The approach to video data mining is based on the detection of repeating spatial arrangements of local features. The input to the mining algorithm consists of subsets of feature-codebook entries for each video frame, encoded into "transactions" [1]. Information on spatial arrangement of features in transactions is included. In a transaction also is included information on how to select the neighborhood defining the subset of image features. For scenes with significant motion, the neighborhood is defined via motion segmentation. A simple and very fast technique for motion segmentation on feature codebooks is introduced.

Shot Detection, Features and Visual Words

The main processing stages of the considering system rely on the prior subdivision of the video into shots. A shot partitioning algorithm is applied [10], and four "keyframes" per second within each shot are picked. In each keyframe two types of affine covariant features – *regions* – are extracted. These are Hessian-Affine [11] and MSER [12]. Affine covariant features provide robustness against viewpoint changes. Each normalized region is described with a SIFT-descriptor [13]. Further, a visual vocabulary is constructed by clustering the SIFT descriptors with an optimized hierarchical agglomerative technique [14]. In a video this results in about 8000 appearance clusters for each feature type. As a final polishing, the 'stop-list' method known from text-retrieval is applied [15]. It means that very frequent and very rare visual words are removed from the codebook (5% most and 5% least frequent). The processing stages use only the spatial location of features and their assigned appearance-codebook id's. The appearance descriptors are no longer needed.

Video Mining Approach

The goal is to find frequent spatial configurations of visual words in video scenes. If a configuration is just an unordered set of visual words, for a codebook of size *d* there are 2*d* possible subsets of visual words. For each of two feature types there is a codebook with about 8000 words, which means *d* is typically > 10000, needing large search space. Frequent itemset mining methods are capable of dealing with such a large dataset and return frequently occurring word combinations. They solve analogous problems for other kinds of data.

Incorporating Spatial Information

In visual data the items correspond to visual words. A transaction is created for a frame, or around each feature and consists of an orderles bag of all other words within some neighborhood in the image.
Algorithm includes also spatial information via spatial locations of visual words in the mining process. It is encoded directly in the items. In each image transactions from the neighborhood around a limited subset of selected words $\{v_c\}$ are created. These words must appear in at least f_{min} and at most in f_{max} frames. Each v_c must also have a matching word in the previous frame, if both frames are from the same shot. Usually with these restrictions about 1⁄4 of the regions in a frame are selected (Fig.2.1).
For each v_c a transaction is created which contains the surrounding *k* nearest words together with their

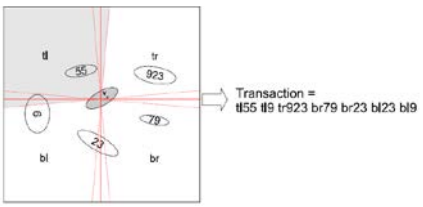

Figure 2.1.Creating transaction from a neighborhood. The label (tl, tr, bl, br)of each section is appended to the visual word id's. The transaction for v_c is T = {tl55, tl9, tr923, br79, br23, bl23, bl9}

rough spatial arrangement. The neighborhood around v_c is divided into *B* sections. With *B* = 4 each section covers 90 ± plus an overlap *o* = 5 ± with its neighboring sections. This leads to robustness against small rotations.
The approach works for small rotations. The overlap *o* adapts well rotations of the neighborhood coming from perspective transformations. Augmenting the items in this way increases their total number by a factor *B* without changes of the frequent itemset mining algorithm. When the central visual words v_c are carefully selected, the number of transactions and the runtime of the algorithm are reduced.

Exploiting Motion

Because a rough segmentation of the scene into object candidates is obtained, the neighborhood for

a transaction can be restricted to the segmented area for each candidate. As the central visual words v_c, the two closest regions to the center of the segmented image area are picked. All other visual words inside the segmented area are included in the transaction (Figure 2.2). This simplifies the task of the mining algorithm.

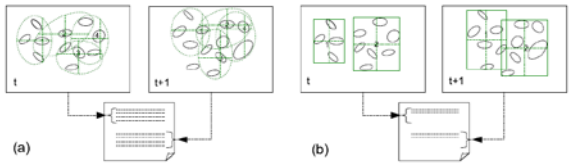

Figure 2.2.Creating transactions: (a) static shots: transactions are formed around each v_c from the k-neighborhood. (b) shots with considerable motion: a motion group is the basis for a transaction, the number of items in a transaction is not fixed but given by the size of the motion group.

The motion segmentation algorithm to find such object candidates lies on the assumption that groups of visual words which translate consistently from frame to frame can be identified. This is done by two steps:

Step 1. Matching words. A pair of words from two frames $f(t)$, $f(t+n)$ at times t and $t + n$ is considered matched if they have the same visual word id's, and if the translation is below a maximum translation threshold t_{max}. This matching step is extremely fast, since it uses only cluster id correspondences.

Step 2. Translation clustering. At each timestep t, the pairs of regions matched between frames $f(t)$ and $f(t+n)$ are grouped according to their translation using k-means clustering. The initial number of motion groups k is found with a leader initialization of k-means [16], on the translation between the first two frames. For each remaining timestep, k-means is run three times with different values for k

$$k(t) \in \{k(t-1)-1, k(t-1), k(t-1)+1\}, k(t) \in [2,...,6] \quad (2.1)$$

where $k(t-1)$ is the number of motion groups in the previous timestep.

This prevents the number of motion groups from changing unexpectedly from frame to frame. To improve stability, the algorithm is run twice for each k with different random initializations. The clustering with better mean silhouette value [17] is kept. For further improving of the quality of the motion groups for each motion group, a series of bounding boxes is estimated, that contains as minimum 80% of regions closest to the spatial median of the group. The bounding box with the maximal density *number of regions bounding box area* is used further. This procedure removes from the motion groups regions located far from most of the others. The closest two visual words to the bounding box center are now selected as the central visual word v_c for the motion group.

Mining an Entire Video

Choosing a Support Threshold

The choice of a good minimal support threshold s in frequent itemset mining is not easy, especially in untraditional setting where items and itemsets are constructed without supervision. If the threshold is too high, no frequent itemsets are mined. If it is too low, too many are mined. In order to define a fixed threshold, the algorithm with several thresholds is run, until the number of frequent itemsets falls within a reasonable range (usually from 100 to 100000 sets). A binary split search strategy is helpful. Two extremal support thresholds are defined, s_{low} and s_{high}. Let n be the number of itemsets mined in the current step of the search, and s be the corresponding support threshold. If the number of itemsets is not in the desired range, the following rule is used for updating s:

$$s^{(t+1)} = \begin{cases} s^{(t)} + \dfrac{s_{high} - s^{(t)}}{2}, & s_{low} = s^{(t)}, \quad n > n_{\max} \\ s^{(t)} - \dfrac{s^{(t)} - s_{low}}{2}, & s_{high} = s^{(t)}, \quad n < n_{\min} \end{cases} \quad (2.2)$$

and rerun the mining.

Finding Interesting Itemsets

The output of the APriori algorithm is usually a rather large set of frequent itemsets, depending on the minimal support threshold. Finding *interesting* itemsets (and association rules) is a much discussed topic in the data mining literature. Itemsets with statistical dependence items are interesting. The measure (1.7) defined in paragraph 1.3 is used to improve the quality of the mining results.

Itemset Clustering

Since the frequent itemset mining typically returns spatially and temporally overlapping itemsets, they are merged with final clustering stage. Pairs of itemsets which jointly appear in more than F frames and share more than R regions are merged. Merging starts from the pair with the highest sum $R + F$. If any of the two itemsets in a pair is already part of a cluster, the other itemset is also added to that cluster. Otherwise, a new cluster is created.

TABLE 2.1
MOTION SEGMENTATION AND 40-NN MINING METHODS COMPARED

Method	Regions	Transactions	Time of FIM	Support threshold	FI	FIF (n_s)	Clusters (F,R)
M-Seg.	$2.87*10^6$	8056	56.12s	0.015	27654	308 (2)	11 (2,2)
40-NN	$2.87*10^6$	511626	18.79s	0.0001	285	285 (0)	55 (2,2)

In Table 2.1 it is compared quantitatively mining with motion segmentation, and with a fixed 40-neighborhood for a clip. There are 8056 transactions when using motion segmentation, compared versus more than half a million when using a fixed 40-neighborhood. The runtime is very short for both cases, 40-NN method is faster, because transactions are shorter and only shorter itemsets were frequent. The mean time for performing motion segmentation matching depends on the number of features detected per frame. Hence, the mining approach based on frequent itemsets is a suitable and efficient tool for video mining. Restricting the neighborhood by motion

grouping is useful for detecting objects of different sizes at the same time.

Mining Frequent Feature Configurations

The considered approach for mining frequently occurring objects from video data quantifies local features and outputs specific objects (person or scene). Local features are helpful for object *class* detection. The local feature extractor is run without prior knowledge of the object class. For a typical image it returns a large number of features, of which only some fraction lie on the object of interest. This complicates the object detectors and other higher-level processes, as they have to find their way to the object through many background features.

A mining-based method to filter this large mass of features can be applied. It selects features which have high probability of lying on instances of the object class of interest. This technique is as an intermediate layer between feature extraction and object class detection.

The filtered set of features can then be fed into a higher-level object detector. Thanks to this, it performance is improved. The method leads to lower false-positive rates and to higher detection rates. Starting from a cleaner set of features facilitates segmenting objects from the background, or determining their pose.

The method's input is a set of positive training images, containing different instances of the object class, and a set of negative background images. The local features are organized in semi-local neighborhoods in a way suitable for data mining. Further, Frequent Itemset Mining is applied on the large set of all neighborhoods and the output is spatial configurations of local features frequently re-occurring over the training images. From these configurations Association Rules are collected. These rules hint the presence of the object in positive images with high confidence and reject only rarely on background images.

In case of a novel image, first the mined configurations is matched to it, and then a confidence value to each feature is associated that expresses how it should is to lie on an instance of the object class. This is obtained by accumulating the activation scores of all matched configurations involving the feature.

This approach has several advantages. The mining algorithm is designed for scalability and allows to process large training sets rapidly. The set of rules collected from the data in this fashion are distinguished and easy to interpret. The spatial configurations of neighboring features give higher discriminative power compared to individual features. In addition, the rules often catch configurations of local features corresponding to semantic object parts. The per-feature confidence values produced by this approach effectively prune away the majority of background features and help subsequent object detectors.

Unlike in the video mining presented above, there are no motion hints, and thus the mining doesn't rely on a motion segmentation to identify neighborhoods to create transactions. Hence, this is an extended and refined method for including spatial arrangement of features in the itemset mining process, which also works for unsorted images containing instances of an object class, instead of an ordered sequence of images showing a specific object.

Frequent Feature Configurations

The proposed technique for mining frequent feature configurations can be summarized as follows. The training set is composed of positive images and negative images. The positive images contain object instances annotated by a bounding-box. The negative images do not contain any instance of the class of interest.

First step: A large number of spatial configurations of local image features are collected from all training images. A mining algorithm is then used to select frequently occurring configurations from this large set.

Second step: These configurations are transformed into association rules These rules select frequent configurations with presence of the object class with high confidence and rare occurring on the negative images or on non-object areas of the positive images. These *discriminative rules* are the building blocks for a generating class specific confidence values for features of novel images.

Third step: The itemset mining algorithm as described above for video mining is run. Association rules from the mined itemsets are also formed. They have two desirable properties: 1) they can be extracted even from very large bodies of data; 2) the rule notation is easily interpretable and can be used to gain global insights into large datasets or can be analyzed by experts. These properties lead to their application in web usage mining [18].

The lowest layer of the system is again built on a set of local features extracted in each image. Difference of Gaussian (DoG) detector is used to extract regions and the SIFT descriptor [13] is used to describe their appearance. The SIFT feature vectors are clustered into a visual vocabulary with hierarchical agglomerative clustering.

Because of the uncertainty of the unsupervised clustering process, each feature is assigned to all codebook clusters whose center c is closer than a distance threshold d_{min}. Thus, a description of each region R_i by a set of codebook labels is obtained

$$\zeta_i = \{c_j \mid d(R, c_j) < d_{\min}, \quad j \in 1,...,N\} \qquad (2.3)$$

where N is the total number of appearance clusters.

Neighborhood-based Image Description

The second layer of the system constructs an image representation from the codebook labels. The simplest representation would be a global histogram (*bag of features*). The aim is unsupervised mining and learning useful representations for object classes. That is why, a more informative description is necessary. It is achieved by encoding not only the presence of visual words, but also their spatial arrangement leading to a much stronger descriptor. Thus, image is described as a set of semi-local neighborhoods.

Further follows sampling spatial neighborhoods from an image. The locations R_c of the neighborhood centers are defined by sampling of

the feature extractor [19] and the size of the neighborhood is defined on the scale of the central region R_c [15]. More precisely, all regions falling within a square of side proportional to the scale of R_c are inside the neighborhood. Each neighborhood is split into Q tiles. For each tile an activation vector indicating which visual words it contains is created. Multiple occurrences are not counted of the same visual word in a particular tile. The resulting Q activation vectors are connected to form the neighborhood descriptor: a (N * Q)-dimensional sparse binary vector.

The activation vector can equivalently be written as a *transaction* (figure 2.3.c). The neighborhood description is a generalized version of the neighborhood with only 4 tiles. Since a neighborhood for every region in every training image is formed, this results in a very large number of neighborhoods (or transactions). Itemset mining can handle these amounts of data easily through a recent parallel implementation of FP-Growth [4].

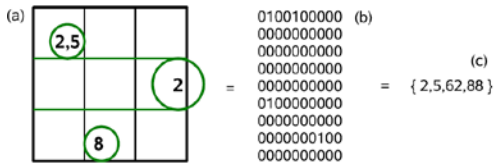

Figure 2.3.(a) A neighborhood with 9 tiles and 10 appearance clusters. (b) Activation vector. (c) Transaction.

Mining Frequent and Distinct Configurations

The introduced tools are applied for mining *distinctive* configurations appearing frequently on the object and rarely on the background.

As discussed above, generating a transaction each neighborhood is described by a list of non-zero indices. The input to the mining algorithm is the database containing all transactions. In order to differentiate the object against background data, transactions from the negative training set to the database are added. All transactions originating from instances of the object class are assigned the label "object" as an additional item "background" appended the item to background transactions. For the neighborhood in figure 3.10 it is {2, 5, 62, 88, *object*}.

The APriori algorithm on the transaction database is than run in order to mine frequent itemsets and association rules. The resulting rules are filtered to keep only those which hint the object label with high confidence, *i.e*.

$$conf(C \rightarrow object) > conf_{\min} \qquad (2.4)$$

where C is a frequent configuration and $conf_{min}$ is a confidence threshold. The frequent itemset mining approach finds frequent *and* distinctive feature configurations. These prototypical configurations are found efficiently from the large search space of all 2^{N*Q} possible configurations. As additional advantage, many of the mined rules have semantic qualities.

Class-specific Feature Confidence

The frequent feature configurations C mined from the neighborhoods in the training images represent frequent and distinguished fragments of an object class. They describe neighborhoods characteristic for the object class.

For a new test image, the mined configurations can be matched to it, and hence discover features lying on instances of the object class. To achieve this, all neighborhoods P of the new image (one for each region) are firstly generated. Every mined configuration C is now matched to each image neighborhood P.

A configuration can be written as a sparse activation vector. Hence, the test image neighborhoods can be matched efficiently by a sparse dot product

$$m(C,P) = \begin{cases} 1 & if \quad C * P = |C| \\ 0 & if \quad C * P \neq |C| \end{cases} \qquad (2.5)$$

where $|C|$ is the number of features in C, and $m(C,P)$ = 1 indicates a match. In other words, a frequent configuration C matches a candidate neighborhood P if their dot product equals the number of visual words in C.

From matched neighborhoods of the test image a measure of the probability for a feature whether it lies on an instance of the object class can be derived. This measure effectively enables to pre-select features lying on the object and facilitates a subsequent object detector. This class-specific feature confidence measure for each feature R_i is defined as:

$$conf(R_i) = \frac{1}{M * W} \sum_{c} \sum_{\{P | R \in P\}} \frac{1}{k} * m(C,P) \qquad (2.6)$$

where M is the number of configurations mined on the training data, W is the number of neighborhoods in the test image, k is the number of appearance clusters to which R_i was soft-assigned (eq. (2.2)).

Quantitative Evaluation of Feature Selection

The performance of the method for assigning class-specific confidences to features is quantified in two ways. The first measure is *bounding box hit rate* (BBHR) over the positive test sets that is the number of *bounding box hit* (BBH) divided by the total number of object instances in the positive test set. To perform this evaluation we use ground-truth bounding-box annotations available for the test images. The relation of BBHR with the false positive rate (FPR) as well is considered. This is the number of selected features lying outside the bounding box, divided by the total number of selected features in the image. *i.e*. it delivers the proportion of irrelevant features (the lower the better).

The method is compared against a baseline, where the confidence for a feature is computed. For each visual word in the codebook the number of appearances inside the bounding-box annotations of the training data is counted. This way a visual word, which appears often on the annotated training objects is weighted higher.

On a test image, features to the codebook are matched and BBHR is defined by summing the weighted matches for each feature. This allows the considered method to be compared to the default input to an object recognition system.

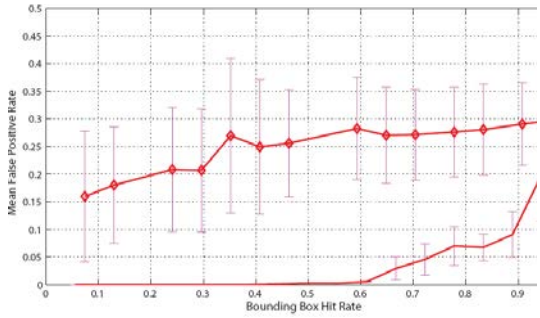

Figure 2.4.Bounding box hit rate (lower is better, baseline with diamond marker).

Figure 2.4 shows FPR on the y-axis and BBHR on the x-axis, for k = 5. The error bars show the standard deviation of the FPR at a given BBHR. The curve is generated by varying the selection threshold over the feature confidence. As the plots show, the feature selection method is very precise and delivers a low FPR (below 20% to a moderate 35% for high BBHR). This is an important characteristic, because it enables later processes to rely on a clean input, composed of a large majority of features on the object. This is especially worth when compared to the low signal-to-noise ratio of the initially extracted features (there are typically 500 − 1000 features in an image, out of which about 10 − 200 lie on the object). The experiments also reveal performance improvement over the baseline.

The feature selection ability comes at a low price concerning missed objects. The method selects at least 5 features (typically many more) on about 90% of the object instances. A lower BBHR might appear at a very high support threshold for mining or at a bad visual vocabulary.

The method is evaluated as well on the negative test sets (*i.e*. on image without any instance of the object class). The idea is to measure how distinctive the method is: does it select few features on negative images? This is relevant because the number of features selected on negative images relates to the computational resources the later processing stages will consume on irrelevant data.

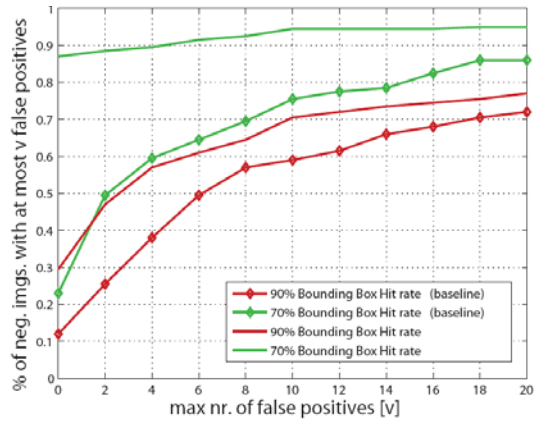

Figure 2.5.False positives on negative test images (higher is better).

Figure 2.5 reports the percentage of negative images (y-axis) where at most v features are selected (x-axis). The feature selection threshold is left fixed for ths curve, to the one giving 70% to 90% BBHR on the positive dataset (sensible operating point). As the plot shows, at 70% BBHR the method returns very few features on the negative images (on 90% of the images it returns less than 3 features). As it can be expected, at the operating point of 90% BBHR the method returns more features, but it remains distinctive even in this case: 1 in 3 negative images have no selected features, and 70% of the images have less than 10 (starting from 500 − 1000). The baseline is evaluated in the same manner as for the BBHR plots, and it performs worse than the proposed method.

Computation times

The computation times are given in Table 3.2. The time is measured for the frequent itemset mining stage including rule creation after feature extraction and neighborhoods construction. This is because the required processing can be done offline and the required time scales linearly with the number of images. For the mining the APriori algorithm is used. All experiments were done on a 3 GHz Intel Pentium 4 with 1*GB* RAM. The measurements show the scalability of the mining approach. The most feature configurations are extracted from tens of thousands of candidates in about a second. The mined configurations might be used readily within other frameworks. Table 2.2. also gives the used mining parameters.

In summary, the experimental evaluation demonstrates that the class specific confidence measure acts as a good feature selector. Hence, the considered technique offers a valuable intermediate layer between feature extraction and object detection or other higher-level processes.

TABLE 2.1
STATISTICS FOR THE MINING EXPERIMENTS

Transactions	$supp_{min}$ / $conf_{min}$	number of tiles	Comp. time (*s*)
26054	0.20% / 100%	9	2.58 *s*
42390	0.25% / 95%	9	0.91 *s*
29001	0.28% / 100%	9	0.90 *s*
74296	0.10% / 90%	9	53.02 *s*

CONCLUSION

In the work itemset mining methods on object recognition tasks are applied. The visual words allow to encode an image as a set of items. Based on this idea, methods for video mining are derived, for mining of feature configurations for object class recognition, and also evaluated alternative graph mining methods. Proposed methods do not consider local features as an orderless set. They add an extra processing layer which encodes the spatial relationships prior to mining. This results in datasets with up to millions of items. The work shows that itemset mining offers a suitable method to handle these large amounts of data and to find frequent patterns efficiently. The method is able to mine the most frequently occurring objects in video clips. The method for mining frequent feature configurations was evaluated for object class recognition. The mined configurations of features have higher discriminative power than individual features. The mined itemsets and rules exhibit the same good properties as in other fields: they can be analyzed and are easily interpretable by humans.

One possible extension of the method would be to make matching of already mined configurations in novel images even more efficient using itemset mining algorithms such as FP-Growth.

Only the first few steps with itemset mining in databases of visual words have been presented. Mining is well combined with learning a model and simple exemplar based approaches. The data is efficiently analyzed for the most essential patterns, neglecting irrelevant information. The latter directly leads to the availability of huge datasets. Enormous amounts of data (*e.g*. on the Internet) might lead to approaches which do not require any models any more, but efficient analysis of the data.

BIBLIOGRAPHY:

[1] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases", *SIGMOD'93*, 1993.
[2] R. Cooley, J. Srivastava, and B. Mobasher, "Web mining: Information and pattern discovery on the world wide web", *ICTAI*, 1993.
[3] A. Metwally, D. Agrawal, and A. Abbadi, "Using association rules for fraud detection in web advertising networks", *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, 2005.
[4] H. Li, Y.Wang, D. Zhang, M. Zhang, and E. Y. Chang, "Pfp: Parallel fp-growth for query recommendation", *ACM Recommendation Systems 08*, 2008.
[5] D.J. Hand, *Principles of Data Mining*. MIT Press, 2001.
[6] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000.
[7] A. Inokuchi, T. Washio, and H. Motoda, "Complete mining of frequent patterns from graphs: Mining graph data", *Machine Learning*, vol. 50, pp.321—354, 2003.
[8] M. Kuramochi and G. Karypis, "Frequent subgraph discovery", *ICDM'01: 1st IEEE Conf. Data Mining*, pp. 313–320, 2001.
[9] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining", *ICDM '02: Proceedings of the 2002 IEEE ICDM'02*, 2002.
[10] M. Osian and L. Van Gool, "Video shot characterization", *Machine Vision Applications*, vol. 15 (3), pp. 172–177, 2004.
[11] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors", *IJCV*, vol. 60(1), pp. 63–86, 2004.
[12] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide-baseline stereo from maximally stable extremal regions", *BMVC'02*, 2002.
[13] D. Lowe, "Distinctive image features from scale-invariant keypoints". *IJCV*, vol. 60 (2), 2004.
[14] B. Leibe and B. Schiele, "Interleaved object categorization and segmentation", *BMVC'03*, 2003.
[15] J. Sivic and A. Zisserman, "Video data mining using configurations of viewpoint invariant regions", *CVPR'04*, 2004.
[16] A. Webb, *Statistical Pattern Recognition*. Wiley, second edition, 2002.
[17] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, 1990.
[18] R. Cooley, J. Srivastava, and B. Mobasher, "Web mining: Information and pattern discovery on the world wide web", *ICTAI*, 1993.
[19] N. Dalal and B, "Triggs. Histograms of oriented gradients for human detection, *CVPR'05*, 2005.