Visual Surveillance Metadata Management

Petr Chmelar, Jaroslav Zendulka
DIFS FIT, Brno University of Technology
{chmelarp, zendulka}@fit.vutbr.cz

Abstract

The paper deals with a solution for visual surveillance metadata management. Data coming from many cameras is annotated using computer vision units to produce metadata representing moving objects in their states. It is assumed that the data is often uncertain, noisy and some states are missing.

The solution consists of the following three layers: (a) data cleaning layer – improves quality of the data by smoothing it and by filling in missing states in short sequences referred to as tracks that represent a composite state of a moving object in a spatiotemporal subspace followed by one camera. (b) Data integration layer – assigns a global identity to tracks that represent the same object. (c) Persistence layer – manages the metadata in a database so that it can be used for online identification and offline querying, analyzing and mining. A Kalman filter technique is used to solve (a) and a classification based on the moving object's state and its visual properties is used in (b). An object model for layer (c) is presented too.

1. Introduction

Nowadays there is a lot of data produced by wide area surveillance networks. This data is a potential source of useful information both for online monitoring and offline querying, analyzing and mining. Machine vision techniques have dramatically increased in quantity and quality over the past decade. However, they still do not provide the satisfactory knowledge, except some simple problems such as people counting or left luggage detection.

We have found out that security and rescue professionals prefer keeping human in the loop. But they need support for the burden of continuous concentration on monitoring and subsequent analysis of large amount of data where an "interesting" event is expected. We have found out the management and analysis of high volumes of low quality spatiotemporal annotation data as an interesting problem. Thus, we propose solution for robust multiple camera fusion and moving objects data management. Our approach enables a behavior analysis in a wider context than it can be accomplished using single camera data processing and online event analysis.

Imagine a pickpocket has stolen a handbag of a lady. About two hours later not far from there, a flat caught fire and an almost suffocated guy has been taken to hospital. His drug test was positive later. Imagine also a system that can identify the guy at the main station he was before. It may find a location, where he met the lady. In addition to that, the system can show a person, who has sold him drugs.

For that scenario, it is expected a network of many surveillance cameras. Their fields of view might be overlapping, but more usually they are far from each other. The raw video data are processed using machine vision units (low-level techniques, motion estimation and object classification) to provide annotation of the scene in the meaning, not necessarily form, of the MPEG-7 descriptors [6] including color, shape, texture and trajectory of moving objects in image coordinates or real objects in case the camera system is calibrated. Although we can't presume the data is of high quality, to know the geometric and classification correspondence is always advantageous, because the more prior information the more accurate and relevant results. This annotation data will be also referred to as metadata in the paper.

The paper is structured as follows: The next chapter introduces basic concepts, Chapter 3 is an overview of related works. Chapters four to five present the three layers of the proposed solution.

2. Problem Formulation

Suppose a surveillance system with many vision *sensors c*. We understand the sensor as a camera equipped with a machine vision unit, that extracts the necessary metadata and compress the video data so it can be efficiently stored in the video repository – the sensor has a network communication interface.

The metadata received from a sensor represents information about an *object o* – its *state* x_t, the state is of two types of features – visual properties (appearance) and space-time features. The latter are represented by *location* $[x, y, t]$ and *velocity* $[d_x, d_y]$. $[x, y]$ will be referred to as (2D) *position* $[x, y]$ of the

object at the *time t*. A *track* segment *s* is a sequence of such states in a spatiotemporal subspace followed by one camera.

There are many cameras in the system, thus the object o_k is supposed to have a globally unique *identifier* (key) *k*, which can be (temporarily) unknown. A global sequence of states (tracks) that correspond to one object is called object *trajectory*. The central here problem is how to find correspondence between tracks and global identifiers.

Data is supposed to be uncertain – noisy, some states might be missing and even visual properties of objects are biased within the network. The uncertainty has several reasons – the first is the appearance and existence uncertainty in machine vision. It comes from the curse of dimensionality – central projection, segmentation errors and appearance manifolds. It is even hard to determine, based on visual perception, if it is an object. Although, we don't propose solution, we may contribute it by the probabilistic modeling and by solving the following problems.

The uncertainty of state x_t can be understood as a measurement noise. A moving object observed by cameras is supposed to be discrete time linear (first-order) dynamic system with Gaussian noise [9], its filtering results in observation y_t.

The state incompleteness may be predicted by the filtering for a few seconds [2]. However, wide-area camera networks need a robust solution for handover – passing tracked objects between sparse cameras, where the objects temporarily appear.

The last but not least problem is an efficient storing and retrieval of constantly changing uncertain spatial data that might be highly structured and unidentified.

3. Related work

After researchers dealt with tracking multiple objects using single cameras [4], there has been recognized the problem of object correspondence – identification [10] within multiple cameras with overlapping field of view first. Later, the problem was stated also for the non-overlapping view-fields [12], which have been constantly improved [5] and many researchers have been filling gaps within cameras and bridging blind regions in theory and practice [2].

However these frameworks still need a great deal of work, because the more information they neglect the worse is the overall precision even in small environments with the only blind region [2]. This is solved by making simplifying presumptions such as non-overlapping cameras only [5], or by using redundant cameras [14] set somehow to make it work, which is doesn't correspond to real surveillance networks.

Although machine vision experts have started the research, they are interested in modeling of objects' visual properties rather than modeling uncertain spatiotemporal data and distributed database architectures. The results from communities including both vision and database experts are coming gradually. For instance, DIVA [11] provides the simple identification of cars using two cameras based on color and time only.

In contrast to the standardized (object-)relational, spatial and newly temporal (history) data, their combination is not well posed yet [13]. The management of trajectory data in relatively constant environment is quite simple [3], [6]. The worse case is data (existentially) uncertain, encumbered by noise and information loss [7].

4. Data Cleaning

Moving objects are modeled using a discrete time dynamic system. At the moment, the object state x_t is characterized by state vector *x*, containing position and velocity vector [*x*, *y*, d_x, d_y]. In that way, the track is constructed (in both image and 2D real world coordinates). However it cannot be observed directly, because it is encumbered by hidden (Gaussian) noise *w*. The system produces visible output vector *y* that is a simple linear observation (*x* evolves first-order Markov process), encumbered by noise *v*. At time *t*, the model can be written as in the Figure 1.

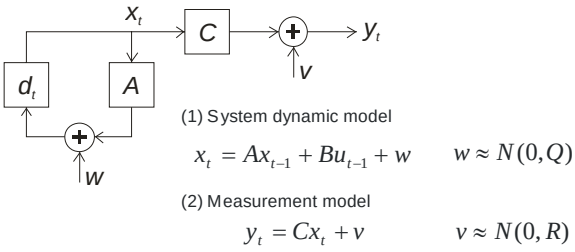

Figure 1. The dynamic process illustration.

In the Figure 1, d_t is a delay, *A* is a state transition matrix; *B* the optional control matrix with input *u*, *C* is the observation or measurement matrix. *N* is a normal distribution of mean 0, the (covariance) matrices of *w* and *v* are *Q* and *R*, respectively. Based on Bayesian probability $P(x_t \mid y_t) \approx N(\hat{x}_t, P_t)$, as described below.

An efficient computational (recursive) tool to estimate the state of a process, in a way that minimizes the mean of error, is the Kalman filter.

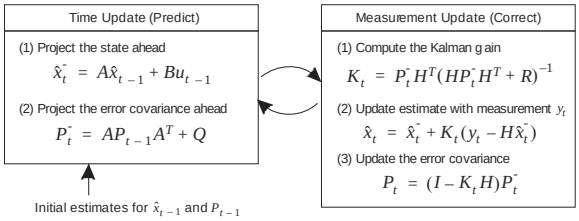

Figure 2. Operation of Kalman filter, combining the high-level diagram with the equations [15].

The Kalman filter iteratively applies two stages of computations using feedback control (see Figure 2):

1. Time update computations (prediction). Here $\tilde{x}_t^-$ is the prior state estimate (predicted) at time t, $\hat{x}_t$ is the posterior (corrected) state estimation at the time ($\hat{x}_t = E[x_k]$), P_{t-1}^- is the prior estimate error covariance matrix and P_{t-1} is the posterior error covariance matrix ($P_t = E[(x_t - \hat{x}_t)(x_t - \hat{x}_t)^T]$).

2. Measurement update computations (correction). There H is a measurement matrix and K_t is a Kalman gain matrix. The weighting by K minimizes the posterior error covariance, while R approaches zero, the actual measurement y_t is trusted more to the detriment of measurement prediction $H\hat{x}_t^-$. The detailed explanation is provided in [15], [9].

We apply the Kalman filter in data cleaning layer to smooth and predict missing states. We have implemented it in Java. It is freely available under the GPL license (at *www.fit.vutbr.cz/~chmelarp/public/*) together with the sample code for 2D object tracking.

5. Data Integration

As it was stated in the problem formulation, the vision sensor fusion is formalized as the identification problem. The goal is to compute the optimal global identifier k for each unresolved object o and the corresponding track segment s as the following posterior probability maximization:

$$\kappa^*(o) = \arg\max_k P(k \mid o, s) \qquad (1)$$

Bayes theorem is used to solve (1). We use the naïve (but politically correct) assumption that the object appearance and trajectory are independent. Presuming the uniform distribution on k:

$$P(k \mid o, s) \approx \begin{cases} P(o \mid k)P(s \mid k) & if > P_0 \\ P_0(o) & \leq P_0 \end{cases} \qquad (2)$$

In (2), the likelihood $P(o \mid k)$ is the normalized similarity of an observed object o and the object corresponding to the key k in the database. $P(s \mid k)$ is a likelihood of a track s belonging to o matching the previous (last) trajectory of the o_k. P_0 is the likelihood of being a new object and to assign new identifier to o.

We have proposed identification as a Bayesian combination of learned (summarized) object appearance model and (SVM) classification using an inverted Kalman filter (state $\neg\hat{x}_{t1}$), applied to the current track segment s that belongs to the subspace of the actual camera c_{act}. Especially, we want to find out the previous camera c_{prev} of the wide area surveillance network in which the object o has been seen last time (even the same). In fact, we get the probability estimates of each camera label using a library for support vector machines [8].

The inverse Kalman state $\neg\hat{x}_{t1}$ means, while tracking an (unidentified) object in a field of view for sufficient amount of time necessary to determine its visual and motion parameters, the Kalman filter velocity vector is inverted and the object virtually returns to the location it was firstly detected by the sensor. The inversion is necessary because the Kalman filter can't determine the velocity within the first measurement, and it is encumbered by the non-optimal (high) noise.

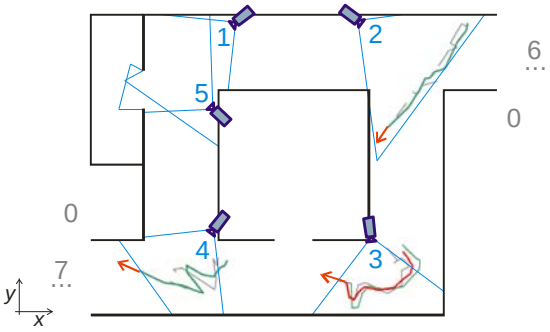

Figure 3. The illustration of large area surveillance.

In the area (see Figure. 3), a moving object (thin gray lines) is captured using cameras 4, 3, 2, and annotated by a machine vision unit in real coordinates. The predicted continuous tracks fill in the missing measurement (thick green, sensors 4, 2) by continuous applying the prediction step of the Kalman filter. The corrected values (illustrated at the sensor 3, thick red) are stored in the database (within several seconds). After some time the velocity vector is inverted; and the inverse Kalman state of the first position is determined (red arrows represent the state $\neg\hat{x}_{t1}$).

Example 1. The classification of the inverted Kalman state at the current location (subspace observed by a camera) determines the probability of being either a previously identified object, potentially passing from other area (cells), or a new object. There are 5 cameras, 2 neighboring cells – 6, 7, and 1 class 0 acting as a new object; it means 8 classes in the

Figure 3. Suppose, there is an object captured by camera 4. The classification probability $P(s \mid k)$ of its $\neg\hat{x}_{t1}$ has maximum e.g. label 7 – 60% coming from neighboring area and label 0 – 30% being a new object (P_0). The identification layer then selects unidentified objects from the camera (neighboring camera of the cell 7) and compares its visual appearance. The maximum selected similarity $P(o \mid k)$ to the unknown object is e.g. 40%. According to (1) 40%*60% = 24% is smaller then the P_0, thus the maximum probability of *o* is to be a new object and a new identifier is assigned to it.

In contrast to the example, our solution is open to more information. For instance, it is the last (truly) known location of an object, which is also predicted for some time in case other object occludes. Before the previous location (camera) classification is applied, the database is queried to select an object which is supposed to be in the same spatiotemporal location to correct the visual segmentation, merging, splitting and other computer vision based misinformation.

The training is performed according these weaknesses. The location and the similarity-based queries are trained concurrently. Although some researchers think, that the identity correspondence can be learned automatically [5], [2], we don't share this opinion. To achieve a higher identification precision and to avoid the object appearance bias within multiple cameras, the iteratively trained model is used. First, only the appearance is offered to the annotator, who verifies the automatic identification presumptions.

6. The Persistence Layer

The persistence layer is responsible for storing and querying metadata. The visual surveillance system stores not only metadata representing moving objects but also training data and environment, in which they move, sensors etc. Here we will only focus on metadata directly related to moving objects.

In addition, we will apply object-oriented view on this metadata and we will use UML [1]. There are three main classes that correspond to the fundamental concepts presented in Chapter 2: *Object*, *Track* and *State*.

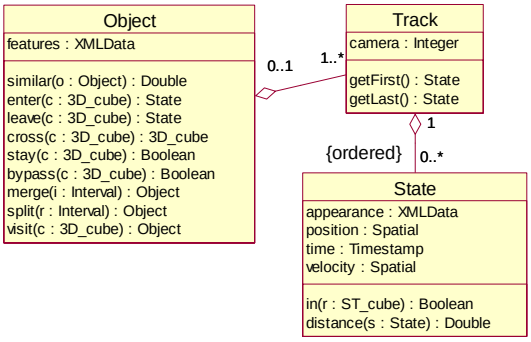

Figure 4. Conceptual class diagram.

We do not model trajectories as instances of a corresponding class because they do not have any other attribute but a set of tracks that belong to the same object ordered by time. This information can be easily derived (see Figure 4).

The layer must provide support for querying and more sophisticated analysis. We can distinguish two types of operations concerning the movement of objects that must be supported: (a) *environmental operations* - look for relationships of an object's trajectory and a specified spatial or spatiotemporal environment (referred to as *environmental relationships*). (b) *trajectory operations* that look for relationships of two or more trajectories restricted by time, spatial or spatiotemporal constraints (referred to as *trajectory relationships*).

We consider 2D spatial space therefore a spatial environment or constraint is defined by a 2D rectangle. Spatiotemporal environment or constraint is represented by a 3D cube where time is the third dimension. A time constraint is specified as time interval.

We have identified the same five basic environmental relations like in [3]: *enter*, *leave*, *cross*, *stay* and *bypass*.

We have also identified the following trajectory relationships:

together – two or more objects (represented by their trajectories) go together in some time interval, some rectangle or spatiotemporal cube,

merge – two or more objects moving separately merge each other, the opposite operation is *split*,

visit – two or more objects visit the same rectangle.

Because trajectories are not modeled explicitly and related information is available from class Object, all these operations are operations of that class.

Most operations exist with several different signatures. Only some of them are presented. In addition, some data types, as Spatial, spatiotemporal

interval 3D_cube etc., are abstract and only indicate the dimensionality or are related to concepts.

7. Conclusions

The paper presents solution of metadata management for a surveillance system with many cameras equipped with computer vision units that produce data with high level of uncertainty. The main contribution of our current solution is improved quality of metadata stored in the database together with the identification method. The improved quality is achieved using Kalman filter. The identification method is based on combination of visual features and the inverted Kalman state classification.

The Java implementation using (not limited to) the Oracle database is, internally called NCST (Non-Certain Spatiotemporal), acts as a (real-time) database front-end. The solution is generally open to accept many kinds of data and network connections from (not only) vision modules and provide the knowledge to the monitoring and data mining applications by data querying and simple analysis. However, currently it is limited by data processed and coming from the vision sensors, we are at the integration stage now.

The future research and development will focus on (real-time) processing of continuous queries, tuning database performance and integrating additional functions (similar to HMM) to recognize other behavior aspects of the object (human) to discover its internal state (waving, sick) in addition to the proposed.

8. References

[1] J. Arlow, I. Neustadt, "UML 2 and the Unified Process". Second Edition, *Practical Object-Oriented Analysis and Design*. Addison/Wesley, 592 p., 2005.

[2] J. Black, D. Makris, "Validation of Blind Region Learning and Tracking", *PETS*, 2005.

[3] S. Brakatsoulas, D. Pfoser, N. Tryfona, "Modeling, Storing and Mining Moving Object Databases". *IDEAS*. 2004.

[4] C. Cedras, M. Shah, "Motion-Based Recognition: A Survey", *IVC*. 1995.

[5] O. Javed et al., "Tracking Across Multiple Cameras With Disjoint Views", *Proc. of the Ninth IEEE International Conference on Computer Vision*. 2003.

[6] J. M. Martínez, *MPEG-7 Overview*, 2004 [cit. 2005-10-22], http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

[7] D. Pfoser, N. Tryfona, "Capturing Fuzziness and Uncertainty of Spatiotemporal Objects", *Advances in Databases and Information Systems*: Proc. 5th East European Conference. 2001.

[8] C. Chang and C. Lin, *LIBSVM : a library for support vector machines*, 2001.

[9] S. Roweis, Z. Ghahramani, "An Unifying Review of Linear Gaussian Models", *Neural Computation*, 1999.

[10] T. Huang, S. Russell. "Object identification in a bayesian context", *Proceedings of IJCAI*, 1997.

[11] M. M. Trivedi, T. L. Gandhi, K. S. Huang, "Distributed Interactive Video Arrays for Event Capture and Enhanced Situational Awareness", *IEEE Intelligent Systems*, 2005.

[12] V. Kettnaker, R. Zabih, "Bayesian multi-camera surveillance". *Proceedings of CVPR*. 1999.

[13] Whitemarsh Information Systems Corporation, *SQL Standards*. [cit. 2007-01-10] http://www.wiscorp.com/SQLStandards.html.

[14] T. Zhao et al. "Real-time Wide Area Multi-Camera Stereo Tracking", *Proceedings of the CVPR*, 2005.

[15] G. Welch, G. Bishop, "An Introduction to the Kalman Filter", 2006 [cit. 2007-01-10], http://www.cs.unc.edu/~welch/kalman/.