GENERIC ALGORITHM FOR TRACKING GENERIC OBJECTS IN A MOTION PICTURE

D.V. Chandra Shekar*1, Y. Suresh Babu2 and Dr. G. S. Prasad

*1Asst Prof, Dept of Computer Science, T.J.P.S College, P.G Courses, Guntur, chand.info@gmail.com
2Asst Prof , Dept of Computer Science J.K.C College, Guntur


Abstract: Tracking of multiple objects from ‘real world’ is one of the most complicated problems in computer vision since the behavior/motion of these objects is unpredictable and cannot be assumed. Tracking is also "application depended" task; there is no one general tracking methodology for solving all tracking problems. Therefore, different algorithms and procedures are needed and used for different applications. However, it is possible to rank tracking applications in two general categories: the ones that need a moving camera and the ones where a stable camera is enough. This paper deals with the second category and the technique of the "closed-world" for object tracking. It defines the lower level problem, the possible circumstances of the objects movements and the algorithms to solve some of the tracking situations. The obtained experimental results are pretty promising, approaching 100% hit rate, even in cases of two-object collision. In three-object collision occasional re-initialization is required.

INTRODUCTION

Object tracking from “real-world” is one of the most complicated problems in computer vision. Projected objects size and radiometric values change from frame to frame, making almost impossible the use of a description template. Also, their motion is unpredictable. They can move rapidly to any direction; objects can also move isolated, together with other objects or been hidden for some time before they reappear in the tracking scene.

In this paper, a tracking algorithm of multiple, non-rigid objects, that uses a stable camera, is described. The algorithm is based on the “closed-world” assumption (Intille 1994, Intille et al 1995,1996) but, contrary to this technique, it does not use any contextual information in order to keep the algorithm as more general as it can be.

ESTIMATION OF THE BACKGROUND

Extracting the background image from sequences of frames is a very important task in order to help tracker detect motion. This task is repeated from time to time in order to incorporate any changes in the illumination of the tracking scene. There are several methods used to extract the background image from a sequence of frames but three are the most popular. These are based on statistical characteristics on the pixels of the frames: mean, median and highest appearance frequency methods. In all methods, every pixel of the background image is separately calculated using the mean or the median or the highest appearance frequency value from the series of frames.

Each method has its advantages and disadvantages - shown on Table 1 - ranging from CPU time and memory requirements to easiness of adaption on light changes and objects motion.

THE DIFFERENCE IMAGE

A difference image is the image that is produced by subtracting the background image from each frame and it is used in order to detect the parts of the image where movement is taking place. Additionally, in order to avoid noise and camera jitter, that produces false motion detection, the difference image must be thresholded. Threshold value cannot be calculated theoretically; it is usually determined after the sampling of many frames and its determination is based on the highest pixel differences. However, it is possible that some noise will still be present after threshold.

In general, if G(i,j) is the background image and F(i,j) any single frame, then the difference image D(i,j) is calculated and thresholded using the following formula:

\[ D(i,j) = \begin{cases} 0, & \text{if } |F(i,j) - G(i,j)| \leq \text{threshold} \\ 1, & \text{if } |F(i,j) - G(i,j)| > \text{threshold} \end{cases} \]

All pixels of the difference image that have value 1 include motion and probably belong to an object, whereas the pixels with value 0 are the same as the background and are
ignored. In color images, where there are usually three bands of color information, the difference image can be calculated either separated in each band or in a single grayscale band, which is the combination of the three-color bands.

![Image](image.png)

**Figure 1:** Moving objects in a difference image from a Traffic Junction in Gray and Tri-color

---

### BLOBS AND OBJECTS

Blobs have a long history in computer vision as a representation of image features (Kauth et al 1977, Pentland 1976). A blob can be described as a set of connected pixels that share a common attribute. This attribute can be any one of the color, texture, brightness, shading or other salient spatio-temporal attribute, derived from the image sequence, or any combination of the above. In tracking, the common attribute is usually motion.

All motion pixels in the difference image are clustered in blobs. There are several techniques to cluster and extract blobs; most common techniques are “meandering process” (Rossey, 1997) and “connected pixels”. In our implementation we have used an updated version of the popular image processing FILL procedure; the implementation can extract blobs either from the whole image or from a specific part of it.

After blob detection, every blob can be represented by its properties; the most important of them are size, color and position.

**Size:** The size $N$ of a blob is the number of the pixels that comprise the blob.

**Color:** The color $C_m$ of a blob is the mean color of the $N$ pixels that comprise the blob.

\[
C_m = \frac{1}{N} \sum_{i,j} F(i,j) \quad (2)
\]

It must be noticed here that the pixel color values $F(i,j)$ are taken from the original frame image.

\[
X_m = \frac{1}{N} \sum_{i,j} x(i,j), \quad Y_m = \frac{1}{N} \sum_{i,j} y(i,j) \quad (3)
\]

**Position:** The position $P(x_m, y_m)$ of a blob is the geometric center of the $N$ pixels that comprise the blob.

**Objects**

Since objects create motion, difference image detects motion and blobs are image features that describe motion, objects can be described by blobs. One would expect that every object could be described by only one blob but the common case is that an object is comprised by more than one blobs, as shown in image 2.

![Image](image.png)

**Figure 2:** An object (Traffic Junction) that is comprised by two blobs

Much like blobs, every object can be represented with its properties such as color, size and position. These properties are calculated just like the blob ones, with the difference that in calculation all the actual blobs (and not their properties) are used. Additionally, the object must have some more properties to help tracking. The most important of these additional properties is the bound box that includes all the blobs of the object.

---

### TRACKING ALGORITHM

The tracking algorithm uses two basic data structures:

a. A list of all active objects with their current properties
b. A history list that keeps all objects properties and information about matching type and score in time space.

The process of tracking follows the next steps:

a. Initialization of the objects to be tracked
b. Prediction of objects future status
c. Find new objects in predicted positions
d. Matching of objects
e. Register objects with new properties
f. Update history list
g. Go to step 2 and repeat process

**Prediction of new Object Status:**

In order to help tracker find the objects in a new frame, the “expected status” of all objects is predicted. The “expected status” includes all object’s main properties.
The most common technique to predict these properties is the use of Kalman filters, technique that gives very good results. Alternatively, if tracking speed is high (usually more than 5 FPS) and the speed of the objects is relatively low, the motion of the objects can be assumed locally as linear and linear interpolation can be used with very good results, too.

**Collision Detection:**

Since objects in “real world” applications can move unexpectedly in the tracking scene, they can be projected isolated or in touch with another or been hidden by some other. And, since the tracking of isolated objects is much easier than in other two situations, it is useful to help tracker predict in which status the objects are. This task can be achieved by predicting possible object collisions.

The property that is used to achieve collision detection is the object’s bounding box. Every corner of each object’s box is examined whether it is inside any other box. If any of the corners are inside, the objects are in collision and the tracking algorithm changes for these objects.

Three different collision stages may exist:

i. An object is isolated  

ii. An object is collided with one or more but it’s blobs are not in touch with other objects blobs  

iii. Object is collided with one or more and it’s blobs are in touch with other object blobs, producing common huge blobs

**Assigning Blobs to Objects:**

For every object to be tracked (from now on it will be referred as old object) a new one is created; it inherits all the properties of the old one except it’s main properties. Then, the blobs that exist into the expected bounding box are assigned to new object - depended on old object’s expected collision stage - in order to calculate the new main properties that will be used to identify if two objects are similar.

If the old object is expected to be isolated, all blobs in expected bounding box are assigned to the new object. Additionally, from time to time, the expected bounding box can grow up in order to avoid a bad prediction of its size or to re-estimate the actual object size.

If the old object is expected to be in collision stage (ii) or (iii), only the blobs in expected bounding box that their distance is smaller than a distance threshold are assigned to the new object. The others are dropped. The distance threshold value depends on image scale and object mean size.

Blob dropping may help in distinguish two or more collided objects but also leads to shrinking of the object’s actual size. Therefore, it must be used with caution and only in collision stages. Additionally, when the object comes to isolated stage in later frames, its expected bounding box must grow up to re-estimate object’s actual size.

After blob assignment, new object’s properties are calculated, as described in §4.2, and object match procedure follows.

**Matching an old object with the new one:**

The term matching has the meaning of identification. The new object matches to old one if they are similar. If they do match, the position (goal of tracking) of the old object in new frame will be also the position of the new object.

Object main properties are used to calculate the similarity between two objects. That is they are similar if

a. the weighted sum of color and size differences is larger than a threshold AND  

b. the distance between the two where

\[
\begin{align*}
    s_i &= p_c \sqrt{1 - (c_o - c_i) / c_o} \cdot p_s \sqrt{1 - (N_o - N_i) / N_o} \\
    s &= \sqrt{s_i^2 + s_j^2} \\
    d &= \sqrt{(i_o - i_c)^2 + (j_o - j_c)^2} \\
    \text{match if } (s > t_1) \text{ AND } (d < t_2) \\
\end{align*}
\]

\(p_c, p_s\) = weights for color and size  
\(c_o, N_o\) = color and size of new object  
\(c_i, N_i\) = color and size of old object OR predicted ones  
\(i_o, j_o\) = coordinates of new object  
\(i_c, j_c\) = coordinates of old object OR predicted ones  
\(t_1\) = threshold  
\(t_2\) = distance threshold
Parameters $p_a$, $p_b$, and threshold $t_i$ are application depended and estimated after sampling on many frames. Threshold $t_2$ depends on object’s maximum speed vector. There are two weighted sums that calculated:
- Between new and old object actual properties and
- Between new object and olds’ predicted properties.

The second weighted sum is very helpful when sudden or great changes in color or size of the object happen. If any of the above sums passes the threshold, the two objects are assumed similar and the new object replaces the old one. Additionally, separate comparisons are made on every property and flag successful or failed individual matching.

The above matching formula (6) works with very good results when object is isolated or in collision stage (ii). If object is in stage (iii) (Fig3, c&d) then it’s size grows up suddenly with the objects is lower than a distance threshold. The possibility that also its color differs a lot from previous frame; the weighted sums then cannot pass the threshold and the matching flag on size and possibly on color is failure. In this case, matching fails and the position of the object is re-estimated using an adaptive correlation matching procedure.

**Adaptive Correlation Matching:**

The adaptive correlation matching is a variation of the classic cross correlation technique. Their difference stands to the pixels that participate in correlation. In adaptive correlation, only pixels that belong to object’s blobs participate, as shown in fig. 4, in order to avoid background disturbance.

As in typical correlation, it is possible that a lot of template positions will have high correlation scores and the correct object’s position is not that with the highest one; in this case, the assumed correct position is the one with the minimum distance from the predicted position. The object’s position is updated in the current frame but its size and color retain their values from the last frame.

If maximum correlation factor is very low, the matching fails; But in this case, the following assumption is made:

a. If there are only two objects that are in stage (iii) then the object is assumed hidden by the other; therefore, its position is the position of the object that stands in front of it.

b. If there are more than two objects in stage (iii) then it is unknown which object hides the wanted one and the object needs re-initialization in next frame.

Correlation match will work for a short range of sequent frames. If an object is in stage (iii) for a long time, then its shape will be very different after a number of frames from the shape it had at the start of the sequence. Additionally, there will be a drift on template center, as time passes (Intille, 1996). Therefore, correlation matching must be used for a short time period (for few frames).

**TESTING THE ALGORITHM - PERFORMANCE**

The described tracking technique was tested on a 430 frames video sequence of a basketball game, having 5 fps speed and image size 768x576 pixels. The scale of the tracking scene

**Example:**

Varied from 1: 130 up to 1:70 while the size of the objects varied from 116 to 400 pixels. The color variation was different for each object and was from 7 gray shades up to 40 (in extreme cases), with a mean variation of 20 gray shades.

The parameters that were used for object identification (matching) were:

\[ p_a : 0.60, p_b : 0.40, t_1 : 0.65, t_2 : 15 \]

**Performance:**

The tracker managed to track all isolated objects in all frames with no failure and collision stage (ii) objects with two failures. In case of collision stage (iii) objects, tracker succeeded only on two objects collision and in few cases on three objects, when objects shared the same space for a short time period; In general, it failed when more than two objects shared the same space. In these cases re-initialization was performed.

**CONCLUSIONS**

The described technique works very good when the objects do not share the same blobs or only two objects are interacting. In general, it suffers when more than two objects are in the same space or the movement of the objects is rapid.

Since the tracking depends on correct interpreting of which objects share the same blobs, in conjunction with objects motion, current research is focused on this aspect.
REFERENCES


Evaluation of Multiple-Object Tracking Algorithms using Performance Metric

1D.V. Chandra Shekar
Assoc Prof, Dept of Computer Science, T.I.P.S College, P.G Courses, Guntur,
chand.info@gmail.com
Andhra Pradesh-522006, India

2Y. Suresh Babu
Assoc Prof, Dept of Computer Science J.K.C College, Guntur
Andhra Pradesh - 522006.
yalavarti_s@yahoo.com

3 Dr. G.S. Prasad, Prof. Dept of CSE, RVR & JC College, Guntur-522019

Abstract

This paper deals with the non-trivial problem of performance evaluation of motion tracking. We propose a rich set of metrics to assess different aspects of performance of motion tracking. We use six different video sequences that represent a variety of challenges to illustrate the practical value of the proposed metrics by evaluating and comparing two motion tracking algorithms. The contribution of our framework is that it allows the identification of specific weaknesses of motion trackers, such as the performance of specific modules or failures under specific conditions.

1. Introduction

Significant research effort has focused on video-based motion tracking [1] [2] [3] [4] and attract the interest of industry. Performance evaluation of motion tracking is important not only for the comparison and further development of algorithms from researchers, but also for the commercialisation and standardisation of the technology as typified by the i-LIDS Programme in the UK [5].

In this paper, we have selected a set of motion tracking metrics that are used to highlight different aspects of motion tracking performance. We illustrate the purpose of the proposed metrics in the evaluation of two motion tracking algorithms, using a variety of datasets.

Ellis [6] investigated the main requirements for effective performance analysis for surveillance systems and proposed some methods for characterising video datasets.

Nascimento and Marques [7] proposed a framework which compares the output of different motion detection algorithms against given ground truth and estimates objective metrics such as Correct Detections, False alarms, Detection failures, Merges and Splits. They also proposed ROC-like curves that can characterize algorithms over a range of parameters.

Lazarevic-McManus et al [8] developed an object-based approach to enable evaluation of motion detection, based on ROC-like curves and the F-measure. The latter allows straightforward comparison using a single value that takes into account the application domain.

Needham and Boyle [9] proposed a set of metrics and statistics for comparing trajectories and evaluating tracking motion systems.

Brown et al [10] suggest a motion tracking evaluation framework that estimates the number of True Positive (TP), False Positive (FP) and False Negative (FN), Merged and Split trajectories. However their definition, based on the comparison of the system track centroid and an enlarged ground truth bounding box) favours tracks of large objects.

Bashir and Porikli [11] gave definitions of the above metrics based on the spatial overlap of ground truth and system bounding boxes that are not biased towards large objects. However they are counted in terms of frame samples. Such an approach is justified when the objective of performance evaluation is object detection [7] [8]. In object tracking, measuring TP, FP and FN in terms of tracks rather than frames is a natural choice that is consistent to the expectations of the end-users.

This paper is organized as follows: Section 2 defines provides the definitions for motion tracking and track. Section 3 describes the performance evaluation methodology and gives definition of different metrics. Results are presented and discussed in section 4. Section 5 concludes the paper.

2. Motion Tracking

We define motion tracking as the problem of estimating the position and the spatial extent of the non-background objects for each frame of a video sequence. The result of motion tracking is a set of tracks $T_j$, $j=1..M$, for all M moving objects of the scene. A track $T_j$ is defined as: $T_j = \{x_{ij}, B_{ij}\}$, $i=1..N$, where $x_{ij}$ and $B_{ij}$ are the centre and the spatial extent (usually represented by a bounding box) respectively of the object $j$ for the frame $i$ and $N$ is the number of frames.
3 Performance Evaluation

3.1 Preparation

We propose a set of metrics that compare the output of motion tracking systems to a Ground Truth in order to evaluate the performance of the systems.

Before the evaluation metrics are introduced, it is important to define the concepts of spatial and temporal overlap between tracks, which are required to quantify the level of matching between Ground Truth (GT) tracks and System (ST) tracks, both in space and time.

The spatial overlap is defined as the overlapping level \( A(GT_i, ST_j) \) between \( GT_i \) and \( ST_j \) tracks in a specific frame \( k \) (Fig. 1).

\[
A(GT_{ik}, ST_{jk}) = \frac{\text{Area}(GT_{ik} \cup ST_{jk})}{\text{Area}(GT_{ik} \cap ST_{jk})} \quad (1)
\]

![Figure 1: Area \((GT_i \cup ST_j)\) and Area \((GT_i \cap ST_j)\)

We also define the binary variable \( O(GT_{ik}, ST_{jk}) \), based on a threshold \( T_{ov} \) which in our examples is arbitrarily set to 20%.

\[
O(GT_{ik}, ST_{jk}) =
\begin{cases} 
1 & \text{if } A(GT_{ik}, ST_{jk}) > T_{ov} \\
0 & \text{if } A(GT_{ik}, ST_{jk}) \leq T_{ov}
\end{cases} \quad (2)
\]

Temporal overlap \( TO(GT_{ik}, ST_{jk}) \) is a number that indicates overlap of frame span between system track \( j \) and GT track \( i \):

\[
TO_{E} - TO_{S}, \quad TO_{E} > TO_{S} \quad (3)
\]

\[
0, \quad TO_{E} \leq TO_{S}
\]

where \( TO_{E} \) is the maximum of the first frame indexes of \( TO_{S} \) is the minimum of the last frame indexes of the two tracks. The temporal and spatial overlap between tracks is illustrated graphically in Fig. 2:

![Figure 2: Example of track overlapping

We use a temporal-overlap criterion to associate systems tracks to GT tracks according to the following condition:

\[
\sum_{k=1}^{N} A(GT_{ik}, ST_{jk}) > N \quad (4)
\]

where \( N \) is the number of frames and \( T_{ov} \) an arbitrary threshold. If Eq.4 is true, then, we associate the system track with the GT track and start evaluating the performance of the system track.

3.2 Metrics

In this section, we give definitions of high level metrics such as True Positive (TP), False Positive (FP) and False Negative (FN) tracks. Such metrics are useful because they are the base for estimating metrics such as Specificity and Accuracy [11] and allow the construction of ROC-like curves [8]. Metrics such as Track Fragmentation and ID Change assess the integrity of tracks. Finally, we define metrics that measure the accuracy of motion tracking in estimating the position (Track Matching Error), the spatial extent (Closeness), the completeness and the temporal latency.

Correct detected track (CDT) or True Positive (TP): A GT track is considered to have been detected correctly if it satisfies both of the following conditions:

**Condition 1:** The temporal overlap between GT Track \( i \) and system track \( j \) is larger than a predefined track overlap threshold \( T_{ov} \) which in our examples is arbitrarily set to 15%.

\[
\frac{\text{Length}(GT_{i})}{\text{Length}(ST_{j})} \geq T_{ov} \quad (5)
\]

**Condition 2:** The system track \( j \) has sufficient spatial overlap with GT track \( i \).

\[
\sum_{k=1}^{N} A(GT_{ik}, ST_{jk}) \geq N \quad (6)
\]

Each GT track is compared to all system tracks according to the conditions above. Even if there is more than one system track meets the conditions for one GT track (which is probably due to fragmentation), we still consider the GT track to have been correctly detected. Fragmentation errors are
counted separately (see below). So, if each of the GT tracks is detected correctly (by one or more system tracks), the number of CDT equals the number of GT tracks.

**False alarm track (FAT) of False Positive (FP):**

Although it is easy for human operators to realise what is a false alarm track (event) even in complex situation, it is hard for an automated system to do so. Here, we give a practical definition of false alarm track (Fig. 3). We consider a system track as false alarm if the system track meets any of the following conditions:

**Condition 1:** A system track $j$ does not have temporal overlap larger than $T_{OV}$ with any GT track $i$.

\[
\frac{\text{Length}(GT_i \cap ST_j)}{\text{Length}(ST_j)} < T_{ov}
\]  

**Condition 2:** A system track $j$ does not have sufficient spatial overlap with any GT track although it has enough temporal overlap with GT track $i$.

\[
\frac{\sum_{k=1}^{N} A(GT_{ik}, ST_{jk})}{N} < T_{ov}
\]  

**Track Fragmentation (TF):**

Fragmentation indicates the lack of continuity of system track for a single GT track. Fig. 4 shows an example of track fragmentation error:

\[
N \sum_{j=1}^{N} D_{j,k} < T_{ov}
\]

**ID Change (IDC):**

We introduce the metric $IDC_j$ to count the number of ID changes for each ST$_j$ track. Note that such a metric provides more elementary information than an ID swap metric.

For each frame $k$, the bounding box $D_{j,k}$ of the system track ST$_j$ may be overlapped with GT areas, where $N_{D_{j,k}}$ is given by:

\[
N_{D_{j,k}} = \sum_{i} A(GT_{ik}, D_{j,k})
\]
We take into account only the frames for which \( N_{Dj,k} = 1 \) (which means that the track \( ST_j \) is associated (spatially overlapped) with only one GT Track for each of these frames). We use these frames to estimate the ID changes of \( ST_j \) as the number of changes of associated GT tracks.

We can estimate the total number of IDC changes in a video sequence as:

\[
IDC = \sum_{j} IDC_j
\]

The procedure for counting ID change is shown in Fig. 5. Some examples of estimating ID changes are shown in Fig. 6:

For each System track \( j \):

\[
IDC_{j} = -1
\]

For every frame \( k \):

- If there is no occlusion, and \( N_{Dj,k} = 1 \):
  - If \( IDC_{j} = -1 \) or \( M(IDC_{j}) \neq i \) (\( i \) is the ID of GT track whose area is overlapped with the GT track in this frame.)
  - Then, \( IDC_{j} = IDC_{j} + 1 \); \( M(IDC_{j}) = i \); End

End

End

Figure 5: Pseudo-code for estimated ID Changes (IDC)

Latency of the system track (LT):

Latency (time delay) of the system track is the time gap between the time that an object starts to be tracked by the system and the first appearance of the object (Fig. 7). The optimal latency should be zero. A very large latency means the system may not be sensitive enough to trigger the tracking in time or indicates that the detection is not good enough to trigger the tracking.

It is estimated by the difference in frames between the first frame of system track and the first frame of GT track.

\[
LT = \text{start frame of } ST_j - \text{start frame of } GT_i
\]

Closeness of Track (CT):

For a pair of associated GT track and system track, a sequence of spatial overlaps (Fig. 2) is estimated by Eq.2 for the period of temporal overlap:

\[
CT(GT_i, ST_j) = \{A(GT_{i1}, ST_{j1}), A(GT_{i2}, ST_{j2}), \ldots, A(GT_{iN_{ES}}, ST_{jN_{ES}})\}
\]

From Eq.14, we can estimate the average closeness for the specific pair of GT and system tracks. To compare all \( M \) pairs in one video sequence, we define the closeness of this video as the weighted average of track closeness of all \( M \) pairs:

\[
CTM = \frac{\sum_{i=1}^{M} \sum_{t=1}^{N_{ES}} CT_{t}i}{\sum_{i=1}^{M} \text{Length}(CT_i)}
\]

and the weighted standard deviation of track closeness for the whole sequence:

\[
CTD = \frac{\sum_{i=1}^{M} \text{Length}(CT_i) \cdot \text{std}(CT_i)}{\sum_{i=1}^{M} \text{Length}(CT_i)}
\]

where std(CT_i) is the standard deviation of CT_i

Track Matching Error (TME):

This metric measures the positional error of system tracks. Fig. 8 shows positions of a pair of tracks.

TME is the average distance error between a system track and the GT track. The smaller the TME, the better the accuracy of the system track will be.
\[
TME = \frac{\sum_{k=1}^{N} \text{Dist}(GT_{ik}, STC_{jk})}{\text{Length}(GT_{i}, 1ST_{j})}
\] (17)

where \(\text{Dist()}\) is the Euclidean distance between the centroids of GT and the system track:

\[
\text{TMED} = \sqrt{\frac{\sum_{k=1}^{N} (\text{Dist}(GT_{ik}, STC_{jk}) - TMEM)^2}{\text{Length}(GT_{i}, 1ST_{j}) - 1}}
\] (18)

Similarly, track matching error (TMEMT) for the whole video sequence is defined as the weighted average over the duration of overlapping of each pair of tracks as the weight coefficient:

\[
\text{TMEMT} = \frac{\sum_{i=1}^{M} \text{Length}(GT_{i}, 1ST_{j}) \cdot TME_{i}}{\sum_{i=1}^{M} \text{Length}(GT_{i}, 1ST_{j})}
\] (19)

and the standard deviation of track matching errors for the whole sequence:

\[
\text{TMEMTD} = \frac{\sum_{i=1}^{M} \text{Length}(GT_{i}, 1ST_{j}) \cdot \text{TMED}_{i}}{\sum_{i=1}^{M} \text{Length}(GT_{i}, 1ST_{j})}
\]

4. Results

We demonstrate the practical value of the proposed metrics by evaluating two motion tracking systems (an experimental industrial tracker from BARCO and the OpenCV1.0 blobtracker [12]). We run the trackers on six video sequences (shown in Fig.9-Fig.14) that represent a variety of challenges, such as illumination changes, shadows, snow storm, quick moving objects, blurring of FOV, slow moving objects, mirror image of objects and multiple object intersections. The ground truth for all videos was manually generated using Viper GT [13].

Figure 9: PETS2001 PetsUT1TeC1.avi sequence is 2686 frames (00:01:29) long and depicts 4 persons, 2 groups of persons and 3 vehicles. Its main challenge is the multiple object intersections.

Figure 10: i-LIDS SZTRA103b15.mov sequence is 5821 frames (00:03:52) long and depicts 1 person. Its main challenges are the illuminations changes and a quick moving object.

Figure 11: i-LIDS SZTRA104a02.mov sequence is 4299 frames (00:02:52) long and depicts one person.
Figure 12: i-LIDS PVTRA301b04.mov sequence is 7309 frames (00:04:52) long and depicts 12 persons and 90 vehicles. Its main challenges are shadows, moving object in the beginning of sequence and multiple object intersections.

Figure 13: BARCO 060306_04_Parkingstab.avi is 7001 frames long and depicts 3 pedestrians and 1 vehicle. Its main challenge is the quick illumination changes.

Figure 14: BARCO 060306_02_Snowdivx.avi is 8001 frames long and depicts 3 pedestrians. Its main challenges are snow storm, blurring of FOV, slow moving objects and mirror image of objects.

The results of performance evaluation are presented in Tables 1-8. Generally, high level metrics such CDT, FAT, TDF show that the BARCO tracker outperforms the OpenCV tracker in almost all cases. The only exception is the i-Lids sequence PVTRA301b04.mov (Fig. 12, Table 6). On the other hand, the OpenCV tracker is generally more accurate in estimating the position of the objects (lower TMEM).

Also, the OpenCV tracker dealt better with the snow scene (Fig.14, Table8) in estimating the position and the spatial and temporal extent of the objects (lower TMEM, higher CTM and TCM), which implies a better object segmentation module for this scene. However, the BARCO tracker has better high level metrics (lower FAT, lower TF), which implies a better tracking policy.

Note that without the rich set of metrics as used here it is very difficult to identify possible causes of poor/good performance in different trackers.

<table>
<thead>
<tr>
<th>Table 1 Evaluation results for PETS2001 Seq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetsD1TeC1.avi</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>GT Tracks</td>
</tr>
<tr>
<td>System Tracks</td>
</tr>
<tr>
<td>CDT</td>
</tr>
<tr>
<td>FAT</td>
</tr>
<tr>
<td>TDF</td>
</tr>
<tr>
<td>TF</td>
</tr>
<tr>
<td>IDC</td>
</tr>
<tr>
<td>LT</td>
</tr>
<tr>
<td>CTM</td>
</tr>
<tr>
<td>CTD</td>
</tr>
<tr>
<td>TMEMT</td>
</tr>
<tr>
<td>TMEMTD</td>
</tr>
<tr>
<td>TCM</td>
</tr>
<tr>
<td>TCD</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 2 BARCO track association results for Pets2001</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetsD1TeC1.avi: BARCO tracker</td>
</tr>
<tr>
<td>GT-ID</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3 OpenCV track association results for PETS2001</th>
</tr>
</thead>
<tbody>
<tr>
<td>PetsD1TeC1.avi: OpenCV tracker</td>
</tr>
<tr>
<td>GTID</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>
5. Conclusions

We presented a new set of metrics to assess different aspects of performance of motion tracking. We proposed statistical metrics, such as Track matching Error (TME), Closeness of Tracks (CT) and
Track Completeness (TC) that indicate the accuracy of estimating the position, the spatial and temporal extent of the objects respectively and they are closely related to the motion segmentation module of the tracker.

Metrics, such as Correct Detection Track (CDT), False Alarm Track (FAT) and Track Detection Failure (TDF) provide a general overview of the algorithm performance. Track Fragmentation (TF) shows the temporal coherence of tracks. ID Change (IDC) is useful to test the data association module of multi-target trackers.

We tested two trackers using six video sequences that provide a variety of challenges, such as illumination changes, shadows, snow storm, quick moving objects, blurring of FOV, slow moving objects, mirror image of objects and multiple object intersections.

The variety of metrics and datasets allows us to reason about the weaknesses of particular modules of the trackers against specific challenges, assuming orthogonality of modules and challenges. This approach is a realistic way to understand the drawbacks of motion trackers, which is important for improving them.

In future work, we will use this framework for evaluating more trackers. We will also extend the framework to allow evaluation of high level tasks such as event detection and action recognition.

6. References


 AUTHORs PROFILE

D.V. Chandra Shekar, received Master of Engineering with Computer Science & Engineering from ANNA University, Chennai, Tamilnadu, India. He is currently working as Associate Professor, in the Department of Computer Science, T.J.P.S COLLEGE (P.G COURSES),Guntur, which is affiliated to Acharya Nagarjuna University. He has 12 years teaching experience and 1 years of Industry experience. He has published 30 papers in National & International Journals.

Y. Suresh Babu , working as Associate Professor, in the Department of Computer Science, J.K.C COLLEGE, Guntur, which is affiliated to Acharya Nagarjuna University. He has 18 years teaching experience. He has published 8 papers in National & International Journals.

Dr. G. Satyanarayana Prasad received Ph.D. degree in Computer Science in the Faculty of Engineering in 2006 from Andhra University, Andhra Pradesh. He completed MS in Computer Science & Engineering from A & M University,USA,1984. He is currently working as Professor in the Department of Computer Science and Engineering, R.V.R & J.C Engineering College, Guntur. His current research is focused on Image Processing. He has published several papers in IEEE journals and various National and International Journals.
A Frame Work Using Hyper-Based Methods for Image Registration and Super Resolution

Suresh Babu Yalavarthi*  
Asst. Prof.  
Computer Science, JKC College, Guntur, India  
yalavarthi_s@yahoo.com

D.V. Chandra Shekar  
Asst Prof.  
Computer Science, T.J.P.S College, Guntur, India  
chand.info@gmail.com

Dr. G.S. Prasad  
Prof. Dept of CSE, R.V.R. & J.C. College of Engineering, Chowdavaram, Guntur, India  
satyam_gp@yahoo.com

Abstract: High quality-resolution algorithms combine diversified low resolution images into a single Super resolution image. They have received a lot of attention recently in various application domains such as HD Systems, satellite imaging, and video surveillance. These techniques take advantage of the aliasing present in the input images to reconstruct high frequency information of the resulting image. One of the major challenges in such algorithms is a good alignment of the input images: subpixel precision is required to enable accurate reconstruction. In this paper, we give an overview of some subspace techniques that address this problem. We first formulate super-resolution in a multi-channel sampling framework with unknown offsets. Then, we present three registration methods: one approach using ideas from variable projections, one using a Fourier description of the aliased signals, and one using a spline description of the sampling kernel. The performance of the algorithms is evaluated in numerical simulations.

Index Terms—Image registration, Image resolution, Image restoration, Spectral analysis, Spline functions

I. INTRODUCTION

A high resolution input image is required in many imaging applications: recognition of people or license plates, satellite imaging, viewing video material on HD displays, etc. Unfortunately, capturing devices with such a high resolution are often still prohibitively expensive. In some cases, such as satellite imaging, it can also be practically impossible to replace a camera by a higher resolution version when it becomes available. The wide adoption of HD TVs generates an increasing demand for the conversion of legacy SD video material (or even freshly shot camera phone images and movies) to HD resolution. This is exactly the goal of super-resolution algorithms: to generate a high resolution image from a set of low resolution images.

Typically, the input images need to be aliased. If multiple images are then taken with small relative motion, the aliasing information from the different images can be used to reconstruct the high frequency part of the high resolution image. The idea of super-resolution imaging was first introduced by Tsai and Huang in 1984 [4]. They used a frequency domain algorithm to minimize the energy outside the known frequency range of the images. In the past fifteen years, a vast number of algorithms has been presented. Good overviews are given in the special issues on this topic in the IEEE Signal Processing Magazine [5] and the EURASIP Journal of Applied Signal Processing [6]. Superresolution reconstruction is an ill-posed problem. It is usually turned into a well-posed problem using regularization or an image model. We will consider the second case, where we assume our image (after sampling) to be in the Fourier or spline space. Most algorithms treat image registration (aligning the input images) and reconstruction (combining the aligned images to a single high resolution image) as two separate problems. A standard image registration algorithm is often used for the alignment, and emphasis is put on the reconstruction part of the algorithm [7, 8]. Recently, new algorithms have been developed that specifically consider registration methods for (aliased) input images, and are successfully integrated in the superresolution algorithm [1, 2, 3, 9, 10]. In this paper, we present three of those registration algorithms in a common framework based on multichannel sampling. In the next section, we will set up this framework by describing super-resolution as a multichannel sampling problem with unknown offsets. We will then use this setup to describe three algorithms that solve this problem in Section 3: using projections, Fourier analysis, and spline analysis. Simulation results of each method are presented in Section 4, and Section 5 gives some concluding remarks.

II. SUPER-RESOLUTION AS A MULTI CHANNEL SAMPLING PROBLEM

We will now analyze the super-resolution setup mathematically. While this description is given in 1D for simplicity, it can straight forwardly be extended to 2D. Let \( f(t) \) be a continuous-domain signal in an L-dimensional Hilbert space \( H \) with basis \( \{ \phi(t) \}_{t=0}^{L-1} \):

\[
f(t) = \sum_{t=0}^{L-1} a_t \phi(t).
\]
where $\alpha_j$ is the expansion coefficient corresponding to the $l$-th basis function. Examples of such signals are truncated Fourier series, wavelets, splines, etc. Without loss of generality, we will only consider the signal on the interval $[0, 1]$ here ($0 \leq t < 1$).

We now take a first set of $N$ uniformly distributed samples (corresponding to the first image) with sampling rate $1/N$ and sampling kernel $\varphi(t)$.

$$y_k(n) = \varphi(t), \varphi(t - \tau_k - \frac{n}{N})$$

Next, we take $K - 1$ other sets of samples (images), with offsets $\{t_k\}_{k=1, \ldots, K}$ with respect to the first set ($t_0 = 0$).

$$y_k(n) = \sum_{k=0}^{K-1} c_t \varphi_t, \varphi(t - \tau_k - \frac{n}{N})$$

The offsets $t_k$ can take arbitrary real values between 0 and 1, and are supposed to have a part that is not a multiple of $L/N$ (subpixel shifts). We can rewrite this in vector notation as

$$y_k = \Phi_t \alpha$$

where $y_k$ and $\alpha$ are column vectors containing the samples $y_k(n)$ and the expansion coefficients $\alpha_j$, respectively. The $K \times N$ matrix $\Phi_t$ consists of the basis functions $\varphi(t)$ sampled uniformly with a sampling kernel $\varphi(t)$, and unknown offset $t_k$. The $K$ sets of samples can be combined in a single vector, resulting in

$$y = \begin{pmatrix} Y_0 \\ Y_1 \\ \vdots \end{pmatrix} - \begin{pmatrix} \Phi_{t_0} \\ \Phi_{t_1} \\ \vdots \end{pmatrix} \alpha = \Phi_{t-k} \alpha$$

This results in $K$ sets of $N$ uniform samples with unknown off-sets between the sets of samples, just like in super-resolution imaging, where one takes $K$ pictures of $N$ pixels each, with unknown camera motion between the images (horizontal and vertical shifts or more complex motion). It can be shown that if the total number of samples $K \times N$ is larger than the total number of unknowns $L + K - 1$ (expansion coefficients and offsets), the problem is well-defined, and has a single solution [11]. We will assume here that this is the case (except in Section 3.3, where no special assumptions are made).

From (5), we can see that this problem is linear in the unknown expansion coefficients or, but non-linear in the unknown offsets $t_k$. Once the offsets are known, we can easily reconstruct the signal co-efficient by solving a set of linear equations. We will therefore concentrate in this paper on the computation of the unknown offsets image registration.

III. SUBSPACE-BASED IMAGE REGISTRATION

In this section, we present a set of algorithms based on a subspace analysis of super-resolution imaging. First, we present a projection based algorithm, followed by a Fourier- and a spline-based algorithm. In the first two algorithms, we assume the sampling kernel is a Dirac (or can be factored out after reconstruction), and place restrictions on the (sampled) signal model. The third method uses splines to explicitly model the sampling kernel, and does not restrict the signal model itself.

A. Projection-Based Algorithm:

From (5), it is clear that our set of images $y$ belongs to the subspace spanned by the sampled basis functions $\Phi_t$. In practice, the exact sampling points of those basis functions are unknown, due to the unknown shifts. If we take an arbitrary set of shift values $\tilde{t} \neq t$ and the corresponding subspace $\Phi_{\tilde{t}}$, it does not contain the set of images $y$ (taken with shifts $t$). We can therefore compute the shift values by minimizing the difference between the sample vector $Y$ and its projection $P_{\Phi_{\tilde{t}}} Y$ onto the subspace $\Phi_{\tilde{t}}$.

$$t = \text{arg min} \| y - P_{\Phi_{\tilde{t}}} y \|_2$$

As discussed in Section 2, once the shifts are known, the signal coefficients (and thus a high resolution image) can be reconstructed by solving the set of linear equations from (5). We compute a least squares solution for increased robustness against noise. Note that this algorithm uses ideas similar to separable nonlinear least squares [12] (which was also used with different descriptions in [9, 10]).

B. Fourier-Based Algorithm:

Let us now assume $f(t)$ can be expressed in a Fourier basis. We can then rewrite (4) as

$$y_k = F^* D_{\tilde{t}} a_k \alpha \quad \ldots \ldots \ldots (7)$$

with $F^*$ an $N \times L$ inverse Discrete Fourier Transform (OFT) matrix, and $D_{\tilde{t}}$ an $L \times L$ diagonal matrix with elements $D_{\tilde{t}} (l, l) = e^{j2\pi l N \tilde{t}}$. The notation $F^*$ is used for the Hermitian transpose of the forward transform matrix $F$. Note that due to the undersampling, $F^*$ is an extension of a square $N \times N$ inverse OFT matrix $F_{\tilde{t}}$, where some columns are repeated, and not a submatrix of an $L \times L$ matrix $F_{\tilde{t}}^L$.

The Fourier transform of (7) can then be written as

$$y_k = F y_k = F^* F^* = D_{\tilde{t}} a_k \alpha \quad \ldots \ldots \ldots (8)$$

The vector $Y k$ is a phase shifted and aliased version of the original Fourier coefficient vector $a$. If we assume the length $L$ of $a$ is a multiple of $N$ (otherwise we can always add zeros to $a$ to make it a multiple), a can be split in $S = L/N$ blocks $a_i$ of length $N$. We can then rewrite the Fourier transform as

$$y_k = F_N (F^*_N F^*_N \ldots \ldots \ldots) D_{\tilde{t}} a_k \alpha \quad \ldots \ldots \ldots (9)$$

with $D_{\tilde{t}} a_k$ the $N \times N$ central part of the matrix $D_{\tilde{t}}$. The subvectors $a_i$ represent the overlapping parts of the spectrum due to under sampling, and the sum is therefore over each of those $S$ overlapping parts of the spectrum.

From (9), we can see that for each set of samples, the vector $D_{\tilde{t}}^{-1} y_k$ belongs to the same S-dimensional subspace spanned by the vectors $a_i$. As the diagonal matrix $D_{\tilde{t}}^{-1}$ depends on $t_k$, we can therefore compute the offsets $\{t_k\}$ as the values for which
\[ \text{rank} \left( Y_0 \ D_{\epsilon^{-1}} \ Y_1 \ ... \ D_{\epsilon^{-1}} \ Y_{K-1} \right) = S \]
\[ ... \ ... \ ... (10) \]

where we require \( K > S = r \ L/ N \).

C. **Spline-Based Algorithm:**

In many situations the sampling kernel \( I/I(t) \) can be estimated or is known a-priori and this knowledge can be used to devise more effective registration techniques.

In the idealized acquisition model of digital cameras, the samples are related to the original view through the point spread function (PSF), which fundamentally models the blur due to the camera lenses and the sensor structure. The PSF of a camera is normally modelled with a Gaussian pulse, however, in this work we assume that the PSF can be approximated with a B-splines function. We make this choice for two main reasons: first, B-splines of sufficiently high order have a shape which is very close to that of a Gaussian function. Second, B-splines possess the polynomial reproduction property we can take advantage of in the registration step.

For those not familiar with polynomial splines, let us just recall that a B-spline \( \beta_L(t) \) of order \( L \) is obtained from the \((L+1)\)-fold convolution of the box function \( \beta_0(t) \), that is:

\[ \beta_L(t) = \beta_0(t) * \beta_0(t) * ... * \beta_0(t), \text{with} \ \beta_0(w) = \frac{1 - e^{-jw}}{jw} \]

Here, \( \beta_0(w) \) denotes the Fourier transform of \( \beta_0(t) \). Moreover, a linear combination of shifted versions of B-splines of order \( L \) can reproduce polynomials up to order \( L \). More precisely, there exist coefficients \( C_m, n \) such that

\[ \sum_{n \in \mathbb{Z}} c_{m,n} \beta_L(t - n) = t^m, \quad m = 0, 1, ..., L \quad (11) \]

Let us now reconsider the sampling problem of the previous section. In this new acquisition scenario, we have that the measurements \( y_k(n) \) observed by the \( k \)-th digital camera are given by:

\[ y_k(n) = \langle f(t), \beta_L(t - t_k - n/\mathcal{N}) \rangle \]

We now observe the following:

\[ \sum_n c_{m,n} y_k(n) \equiv \langle f(t), \sum_n c_{m,n} \beta_L(t - t_k - n/\mathcal{N}) \rangle \]

\[ \equiv \int_{-\infty}^{\infty} f(t) (t - t_k)^m dt \equiv T_{m,k} \]

\[ m = 0, 1, ..., L \quad (12) \]

Where \( \mathcal{N} \) follows from the linearity of the inner product and \( f(t) \) from the polynomial reproduction formula (11). The above equation is therefore showing that it is possible to retrieve the exact moments \( T_{m,k} \) of the signal \( f(t - t_k) \) from its samples. From the moments it is then almost straightforward to retrieve the offsets \( t_k \). In fact we have that

\[ t_k = (T_{1,k} - T_{1,0})/T_{0,0} \]

In the case of two-dimensional signals like images, the transformation that relates two signals can be more complicated than a simple translation. For example, the same object in two different images might be related by an affine transformation. One can show that a moment-based registration as described above for a simple shift is also possible for such more complex motion. Since an affine transformation has six degrees of freedom, more moments are required. More precisely, third order moments along the x and Y axis are needed in order to retrieve the affine transformation.

**a. Local algorithm** - There might be situations where the above approach might become impractical or unstable. In this case, a registration approach based on local features might be more convenient. Many registration algorithms are based on corner detection and then on the matching of the corners of two different images in order to retrieve the transformation between the two images.

In this work, we propose to locate corner points at the intersection of two straight edges. A single edge is parameterized by its angle \( \theta \), height \( \zeta \) and shift \( \gamma \). Such parameters can be exactly retrieved from the samples using the moment of the derivative of the original function. More precisely, denote with \( z(n) = y_{d,n} - y_{d,n-1} \), one can show the following:

\[ z(n) = \langle \frac{df(t)}{dt}, \beta_{L+1}(t - n/\mathcal{N}) \rangle \]

Namely, the new samples \( z(n) \) are equivalent to those obtained by sampling the derivative of \( f(t) \) with a B-spline of order \( L + 1 \) rather than \( L \). It is then possible to retrieve the step parameters from the new samples \( z(n) \). The complete solution for a single step edge is given by:

\[ \zeta = T_{0,n}, \quad \tan \theta = \frac{T_{1,n}}{T_{1,n+1} - T_{1,n}}, \quad \gamma = \frac{(n+1) T_{2,n} - n T_{2,n+1}}{T_{2,n}} \]

where in this case \( T_i, n \) indicates the \( i \)-th order moment of \( df(t)/dt \) along the \( n \)-th row of the sampled image. The derivative \( df(t)/dt \) is computed in discrete domain using discrete differences. The validity of the above schemes will be assessed in the next section.

IV. **RESULTS**

We now apply the presented algorithms on various sets of input images. For the algorithm from Section 3.1, we sub sampled an image of 31 x 31 pixels (without pre filtering, but after applying real-valued shifts), generating 5 randomly shifted and aliased input images of 16 x 16 pixels. One such image is shown in Figure 1 a, and the reconstructed image is shown in Figure 1 b. The shift values are estimated up to a precision of \( 10^{-16} \) (Matlab precision). Similarly, we tested the algorithm from Section 3.2 on a set of 5 shifted and aliased input images of size 32 x 32, generated from a 63 x 63 image. The results are shown in Figure 2. Again, motion parameters are estimated up to working precision.

![Figure 1: Results using algorithm from Section 3.1. The high resolution image (b) is perfectly reconstructed from 5 shifted low resolution images (a).](image-url)
For the algorithm from Section 3.3, we consider a set of real images as they are acquired by a digital camera. The registration approach considered here is based on continuous moments. Since it takes a sampling point of view, image samples should be modified as little as possible by internal post-processing occurring in a digital camera after acquisition. The set of images is thus acquired by a SLR digital camera (a Nikon D70s) in RAW format. The experiment is presented in Figure 3. Sixty pictures of the moon are taken with a digital SLR camera and a lens with a focal length at 38mm (35mm equivalent: 57mm) and settings: F16, 1/60s, ISO 200. The PSF in this case is not estimated and is directly approximated with a cubic B-spline at scale 1. The MRNSD algorithm (modified residual norm steepest descent) is used as restoration method [13]. Figure 3a shows the moon as acquired by the camera and Figure 3b presents the obtained super-resolved image where details of the moon can be observed.

From the above figures, we can see that all three methods give considerable improvements in image resolution. They also require high computational power. For the algorithms from Sections 3.1 and 3.2, only computer simulations were shown where the original image is perfectly reconstructed. While the memory requirements are higher for the algorithm in Section 3.1, the algorithm from Section 3.2 requires a slightly larger number of input images (K > S = [L/N] instead of K > (L - 1)/(N - 1). The algorithm from Section 3.3 is tested in an experiment using real images from a digital camera as input. The result can therefore only be evaluated visually. It does not put specific requirements on the minimum number of input images and performs pair wise registration of the images.

V. CONCLUSIONS

We have presented a common framework for some recent subspace based image registration methods for super-resolution imaging. First, we have shown that super-resolution can be described as a multi-channel sampling problem with unknown offsets. Three different solution methods using subspace descriptions were then described: one using ideas from variable projection theory, one using a Fourier analysis of the aliased signals, and finally one based on a spline analysis of the sampling kernel. The performance of the different algorithms is illustrated in numerical simulations.

VI. REFERENCES


Multi-Level Chronological Function algorithm for Tracking Multiple Generic Objects in a Motion Picture

1D.V. Chandra Shekar  
Asst Prof, Dept of Computer Science, T.J.P.S College,  
P.G Courses, Guntur, chand.info@gmail.com  
Andhra Pradesh-522006, India

2Y. Suresh Babu  
Asst Prof, Dept of Computer Science  
J.K.C College, Guntur, Andhra Pradesh – India

3Dr. G. S. Prasad, Prof. Dept of CSER.V.R. & J.C. College of Engineering, Chowdavaram, Guntur - 522 019

Abstract—We propose a Multi-Level path scheme for the class of multiple object tracking problems where the inter-object interaction metric is convex and the intra object term quantifying object state continuity may use any metric. The proposed scheme models object tracking as a multi-path searching problem. It explicitly models track interaction, such as object spatial layout consistency or mutual occlusion, and optimizes multiple object tracks simultaneously. The proposed scheme does not rely on track initialization and complex heuristics. It has much less average complexity than previous efficient exhaustive search methods such as extended Generic Multi-level path programming and is found to be able to find the global optimum with high probability. We have successfully applied the proposed method to multiple objects tracking in video streams.

Keywords-component; formatting; style; styling; insert (key words)

I. INTRODUCTION (HEADING 1)

Tracking multiple objects simultaneously is key for many vision applications, such as visual navigation and object activity recognition. Even though each object can be tracked separately, tracking objects together is important for obtaining good results if objects have complex interactions [1]. We categorize object interactions into two classes. The first type of interaction constrains the object relative locations, i.e., objects tend to keep relative positions or spatial layout during a short period of time. The second type of interaction is object mutual occlusion, i.e., an object in front occludes other objects in the same region.

Explicitly modeling interaction of objects enables tracking multiple objects more robustly, especially in cluttered environments. But, the search space also increases drastically compared to that of tracking objects separately. Naive exhaustive search becomes intractable. Efficient exhaustive searching schemes such as extended Generic programming function [1] are still too complex to be applied to problems with a medium number of observations and objects. We propose a linear programming relaxation scheme for a specific class of multiple object tracking problems, in which the metric for inter object position interaction term is convex while the intra object terms quantifying object state continuity along time may use any metric. The proposed scheme explores a large search space efficiently and almost always gives a global optimum because of the special structure of the formulation. Multiple object tracking has been studied intensively. For example, Kalman filtering has been a classic scheme for object tracking. Recently, particle filtering has been popular for tracking multiple objects such as ants [2] with complex interactions. Particle filtering has also been studied for tracking hockey players [3] in which object interaction is not explicitly modeled. Bayesian networks have been applied to optimizing trajectories of football players in video [5]. This approach does not consider track interaction among objects.

Generic programming function (GENERIC FUNCTION) is also widely applied to multiple object tracking. The single chain Viterbi algorithm can be extended [1] to optimize multiple tracks simultaneously. The computational complexity of extended Generic function is \(O(mnk^2n)\), where \(k\) is the number of observations at each stage, \(n\) is the number of objects and \(m\) is the length of the sequence. Extended generic Function is thus hard to apply to large scale problems. An efficient approximate Generic programming function scheme [4] has been studied to find a single object’s path with heuristics used to determine the sequence of path assignments in a multiple-camera setting. While simple heuristics such as best-track-first assignment works well for multiple camera tracking, it does not always give correct solutions when objects have complex mutual occlusion patterns, especially for single camera applications. Linear programming (Chronological Programming) is another approach that can be used for more efficient search in object tracking. Optimizing object tracks using 0-1 Integer Programming [6] has been studied for radar data association. This formulation is different from the proposed scheme in that a variable is defined for each feasible trajectory and object tracking is solved as a set packing problem. Other approximation methods for solving similar integer Chronological Programming formulations as [6] are studied in [7, 8], which turn out to be quite similar to the sequential generic method [4]. Unlike previous Chronological Programming methods, our proposed scheme is based on a multiple-shortest-path model that tries to connect edges into paths and has much fewer variables. Belief Propagation (BP) [9] has also been used for optimizing hand tracking. Occlusion is explicitly modeled in this method. However, multiple object tracking results in a loopy graph structure making it difficult to guarantee convergence to a global optimum. Even though intensively studied, robust and
efficient tracking of multiple objects with complex interactions remains unsolved. In this paper, we propose a novel linear programming relaxation scheme to optimize multiple object tracks simultaneously by explicitly modeling spatial layout constraints and mutual occlusion constraints. We formulate object tracking as a multipath searching problem. Each path is composed of a sequence of states, e.g., locations and appearances, of an object through time represented by nodes in a graph. Different tracks are constrained so that objects cannot occupy the same spatial region. Convex penalty terms are included to constrain the consistent objects’ layout in space, i.e., the objects’ relative positions do not change abruptly from frame to frame. The state continuity metric term along time may use any metric. Based on the special structure of our formulation, a linear programming relaxation approach effectively solves the path searching problem when paths overlap and objects occlude each other. As our results illustrate, the linear program almost always yields integer solutions that globally optimize object tracks and has low order polynomial average complexity.

2. Multiple Object Tracking

In this section, we describe our Chronological programming based method for optimizing multiple object tracks in continuous video frames. Intuitively, at each frame we represent all the possible spatial locations of each object from the observations as nodes based on attributes of the objects. (In our examples, we determine possible bounding boxes for objects’ locations based on background subtraction or appearance characteristics of objects. These bounding boxes are also used to determine what it means for one object to occlude another.) Over a window of frames, these nodes form a graph where a path connecting nodes represents a possible spatial trajectory of an object over time in the video. This is represented in Fig. 1. However, if one object occludes another, there is a break in the track of one object. We have a special occlusion node that allows the path for an occluded object to be accounted for in that particular frame if there is no other non-overlapping location for the potentially occluded object. This graph forms the basis for formulating a cost function based on all the possible paths and constraints, leading to a linear program that may be efficiently solved. The algorithm optimizes the states for all the objects together. Thus, it finds consistent paths for all the objects over a window of video frames and assigns a meaningful interpretation of location or status of occlusion to each object as described more formally below.

2.1. Problem Statement

In multiple object tracking, we need to locate objects (positions, poses etc.) through a sequence of video frames. For each video frame, we assume that there is a set of observations for each object, which are obtained by using methods such as background subtraction or template matching. These observations are not reliable and may contain many false positives. Misdetection of an object may also occur. We wish to obtain object locations in a sequence of video frames based on the assumption that an object usually does not change appearance and location abruptly. Apart from finding the correct trajectories for all the objects, we also need to determine whether an object is visible in a video frame: objects may disappear due to occlusion or moving out of scene.

2.2. Network Model

In the following, we study multiple object tracking based on a network model in which sub-models in our formulation interact with each other. This approach contrasts with The previous trellis model used in single-chain Generic programming function.

Figure 1. The network model for multiple object tracking. programming.

In Fig. 1, an object’s possible location and appearance states are represented as round nodes. For a given frame, hypothesized locations (i.e., observations) for each object may be different, and therefore the sub-network for each object may contain a different number of nodes. The rectangular nodes in Fig. 1 are the occlusion nodes that provide a node to represent that an object is occluded and does not have a spatial location. A source node and a sink node, shown as diamond nodes in Fig. 1 are also included for each object sub-network to represent the start and end of the object tracking sequence. Sink nodes are included just for convenience; they do not correspond to states of objects. The solid arcs between nodes indicate possible state transitions. A connected set of nodes between a source and sink node represents the spatial trajectory of an object. We also model mutual occlusion among objects in the network. A spatial conflict set is defined for each node in the network. Nodes in a spatial conflict set correspond to object states occupying the same spatial location. As shown in Fig. 1 the spatial conflict set for node Vn, m, i includes the node itself and nodes in the ovals in the other objects’ sub networks that would overlap the region of Vn,m,i. Note that the occlusion node for each object never has a spatial conflict, so it will never be in a spatial conflict set. Only one node in a spatial conflict set may be selected for connecting an object path as this represents the
visible object at that location in space. Once a node is selected for one of the objects, all the other objects must either select a node that includes a different spatial location for that frame or the occlusion node. The above condition is defined as the object mutual occlusion constraint. We also include a spatial layout constraint for all the objects. This is defined in the network model to constrain objects’ relative locations at each time instant. Multiple object tracking can thus be modeled as finding optimal paths from the source nodes to the sink nodes for all objects, which satisfies the object interaction constraints. We use the following notation to precisely define the problem in an Chronological Programming framework.

For object \( n \), its source node is denoted as \( s_n \) and its sink node as \( t_n. \) \( s_n \) corresponds to the location and appearance of object \( n \) in frame 0. The source node also provides an initial template node for computing trajectory costs as described below. For each video frame, we insert nodes corresponding to all the observations of object \( n \) at each time instant together with an occlusion node. \( V_{n,m,i} \) denotes the node indicating that object \( n \) is assigned state \( i \) in frame \( m. \) The occlusion node is always the node with the largest state number \( i. \) The source node \( S_n \) is also denoted as \( V_{n,0,0} \), and the sink node \( m \) as \( V_{n,M+1,0} \), where \( M \) is the length of video sequence. We connect nodes in successive frames with arcs as shown in Fig. 1 using a fully connected pattern. For most applications, partially connected patterns can also be used to simplify the problem based on heuristics, for example, that objects do not move far between successive frames. A cost \( c(V_{n,m,i}, V_{n,m+1,j}) \) is assigned to each arc, which indicates the cost of state \( i \) at time \( m \) and state \( j \) at time \( m+1 \) being on the trajectory of object \( n. \) The cost function can be convex or non-convex. An arc’s cost usually contains two parts: the cost of choosing a state at a time instant and the cost of state transition from \( i \) to \( j. \) In this paper, the cost of arc connecting node \( V_{n,m,i} \) and \( V_{n,m+1,j} \) is defined as

\[
c(V_{n,m,i}, V_{n,m+1,j}) = \begin{cases} 
\lambda_1 \cdot g(v_{n,m,i}, v_{n,m,j}) + \lambda_2 \cdot d(v_{n,m,i}, v_{n,m,j}), & \text{if both nodes are non-occlusion nodes and not occlusion nodes and not sink nodes.} \\
\delta_0 \cdot \delta_{const} + \delta_1 \cdot \delta_I, & \text{otherwise.}
\end{cases}
\]

Appearance corresponding to nodes in the network, e.g., by comparing color histograms in bounding boxes; \( d(.) \) computes the spatial distances of two states, e.g., the distance of two bounding boxes. \( \lambda_1 \) and \( \lambda_2 \) are constant coefficients to control the weight of temporal smoothness. \( \delta_0 \) and \( \delta_{const} \) are constant costs penalizing when an object disappears or reappears. Thus, if an arc leads into an occlusion node or a sink node, it bears a constant cost. The cost of an arc from an occlusion node to a non occlusion node includes the similarity measurement of the destination node to the template object (the source node) plus a constant. When both of the nodes are non-occlusion nodes, the edge connecting the nodes has weight equaling the summation of three terms: the similarity of the target node to the template object, the appearance similarity of detections in two successive frames and a term that penalizes large spatial displacement between video frames.

In modeling the object occlusion constraint, we need to specify the spatial conflict set for each non-occlusion node \( V_{n,m,i}. \) The spatial conflict set for node \( V_{n,m,i} \) is denoted as \( O(V_{n,m,i}) \) which includes \( V_{n,m,i} \) and nodes from other sub-networks whose regions are highly overlapping with the region of node \( V_{n,m,i}. \) To determine whether nodes are included in a spatial conflict set, we consider two types of overlapping regions. The first one includes partially overlapped regions as shown in Fig. 2 (a). The second one includes completely overlapped regions as shown in Fig. 2 (b). There are multiple approaches to determine whether to include a node in the spatial conflict set. For example, one approach uses the probability of two bounding boxes overlapping. This probability is calculated using the ratio of the overlapping area to the average area of the rectangular regions. If the ratio is sufficiently large, the two regions cannot be visible at the same time and nodes corresponding to these regions are in the same spatial conflict set. Another approach uses a simpler measurement based on the total city-block distance of the 4 corners of the two bounding boxes. In this case, if the difference is below some threshold, then the two bounding boxes are overlapping and the nodes should be included. If the difference is large then either the objects are not overlapping or the size of two objects is very different and the corresponding nodes do not belong to a spatial conflict set. We use this latter approach in our examples. Apart from the occlusion constraint, we also would like to keep the spatial layout of objects stable over a short period of time. To model this constraint, we keep the spatial displacement vectors between objects as similar as possible.
across time. As shown in Fig. 3, the vectors from object \( n_2 \) to object \( n_1 \) tend to remain unchanged at time instant \( m \) and instant \( m+1 \), i.e., \(|(P_{n_1,m+1} - P_{n_2,m+1}) - (P_{n_1,m} - P_{n_2,m})|\) tends to be a small number. In fact, vector \( p \) can be more than 2D. For example, \( p \) can be a 4D vector representing the 2 corners of bounding boxes. This second constraint is a soft one and implemented as a regularization term in the objective function.

### 2.3. Discrete Optimization

An energy function for optimizing object tracks can thus be written as follows

\[
\min_{p_{n,m} \in P_{n,m}} \sum_{m=1}^{M} \sum_{i \in N} c(e_{n,m-1,m+1,j}) + \mu \sum_{m \in M} \sum_{i \in N} ||(P_{n_1,m+1} - P_{n_2,m+1}) - (P_{n_1,m} - P_{n_2,m})||
\]

s.t. at most one path goes through \( O(V_{n,m,i}) \), \( \forall V_{n,m,i} \) where \( P_{n,m} \) is the location of object \( n \) at time instant \( m \). For instance, if we use bounding boxes to quantity.

Identify the location of an object, \( P_{n,m} \) is a 4-element vector \((P_{n,m,1}, P_{n,m,2}, P_{n,m,3}, P_{n,m,4})\) in which \((P_{n,m,1}, P_{n,m,2})\) is the top-left corner \( x-y \) coordinate of the bounding box and \((P_{n,m,3}, P_{n,m,4})\) is the right-bottom corner \( x-y \) coordinate. \( N \) is the set of neighboring objects. \( \mu \) is a coefficient to control the weight of the spatial layout regularization term. In this paper, we assume all the object pairs are neighbors, i.e., \( N \) contains all the object pairs. We assume that the norm \(||.||\) is the \( L_1 \) norm. Using the \( L_1 \) norm enables us to relax the optimization into a simpler linear program. In fact, the \( L_2 \) norm can also be used and the relaxation is a quadratic program which can also be efficiently solved. In the following, we use the \( L_1 \) norm and Chronological Programming relaxation to illustrate the concept. Because of path interaction, searching algorithms need to consider all the paths simultaneously and thus have to search a large space. Naive exhaustive search is not an tractable option. This optimization problem has convex \((L_1)\) inter-object regularization terms, while the intra-object regularization term embedded in the arc cost may use any metric. As shown in the following section, this type of problem can be relaxed into a convex program that can be efficiently solved.

\[
Y_{n,m,i} = \sum_{j=0}^{K(n,m-1)} \xi_{(n,m-1,j), (n,m,i)}
\]

where \( Y_{n,m,i} \) is the summation of \( \xi \) corresponding to all the incoming arcs of node \( V_{n,m,i} \). Let \( K(n,m-1) \) be the number of nodes for object \( n \) at time \( m-1 \), \( Y_{n,m,i} = K(n,m-1) + 1 \). \( \xi_{(n,m-1,j), (n,m,i)} \) indicates whether node \( V_{n,m,i} \) is on the path of object \( n \). In the ideal case, \( Y_{n,m,i} \) will be 1 if the node is on the path and 0 otherwise. Object location is represented with variables \( p \). \( P_{n,m,l} \) is the \( l \)th element of the location of object \( n \) at time \( m \). \( P_{n,m,l} \) equals the linear combinations of observations with coefficients \( Y_{n,m,i} \). Fig. 4 illustrates these notations with a simple case. Based on the energy function defined, the cost of a path is thus the linear combination of edge costs plus an \( L_1 \) norm regularization term. By introducing non-negative auxiliary variables, we can further turn the \( L_1 \) norm terms into linear functions. The path finding can therefore be relaxed into the following linear program:
In the above equation, \( R_{n,m,i} \) is the location vector, e.g., bounding box coordinates, corresponding to node \( V_{n,m,i} \). Occlusion nodes correspond to a special location, e.g., zero size bounding box at the center of an image. \( p + n_1,n_2,m,l \) and \( p - n_1,n_2,m,l \) are non-negative auxiliary variable pairs, which are used to turn the L1 norm smoothness term into a linear function. We use a standard Chronological programming trick \[10\] to convert an absolute value term into a Chronological function. In the constraint, the difference of the auxiliary variable pair \( p + n_1,n_2,m,l \) and \( p - n_1,n_2,m,l \) equals the location vector difference of two neighboring objects, for which we would like to compute the absolute value. When the linear program is finally optimized, at least one of the auxiliary variables in each pair will be zero. Otherwise, we can always subtract the smaller one of the pair from each variable and get a feasible solution with smaller objective function and one variable in the pair becomes zero, which contradicts the optimum solution assumption. Therefore the sum of the auxiliary variables in the objective function equals the absolute when the Chronological PROGRAMMING is optimized. The Chronological program is equivalent to the original discrete optimization if the linear cost term equals the original cost term, which will be the case if \( \xi \) are further constrained to be 0 or 1. The linear program is thus a Chronological approximation or relaxation of the discrete optimization problem. The first three constraints set out the unity flow continuity constraints that are necessary conditions for the solution to be a path for each object. The constraint on \( y \) guarantees that no two paths go through the same spatial conflict set, i.e., if one path goes through a position other tracks tend to pass these positions will be occluded. The spatial conflict set is also illustrated in Fig. 4.

If we constrain the variables of \( \xi \) to be 0 or 1, the integer program exactly solves the multiple object tracking problem. We drop the integer constraint and obtain a Chronological programming relaxation which can be solved efficiently. There is no guarantee that the linear program always gives integer solutions for \( \xi \). For real problems, most of \( \xi \) are indeed 0s or 1s and therefore gives the globally optimized solution. As shown in the experiments, the linear program has a high probability of directly giving the global optimal solution. The simplex method for Chronological programming has exponential complexity in the worst case. Linear programming is fast for real applications \[10\]: for our Chronological PROGRAMMING formulation, its average complexity is approximately \( O(n^2km)(2\log(k) + 2\log(n) + \log(m)) \), in which \( k \) is the number of observations for each object, \( n \) is the number of objects and \( m \) is the number of frames in optimization. In comparison to extended Generic function, the linear program has much lower average complexity.

**Proposed Algorithm**

**Algorithm of the Boundary Detection of Multiple Objects**

The algorithm of the boundary detection of multiple objects is shown as follows:

- **Step 1**: Convergence process: calculate the energy functions and minimize the energy terms of pipee points. If the iteration reaches the final step, stop. Otherwise, go to step 2.

- **Step 2**: The process of determining intersection: if the pipe point intersects segment \( S_i \) estimated by the equation (11), then go to step 3. Otherwise, go to step 1.

- **Step 3**: The splitting and connecting process: split the contour by removing the unnecessary point \( v_k \). pipe points, which belong to the same side, are connected by equation (12) and then, go to step 4.

- **Step 4**: Reorganizing the sequence of the pipe point process: a new sequence is formed for each contour. Go to step 1. The procedure of the proposed method for the detection of the boundaries of multiple objects.

**Algorithm for tracking**

Merging path algorithm for tracking Multiple Objects in video Stream

Construct various path by preparing objects maps \( G \) for an input video stream from equation 12

\[
P_1 \leftarrow \text{prepare virtual link path(from G, start, minpointk)} \text{from video from equation 12 a||} \text{calculate the pipe function for calculating minimum pipe points \text{repeat the iteration reaches the final stage, stop , otherwise go to step 3}}
\]

\[
P_1 \leftarrow \{p1*\} \text{do {if check for existence of virtual trace from various point in pipe flow for existence of objects and non-objects, by check the various virtual tracks wrt table 4||} Determine the pipe trace point for intersects segment by the estimation of equation 11 goto step 4 otherwise step 2}
\]
Transform virtual trace pipe points wrt equation 15 and gain shortest trace pipe point from \((G, V_{\text{trace}}, \text{pipe points})\) | | split the pipe points by removing the un-necessary point \(V_k\), pipe point, which belong to the same side, are connected by equation 12 and then, go to step 4

Interlacing \((P1)\) and \(P_{i+1} \leftarrow p_{1Up}^*\) re-organize the virtual trace point | | reorganizing the sequence of the pipe point process and create a new sequence from the contour go to step 1

Example 1: To illustrate how our approach works we track 2 objects in 340 consecutive video frames. We assume that object histograms are known. At each time instant, potential object locations are detected as bounding boxes. Each bounding box is represented using a 4-element vector representing 2 opposite corners. Spatial conflict sets are then determined for each bounding box. In this example, all the bounding boxes detected are candidates for object 0 or 1, hence, the sub-networks for each object are the same. Grayscale color histograms with 64 bins are used as the features for object appearance identification. In this example, a neighboring set only contains one pair \(\{0,1\}\). We build a linear program for this problem based on our proposed Chronological Programming relaxation scheme. The Chronological Programming takes 4628 simplex iterations. Values of \(p\) give locations of objects. If the value of \(y\) at an occlusion node is greater than 0.5, the object is set occluded at the time instant. The tracking result is shown in Fig. 5. The top-left corner \(x\) and \(y\)-coordinate of the bounding boxes for both objects are shown in Figs. 5 (a) and (b). For this example, Chronological PROGRAMMING relaxation has integer solutions for \(\xi\) and therefore achieves its global optimum. As shown in Fig 5 the object paths are quite good for both \(x\) and \(y\) coordinates even when the objects overlap each other. As a comparison, we apply Generic function with best-track-first assigned heuristics to the same data. The energy function of Generic Function is the same as the proposed scheme except for the spatial layout consistency term. Approximate generic Function not easily extended to include such regularization terms since it optimizes each track separately and then assigns tracks sequentially. Fig. 6 shows the tracking result of approximate Generic Function. In this example, generic function first picks the object 0 track as a better fit and determines the track for object 1 after removing assigned boxes for object 0. As shown in this example, greedy track assignment selected wrong labels at the first and third occlusion instances. Simply reducing the occlusion label cost will not solve the problem and it also causes many missed detections

2.5. Online Multiple Object Tracking

We have studied an Chronological Programming based method to track multiple objects by optimizing tracks in a sequence of video frames. This scheme can be extended to online video tracking by applying the tracking scheme as a moving window filter. For our long video sequences we use a video segment window size of between 15 to 300 frames with 1 frame overlapping between segments. An object list keeps the histogram of object templates. The locations of object templates are also updated at the end of each video segment. Objects can also be detected automatically for background subtraction based object tracking. If we find a consistent object which is not on the track of previous video segment, we insert it into the object list. The consistency is measured by a backward and forward testing approach based on the proposed
tracking scheme. We check the duration of visibility and the cost of track in backward and forward tracking. If a new object has track cost lower than a threshold and appears in more than 75% of the testing period, it is inserted into the template list.

Figure 6. Tracking result using approximate Generic function for Example 1

3. Experiment Results

We report our results using our method for tracking multiple objects on 4 different video sequences. These video sequences are in CIF format with frame rate 15–30 frames/second.

3.1. Tracking Two Stuffed Animals

Fig. 7 shows the tracking result of the proposed method for a 307-frame video. 2 toy objects are tracked through the video frames. There are complex occlusions between the two objects. The templates for the two objects are set using the first video frame. A sub-image is used as the feature in tracking. Object observations are obtained at local peaks of the template matching map. Approximately 80 detections are found for each object in each video frame which appear as nodes in the graph providing many path possibilities.

In this experiment, Chronological Programming optimizes each 20-frame segment including the template frame in a sliding window fashion. Despite complex occlusions, the proposed method tracks the objects correctly along the video sequence.

3.2. Tracking Fast Moving Squash Players

In another experiment, we apply the proposed scheme to a 1351-frame squash video sequence with 2 players as shown in Fig. 8. The candidate objects are detected by background subtraction similar to method used in [4]. The video includes complex object interaction and mutual occlusion. Noisy background subtraction also makes object tracking a hard task. In this experiment, we convert color image into grayscale and use a rough 64-bin histogram as features. The proposed Chronological programming relaxation is then applied to the video sequence in sliding window fashion the same as the first experiment. The proposed scheme accurately follows the object

Object locations through the video sequence, Fig. 8 illustrates sample frames of the tracking result and Fig. 9 shows the object locations at each time instant through time (occluded objects are not shown). In the 1351-frame video sequence, object 0 has 7 wrong label assignments and object 1 has 5 wrong detections. The average object tracking precision is about 99% for this example. Chronological PROGRAMMING also has a high probability of directly obtaining the global optimal solution. Only 3 segments do not have fully integer solutions for ξ in 75 video segments.

3.3. Comparison with Generic function on Tracking Three People

Walking in an Office Fig. 10 and Fig. 13 show the result of tracking three objects with the proposed method for a 2431-frame video. In this experiment, we use background subtraction to detect bounding boxes for potential object locations. The features of objects are grayscale image histograms with 64 bins inside a bounding box. Bounding boxes detections are noisy because of the large compression ratio of the video and complex object interaction. The scales of bounding boxes are also not accurate, which results in large portions of the background inside some bounding boxes. The sliding window setting is the same as previous experiments. Objects are automatically detected in this example using the method in Sec. 2.5.

Objects are detected in this example using the method in Sec. 2.5. The proposed scheme can deal with complex occlusions and objects moving out of the scene and coming back. Object 0 has 5 wrong detections, object 1 has 22 wrong detections and object 2 has 125 wrong detections. Overall the accuracy rate is 94% per frame. In this experiment, 4 segments do not have fully integer solutions for ξ in a total of 135 video segments.

To compare methods, we apply Generic Function each single person with exactly the same network weight settings. The result is shown in Fig.11. Because no object interaction constraint is enforced, generic function often assigns different labels to the same object and sometimes fails to locate an object in the scene. Simple heuristics do not always give the correct solution.

Generic Function with best-track-first assigned heuristics has 67, 37 and 319 wrong tracking errors for object 0, 1 and 2 respectively. The accuracy is 83% per frame. Fig. 12 shows sample video frames where the Chronological PROGRAMMING approach improves the tracking result.
Tracking 4 Players in a Double-Squash Game

In Fig. 14, we applied our method to a 500-frame double squash video sequence. There are four objects in the video and there are about 10 detections in each frame. The players in the same team wear the same clothing. In this experiment, we use the proposed scheme to optimize tracking in the whole video sequence rather than shorter segments. We would like to obtain a global optimal solution considering only the occlusion constraint. We use a basic branch and bound method to obtain the global solution. Our method finds the global optimal in 3 minutes using a 2.6GHz PC which is much faster than extended Generic function which needs about an hour to compute the result. Since CHRONOLOGICAL PROGRAMMING solution is very near the global optimum, branch and bound converges very soon. We use branch and bound method here to obtain a global optimum so that we can have a fair comparison with extended Generic Function.

Figure 7. Tracking 2 toy objects with the proposed scheme. Selected frames from 307 frames

Figure 8. Squash. Selected frames from 1351 frames.

As shown in Fig. 14 and Fig. 15, the tracker works well in following multiple objects during a long sequence. In Fig. 15, when objects are occluded, their spatial locations are set to (1,1), which are shown as abrupt drops in the curves. Even though we obtain a global optimal solution, the result is not perfect. Sometimes errors occur for dark team players (player 0 and 2) when the two players occlude each other and cause their identities to be exchanged. Such errors happen due to both unreliable bounding box detection using background subtraction and occlusion between objects with very similar appearance. Fig. 16 shows typical average running times of the linear program using a 2.6GHz PC. Random observations and color histograms are generated in each frame. Each experiment

Figure 9. Object locations for 2 squash players. (a): X-Locations of objects; (b): Y-Locations of objects.

Figure 12. Sample frames where Generic Function with simple heuristics does not yield correct solution while the proposed scheme does. The first row shows sequence generic Function frames. Second row shows results with the proposed method.

Figure 13. Objects locations for 3-people tracking. (a): XLocations of objects; (b): Y-Locations of objects.

The equations are an exception to the prescribed specifications of this template. You will need to determine whether or not your equation should be typed using either the Times New Roman or the Symbol font (please no other font). To create multileveled equations, it may be necessary to treat the equation as a graphic and insert it into the text after your paper is styled.
4. Conclusion

In this paper, we propose a novel framework for optimizing multiple object tracking that can be solved efficiently based on a linear programming relaxation. The proposed scheme explicitly models track interaction such as the spatial layout constraint and object mutual occlusion. Experiments show that the proposed scheme works robustly in tracking objects with complex interactions in long video sequences. The linear program relaxation can also be solved more efficiently than previous methods such as extended Generic programming function. Thus, we believe our approach provides a useful method for multiple objects tracking in video sequences.

References


AUTHORS PROFILE

D.V. Chandra Shekar, received Master of Engineering with Computer Science & Engineering from ANNA University, Chennai, Tamilnadu, India. He is currently working as Associate Professor, in the Department of Computer Science, T.J.P.S COLLEGE (P.G COURSES),Guntur, which is affiliated to Acharya Nagarjuna University. He has 12 years teaching experience and 1 years of Industry experience. He has published 30 papers in National & International Journals.

Y. Suresh Babu, working as Associate Professor, in the Department of Computer Science, J.K.C COLLEGE,Guntur, which is affiliated to Acharya Nagarjuna University. He has 18 years teaching experience. He has published 8 papers in National & International Journals.

Dr. G. Satyanarayana Prasad received Ph.D. degree in Computer Science in the Faculty of Engineering in 2006 from Andhra University, Andhra Pradesh. He completed is MS in Computer Science & Engineering from A & M University,USA,1984. He is currently working as Professor in the Department of Computer Science and Engineering, R.V.R & JC Engineering College, Guntur. His current research is focused on Image Processing. He has published several papers in IEEE journals and various National and International Journals.
Generic Objects Classifying and Segregate Multiple Objects Motion Picture

1D.V. Chandra Shekar
Asst Prof, Dept of Computer Science, T.J.P.S College,
P.G Courses, Guntur, chand.info@gmail.com
Andhra Pradesh-522006, India

2Y. Suresh Babu
Asst Prof, Dept of Computer Science
J.K.C College, Guntur, Andhra Pradesh –India

3Dr. G. S. Prasad, Prof. Dept of CSER.V.R. & J.C. College of Engineering, Chowdavaram, Guntur - 522 019

Abstract— Video object (VO) extraction is of great importance in multimedia processing. In recent years approaches have been proposed to deal with VO extraction as a classification problem. This type of methods calls for state-of-the-art classifiers because the performance is directly related to the accuracy of classification. Promising results have been reported for single object extraction using Support Vector Machines (SVM) and its extensions. Multiple object extraction, on the other hand, still imposes great difficulty as multi-category classification is an ongoing research topic in machine learning. This paper introduces a new class of multi-category learning scheme for multiple VO extraction, and demonstrates its effectiveness and advantages by experiments.

Keywords
VO extraction, multiple object tracking, Ψ-learning, Support Vector Machines (SVM), multi-class classification

I. INTRODUCTION

Video object (VO) extraction, the process of segmenting and tracking semantic entities with pixel-wise accuracy [1], is an important yet challenging task for content-based video processing. For the purpose a great deal of approaches have been proposed [2]–[10], which provide satisfactory results for extracting VOs of homogeneous motion characteristics. Unfortunately, dealing with VOs with abrupt motions or occlusions remains a challenge. In recent years classification based approaches have been proposed to meet the challenge by handling object tracking as a classification problem [11]–[13]. Each VO is considered as a class, and VO extraction is achieved by classifying every pixel to one of the available classes. By doing so temporal associations of objects between frames are automatically maintained through correct classifications which is therefore motion-assumption free. As a result, the approaches are more robust to complicated motion fluctuations. What learning algorithm to use is key to the success of the classification-based approaches. By using powerful classifiers high classification accuracy can be achieved which leads to better performance for VO extraction. However, most of the results reported are limited to single object scenarios. In other words, only binary classification between the object and the background has been tackled. At the first glance, the extension from single object to multiple object extraction is straightforward since conceptually one only needs to replace the binary classifier with a multiclass classifier. Unfortunately, the implementation of such an extension is far more difficult since it appears that multi-category classification is still an ongoing and immature research topic itself in machine learning. Only recently have works emerged to offer new tools that can help tackle the multi-object problem. This work presents an attempt of such. Over the last decade, margin-based classification technologies for which the best known example is SVM [14] have drawn tremendous attention due to their theoretical merits and practical success. Instead of directly estimating the conditional probabilities, the margin-based classifiers focus on the decision boundary which, however, makes it difficult to generalize their applications from binary to multi-class scenario. “Single machine” and “error correcting” are two mainstreams for multi-class margin-based classification. As its name suggests, the “single machine” type of approaches attempts to construct a multi-class classifier by solving just a single optimization problem [15]–[19]. On the contrary, the “error correcting” type of approaches [20]–[21] works with a collection of binary classifiers, for which the primary goal is to determine what binary classifiers should be chosen to train and how to combine their classification results to make the final decision. Among all the methods published in the literature, “one-against-all”, “one-against-one” and directed acyclic graph (DAG) [22] are most popular choices in solving real-world problems. A good overview of multi-class classification can be found in [23] and [24].

As a natural extension of binary large margin classification, the “single machine” type of approaches is intuitively appealing. It has drawn even more attention when certain formulations are reported to yield classifiers with consistency approaching the optimal Bayes error rate in the large sample limit [25]. Multi-class Ψ-learning is such a learning algorithm [26]. Moreover, Ψ-learning aims directly at minimizing the generalization error (GE), which is the reason why its binary version has shown significant advantage over SVM in terms of generalization both theoretically and experimentally [27]. The extended multi-class Ψ-learning retains the desirable properties of its binary counterpart. In addition a computational tool based on the recent advance in global optimization has been developed to reduce the time of training for the “single machine” [28]. The purpose of this paper is twofold. First, it introduces multi-category Ψ-learning [26] to tackle the multiple VO extraction problem. Secondly, it reports the performance of the new learning algorithm on several MPEG4 standard video sequences instead of synthetic data which many multi-class learning algorithms are tested on. The rest of the paper is organized as follows. Section II gives an introduction of multi-class Ψ-learning. Then a multiple VO extraction method using this new learning methodology is explained in Section III. Section IV provides the experimental results which is followed by conclusions in Section

(a) FGE = 1-sign(u) (b) FSVM function (c) Ψb function

April Issue Page 100-118 ISSN 2229 5280
(a) and 1(b) there is significant difference between this on sign, and for a sample
rewritten as \[26\]
above1, the cost function of the well
learning algorithm. For example, in the coding system described
A. Mult
direct consideration of GE. Defined as the probability of
As mentioned before, the most prominent feature of \(\Psi\)
are defined as follows
to simplify the notations an (\[\Psi\]) where
multiple comparisons between classes need to be performed. In order
class \(\Psi\) where the class label
is coded as \(y \in \{1,2,\ldots,M\}\), and for a sample \(x \in \mathbb{R}^d\) the decision
rule is
\[
y = \arg \max_{i=1,\ldots,M} f_i(x),
\]
where \(M\) is the number of classes and \(f_i\) is the decision function of
class \(i\) for \(i = 1,\ldots,M\). For the linear classifier
\(f_i(x) = w_i^T x + b_i\) with
\(w_i \in \mathbb{R}^d\) and \(b \in \mathbb{R}\). As a characteristic of multi-class problems,
multiple comparisons between classes need to be performed. In order
to simplify the notations an \((M,1)\) dimensional function vector
\(g(x; y)\) and a multivariate sign function sign(\(u\)) where \(u = (a_1,\ldots,a_{M-1})\)
are defined as follows
\[
g(x; y) = \left(f_y - f_1, f_y - f_2, \ldots, f_y - f_{y-1}, f_y - f_{y+1}, \ldots, f_y - f_M\right),
\]
\[
sign(u) = \begin{cases} 1, & \text{if } u_{\min} = \min(u_1, u_2, \ldots, u_{M-1}) > 0; \\
0, & \text{if } u_{\min} < 0. 
\end{cases}
\]

As mentioned before, the most prominent feature of \(\Psi\)-learning is the
direct consideration of GE. Defined as the probability of
misclassification, GE yielded by an M-class classifier is
A. Multi-Category \(\Psi\)-Learning
Seeking a function \(f\) to minimize GE is the ultimate goal for any
learning algorithm. For example, in the coding system described
above1, the cost function of the well-known linear SVM can be
rewritten as \[26\]
\[
GE = E[Y \neq \arg \max_{i=1,\ldots,M} f_i(X)].
\]
\[
GE = \frac{1}{2} E[1 - \text{sign}(g(X, Y))].
\]

where \(N\) is the number of training samples and the sum-to-zero
constraint is
invoked to eliminate the redundancy in \((f_1, f_2)\). Here the
so-called hinge loss \(FSVM(u) = 0\) if \(u > 1\), and \(2(1 - u)\) if \(u < 1\) is a
\[
\begin{align*}
\text{minimize:} & \quad \frac{1}{2} \sum_{i=1}^{N} \|w_i\|^2 + C \sum_{i=1}^{N} \psi(f_i(x_i) - f_{y_i}(x_i)), \\
\text{subject to:} & \quad \sum_{i=1}^{N} f_i(x) = 0 \quad \forall x.
\end{align*}
\]

As shown in Fig. 1(a) and 1(b) there is significant difference between this
convex envelope and \((1; \text{sign}(u))\) itself especially when \(u < 0\),
which corresponds to the inevitable misclassifications in non
separable 1 Conventionally, the formulation of SVM is expressed
in the coding system where the class label \(y \in \{-1,1\}\).

cases. Motivated by this consideration, Shen et al.
proposes to replace FSVM with a non-convex \(\Psi\) function \[27\] \[26\] as Here \(\Psi\) can be any function satisfying \(R > \Psi(u), 0\) if \(u \in [0, \frac{1}{2}]\)
and \(\Psi(u) = 1 - \text{sign}(u)\) otherwise, where \(\Psi(u)\) is non-increasing
in \(u\) and \(\Psi \in [0, 1]\). An example of such a function is shown in Fig.
1(c). Evidently because of the constant penalty for
misclassification, \(\Psi\) is much closer to \((1; \text{sign}(u))\) than FSVM,
which explains why \(\Psi\)-learning is expected to deliver higher
accuracy performance for the nonseparable case. A graphical
illumination is given in Fig. 2. In analogy to Eq. (4) which is for
binary classification, the multi-category \(\Psi\)-learning is formulated as
\[
\begin{align*}
\text{minimize:} & \quad \frac{1}{2} \sum_{i=1}^{N} \|w_i\|^2 + C \sum_{i=1}^{N} \psi(f_i(x_i) - f_{y_i}(x_i)), \\
\text{subject to:} & \quad \sum_{i=1}^{N} f_i(x) = 0 \quad \forall x.
\end{align*}
\]

where \(0 < T_1,\ldots, T_{M-1} < 1\) and \(\Psi(u)\) is non-increasing in each \(u\).
The multi-category \(\Psi\)-learning preserves the desired properties
of its binary counterpart. More specifically speaking, for any \(x\)
satisfying \(\text{sign}(g(x; y)) = -1, \Psi\) assigns a constant penalty which is
in the same spirit as GE. As a result, it is less sensitive to outliers
and offers better learning ability. The cost, however, is the
computational advantage since \(\Psi\) is not a convex function any
more. Fortunately the selection of the \(\Psi\) function is relatively
flexible. To utilize the difference convex (d.c.) decomposition
which is a global optimization strategy, a specific \(\Psi\) function
\[
\begin{align*}
\text{minimize:} & \quad \frac{1}{2} \sum_{i=1}^{N} \|w_i\|^2 + C \sum_{i=1}^{N} \psi(f_i(x_i) - f_{y_i}(x_i)), \\
\text{subject to:} & \quad \sum_{i=1}^{N} f_i(x) - \sum_{i=1}^{N} (w_i^T x_i + b_i) = 0.
\end{align*}
\]
B. Theoretical Advantage of Multi-Category $\Psi$-Learning

The lack of statistical learning theories for multi-category classification manifests the immaturity of this area. Only recently theoretical analysis of margin-based classification has been investigated, most of which is focused on the asymptotical scenario. Therefore practical performances of multi-category approaches in general remain empirical and theoretically unclear. Fortunately a statistically learning theory has been developed for multi-category $\Psi$-learning which provides insight into $\Psi$-learning's performance with respect to the choice of tuning parameter $C$, the training size $N$ as well as the number of classes $M$ [26], and we summarize it as follows:

1) $\Psi$-learning estimates the Bayes classifier $\sim f$ as opposed to the conditional probability. However, the optimal classification performance of $\sim f$ is realized via the $\Psi(u)$ function which differs from $1\text{-sign}(u)$.

2) The convergence rate of learning decreases as the number of classes $M$ increases although the order remains the same for finite $M$.

3) Unlike the binary case, the optimal performance of linear learning may not be achieved at large $C$ for multi-category problems.

For details of the theory, the readers are referred to [26].

III. MULTI-OBJECT EXTRACTION USING MULTI-CATEGORY $\Psi$-LEARNING

A. Related Work

Filtering and association and representation and localization are two major techniques for object tracking [29]. Rooted in the control theory, the former technique deals with the dynamics of the objects while the latter heavily relies on image processing technologies. The way these two techniques are combined and weighted is application dependent. For example, the filtering and association method prevails in the application of aerial video surveillance processes of object modeling, extracting, and searching are circumvented.

B. The Approach

As mentioned before, the choice of the learning algorithm is key to the success of the current approach because the performance of the algorithm is directly related to the classification accuracy. Considering the single VO extraction as an example, the background and the object are often not separable. As pointed out in Section II $\Psi$-learning aims at the minimization of GE and therefore has the advantages in non-separable cases. For this reason, a method for single VO extraction that employs binary $\Psi$-learning as the classifier is proposed in [12]. To tackle the challenging task of multi-object extraction, multi-category $\Psi$-learning has to be employed. As shown in Fig. 3, the approach consists of two phases: the training phase and the tracking phase. For classification at the pixel level, individual pixels are represented by pixel-wise color or intensity information, which however would result in misclassifications due to the negligence of the spatial relationship among pixels. Another concern is the size of the training set. If every pixel is such as RBG [11], the histogram of colors [13], or the coefficients of the DCT transform of the block centering at the pixel [12]. Then in the second step a classifier is trained and the classification function is obtained to discriminate the pixels that belong to the object from that belong to the background. Different classifiers have been attempted such as neural networks [11], $\Psi$-learning [12] or even an ensemble of linear classifiers [13]. The third step is to evaluate the classification function at every pixel in the subsequent frames. The final step is to generate the tracked object based on the classification results for which the way of implementation varies. For example in [13] a so-called confidence map is first produced according to the classification results, and tracking is then realized by locating the object where the peak of the confidence map occurs. The output of the tracker, however, is a rectangle that tightly encloses the object of interest. For the task of VO extraction the fourth step can even be skipped [11] since after the classification step we already know for every pixel if it belongs to the object. However, for efficiency purpose it is not necessary to do the classification pixel by pixel. By exploiting the spatial redundancy, we introduce the block-level classification instead and design a pyramid refining scheme to refine the boundary in an efficient and scalable manner [12]. Accuracy and complexity are two critical issues for VO extraction which have to be traded off in practice, and the major advantage of the classification-based methods is the potential to achieve both. The methods are accurate because powerful classifiers are designed for the purpose of object/background separation. Low complexity, on the other hand, is achieved through evaluating the classification function at each pixel which involves only simple calculations, e.g., $wT \times b$ for linear SVM while the time-consuming processes of object modeling, extracting, and searching are circumvented.

![Fig. 3. An overview of the proposed approach for multiple VO extraction](image-url)
included, it would contain too many training samples to yield a quick training especially when the frame size is large. The same efficiency issue exists if we do the pixel-by-pixel classification in the tracking phase. Fortunately, in most video sequences there is abundant spatial correlation that we can take advantage of to make the approach more efficient. Let $p$ denote a pixel and $N(p; d)$ the set of pixels within a small distance $d$ from $p$. Due to the spatial correlation of images, the class labels as well as the feature vectors of $p$ and $N(p; d)$ tend to be similar to each other. Based on this observation, we introduce the concepts of object blocks and background blocks, and suggest the representation and classification both be done at the block level as follows. Suppose we have $M$ VOIs of interest. The training phase begins with dividing the first frame, chosen as the training frame, into $(M+1)$ types of blocks (the number of different VOIs plus background) depending on which object or background the pixel at the center of the block belongs to. The block size is empirically chosen as $9 \times 9$, and evidently the number of blocks determines the size of the training set. We use the same method as in [12] to represent each block as well as the centering pixels. Namely, Discrete Cosine Transform (DCT) is first applied to each block and then based on the DCT

$$f_{local} = \left( f_0, f_1, f_2, f_3 \right)^T = \left( \frac{c(0,0)}{\sqrt{\sum_{i=1}^{8} c(0,i)^2}} \right).$$

(8)

Here $f_0$ is the average intensity, and $f_1$ and $f_2$ represent the horizontal and vertical edges, respectively. All the other high frequency information is contained in the last component $f_3$.

<table>
<thead>
<tr>
<th>$B_1$</th>
<th>$B_2$</th>
<th>$B_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B_4$</td>
<td>$B_0$</td>
<td>$B_4$</td>
</tr>
<tr>
<td>$B_7$</td>
<td>$B_6$</td>
<td>$B_5$</td>
</tr>
</tbody>
</table>

Fig. 4. Eight-connected neighboring blocks of block $B_0$.

$$f_{neighbor} = \left( \frac{avg(B_1 + B_2 + B_3)}{avg(B_1 + B_2 + B_3)} \right).$$

(9)

The neighboring features $f_{neighbor}$ are extracted from neighbors which are eight $9 \times 9$ blocks that are adjacent to the block under study in the vertical, horizontal and diagonal directions as shown in Fig. 4. With $avg(Bi)$ denoted as the average intensity of block $Bi$ we compute the neighboring features as:

The calculations given above only consider the grayscale information. When the video sequence is chromatic, we compute Eq. (8) and Eq. (9) for each color component and then concatenate the vectors respectively to form the chromatically local and neighboring features. Now with the training data in place, the next step is to train the machine by solving the optimization problem Eq. (5), which yields $(M+1)$ decision functions that separate the $M$ objects as well as the background. In the tracking phase each subsequent frame is also divided into blocks of $9 \times 9$, and for each block the $(M+1)$ decision functions are evaluated to decide what object the centering pixel belongs to, which consequently determines the class label of the block. Then the tracking mask of every object is formed by the blocks that have been classified in the corresponding class. At this point the resolution of object’s boundary is as large as the size of the block, but by applying a pyramid boundary refining algorithm [12] the object boundary can be refined and the pixel-wise accuracy can be achieved. We refer the details of the latter algorithm to [12].

IV. EXPERIMENTAL RESULTS

In this section we apply the proposed multiple VO extraction method to three standard MPEG-4 test video sequences, which exhibit varieties of temporal and spatial characteristics. These sequences are students, Trevor, and Sun Flower Garden, respectively. Also the performance comparisons are made between multi-category Ψ-learning and three popular multi-class algorithms, namely one-vs-all, one-vs-one and directed acyclic graph (DAG) [22]. All experiments are carried out on a Pentium IV 2.5-GHz PC.

A. Subjective Evaluations

The first one to test is Students. As the major content of this sequence, the two students are chosen as two objects of interest, and along with the background this is a three-class classification problem. As one can see from the original frames shown in columns (a) and (d) of Fig. 5, Students is a typical sequence of slow but heterogeneous motion. For example the male student turns the head and moves his hands while his body stays still most of the time. The extracted objects are shown in columns (b), (c), (e), and (f), respectively. One can see that the proposed method works well, which discriminates the body parts of the students as well as their faces. The latter is not an easy task since the skin color is very similar between the two students. Another sequence containing three people is also tested, and the three people are considered as three objects which makes it a four-class classification problem. The original frames and the extracted objects are shown in Fig. 6. Unlike the Students sequence, the objects in this sequence change the appearance a great deal. Taking the lady who sits at the farthest right as an example, her face changes from frontal to left-side view. Besides, the man in the middle is originally seated but finally standing. As seen in Fig. 6, the main body of the objects are successfully extracted although the boundaries of the objects...
are not perfectly separated due to classification errors. Among the sequences tested in the experiments, Sun Flower Garden is the most challenging one. Different from the previous video-conference kind of sequences, it displays a natural scene that is rich of colors and textures with a non-stationary camera. There are two objects of interest: the house and the tree. For the first few frames, the house is occluded by the tree. Two of such frames are shown in column (a) of Fig. 7, and the two extracted objects (house and tree) by using Ψ-learning are shown in columns (b) and (c), respectively. With the camera moving,

Fig. 5. The extraction performance of Animals. Columns (a) and (e) display the original frames. Columns (b) and (c) display the extracted VO #1. Columns (c) and (f) display the extracted VO #2. tree shifts toward the left hand side of the frame and finally disappears as in column (a). From that point on, only the house can be extracted by the proposed method as shown in the last column of Fig. 7.

B. Performance Comparison

For their simplicity and effectiveness, one-vs-all, one-vs-one and DAG are three widely used multi-category algorithms. Suppose we have M classes. One-vs-all constructs M binary classifiers f_j^\text{OVA}(x) with the jth one separating class i from all the remaining classes. One-vs-one and DAG, on the other hand, construct M(M-1)/2 decision functions f_{ij}(x), each of which is responsible for the binary classification task between class i and j. At the classification step, one-vs-one classifies a sample x to the class for which f_j^\text{OVA}(x) produces the highest value while one-vs-one follows a voting strategy. As for DAG, it builds a directed acyclic graph using the M(M-1)/2 binary classifiers as the internal nodes. The classification is achieved by going through a path from the root of the graph to a leaf node which indicates the predicted class [22]. To see how multi-category Ψ-learning performs against these three popular methods, the classification errors yielded by all the four methods are displayed every 5 frames in Fig. 8 where SVM is the underlying binary classifiers. For the training of each SVM, the classification accuracy is estimated by testing different values of CC/2, 2, 2, 2, 2, and the best one

Fig. 6. The extraction performance of Trevor. Columns (a) and (e) are the original frames. Columns (b) and (f) are the extracted VO #1. Columns (c) and (g) are the extracted VO #2. Columns (d) and (h) are the extracted VO #

The computational complexity of the new approach deserves a discussion. Assume there are M classes and each pixel is represented by a d-dimensional feature vector x. In the tracking phase we need to evaluate M functions f_i = wT_i x + bi each of which performs d multiplications to determine the class label of a given pixel. As a result, the computational complexity is O(Md), which is a linear function of the number of objects M and gives the approach low complexity and good scalability.

V. CONCLUSIONS

VO extraction is of great importance for content-based video analysis, and a great deal of research has been performed for single object extraction. Unfortunately, multi-object scenario which is more realistic imposes a much greater challenge. Following the idea that handles VO extraction as a
classification problem, this paper aims to tackle multiple object extraction by solving a multi-class classification problem and using multi-category $\Psi$-learning which is a newly

REFERENCES


AUTHORS PROFILE

D.V. Chandra Shekar, received Master of Engineering with Computer Science & Engineering from ANNA University, Chennai, Tamilnadu, India, . He is currently working as Associate Professor, in the Department of Computer Science, T.J.P.S COLLEGE (P.G COURSES),Guntur, which is affiliated to Acharya Nagarjuna University. He has 12 years teaching experience and 1 years of Industry experience. He has published 30 papers in National & International Journals.

Y. Suresh Babu , working as Associate Professor, in the Department of Computer Science, J.K.C COLLEGE ,Guntur, which is affiliated to Acharya Nagarjuna University. He has 18 years teaching experience. He has published 8 papers in National & International Journals.

Dr. G. Satyanarayana Prasad received Ph.D. degree in Computer Science in the Faculty of Engineering in 2006 from Andhra University, Andhra Pradesh. He completed MS in Computer Science & Engineering from A & M University,USA,1984. He is currently working as Professor in the Department of Computer Science and Engineering, R.V.R & JC Engineering College, Guntur. His current research is focused on Image Processing. He has published several papers in IEEE journals and various National and International Journals.