A Motion-Flow-Based Fast Video Retrieval System

Chih-Wen Su
Dept. of Comp. Sci. and Info. Eng.
National Central University
Taiwan
lucas@iis.sinica.edu.tw

Hong-Yuan Mark Liao †
Inst. of Info. Sci.
Academia Sinica
Taiwan
liao@iis.sinica.edu.tw

Kuo-Chin Fan
Dept. of Comp. Sci. and Info. Eng.
National Central University
Taiwan

ABSTRACT
In this paper, we propose the use of motion vectors embedded in MPEG bitstreams to generate so-called "motion-flows", which are applied to perform quick video retrieval. By using the motion vectors directly, we do not need to consider the shape of a moving object and its corresponding trajectory. Instead, we simply "link" the local motion vectors across consecutive video frames to form motion flows, which are then annotated and stored in a video database. In the video retrieval phase, we propose a coarse-to-fine strategy to execute the video retrieval task. Motions that do not belong to the main-stream motion flows are filtered out by our proposed algorithm. The retrieval process can be triggered by query-by-sketch (QBS) or query-by-example (QBE). The experiment results show that our method is indeed efficient and accurate in the video retrieval process.

1. INTRODUCTION
The development of video technology in recent years, has become a very important research field. Considering the rapid increase in digital video content, an efficient way to access and manipulate the information in a vast database has become challenging and timely issue. Some commercial search engines; such as Google [19] and Yahoo! [20], have started to extend their services to video searching on the Internet, and it is already possible to search for video clips by typing keywords. However, commonly adopted features, such as color, texture, or motion are still insufficient to describe the rich visual content of a video clip. In the past few years, the area of content-based multimedia retrieval has attracted worldwide attention. Some experimental systems, such as QBIC [1], Virage [9], PhotoBook [7], VisualSeek [8], Video-Q [2], and Netra-V [10], have successfully applied semantic-based visual content to multimedia database retrieval. Among the different types of features used in previous content-based video retrieval (CBVR) systems, the motion feature has played a very important role. A motion-based feature can be further processed into a feature that can simultaneously cover spatial and temporal characteristics. Such a feature has a better chance of effectively executing video retrieval tasks.

In the literature, a large number of motion-based video retrieval systems have been proposed in the past decade [1, 2, 5, 6, 11, 12, 13]. VideoQ [2] is a notable system that directly addresses motion-based characterization of video content. In fact, most existing CBVR systems do utilize motion as one of the features when executing a search process. The major difference among these systems is the way they extract and manage a motion-based feature in the video retrieval process. We can classify existing motion descriptors into two types: (1) statistics-based, and (2) object-based. Researchers who work with the first type use statistics to analyze the tendency and distribution of local motion. For example, Fablet et al. [6] used causal Gibbs models to represent the spatio-temporal distribution of the dynamic content in a video shot. Ma and Zhang [11] generated a multidimensional vector by measuring the energy distribution of a motion vector field. In MPEG-7 [13], some fundamental bases of motion activity, such as intensity, direction, and spatial-temporal distribution, are adopted to retrieve content from video databases. With regard to object-based motion descriptors, Chang et al. [2] proposed grouping regions that are similar in color, texture, shape, and motion

Categories and Subject Descriptors
H.3 [Information Storage and Retrieval]: Content Analysis and Indexing

General Terms
Experimentation

Keywords
motion analysis, video retrieval

† Corresponding author. Tel.: +886-2-27883799 ext.1811; fax: +886-2-27893309.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyright 2005 ACM 1-59593-244-5/05/0011 ...$5.00.
The formation of a trajectory is hard to express precisely, we only that contain similar motion content. Since the temporal in-
based interface, and the system will retrieve a set of shots
a specific motion, he/she can draw a trajectory on a sketch-
motion flows in the database. When a user wants to query
ally, we remove redundant motion flows and save the main
generate continual motion information as a trajectory. Fi-
tional domain, we construct motion flows from the vectors to
several independent moving parts.
An object-based motion descriptor needs to record the
trajectory of a video object; therefore, segmentation of video
objects is indispensable to the process. However, it is well
known that the segmentation of video objects is an ill-posed
problem. As a result, it is impossible to “automatically”
and always “correctly” derive the centroids of video objects
across consecutive frames. On the other hand, it is not fea-
sible to calculate the above mentioned trajectory, because
the process is very time-consuming. For some real world
applications such as web searches, video annotation must
be efficient and precise, because its results can be used to
make a lot of comparisons. Therefore, we propose a new
approach that improves the CBVR process and makes it
as accurate and efficient as possible. Since most stored
videos are compressed, we directly utilize the motion vec-
tors embedded in an MPEG bitstream to develop a motion
descriptor. It is known that motion vectors only record the
direction and magnitude of movement between correspond-
ing macroblocks of two consecutive anchor frames, so they
are only comprised of local data that does not have much
semantic meaning. Here, we propose to make use of the
consistency of motion direction, the color distribution, and
the overlapping area between macroblocks of two consecu-
tive frames, to “link” all neighboring motion vectors. These
linked motion vectors form the so-called motion-flows, which
contain more semantic meaning than the original motion
vector data. Since a large moving object may occupy several
macroblocks and produce multiple motion-flows, we propose
an algorithm to reduce the motion-flows that are similar in
shape to one or several representative motion flows. We
approximate these representative motion flows by generat-
ing control points and storing them in a database as models.
Since there are no computationally expensive processes such
as object segmentation and tracking involved, our method
is more efficient than conventional object-trajectory-based
motion descriptors. In addition, our approach allows multi-
ple representative motion flows for an object if it contains
several independent moving parts.
Fig. 1 shows the procedure of our proposed approach. We
first segment videos into shots, eliminate global camera mo-
tion, and then extract the residual local motion vectors in
each frame. Since motion vectors are dispersed in the spa-
tial domain, we construct motion flows from the vectors to
generate continual motion information as a trajectory. Fi-
ally, we remove redundant motion flows and save the main
motion flows in the database. When a user wants to query
a specific motion, he/she can draw a trajectory on a sketch-
based interface, and the system will retrieve a set of shots
that contain similar motion content. Since the temporal in-
formation of a trajectory is hard to express precisely, we only
consider the spatial information in a QBS process. In the
initial stage, a user can execute a QBS process to retrieve
some candidate shots from the database. Then, he/she can
choose a candidate shot and execute another QBE process
to further extract the video clips that are most similar to
the query.
The remainder of this paper is organized as follows. In
Section 2, we describe how to construct motion flows from an
MPEG video. In Section 3, we introduce a novel comparison
algorithm that calculates the similarity degree between two
distinct motion flows. Finally, we present our experiment
results and conclusions in Sections 4 and 5, respectively.

2. CONSTRUCT MOTION FLOWS FROM
MPEG BITSTREAMS
In this section, we introduce the method for construct-
ing motion flows from an MPEG bitstream. As mentioned
in the previous section, numerous methods [4, 5, 6] have
been proposed for annotating the motion information in a
video. Among these methods, the trajectory-based ap-
proach is probably the most popular. However, its unstable
nature and high computation cost have discouraged peo-
ple from choosing it as a representation/annotation tool.
Furthermore, a trajectory-based representation scheme only
takes the path formed by linking the centroids of a video ob-
jeet appearing across consecutive anchor frames. Therefore,
if a user wants to retrieve the motion contributed by part of
a video object, the trajectory-based representation scheme
can not provide a correct retrieval result. In view of this, we
propose the direct usage of the motion vectors originally em-
bbeded in an MPEG bitstream to construct “motion flows”
in a shot. Although the motion vectors do not always cor-
respond to the “real motion” of the objects in a video as
compared with the optical flow, they are relatively easy to
derive.
Before constructing the motion flows, we have to per-
form some preprocessing steps so that camera motion can
be compensated for. First, we produce a motion vector field
between the last P-frame in the current GOP and the I-
frame in the next GOP by using the B-frames between the
P- and I-frame as proposed in [4]. This yields a forward
reference motion vector field between any two consecutive
anchor-frames (I- and P- frames). It is well known that a
motion vector field is usually composed of camera motion,
object motion, and noises. We assume that the global mo-
tion in a video is mostly contributed by camera motion. As
a result, we use the following four-parameter global motion
model, which is fast and also valid for most videos [4], to
estimate the camera motion from the motion vector field.

\[
MV_{\text{cam}} = \begin{pmatrix} \text{zoom} \\ \text{rotate} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \text{pan} \\ \text{tilt} \end{pmatrix}.
\]
Once the four parameters of the camera’s motion are estimated, we can find the correspondence between each pixel in the current frame and its counterpart in the previous frame. The foreground pixels in the current frame can be identified if their color changes significantly after the camera’s motion. On the other hand, the information about local motion is also derivable if one subtracts the estimated camera motion from the original motion vector field. If several foreground pixels survive the erosion process within a macroblock, then the local motion corresponding to the macroblock can be regarded as robust. Otherwise, we ignore the local motion, because it is probably from the background.

After the camera motion is compensated, we can extract from a video the motion fields from the bitstreams. Each local motion field represents the movement of a macroblock between two consecutive anchor frames. However, this local motion field is not usually continuous in the spatio-temporal domain since the start positions of motion vectors always begin at regular places, but their end positions could be anywhere. Thus, when we try to make the necessary connections between two local motion fields, each macroblock in the current frame may overlap with four macroblocks in the previous frame, as shown in Fig.2. In other words, it is possible that several candidate local motions located in the previous frame could be connected with a local motion in the current frame. Therefore, we propose to use criteria in order to choose the most reliable candidate for the current local motion. The criteria are: (1) the consistency of motion direction, (2) the color distribution, and (3) the overlapping area between macroblocks in the previous and the current frames. Suppose $LM_{Cur}$ is a local motion and $C$ denotes the set of candidates of $LM_{Cur}$ in the previous anchor frame. The most reliable candidate, $LM_{ancestor}$, is chosen based on the following rule:

$$LM_{ancestor} = \arg\min_{m \in C} \|\{\theta_m, \alpha \phi_m, \beta \varphi_m\}\|,$$  \hspace{0.5cm} (2)

where $\theta$, $\phi$, and $\varphi$ denote the angle formed by $LM_{Cur}$ and possible candidate $m$, the color histogram difference between macroblocks, and the overlapping area between macroblocks, respectively. Furthermore, all the parameters are normalized so that all the values fall within the range $[0,1]$, and $\alpha$ and $\beta$ are weighting values that balance the influence of the three factors. Once the most reliable candidate of $LM_{Cur}$ is determined, we can link the local motions acquired at different time spots to form a single motion flow for each macroblock.

Since a large moving object may cover several similar motion flows, we remove the redundant ones and only preserve one or more representative motion flows from all the similar motion flows. Initially, we select the motion flow, $a$, that has the longest duration in a shot. Suppose $a$ starts from time $t_i$ and ends at time $t_j$. $\{B\}$ is the set of motion flows whose start and end times are both within the duration of $t_i$ and $t_j$. We remove a motion flow, $b$, belonging to $\{B\}$, if it satisfies the following condition:

$$\frac{\sum_{t \in [t_s, t_e]} \| (a_t - a_{t_s}) - (b_t - b_{t_e}) \|}{t_e - t_s} < \epsilon,$$  \hspace{0.5cm} (3)

where $t_s$ and $t_e$ are respectively the start and end times of motion flow $b$. $a_t$ and $b_t$ denote the spatial position of $a$ and $b$ at time $t$ respectively; and $a_{t_s}$ and $b_{t_e}$ denote the spatial position of $a$ and $b$ at time $t_s$ respectively. If the average of the relative spatial distance between $a$ and $b$ is smaller than a threshold, $\epsilon$, we consider $b$ to be a subsegment of $a$. As $b$ can be retrieved from $a$ by a partial matching process, it is redundant and can be removed. We then consider $a$ as the representative of the motion flow removed from $\{B\}$ and store it in the database. Next, we again select the motion flow with the longest duration from the remaining motion flows, and repeat the above process until all motion flows are either stored or removed. Here, we only consider motion flows whose durations are longer than 3 seconds in order to avoid the effects of noise and also to reduce the size of database.

Fig.3b shows different representative motion flows of the video clip shown in Fig.3a. The original motion flows (with $\epsilon = 0$) are shown on the left-hand side, and the motion flows after applying the removal process using different thresholds are shown in the middle and on the right-hand side, respectively. It is obvious that the redundant motion flows have been successfully removed, and that the number of representative motion flows is controlled by the value of $\epsilon$. Fig.4 shows another example of representative motion flows derived from a video sequence that contains multiple moving objects and global camera motion. Since the movement of each object is different, several representative motion flows are generated concurrently to represent the way the objects move. Although the extraction of motion flows is much faster and easier than deriving a trajectory; there still exist two problems when we construct motion flows from motion vectors. First, the occlusion of multiple moving objects can cause a macroblock to be intra-coded, which means we could miss the motion information. Thus, a complete motion flow for each moving object could not be derived. The region surrounded by dotted lines in Fig.4 illustrates the above situation. The sudden termination of motion flows is apparently caused by the occlusion of two football players. Second, since we remove the camera motion before extracting the local motion from a video, the motion flow will break off if a moving object stops temporarily.
3. COARSE-TO-FINE TRAJECTORY COMPARISON

Having constructed the motion flows from each shot, we propose a new algorithm that compares the degree of similarity between a query trajectory and the motion flows in a database. Since we are looking for “similar” clips in the database, some geometric transformation, such as scaling or translation, should be handled. Here, we do not consider rotation invariance because the direction, i.e., up-and-down or left-and-right, usually has a semantic meaning in a video. For example, if one wants to query a “jump” motion, the trajectory should start from a lower position, passing through a higher position and then return to the lower position. Also, the issue of partial matching needs to be handled if a user provides an incomplete query. In what follows, we propose a simple, but fast, algorithm for comparing two distinct trajectories (motion flows).

In order to reduce the time complexity and minimize the storage space, we remove redundant points from a trajectory leaving only a few necessary points to represent the trajectory. We choose a famous top-down method — the Douglas-Poiker algorithm [3] — to select the necessary control points from a trajectory. The algorithm starts by using a straight line segment (called the anchor line) to connect the start and end points of a trajectory. Once the perpendicular distance between any intermediate point to the anchor line is larger than a threshold, the trajectory is split into two segments via the farthest intermediate point. The process continues until all the perpendicular distances are smaller than a pre-set threshold. Finally, the chosen intermediate points and the two end points are reserved as the control points of the trajectory.

Traditionally, researchers have used \((x, y, t)\) to denote the position of a control point on a trajectory in the spatio-temporal domain. Unlike conventional approaches, we use six positive real numbers \((x^+, x^-, y^+, y^-, d, t)\) to represent a control point on a trajectory, where \(d\) denotes the cumulative length of the trajectory from the first control point to the current control point; and \(+/-\) denotes the cumulative positive/negative movement along the \(x\)- or \(y\)-axis from the first control point to the current control point. Now, let \(Q\) and \(D\) be the trajectories of the query and a model in the database, respectively. In a QBS case, we normalize the length of both trajectories into a unit length before a comparison is executed. This move guarantees the requirement of scale invariance. Therefore, the parameters \(d, x^+, x^-, y^+, y^-\) and \(y^-\) of each control point on the two trajectories have to be normalized by dividing them by the length of \(Q\) and \(D\), respectively.

According to the six parameters of a control point, \(d\) and \(t\) are utilized to make a fair comparison. We align both \(Q\) and \(D\) by calculating the length \(d\) or duration \(t\) from the first control point. For each control point on \(Q\) \((D)\), we interpolate a corresponding point that has the same cumulative length onto \(D\) \((Q)\). In a query-by-sketch (QBS) video retrieval system, the “\(d\)” value is used as a basis to conduct the alignment task, because we only consider the similarity between \(Q\) and \(D\) in the spatial domain, as shown in Fig.5a. On the other hand, the “\(t\)” value plays a crucial role when a video retrieval system incorporates the query-by-example (QBE) approach, as shown in Fig.5b. The control points and the corresponding points are labeled by circles and triangles, respectively. In this scenario, the insertion of corresponding points on \(Q\) and \(D\) is dependent on either “\(d\)” or “\(t\).” Now, for each control point on the trajectory \(Q\) \((D)\), we can interpolate a corresponding point located on \(D\) \((Q)\). Assume the total number of control points and their corresponding points located on \(Q\) and \(D\) are both \(N\). Let \(Q' : \{Q'_1, Q'_2, \ldots, Q'_N\}\) and \(D' : \{D'_1, D'_2, \ldots, D'_N\}\) be the set of points (including the control points and inserted cor-
responding points) located on Q and D, respectively. We call the points in set \( Q' \) and \( D' \) “check points”, each of which can be represented by \((x^i, x^j, y^i, y^j)\) in the spatial domain. In order to compare two arbitrary trajectories, we define a metric as follows:

\[
EstDist_{i,j}^{Q',D'} = \left| (Q'_j - Q'_i) - (D'_j - D'_i) \right| \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}, \tag{4}
\]

where \( i \) and \( j \) (\( i < j \)) denote the \( i \)-th and the \( j \)-th check point of two partial trajectories located on \( Q' \) and \( D' \), respectively. \( Q'_j - Q'_i \) and \( D'_j - D'_i \) represent the difference between the \( i \)-th and the \( j \)-th check points of \( Q' \) and \( D' \), respectively. \((Q'_j - Q'_i) - (D'_j - D'_i)\) is a \( 1 \times 4 \) vector and its subsequent part in Eq.4 is a \( 4 \times 2 \) matrix. We take the absolute value of each element of \((Q'_j - Q'_i) - (D'_j - D'_i)\) before the matrix operation is executed, because we would like to accumulate all movements including negative ones. Therefore, the operation on the right hand side of Eq.4 generates a \( 1 \times 2 \) vector. Also, \( EstDist_{i,j}^{Q',D'} \) is basically a rough estimation of the distance between two partial trajectories (each trajectory runs from the \( i \)-th check point to the \( j \)-th check point) located on \( Q' \) and \( D' \), respectively. With the above distance metric, we can define the total distance metric between \( Q \) and \( D \) as follows:

\[
Dist(Q, D) = \sum_{i-1}^{N-1} \left\| EstDist_{i,i+1}^{Q',D'} \right\|. \tag{5}
\]

The norm defined in Eq.5 is an \( L_2 \)-norm.

The advantage of the proposed representation scheme is that we do not really need to compare the check points pair by pair. One thing to be noted, however, is that all the elements that form the vector of a check point on a trajectory are positive and their magnitudes are respectively accumulated from the very beginning. Therefore, if we choose an intermediate check point, \( Q'_k \), in \( Q' \) and its corresponding check point, \( D'_k \), in \( D' \), we are positive and their magnitudes are respectively accumulated from the very beginning. Therefore, if we choose an intermediate check point, \( Q'_k \), in \( Q' \) and its corresponding check point, \( D'_k \), in \( D' \), we are sure that

\[
\left\| EstDist_{i,j}^{Q',D'} \right\| \leq \left\| EstDist_{i,k}^{Q',D'} \right\| + \left\| EstDist_{k,j}^{Q',D'} \right\|. \tag{6}
\]

Eq.6 tells us that a coarse-to-fine search strategy is feasible for a trajectory-based query. In the first step of the comparison between \( Q' \) and \( D' \), we simply check the value of \( EstDist_{1,N}^{Q',D'} \). This step only needs to consider four check points \( Q'_1, D'_1, Q'_N, \) and \( D'_N \). Since the value of \( Dist(Q, D) \) must be equal to or larger than that of \( EstDist_{1,N}^{Q',D'} \), we can quickly determine that trajectory \( D \) is not similar to \( Q \) if the returned value of \( EstDist_{1,N}^{Q',D'} \) is larger than a predefined threshold \( \delta \).

Once the value of \( EstDist_{1,N}^{Q',D'} < \delta \), we seek the second check point on \( Q' \) and \( D' \), respectively, by checking \( Q_2 \) and \( D_2 \). If \( Q_2 \) is chosen as \( Q'_2 \), we can insert \( D_2 \) to the right position between \( D_1 \) and \( D_2 \) and vice versa. Furthermore, \( Q' \) and \( D' \) can be divided into four sub-trajectories by \( Q'_2 \) and \( D'_2 \). Under these circumstances, we only compute the sum of \( EstDist_{1,2}^{Q',D'} \) and \( EstDist_{2,N}^{Q',D'} \) as the distance between sub-trajectories. If the distance between two distinct sub-trajectories is still larger than a predefined threshold \( \delta \), \( D \) will be filtered out. Otherwise, we insert \( Q'_3 \) and \( D'_3 \) to further compute \( EstDist_{2,3}^{Q',D'} \) and \( EstDist_{3,N}^{Q',D'} \).

The above newly computed distances replace the value of \( EstDist_{1,2}^{Q',D'} \), and the process is executed repeatedly until the computed distance is larger than \( \delta \), or there are no more intermediate check points within each sub-trajectory. Since most of the trajectories would be filtered out by checking the first few control points, our proposed algorithm is very efficient. Furthermore, assuming the query from a user is complete and correct, the partial matching problem can be solved by choosing two distinct control points in \( D \) as the new start and end points of \( D \).

4. EXPERIMENT RESULTS

To test our method, we extracted more than six thousand motion flows from a three-hour MPEG-7 test video sequence and generated a video database. The database contained more than one thousand video shots of different subjects, including news, sport, documentaries, and home videos. Since we had extracted motion flows directly from the test video, in the experiments we sketched a target trajectory as an input query. The system computed all related
data from the query and compared this set of data with all the data stored in the video database. Therefore, the response time of this querying process was within one second (given a thousand shots in the database). Fig.6 illustrates two sets of experiment results, all derived by the query-by-sketch approach. In Fig.6a, we query the video database by a cyclic motion and the three most similar candidates are retrieved and listed from top to bottom according to the degree of similarity. The first row of the image sequence shows the action of a mower. The second row shows the movement of basketball players, and the third row shows a man shaking his head. All three retrieved image sequences had a cyclic motion. The advantage of our approach is that our algorithm does not need to segment the moving object in a video clip. We use the third retrieved image sequence as an example to show the superiority of our approach. Suppose we segment the man in the third image sequence in Fig.6a. We cannot detect a cyclic motion, because the trajectory formed by linking the centroids of a video object located in consecutive frames is quite steady. However, our approach can retrieve the cyclic motion caused by the movement of the face. In Fig.6b, we show another example in which we adopted a repeated jump motion as the query. The first and second retrieved image sequences both show the movement of girls jumping on a spring mattress. The third retrieved image sequence shows two people dancing in a room.

The response times of the above two video retrieval examples were very short due to our specially designed comparison procedure. Clearly, our system can retrieve reasonably close motion from a video database through a quick sketch. However, if a sketch is too simple, our system will respond with many hard-to-judge retrieval results. Fig.7 illustrates an example of this problem. Suppose we query the video database by a simple straight motion from left to right. A lot of similar candidates will be retrieved so that we need to spend time to browse and find the video clips we requested. This situation is to be expected because the less information we provide to the system, the less specific the solutions we will get. Since the query shown in Fig.7 contains very little information, we can only retrieve very “rough” results. Therefore, if we have more information about the trajectory of a motion, we can draw a “closer” (more complicated) query to retrieve the video clips from the database. Under these circumstances, the retrieved results will be much closer to the query, as shown by the cases in Fig.6a and Fig.6b.

Since the proposed approach can retrieve video clips from a large database in a very efficient manner, and since most of the retrieved results are close to the input query, we can use the approach as a coarse search engine. It is obvious that video databases of the future will be very large indeed. If we do not manage this kind of database, efficiently, it will be not possible to enjoy the multimedia world of the future.

5. CONCLUSION

In this paper, we have proposed a new representation scheme for the local motion in videos that does not need to perform object extraction and tracking in the representation process. We use the information of motion vectors in an MPEG bitstream directly to generate some trajectory-like motion flows for the description of local motion. Since motion vectors are uniformly distributed in each video frame, we can process the case of multiple moving objects in a shot.
We have also proposed a new matching algorithm for retrieving similar trajectories in a more efficient way. Most unnecessary comparisons can be avoided, thereby enhancing the efficiency of the query process. However, there are still some problems in the process of motion flow generation that need to be resolved. First, the information about motion vectors is not always reliable. The worst case is when there are multiple moving objects in a macroblock, the macroblock is usually intra-coded and loses motion information in the bitstream. Second, since we remove camera motion and extract local motion from a video, if a moving object stops suddenly in a frame, the motion flow(s) will break off, even if the object moves again later. We will address these problems in our future work.

6. ADDITIONAL AUTHORS
Additional authors: Chia-Wen Lin (Department of Computer Science and Information Engineering, National Chung-Cheng University, Taiwan) and Hsiao-Rong Tyan (Department of Information and Computer Engineering, Chung-Yuan Christian University, Taiwan.)

7. REFERENCES


