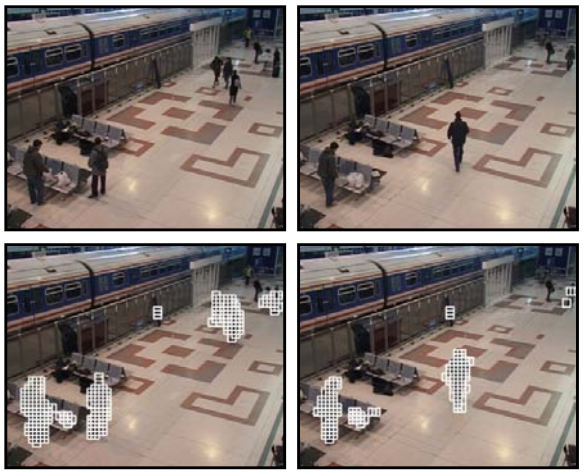
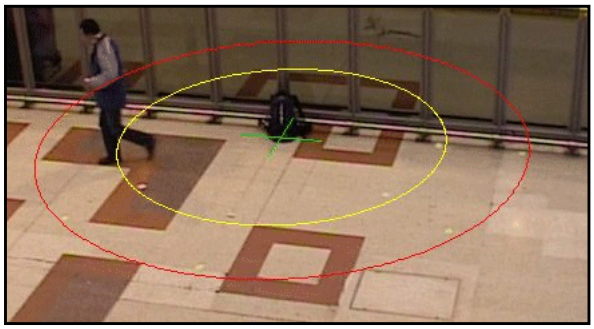
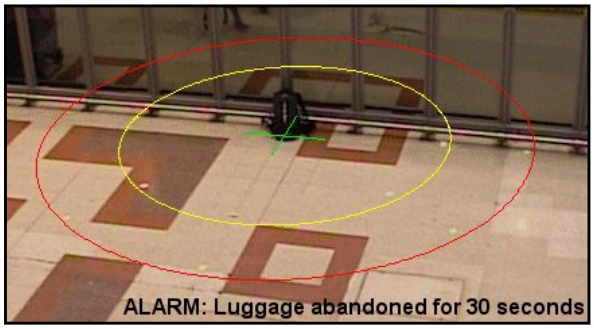
IEEE COMPUTER SOCIETY

Proceedings

Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006)







June 18, 2006
New York, USA

New York City CVPR 2006

In Conjunction with IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)

In Cooperation with IEEE Computer Society
Supported by EU PASR ISCAPS

IEEE Networking the World™

PETS 2006

Supported by

PETS 2006 is supported by

European Union under ISCAPS (Integrated Surveillance of Crowded Areas for Public Security) (SEC4-PR-013800)
IEEE Computer Society
The University of Reading, UK.

Proceedings

Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006)

June 18, 2006
New York, USA

In conjunction with
IEEE Computer Society Conference on
Computer Vision and Pattern Recognition(CVPR'06)

In cooperation with IEEE Computer Society

Edited by

James M. Ferryman
Computational Vision Group
Department of Computer Science
The University of Reading
Whiteknights
Reading RG6 6AY
UK

PETS 2006
Proceedings of the Ninth IEEE International Workshop on
Performance Evaluation of Tracking and Surveillance
June 18, 2006
New York, USA

First Edition

ISBN 0-7049-1422-0

Cover design depicts images taken from contributions within the proceedings.

Compiled in England at The University of Reading.
Printed and bound by The Print Room, Reading, UK.

Table of Contents

Ninth IEEE International Workshop on PETS 2006

Foreword

Welcome to the proceedings of the Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2006), held on June 18, 2006 in New York, USA. The one day workshop is being held in conjunction with the IEEE Computer Society Conference on Vision and Pattern Recognition (CVPR'06).

Visual surveillance is a major research area in computer vision. The recent rapid increase in the number of surveillance cameras has led to a strong demand for automatic methods of processing their outputs. The scientific challenge is to devise and implement automatic systems for obtaining detailed information about the activities and behaviours of people and vehicles observed by a single camera or by a network of cameras. In many applications, even low level information about the positions of people and vehicles is useful. However the full benefits of visual surveillance will only be realised with the development of wide area networks of cameras capable of building up high level descriptions of scene activity over time.

Visual Surveillance is a key technology in the following areas:
- the fight against terrorism and crime including threat assessment
- public safety, especially in transport networks, town centres and public facilities such as schools, hospitals and sports grounds
- the efficient management of transport networks and public facilities

The workshop continues the series of International Workshops on Performance Evaluation of Tracking and Surveillance which started in Grenoble in 2000 in conjunction with FG'2000. Since 2000, a total of nine workshops have been held to date, two in conjunction with the IEEE International Workshop on Visual Surveillance.

We would like to thank all of those who have contributed papers to the workshop. The final programme consists of 15 oral presentations, including two invited speakers. The topics include evaluation projects, frameworks and systems, evaluation of motion segmentation, and two dedicated sessions to the presentation and evaluation of results based on the PETS 2006 dataset.

The dataset for PETS 2006 was provided with the support of the British Transport Police and Network Rail, UK. PETS 2006 is supported by the EU PASR 2004 ISCAPS (Integrated Surveillance of Crowded Areas for Public Security) (SEC4-PR-013800), and the IEEE Computer Society. Finally, thanks to David Thirde (PETS'06 Workshop Organiser) and Chuck Stewart (CVPR'06 Workshops Chair), for their coordination and help with the organisation of the workshop.

We hope that you enjoy the proceedings and look forward to your active participation.

Workshop Co-Chairs
June 2006

Workshop Organisers

Co-Chairs

James L. Crowley, *I.N.P. Grenoble, France*
James Ferryman, *The University of Reading, UK*

Workshop Organiser

David Thirde, *The University of Reading, UK*

Programme Committee

Justus Becker	TNO, The Netherlands
Terrance Boult	Lehigh University, USA
Stefano Bovone	Elsag Spa., Italy
David Cher	SILOGIC, France
Andrew Cooke	BAE Systems, UK
Jean-Francois Daguzan	FRS, France
Xavier Desurmont	Multitel A.S.B.L., Belgium
Andrea Kropp	Datamat, Italy
Pierre-Alain Moellic	CEA, France
Alvaro Morais	GMV, Spain
James Orwell	Kingston University, UK
Arthur Pece	The University of Copenhagen, Denmark
Justus Piater	Université de Liège, Belgium
Fatih Porikli	MERL, USA
Carlo Regazzoni	University of Genoa, Italy
Stan Sclaroff	Boston University, USA
Andrew Senior	IBM T.J. Watson Research Center, USA
Tieniu Tan	NLPR, Institute of Automation, China
Monique Thonnat	INRIA Sophia-Antipolis, France
Christian Viet	SAGEM, France

PETS vs. VACE Evaluation Programs: A Comparative Study

Vasant Manohar, Matthew Boonstra, Valentina Korzhova,
Padmanabhan Soundararajan, Dmitry Goldgof and Rangachar Kasturi
Computer Science & Engineering, University of South Florida, Tampa, FL
{**vmanohar, boonstra, korzhova, psoundar, goldgof, r1k**}**@cse.usf.edu**

Shubha Prasad and Harish Raju
Video Mining Inc., State College, PA
{**sprasad, hraju**}**@videomining.com**

Rachel Bowers and John Garofolo
National Institute of Standards and Technology, Gaithersburg, MD
{**rachel.bowers, john.garofolo**}**@nist.gov**

Abstract

There are numerous distributed efforts on performance evaluation of computer vision algorithms. Information exchange between programs can expedite achievement of the common goals of such projects. As a step towards this, this paper presents a qualitative comparison of the VACE and the PETS programs, the synergy of which, we believe, will have a signi cant impact on the progress of research in the vision community. The comparison is based on the vital aspects of an evaluation project – the framework, the tasks, performance metrics and ground truth data.

1. Introduction

Performance evaluation of computer vision algorithms has received signi cant attention in the past few years with increasing demand and awareness to establish a global platform for benchmarking various algorithms against a common dataset, metrics, and evaluation methodology. This is substantiated by the need to identify speci c aspects of a task that has further scope for improvement, quantify research progress and most important of all acquire diverse data that re ect the inherent problems to be confronted if the system were to be deployed in a real world scenario.

Object detection and tracking is an important and challenging topic in computer vision which addresses the problem of spatially locating an object of interest in the video. This is an important step in object identi cation since it is necessary to localize an object before recognition can be accomplished. Similarly, for a system to detect the occurence of an event, it has to rst detect and further track the relevant objects involved in the event.

There are several current efforts towards the evaluation of object detection and tracking in video. The Video Analysis and Content Extraction (VACE) program, supported by Advanced Research and Development Activity (ARDA) is one such endeavor, established with the objective of developing novel algorithms and implementations for automatic video content extraction, multi-modal fusion and event understanding. The program has several phases and is currently nearing the end of Phase II. During VACE Phase I and Phase II, the program achieved signi cant progress in video content analysis; speci cally in techniques for the automated detection and tracking of scene objects such as faces, hands, and bodies of humans, vehicles, and text in four primary video domains: broadcast news, meetings, surveillance, and Unmanned Aerial Vehicle (UAV) motion imagery. Initial results have also been obtained on automatic analysis of human activities and understanding of video sequences. The performance evaluation initiative in VACE is carried out by the University of South Florida (USF) under the guidance of National Institute of Standards and Technology (NIST).

The Performance Evaluation of Tracking and Surveillance (PETS) program was launched with the goal of evaluating visual tracking and surveillance algorithms. The rst PETS workshop was held in March, 2000. Since then, there have been seven such workshops exploring a wide range of surveillance data. While the theme of the initial workshops was target detection and tracking, in the past few workshops the program has matured to address event level tasks. PETS Metrics[1] is a related, but newer initiative to provide an online service for automatically evaluating surveillance

[1] http://petsmetrics.net

results. Currently, the website supports the motion segmentation metrics but in due course is expected to extend to tracking and event detection.

Video Event Recognition Algorithm Assessment Evaluation (VERAAE), Computers in the Human Interaction Loop (CHIL)[2], Evaluation du Traitement et de l'Interprétation de Séquences Vidéo (ETISEO)[3], and Augmented Multiparty Interaction (AMI)[4] are a few programs that share the central idea of developing new algorithms for video understanding and evaluating them for tracking research progress.

Since the motivation for all the above programs is essentially the same, technology transfer among individual programs will result in faster research growth. The CLassification of Events, Activities and Relationships (CLEAR)[5] Evaluation Workshop was the first international evaluation effort that brought together two programs – VACE and CHIL. The benefits of collaboration are obvious – availability of more data for the research community for algorithm development and evolution of widely accepted performance metrics that provide an effective and informative assessment of system performance, are some of the more important ones.

The inspiration for this paper is akin to the CLEAR evaluation attempt. To that extent, we present a qualitative comparison between the PETS and VACE programs in the following viewpoints – evaluation framework (Section 2), evaluation tasks (Section 3), ground truth data (Section 4) and performance metrics (Section 5). We conclude with a discussion on the factors that in uence the decision on collaboration between any two programs in Section 6.

2. Evaluation Framework

In any evaluation task it is important that each step is clearly defined and executed accordingly. Following this ideology, both the PETS and the VACE programs have a well defined evaluation framework that shares a common perspective.

In VACE the evaluation of a task begins with the task definition. For this to be finalized, the process follows a sequence of steps on a set schedule.

- Micro-corpus: Task definitions are formalized.
  - Two annotators annotate the same clip using a carefully defined set of attributes. (This helps fine tune the annotation guidelines)
  - These annotations are released to the research community.
  - Modifications are made according to the researchers and evaluators need. These include object definitions and attributes.
- Dry Run: Task definitions are frozen. Annotation of training data is initiated and first implementation of the defined metrics in the scoring tool is completed. During this phase, participants familiarize themselves with the output file formatting by using the scoring software.
  - A sub-sample (usually 5 annotated clips) of the training data is released to the participants along with the *index* file on which the participants are asked to submit their results on.
  - Evaluators score and send scores to the participants for verification.
  - Participants use the scoring tool to score their output with the annotation reference and finally verify.
  - Any issues are resolved iteratively with consultations between evaluators and participants.
- Formal Evaluation
  - Release ground truth for the training data (50 clips of 2.5 mins length approx.).
  - Release test data (*index*) file (another 50 clips of 2.5 mins length approx.).
  - Participants submit the results (similar to the Dry Run process).
  - Evaluators score and release scores along with annotations for verification.

PETS also follows a similar process except that the researchers (participants) can submit their results on the web and get their scores generated through PETS Metrics, the online evaluation service. To reach this level of maturity in VACE, the tasks and the framework needs to be frozen beforehand. The schedule so far prevented the evaluators in having a similar setup. Possible helpful scripts can be written so that the participants can submit the results and automated error messages including filenaming conventions, formatting, etc can be generated on the web. This can potentially alleviate generic problems that arise due to the participants not fully implementing the required submission formats.

3. Evaluation Tasks

Object detection and tracking is the primary task evaluated in VACE-II. Broadly speaking, the goal of the detection task is to identify the location of the object of interest in each

[2] http://chil.server.de
[3] http://www.silogic.fr/etiseo
[4] http://www.amiproject.org
[5] http://www.clear-evaluation.org

TASK \ DOMAIN	MEETINGS	BROADCAST NEWS	SURVEILLANCE	UAV
Text detection & tracking	–	VACE	–	–
Face detection & tracking	VACE	VACE	–	–
Hand detection & tracking	VACE	VACE	–	–
Person detection & tracking	VACE	–	PETS & VACE	VACE
Vehicle detection & tracking	–	–	PETS & VACE	VACE

Table 1: Evaluation Tasks Supported in VACE-II (– indicates tasks not supported in either PETS or VACE).

frame. On the other hand, the goal of the tracking task is to locate and track the object by a unique identi er in the video sequence.

Evaluation support for a particular task-domain pair depends on the richness of the target objects being evaluated in that domain. After careful deliberation, 10 task-domain pairs were identi ed for evaluations (Table 1).

The primary focus of PETS is on surveillance technologies. Target detection & tracking and event recognition are the predominant problems addressed in the workshops. A major difference between the de nitions of the detection task between the two programs is the fact that PETS de nition additionally includes object segmentation which requires systems to output a binary bitmap providing the shape of the object. The tracking task is analogous to the VACE de nition with the only difference that the rst instance of tracking failure terminates evaluation in PETS. In VACE, among multiple output tracks for a single ground truth object in the video, the single longest track in time is picked as the corresponding hypothesis, while in all other frames the ground truth is treated as a miss.

Since object detection and tracking is the main problem evaluated in VACE-II, the scope of this paper is restricted to the comparison of these tasks. Metrics, annotations and definitions for event level tasks are not discussed. Table 1 summarizes the evaluations supported in VACE-II and PETS for different task-domain pairs.

4. Ground Truth Data

Ground truthing is the process of manually marking what an algorithm is expected to output. Creating a reliable and consistent reference annotation against which algorithm outputs are scored is an extremely involved procedure. Establishing a valid reference is mandatory to carry out a systematic and authentic evaluation.

The ground truth annotations in VACE-II are done by *Video Mining Inc.* using ViPER[6] (Video Performance Evaluation Resource), a video truthing tool developed by the University of Maryland. The source video is in MPEG-2 standard in NTSC format encoded at 29.97 frames per second at 720 x 480 resolution. Currently, every I-frame (every 12 or 15 frames) in the video is ground truthed.

The annotation approach for marking objects varies depending on the task de nition and the properties of the video source. Domain speci c characteristics such as spatial resolution of objects, time duration of objects' presence, object movement in the sequence and shot boundaries are few such video attributes that in uence this decision. For these reasons, the annotation method is classi ed into two broad categories in VACE [6]:

1. Object bounding annotations – limits of the box edges are based on features of the target object
   - Simple object box (e.g. oriented bounding box for *vehicle* tasks in UAV, *face* and *text* tasks in broadcast news)
   - Composite object box (e.g. head & body box for *person* tasks in meetings)
2. Point annotations – location is based on the features of the target object
   - Single point (e.g. *hand* tasks in meetings)

Figure 1 shows examples of annotations for few representative VACE tasks.

Face & Text Annotations | Hand Annotations

Person Annotations | Vehicle Annotations

Figure 1: Sample Annotations for VACE Tasks.

[6] http://viper-toolkit.sourceforge.net

In addition to the bounding box location, extent, and orientation each object is associated with a set of descriptive attributes, characterizing the region with meta data that are useful during the analysis of evaluation results. For instance, a face object is annotated with the following set of attributes –

- Visible (boolean type)
  - 'TRUE' if 1 eye, nose and part of the mouth is seen
  - 'FALSE' otherwise
- SYNTHETIC (boolean type)
  - 'TRUE' if it is an arti cial face
  - 'FALSE' otherwise
- HEADGEAR (boolean type)
  - 'TRUE' if the person is wearing goggles/caps
  - 'FALSE' otherwise
- AMBIGUITY FACTOR (integer type)
  - '0' = Conclusive evidence – when eyes, nose and mouth are clearly seen
  - '1' = Partially conclusive evidence – when two out of three features are seen
  - '2' = Completely inconclusive – when only one or none of the three features can be seen

The ground truthing process in PETS is more involved when compared to that of VACE. This is reasonable considering the point that producing the reference annotation for object segmentation requires demarcation of the foreground which is signi cantly intricate than just selecting bounding boxes. The University of Reading ground truth annotation tool from the AVITRACK[7] project was adapted to PETS Metrics to generate the reference data. Currently, every 10th frame in the video is annotated. Figure 2 shows an example of the PETS ground truth for the segmentation task[8].

Objects are labeled with different tags (*unoccluded object boundaries*, *partial occlusions* and *complete occlusions*) based on the extent of the visible region. Cases where the target is split by an occluding object are marked with a ag. While the binary bitmap is used to measure the precision of object segmentation, a bounding box is marked to de ne the extent of an object. The bounding box coordinates are used in the evaluation of tracking. Figure 3 shows an example of the PETS reference annotation for the tracking task[9].

[7] http://www.avitrack.net

[8] The raw image in this example is from [1]

[9] This example is from the PETS On-Line Evaluation Service website – http://petsmetrics.net

Sample Image from Video.

Corresponding Ground Truth.

Figure 2: Sample Annotation for PETS Motion Segmentation Task.

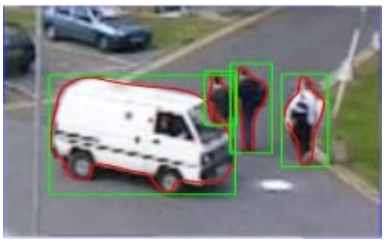


Figure 3: Sample Annotation for PETS Tracking Task.

From the above discussions, it can be noted that among other factors, the de nition of the task being evaluated plays a major role in deciding how the ground truth is generated.

5. Performance Metrics

In VACE, we have developed a suite of diagnostic measures that capture different aspects of an algorithm's performance [3] and a comprehensive measure which captures many aspects of the task in a single score [4, 5]. The diagnostic measures are useful to the researchers in their failure analysis while the comprehensive measures will be used to provide a summative analysis of overall system performance. All metrics are performance measures normalized between 0 and 1, with 0 being the worst and 1 being the best.

Depending on the annotation approach for a speci c task-domain pair, the evaluations can be classi ed into two categories:

1. **Area-based metrics for object bounding annotations**

   An area-based metric, that is based on spatial overlap between ground truth objects and system output objects to generate the score, is used in the case of an object bounding annotation. For the detection task, the metric (Sequence Frame Detection Accuracy, *SFDA*) captures both the detection accuracy (misses and false alarms) and the detection precision (spatial alignment). Similarly, for the tracking task, both the tracking accuracy (number of correct trackers) and the

METRIC	DESCRIPTION
Negative Rate (NR)	measures the pixel-wise mismatches between the ground truth and the system output in each frame
Misclassi cation Penalty (MP)	measures the segmentation performance on an object basis. Penalty for misclassi- ed pixels is based on distance from an object's boundary
Rate of Misclassi cations (RM)	measures the average misclassi ed pixel's distance to an object boundary
Weighted Quality Measure (WQM)	measures the spatial alignment error between the ground truth and the system output as a weighted average of false positive and false negative pixels

Table 2: Motion Segmentation Metrics Currently Implemented in PETS Metrics.

tracking precision (spatial and temporal accuracy) are measured in a single score (Average Tracking Accuracy, *ATA*).

2. **Distance based metrics for point annotations**
   The distance-based metrics (Sequence Frame Detection Accuracy – Distance-based & Average Tracking Accuracy – Distance-based) are parallel to the area-based metrics. The only difference being the fact that in computations for spatial proximity, the spatial overlap between bounding boxes is replaced by a distance measure between corresponding points.

In PETS, the motion segmentation metrics are formulated at the pixel level. Unlike VACE, all metrics are coined as error measures, meaning lower the score, better the performance. Table 2 briefs the four metrics that are currently implemented in PETS Metrics for motion segmentation evaluation [2, 7].

The following ve criteria are used in case of the tracking task: [1]

1. Percentage of dataset tracked – is the ratio of the number of frames in which the algorithm tracked the object to the total number of frames. Evaluation is terminated when a track is lost once.

2. Average overlap between bounding boxes – is the percentage overlap between ground truth and system output bounding boxes over the percentage of dataset tracked.

3. Average overlap between bitmaps – measures the bitmap generated by the algorithm to the ground truth object bitmap.

4. Average *chamfer* distance using the ground truth object bitmap to compute the distance transform against the algorithm generated bitmap.

5. Average *chamfer* distance using the algorithm generated bitmap to compute the distance transform against the ground truth object bitmap.

A debate on using multiple performance metrics vs. a comprehensive metric is dif cult to arbitrate. The task is challenging primarily because of the con icting requirements between algorithm developers and end users. From the research community point of view, multiple metrics aid them in debugging their algorithm by identifying failure components. However, from the perspective of an end user who is presented with numerous systems to choose among, comprehensive metrics assist them in their initial analysis through which entirely na¨ve systems can be instantly discarded. A deeper analysis of selected algorithms can later be done using diagnostic measures to choose the right system for a given application.

6. Discussion

From the above sections, it is clear that a common ground needs to be established before the initiation of a collective effort. In this section, we present some of the factors to be considered during such a decision of collaboration between two programs.

1. *Data domain* – The target domain being addressed in each of the programs should be of suf cient interest to both the projects.

2. *Task de nition* – The task de nitions should be comparable, i.e. the goal of the program in the given domain should be compatible.

3. *Data exchange format* – It is mandatory to agree on a common data format for both ground truth and algorithm output. This is required for a smooth transition in using tools developed in each program.

4. *Tools* – It is optional to agree on the usage of common tools (ground truth authoring tool, scoring tool) as long as the data format is the same. By means of technology transfers, improvements can be made to tools developed in each program.

5. *Ground truth data* – It is necessary that the ground truth annotations are done in a similar manner. Since

ground truth is intended to be what an algorithm is expected to generate, differences in the ground truthing approaches will prohibit cross evaluations on different datasets.

6. *Metrics* – Metrics from each program can be retained as long as the quantities they measure are equivalent, which essentially depends on the task de nitions. Through continuing collaboration, the right set of metrics that is most representative of the system performance can be identi ed.

References

[1] J. Aguilera, H. Wildenauer, M. Kampel, M. Borg, D. Thirde, and J. Ferryman. Evaluation of Motion Segmentation Quality for Aircraft Activity Surveillance. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 293–300, October 2005.

[2] R. Collins, X. Zhou, and S. K. Teh. An Open Source Tracking Testbed and Evaluation Web Site. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (WAMOP-PETS)*, pages 17–24, January 2005.

[3] R. Kasturi, D. Goldgof, P. Soundararajan, and V. Manohar. (Supplement document) Performance Evaluation Protocol for Text and Face Detection & Tracking in Video Analysis and Content Extraction (VACE-II). Technical report, University of South Florida, 2004.

[4] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, M. Boonstra, and V. Korzhova. Performance Evaluation Protocol for Text, Face, Hands, Person and Vehicle Detection & Tracking in Video Analysis and Content Extraction (VACE-II). Technical report, University of South Florida, 2005.

[5] V. Manohar, P. Soundararajan, H. Raju, D. Goldgof, R. Kasturi, and J. Garofolo. Performance Evaluation of Object Detection and Tracking in Video. In *Proceedings of the Seventh Asian Conference on Computer Vision*, volume 2, pages 151–161, January 2006.

[6] H. Raju, S. Prasad, and R. Sharma. Annotation Guidelines for Video Analysis and Content Extraction (VACE-II). Technical report, Video Mining Inc., 2006.

[7] D. Young and J. Ferryman. PETS Metrics: On-Line Performance Evaluation Service. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 317–324, October 2005.

Performance Evaluation of Object Detection and Tracking Systems

Faisal Bashir, Fatih Porikli
Mitsubishi Electric Research Laboratories, Cambridge, MA 02139
fatih@merl.com

Abstract

This paper presents a set of metrics and algorithms for performance evaluation of object tracking systems. Our emphasis is on wide-ranging, robust metrics which can be used for evaluation purposes without inducing any bias towards the evaluation results. The goal is to report a set of unbiased metrics and to leave the final evaluation of the evaluation process to the research community analyzing the results, keeping the human in the loop. We propose metrics from statistical detection and estimation theory tailored to object detection and tracking tasks using frame-based as well as object-based evaluation paradigms. Object correspondences between multiple ground truth objects to multiple tracker result objects are established from a correspondence matrix. The correspondence matrix is built using three different methods of distance computation between trajectories. Results on PETS 2001 data set are presented in terms of 1st and 2nd order statistical descriptors of these metrics.

1. Introduction

The issue of object detection and subsequent tracking from video sequences is a fundamental problem in computer vision. The problem started to gain attention in the wider community of computer vision more than a decade ago. Today, the issue receives more intense pursuit from the narrower but more focused visual surveillance community. In video surveillance domain, object detection and tracking is of central importance for any modern video surveillance and behavioral analysis system. Automated video surveillance systems constitute a network of video sensors observing people as well as other moving and interacting objects in a given environment for patterns of normal/abnormal activities, interesting events, and other domain-specific goals. A vital role envisioned for the modern video surveillance systems is their use as an active tool towards crime prevention, law-enforcement, and pre-emptive interest protection [1]. This is in sharp contrast with most existing systems used mainly as forensic tools for "after the fact" investigation. Automated video surveillance is attractive because it promises to replace the more costly option of staffing video surveillance monitors with human observers. This promise of automated video surveillance can easily turn into its pitfall if the systems don't perform to the desired level. This expected performance level tends to be quite high in case of trained and attentive human operators on a well-staffed facility. On the other hand, the problem of robust object detection and tracking is even harder to address given the requirement that the video surveillance systems have to operate in widely varying weather conditions and all time periods. This situation of high performance expectations and stringent requirements places a minimal margin of error on the performance of these video surveillance systems.

The issue of evaluating the performance of video surveillance systems is becoming more important as more and more research effort is drawn into object detection and tracking. It's a natural question to ask whether there has been quantifiable progress in the form of robust, commercial-grade video surveillance systems as a result of past and ongoing research in this direction. This paper addresses the issue of comprehensive performance evaluation of automatic object detection and tracking systems. We propose several performance evaluation metrics for quantitative assessment of the performance of video surveillance systems. Based on signal detection and estimation theory, these metrics are extended for correct application towards performance evaluation of video surveillance systems. These metrics are evaluated after establishing correspondences between ground truth and tracker result objects. These many-to-many correspondences are established based on association matrices computed from three different methods of trajectory distance computations. The rest of this paper is organized as follows: Section 2 presents a review of the recent and ongoing activity in the domain of performance evaluation for tracking systems; Sections 3 details our evaluation metrics as well as algorithms of data association using correspondence matrices; Section 4 briefly outlines the two different tracker modules developed by us that will be evaluated using the metrics; results of this performance evaluation setup are also reported in this Section; finally, Section 5 concludes the report with summary and future work directions.

2. Related Work

Initial efforts towards performance evaluation of video detection and tracking systems began with the workshops dedicated to the topic, namely PETS (performance evaluation of tracking and surveillance) series of workshops. The main focus of the workshop in early stages was to provide a standard benchmark datasets for participants to evaluate their systems and report results on industry-standard datasets. Later on, the emphasis has

shifted more towards standard metrics used to compute the results of evaluation process. Finally, a recent trend is providing online access portals for research community to submit their intermediate results based on object detection and tracking. The online portal administrators then evaluate the system performance based on standard metrics to compare the candidate algorithms.

As alternative to manual ground truth generation, Black *et al* [3] propose to use pseudo synthetic video to evaluate tracking performance. They synthetically vary the perceptual complexity of tracking task by adding occlusions and inserting increasing number of agents in the scene. Results of object detection and tracking are presented based on metrics derived from a number of sources. These metrics include 'tracker detection rate' (TRDR), 'false alarm rate' (FAR), 'track detection rate' (TDR) and 'track fragmentation' (TF). Lisa *et al* [6] propose algorithms for matching ground truth tracks and system generated tracks and compute performance metrics based on these correspondences. They measure the performance of their system under several different conditions including: indoor/outdoor, different weather conditions and different cameras/view-points. Results on background subtraction and tracking evaluation are reported on the above as well as on the standard PETS 2001 datasets. Stefan *et al* [5] form the correspondence between ground truth and detected objects by minimizing distance between the centroids of ground truth and detected objects. They compute a set of performance metrics including false positive track rate, false negative track rate, average position error, average area error, object detection lag, etc.

An emerging trend in the performance evaluation systems is online portals and websites where contributors can upload the results of their detection and tracking systems in a standard format (mostly in eXtensible Markup Language, XML). The results of various algorithms are then tested for standard performance evaluation metrics to generate results for the comparison of different systems. Collins *et al* [4] report a tracking test-bed to run and log tracking algorithms and their results in real-time. The testbed allows a tracking experiment to be repeated from the same starting state using different tracking algorithms and parameter settings, thereby facilitating comparison of algorithms. On the associated website, tracking results can be uploaded for evaluation using the standard test metrics. A similar approach is taken in [2], where the online service allows researchers to submit their algorithm results on video segmentation to view their algorithm's performance against a set of standard metrics. The approach has been used towards the problem of motion segmentation using seven motion segmentation algorithms to date.

3. Performance Evaluation Metrics

This section outlines the set of performance evaluation metrics we have implemented in order to quantitatively analyze the performance of our object detection and tracking system. We propose a set of both frame-based and object-based metrics for the evaluation. The ground truth information is represented in terms of the bounding box of object for each frame. Similarly, the results of object detection and tracking systems are in terms of the detected or tracked object's bounding box. At the time of evaluation, we employ different strategies to robustly test if the overlap between ground truth and system's results occurs. The simplest form of overlap is testing to see if the system result's centroid lies inside the ground truth object's bounding box. This issue is discussed later in the Section.

Frame-based metrics are used to measure the performance of surveillance system on individual frames of a video sequence. This does not take into account the response of the system in preserving the identity of the object over its lifespan. Each frame is individually tested to see if the number of objects as well as their sizes and locations match the corresponding ground truth data for that particular frame. The results from individual frame statistics are then averaged over the whole sequence. This represents a bottom-up approach. On the other hand, the object-based evaluation measures take the whole trajectory of each object into consideration. Here, the individual tracks of objects which are automatically detected and then tracked over their lifespan are analyzed as separate entities. The various ways of finding the best correspondence (association) between individual ground truth tracks and tracker result tracks are analyzed. Finally, based on a particular association, success and error rates are computed and accumulated for all the objects. This represents a top-down approach. We propose metrics for both the approaches in this section and then present results of the evaluated detection and tracking system in the next section.

3.1. Frame-based Metrics

Starting with the first frame of the test sequence, frame-based metrics are computed for every frame in the sequence. From each frame in the video sequence, first a few true and false detection and tracking quantities are computed.

True Negative, TN: Number of frames where both ground truth and system results agree on the absence of any object.

True Positive, TP: Number of frames where both ground truth and system results agree on the presence of one or more objects, and the bounding box of at least one or

more objects coincides among ground truth and tracker results.

False Negative, FN: Number of frames where ground truth contains at least one object, while system either does not contain any object or none of the system's objects fall within the bounding box of any ground truth object.

False Positive, FP: Number of frames where system results contain at least one object, while ground truth either does not contain any object or none of the ground truth's objects fall within the bounding box of any system object.

In the above definitions, the two bounding boxes are said to be *coincident* if the centroid of one of the boxes lies inside the other box. Also, total ground truth TG is the total number of frames for the ground truth objects and TF is the total number of frames in the video sequence. Once the above defined quantities are calculated for all the frames in the test sequence, in the second step, the following metrics are computed:

$$\text{Tracker Detection Rate (TRDR)} = \frac{TP}{TG} \quad (1)$$

$$\text{False Alarm Rate (FAR)} = \frac{FP}{TP+FP} \quad (2)$$

$$\text{Detection Rate} = \frac{TP}{TP+FN} \quad (3)$$

$$\text{Specificity} = \frac{TN}{FP+TN} \quad (4)$$

$$\text{Accuracy} = \frac{TP+TN}{TF} \quad (5)$$

$$\text{Positive Prediction} = \frac{TP}{TP+FP} \quad (6)$$

$$\text{Negative Prediction} = \frac{TN}{FN+TN} \quad (7)$$

$$\text{False Negative Rate} = \frac{FN}{FN+TP} \quad (8)$$

$$\text{False Positive Rate} = \frac{FP}{FP+TN} \quad (9)$$

Figure 1 shows two of the metrics, TRDR and FAR computed from the six combinations of trackers and detectors on the PETS data set evaluated in this paper. The notched box plots in this figure clearly show the high detection rate and low false alarm rate of the video surveillance system evaluated in this paper. Each vertical bar represents one tracker – detector combination evaluated for the whole data set.

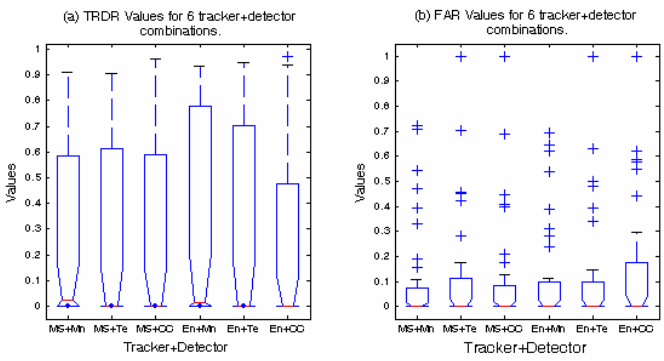


Figure 1: TRDR (a) and FAR (b) for 6 combinations of trackers and detectors.

3.2. Object-based Metrics

Object-based evaluation computes the metrics based on the complete trajectory and lifespan of the individual system and ground truth tracks. Since a given ground truth track could correspond to more than one system tracks and likewise, a correspondence mapping has to be established first. Based on this mapping between object tracks, the frame-based as well as object-based metrics are computed. Figure 2 shows the procedure to compute the core metrics (TN, TP, FN and FP) from object correspondences.

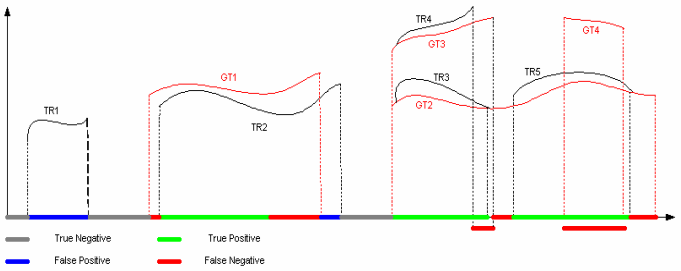


Figure 2: Definitions of 'true negative', 'true positive', 'false negative' and 'false positive'. Note how metrics for multiple objects in a single frame are computed.

The first set of metrics we present are based on simple threshold-based correspondence. For each common frame between a system track and ground truth track, the Euclidean distance between their centroids is computed. The cumulative Euclidean distance is then normalized by the total number of overlapping frames between the ground truth – system track pair under investigation. Finally, two ground truth – system track pairs are declared corresponding if their total normalized distance is within a threshold. Once the correspondence is established, we compute the true positive (TP), false positive (FP) and total ground truth (TG) as explained previously in the context of frame-based metrics. The tracker detection rate

(TRDR) and false alarm rate (FAR) are then computed as before. We also compute the object tracking error which is the average discrepancy between the ground truth bounding box centroid and the centroid of the system result:

$$\text{Object Tracking Error (OTE)} = \frac{1}{N_{rg}} \sum_{i \in g(t_i) \wedge r(t_i)} \sqrt{\left(x_i^g - x_i^r\right)^2 + \left(y_i^g - y_i^r\right)^2} \qquad (10)$$

where N_{rg} represents the total number of overlapping frames between ground truth and system results, x_i^g represents the x-coordinate of the centroid of object in i^{th} frame of ground truth, x_i^r represents the x-coordinate of the centroid of object in i^{th} frame of tracking system result.

The above approach is good enough for quick and simple evaluations. A detailed analysis of the kind of errors most frequently made by practical object tracking systems calls for more sophisticated measures of object correspondence. To provide motivation for our approaches towards object correspondence, imagine the hypothetical scenario in Figure 3. This figure shows two scenarios of error highlighted in a hypothetical yet realistic setting. In (a) and (b), the figure on left hand side shows ground truth tracks and labeling while the figure in the right hand side shows system output on the same sequence. Each of the two figures also shows its binary correspondence map in figures (c) and (d), which reveal the four types of errors consistently. In each binary correspondence map, columns represent the system detected and tracked objects, while rows represent the ground truth detected and tracked objects. A dark cell in the correspondence map shows correspondence between the two objects. We have two algorithms to establish correspondence: many-to-many correspondence associates the tracks in ground truth and tracker results with multiple associations as long as the temporal and spatial overlap between the two tracks is within a threshold; unique correspondence associates the two tracks with most overlap allowing one ground truth track to associate with only one system track and vice versa. For many-to-many correspondence, we look at the binary correspondence map from two views. First we look at the rows of the binary map for each ground truth track to obtain all the matching system tracks. This procedure captures track false negatives and fragmentation errors. In the second pass, we look at the columns of the same binary map to associate each system tracks with all ground truth tracks it matches to. This procedure reveals track false positive errors and track merge errors. For unique correspondence, we use the same two-pass approach, but this time on the two different correspondence maps. In the first pass, each ground truth track is matched against all the tracker result tracks. The resulting correspondences are shown in left hand side of the figure for unique correspondence. This reveals track false negative but fails to capture track fragment errors (no multiple associations allowed). In the second pass, each tracker result is matched against all ground truth tracks. The resulting correspondences are shown in the binary maps on the right hand sides. This reveals track false positive but fails to capture track merge errors.

Figure 3(a) shows an example situation where track merge error (object 1 and 2 joined as 1) and track false positive error (system detects an extra object, 2) occurs. Also, there is no track fragmentation error (one ground truth object split into two) or track false negative error (a ground truth object missed). Here, object 1 leaves the scene in the right hand center corner of the camera field of view (FOV). After that, a second object enters from a close location and moves towards the left hand side of the FOV. A tracker system which bases its tracking on the history of motion pattern (using Kalman prediction or particle filtering based approach), always has a certain time lag to allow for consistent tracking in the event of occlusion. This property can result in object labeling error, where the tracker system mistakes the object 2 entering the scene from neighboring location not too long after object 1 has left. In this situation, the tracker system can mistake object 2 for object 1 and merge the two objects into 1. This error resulting from object label ambiguity causes the corresponding system tracks to be labeled as merged tracks. In this particular example, ground truth tracks 1 and 2 will be labeled as merged tracks. Similarly, Figure 3(a) shows another error visible in the system track's output. The system track 2 has no corresponding ground truth. It could have resulted from some noise in the video sequence data. Both these errors are captured in many-to-many correspondence maps of (c). Here, column 1 reveals that tracker track 1 is matched to both ground truth tracks 1 and 2, resulting in track merge error. Also, tracker track 2 has no corresponding ground truth track, resulting in track false positive. In the unique correspondence results, the left hand correspondence map is the same as for many-to-many correspondence resulting in the same false positive, but no merge errors as expected. Also, both these algorithms yield no track fragment and false negative errors.

On similar lines, Figure 3 (b) shows a situation where track fragment error and track false negative errors occur, but there are no track merge and track false positive errors. In this situation, tracker loses the object 1 for a brief period of time due to occlusion. After a while it detects it again, but this time assigns it a different label due to object label ambiguity. Also, ground truth track 2 is not even detected. Both the errors resulting in this scenario are captured in the many-to-many binary correspondence maps shown in figure (d). Unique

correspondence misses the track fragment error as expected. Also, no track false positive or track merge errors are detected.

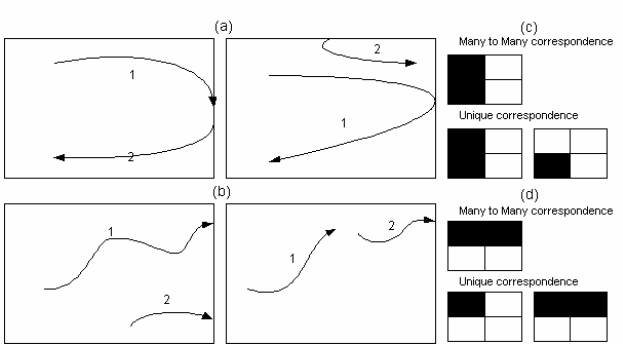


Figure 3: (a) Example of a track merge error scenario. (b) Example of a track fragment error scenario. (c) & (d) Object correspondence maps for the two scenarios.

Figure 4 outlines the algorithm used for generating correspondences. For simplicity, it shows the complete portion for first pass detailing system results to ground truth mapping. The second pass, which matches ground truth to system results, is a straightforward symmetrical implementation and is skipped for brevity. In this algorithm, the core problem is detecting whether the two bounding boxes, one from system track results and the other from ground truth coincide or not. To answer this question, we have implemented three strategies as explained below.

The first strategy is based on Euclidean distance between centroids of the two boxes. If the distance is between a preset threshold, the two boxes are said to be coincident and an association declared between the two objects. The second approach tests if the centroid of one of the boxes is within the bounds of the other box. It inflates the first box by a fixed percentage in both x- and y- directions, and then tests if the centroid of second box lies inside the first one or not. Finally, the third approach computes the ratio of the intersection and union of the two boxes. If this ratio is with a fixed threshold between 0 and 1, the two objects are declared to have a match.

4. Simulation Environment and Results

This section outlines our simulation test environment, test dataset and results obtained using the metrics discussed in the previous section.

4.1 Automatic Object Detection and Tracking Systems

This section very briefly outlines the automatic object detection and tracking system for video surveillance that we have tested. These systems will be evaluated based on metrics detailed in the previous section. The video surveillance system consists of a background generation module coupled with several object detection and tracking modules in an easy-to-bind API (application programming interface). This allows us to provide a convenient user interface for testing various combinations of object detection and tracking modules. We have tested three object detection modes and two tracking modules for our surveillance system. The first mode is manual detection where the user draws an object template (bounding rectangle) on an initial frame; the second detection mode is template detection from background estimation; the third detection mode is connected component-based detection from background estimation. We have tested two of recently reported tracking systems for performance evaluation. The first tracking system is 'multi-kernel mean-shift' which uses multiple meanshift kernels centered at the high motion areas. Details about this system can be found in [9]. The second system is 'ensemble tracker' which poses tracking as a binary classification problem, where an ensemble of weak classifiers is trained online to distinguish between object and background [10]. All the results in this paper are tested on the output generated by these two trackers in conjunction with three detectors to yield six combinations for evaluation.

```
for each tracker track i
{
  for each GT track j
  {
    for each overlapping frame k
    {
      tov[i][j] ++
      if (box[i][k] coincides box[j][k])
        sov[i][j] ++
    }
    tov[i][j] /= totalframes
    sov[i][j] /= totalframes
    fov[i][j] = α1 tov[i][j] + α2 stov[i][j]
    if (fov[i][j] > T1)
      TR_GT[i] ++
  }
  if (TR_GT[i] == 0)
    TFP ++
  if (TR_GT[i] > 1)
    TME += TR_GT[i]
}
```

Figure 4: Algorithm for many-to-many tracker to ground truth matching.

4.2. Simulation test-bed

In order to perform the evaluation tests of our system against ground truth data, we have developed a GUI-driven desktop application (different from the video surveillance system application mentioned in Section 4.1). For a test sequence, the user can load object localization information from ground truth and tracker results files. The data from ground truth and tracker results is stored in the eXtensible Markup Language (XML) format. The specific XML format followed is that proposed by Computer Vision Markup Language (CVML) [8]. This xml-based file format ensures that results from different teams can be tested on our systems without any portability issue. The cvml-parser module of the application parses the input data files from both tracker results and ground truth to fill in internal data structures. The two results are then compared according to the rules and metrics discussed in Section 3. An attractive feature of the application is batch processing mode, where a set of sequences to be evaluated can be specified in a separate file and the system performs evaluation on all these sequences automatically. Although the processing time for each sequence is quite manageable (around 15-20 seconds per sequence on a Pentium-IV, 3.0 GHz desktop), running the application for hundreds of sequences becomes tedious. In batch processing mode, the application can be left running unsupervised to generate results. The outputs of individual sequences are written out as soon as they are available for each sequence. At the end, the output of batch processing is written out along with means and variances for each metric.

4.2. Dataset

We have used a mix of both standard and in-house datasets for our evaluation purposes. The standard dataset is based on the PETS 2001 dataset. We have also tested our system on some other in-house and publicly available sequences of varying length. The total number of sequences tested in our evaluation experiments is around 40. The test sequences consist of more than 50,000 frames and depict both indoor and outdoor scenarios; partial and full occlusions; various object types, such as pedestrians, vehicles, bicycles, etc. The results of these experiments are detailed next.

4.3. Results

We have tested automatic object detection and tracking systems using test metrics discussed in the previous section on the dataset in batch processing mode. Partial results of this evaluation based on TRDR and FAR are reported in Figure 1. The full results of this evaluation in the form of means and variances of each metric are presented in Table 1 for 'multi-kernel meanshift' tracker and in Table 2 for 'ensemble tracker'.

5. Summary and Conclusions

In this report, we have addressed the issue of unbiased performance evaluation of object detection and tracking systems. We have made contributions in two areas: a set of novel performance evaluation metrics have been proposed for detection and tracking; novel methods of establishing correspondence between ground truth tracks and system generated tracks have been proposed. Our set of metrics contains both frame-based metrics (to test the performance of detection system) as well as object-based metrics (to test the tracking capabilities including consistent object labeling). Experiments have been conducted on a standard dataset containing more than 50,000 frames. The cumulative results of these experiments in terms of mean and variance values for each metric are reported.

References

[1]. Robert T. Collins, Alan J. Lipton, Takeo Kanade, "Introduction to the Special Section on Video Surveillance", IEEE Transactions on PAMI, Vol. 22, No. 8, August 2000, pp. 745 - 746.

[2]. David P. Young, James M. Ferryman, "PETS Metrics: On-Line Performance Evaluation Service", Proceedings 2nd Joint IEEE International Workshop on VS-PETS, Beijing, October 15-16, 2005.

[3]. James Black, Tim Ellis, Paul Rosin, "A Novel Method for Video Tracking Performance Evaluation", The Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, October, Nice, France, pp. 125-132. (2003).

[4]. Robert Collins, Xuhui Zhou, Seng Keat The, "An Open Source Tracking Testbed and Evaluation Web Site", IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2005), January 2005.

[5]. Stefan Muller-Schneiders, Thomas Jager, Hartmut S. Loos, Wolfgang Niem, "Performance Evaluation of a Real Time Video Surveillance Systems", Proceedings 2nd Joint IEEE International Workshop on VS-PETS, Beijing, October 15-16, 2005.

[6]. Lisa M. Brown, Andrew W. Senior, Ying-li Tian, Jonathan Connell, Arun Hampapur, "Performance Evaluation of Surveillance Systems under Varying Conditions", IEEE Int'l Workshop on Performance Evaluation of Tracking and Surveillance, Colorado Jan., 2005 .

[7]. Vladimir Y. Mariano, Junghye Min, Jin-Hyeong Park, Rangachar Kasturi, David Mihlacik, Huiping

Li, David Doermann, Thomas Drayer, "Performance Evaluation of Object Detection Algorithms", International Conference on Pattern Recognition, pp. 965-969, 2002 .

[8]. Thor List, Robert B. Fisher, "CVML – An XML-based Computer Vision Markup Language", 17th International Conference on Pattern Recognition (ICPR 2004), Vol. 1, pp. 789-792. 2004.

[9]. Fatih Proikli, Oncel Tuzel, "Multi-Kernel Object Tracking", IEEE International Conference on Multimedia and Expo (ICME 2005), pp. 1234-1237, July 2005.

[10]. Shai Aviden, "Ensemble Tracking", IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), ISSN: 1063-6919, Vol. 2, pp. 494-501, June 2005.

Table 1: Tabular results of performance evaluation on 'Multi-kernel Meanshift Tracking System' with various automatic object detection methods for around 40 video sequences from PETS and in-house data set.

Metric			Manual Meanshift		Template Meanshift		CC Meanshift	
			Mean	Var	Mean	Var	Mean	Var
Frame-based	TRDR		0.480929	0.060065	0.545417	0.040047	0.513411	0.07145
	FAR		0.225161	0.043955	0.2133	0.038476	0.225849	0.051534
	Detection Rate		0.480929	0.060065	0.545417	0.040047	0.513411	0.07145
	Specificity		0.654807	0.054624	0.581161	0.059394	0.617162	0.085862
	Accuracy		0.547126	0.04946	0.589513	0.026975	0.551378	0.050211
	Positive Predictive Value		0.774839	0.043955	0.7867	0.038476	0.774151	0.051534
	Negative Predictive Value		0.314004	0.046597	0.285508	0.029245	0.296549	0.051285
	False Negative Rate		0.519071	0.060065	0.454583	0.040047	0.486589	0.07145
	False Positive Rate		0.345193	0.054624	0.418839	0.059394	0.382838	0.085862
Object-based	Euclidean Threshold	TRDR	0.646242	0.068983	0.717953	0.030324	0.813559	0.035763
		FAR	0.203123	0.043281	0.166291	0.023183	0.17383	0.04095
		Detection Rate	0.696041	0.061334	0.610564	0.038849	0.787851	0.047174
		Specificity	0.688858	0.04373	0.627918	0.048623	0.635475	0.078635
		Accuracy	0.734413	0.031448	0.779488	0.012852	0.834226	0.020197
		Positive Predictive Value	0.796877	0.043281	0.833709	0.023183	0.82617	0.04095
		Negative Predictive Value	0.528668	0.062497	0.36017	0.061787	0.561818	0.079523
		False Negative Rate	0.29265	0.056773	0.381619	0.037059	0.204333	0.042613
		False Positive Rate	0.311142	0.04373	0.372082	0.048623	0.364525	0.078635
		Track False Positive	0.212707	0.167463	1.251522	6.453521	1.069031	3.504285
		Track False Negative	0.391796	0.508262	0.168998	0.15607	0.060968	0.072883
		Track Merge Error	0	0	0.05093	0.048336	0.10803	0.09636
		Track Fragment Error	0.235736	0.180164	0.82442	0.765946	0.286737	0.420579
	Centroid in Rectangle	TRDR	0.597409	0.084026	0.675086	0.042262	0.65172	0.101447
		FAR	0.255427	0.070712	0.209608	0.046633	0.33233	0.103508
		Detection Rate	0.699214	0.071628	0.65212	0.048812	0.703334	0.100348
		Specificity	0.715978	0.02892	0.643301	0.049125	0.603593	0.063744
		Accuracy	0.803126	0.033313	0.804321	0.015131	0.842247	0.022605
		Positive Predictive Value	0.744573	0.070712	0.790392	0.046633	0.66767	0.103508
		Negative Predictive Value	0.621375	0.087312	0.470762	0.074644	0.658076	0.08775
		False Negative Rate	0.252056	0.049838	0.340063	0.046372	0.192914	0.047391
		False Positive Rate	0.284022	0.02892	0.356699	0.049125	0.396407	0.063744
		Track False Positive	0.807649	0.162651	1.911387	8.703765	1.735889	4.240596
		Track False Negative	0.885987	0.53496	0.545911	0.672115	0.582771	0.926545
		Track Merge Error	0	0	0	0	0	0
		Track Fragment Error	0.248278	0.299929	0.70882	0.780362	0.105397	0.094289
	Area Ratio Overlap	TRDR	0.235838	0.120703	0.431792	0.103333	0.251769	0.102233
		FAR	0.754213	0.13532	0.491049	0.132944	0.738237	0.111596
		Detection Rate	0.307273	0.187835	0.448761	0.105656	0.353636	0.143692
		Specificity	0.669942	0.014249	0.565374	0.041517	0.532402	0.02464
		Accuracy	0.971789	0.004212	0.827823	0.026912	0.899033	0.022285
		Positive Predictive Value	0.245787	0.13532	0.508951	0.132944	0.261763	0.111596
		Negative Predictive Value	0.920536	0.029111	0.613843	0.123858	0.814865	0.070679
		False Negative Rate	0.033428	0.007289	0.265242	0.05317	0.171212	0.057013
		False Positive Rate	0.330058	0.014249	0.434626	0.041517	0.467598	0.02464
		Track False Positive	1.829033	0.899422	2.618809	8.816763	2.591081	7.236438
		Track False Negative	1.885679	0.745644	0.981076	1.285967	1.476057	1.550727
		Track Merge Error	0	0	0	0	0	0
		Track Fragment Error	0	0	0	0	0	0

Table 2: Tabular results of performance evaluation on 'Ensemble Tracking System' with various automatic object detection methods for around 40 video sequences from PETS and in-house data set.

Metric			Manual Ensemble		Template Ensemble		CC Ensemble	
			Mean	Var	Mean	Var	Mean	Var
Frame-based	TRDR		0.583182	0.084726	0.613543	0.060189	0.447507	0.065313
	FAR		0.238632	0.049738	0.210794	0.044481	0.282596	0.056876
	Detection Rate		0.583182	0.084726	0.613543	0.060189	0.447507	0.065313
	Specificity		0.610320	0.051911	0.598695	0.059042	0.532792	0.080033
	Accuracy		0.625534	0.06033	0.642713	0.041356	0.486041	0.051821
	Positive Predictive Value		0.761368	0.049738	0.789206	0.044481	0.717404	0.056876
	Negative Predictive Value		0.359372	0.048083	0.340331	0.036629	0.242104	0.035727
	False Negative Rate		0.416818	0.084726	0.386457	0.060189	0.552493	0.065313
	False Positive Rate		0.389680	0.051911	0.401305	0.059042	0.467208	0.080033
Object-based	Euclidean Threshold	TRDR	0.74025	0.059105	0.736054	0.038256	0.779085	0.069626
		FAR	0.205657	0.047088	0.182218	0.033953	0.22252	0.069671
		Detection Rate	0.792952	0.053303	0.772397	0.033471	0.741613	0.079402
		Specificity	0.647741	0.043084	0.660785	0.043207	0.576553	0.062554
		Accuracy	0.826679	0.018761	0.819638	0.014937	0.854983	0.015251
		Positive Predictive Value	0.794343	0.047088	0.817782	0.033953	0.77748	0.069671
		Negative Predictive Value	0.618142	0.048588	0.531993	0.044368	0.541479	0.097811
		False Negative Rate	0.182036	0.038022	0.219787	0.029152	0.201533	0.048697
		False Positive Rate	0.352259	0.043084	0.339215	0.043207	0.423447	0.062554
		Track False Positive	0.353382	0.319173	1.260737	5.025566	1.548544	5.271998
		Track False Negative	0.493006	0.717945	0.402995	0.562953	0.200675	0.448046
		Track Merge Error	0	0	0.05093	0.048336	0.05093	0.048336
		Track Fragment Error	0.188662	0.153069	0.331578	0.323494	0.627201	1.042936
	Centroid in Rectangle	TRDR	0.657232	0.094717	0.692611	0.054343	0.581208	0.149388
		FAR	0.290587	0.091443	0.225234	0.057475	0.414578	0.149851
		Detection Rate	0.827868	0.062887	0.803629	0.040309	0.583884	0.152444
		Specificity	0.680205	0.031417	0.679881	0.03387	0.565269	0.053782
		Accuracy	0.876542	0.017114	0.830707	0.022513	0.861936	0.027627
		Positive Predictive Value	0.709413	0.091443	0.774766	0.057475	0.585422	0.149851
		Negative Predictive Value	0.741183	0.051225	0.61244	0.052718	0.658277	0.109339
		False Negative Rate	0.139386	0.040342	0.188554	0.035502	0.182695	0.058798
		False Positive Rate	0.319795	0.031417	0.320119	0.03387	0.434731	0.053782
		Track False Positive	0.992348	1.80733	1.835692	7.703544	2.464621	8.711641
		Track False Negative	0.943311	1.853212	0.697301	1.099339	0.832812	2.121629
		Track Merge Error	0	0	0	0	0	0
		Track Fragment Error	0.049037	0.046633	0	0	0.292332	0.265949
	Area Ratio Overlap	TRDR	0.230048	0.095952	0.384071	0.122701	0.179283	0.059764
		FAR	0.746713	0.106427	0.563954	0.154915	0.821946	0.062503
		Detection Rate	0.392012	0.197961	0.505314	0.180036	0.330953	0.159903
		Specificity	0.591268	0.011419	0.627483	0.017292	0.516166	0.013807
		Accuracy	0.948076	0.013274	0.910729	0.014953	0.888639	0.0217
		Positive Predictive Value	0.253287	0.106427	0.436046	0.154915	0.178054	0.062503
		Negative Predictive Value	0.904852	0.045538	0.805123	0.062678	0.888601	0.028055
		False Negative Rate	0.070059	0.024773	0.097039	0.017687	0.154731	0.069269
		False Positive Rate	0.408732	0.011419	0.372517	0.017292	0.483834	0.013807
		Track False Positive	1.587296	1.775537	2.368109	8.785345	3.405052	11.655499
		Track False Negative	1.587296	1.775537	1.229719	1.468704	1.480912	0.905387
		Track Merge Error	0	0	0	0	0	0
		Track Fragment Error	0	0	0	0	0	0

Performance Evaluation of Frequent Events Detection Systems

X. Desurmont[a], R. Sebbe[b], F. Martin[a], C. Machy[a], J-F. Delaigle[a]
[a]*Multitel A.S.B.L., Av Copernic, 1, 7000 Mons, Belgium.*
[b]*Faculté Polytechniques de Mons, Mons, TCTS, Av Copernic, 1, 7000 Mons, Belgium.*

Abstract

In recent years, the demand on video analysis applications [1, 2] such as video surveillance and marketing is growing rapidly. A number of solutions exist but they need to be evaluated. This point is very important for two reasons: the proof of the objective quality of the system for industrials, the possibility to highlight improvement during research and thus to understand better how the system works to improve it adequately. This paper describes a new algorithm that can evaluate a class of detection systems in the case of frequent events; for example, people detection in a corridor or cars on the motorways. To do so, we introduce an automatic re-alignment between results and ground truth by using dynamic programming. The second point of the paper describes, as an example, a practical implementation of a stand alone system that can perform counting of people in a shopping center. Finally we evaluate the performance of this system.

Keywords: Performance evaluation, event detection, counting people, alignment, dynamic programming.

1. Introduction

Since the last decade, many vision algorithms have been proposed that try to solve the problem of scene understanding in many specific applications. The level of understanding varies highly from only detecting moving objects and outputting their bounding boxes (e.g. the Open Source project "Motion[1]"), to tracking of the objects over multiple cameras, thereby learning common paths and appearance points [3, 4] or depth maps and amount of activity in the scene [5]. Thus, many applications could be derived from these trajectories. Examples are traffic monitoring, the behavior analysis and the counting of people for marketing applications. However to check if a product satisfies the requirement of an application, objective evaluation is needed.

The importance of performance evaluation of Video content Analysis (VCA) algorithms has been addressed by different projects, and has led to various research publications. From the year 2000, the IEEE holds a yearly workshop on Performance Evaluation of Tracking and Surveillance (PETS).

Since a complete VCA system includes multiple semantic levels, evaluation can be performed at certain levels. VCA algorithms that only perform segmentation of moving objects and no tracking over multiple video frames can only be evaluated on their pixel-based segmentation. Algorithms that only output alarms (e.g. car detected) can only be evaluated for proper classification, and not for their segmentation quality. Therefore, the first requirement is to define exactly what should be evaluated for each semantic level. For each level, different metrics are required.

Multiple semantic levels for evaluation can be defined [6].

1) Pixel-level: segmentation in moving objects and static background.

2) Object-based evaluation per frame (object-features including object type, size).

3) Object-based evaluation over an object life time (object-features including speed, trajectory).

4) Behavior of objects; e.g. person entering a room.

In this article we present a new algorithm that evaluates level 4. Some related work exists [7, 8] but is related to rare events including left luggage detection or other alarm triggering. The new presented approach has been designed to cope with frequent event detection that needs automatic re-alignment between result (RS) and ground truth (GT). We propose to use Dynamic Programming (DP).

The paper is organized as follows: Section 2 describes the related work; Section 3 is devoted to the

[1] Project Motion: http://sourceforge.net/projects/motion/

presentation of the new approach. Section 4 introduces a people counting system and gives some results. Section 5 concludes and indicates future work.

2. Related work

A video event detection system is a method that triggers events when something specific happens in the scene. This event could happen when someone leaves a luggage, if there is a fight between two people or just during the crossing of a region (for counting applications). Typically, the output of such a system is a set of time-stamped events. Thus the performance evaluation of the system, given a set of video sequences, is perfomed by comparing the result set and the ground truth set.

The approach of Desurmont *et al* [7], for evaluation of left luggage detection gives a Boolean result for each sequence: they have tested over 217 sequences among which 26 contain a left luggage. The output of the system consists in outputting "positive" or "negative" for each sequence. The overall evaluation result for each sequence is concatenated in two metrics: Detection Rate (DR) and False Alarm Rate (FAR). The major drawback of this approach is that it is not possible to cope with more than one alarm in a sequence; moreover it is not possible to distinguish any delay or anticipation of alarm triggering.

Bruneaut *et al* [8] propose a metric in the framework of Challenge of Real-time Event Detection Solutions (CREDS) in 2005. It matches events of ground truth and result with handling of temporal shift. The major drawback of this method occurs when the events are so frequent that the possibility of anticipated and delayed events entails the overlapping of several events. In this case, an alignment seems to be the solution of the problem. It has been already introduced by Lopresti [9] for performance evaluation of text processing. Indeed, he proposes to use an approximate string matching to align two linear streams of text, one representing Optical Character Recognition (OCR) results and the other representing the ground truth. This can be solved using a DP algorithm where deletions, insertions, and mismatches are charged positive costs, and exact matches are charged negative costs. Lopresti shows that by maintaining the decision used to obtain the minimum in each step, we obtain, in essence, an explanation of the errors that arise in processing the input.

In the field of object tracking, Brown *et al* [10] propose a tracking evaluation with a framework to determine a match between results tracks and ground truth tracks. However, this match is not necessarily the best one. Needham *et al* [11] propose, for the matching, many methods to compare trajectories that can differ by spatial or temporal bias. They also bring up the possibility to use dynamic programming to align trajectories.

3. The dynamic re-alignment

3.1. Introduction

We assume that the possible deviations of the event detection system are a set of false positives, false negatives, delays and anticipations. In practical terms, it means that, sometimes, it allows no match between events of ground truth and results, matches could also be performed with events having different time-stamps. Figure 1 shows an example: α, β, γ and δ are real events of the same type in the ground truth sequence; A, B, C and D are the results of a detection system.

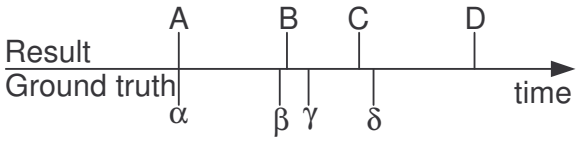


Figure 1. Representation of events in the time-line.

There are 2 basic interpretations that depend on the constraints of matching. When looking at all the events in the same timeline to analyze the system, one will probably match as follow: A-α, B-β, C-δ and thus concludes that there are 3 good detections, 1 false alarm (D) and a misdetection (γ). This possible interpretation I_1 is not the only one but looks the most plausible and suggests a re-alignment of B-β and C-δ. Another interpretation I_2 could match: A-α, B-β, C-γ, D-δ and conclude to neither misdetection nor false alarms but assume a higher delay of the evaluated system. In fact, the most probable interpretation I_1 or I_2 depends directly of the values of the maximum delays (e.g. to allow to match D and δ). That is why we need to add two constraints which are the maximum for the delays and the anticipations.

The aim of the proposed approach is to process this dynamic re-alignment of the system's output according to the ground truth in an objective and automatic way. Each type of event is evaluated independently.

3.2. Costs and matching

The idea is to minimize a global cost of matching events. We define a distance $dist_{i,j}$ between two events i,j $dist_{i,j}=|t_i-t_j|$; t_i and t_j are the timestamps in seconds

of events *i* and *j* respectively. Table 1 and Figure 2 show the result for our example. This distance could also be asymmetric in order to give advantage to anticipation or deletion. More complex distances could be introduced: The spatial position of the event in the scene could help to match better events. We also set a cost to false positive (*FPDist*) and false negative (*FNDist*). These costs could be tuned to decide the maximum delay and anticipation. We setup *FPDist= FNDist=3s.*

From these constraints we can compute the global distance of the two interpretations. Figure 3 shows two paths from the two possible interpretations made in Section 3.1. I_1 has the lower cost. A circle is a match; a cross is a false positive (FP) or false negative (FN).

Table 1. Time of occurrence of events.

Event	α	A	β	B	γ	C	δ	D
Time	0	0	10	11	13	17	18	25

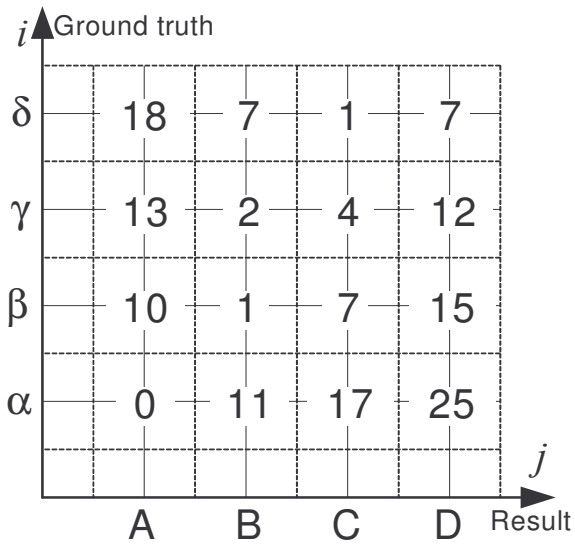


Figure 2. Matrix showing the distances between ground truth (GT) and system result events (RS).

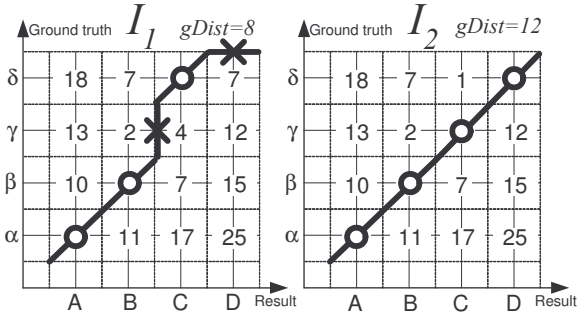


Figure 3. Two different interpretations of system results.

3.3. Dynamic programming

In this section, we will describe how the DP finds automatically the best path. We define $gDist_{i,j}$ as the global distance between the first *i* ground truth events and *j* result events. The initial condition and the recurrence of the DP are:

$$gDist_{0,0} = 0 \tag{1}$$

$$gDist_{i,j} = \min \begin{cases} gDist_{i-1,j} + FNDist & if \quad i > 0 \\ gDist_{i,j-1} + FPDist & if \quad j > 0 \\ gDistM_{i-1,j-1} & if\,(i > 0 \,\&\, j > 0) \end{cases} \tag{2}$$

$$gDistM_{i,j} = gDist_{i,j} + dist_{i,j} \tag{3}$$

$gDistM_{i,j}$ considers a match of *i* and *j* (when two lines of the grid cross). On the other hand $gDist_{i,j}$ deals with FN or FP (when two "dashed" lines of the grid cross). By maintaining the minimum in each step, the dynamic programming algorithm allows to backtrack the best matching set. Figure 4 shows that process and highlights the best path, which is in this case I_1.

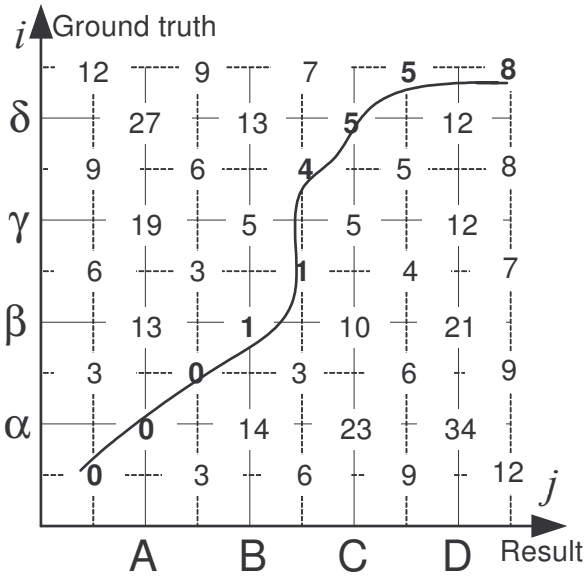


Figure 4. Schema showing the computation of the DP and the backtracking in bold.

3.4 Results display

From the best path, it is straightforward to count the number of matches, the number of false detections and the number of non-detections. It is also possible to display them on a graphical time-line as is shown in the analysis tool implementation in figure 5. At the

bottom, there are the ground truth events, on the top, the results' events. We display two types of events "green" and "blue" as little disks. When events match, there is a line matching them. When an event is not matched (false detection or non detection) there is a red cross.

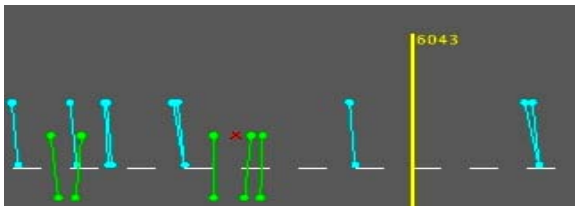


Figure 5. View of a tool that displays matchings.

4. Counting method overview

The goal of the counting method is to count people crossing a region, maintaining 2 counters, one for each direction, positive and negative. The positive direction is provided as an input parameter, as are a number of detection lines, as shown on figure 6. A previous method using foreground extraction and multiple hypothesis tracking of blobs has already been presented in previous work [12]. Other works can be found in PETS 2002 [13].

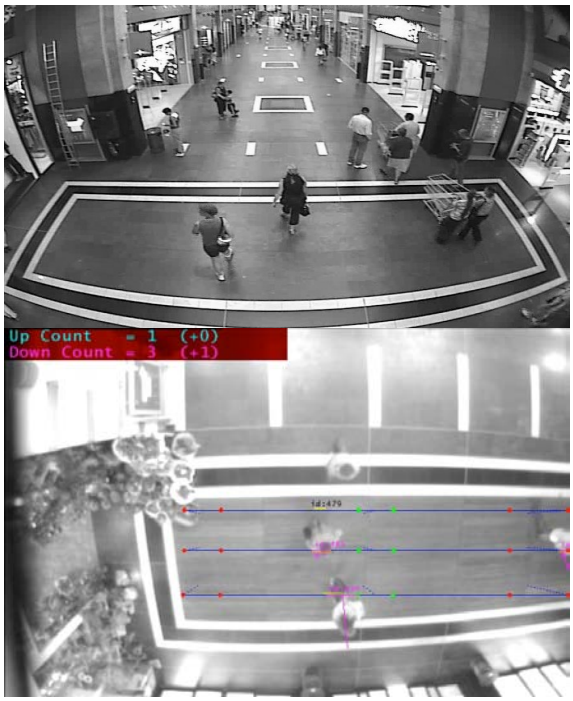


Figure 6. View of a scene of a shopping center. On the top, one can see the configuration of the scene. On the bottom, the camera view is shown.

Other input parameters of the methods are the expected mean size of people along detection lines, a sensibility threshold, as well as other timing and luminance thresholds.

4.1. Detection lines

Detection of people is performed individually on the multiple detection lines. A number of steps is performed in this order:

1) compute background estimation
2) compute foreground extraction
3) compute automatic thresholding using Otsu method [14] + mathematical morphology operators
4) perform shadows removal
5) split/merge blobs according to size parameters to obtain candidates
6) evaluate the speed of candidates
7) perform tracking and collision detection from previous frames

This is mainly 1D processing, that is, the pixels of each detection line. 2D processing is used only for step 6.

Background estimation is performed, in step 1, using two classes of algorithms, either Gaussian mixture modeling [15] or simple time recursive filtering of the form of Equation (4) where *B(i)* and *I(i)* are respectively background and Image at time *i* and α a parameter between 0 and 1.

$$B(i) = (1-\alpha)\cdot B(i-1) + \alpha \cdot I(i) \qquad (4)$$

Step 2 speaks by itself, and step 3 permits the separation of both classes, taking the frame difference as input. The Otsu [14] method assumes the presence of 2 classes in the histogram, which may not be the case (nobody on the detection lines), which led us to clamp the threshold to a meaningful luminance interval. If a value for this threshold is outside of that interval, we conclude that the 2-classes hypothesis is not valid and that there is nobody on the detection line. Mathematical morphology is performed to fill holes and remove unwanted zones (too small) according to the expected size of people.

Shadow removal in step 4 is achieved by taking into account these simple considerations about shadows, that are: a shadow zone is darker than the same zone lit; a shadow is a constant offset, for neighbor pixels, in the luminance domain; a shadow generally affects zones of a certain size (bigger than one pixel); a shadow is darker, but generally not black. The shadow

removal technique works by comparing pixel luminance of the current frame to the background frame. If the value is lower, then we evaluate the standard deviation of the luminance value along a pixel window, and compare it to a threshold that is provided as a parameter. Mathematical morphology dilatation is performed to fill holes and preserve shadow boundaries, where the standard deviation is not zero. Finally, a threshold on the luminance difference is used. Shadows are generally correctly evaluated, except for the case of people having a texture similar to the background but constantly darker (which is quite rare). This is illustrated on figure 7, yellow lines showing the parts of the image along the detection lines that are evaluated as shadows.

In step 5, prior knowledge of expected people size is used to split the blobs along the detection lines. We do not consider occlusion at this time, although a more complete method is currently being developed, accounting for camera perspective and occlusions.

Next, in step 6, the system requires the walking direction to increment the corresponding counter, positive or negative. The speed of the blobs, that we name candidates, is computed using the block-matching algorithm on 8x8 blocks on a subsampled image, and only in the neighborhood of the detection lines. The reason for subsampling is to capture details that are specific to the walking persons into the 8x8 window; the factor thus depends on people size in the original image. Mean candidate speed is evaluated by taking the mean of the speeds provided by the block-matching algorithm in the blob region.

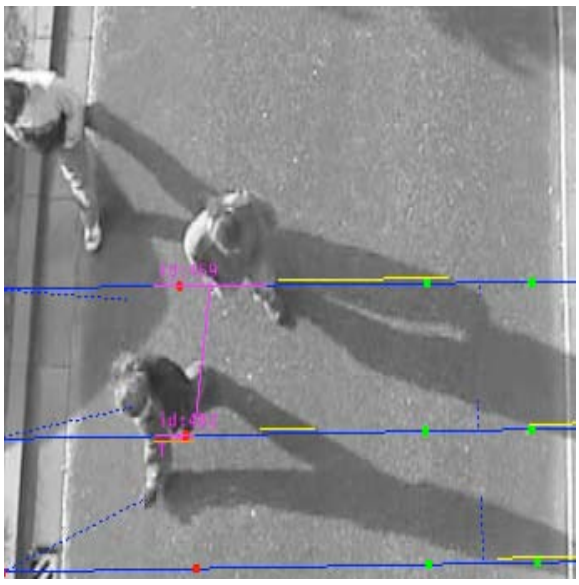


Figure 7. Shadow detection/removal.

Finally, tracking is performed in step 7, where candidates from successive frames are matched against each other. This permits to differentiate between distinct people. This is accomplished by comparing the candidates of the current frame to the ones of previous frames, in terms of spatial distance. New candidates replace old ones if the corresponding spatial region stays inactive for a given duration (we generally use value 0.3 s.).

4.2. Combining multiple detection lines

Using a single counting line is generally too sensitive, and over counts people. This leads us to fuse information from multiple counting lines. The scheme for combining that information is to consider that a person going in one direction should cross the lines in a specific order, compatible with the estimated speed.

N lines are used, and that correspondence (speed / cross time) is searched in their sets of candidates. Additional criteria, such as compatibility between speed and covered distance across lines, could also be used.

In practice, using three lines leads to more robust results than when using a single one. Using more than three lines does not improve the results quantitatively, although the processing time is directly proportional to the number of lines.

4.3. Visual setup and debugging

To help setup the system, a number of visual feedback mechanisms were implemented. This allows to graphically assessing the quality of candidate detection, speed estimate and various other subsystems. This is illustrated in figure 3.

5. Results and performance

We report hereunder the results. We have decided to test it on 3 real sequences taken from a shopping center (figure 6) with a cumulated duration of 1 hour of video at different times in the day. All sequences are MPEG4 8 bits grey levels 25 fps and were taken from fixed PAL (768 x 576 pixels) cameras. We setup the number of detection lines to three.

5.1. Results and discussion

We have use the Boolean contingency table as described in [16]: N_{tp} is the number of observations confirmed by the ground truth. N_{fp} is the number of observations not matched in the ground truth. N_{fn} is the

number of observations erroneously accepted as belonging to the ground truth. Note that a prior step separates each type of events (e.g. "left luggage" and "people fighting") to measure the deviation of each type. Here, there are two types of events: "N1" and "N2" which are triggered when a person crosses over the detection lines respectively in each direction. Table 3, 4 and 5 respectively give the results for the three sequences.

Table 2. Boolean contingency table.

	Ground truth	
System Observation	Positive	Negative
Positive	N_{tp} (true positives)	N_{fp} (false positives)
Negative	N_{fn} (false negatives)	N/A

Table 3. Result for Sequence 1.

Sequence 1: starting at 12:00:00, duration 15 min							
N1	**GT**	**RS**	**Error**	**N2**	**GT**	**RS**	**Error**
	71	74	+04%		55	66	+20%
N_{tp}		64		N_{tp}		53	
N_{fn}		7		N_{fn}		3	
N_{fp}		10		N_{fp}		12	
DR		90%		**DR**		96%	

Table 4. Result for Sequence 2.

Sequence 2: starting at 16:00:00, duration 15 min							
N1	**GT**	**RS**	**Error**	**N2**	**GT**	**RS**	**Error**
	72	76	+05%		81	99	+22%
N_{tp}		64		N_{tp}		75	
N_{fn}		8		N_{fn}		7	
N_{fp}		12		N_{fp}		23	
DR		89%		**DR**		93%	

Table 5. Result for Sequence 3.

Sequence 3: starting at 18:00:00, duration 15 min							
N1	**GT**	**RS**	**Error**	**N2**	**GT**	**RS**	**Error**
	216	225	+04%		204	257	+26%
N_{tp}		192		N_{tp}		196	
N_{fn}		24		N_{fn}		8	
N_{fp}		33		N_{fp}		61	
DR		88%		**DR**		96%	

From an application point of view, the interesting feature is the error of the counting, i.e. the difference between the number of people counted by the system and during the ground truth building. For direction N1, it is around 5%, but for direction N2, it is around 20% ! When looking deeper at the results obtained by the new evaluation method, one can explain the results because for N2, there are better detection rates than for N1, but the number of false alarms is double. We found that the false alarm rate is bigger in one direction, because people that leave the commercial center usually stop around the counting zone.

However, the implemented method of displaying the matching of events between ground truth and systems results (figure 5), allows us to display non detection and false detection one-by-one with the video shot. Thus it turned out that the ground truth sometimes contains errors.

6. Conclusion and future work

In this paper, we have proposed a new robust metric for global performance evaluation of events detection. It is interesting to find out precisely which events are "false". We described a counting method and evaluated it according to the new framework. Future work will extend the metric to the handling of substitution of events or aggregation of events (e.g. sometimes there are two detections for a single event). Future work will also take into account the implementation practical standardization of the metric (the definition of the CEP/XML format, etc.).

Acknowledgements: This work is granted by the Walloon Region under the Itea Eureka Serket project. We thank the reviewers for the helpful comments.

7. References

[1] A.Cavallaro, D. Douxchamps, T. Ebrahimi and B. Macq, "Segmenting moving objects : the MODEST video object kernel", Workshop on Image Analysis for Multimedia Interactive Services, Tampere, Finland, May 16-17, 2001.

[2] F. Cupillard, F. Brémond and M. Thonnat, "Tracking groups of people for video surveillance", Proceedings of the 2nd European Workshop on Advanced Video-Based Surveillance Systems, Kingston University, London, pp 88-100 , Sep 2001.

[3] D. Makris, T.J. Ellis and J. Black, "Learning scene semantics", ECOVISION 2004, Early Cognitive Vision Workshop, Isle of Skye, Scotland, UK, May 2004.

[4] T.J. Ellis, D. Makris and J. Black, "Learning a Multi-camera Topology", Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS), ICCV 2002, pp. 165—171, Nice, France, 2003.

[5] D. Greenhill, J. Renno, J. Orwell and G.A. Jones, "Learning the semantic landscape: embedding scene knowledge in object tracking", Real-Time Imaging, Special Issue on Video Object Processing for Surveillance Applications, pp 186-203, Jan. 2004.

[6] X. Desurmont, R. Wijnhoven, E. Jaspert, O. Caignard, M. Barais, W. Favoreel and J.F. Delaigle, "Performance evaluation of real-time video content analysis systems in the CANDELA project", conference on Real-Time Imaging IX, part of the IS&T/SPIE Symposium on Electronic Imaging 2005, 16-20 January 2005 in San Jose, CA USA.

[7] "A Seamless Modular Approach for Real-Time Video Analysis for Surveillance", X. Desurmont, A. Bastide, J-F. Delaigle, 5th International Workshop on Image Analysis for Multimedia Interactive Services, April 21-23, 2004, Instituto Superior Tecnico, Lisboa, Portugal.

[8] P. Bruneaut, A. Cavallaro, T. Kelliher, Lucio Marcenaro, F. Porikli, Sergio Velastin, Francesco Ziliani, , "Performance Evaluation Of Event Detection Solutions: the CREDS experience", CREDS - Special session, AVSS, pp. 201-206, September 2005.

[9] D. Lopresti , "Performance Evaluation for Text Processing of Noisy Inputs", Symposium on Applied Computing, pp 759 - 763, March 13-17, 2005, Santa Fe, New Mexico, USA.

[10] L. M. Brown, A.W. Senior, Y. Tian, J. Connell, A. Hampapur, C. Shu, H. Merkl, M. Lu, "Performance Evaluation of Surveillance Systems Under Varying Conditions", Proceedings IEEE International Workshop on PETS, pp 1-8, Breckenridge, CO USA, January 7 2005.

[11] C.J. Needham and D. Boyle, “Performance Evaluation Metrics and Statistics for Positional Tracker Evaluation”, Proc. of the Computer Vision Systems: Third International Conference, ICVS 2003, vol. 2626, pp 278-289, Graz, Austria, April 2003.

[12] X. Desurmont, J-F. Delaigle, A. Bastide, B. Macq, "A generic flexible and robust approach for intelligent real-time video-surveillance systems", Proceedings of Real-Time Imaging VIII, IS&T/SPIE Symposium of Electronic imaging.

[13] Proceedings of the Third IEEE International worhshop on Performance Evaluation of Tracking and Surveillance (PETS’2002), June 1, 200, Copenhagen, denmark.

[14] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms”, IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, 1979.

[15] C. Stauffer and W.E.L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99), vol. 2, pp. 246-252, June 1999.

[16] Tim Ellis, "Performance Metrics and Methods for Tracking in Surveillance", Proceeding 3rd IEEE Int. Workshop on PETS, pp 26-32, Copenhagen, June 1 2002.

Designing Evaluation Methodologies: The Case of Motion Detection

N. Lazarevic-McManus, J. Renno, D. Makris and G.A. Jones
Digital Imaging Research Centre, Kingston University,
Penrhyn Road, Kingston upon Thames, Surrey, UK KT1 2EE
www.kingston.ac.uk/dirc

Abstract

Motion detection is a fundamental processing step in the majority of visual surveillance algorithms. While an increasing number of authors are beginning to perform quantitative comparison of their algorithms, most do not address the complexity and range of the issues which underpin the design of good evaluation methodology. In this paper we explore the problems associated with optimising the operating point of detection algorithms and objective performance evaluation. A motivated and comprehensive comparative evaluation methodology is described and used to compare two motion detection algorithms reported in the literature.

1. Introduction

Motion detection is a fundamental processing step in the majority of visual surveillance algorithms. Motion detection algorithms aim to detect moving objects whilst suppressing false positives caused by lighting changes, moving background, shadows and *ghosts*. Sudden lighting conditions are particularly problematic for motion detection algorithms. Compounded by the compensating response taken by most cameras, the result is a signi cant change in both intensity and colour.

While an increasing number of authors are beginning to perform quantitative comparison of their algorithms, most do not address the complexity and range of the issues which underpin a good evaluation methodology. Such issues include the distinction and relative merits of pixel-based versus object-based metrics; the motivation of appropriate metrics; the impact of de ning the end application; making explicit evaluation parameters and selecting appropriate values; and the role of ROC optimisation of algorithms. In this paper we review the performance evaluation literature, discuss some of the more complex issues, and propose a motivated and comprehensive comparative evaluation methodology based on ROC optimisation and a proposal for standardised end-user applications as context.

2. Previous Work

A number of techniques have been proposed dedicated to performance analysis of visual surveillance algorithms. Many of them deal with evaluation of detection of moving objects [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], whereas others address evaluation of the tracking of detected objects throughout the video sequence or both[13, 14, 15, 16, 17, 18]. Since successful tracking relies heavily on accurate object detection, the evaluation of object detection algorithms within a surveillance systems plays an important part in overall performance analysis of the whole system.

Ground Truth

Evaluation based on GT offers a framework for objective comparison of performance of alternate surveillance algorithms. Such evaluation techniques compare the output of the algorithm with the GT obtained manually by drawing bounding boxes around objects, or marking-up the pixel boundary of objects, or labelling objects of interest in the original video sequence. Manual generation of GT is an extraordinarily time-consuming and tedious task and, thus, inevitably error prone even for motivated researchers. (See List *et al*[19] for an interesting study on inter-observer variability in this context.) Black *et al* recently proposed the use of a semi-synthetic GT where previously segmented people or vehicles are inserted into real video sequences[13].

Interpretation of evaluation results is obviously based on the type of GT used for comparison. However, established standards for GT are only just emerging. There are several ambiguities involved in the process of GT generation. For example, whether to account only for individual objects or also for groups of objects, or whether to look at the bounding boxes or exact shapes of objects. Several GT generation tool are available: ViPER[5], ODViS[16], CAVIAR[20]. Standardisation of datasets has been championed by PETS. Nationally funded initiatives currently preparing datasets include the French *ETISEO* project[1] and the UK Home Office *iLIDS* project[2].

[1] www.silogic.fr/etiseo/
[2] http://scienceandresearch.homeof ce.go v.uk/hosdb/news-

Common Performance Metrics

Performance evaluation algorithms based on comparison with ground truth can be further classi ed according to the type of metrics they propose. Typically, ground-truth based metrics are computed from the *true positives* (TP), *false positives* (FP), *false negatives* (FN), and *true negatives* (TN), as represented in the *contingency table* below.

Output Class	True Class	
	Foreground	Background
Fore	True Positives (TP)	False Positives (FP)
Back	False Negatives (FN)	True Negatives (TN)

Table 1: Contingency Table

For *pixel-based* metrics FP and FN refer to pixels misclassi ed as foreground (FP) or background (FN) while TP and TN account for accurately classi ed pixels[2, 3, 4, 7, 14, 17, 13]. Usually, they are calculated for each frame and an overall evaluation metric is found as their average over the entire video sequence. For object-based metrics TP refers to the number of detected objects suf ciently overlapped by GT, FP to the number of detected objects not suf ciently overlapped by the GT, and FN to the number of GT objects not suf ciently covered by any automatically detected objects[6, 15, 11]. (Note that this *degree of ovelap* is a parameter of the evaluation process.) Some authors combine both types[5]. Furthermore, a number of methods evaluate individual objects by weighting misclassi ed pixels according to their impact on the quality of segmented object[1, 8, 9, 10, 12] - in essence, pixel-based methods.

Typical metrics computed per-frame or per-sequence are the *true positive rate* (or *detection rate*) t_p, *false positive rate* f_p, *false alarm rate* f_a and *speci city* s

$$t_p = \frac{N_{TP}}{N_{TP}+N_{FN}}, \quad f_p = \frac{N_{FP}}{N_{FP}+N_{TN}}, \quad (1)$$

$$f_a = \frac{N_{FP}}{N_{TP}+N_{FP}}, \quad s_p = \frac{N_{TN}}{N_{FP}+N_{TN}} \quad (2)$$

where N_{TP}, N_{FP}, N_{TN} and N_{FN} are the number of pixels or objects identi ed as *true positives*, *false positives*, *true negatives* and *false negatives* respectively.

In some applications (*e.g.* facial identi cation) competing algorithms are presented with images which contain known *clients* and *imposters*. For object-based motion detection evaluation, there is no equivalent prior set of known *imposters* .e. false objects in the ground truth! Thus, as it is not possible to identify *true negatives*, the *false positive rate* cannot be computed.

The great majority of proposed metrics are restricted to pixel-level discrepancy between the detected foreground and the ground-truth - namely false positive and false negative pixels. These metrics are useful to assess overall segmentation quality on a frame-by-frame basis but fail to provide an evaluation of individual object segmentation. Often these measures are normalised by image size or the amount of detected change in the mask[10], or object *relevance*[1]. However, the more principled approach is based on Receiver Operating Curves (ROCs).

Evolved to characterise the behaviour of binary classi ers, ROC curves plot *true positive rate* against *false positive rate* to facilitate the selection of optimal classi cation parameters and compare alternative classi cation techniques[3, 6]. An ROC graph is an alternative presentation to plotting metrics for each frame in the sequence which is often dif cult to assess by a reader[9].

Other Performance Metrics

All pixel-based methods which evaluate individual object segmentation rely on existence of shape-based ground-truth mask generated by costly process if they are to avoid errors. In addition to the advantage of avoiding hand labelling individual foreground pixels in every frame, object-based methods only require ground-truth in the form of bounding-boxes[6, 15, 11]. The object-level evaluation proposed by Hall et al[11] plots detection rates and false alarm rates using various values of overlap threshold to determine association with the GT. As they do not have true-negatives, false alarm rate is computed as alternative to false positive rate and the area under the curve is used as a measure of performance. Other object-based metrics proposed are based on the similarity of detected and ground-truth objects *i.e.* relative position[16, 11] or shape, statistical similarity and size[1, 11].

A major problem in motion detection is the fragmentation and merging of foreground objects. While these will impact on pixel-based metrics, a number of explicit metrics have been proposed[6, 5]. Typically these measure the average number of detected regions overlapping each ground-truth object and average number of ground-truth objects associated with multiple detected regions.

Metrics may also be designed to take account of human perception of error where false positives and false negatives hold different levels of signi cance by introducing weighting functions for misclassi ed pixels on an object-by-object basis[8, 10]. Villegas and Marichal[8] increase the in uence of misclassi ed pixels further from the boundary of ground-truth objects. Cavallaro *et al*[10] account for temporal effects of *surprise* and *fatigue* where sudden changes in quality of segmentation ampli es error perception.

events/270405

Figure 1: Evaluation Dataset

3. Evaluation Methodology

We develop an evaluation methodology in which we compare the well-known Stauffer and Grimson algorithm [21] with a recently published algorithm [22]. This latter technique focuses on handling rapid lighting variations based on the observation that these global changes give rise to correlated changes in UV. A search region for constraining these colour changes is controlled by the global *mean colour difference* of the frame.

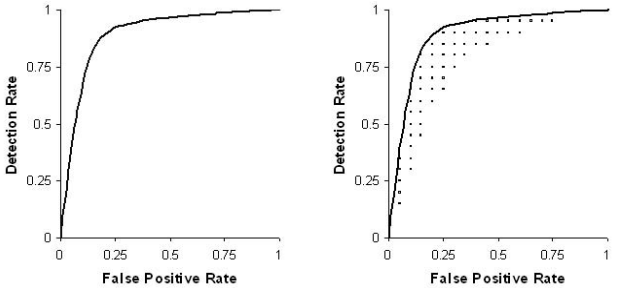
3.1. Dataset and Ground Truth

The dataset consists of 8210 frames recording activities in a car park at full frame-rate covering a period of ve and a half minutes. (Example frames shown in Figure 1.) The CCTV camera has iris auto-correction and colour switched on. The video sequence includes a total of 24 moving objects, people and vehicles appearing at close, medium and far distances from the camera. There are a variety of both gradual and sudden lighting changes present in the scene due to English weather conditions (bright sunshine interrupted by fast moving clouds, re ections from windows of vehicles and buildings). There are both static and dynamic occlusions present in the scene with moving objects crossing paths and disappearing partly or totally behind static objects. In addition, a strong wind causes swaying trees and bushes to disturb the background.

The ground truth is generated manually (by one person) using an in-house semi-automatic tool for drawing bounding boxes for every target within each frame of the video sequence. Ground truth provides the temporal ID of the object, its bounding box enclosing pixels of interest, de nes the type of the object whether person or vehicle, and de nes the degree of occlusion with other objects *i.e.* unoccluded, semi-occluded or fully-occluded.

3.2. ROC-based Analysis

Receiver Operating Curves (ROC) are a useful method of interpreting performance of a binary classi er . ROC curves graphically interpret the performance of the decision-making algorithm with regard to the decision parameter by plotting the *True Positive Rate* (t_p) against the *False Positive Rate* (f_p). Each point on the curve is generated for the range of decision parameter values - see Figure 2(a). In foreground detection, such decision parameters could be a threshold on the greylevel difference between incoming pixel and reference pixel, or a threshold on the size of any foreground object. When there is more than one classi cation parameter, a distribution of points representing all parameter value combinations is generated in the ROC space. The required ROC curve is the top-left boundary of the convex hull of this distribution as shown in Figure 2(b).



(a) Single Classi cation Parameter (b) Multiple Parameters

Figure 2: Generating ROC Curves

In general the optimal operating point is chosen in the top left quadrant of the ROC space and is de ned as the classi cation parameter value on the iso-performance line with the lowest misclassi cation cost. The gradient λ of this line is de ned as

$$\lambda = \frac{(1 - P_T)}{P_T} \frac{C_{FP}}{C_{FN}} \tag{3}$$

where P_T is the prior probability of a foreground pixel (or object), and C_{FN} and C_{FP} are the cost of classifying a moving pixel (or object) as stationary and vice versa. Misclassi cation costs depends on the intended application of motion detection (*e.g.* tracking, counting people, alarming,

detecting a speci c person, *etc*) and the ratio of foreground pixels (or objects) to background in the GT. Points on the graph lying above-left of the line have a lower misclassi - cation cost while points below-right of the line have larger costs. The misclassi cation cost C at the operating point t_p^*, f_p^* is given by

$$C(t_p^*, f_p^*) = (1 - P_T)C_{FP}f_p^* + P_T C_{FN}(1 - t_p^*) \quad (4)$$

Cost Scenarios

To explore the effect of the cost ratio, we shall introduce two different cost scenarios: the *Ticket Fraud* scenario in which, say, the cost of **detaining** an innocent member of the public C_{FP} is de ned as double the cost of failing to catch a ticket fraudster C_{FN}; and the *Evidence Gathering* scenario in which the cost of **video-ing** an innocent passerby C_{FP} is, say, 10 times smaller than the cost of failing to video a terrorist bomber C_{FN}[3]. The relative costs are arbitrary, and the relationship of these applications to motion detection is indirect. However, these different cost ratio scenarios ensure we are mindful of the ultimate application in the evaluation stage. (Obviously de ning the social costs of violations of *libertarian* and *public safety* concepts is extremely fraught!)

Evaluation Parameters

Typical performance evaluation takes the output of some visual surveillance algorithm and compares it with *ground truth* data - as illustrated in Figure 3. The performance of any surveillance algorithm will depend on the choice of the internal parameters. Optimal performance is typically determined using the ROC methodology discussed above.

Crucially, however, the result will also depend on the inevitable array of parameters required by the evaluation module *e.g.* the degree of overlap between the detected moving regions and the ground truth objects. How would you select appropriate values for these? A naive approach would be to include these evaluation parameters within the ROC methodology to select the optimal algorithm **and** evaluation parameter values. However, the result would be to evaluate each alternative surveillance algorithm with a **different** evaluation algorithm. Hardly an objective comparison! Furthermore, the ROC methodology does not apply at object-level where true negatives are not available. We explore the issue of optimisation at various stages of a typical evaluation system.

3.3. Proposed Metrics

In pixel-based performance analysis, the label of each pixel (TP, FP, FN, and TN) is de ned as follows: detected foreground pixels inside the GT bounding box (TP), detected foreground pixels outside the GT bounding box (FP), detected background pixels inside the GT bounding box (FN) and detected background pixels outside the GT bounding box (TN). As the ground-truth is only speci ed to the level of the bounding box, true-positive rates do not reach 100%.

[3]This must also capture the storage and post-event search costs

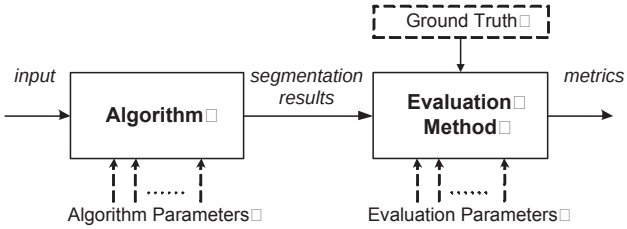


Figure 3: Typical performance evaluation system

ROC Analysis and Classi cation Costs: ROC analysis will be performed for each presented algorithm to identify optimal parameters (irrespective of published values) for the proposed *Ticket Fraud* and *Evidence Gathering* scenarios. The *Classi cation Cost* (Cost) at these operating points for each scenario will be recorded per algorithm.

Signal Rate: The signal rate s_r provides an insight into the localised quality of segmented objects, and is de ned as

$$s_r = \frac{N_{TP}}{N_{TP} + N_{FP}^*} \quad (5)$$

where N_{TP} is the number of detected pixels within the bounding boxes of the ground truth, and N_{FP}^* represents any false positives that have pixel connectivity to the ground-truth. The behaviour of this metric is illustrated in Figure 4. Rather than measure the global signal confusion, tthis version attempts to measure the degree of local confusion surrounding detected objects.

Figure 4: SNR Metric: (Left) Original with GT, (Middle) Detected Blobs (Right) Connected false positives

Speci city: During sudden lighting changes, the detection method should ideally avoid classifying large parts of the image as foreground. Many metrics do not necessarily capture this behaviour as the number of true positive pixels paradoxically increases. The *Speci city* metric (see equation 2) essentially measures the number of background pixels which remain background, and hence drops dramatically if the detection method fails to cope with light changes.

To motivate the selection of appropriate object-based metrics, we note that motion detection is usually followed by a blob tracker to establish the temporal history of any detected object. The performance of this tracker will depend crucially on (i) the proportion of true objects located, (ii) the degree of local confusion caused by falsely detected objects, and (iii) the degree of fragmentation of true objects.

Object-based metrics pose a problem which does not arise for pixel-based metrics: establishing the correspondence between GT objects and the inevitably fragmented, merged and displaced detected blobs - particularly problematic as the density of objects rises and in the presence of noise objects. Following a typical approach, we use the degree of overlap between detected objects and GT bounding boxes to establish this correspondence. In general, this can result in one-to-many and many-to-one relationships. Thus the number of true positions N_{TP} and false negatives N_{FN} can be larger than the number of ground truth objects N *i.e.* $N_{TP} + N_{FN} \geq N$.

In object-based performance analysis, the label of each object (TP, FP, and FN) is de ned as follows. A TP is a detected foreground blob which overlaps a GT bounding box, where the area of overlap is greater than a proportion Ω_b of the blob area **and** greater than a proportion Ω_g of the GT box area. A FP is a detected foreground object which does not overlap a GT bounding box. Finally, a FN is a GT bounding box not overlapped by any detected object. Note there are no de nable true negatives for objects.

Object Detection Rate: Object detection rate (or true positive rate) measures the proportion of ground-truth objects correctly detected - see equation 1.

False Alarm Rate: False alarm rate (equation 2) determines the degree to which falsely detected objects (FP) dominate true objects (TP). In fact, tracking processes can robustly ignore most false objects but are especially vulnerable to false objects located near the position of the true object. We therefore rede ne our *false alarm rate* as follows

$$f_a = \frac{N^*_{FP}}{N_{TP} + N^*_{FP}} \tag{6}$$

where N^*_{FP} is a count of falsely detected objects *near to* true objects. The degree of proximity ω is of course an evaluation parameter which requires determining.

Fragmentation Rate: Fragmented targets present a subsequent tracker with the opportunity to incorrectly update a trajectory and subsequently fail to locate any observations. Our fragmentation rate ϕ measures the number of observed blobs assigned as TP per GT object. False negative objects are not included in this metric. Thus

$$\phi = \frac{N_{TP}}{N - N_{FN}} \tag{7}$$

Scenario	Evidence Gathering	Ticket Fraud
$\tau_{\mathcal{B}}$	0.60	0.975
$\tau_{\mathcal{S}}$	0.60	0.975
Updating	Adaptive	Adaptive
$\alpha_{\mu}^{\text{fast}}$	1.0×10^{-2}	2.0×10^{-3}
$\alpha_{\Sigma}^{\text{fast}}$	1.0×10^{-4}	4.0×10^{-6}
$\alpha_{\mu}^{\text{slow}}$	5.0×10^{-3}	1.0×10^{-3}
$\alpha_{\Sigma}^{\text{slow}}$	2.5×10^{-5}	1.0×10^{-6}
Thresholding	Constant	Linear
τ_A	25	-
$\tau_{A_{\min}}$	-	50
γ_A	-	3

Table 2: Optimal Operating Points

Scenarios	Renno *et al*	Stauffer
Ticket Fraud	0.0185	0.0177
Evidence Gathering	2.483	2.320

Table 3: Classi cation Costs

where N is number of GT objects and N_{FN} the number of GT objects without supporting detected blobs.

4. Optimising the Detection Algorithm

Every algorithm has a variety of algorithm parameters whose optimal values must be identi ed. The Renno *et al* method has the parameters shown in Table 2. This section will determine the optimal parameter set for this motion detection algorithm using the ROC methodology described in Section 3.2 applied to both the *Ticket Fraud* and *Evidence Gathering* scenarios. Using the metrics de ned in Section 3.3, the optimised method will then be compared in Section 5 to the Stauffer and Grimson method[21].

Figure 5 presents the ROC space populated with t_p, f_p pairs for all evaluated combinations of algorithm parameters for the Renno *et al* algorithm. Currently the search range of each parameter is evenly sampled seven times. Each of these points is evaluated using equation 4 to identify the optimal parameter set for the two scenarios. Optimal parameter values are reported in Table 2 for each scenario.

5. Comparative Evaluation

The optimised versions of the Renno *et al* algorithm are now compared with the Stauffer and Grimson method[21]. ROC analysis is also used to optimise the parameters of the Stauffer and Grimson for the two scenarios - see Figure 6. Compared to the Renno *et al* algorithm, the Stauffer and Grimson implementation performs a little better for both scenarios. The classi cation costs of the algorithms are presented in Table 3.

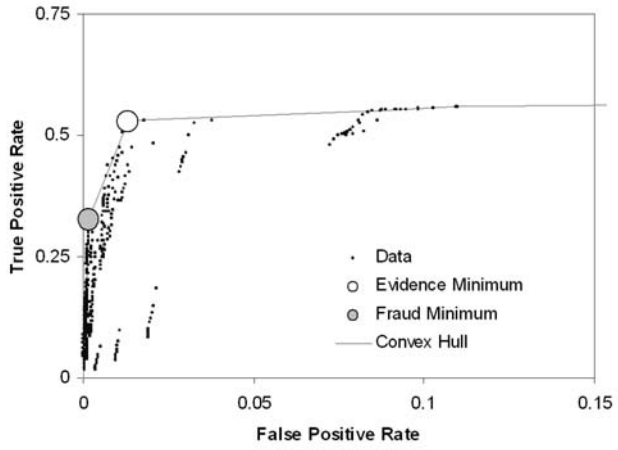


Figure 5: Determining Operating Points: Renno *et al*

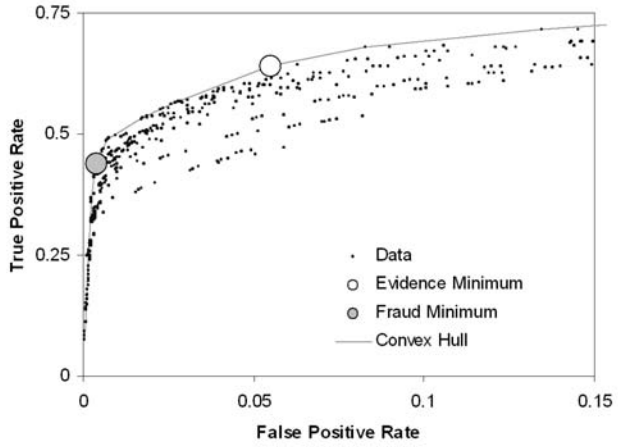


Figure 6: Determining Operating Points: Stauffer and Grimson

To gain more insight, let us turn to the metrics displayed in Tables 4 and 5. The *Evidence Gathering* version of the Renno *et al* method and the Stauffer and Grimson exhibit similar performance. About 99% of objects are located. From the localised FAR metric, each detected object has between fteen and twenty surrounding noise blobs, and is often fragmented into one to two blobs. The *Ticket Fraud* does exhibit lower detection rates ($\approx 94\%$) and an increased tendency for blob fragmentation. However, the amount of local confusion is very low - ideal for subsequent blob tracking purposes. Even more signi cant, however, is the *Speci city* metric which demonstrates the ability to successfully cope with the sudden, frequent and large lighting variations contained in the dataset.

6. Impact of Evaluation Parameter

ROC analysis offers a solution for operating point optimisation using pixel-based metrics and misclassi cation cost constraints. However, the foreground/background pixel classi cation does not provide suf cient insight into the quality of detected blobs, and their impact on the subsequent tracking and application processes. Ideally, we want to use object-based evaluation metrics to select the optimal OP. However, these metrics are likely to depend on the choice of evaluation parameters - in our case the overlap threshold parameters.

Metric	Evidence Gathering		Ticket Fraud	
	Renno	Stauffer	Renno	Stauffer
SR	0.897	0.788	0.965	0.884
Spec	0.858	0.787	0.963	0.988

Table 4: Comparison of Pixel-based Metrics

Metric	Evidence Gathering		Ticket Fraud	
	Renno	Stauffer	Renno	Stauffer
DR	0.993	0.986	0.942	0.940
FAR	0.931	0.951	0.184	0.242
FR	2.286	1.581	3.623	2.931

Table 5: Comparison of Object-based Metrics

This dependency is illustrated in Figures 7 and 8. We have two evaluation parameters as de ned in Section 3.3: the ground truth overlap threshold Ω_g and the blob overlap threshold Ω_b. In Figure 7 we explore the sensitivity of the object-based evaluation metrics (DR, FAR, FR) to the choice of blob overlap threshold while holding x ed the ground truth overlap threshold. Conversely, in Figure 8 we explore the sensitivity of the evaluation metrics to the choice of ground truth overlap threshold while holding x ed the blob overlap threshold. In each case we explore this sensitivity for the operating points of the *Ticket Fraud* and *Evidence Gathering* scenarios.

In Figure 7 for both scenarios the evaluation metrics do not depend on the overlap threshold Ω_b provided this threshold is less than 50% of the blob area. The dependency on Ω_g is more complicated. Figure 8 shows that the evaluation metrics produced for the *Evidence Gathering* do not significantly depend on Ω_g for thresholds less than 10% of the ground truth area. On the other hand, the evaluation metrics produced for the *Ticket Fraud* have a signi cant dependency on the threshold Ω_g.

7. Conclusions

The primary purpose of this paper was to expose the surprisingly complex issues that arise when creating a well-designed evaluation methodology. The speci c evaluation issues we explored were: good motivation of appropriate metrics; the distinction and relative merits of pixel-based versus object-based metrics; the need to de ne standardised application scenarios to provide context; the inevitable existance of evaluation parameters and the need to select ap-

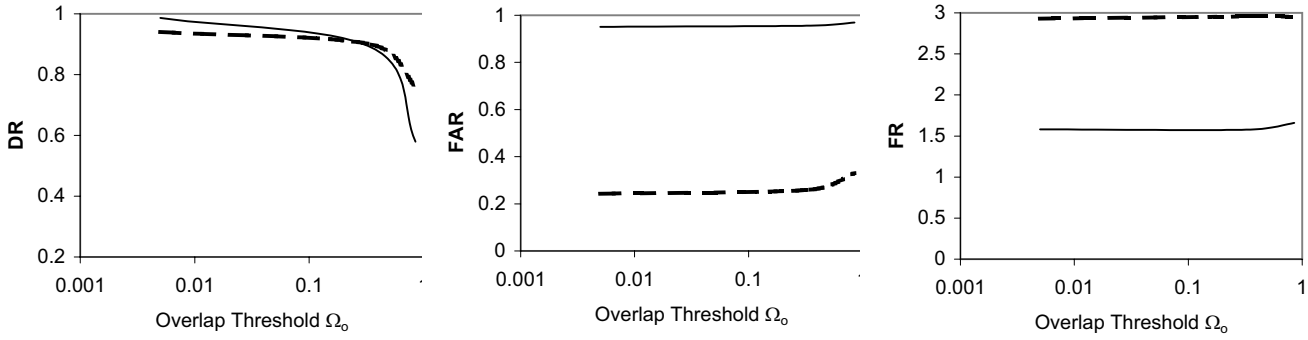


Figure 7: (b) Varying Overlap Threshold Ω_b (*Evidence Gathering* solid line, *Ticket Fraud* dotted line)

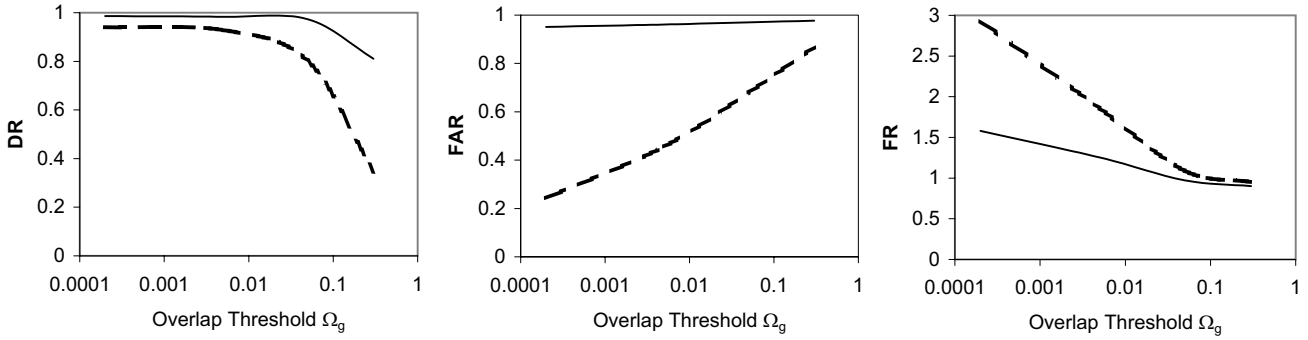


Figure 8: (a) Varying Overlap Threshold Ω_g (*Evidence Gathering* solid line, *Ticket Fraud* dotted line)

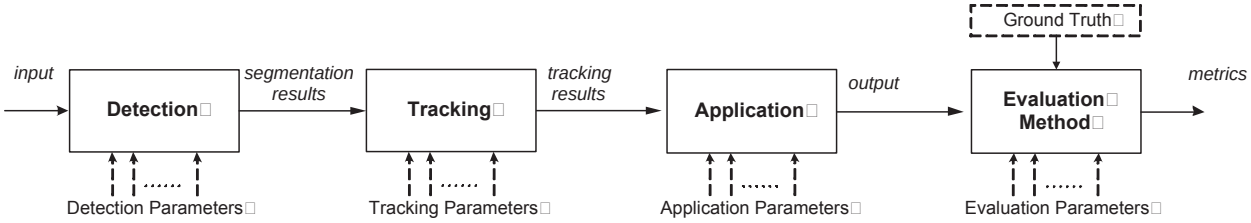


Figure 9: Evaluating Motion Detection based on application results

propriate values; and the role of ROC optimisation of algorithms. An evaluation methodology provides a mechanism for both comparison and *algorithm optimisation*, i.e. the selection of optimal algorithm parameters.

Dependency of the operating point on both algorithm parameters and evaluation parameters presents an additional dif culty to the problem of designing an ef cient evaluation methodology. A well-designed evaluation method should avoid this dependency. ROC optimisation is only applicable at pixel level. The absence of true negatives makes it impossible to use at object-level. Further investigation is needed to show whether ROC optimisation might be practical further down the evaluation pipeline - as shown in Figure 9. In the future we intend to address some of the current weaknesses: comparison with more methods reported in the literature; a bigger range of datasets; and the need to evaluate motion segmentation by its impact on the performance of subsequent processing stages.

We have illustrated the effectiveness of our object-based evaluation metrics and our approach based on scenarios by comparing two previously reported motion detection algorithms: Stauffer and Grimson[21] and Renno *et al*[22]. Though more computationally demanding, the Stauffer and Grimson algorithm has a greater capacity to represent multimodal greylevel PDFs. However, sudden lighting conditions (and camera compensation) are not modelled well by a set of modes. Rather they are a dynamic process involving correlated changes in U,V. Although unimodal, the Renno *et al* method models the possible UV variations using a measure of the global colour content of the frame. Though

explicitly optimised for speci c application scenarios, the method performs well compared to the Stauffer and Grimson algorithm.

References

[1] P. Correia and F. Pereira. "Objective Evaluation of Relative Segmentation Quality". In *IEEE International Conference on Image Processing*, pages 308–311, Vancouver, Canada, September 10-13 2000.

[2] L. Di Stefano, G. Neri, and E. Viarani. "Analysis of Pixel-Level Algorithms for Video Surveillance Applications". In *11th International Conference on Image Analysis and Processing, ICIAP2001*, pages 542–546, September 26-28 2001.

[3] X. Gao, T.E. Boult, F. Coetzee, and V. Ramesh. "Error analysis of background Adaption". In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 503–510, 2000.

[4] E.D. Gelasca, T. Ebrahimi, M.C.Q. Farias, M. Carli, and S.K. Mitra. "Towards Perceptually Driven Segmentation Evaluation Metrics". In *CVPR 2004 Workshop (Perceptual Organization in Computer Vision)*, page 52, June 2004.

[5] Vladimir Y. Mariano, Junghye Min, Jin-Hyeong Park, Rangachar Kasturi, David Mihalcik, Huiping Li, David S. Doermann, and Thomas Drayer. "Performance Evaluation of Object Detection Algorithms". In *16th International Conference on Pattern Recognition (ICPR)*, volume 2, pages 965–969, 2002.

[6] J. Nascimento and J. S. Marques. "New Performance Evaluation Metrics for Object Detection Algorithms". In *IEEE Workshop on Performance Analysis of Video Surveillance and Tracking (PETS'2004)*, May 2004.

[7] P. Villegas, X. Marichal, and A. Salcedo. "Objective Evaluation of Segmentation Masks in Video Sequences". In *Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'99)*, pages 85–88, May 1999.

[8] P. Villegas and X. Marichal. "Perceptually Weighted Evaluation Criteria for Segmentation Masks in Video Sequences". *IEEE Transactions on Image Processing*, 13(8):1092–1103, August 2004.

[9] J. Aguilera, H. Wildernauer, M. Kampel, M. Borg, D. Thirde, and J. Ferryman. Evaluation of Motion Segmentation Quality for Aircraft Activity Surveillance. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Beijing, October 15-16 2005.

[10] A. Cavallaro, E.D. Gelasce, and T. Ebrahimi. Objective Evaluation of Segmentation Quality using Spatio-Temporal Context. In *IEEE International Conference on Image Processing*, page 301304, September 2002.

[11] D. Hall, J. Nascimento, P. Ribeiro, E. Andrade, and P. Moreno. Comparison of target detection algorithms using adaptive background models. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Beijing, October 15-16 2005.

[12] C.E. Erdem and B. Sankur. "Performance Evaluation Metrics for Object-Based Video Segmentation". In *X European Signal Processing Conference (EUSIPCO)*, September 4-8 2000.

[13] J. Black, T.J. Ellis, and P. Rosin. "A Novel Method for Video Tracking Performance Evaluation". In *IEEE Workshop on Performance Analysis of Video Surveillance and Tracking (PETS'2003)*, pages 125–132, October 2003.

[14] C.E. Erdem, A.M. Tekalp, and B. Sankur. "Metrics for performance evaluation of video object segmentation and tracking without ground-truth". In *IEEE International Conference on Image Processing (ICIP)*, October 7-10 2004.

[15] B. Georis, F. Bremond, M. Thonnat, and B. Macq. "Use of an Evaluation and Diagnosis Method to Improve Tracking Performances". In *IASTED 3rd International Conference on Visualization, Imaging and Image Processing*, September 8-10th 2003.

[16] C. Jaynes, S. Webb, M. Steele, and Q. Xiong. "An Open Development Environment for Evaluation of Video Surveillance Systems". In *IEEE Workshop on Performance Analysis of Video Surveillance and Tracking (PETS'2002)*, June 2002.

[17] T. Schlogl, C. Beleznai, M. Winter, and H. Bischof. "Performance evaluation metrics for motion detection and tracking". In *17th International Conference on Pattern Recognition (ICPR)*, volume 4, pages 519–522, August 23-26 2004.

[18] H. Wu and Q. Zheng. "Self-evaluation for video tracking systems". In *Proceedings of the 24th Army Science Conference*, November 2004.

[19] Thor List, Jos Bins, Jose Vazquez, and Robert B. Fisher. "Performance Evaluating the Evaluator". In *IEEE Joint Workshop on Visual Surveillance and Performance Analysis of Video Surveillance and Tracking (VS-PETS 2005)*, 15-16th October 2005.

[20] R. Fisher. Caviar - context aware vision using image-based active recognition. In *http://homepages.inf.ed.ac.uk/rbf/CAVIAR*.

[21] C. Stauffer and W.E.L. Grimson. "Adaptive background mixture models for real-time tracking.". In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR'2000)*, pages 246–252, Fort Collins, Colorado, June 23-25 1999.

[22] J. Renno, N. Lazarevic-McManus, D. Makris, and G.A. Jones. "Evaluating Motion Detection Algorithms: Issues and Results". In *IEEE International Workshop on Visual Surveillance*, pages 97–104, Graz, Austria, May 13 2006.

Context-Controlled Adaptive Background Subtraction

Liyuan Li, Ruijiang Luo, Weimin Huang, and How-Lung Eng
Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore, 119613
{lyli,rjluo,wmhuang,hleng}@i2r.a-star.edu.sg

Abstract

Adaptive background subtraction (ABS) is the r st step for video surveillance. Existing algorithms almost all use a constant learning rate to update the background model. The performance will degrade signi cantly in some complex situations, e.g., frequent crowds or targets staying somewhere for a long time. In this paper, we propose to control the pixel-level background maintenance based on perception of contextual background objects. Two types of contextual background regions (CBRs) in public places are exploited. The type-1 CBRs are the xed facilities for the public, e.g., ATM machines, counters, benches, etc. The type-2 CBRs are the distinctive homogenous surfaces, e.g., the ground surfaces and wall surfaces. Both appearance and spatial models of each CBR are built. The likelihood of observing a CBR in a frame is then derived from the contextual appearance model. A base of appearance models for each CBR is learned incrementally for the illumination changes from day to night. The pixel-level background maintenance is controlled by contextual interpretation under a Bayes framework. Experimental results and quantitative evaluation on the sequences containing overstays of target objects, overcrowding, and frequent foreground activities are presented. Improved performance of ABS for such situations of high foreground complexities has been obtained.

1. Introduction

Adaptive background subtraction (ABS) is the rst fundamental step for video surveillance [3, 6, 13, 15]. A typical surveillance system consists of a stationary camera directed at the scene of interest. A pixel-level background model is then generated and maintained to keep the track of time-evolving background. Background maintenance is the essential part that may affect the performance of background subtraction in the time-varying situations [4, 9].

The methods of basic background subtraction employ a single reference image corresponding to the empty scene as the background model. A Kalman lter is usually used to follow the slow illumination changes [8]. It was realized that such a simple model was not suitable for surveillance in real-world situations. Adaptive background subtraction (ABS) techniques based on statistical models to characterize the background appearances at each pixel were then developed for various complex backgrounds. Wren [15] employed a single Gaussian to model the color distribution for each pixel. In [13], mixture of Gaussians is proposed to model the background of multiple states, e.g., normal and shadow appearance, and complex variations, e.g., moving bush under windy conditions. Many enhanced variants of MoG have been proposed. Some of them integrated the gradients [7], depth [5], or local features [2] into the Gaussians, and others employed the non-parametric models, e.g. kernels, to replace the Gaussians [1, 12]. In [10], a model of principal feature representation (PFR) to characterize each background pixel was proposed. Using PFR, multiple features of the background, such as color, gradient, and color co-occurrence, can be learned automatically and integrated in the classi cation of background and foreground. By employing various statistical models and multiple features for background modelling, the adaptive background subtraction (ABS) methods become more and more robust with respect to a variety of complex backgrounds.

When applied to real applications of surveillance around the clock, most of the existing methods of adaptive background subtraction employ a constant learning rate for background updating [10, 13, 15]. A few methods update the background model in a constant period of time [12, 14]. With a constant learning rate, existing methods gradually forget the old background and absorb the new background appearance into the background model. The foremost assumption behind it is that the most frequently observed features at a pixel should come from the background. This assumption is valid for situations where the foreground objects simply pass the scene in a low or moderate frequency, even though the background is highly complex, e.g., a scene of various dynamic properties. However, when some background parts are frequently occluded by foreground objects, e.g., by a person staying motionless for a while or by frequent heavy crowds which are often occur in normal public

sites, this assumption becomes violated. A few approaches tried to control the learning rate according to the results of segmentation or tracking [4, 9]. However, this control is based on positive feedback since it depends on the results of background subtraction. It may not be able to correct the errors caused by background subtraction itself.

In this paper, we propose to control the pixel-level background maintenance to extend adaptive background subtraction methods for the situations of foreground complexities, e.g., overstays of target objects, overcrowding, frequent foreground activities, etc. It is observed that in images captured by a surveillance camera in a public place, part of the scene are frequently covered by foreground objects during the various activities, e.g., ground and wall surfaces, counters, benches, etc. Such regions have distinctive and constant global features and are x ed in the scene. Human beings can easily identify exposed contextual background objects. Motivated by this observation, a novel method to control the background maintenance based on contextual interpretation is proposed. Contextual descriptors to characterize the global features of contextual background objects are developed. Likelihoods of observing contextual background objects are then derived. A base of models are learned incrementally for each contextual background region to deal with appearance changes from day to night. Based on contextual interpretation, three learning rates can be applied at each pixel for different situations. Experimental results show that the context-controlled background maintenance can signi cantly improve the performance of adaptive background subtraction in the situations of complex foregrounds at busy public places.

The rest of the paper is organized as follows. Section 2 describes the modelling and likelihood evaluation of contextual background regions. Section 3 presents a strategy to control the pixel-level background maintenance based on contextual interpretation. Experimental results are presented in Section 4 and conclusions are given in Section 5.

2. Contextual Background Representation

In an open public place, the image of the empty scene usually contains a few large homogeneous regions, like ground planes and wall surfaces, and some facilities for the public, e.g., the counters and benches. The objects of interest, e.g., human objects, luggage, and vehicles, usually appear and stay in the regions in the image. These distinctive background regions, or *contextual background regions* (CBRs), can be classi ed into two categories:

- Type-1 CBR: a facility for the public in the scene;
- Type-2 CBR: a large homogenous region.

Contextual descriptors are developed to characterize the distinctive appearances of CBRs and evaluate the likelihoods of observing them.

2.1. Contextual Descriptors and Likelihoods

2.1.1 Contextual Descriptors

Different contextual background regions may have different appearance features. Some manifest signi cant structural features, while others may have homogeneous color distributions. We employ *Orientation Histogram Representation* (OHR) to describe the structural features of a region and *Principal Color Representation* (PCR) to describe the distribution of dominant colors.

Let R_b^i be the i-th CBR in the empty scene $I(\mathbf{x})$, $G(\mathbf{x})$ and $O(\mathbf{x})$ be the gradient and orientation images of $I(\mathbf{x})$, respectively. Suppose the orientation values are quantized into 12 bins each covering 30°. The orientation histogram for R_b^i is de ned as

$$H_b^i(k) = \sum_{\mathbf{x} \in R_b^i} \mu_{T_G}(G(\mathbf{x}))\delta_k(O(\mathbf{x})) \tag{1}$$

where $\mu_T()$ is a binary function on the threshold T and $\delta_k()$ is a delta function de ned as

$$\delta_k(O(\mathbf{x})) = \begin{cases} 1, & \text{if } k \le O(\mathbf{x})/30^\circ < k+1 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The *Orientation Histogram Representation* (OHR) H_b^i is a simple and ef cient variant of the robust local descriptor SIFT [11] for real-time processes. It is less sensitive to illumination changes and slight shift of object position.

By scanning the region R_b^i, a table of the *Principal Color Representation* (PCR) can be obtained. The PCR for R_b^i is de ned as

$$T_b^i = \{p_i, \{E_i^k = (\mathbf{c}_i^k, p_i^k)\}_{k=1}^{N_i}\} \tag{3}$$

where p_i is the size of R_b^i, $\mathbf{c}_i^k$ is the k-th most signi cant color of R_b^i and p_i^k is its signi cance value. The signi cance value is computed by

$$p_i^k = \sum_{\mathbf{x} \in R_b^i} \delta(\mathbf{I}(\mathbf{x}), \mathbf{c}_i^k) \tag{4}$$

$\delta(\mathbf{c}_1, \mathbf{c}_2)$ is a delta function. It equals to 1 when the color distance $d(\mathbf{c}_1, \mathbf{c}_2)$ is smaller than a small threshold ε, otherwise, it is 0. The color distance used here is

$$d(\mathbf{c}_1, \mathbf{c}_2) = 1 - \frac{2 < \mathbf{c}_1, \mathbf{c}_2 >}{\|\mathbf{c}_1\|^2 + \|\mathbf{c}_2\|^2} \tag{5}$$

where $< \cdot, \cdot >$ denotes the dot product [10]. This color distance is robust to illumination variations. The principal color components E_i^k are sorted in descendent order according to their signi cance values p_i^k. The rst N_i components

which satis es $\sum_{k=0}^{N_i} p_i^k \geq 0.95 p_i$ are used as the PCR of the region R_b^i, which means the principal colors in PCR cover more than 95% colors from R_b^i. The *Principal Color Representation* (PCR) is ef cient to describe large regions of distinctive colors.

2.1.2 Likelihood of type-1 CBR

A type-1 CBR is associated with a facility which has the distinctive structure and colors in the image. Both OHR and PCR are used for it. Let R_{b1}^i be the i-th type-1 CBR in the scene. Its contextual descriptors are H_{b1}^i and T_{b1}^i. A type-1 CBR has just two states: occluded (occupied) or not. The likelihood of observing a type-1 CBR is evaluated on the whole region. Suppose the contextual descriptors of the region $R_t(\mathbf{x})$ from the corresponding position of R_{b1}^i in the current frame $I_t(\mathbf{x})$ are H_t and T_t. The likelihood of R_{b1}^i being exposed can be evaluated by matching $R_t(\mathbf{x})$ to R_{b1}^i.

Based on OHR, the matching of $R_t(\mathbf{x})$ and R_{b1}^i is de ned as

$$P_L(H_t|H_{b1}^i) = \frac{2 < H_{b1}^i, H_t >}{\|H_{b1}^i\|^2 + \|H_t\|^2} \tag{6}$$

Obviously, if R_{b1}^i and $R_t(\mathbf{x})$ are similar, $P_L(H_t|R_{b1}^i)$ is close to 1, otherwise, it is close to 0.

Let $T_{b1}^i = \{p_{b1,i}, \{E_{b1,i}^k = (\mathbf{c}_{b1,i}^k, p_{b1,i}^k)\}_{k=1}^{N_{b1}^i}\}$ be the PCR of R_{b1}^i and $T_t = \{p_t, \{E_t^n = (\mathbf{c}_t^n, p_t^n)\}_{n=1}^{N_t}\}$ the PCR $R_t(\mathbf{x})$. According to the law of the total probability, the likelihood of $R_t(\mathbf{x})$ belonging to R_{b1}^i is

$$P(T_t|T_{b1}^i) = \sum_{k=1}^{N_{b1}^i} P(T_t|E_{b1,i}^k) P(E_{b1,i}^k|T_{b1}^i) \tag{7}$$

The second term in the sum is the weight of the principal color $\mathbf{c}_{b1,i}^k$ in the PCR of R_{b1}^i, i.e., $P(E_{b1,i}^k|T_{b1}^i) = p_{b1,i}^k / p_{b1,i}$. The rst term is the likelihood based on the partition evidence of principal color $\mathbf{c}_{b1,i}^k$, i.e.,

$$P(T_t|E_{b1,i}^k) = \frac{1}{p_{b1,i}^k} \min\left\{ p_{b1,i}^k, \sum_{n=1}^{N_t} \delta(\mathbf{c}_{b1,i}^k, \mathbf{c}_t^n) p_t^n \right\} \tag{8}$$

Then there is

$$P(T_t|T_{b1}^i) = \frac{1}{p_{b1,i}} \sum_{k=1}^{N_{b1}^i} \min\left\{ p_{b1,i}^k, \sum_{n=1}^{N_t} \delta(\mathbf{c}_{b1,i}^k, \mathbf{c}_t^n) p_t^n \right\} \tag{9}$$

$P(T_{b1}^i|T_t)$ can be obtained in a similar way. Now the matching of R_{b1}^i and $R_t(\mathbf{x})$ based on PCR is de ned as

$$P_L(T_t|T_{b1}^i) = \min\{P(T_t|T_{b1}^i), P(T_{b1}^i|T_t)\} \tag{10}$$

Assuming that colors and the gradients are independent and different weights are used, the log likelihood of observing R_{b1}^i at time t is

$$L_{b1}^{i,t} = \omega_s \log P_L(H_t|H_{b1}^i) + (1-\omega_s) \log P_L(T_t|T_{b1}^i) \tag{11}$$

where $\omega_s = 0.6$ is chosen empirically.

2.1.3 Likelihood of type-2 CBR

The type-2 CBRs are large homogeneous regions. Only PCR descriptor is used for each of them. Usually only part of a type-2 CBR is occluded when a foreground object overlapping it. The likelihood of observing it is evaluated locally. Let $T_{b2}^i = \{p_{b2,i}, \{E_{b2,i}^k = (\mathbf{c}_{b2,i}^k, p_{b2,i}^k)\}_{k=1}^{N_{b2}^i}\}$ be the PCR of the i-th type-2 CBR R_{b2}^i. Suppose $R_t(\mathbf{x})$ is a small neighborhood centered at $\mathbf{x}$, e.g., a 5×5 window. The likelihood of $R_t(\mathbf{x})$ belonging to R_{b2}^i is de ned as

$$P(R_t(\mathbf{x})|R_{b2}^i) = \frac{1}{|R_t(\mathbf{x})|} \sum_{\mathbf{s} \in R_t(\mathbf{x})} \mathcal{B}(I_t(\mathbf{s})|R_{b2}^i) \tag{12}$$

where $|R_t(\mathbf{x})|$ is the size of the window and $\mathcal{B}(I_t(\mathbf{s})|R_{b2}^i)$ is a Boolean function de ned as

$$\mathcal{B}(I_t(\mathbf{s})|R_{b2}^i) = \begin{cases} 1, & \exists \mathbf{c}_{b2,i}^k,\ \delta(I_t(\mathbf{s}), \mathbf{c}_{b2,i}^k) = 1 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Straightforwardly, the log likelihood of that the pixel $\mathbf{x}$ in the current frame belongs to R_{b2}^i is

$$L_{b2}^{i,t}(\mathbf{x}) = \log P(R_t(\mathbf{x})|R_{b2}^i) \tag{14}$$

2.2. Models of a CBR

Each contextual background region (CBR) is described by two models, e.g., an active *appearance model* and a *spatial model*. The appearance model is composed of contextual descriptors and the spatial model is the position or range of the CBR.

To adapt to lighting changes from day to night, besides the active appearance model, a model base which contains up to K_b appearance models is used. The models in the base are learned incrementally and updated gradually. The active appearance model is the one from the model base which best ts the current appearance of the CBR.

Natural lighting changes slowly and smoothly. Let D be a time duration of 3 to 5 minutes. If suf cient samples of a CBR have been observed and the average likelihood values of them are low for two consecutive durations, the active appearance model will be updated. A new appearance model can be obtained from the latest duration. The new appearance model is compared with the ones in the model base. If one is suf ciently close to the new model, it will be used to

replace the active model. Otherwise, the active model will be replaced by the new model. Meanwhile, the new model is also placed into the base. If the model base is already full, the oldest one is then replaced by the new model.

The *appearance model* of a type-1 CBR consists of its OHR and PCR. For the i-th type-1 CBR R^i_{b1}, the *appearance model* is de ned as $M_a(R^i_{b1}) = (H^i_{b1}, T^i_{b1})$, and the *spatial model* is de ned as its bounding box and the center point, i.e., $M_s(R^i_{b1}) = (B^i_{b1}, \mathbf{x}^i_{b1})$. The appearance model base is $MB(R^i_{b1}) = \{M^k_a(R^i_{b1}), k \leq K_b\}$. The active appearance model $M_a(R^i_{b1})$ is from the model base $MB(R^i_{b1})$.

Let T^i_{b2} be the PCR descriptor of the i-th type-2 CBR R^i_{b2}, the *appearance model* of R^i_{b2} is then de ned as $M_a(R^i_{b2}) = (T^i_{b2})$. The *spatial model* of R^i_{b2} describes the range of the homogeneous region in the image. A binary mask $U^i_{b2}(\mathbf{x})$ is used for it, i.e., $M_s(R^i_{b2}) = (U^i_{b2}(\mathbf{x}))$. The spatial model may have to be adjusted in initialization duration when suf cient samples have been observed according to the likelihood values. The model base of R^i_{b2} is $MB(R^i_{b2}) = \{M^k_a(R^i_{b2}), k \leq K_b\}$. The active appearance model $M_a(R^i_{b2})$ is from the base $MB(R^i_{b2})$.

At each time step, if the large part of the type-2 CBR R^i_{b2} is exposed, a sample of it is obtained. If suf cient samples are observed and the average likelihoods to the active appearance model are low for two consecutive durations, the active appearance model will be updated. Here, the normalized PCRs are used to evaluate the likelihood between the active model and the new model since part of the CBR may be occluded by foreground objects in the scene at different times.

3. Context-Controlled Background Maintenance

3.1. Contextual Interpretation

Let $\{R^i_b\}^{N_b}_{i=1}$ be the CBRs of a scene. Given an incoming frame $I_t(\mathbf{x})$ and a local region $R_t(\mathbf{x})$ centered at $\mathbf{x}$ in $I_t(\mathbf{x})$, the posterior probability of $R_t(\mathbf{x})$ belonging to a CBR R^i_b can be evaluated as

$$P(R^i_b|R_t(\mathbf{x})) \propto P(R_t(\mathbf{x})|R^i_b)P(R^i_b|\mathbf{x}) \quad (15)$$

Then the log posterior probability is de ned as

$$Q^i_b(R_t(\mathbf{x})) = \log P(R_t(\mathbf{x})|R^i_b) + \log P(R^i_b|\mathbf{x}) \quad (16)$$

The position of a type-1 CBR is already determined by its spatial model. The prior $P(R^i_b|\mathbf{x})$ is 1 for the position and 0 otherwise. Then, the log posterior is equivalent to the log likelihood at the position, i.e., $Q^{i,t}_{b1} = L^i_{b1}(R_t(\mathbf{x}^{i,t}_{b1})) = L^{i,t}_{b1}$ for R^i_{b1}. To lter out the disturbance, a rate of occluded times over recent frames for each type-1 CBR is used. For R^i_{b1}, the rate is computed as

$$r^{i,t}_{b1} = \beta r^{i,t-1}_{b1} + (1-\beta)(1 - \mu_{T_{L1}}(Q^{i,t}_{b1})) \quad (17)$$

where β is a smooth factor and $\beta = 0.5$ is chosen. A high rate value indicates that R^i_{b1} has been occluded in recent frames. In (17), T_{L1} is a threshold which determines whether the CBR is exposed. It is found that the classi cation is not sensitive to the threshold T_{L1} when it is selected from [-0.3, -0.2]. Hence, $T_{L1} = -0.25$ is chosen in this paper.

From the spatial model $U^i_{b2}(\mathbf{x})$ of the i-th type-2 CBR R^i_{b2}, the prior probability of a pixel $\mathbf{x}$ belonging to the region R^i_{b2} can be de ned as

$$P(R^i_{b2}|\mathbf{x}) = \frac{1}{|R_t(\mathbf{x})|} \sum_{\mathbf{s}\in R_t(\mathbf{x})} U^i_{b2}(\mathbf{s}) \quad (18)$$

Combining (16), (14), and (18), the log posterior of that $\mathbf{x}$ is an exposed point of R^i_{b2} is

$$Q^{i,t}_{b2}(\mathbf{x}) = L^{i,t}_{b2}(\mathbf{x}) + \log P(R^i_{b2}|\mathbf{x}) \quad (19)$$

Again, a rate of occluded times over recent frames at each pixel for each type-2 CBR is used. First, to be robust to noise and effect of boundaries, an occluded pixel of a type-2 CBR is con rmed on the local neighborhood $R_t(\mathbf{x})$. Let $r_1 = P(R^i_{b2}|\mathbf{x})$ which also represents the proportion of pixels belonging to R^i_{b2} in the neighborhood region, and r_2 be the proportion of exposed pixels of R^i_{b2} in $R_t(\mathbf{x})$ according to the posterior estimates, i.e.,

$$r_2 = \frac{1}{|R_t(\mathbf{x})|} \sum_{\mathbf{s}\in R_t(\mathbf{x})} \mu_{T_{L2}}(Q^{i,t}_{b2}(\mathbf{s})) \quad (20)$$

where T_{L2} is a threshold which determines whether the pixel is of exposed part of the CBR. Again, the classi cation is not sensitive to T_{L2} when it is selected from [-0.35, -0.25] and $T_{L2} = -0.3$ is chosen in this paper. An occluded pixel of R^i_{b2} is con rmed if the majority of the pixels within its neighborhood are of R^i_{b2} and less of them are observed in the current frame. Now the rate is computed as

$$r^{i,t}_{b2}(\mathbf{x}) = \beta r^{i,t-1}_{b2}(\mathbf{x}) + (1-\beta)[\mu_{T_H}(r_1) \cdot \mu_{T_H}(1-r_2)] \quad (21)$$

where $T_H = 75\%$ is chosen semantically.

3.2. Context-Controlled Background Updating

According to the result of contextual interpretation, three learning rates can be applied at each pixel for different situations:

- Normal learning rate to exposed background pixels with small variations;

- Low learning rate to occluded background pixels;
- High learning rate to exposed background pixels with signi cant changes.

Let $C_t(\mathbf{x})$ be a control code image generated at time t according to contextual interpretation, where the value of $C_t(\mathbf{x})$ is 1, 2, or 3 for the low, normal, or high learning rates. First, for the pixels not associated with any contextual background region, $C_t(\mathbf{x}) = 2$ is set. For a pixel $\mathbf{x}$ within a CBR, If it is occluded by a foreground object (i.e., $r_{b1}^{i,t}$ or $r_{b2}^{i,t}(\mathbf{x})$ is large), $C_t(\mathbf{x}) = 1$ is set. Otherwise, if $I_t(\mathbf{x})$ is detected as a background point by background subtraction, $C_t(\mathbf{x}) = 2$ is set since the pixel of the CBR is exposed and no signi cant appearance change is found, but if $I_t(\mathbf{x})$ is detected as a foreground point by background subtraction, a high rate should be applied since it is detected as an exposed CBR point with signi cant appearance change, i.e., $C_t(\mathbf{x}) = 3$. Then, scan each foreground region generated by background subtraction. If most of the points in a region which are associated with CBRs are the occluded ones, the control codes for all pixels of the region are set as $C_t(\mathbf{x}) = 1$.

To smoothen the control code temporally at each pixel, four control code images are used. The rst two are the previous and current control code images described above, i.e., $C_{t-1}(\mathbf{x})$ and $C_t(\mathbf{x})$, and the second two images are the control codes which really applied for pixel-level background maintenance, i.e., $C^*_{t-1}(\mathbf{x})$ and $C^*_t(\mathbf{x})$. The applied control code to current frame at pixel $\mathbf{x}$ is determined by the votes from three other control codes $C_{t-1}(\mathbf{x})$, $C_t(\mathbf{x})$, and $C^*_{t-1}(\mathbf{x})$. If at least two of the three codes are the same, the control code of high votes is selected. If the three codes are different from each other, the normal learning rate is used, i.e., $C^*_t(\mathbf{x}) = 2$.

4. Experimental Results

To evaluate the effect of context-controlled background maintenance on adaptive background subtraction, two existing methods of ABS were implemented. They are the methods based on Mixture of Gaussian (MoG) [13] and Principal Feature Representation (PFR) [10]. Hence, four methods, MoG, Context-Controlled MoG (CC_MoG), PFR, and Context-Controlled PFR (CC_PFR) were compared. In the test, the normal learning rate is the constant learning rate used for the existing methods of ABS. The high learning rate is the double of the normal learning rate and the low learning rate is zero.

The experiments have been conducted on a variety of sequences containing both various background changes and complex activities of foreground. The background changes are caused by imaging noise, shadows, and illumination variations due to weather conditions and auto-gain adjustments, etc. The complex foreground activities include the overstays of targets, queues, large crowds, and very frequent activities of foreground objects. Some examples and corresponding quantitative evaluations on ve sequences are presented in this section. Five snapshots, each being an empty scene from one of the sequences with marked CBRs on it, are shown in Fig. 1.

The images in the gures in the rest of this section are laid in the same format. In each gure, the rst column are the sample frames from the sequence, the second column are the corresponding ground truths of the foreground, and the rest images from left to right are: the segmented results by MoG, CC_MoG (Context-Controlled MoG), and the corresponding control image, the segmented results of PFR and CC_PFR (Context-Controlled PFR), and the corresponding control image. In the control images, the black regions do not belong to any CBR, the gray regions are the exposed parts of CBRs with no signi cant appearance changes, the white regions are the occluded parts of CBRs, and the light gray regions are the exposed parts of CBRs with signi cant appearance changes. That means, for black and gray pixels, normal learning rate is applied, for white pixels, low learning rate is used, and for light gray pixels, high learning rate is used.

The rst example shows the comparison on the scenarios of overstays of targets. The example shows the test on two sequences from the PETS 2001 dataset which are captured by two cameras monitoring the same scene from different view points. In this scene, the vehicles stopping on the road for a long time might be not allowed. Hence, one type-2 CBR of the road surface is exploited for each scene. The sequences were captured at 25fps and contain 2687 frames (about 107s). During the time, a white van stopped on the road two times at different places (from frame 985 to 1590 and from 1920 to 2470). The segmentation results at frames 1211 and 2111 are shown in Fig. 2. It can be seen that, without control on the learning rate, the van was absorbed into background gradually once it stopped. For quantitative evaluation, 17 frames, chosen every 100 frames from Frame 1011, are used. The comparison with the ground truth is listed in the rst two lines in Table 1.

The 3rd sequence shows a scene of ATM machines in a public place, in which persons often stay in front of the ATM machines for minutes. Hence, two type-1 CBRs for the two ATMs and one type-2 CBR of the ground surface are employed. This sequence lasts for about 15 minutes and digitized at 8fps. During the time, there were dozen of persons using the two ATM machines. In some cases, several persons queued up for the ATM machines. With a constant learning rate, almost all of them were absorbed into the background soon after they standing besides the ATM machines. However, with the control of learning rate based

Figure 1. The scenes with marked CBRs from the ve test sequences are displayed from left to right. In each scene, the type-2 CBRs are marked by white or black boundaries, and the type-1 CBRs are marked by red boxes.

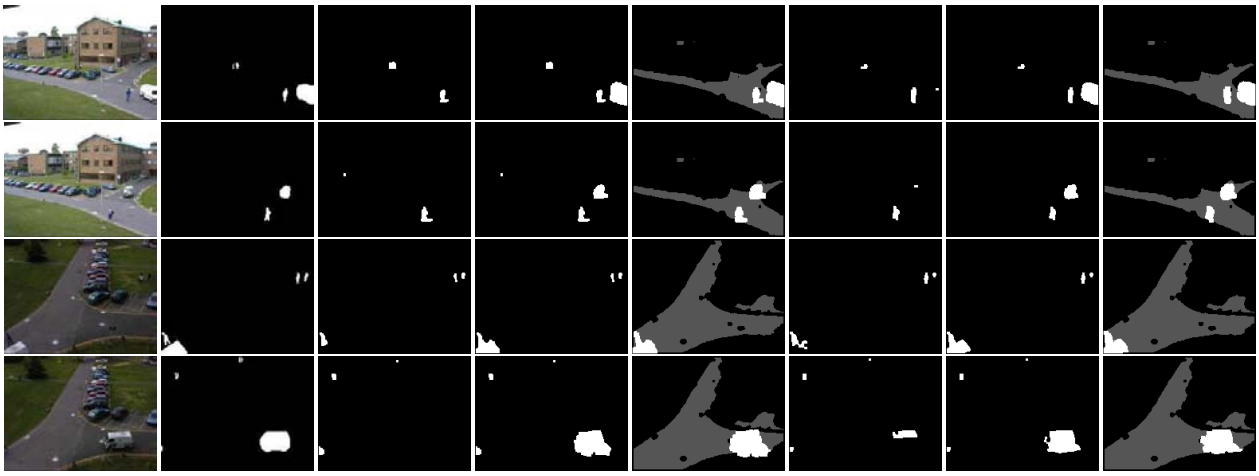


Figure 2. Example of detecting vehicles stopping on the road on the sequences from the PETS 2001 dataset. Upper two rows show the results on the frame 1211 and 2111 from camera1, and the lower two rows show the results on the same frames from camera2.

on contextual interpretation, none of them were lost in the sequence while the pixel-level background model can still adapt to environment variations of the outdoor scene. Three samples of the segmentation are shown in Fig. 3. It can be seen that when there are persons using the machines or queuing up for them, most parts of the foreground regions are lost. The quantitative evaluation on 18 sample frames chosen every 200 frames from Frame 2910 was made and the result is listed in the third line in Table 1.

The 4th sequence captured the scene of a platform for a lift. This sequence contains the scenarios of frequent crowds waiting on the platform for the lift. The concrete ground surface is marked as a type-2 CBR. During the peak hour, people gathering on the platform for the lift. Sometimes a person has to wait for more than one time before taking it. This sequence was captured during the peak hour for about 22 minutes (8fps). Three samples containing people gathering on the platform for the lift are shown in Fig. 4. Without control of learning rate, the ABS methods failed to detect foreground objects when persons frequently gathering on the platform, but with the control of learning rate based on contextual interpretation, the ABS methods succeed to maintain a satis ed pixel-level background model which keeps sensitive to foreground objects and adaptive to background changes throughout the sequence. The result of quantitative evaluation on 19 frames sampled from the sequence every 200 frames started from Frame 1110 is listed in the fourth line in Table 1.

The last example presents the test on the scenario of overcrowding in public places. The sequence was captured at 12fps in a foyer of a public building which is often overcrowded during public activities. Two type-2 CBRs for the ground surface, the marble and carpets regions, are exploited. The sequence contains a peak of overcrowded human o w which lasts for about 2 minutes. Three sample frames captured at about 73s, 82s, and 90s after the start

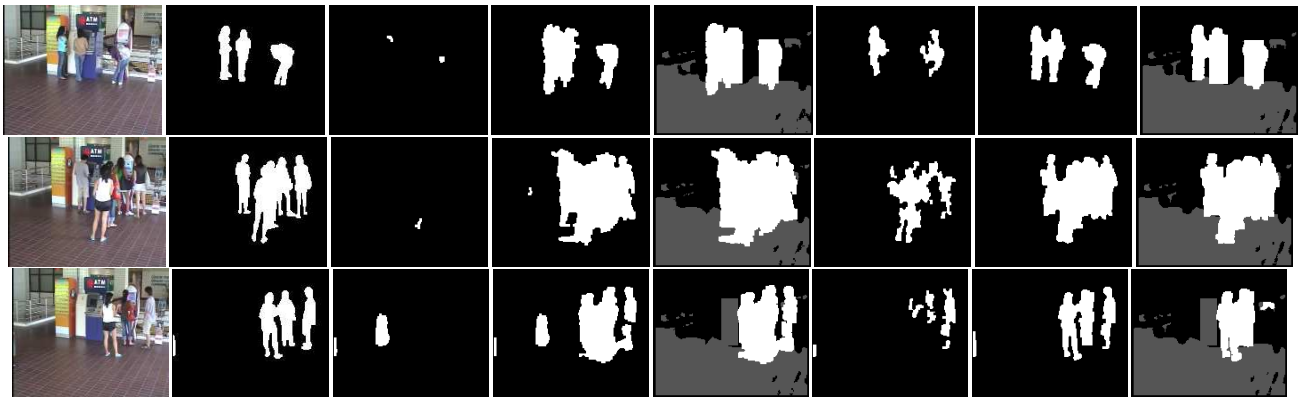


Figure 3. Three examples of persons using or queuing up for ATM machines.

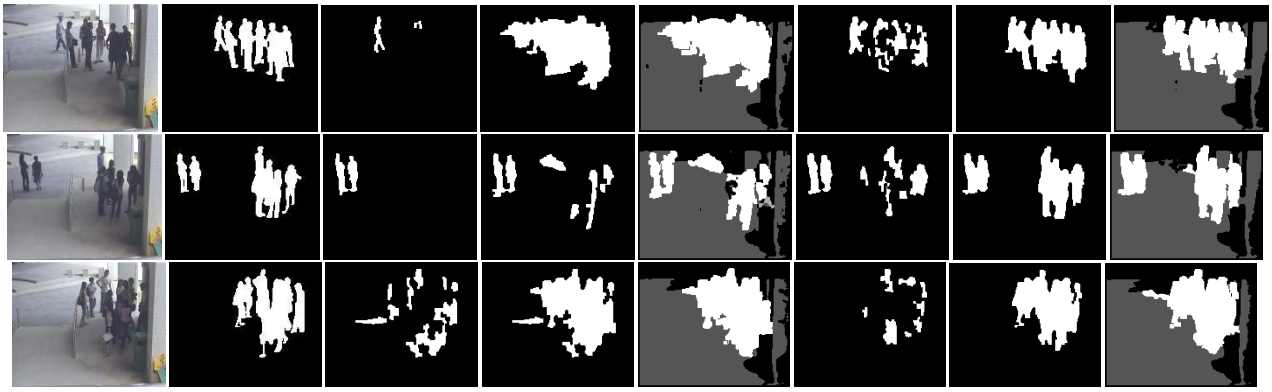


Figure 4. Three examples of the test on the sequence which contains frequent crowds in the scene.

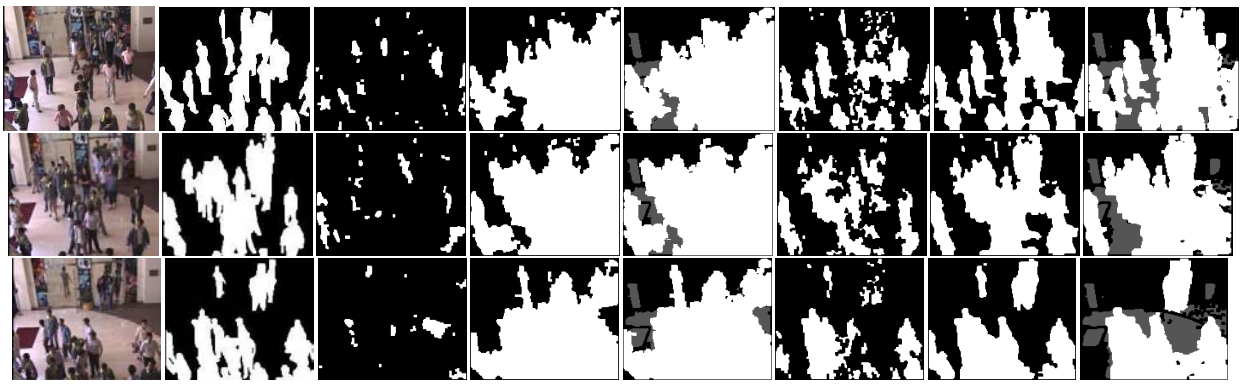


Figure 5. Three examples of foreground segmentation on the scenarios of overcrowding.

Table 1. Quantitative evaluation results.

	MoG	CC_MoG	PFR	CC_PFR
PETS_D1_C1	0.2483	0.6966	0.3868	0.7206
PETS_D1_C2	0.2388	0.6776	0.4222	0.7192
NUS_ATM	0.2167	0.4213	0.5020	0.7739
NUS_Lift	0.2358	0.5841	0.4509	0.7669
I2R_Forum	0.2375	0.3821	0.6329	0.7432
Average	0.2354	0.5523	0.4790	0.7448

of the human o w and the corresponding segmentation results are shown in Fig. 5. Compared with conventional ABS methods, background maintenance based on contextual interpretation can avoid contaminating the pixel-level background model when the scene is overcrowded by whether motionless human groups or moving human o ws. Quantitative evaluation on 19 frames sampled every 100 frames over the duration of the peak of human o w is listed in the fth line in Table 1.

The metric of quantitative evaluation is de ned as the ratio between the intersection and union of the ground truth and the segmented regions [10]. According to [10], the performance is good if the metric value is larger than 0.5 and nearly perfect if the metric value is larger than 0.8. From Table 1, it can be seen that, by using context-controlled background maintenance, the performance of adaptive background subtraction on situations of complex foreground activities can be improved signi cantly . The demo videos are available at http://perception.i2r.a-star.edu.sg/PETS2006/Contextual.htm.

5. Conclusions

The foremost assumption behind the background maintenance in existing methods of adaptive background subtraction is that the most frequently observed features at each pixel should belong to the background. When the foreground activities become complex, e.g., overstays of target objects, overcrowding, and frequent foreground activities in a busy scene, this assumption is violated since some parts of the scene are occluded by foreground objects most of the time. In this paper, we propose a method to use contextual background information to control the pixel-level background maintenance for adaptive background subtraction. The global features of distinctive contextual background objects provide a strong cue for background perception once the background part is exposed. With contextual interpretation, we may let the pixel-level background model be updated just on correct samples. Then the pixel-level background model can be used to achieve a precise segmentation of foreground objects even some background parts are frequently occluded by foreground objects. Experimental results show that it is a promising method to improve the performance of adaptive background subtraction for the situations of high foreground complexities.

References

[1] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *Proc. European Conf. Computer Vision*, 2:751–767, 2000.

[2] H. Eng, J. Wang, A. Kam, and W. Yau. Novel region-based modeling for human detection within high dynamic aquatic environment. *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2:390–397, 2004.

[3] I. Haritaoglu, D. Harwood, and L. Davis. W^4: Real-time surveillance of people and their activities. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):809–830, August 2000.

[4] M. Harville. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. *Proc. European Conf. Computer Vision*, pages 543–560, 2002.

[5] M. Harville, G. Gordon, and J. Wood ll. Foreground segmentation using adaptive mixture model in color and depth. *Proc. IEEE Workshop Detection and Recognition of Events in Video*, pages 3–11, 2001.

[6] W. Hu, T. Tan, L. Wang, and S. Maybank. A survey on visual surveillance of object motion and behaviors. *IEEE Trans. Systems, Man, and Cybernetics – Part C*, 34(3):334–352, August 2004.

[7] O. Javed, K. Sha que, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. *Proc. IEEE Workshop Motion and Video Computing*, pages 22–27, 2002.

[8] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, and S. Russel. Toward robust automatic traf c scene analysis in real-time. *Proc. Int'l Conf. Pattern Recognition*, pages 126–131, August 1994.

[9] L. Li, I. Y. H. Gu, M. K. H. Leung, and Q. Tian. Adaptive background subtraction based on feedback from fuzzy classi cation. *Optical Engineering*, 43(10):2381–2394, 2004.

[10] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian. Statistical modeling of complex background for foreground object detection. *IEEE Trans. Image Processing*, 13(11):1459–1472, 2004.

[11] D. Lowe. Distinctive image features from scale-invariant key-points. *Int'l J. Computer Vision*, 60(2):91–110, 2004.

[12] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27:1778–1792, 2005.

[13] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(8):747–757, August 2000.

[14] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wall o wer: principles and practice of background maintenance. *Proc. IEEE Int'l Conf. on Computer Vision*, pages 255–261, 1999.

[15] C. Wren, A. Azarbaygaui, T. Darrell, and A. Pentland. *P nder* : Real-time tracking of the human body. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

Autonomous Learning of a Robust Background Model for Change Detection *

Helmut Grabner Peter M. Roth Michael Grabner
Horst Bischof
Graz University of Technology
Institute for Computer Graphics and Vision
Inffeldgasse 16/II, 8010 Graz, Austria

Abstract

We propose a framework for observing static scenes that can be used to detect unknown objects (i.e., left luggage or lost cargo) as well as objects that were removed or changed (i.e., theft or vandalism). The core of the method is a robust background model based on on-line AdaBoost which is able to adapt to a large variety of appearance changes (e.g., blinking lights, illumination changes). However, a natural scene contains foreground objects (e.g., persons, cars). Thus, a detector for these foreground objects is automatically trained and a tracker is initialized for two purposes: (1) to prevent that a foreground object is included into the background model and (2) to analyze the scene. For ef ciency reasons it is important that all components of the framework are using the same ef cient data structure. We demonstrate and evaluate the developed method on the PETS 2006 sequences as well as on own sequences of surveillance cameras.

1. Introduction

For most video surveillance systems a foreground/background segmentation is needed (at least in the very beginning). Usually this segmentation is obtained by rst estimating a robust background model and second by thresholding the difference image between the current frame and the background model. Such methods are referred as background subtraction methods. Let $\mathbf{B}_t$ be the current estimated background image, $\mathbf{I}_t$ the current input image and θ a threshold, then a pixel is classi ed as foreground if

$$|\mathbf{B}_t(x,y) - \mathbf{I}_t(x,y)| > \theta. \quad (1)$$

*This work has been supported by the Austrian Federal Ministry of Transport, Innovation and Technology under P-Nr. I2-2-26p VITUS2 and by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04. In addition this work has been sponsored by the MISTRAL Project which is nanced by the Austrian Research Promotion Agency (www.ffg.at) and the EU FP6-507752 NoE MUSCLE IST project.

(a) (b)

Figure 1: A background model is used for detecting foreground objects in a static scene (colored boxes in the right image). The automatic learning of allowed objects (yellow) allows the detection of unknown objects (red).

As for realistic applications different environmental conditions (e.g., changing lightening conditions or foreground models moving to background and vica versa) have to be handled. Therefore, several adaptive methods for estimating a background model $\mathbf{B}_t$ have been proposed that update the existing model with respect to the current input frame $\mathbf{I}_t$ (e.g., running average [8], temporal median lter [10], approximate median lter [11]). A more detailed discussion of these different background models can be found in [1,6,15].

But all of these methods have two drawbacks: First, as the foreground objects may have a similar color as the background, these objects can not be detected by thresholding. Second, these methods are only slowly adapting to slightly changing environmental conditions. Thus, faster changes as a ashlight signal or uttering leaves in the wind can not be modeled. To overcome this problem a multi-modal model such as Mixture of Gaussian [20], Eigenbackgrounds [13] or more ef ciently by analyzing of foreground models [21] can be applied.

For left luggage detection this simple segmentation process has to be extended. We are not only interested in a foreground/background segmentation but we are also interested in differentiating between *known objects* (e.g., persons) and *unknown objects* (e.g., left luggage). Thus, we propose a

framework that combines a background model and a known object identi er . An object detector is trained and the obtained detections are eliminated from an already computed foreground/background segmentation. As a result we obtain regions that can not be modeled and may represent unknown objects. An example is depicted in Figure 1. The small red and blue squares represent regions that can not be explained by the background model. By using a person detector (yellow bounding boxes) the regions according to the blue boxes are recognized as parts of a person. All other regions (red boxes) can not be explained and are therefore unknown objects.

The background model is learned by observing a scene from a static camera and by learning everything that is present in the scene by on-line AdaBoost [4]. Thus, all dynamic changes (even moving objects) are assumed to be normal and are therefore learned as an allowed mode. The major bene t of such a background model is its capability to adapt to any background and its ability to model even complex scenes (e.g., containing dynamic background such as blinking lights). Furthermore, it allows to adapt to continuous changes (e.g., illumination changes in outdoor scenes) while observing the scene.

The object detector is trained by Conservative Learning [17,18] and is applied for two purposes. First, the detection results are used to distinguish between relevant changes (e.g., left luggage, lost cargo, etc.) and natural occurrences (e.g., walking persons). Second, as the background model can be updated on-line it is used to de ne an update policy. Thus, image areas where a foreground object was detected can be excluded from the update process and a background model can be estimated even if foreground objects are present during the learning stage. To increase the stability of the detector a tracker is used and an object is even detected if there are larger changes in its appearance.

The proposed framework can be applied to detect (dynamic as well as non-dynamic) changes in a static scene. Thus, unknown objects (i.e., left luggage or lost cargo) as well as objects that were removed or changed (i.e., theft or vandalism) can be detected. As the known-object detector can be trained without any user interaction directly from video data and the background model is estimated automatically we have a fully automatic framework.

The outline of the paper is as follows: First, the framework is introduced and the different modules are described in Section 2. Next, experiments and results are shown in Section 3. Finally, the paper is summarized in Section 4.

2. Video Surveillance Framework

The main components of the framework (see Figure 2) are a robust block based background model and a known-object identi er (detector and tracker). First, a foreground object detector is trained for “known objects” by Conservative Learning [17, 18]. To be more robust additionally a tracker [5] is initialized when a known object is detected in the scene for the rst time. Next, the background model is estimated by observing the scene assuming that all input frames contain only (even changing) background. When a rst model was estimated new input frames are evaluated. All non-background regions are detected and veri ed by the detection results. Thus, only regions are reported that can not be explained by any of the models. Finally, in a post-processing step only detections are considered that are stable over time.

In addition, the background model is updated for every new frame. To avoid that foreground objects are included into the background model a special update policy is de ned (detection results are used to de ne areas where no updates should be performed for some time).

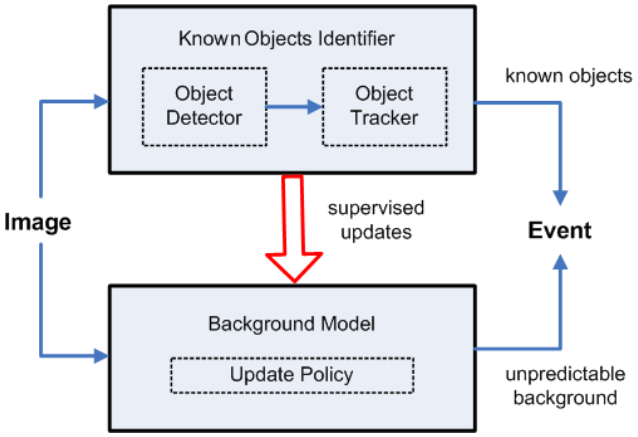


Figure 2: Overview of the proposed framework.

All modules (background model, detector and tracker) are based on the same type of classi er that is trained using the same features. In particular we apply Haar-like features [23], orientation histograms (with 16 bins) similar to [2, 9] and a simpli ed version (4-th neighborhood) of local binary patterns (LBP) [12]. Using integral data structures the features can be estimated very ef ciently [16] because this data structure has to be computed only once for all modules. To have an ef cient system we also need an estimate of the ground plane which can be automatically done by, e.g., [14].

2.1. Background Model

For estimating the change detection we apply a new *classi er -based* background model [4] that is based on the idea of a block based background model [7]. Thus, the gray-scale image is partitioned into a grid of small (overlapping) rectangular blocks (patches). For each of them a separate classi er is computed by combining the image features that

were described in the Section 2. For training the classi ers we use boosting for feature selection from this highly over complete representation (each feature prototype can appear at different positions and scales). The overall principle is depicted in Figure 3.

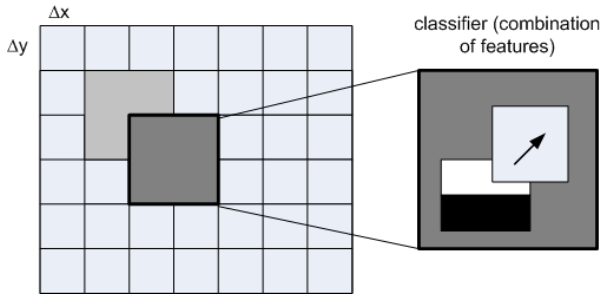


Figure 3: The background model is formed by a grid of regular aligned classi ers with an overlap of $\Delta x = \Delta y = 50\%$. Each cell is represented by a strong classi er obtained by boosting which combines several weak classi ers based on visual features. Ef c ient computation is achieved by using fast computable features via integral structures.

In general Boosting (see [3] for a good introduction) is a widely used technique in machine learning for improving the accuracy of any given learning algorithm. In fact, boosting converts a weak learning algorithm into a strong one. Therefore, for an input vector $\mathbf{x}$ a strong classi er $h^{strong}(\mathbf{x})$ is computed as linear combination of a set of N weak classi ers $h_n^{weak}(\mathbf{x})$:

$$h^{strong}(\mathbf{x}) = \text{sign}(conf(\mathbf{x})) \tag{2}$$

$$conf(\mathbf{x}) = \frac{\sum_{n=1}^{N} \alpha_n \cdot h_n^{weak}(\mathbf{x})}{\sum_{n=1}^{N} \alpha_n} \tag{3}$$

A weak classi ers is a classi er that has to perform only slightly better than random guessing, i.e., for a binary decision task, the error rate must be less than 50%. As $conf(\mathbf{x})$ is bounded by $[-1, 1]$ it can be interpreted as a con dence measure (which is related to the margin). The higher the absolute value, the more con dent is the result.

Boosting for feature selection was rst introduced by Tieu and Viola [22] and has been widely used for different applications (e.g., face detection [23]). The main idea is that each feature corresponds to a single weak classi er and boosting selects from these features. Given a set of possible features in each iteration step n all features are evaluated. The best one is selected and forms the weak hypothesis h_n^{weak} which is added to the nal strong classi er h^{strong}.

This process as described above works off-line. Thus, all training samples must be given in advance. But for learning a dynamic background model we need an on-line learning method that can adept to changing environmental conditions as new frames arrive. Thus, we apply an on-line version of boosting for feature selection [4]. The main idea is to introduce "selectors" and to perform boosting on these selectors and not directly on the weak classi ers. Each selector $h^{sel}(\mathbf{x})$ holds a set of M weak classi ers $\{h_1^{weak}(\mathbf{x}), \ldots, h_M^{weak}(\mathbf{x})\}$ and selects one of them according to an optimization criterion based on the estimated error e_i of the classi er h_i^{weak}:

$$h^{sel}(\mathbf{x}) = h_m^{weak}(\mathbf{x}), \quad m = \arg\min_i(e_i) \tag{4}$$

Moreover, the importance/dif culty of a sample is estimated by propagating it through the set of selectors. For more details see [4]. But since the creation of weak classi ers is very important for our speci c task this step is described explicitly in this paper.

For on-line learning a weak classi er h_j^{weak} for a feature j we rst build a model by estimating the probability $P(1|f_j(\mathbf{x})) \sim \mathcal{N}(\mu^+, \sigma^+)$ for positive labeled samples and $P(-1|f_j(\mathbf{x})) \sim \mathcal{N}(\mu^-, \sigma^-)$ for negative labeled samples, where $f_j(\mathbf{x})$ evaluates this feature on the image $\mathbf{x}$. The mean and variance are incrementally estimated by applying a Kalman- ltering technique. Next, to estimate the hypothesis for the Haar-Wavelets we use either simple thresholding

$$h_j^{weak}(\mathbf{x}) = p_j \cdot \text{sign}(f_j(\mathbf{x}) - \theta_j), \tag{5}$$

where

$$\theta_j = |\mu^+ + \mu^-|/2, \quad p_j = \text{sign}(\mu^+ - \mu^-) \tag{6}$$

or a Bayesian decision criterion

$$\begin{aligned} h_j^{weak}(\mathbf{x}) &= \text{sign}(P(1|f_j(\mathbf{x})) - P(-1|f_j(\mathbf{x}))) \\ &\approx \text{sign}(g(f_j(\mathbf{x}|\mu^+, \sigma^+) - g(f_j(\mathbf{x})|\mu^-, \sigma^-)), \end{aligned} \tag{7}$$

where $g(x|\mu, \sigma)$ is a Gaussian probability density function. For histogram features (orientation histograms and LBPs), we use nearest neighbor learning with a distance function D (e.g., Euclidean):

$$h_j^{weak}(\mathbf{x}) = \text{sign}(D(f_j(\mathbf{x}), \mathbf{p}_j) - D(f_j(\mathbf{x}), \mathbf{n}_j)) \tag{8}$$

The cluster centers for positive $\mathbf{p}_j$ and negative $\mathbf{n}_j$ samples are learned by estimating the mean and the variance for each bin separately.

For the application of background modeling we *do not have negative examples*. But, we can treat the problem as an one-class classi c ation problem and use only positive samples for updating. The key idea is to calculate the negative distribution for each feature directly without learning. We model the gray value of each pixel as uniformly distributed

with mean 128 and variance $\frac{256^2}{12}$ (for an 8 bit image). Applying standard statistics the parameters of the negative distribution μ^- and σ^- of Haar features can be easily computed. For orientation histogram features the negative cluster $\mathbf{n}_j$ consists of equally distributed orientations. The characteristic negative cluster for a 16 bin LBP-feature is given by $\mathbf{n}_j = \frac{1}{50} \cdot [6, 4, 4, 1, 4, 1, 1, 4, 4, 1, 1, 4, 1, 4, 4, 6]$ for the binary patterns $[0000_2, 0001_2, 00010_2..., 1111_2]$[1].

Thus, we are able to compute the weak classi ers and use them for predicting the background (including statistical predictable changes). In the initial learning stage a separate classi er is built for all image patches assuming that all input images are positive examples. Later on, new input images are analyzed and the background model is updated according to a given policy.

Evaluation

A region is labeled as foreground if it can not be modeled by the classi er , i.e., the obtained con dence of the classi er is below a certain threshold:

$$conf(\mathbf{x}) < \theta^{eval} \tag{9}$$

Update

For updating the classi ers we adopt the following very simple policy. We update a classi er if its con dence response is within a certain interval:

$$\theta_{lower}^{update} < conf(\mathbf{x}) \leq \theta_{upper}^{update} \tag{10}$$

Usually $\theta_{lower}^{update} = \theta^{eval}$ and the upper threshold is set to avoid over- tting. In addition, in the post-processing steps several regions (known objects) are excluded from updating for a certain time.

Due to the speci c type of features used for training the classi ers the background model is high sensitive and even small changes can be detected. Moreover, the proposed background model is capable of modeling dynamically changing backgrounds (e.g., ashing lights, moving leaves in the wind, ag waving, etc.). Since an ef cient data structure (integral image) is used the evaluation can be implemented very ef ciently

2.2. Known-object Identi er

Object Detector

For automatically learning a person model we apply Conservative Learning [17, 18]. Starting with motion detection an initial set of positive examples is obtained by analyzing the geometry (aspect ratio) of the motion blobs. If a blob ful lls the restrictions the corresponding patch is selected. Negative examples are obtained from images where no motion was detected. Using these data sets a rst discriminative classi er is trained using an on-line version of AdaBoost [4]. In fact, by applying this classi er all persons are detected (a general model was estimated) but there is a great amount of false positives. Thus, as a generative classi er robust PCA [19] is applied to verify the obtained detections and to decide if a detected patch represents a person or not. The detected false positives are fed back into the discriminative classi er as negative examples and the true positives as positive examples. As a huge amount of data (video stream) is available very conservative thresholds can be used for these decisions. Thus, most of the patches are not considered at all. Applying these update rules an incrementally better classi er is obtained. Moreover, an already trained classi er can be re-trained on-line and can therefore easily be adapted to a completely different scene. As the framework is very general we can apply it to learn a person detector as well as to learn a model for cars. For the whole procedure no user interaction is needed.

Object Tracker

After a target object was successfully detected a tracker is initialized. In particular we apply a tracker [5] that is based on on-line boosting [4] similar to the background classi er . First, to initialize the tracker, a detected image region is assumed to be a positive image sample. At the same time negative examples are extracted by taking regions of the same size as the target window from the surrounding background. Using these samples several iterations of the on-line boosting algorithm are carried out. Thus, the classi er adapts to the speci c target object and at the same time it is discriminating against its surrounding background. The tracking step is based on the approach of template tracking. We evaluate the current classi er on a region of interest and obtain a con dence value for each sub-patch. We analyze the con - dence map and shift the target window to the (new) location where the con d ence value is maximized. Once the object has been tracked the classi e r has to be updated in order to adjust to possible changes in appearance of the target object and its background. The current target region is used as a positive update of the classi er while the surrounding regions represent the negative samples. As new frames arrive the whole procedure is repeated. The classi er is therefore able to adapt to changes in appearance and in addition it becomes robust against background clutter. Note that the classi er focuses on the current target object while at the same time it attempts to distinguish the target from its surrounding. Moreover, tracking of multiple objects is feasible just by initializing a separate classi er for each target object.

[1] These numbers are obtained by a lengthly calculation assuming equal probability of all patches.

3. Experimental Results

To show the power of the approach we applied the proposed framework on three different scenarios. The rst experiment was carried out on the PETS 2006 Benchmark Data that is publicly available[2]. But as this paper is mainly focused on change detection (left objects as well as objects that were removed) by using a robust background model the detection of left luggage (briefcase, bag, etc.) is not limited to the luggage that was left by a certain person. In addition, we have created various sequences showing a corridor in a public building and a tunnel. Each of the sequences demonstrates a special dif culty that can be handled by our framework.

To obtain comparable results we have used the same parameter settings for all experiments. For training the classi ers (background, detector) the size of the pool of weak classi ers was limited to 250 for all modules. The classi er for the tracker and the background model was estimated from a linear combination of 30 weak classi ers (selectors) whereas for the more exact detector 50 weak classi ers where used. The search region for the tracker was set twice as large as the current object dimension. For estimating the background model patches of 20×20 pixels with 50% overlapp were de ned. The evaluation threshold was set to $\theta^{eval} = 0$ and for the update policy the parameters $\theta^{update}_{lower} = 0$ and $\theta^{update}_{upper} = 0.5$ were used. For our experiments we achieve a frame-rate of 5 to 10 frames per second on an 1.6 GHz PC with 1 GB RAM.

3.1. PETS 2006 Benchmark Data

First, we demonstrate our approach on the publicly available PETS 2006 Benchmark Data with supplied ground truth. Therefore, we have selected two sequences from different camera positions[3]. We have chosen these special sequences to show that we are not limited to a certain camera position or camera geometry. In addition, we can demonstrate that our background model can handle the occurrence of shadows and re ections.

The rst sequence was taken from a "side view". People are walking around and a man is entering the scene and leaves a suitcase behind. Typical frames of this scene are shown in Figure 4(a). Thus, the de ned task was to detect the suitcase. Therefore, rst the suitcase as well as the persons are detected as non-background objects (Figure 4(b)). Second, the "known-object identi er" (detector/tracker) detects all known objects, i.e., the persons walking or standing around (Figure 4(c)). Finally, when combining the results from the background model and the detections obtained by the tracker only the suitcase is detected (Figure 4(d)). In addition to the "known-object identi er" a region growing algorithm is applied. All non-background patches that are within the same region as a detected person are assumed to be part of the person. Thus, an outstretched arm or a drawn suitcase (see Figure 4, second row) are recognized as a part of a person and are therefore not labeled as unknown objects. Since the current implementation of the person detector can not detect partial persons (person entering or leaving the scene) such patches were not considered at all.

The second sequence taken from a "semi frontal view" is more dif cult for our method because persons and other unknown objects (luggage) are present in the scene from the very beginning (see Figure 5). As can be seen in Figure 5(b-d) (second row) persons are not a problem. If a person was detected the corresponding region is not used for updating the background model. Thus, the person in not included in the background. When combining the results of both modules the person is not detected any more. In contrast to persons unknown objects are modeled as background. The same applies for persons that are not detected (e.g., children may not detected due to the restriction from the calibrated ground plane). But all other objects, i.e., the ski-sack near to the glass wall, the bag and even the newspaper on the bench were detected as left luggage!

For evaluation we analyzed the detections of the suitcase (Sequence 1) and of the ski-bag (Sequence 2). Thus, the true positives and the false positives were counted starting with the rst occurrence of the left object (Sequence 1: frame 1210/3400, Sequence 2: frame 1962/2400). The results are summarized in Table 1.

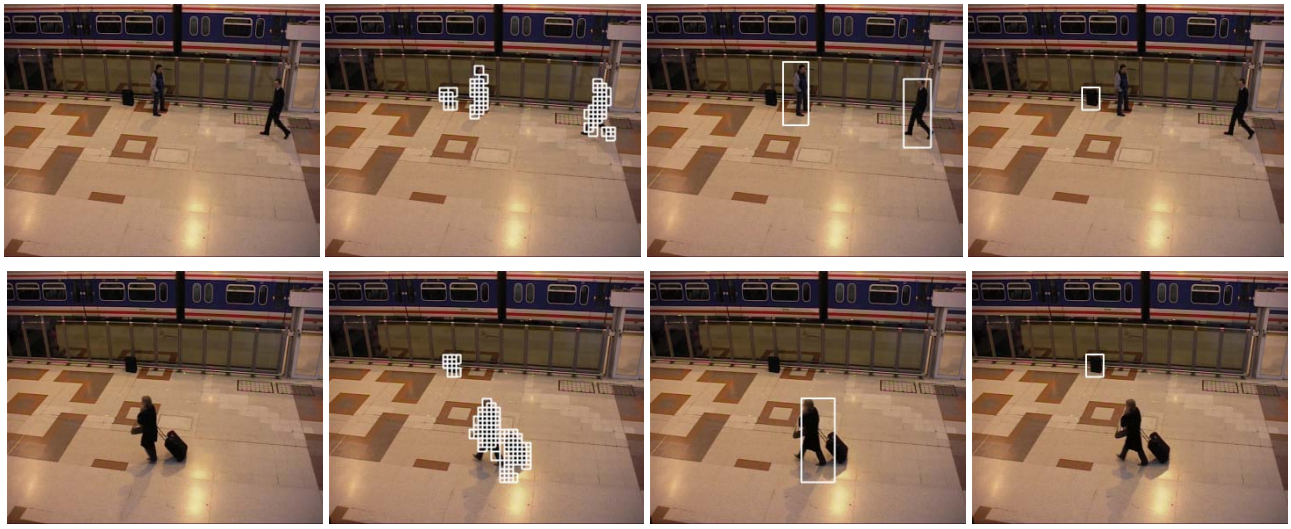
sequence	true pos.	false pos.
PETS Seq. 1	91.5%	3.4%
PETS Seq. 2	96.6%	1.9%

Table 1: Evaluation Results for the PETS 2006 Sequences.

For both sequences we obtain a detection rate of more than 90%, where most of the misses arise from non-background regions growing together (e.g., a person is passing by a left object very closely) which can be easily avoided by temporal lters. Thus, the detection rate for the second sequence is much better compared to the rst sequence since less persons are passing by the left object very close. The false positives are the result of temporary unstable detections (e.g., a thrown suitcase or a cast shadow is detected as left luggage) or may be caused by changes in background that were not learned during the training stage. But by using simple logic and time constraints the number of false alarms can be reduced.

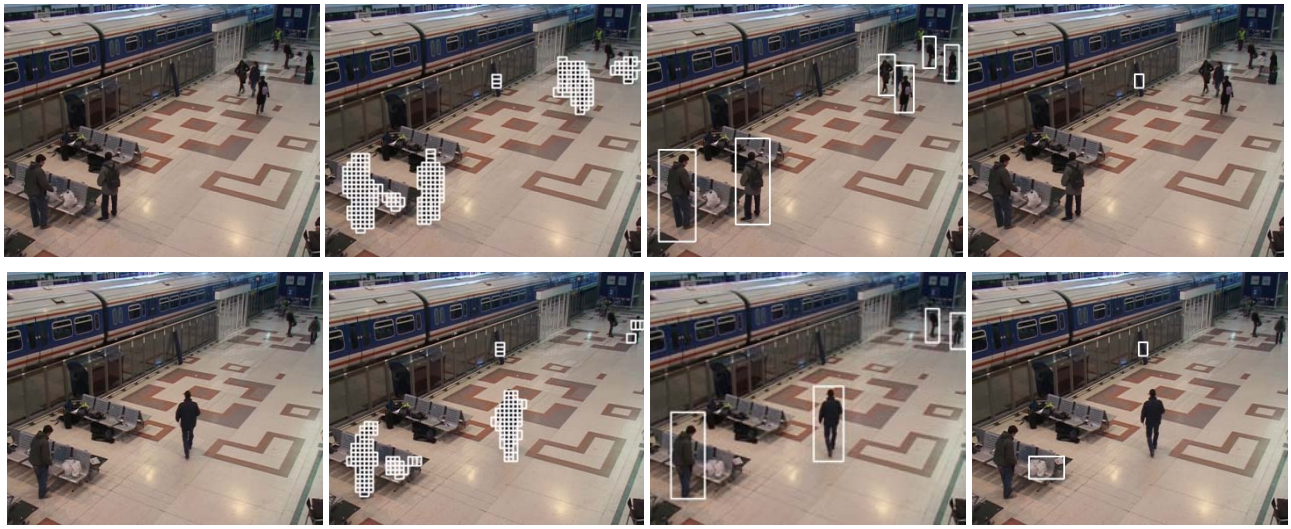
[2] http://www.pets2006.net, April 25th, 2006

[3] Dataset S7 (Take 6-B), Camera 3 and Dataset S5 (Take 1-G), Camera 4.



(a) (b) (c) (d)

Figure 4: PETS 2006 Dataset - Sequence 1: (a) original image, (b) background patches that can not be explained by the background model, (c) detected (tracked) persons, (d) nally detected the suitcase.



(a) (b) (c) (d)

Figure 5: PETS 2006 Dataset - Sequence 2: (a) original image, (b) background patches that can not be explained by the background model, (c) detected (tracked) persons, (d) nally detected the ski-sack (and in addition the bag and the newspaper on the bench).

(a) (b) (c) (d) (e)

Figure 6: CoffeeCam / Larceny Scenario - Changes in the background model: (d) the missing poster (object removed) and (e) the poster lying on the oo r (object added) are detected.

(a) (b) (c) (d)

Figure 7: Tunnel Safety - Objects that were thrown out of the car are detected: (a) no object, (b) chock, (c) car tire, (d) safety cone.

3.2. CoffeeCam

Next, we demonstrate that our framework can cope with dynamic backgrounds. Therefore, we have taken several sequences showing a corridor in a public building near to a coffee dispenser. The dynamic background was simulated by using a ashlight. Several typical scenarios including left luggage and the larceny of paintings were de ned and evaluated.

In the following the results obtained for the larceny of paintings scenario are presented. After a background model was learned from dynamically changing background images (blinking alarm light) the corridor was kept under surveillance. To simulate the larceny the poster was removed by one person and thrown down to the oor . Figure 6 shows the detection results of ve consecutive frames. In the beginning, persons are walking around and nothing suspicious is detected (Figure 6(a-b)). Then, the poster is removed but the changed background area is occluded by the person (Figure 6(c)). Finally, the missing poster (Figure 6(d)) as well as the poster lying on the oor (Figure 6(e)) are detected.

3.3. Tunnel Safety

Finally, to show the generality of our approach we demonstrate the method on a tunnel safety task. In addition, we show that we can also detect objects of low contrast that would not be detected by using a standard approach. Figure 8 shows the rs t (a) and the last (b) frame of a test sequence. It is even hard for a human to detect all three objects that were thrown out of the car! Moreover, each of the objects (a chock, a car tire and a safety cone) has a size of only a few pixels. Due to lights of cars, warning lights of trucks etc. the background is changing over time; a dynamic multi-modal background model is needed to robustly detect changes. In fact, our framework handles both, the dynamic background and the low contrast video data. Thus, detection results of four subsequent frames are shown in Figure 7.

(a) (b)

Figure 8: Tunnel Safety: Due to lightening conditions and to low quality cameras the contrast is very low.

4. Summary and Conclusion

We have presented a framework for detecting changes in the background. Thus, we are able to detect unknown objects or objects that were removed. A new robust background model that is based on on-line learning feature based classi ers and an object detector (tracker) are combined. Thus, detected changes are veri ed by the detector and all regions that can not be explained are returned as unknown foreground objects. In addition, detected regions are excluded from updating. Thus, a background model can be learned even if (known) foreground objects are present in the scene. The proposed background model is very sensitive, i.e., even objects in low contrast images are detected. In addition, it can handle multi-modalities, i.e., dynamic changes in the background. As for all components the same data structures (integral representations) are used the whole framework can be implemented in a very ef cient way. Moreover, all components run in an unsupervised manner.

References

[1] S.-C. S. Cheung and C. Kamath. Robust techniques for background subtraction in urban traf c video. In *Proc. SPIE Visual Communications and Image Processing*, pages 881–892, 2004.

[2] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[3] Y. Freund and R. Schapire. A short introduction to boosting. *Journal of Japanese Society for Arti cial Intelligence*, 14(5):771–780, 1999.

[4] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2006. (in press).

[5] M. Grabner, H. Grabner, and H. Bischof. Real-time tracking with on-line feature selection. In *Video Proc. for IEEE Conf. on Computer Vision and Pattern Recognition*, 2006. (in press).

[6] D. Hall, J. Nascimento, P. C. R. E. Andrade, P. Moreno, S. P. T. List, R. Emonent, R. B. Fisher, J. Santos-Victor, and J. L. Crowley. Comparision of target detection algorithms using adaptive background models. In *Proc. IEEE Workshop on VS-PETS*, pages 113–120, 2005.

[7] M. Heikkilä and M. Pietikäinen. A texture-based method for modeling the background and detecting moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28:657 – 662, 2006.

[8] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B.Rao, and S. Russell. Towards robust automatic traf c scene analysis in real-time. In *Proc. Intern. Conf. on Pattern Recognition*, volume I, pages 126–131, 1994.

[9] K. Levi and Y. Weiss. Learning object detection from a small number of examples: The importance of good features. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 53–60, 2004.

[10] B. Lo and S. A. Velastin. Automatic congestion detection system for underground platforms. In *Proc. IEEE Intern. Symposium on Intelligent Multimedia , Video and Speech Processing*, pages 158–161, 2001.

[11] N. J. McFarlane and C. P. Scho eld. Segmentation and tracking of piglets. *Machine Vision and Applications*, 8(3):187–193, 1995.

[12] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classi cation with local binary patterns. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[13] N. M. Oliver, B. Rosario, and A. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[14] R. P ugfelder and H. Bischof. Online auto-calibration in man-made worlds. In *Digital Image Computing: Technqiues and Applications*, pages 519 – 526, 2005.

[15] M. Piccardi. Background subtraction techniques: a review. In *Proc. IEEE Intern. Conf. on Systems, Man and Cybernetics*, volume 4, pages 3099–3104, 2004.

[16] F. Porikli. Integral histogram: A fast way to extract histograms in cartesian spaces. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 829–836, 2005.

[17] P. M. Roth and H. Bischof. On-line learning a person model from video data. In *Video Proc. for IEEE Conf. on Computer Vision and Pattern Recognition*, 2006. (in press).

[18] P. M. Roth, H. Grabner, D. Skocaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Proc. IEEE Workshop on VS-PETS*, pages 223–230, 2005.

[19] D. Skocaj and A. Leonardis. Weighted and robust incremental method for subspace learning. In *Proc. IEEE Intern. Conf. on Computer Vision*, volume II, pages 1494–1501, 2003.

[20] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume II, pages 246–252, 1999.

[21] Y.-L. Tian, M. Lu, and A. Hampapur. Robust and ef cient foreground analysis for real-time video surveillance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 1182 – 1187, 2005.

[22] K. Tieu and P. Viola. Boosting image retrieval. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 228–235, 2000.

[23] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, volume I, pages 511–518, 2001.

Overview of the PETS2006 Challenge

David Thirde, Longzhen Li and James Ferryman
Computational Vision Group
School of Systems Engineering
The University of Reading
Whiteknights, Reading, RG6 6AY, UK

Abstract

This paper describes the event recognition challenge that forms part of the PETS 2006 workshop. The aim of this challenge is to use existing systems for the detection of left (i.e. abandoned) luggage in a real-world environment. The dataset scenarios were lmed from multiple cameras and involve multiple actors.

1. Introduction

Visual surveillance is a major research area in computer vision. The recent rapid increase in the number of surveillance cameras has led to a strong demand for automatic methods of processing their outputs. The scienti c challenge is to devise and implement automatic systems for obtaining detailed information about the activities and behaviours of people and vehicles observed by a single camera or by a network of cameras. The growth in the development of the eld has not been met with complementary systematic performance evaluation of developed techniques. It is especially dif cult to make comparisons between algorithms if they have been tested on different datasets under widely varying conditions.

PETS 2006 continues the theme of the highly successful PETS workshop series [2]. For PETS 2006, the theme is multi-sensor event recognition in crowded public areas. As part of this workshop a challenge was set to describe an approach to event recognition and report results based on annotated datasets made available on the website [1]. This paper details the datasets and the challenge that the contributing authors had to present solutions for.

2. The PETS2006 Challenge — Detection of Left-Luggage

The aim of the PETS 2006 challenge is to detect left items of luggage within a public space. Automated left-luggage detection systems will allow security services to respond quickly to potentially hazardous situations, improving the security and safety of public areas. Left-luggage in the context of PETS 2006 is de ned as items of luggage that have been abandoned by their owner. In the published scenarios each item of luggage has one owner and each person owns at most one item of luggage. To implement a system based on this de nition there are three additional components that need to be de ned:

What items are classed as luggage? Luggage is de ned to include types of baggage that can be carried by hand e.g. trunks, bags, rucksacks, backpacks, parcels, and suitcases. Five speci c types of luggage considered in this study are: a briefcase, a suitcase, a 25 litre rucksack, a 75 litre backpack, and a ski gear carrier.

What constitutes attended and unattended luggage? In this study three rules are used to determine whether luggage is attended to by a person (or not):

1. A luggage is owned and attended to by a person who enters the scene with the luggage until such point that the luggage is not in physical contact with the person (contextual rule).

2. At this point the luggage is attended to by the owner ONLY when they are within a distance a metres of the luggage (spatial rule). All distances are measured between object centroids on the ground plane (i.e. $z = 0$). Figure 1 shows a person within a $(= 2)$ metres of their luggage. In this situation no alarm should be raised by the system.

3. A luggage item is unattended when the owner is further than b metres (where $b >= a$) from the luggage. Figure 2 shows a person crossing the line at b $(= 3)$ metres. In this situation the system should use the spatio-temporal rule, below, to detect whether this item of luggage has been abandoned (an alarm event).

Note that if distance $b > a$, the distance between radii a and b is determined to be a warning zone where the luggage is neither attended to nor left unattended. This zone

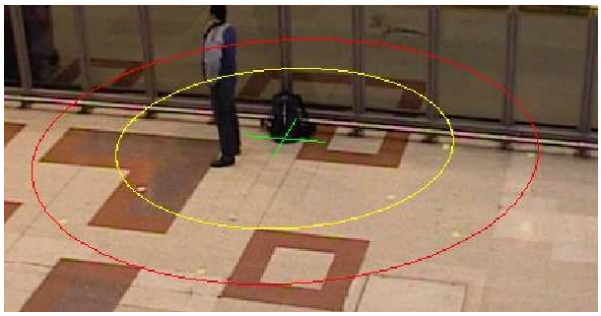


Figure 1: A person within the a ($= 2$) metre radius (marked in yellow) around their luggage (marked by a green cross).

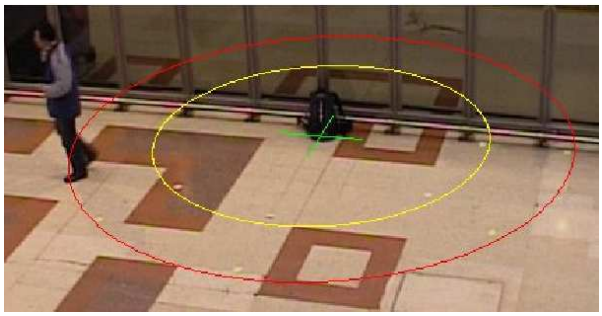


Figure 2: A person crossing the b ($= 3$) metre radius (marked in red) around their luggage (marked by a green cross).

is de ned to separate the detection points of the two states, reducing uncertainties introduced due to calibration / detection errors in the sensor system etc. Figure 3 shows a person crossing the line at a ($= 2$) metres, but within the radius b ($= 3$) metres. In this scenario the system can be set up to trigger a warning event, using a rule similar to the spatio-temporal rule below.

What constitutes abandonment of luggage by the owner? The abandonment of an item of luggage is de ned spatially and temporally. Abandonment is de ned as an item of luggage that has been left unattended by the owner for a period of t consecutive seconds in which time the owner has not re-attended to the luggage, nor has the luggage been attended to by a second party (instigated by physical contact, in which case a theft / tampering event may be raised). Figure 4 shows an item of luggage left unattended for t ($= 30$) seconds, at which point the alarm event is triggered.

2.1. The Datasets

Seven datasets were recorded for the workshop at Victoria Station in London, UK. The datasets comprise multi-sensor

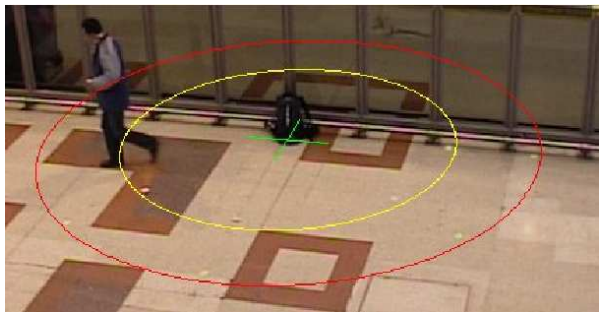


Figure 3: A person crossing the a ($= 2$) metre radius, but within the b ($= 3$) metre radius around their luggage (marked by a green cross).

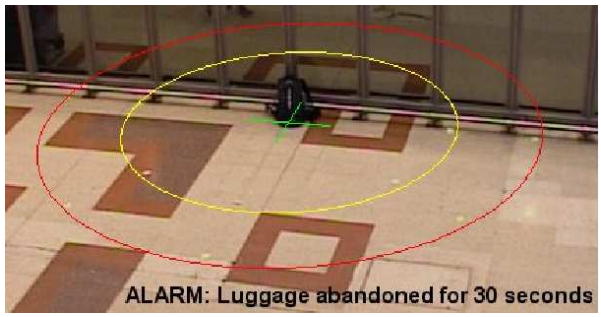


Figure 4: An item of luggage left unattended for t ($= 30$) seconds, at which point the alarm event is triggered.

sequences containing left-luggage scenarios with increasing scene complexity.

Dataset S1 (Take 1-C) This scenario contains a single person with a rucksack who loiters before leaving the item of luggage unattended. Figure 5 shows representative images captured from cameras 1-4.

Dataset S2 (Take 3-C) This scenario contains two people who enter the scene from opposite directions. One person places a suitcase on the ground, before both people leave together (without the suitcase). Figure 6 shows representative images captured from cameras 1-4.

Dataset S3 (Take 7-A) This scenario contains a person waiting for a train, the person temporarily places their briefcase on the ground before picking it up again and moving to a nearby shop. Figure 7 shows representative images captured from cameras 1-4.

Dataset S4 (Take 5-A) This scenario contains a person placing a suitcase on the ground. Following this a second person arrives and talks with the rst person. The rst person leaves the scene without their luggage. Distracted by a

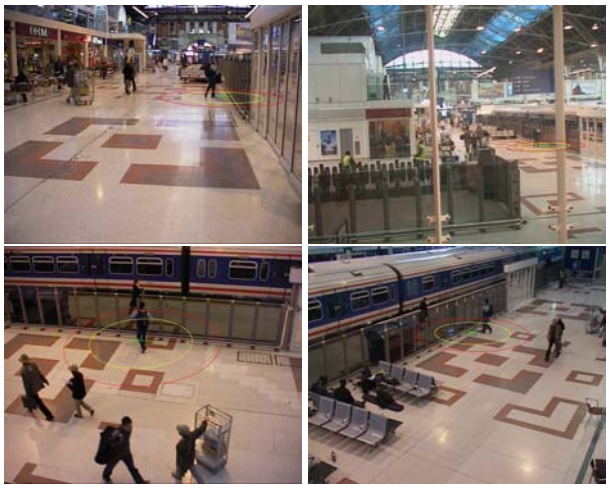


Figure 5: Dataset S1, frame 01080.

Figure 6: Dataset S2, frame 01400.

newspaper, the second person does not notice that the rst person's luggage is left unattended. Figure 8 shows representative images captured from cameras 1-4.

Dataset S5 (Take 1-G) This scenario contains a single person with ski equipment who loiters before abandoning the item of luggage. Figure 9 shows representative images captured from cameras 1-4.

Dataset S6 (Take 3-H) This scenario contains two people who enter the scene together. One person places a rucksack on the ground, before both people leave together (without the rucksack). Figure 10 shows representative images captured from cameras 1-4.

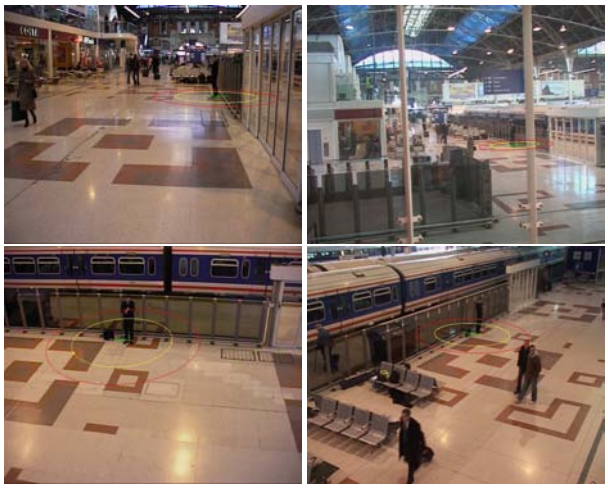


Figure 7: Dataset S3, frame 01110.

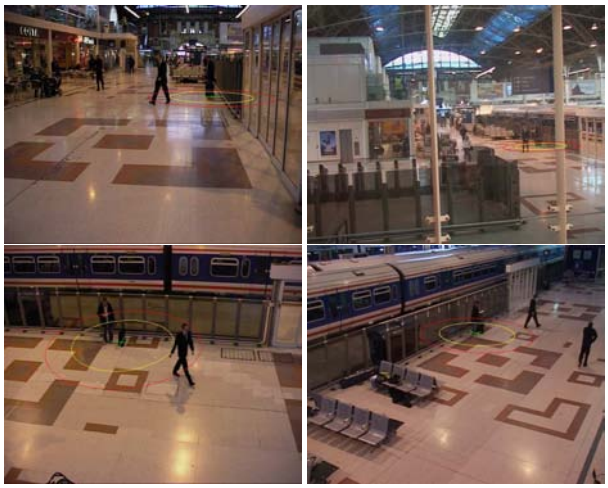


Figure 8: Dataset S4, frame 01858.

Dataset S7 (Take 6-B) This scenario contains a single person with a suitcase who loiters before leaving the item of luggage unattended. During this event ve other people move in close proximity to the item of luggage.

2.2. XML Ground Truth

All scenarios are provided with two XML les. The rst of these les contains the camera calibration information. The geometric patterns on the oor of the station were used to calibrate the cameras using the Tsai model [3]. All datasets were lmed using 2× Canon MV-1 (1×CCD w/progressive scan) and 2× Sony DCR-PC1000E (3×CMOS) DV cameras. The resolution of all sequences are PAL standard (768 x 576 pixels, 25 frames per second) and compressed as JPEG image sequences (approx. 90% quality).

The second XML le provided contains scenario details,

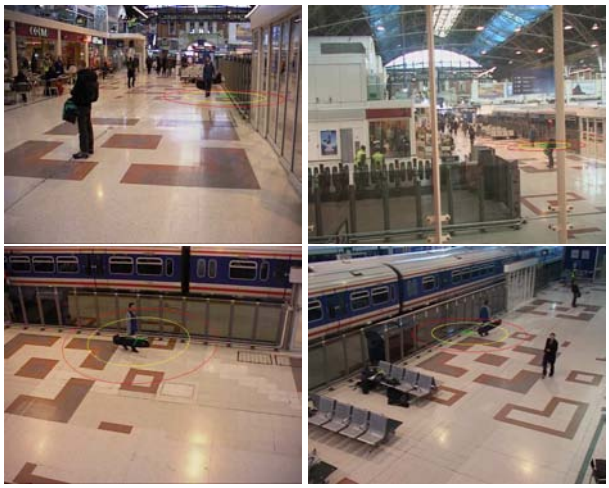


Figure 9: Dataset S5, frame 01460.

Figure 10: Dataset S6, frame 01660.

parameters and ground-truth information (e.g. the radii distances, luggage location, warning / alarm triggers etc). More details on the XML schemas can be found on the PETS 2006 website [1].

3. Discussion

In this paper we have described the PETS 2006 datasets and challenge. This workshop is addressing the problem of left-luggage detection within a public space. The PETS 2006 challenge provide researchers with the opportunity to evaluate their existing left luggage detection algorithms on datasets captured in a real-world environment.

Figure 11: Dataset S7, frame 01650.

Acknowledgements

This work is supported by the EU, grant ISCAPS (SEC4-PR-013800).[1]

Legal Note

The UK Information Commisioner has agreed that the PETS 2006 data-sets described here may be made publicly available for the purposes of academic research. The video sequences are copyright ISCAPS consortium and permission for the publication of these sequences is hereby granted provided that the source is acknowledged.

References

[1] "Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance", http://www.pets2006.net.

[2] "PETS: Performance Evaluation of Tracking and Surveillance", http://www.cvg.cs.rdg.ac.uk/slides/pets.html.

[3] R. Tsai. An ef cient and accurate camera calibration technique for 3d machine vision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 323–344, 1986.

[1]This paper does not necessarily represent the opinion of the EU; the EU is not responsible for any use which may be made of its contents.

Left-Luggage Detection using Homographies and Simple Heuristics

Edouard Auvinet, Etienne Grossmann, Caroline Rougier, Mohamed Dahmane and Jean Meunier

Department of Computer Science and Operations Research
University of Montreal
Montreal, CANADA H3C 3J7

Abstract

Today, video surveillance is commonly used in security systems, but requires more intelligent and more robust technical approaches. Such systems, used in airports, train stations or other public spaces, can bring security to a higher level. In this context, we present a simple and accurate method to detect left luggage in a public area, which is observed by a multi-camera system and involving multiple actors. We first detect moving objects by background subtraction. Then, the information is merged in the ground plane of the public space floor. This allows us to alleviate the problem of occlusions, and renders trivial the image-to-world coordinate transformation. Finally, a heuristic is used to track all objects and detect luggage items left by their owners. An alarm is triggered when the person moves too far away from his luggage during a too long period of time. Experimental results prove the efficiency of our algorithm on PETS 2006 benchmark data.

1. Introduction

Faced with the increasing need of security in public spaces, public and commercial interest pushes research to develop active prevention solutions, capable of detecting suspicious events while they occur, rather than just recording them. For example, iOmniscient [4] claims to provide intelligent video surveillance software that detects objects left in crowded or busy areas, using a Non-Motion Detection (NMD) technique.

Surveillance applications developed nowadays are part of third generation surveillance systems [12], that cover a wide area using a multi-camera network. Typical watched areas are sensitive public places and infrastructures, that are susceptible of being crowded. Tracking people in a crowded environment is a big challenge, since, in image space, we must deal with merging, splitting, entering, leaving and correspondence. The problem is more complicated when the environment is observed by multiple cameras. To deal with this, approaches have been proposed which can be classified in two categories : uncalibrated and calibrated.

An interesting example of the uncalibrated method is proposed by Khan and Shah [5]. They take advantage of the lines delimiting the field of view of each camera, which they called *Edges of Field of View*. Similarly, Calderara *et al.* [1] introduce the concept of *Entry Edges of Field of View* to deal with false correspondences.

Among calibrated methods, we can cite the work of Yue *et al.* [13] who use homographies to solve occlusions. A second method is proposed by Mittal and Davis [7] which is based on epipolar lines.

The advantage of having calibrated cameras is that it greatly facilitates the fusion of visual information produced by many cameras.

A partial calibration, in which only camera-to-ground-plane homographies are known, is often used. Indeed, homographies are much easier to obtain than general calibration, while still providing a very useful image-to-world mapping.

In this paper, we will present an algorithm to detect abandoned luggage in a real world public environment. This is a typical challenge of nowadays surveillance systems. For testing purposes, we will use the PETS[1] datasets [10], described below in section 2, that provides multi-camera sequences containing left-luggage scenarios. We will exploit the fact that the PETS datasets provides calibration and sufficient data to estimate homographies.

The method we developed here is similar to the technique recently proposed by Khan and Shah [6]. They present a planar homography constraint to resolve occlusions and detect the locations of people on the ground plane corresponding to their feet.

Our video surveillance process is described in Figure 1. First, we perform a background subtraction in the image plane of each camera (see Section 3.1). Then, a homographic transformation is performed to merge information from all cameras in the scene floor homographic image (see Section 3.2). Finally, we work in the homographic image to track people using a heuristic method to detect suspect events (see Section 3.3).

[1] Performance Evaluation of Tracking and Surveillance

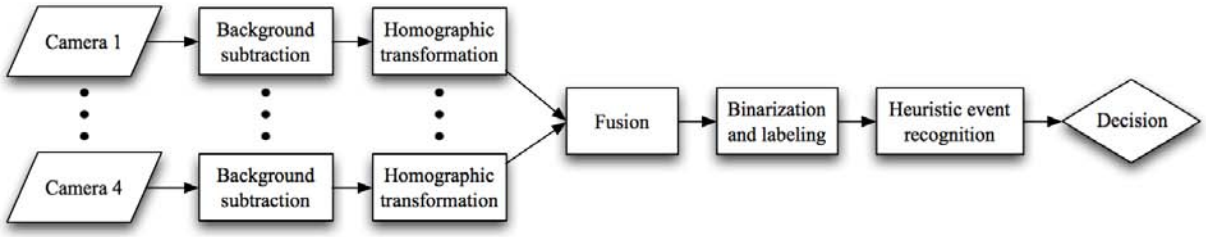


Figure 1: Scheme of the proposed algorithm.

Our main contribution is to present results obtained by a simple modular system. Its principal merit is that it has few parameters, most of them being easily identified physical quantities (e.g. minimum detected object size). In the technical description below, and in the conclusion, we will discuss principled ways of reducing even further the number of parameters.

2. Datasets

To test our algorithm, we used the datasets provided by the PETS 2006 workshop organization. These videos were taken in a real world public setting, a railway station, and made possible by the support and collaboration of the British Transport Police and Network Rail. There is a total of seven multi-camera sequences containing left-luggage scenarios with increasing scene complexity. Luggage items are of several different types (briefcase, suitcase, rucksack, backpack and even a ski gear carrier). Briefly stated, in the context of PETS 2006, a luggage has been left abandoned if the owner is farther than a given distance from the luggage (300cm) for a certain period of time (30 seconds). For these benchmark videos, calibration data for each individual camera are given and were computed from specific point locations (also given) taken from the geometric patterns on the floor of the station. Ground-truth information such as luggage locations and abandoned-luggage detection times are provided with each datasets. The scenarios involve up to 6 persons with a left-luggage occurrence in each one of them. They were filmed with four DV cameras with PAL standard resolution of 768 x 576 pixels and 25 frames per second.

3. Method

3.1. Motion Detection

Moving visual objects are segmented using a typical simple background subtraction with shadow removing. We construct the background image and classify as foreground any pixel that deviates from it by more than a certain threshold. This threshold is set so that at most ~ 1 % of background pixels are misclassified.

Automatic background model estimation techniques could have been used, but in the present work, a simple semi-automatic method was sufficient.

The background model consists of the median value C_{med} of each color component, for each pixel. The median is computed from ten images taken one second apart at the beginning of the sequence.

We now explain how the threshold is set. We consider the residues of RGB color components with respect to the background (the median) and the threshold is set so that 1 % of the background pixels are misclassified.

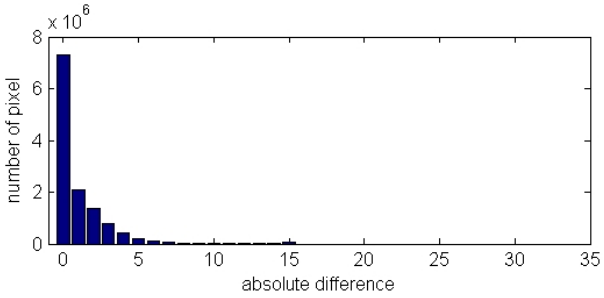


Figure 2: Values of the residues computed as in Eq. 1.

Figure 2 shows the residues computed from the ten images in one of the provided sequences. In the abscissa is the value:

$$\frac{\sum_{n=1}^{N} |C_n(i,j) - C_{med}(i,j)|}{N}, \quad (1)$$

where (i, j) is the pixel coordinate, and N the number of images (N=10 in our case).

The general shape of the histogram varies little from camera to camera, and sequence to sequence. A threshold of 15 gray levels is appropriate for all sequences and cameras, and used everywhere in this work. Potential foreground pixels are thus those that verify:

$$|I(i,j) - C_{med}(i,j)| \geq 15.$$

To obtain the final foreground silhouettes, shadows must be removed. For this purpose, the pixels with a reasonable

darkening level and a weak chromatic distortion [2], below a certain threshold, considered as shadow, are therefore ignored. Isolated spurious pixels are removed by a single erosion operation. We then perform 5 dilations with a 3x3 kernel. We will justify this dilation step in the Section 3.2.1.

Once the silhouette binary images are obtained, we are ready to fuse them in the floor homographic plane or orthoimage.

3.2. Information Fusion in the Orthoimage

To obtain coherence information from the different views, we work in the ground orthoimage obtained with homographic transformations. This allows us to overcome the problem of occlusions. Indeed, this orthoimage provides a birds-eye view of the ground plane with information on moving objects in contact with the floor.

3.2.1 Homographic transformation

We use PETS 2006 benchmark data which provides corresponding points on the floor in each field of view of the camera. Two methods could be used to remap the images to the ground plane (i.e. build orthoimages). The first is to use the camera calibration parameters provided in the PETS dataset (computed with the Tsai algorithm [11]). The second method is to compute just the homographic transformations from that maps corresponding points from the image plane to the ground plane. The homographic matrix is a 3x3 matrix and the homographic transformation is:

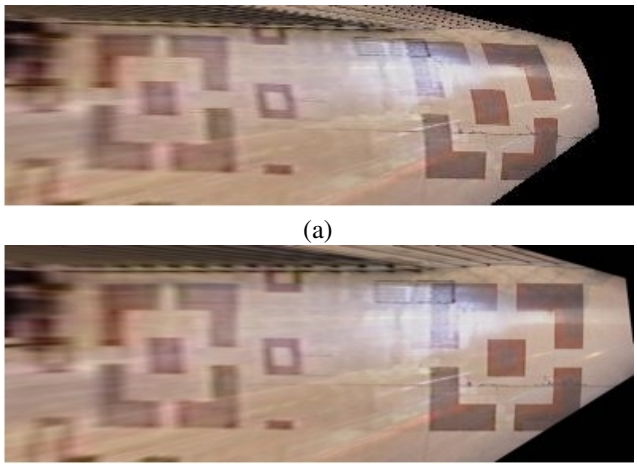
$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \cdot \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

with (x_1, y_1) and (x_2, y_2) a pair of corresponding points.

Using at least four pairs of corresponding points, we are able to compute the homographic matrix using a least-squares method. Since more than four pairs are provided, we use the linear (non-optimal) least-squares estimation method.

Figure 3 shows an example of the result obtained for the two methods for one of the cameras. We notice that the provided Tsai calibration parameters yield a less good orthoimage: apparently the radial distortion correction goes wrong (Figure 3 (a)). In comparison, the homographic transformation computed from the provided point correspondences is relatively free of distortion, so we decided to use only the homographic transformations.

After that, each silhouette binary image is transformed to the orthographic plane using the corresponding homographic matrix. The fusion in the orthoimage is made by adding the pixel values of all four images. In the resulting



(a)

(b)

Figure 3: The birds-eye view obtained with (a) the Tsai camera model and from (b) homographic transformation estimated from the PETS corresponding points. These views correspond to camera #1.

birds-eye view image, Figure 4 (e), the intersection of each person's silhouette is at the level of the feet.

Figure 5 shows the fusion without silhouette dilatation. We observe that there is a mapping error since the foot in each camera are not exactly overlapping after fusion in the floor homographic plane. This error is about 5-10 pixels and is mostly due to camera optical distortion that was not taken into account in our methodology. This is why, to overcome this problem and keep the algorithm as simple as possible, we decided to perform five silhouette dilatations to improve foot overlapping. Another advantage of dilatation is that it ensures the fusion of each foot blob into a unique more stable blob representing the person.

Figure 5: Mapping error in the orthoimage.

3.2.2 Extraction of the blobs

The orthographic image is used to detect the objects that touch the ground. In order to extract the corresponding blobs, we use the information of the overlapping field of

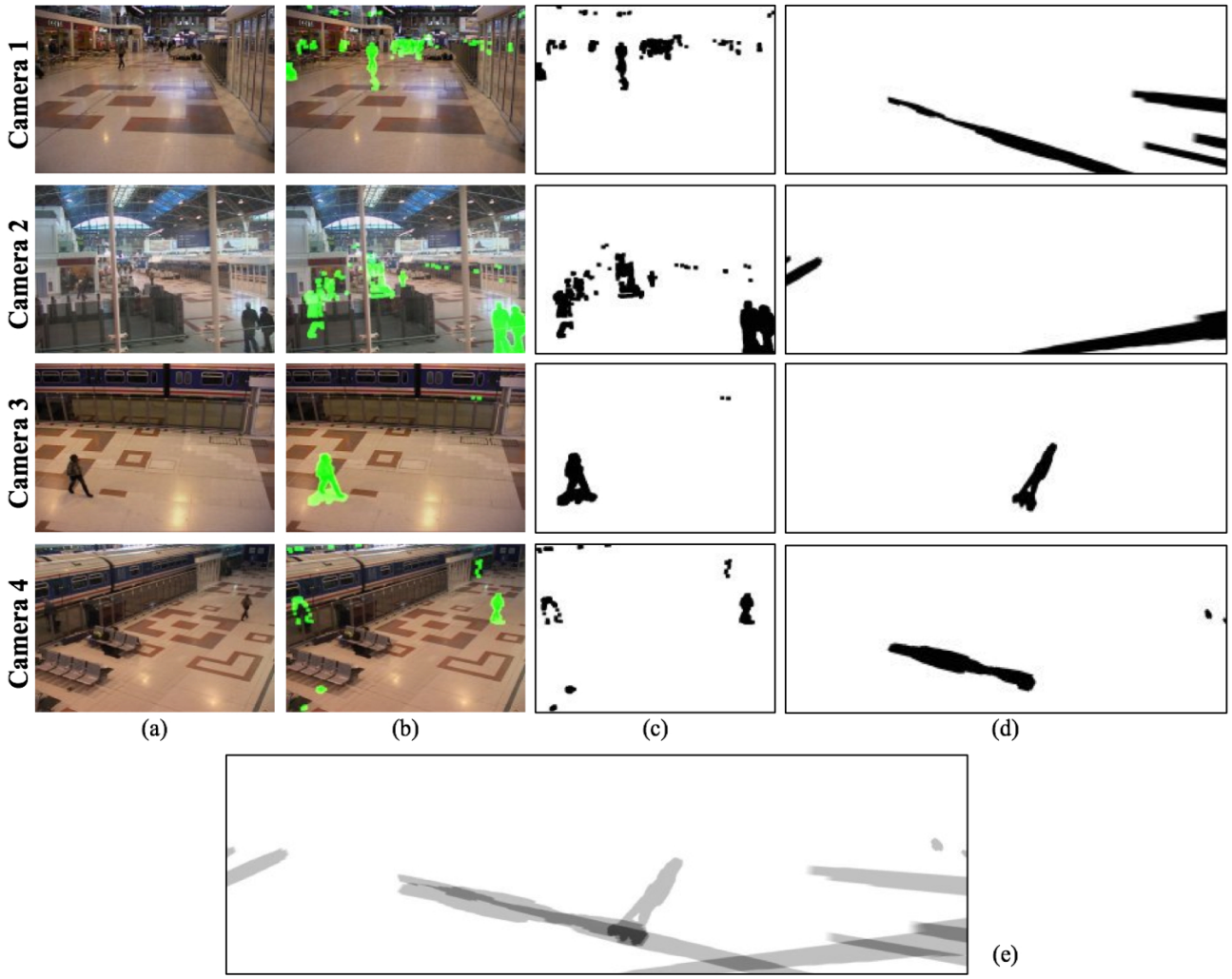


Figure 4: Silhouette extraction and fusion in the orthographic plane for frame #389 sequence 3 of the PETS 2006 dataset. (a) Original images, (b)(c) extracted silhouettes, (d) silhouette mappings and (e) silhouette fusion in the homographic plane.

view of the cameras (Figure 6). For instance, in a four-camera field of view (white area), the threshold for blob detection is set to three. This means that an overlapping of at least three silhouettes is necessary to create blobs: blobs can be detected in the white or light gray area of Figure 6.

3.3. Heuristic Event Recognition

Once the segmentation and blob labeling are performed in the orthoimages, only blob centroid positions (x,y), number of blob pixels N_{pix} and bounding boxes are passed to the event detection heuristic. The event detection heuristic has three main components:

1. Tracking of blobs from frame to frame, thus forming *spatio-temporal entities*.
2. Detection of spatio-temporal forks.
3. Detection of immobile blobs.

Based on the output of these components, a warning is raised when a fork has been detected and one branch (the "luggage" or "immobile object") is immobile, while another branch (the "owner") is more than $b = 300\ cm$ away. An alarm is raised when, in addition, the owner of the immobile object stays at a distance greater than b during 30 seconds. These are the definitions used in our results reported below.

We now describe each of the components in detail.

3.3.1 Tracking

For the purpose of tracking, we model blobs as circles of radius $\rho = \sqrt{N_{pix}/\pi}$, where N_{pix} is the known number of pixels in the blob. Two blobs are said to touch if their circles intersect, that is, if $\| (x_1, y_1) - (x_2, y_2) \|_2 \leq (\rho_1 + \rho_2)$

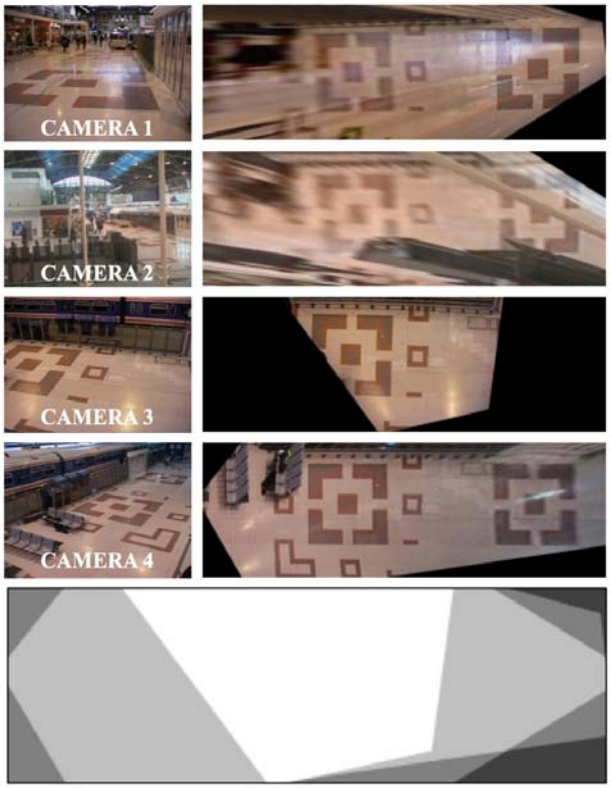


Figure 6: Original images and their homographic transformations, and the overlapping fields of view of all cameras.

where (x_1, y_1), (x_2, y_2) are the blob centroid positions, and ρ_1, ρ_2 are their radii.

If two blobs in consecutive frames touch, then they will be said to pertain to the same spatio-temporal entity (Figure 7).

We record the history of each entity, so that, when two distinct entities touch the same blob in a new frame, that blob will be said to pertain to the entity that has the largest number of previous blobs (Figure 8).

Figure 12 shows all the detected blobs, each colored according to its given label. Note that, with our definition of tracking, two blobs, in the same frame, that do not touch, may still pertain to the same entity (Figure 9).

3.3.2 Detection of spatio-temporal forks

Spatio-temporal forks correspond to objects that separate after having been in contact. Recognizing such patterns is fundamental to detect abandoned objects. A (possibly multi-pronged) fork is said to occur whenever two blobs that do not touch pertain to the same entity. In particular, we are interested in detecting forks in which one branch moves away, while the other remains immobile.

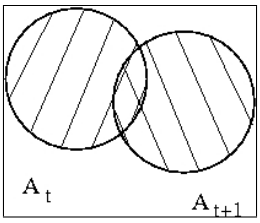


Figure 7: Tracking: blobs A_{t+1} and A_t, observed at times $t+1$ and t are considered to pertain to the same spatio-temporal entity, because they intersect in the orthoimage.

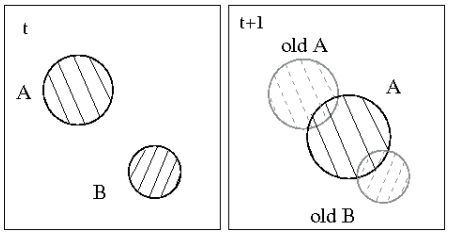


Figure 8: Merging of spatio-temporal entities: when two spatio-temporal entities A and B meet, only the label of the one with most previous blobs is kept.

3.3.3 Detection of immobile objects

Ideally, an immobile object would be characterized by a foreground blob that remains constant in time. In practice, blobs are subject to misdetections, spurious detections and poor localization.

We represent an immobile object as a ground position such that there exists a blob at less than 30cm in each frame, during more than 3 seconds. The immobile object exists as long as there is a blob at less than 30cm from its position. The position y_t of the immobile object (after t frames of existence) is defined as the mean of the closest blobs, at each frame:

$$y_t = \sum_{s=1}^{t} x_s,$$

where x_s is the blob, amongst all blobs x detected in frame s, that is closest to y_s: $x_s = \arg\min_x |x - y_s|$.

The distance of 30 cm corresponds to the uncertainty in the localization of the blob centroid. It has been chosen by examining the distribution of distances of blobs around potential immobile objects. The 3 second delay serves only to limit the number of potential immobile objects. Also, the position of a newly-abandoned object is unstable, due to the proximity of the owner. The 3 second delay also gives time for the object position to stabilize. Figures 10 and 11 further

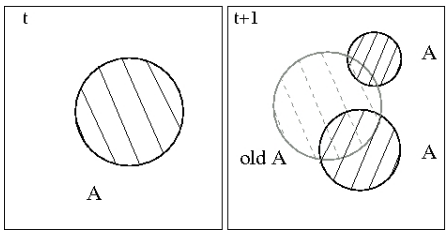


Figure 9: Forking, or branching, of spatio-temporal entities: blobs that are not connected may pertain to the same spatio-temporal entity, as a result of a fork

justify the choice of these quantities. Figure 10 shows the localizations of all detected blobs.

The local maximum of blob density would be the "ideal position" of the detected blob. The ground-truth position of the luggage (given in the PETS dataset), identified by a cross in the zoomed image, is slightly aside the maximum due to imprecision in the orthoimage construction.

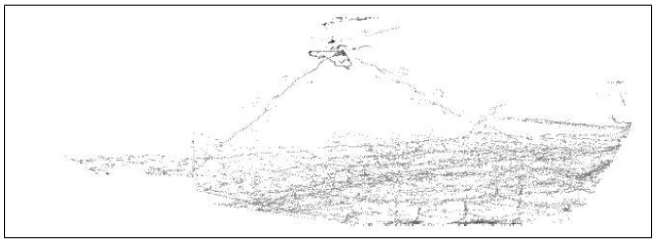
Although the local maximum appears well localized, there are frames in which no blob is in fact detected. In these frames the blob nearest to the object is much farther. These frames form spikes in Figure 11, which shows the distance between the ideal position (maximum density) and nearest detected blob, in Sequences 1 and 5. Given the amplitude of the spikes in this last sequence, a distance of 30 pixels is a safe choice.

We now explain how the three components above are combined to detect left luggage. At each time frame, we identify forks in which one branch is immobile and another (the "owner") is more than $b = 300\ cm$ away from the immobile branch. In such forks, we raise a warning and change the entity label of the owner, to be able to identify it later. If the luggage remains in place for 30 seconds, during which time the owner does not move closer than $b = 300\ cm$ from the luggage, then the object will be considered to be abandoned, and an alarm is raised.

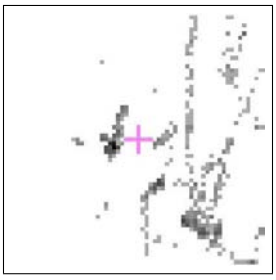
4. Results

In this section, we report our results on all seven PETS datasets. Two sets of results are given, one without the shadow suppression method, the other with shadow suppression. All parameters are otherwise exactly the same in all reported results.

An important information in video surveillance is the computation time. The pixel treatment is compiled in C++ with the OpenCV library [9], and the tracking runs under the Octave software [8]. On a Centrino, 2.26 GHz, the image treatment takes about 0.4 s per frame for 1200x400 homographic images. The tracking of objects with Octave takes 0.02 seconds per frame on a 1.4 GHz Celeron M.



Density of blob occurrences PETS 2006 in Sequence 1



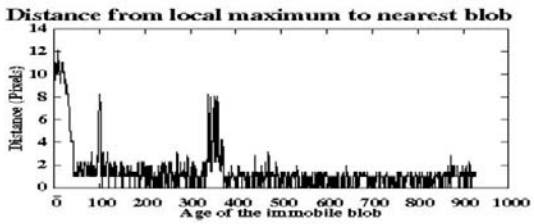
Zoom around the true abandoned object position (cross)

Figure 10: Density of blob occurrences in Sequence 1. The gray level represents the number of blobs detected at that pixel during the sequence. Top: Complete surveilled area; note the high density near the top, corresponding to the object, with the streaks left by the object carrier coming and going. Bottom: Zoom around the true object (location, marked with a cross), and the nearby local maximum of the number of detected blobs (dark mass).

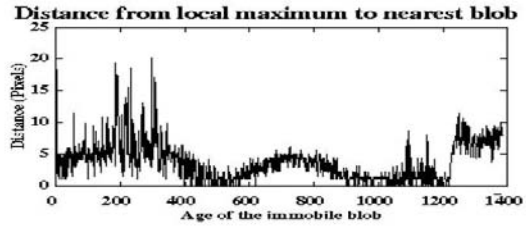
Seq.	TP	FP	Spatial error for TP (cm)	Temporal error for TP (s)	Subjective difficulty
1	1	0	25.8	+0.1	*
2	1	0	16.9	+2.5	***
3	0	0	-	-	*
4	1	0	63.9	+1.7	****
5	1	13	43.8	+0.2	**
6	1	0	43.9	+12.2	***
7	1	3	59.3	+0.5	*****

Table 1: Left-luggage detection without shadow removal. TP : True Positive (correct detection), FP : False Positive (incorrect detection). * : very easy, ***** : very difficult.

Tables 1 and 2 show the results of our algorithm on the seven datasets given by the PETS workshop organization. As expected the errors are usually larger when the shadows are not removed. Figure 13 shows the results in 3D: Warnings and alarms are represented by yellow and red spheres, respectively. Considering the simplicity of our methodology, the results are very satisfactory.



Blob occurrences in Sequence 1

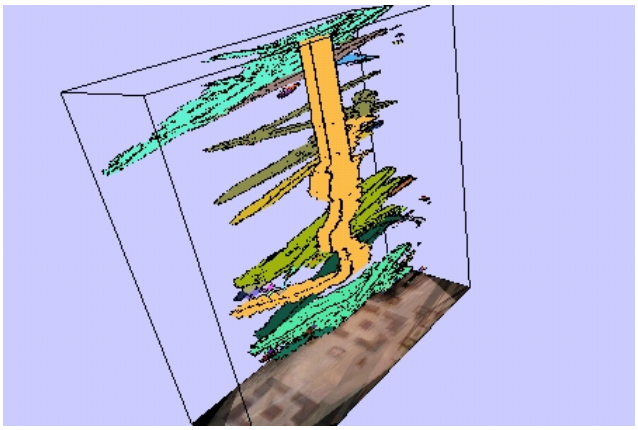


Blob occurrences in Sequence 1

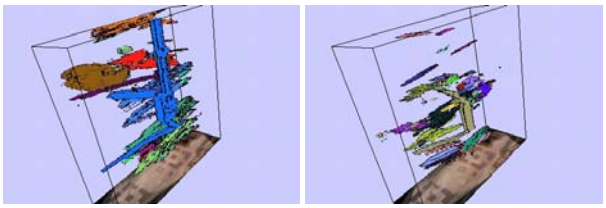
Figure 11: Distance, in cm, between local maximum of blob density and nearest blob, for the time interval during which the object is present there. The spikes in these curves justify a tolerance of 30cm during the localization of immobile objects.

Seq.	TP	FP	Spatial error for TP (cm)	Temporal error for TP (s)	Subjective difficulty
1	1	0	11.8	+0.0	*
2	1	0	15.6	+0.2	***
3	0	0	-	-	*
4	1	0	37.7	+1.0	****
5	1	5	48.4	+0.2	**
6	1	0	10.3	+2.3	***
7	1	0	70.9	+0.7	*****

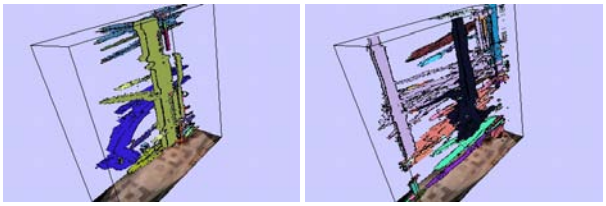
Table 2: Left-luggage detection with shadow removal. TP : True Positive (correct detection), FP : False Positive (incorrect detection). * : very easy, ***** : very difficult.



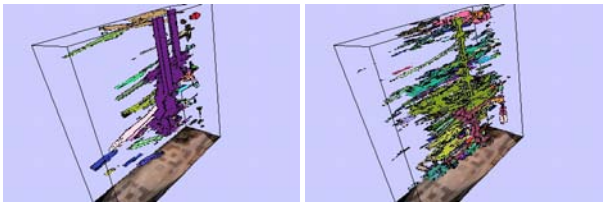
Sequence 1



Sequence 2 Sequence 3



Sequence 4 Sequence 5



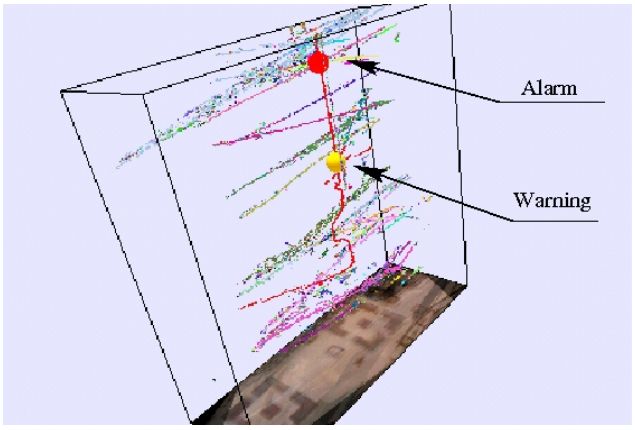
Sequence 6 Sequence 7

Figure 12: Spatio-temporal entities, identified by the heuristic of Section 3.3.1. Each entity is given a distinct color. Here, each blob is represented by its rectangular bounding box. One may identify some individual trajectories by a single color, while others are merged into a single entity.

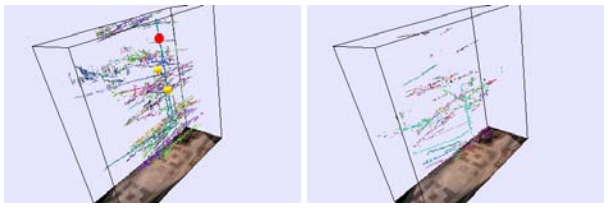
5. Conclusion

The proposed algorithm has the important advantage of being very simple. It has few parameters and we have shown how to set these parameters based on the input data.

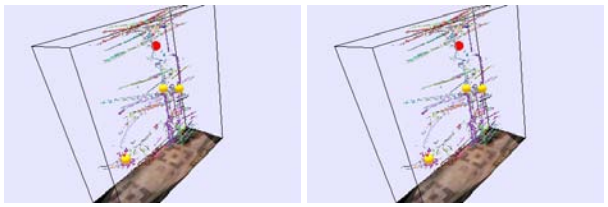
As a consequence of its simplicity, our algorithm has some limitations. For instance, the tracking algorithm exploits the fact that the motions of the blobs of interest is typically small with regard to the blobs' spatial extents. This allows simple correspondence of blobs by means of bounding circles. However, if the motion becomes larger, this simple tracking methodology will fail and more complex motion description or prediction will be necessary. Another weakness of the tracking algorithm is the somewhat limited supervision of the temporal evolution of the blobs. For instance when two blobs merge and, after a while, split again into two blobs, there is no way in the proposed algorithm to identify each one of them (or to make the correspondence between the blobs before and after the merging). For in-



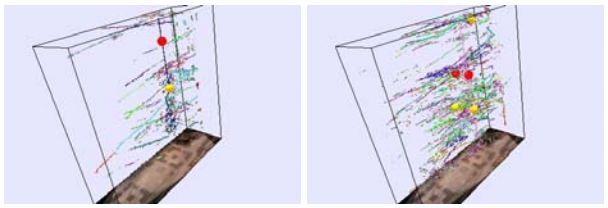
Sequence 1



Sequence 2 Sequence 3



Sequence 4 Sequence 5



Sequence 6 Sequence 7

Figure 13: 3D visualization of the blobs and events detected in each of the seven sequences. These results are from the experiment reported in Table 2. Here, each blob is represented by a single colored point, while warnings and alarms are represented by yellow and red spheres, respectively.

stance, this could be a problem if the owner of a luggage meets another person forming a unique blob. After a few moments, if the owner leaves the scene, the algorithm will not be able to identify the leaving blob (the owner or the visitor?). Blob correspondence could be implemented based on the color histogram of the individual blobs (and corresponding silhouettes) before their grouping; this would allow the determination of the corresponding persons after an eventual future splitting but at the price of a more complex algorithm.

References

[1] S. Calderara, R. Vezzani, A. Prati and R. Cucchiara, "Entry edge of field of view for multi-camera tracking in distributed video surveillance", In *Proc. of IEEE Conference on Advanced Video and Signal Based Surveillance*, pp. 93-98, Sept. 2005

[2] M. Dahmane and J. Meunier, "Real-Time Video Surveillance with Self-Organizing Maps", In *The 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, pp. 136-143, 2005.

[3] L.M. Fuentes and S.A. Velastin, "People Tracking in Surveillance Applications", In *Proc. of the 2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance (PETS2001)*, 2001

[4] `http://www.iomniscient.com`

[5] S. Khan and M. Shah, "Consistent labelling of tracked objects in multiple cameras with overlapping fields of view", In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, Issue 10, pp. 1355-1360, Oct. 2003

[6] S. Khan and M. Shah, "A multiview approach to tracking people in crowded scenes using a planar homography constraint", In *European Conference of Computer Vision*, May 2006

[7] A. Mittal and L. Davis, "Unified multi-camera detection and tracking using region-matching", In *Proc. of IEEE Workshop on Multi-Object Tracking*, pp. 3-10, July 2001

[8] J.W. Eaton, "GNU Octave Manual", *Network Theory Limited*, isbn : 0-9541617-2-6, 2002

[9] `http://www.intel.com/technology/computing/opencv/index.htm`

[10] `http://pets2006.net`

[11] R.Y. Tsai, "An efficient and accurate camera calibration technique for 3D machine vision", In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, FL, pp. 364-374, 1986

[12] M. Valera and S.A. Velastin, "Intelligent distributed surveillance systems: a review", In *IEEE Proceedings Vision, Image and Signal Processing*, Vol. 152, Issue 2, pp. 192-204, April 2005

[13] Z. Yue, S.K. Zhou and R. Chellappa, "Robust two-camera tracking using homography", In *Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 1-4, May 2004

Automatic Left Luggage Detection and Tracking Using Multi-Camera UKF

Jesús Martínez-del-Rincón, J. Elías Herrero-Jaraba, Jorge Raúl Gómez, Carlos Orrite-Uruñuela
Computer Vision Lab, Aragon Institute for Engineering Research, University of Zaragoza, Spain
{jesmar, jelias, jrg, corrite}@unizar.es

Abstract

In recent years, video-surveillance has undergone a vertiginous development due to the increasing concern about security and crime prevention. The wide use of video surveillance systems has caused a spectacular increase in the amount of data that has to be collected, monitored and processed. It is therefore crucial to support human operators with automatic surveillance applications that notify events potentially relevant to security. In this paper, we present a surveillance system which supports a human operator by automatically detecting abandoned objects. It consists of two major parts: a multi-camera tracking algorithm based on UKF and a blob object detection system which identifies abandoned and static objects. If an abandoned object is detected, an alarm event is triggered.

1. Introduction

Visual surveillance is currently one of the most active research topics in computer vision. The increasing concern about public safety and law enforcement has caused a great deal of growth in the number of surveillance cameras. Due to this fact, the necessity of automatic techniques which process and analyse human behaviours and activities is more evident each day.

Many publications about human behaviour analysis exist, but most of them are exclusively focused on people. Few tasks can be found about object-human interaction: In [1] the proposed system is able to detect if a person carries an object, and the system [2] tries to recognize theft. M. Spengler and B. Schiele propose an approach [3] for detecting abandoned objects and tracking people using the CONDENSATION algorithm in monocular sequences. A distributed surveillance system for the detection of abandoned objects in public environments is presented in [4], [5], [6] and [7]. In [8] and [9], a multi-camera surveillance and tracking system for monitoring airport activities is discussed.

In this work we present a method for detecting abandoned objects. The system is able to identify left luggage at the scene (suitcases, rucksacks, bags, etc.), identify the person who has abandoned the baggage and send an alarm if the object is abandoned for a period of time. The algorithm has been tested with the PETS 2006 database composed of sequences of video, in which people abandon luggage at a train station.

The present approach can be divided into two parts: a detection stage consisting of locating left luggage and the person who abandoned it; and a tracking stage which integrates the detection results of several cameras in order to track the person and the object. A piece of left luggage is a static object which fulfills certain requirements in the same way that a person is a dynamic object which fulfills others requirements. Thanks to trackers we can know their positions at each time step and monitor them accordingly. A brief scheme is shown in Figure 1.

The rest of this paper is organized as follows: in Section 2 we explain detection procedures to obtain the static objects and the motion objects in an image. In Section 3, we briefly present the required calibration of the cameras. In Section 4 we introduce the tracking algorithm which takes into account information provided by several cameras. Results are presented in Section 5 and conclusions in Section 6.

2. Object detection

2.1 Static object detection

We can define a static object as an object which has been abandoned at the scene and which does not move, but was not there at the beginning. We have developed a method based on a double background subtraction which is capable of detecting these kind of objects [10].

Long-term background represents a “clean” scene. All objects located in this background are not considered static objects: they are part of the scene. The long-term background is initialized with a temporal median filter of the initial frames and it is updated using a temporal median filter with a set of short-term backgrounds.

Short-term background shows static objects abandoned at the scene. This image is made up of the last background image updated with pieces of the current

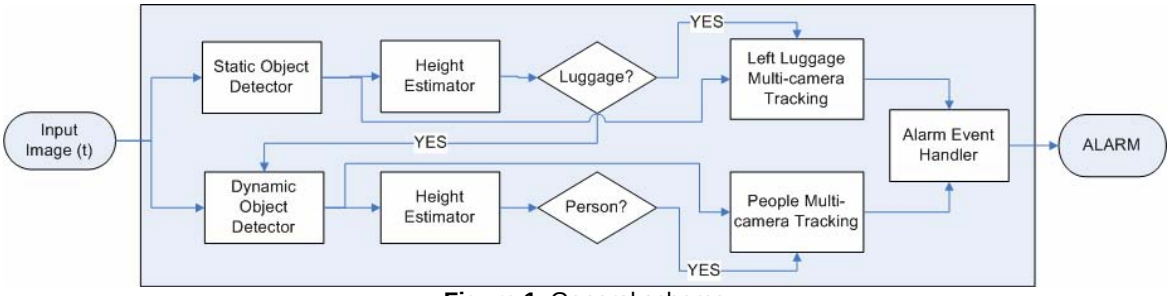


Figure 1. General scheme

image. These pieces are then selected by a static object binary mask. This mask contains new blobs detected at the scene (given by a subtraction between the current frame and last stored background) and they are not currently moving (given by the opposite of a subtraction between current frame and the previous frame). The intersection between both subtractions is made in the blob level, not in the pixel level, that is, if blobs from both subtractions touch each other, they will be included in the static mask, even though they do not completely fit in.

Once both backgrounds have been calculated, their subtraction is accumulated. When the accumulation image rises above a value corresponding to a fixed time, the blob is classified as static object.

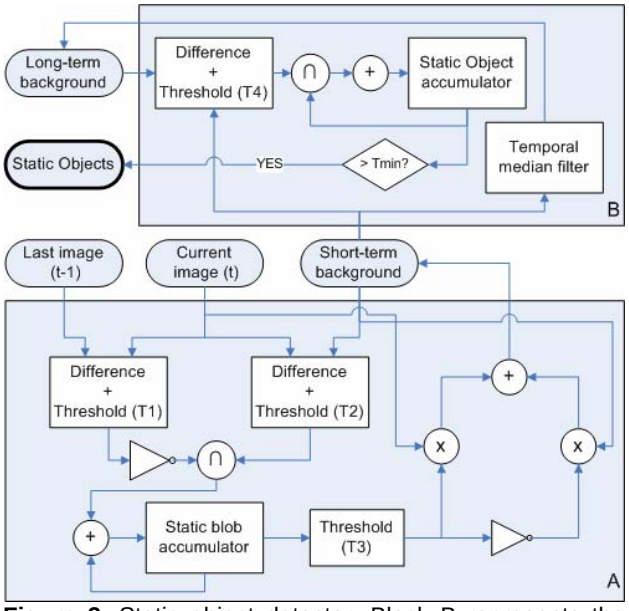


Figure 2. Static object detector. Block B represents the static object detection algorithm. Block A represents the short-term background creation algorithm.

2.1.1 Left luggage requirements. A static object will be considered as luggage if it fulfills several requirements. These requirements are a set of norms whose parameters can be changed in order to adapt the detection to different kinds of objects.

In our case we define a luggage item as a static object which has an area greater than a minimum area *A_min* (to eliminate noise), a maximum size equivalent to half the size of a person in the same point (the number of pixels of a person at this point will be obtained with the method explained in section 3.2) and a height-width ratio near one (*r_H/W* is the maximum error).

2.2 Dynamic object detection

Dynamic object detection algorithm is a very simple process consisting of a subtraction and a binarization between the current image and the long-term background. Short-term background is not used in the subtraction due to the fact that a person who waits a few seconds without moving will not appear in the resulting image.

Figure 3. Dynamic object detector.

2.2.1 Person requirements. We define a person as a dynamic blob with a height roughly equal to the number of pixels given by the height estimator (section 3.2). The other dynamic blobs will not be taken into account (*r_std_H* is the maximum error).

However, blobs can be fragmented. In such a case, a blob corresponding to a person does not fulfill the height condition. In order to avoid this effect, we apply an algorithm which groups blobs if they are in the bounding box corresponding to a person located in this position.

3. Multi-camera scene calibration

In order to integrate measurements from different cameras, a shared reference is needed. Our reference is a plan of the train station, and a homographic matrix [11] for each camera has been calculated. Thus, we can transform points from the image to the coordinate system of the plan.

When the measurements from the cameras have been projected onto the plan, the tracking algorithm can be applied. The use of a plan as reference has another implicit advantage. The motion of an object in the image presents a distortion due to the perspective of the camera. The homographic transformation corrects this effect simplifying the search of an adequate motion model.

3.1 Homographic transformation

Taking a minimum of four points we can establish the correspondence between the floor plane in the image and the plan of the train station. With this transformation, we can locate the position of blobs on the plan, assuming that the blob is in contact with the floor. One homographic matrix must be calculated for each camera.

3.2 Height estimator

As we have mentioned in the previous sections, we require a tool for obtaining the number of pixels which represents the average height of a person. Due to the perspective effect, this number is different depending on the location of the person in the image. We are able to ascertain this due to scene calibration [12].

First of all, we have to obtain a perpendicular plane of the floor which has been defined with the four points used to calculate the homographic matrices. For this purpose, we have to extract four points of the walls or any other vertical structure. Knowing both planes, we can calculate three vanishing points (2 horizontal ones and one vertical). These vanishing points permit us to project any point onto the coordinate axes, and elevate it vertically a number of pixels corresponding to the height in this point of the image. This number of pixels has been determined by a reference height in the image, that is, marking two points in the image, which are projected onto coordinate axes, and giving the real height in meters. Due to the fact that we do not know any real height in the training sequences, we will use a person of standard reference height, assuming everybody has the same height.

This methodology is able to return the head point (given the foot point), or return the height (given both points). We use the first application to determine the number of pixels which a person must have in this localization.

The height estimator algorithm is shown in Table 1 and Figure 4. Homogeneous coordinates are utilized to simplify the mathematical operations.

- *Calculate the directional vector of the line H2-H1*
  $v = \frac{y_{H2} - y_{H2}}{x_{H2} - x_{H1}}$
- *Estimate point H2 supposing the height of reference like 1,8 meters, and using the proportion between the number of pixels and the reference height in meters.*
- *Calculate the line L_H2_PFII=H2×PFII in homogeneous coordinates, that is:* $H2=[x_{H2}\ y_{H2}\ 1]'$
- *Calculate the line L_A_PFI=A×PFI*
- *Calculate the axis Y: L_H1_PFII=H1×PFII where H1 is the coordinate origin.*
- *Calculate the point A': A'=L_A_PFI× L_H1_PFII*
- *Calculate the line L_A'_PFIII =A' ×PFIII*
- *Calculate the point B': B'=L_A'_PFIII×L_H2_PFII*
- *Calculate the line L_A_PFIII =A ×PFIII*
- *Calculate the line L_B'_PFI =B' ×PFI*
- *Calculate the point B: B=L_B'_PFI×L_A_PFIII*

Table 1. Height estimation algorithm.

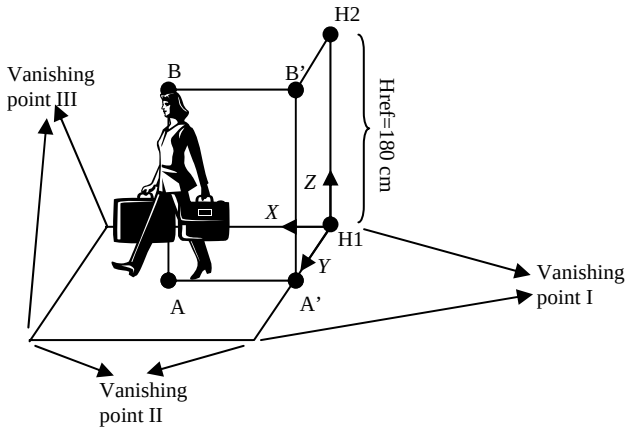


Figure 4. Height estimator.

4. Multi-camera tracking

Once static and dynamic objects have been detected in each camera, we project all the measurements onto the plan in order to have a shared reference space. Each blob is projected converting it in a single point, the point which is in contact with the floor (the lower point).

When these transformations have been made, the multi-camera tracking is applied: one tracker for the luggage item, and other for the owner. However, each tracker receives the measurement from a different source. While the static object tracker uses static object blobs like measurement, the dynamic object tracker uses dynamic object blobs.

The Unscented Kalman filter (UKF) [13, 14] is a very popular tracking algorithm which provides a way of

processing non-linear and non-Gaussian models. In this paper we propose a modified UKF to extend its application to multi-sensor scenes, thus improving the global result. The combination of several independent sensors increases the precision and robustness of our tracking system, since it makes it possible to solve difficult situations, such as occlusions or noise. Multi-camera tracking systems have been exhaustively proposed in previous literature, for instance, in [8, 15, 16].

4.1 Multi camera UKF

In this section we present a modification of the UKF which combines several measurements, provided by different cameras, for each tracked object. Due to the use of several sensors as measurement sources, we will call the algorithm Multi-Camera Unscented Kalman Filter (MCUKF). This algorithm can be extended to different types of sensors.

The filter can be divided into 3 stages: state prediction, measurement prediction, and estimation. This process can be shown in the following scheme (Figure 5). An external matching process must be used in order to make correspondences between trackers and measurements.

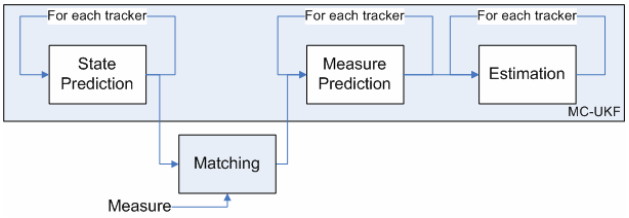


Figure 5. MCUKF algorithm.

4.1.1 State prediction. In the prediction stage, the tracker is initialized with the last estimation done in the previous time step. Hence, knowing the previous state $\hat{\mathrm{x}}_{k-1}$, with $e \times 1$ components, and its state covariance $\hat{\mathrm{P}}_{k-1}$, with $e \times e$ components, both the extended covariance $\hat{\mathrm{P}}^{a}_{k-1}$ and state $\hat{\mathrm{x}}^{a}_{k-1}$ can be obtained concatenating the previous parameter and the state noise. This is the first modification with respect to UKF algorithm, in which both state and measurement noises are used. The measurement noise will be used in the measurement prediction stage.

$$\hat{\mathrm{x}}^{a}_{k-1} = \left[\hat{\mathrm{x}}^{\mathrm{T}}_{k-1} \quad E\{\mathrm{v}_k\}\right]^{\mathrm{T}} \text{ with } E\{\mathrm{v}_k\} = [0 0 \ldots 0]^{\mathrm{T}}$$

$$\hat{\mathrm{P}}^{a}_{k-1} = \begin{bmatrix} \hat{\mathrm{P}}_{k-1} & 0 \\ 0 & \mathrm{R}^{v} \end{bmatrix} \text{ where } \mathrm{R}^{v} \text{ is the state noise matrix.}$$

This approach uses control points, called "sigma points", which enable the use of non-linear dynamic models. The number of sigma points is *2n + 1*, where *n* is the length of the extended state.

Following the classical equation of the Unscented Transform, the first sigma point corresponds with the previous frame estimation, the next *n*-th sigma points are the previous estimation plus each column of the previous estimation matrix, and the last *n*-th points are the previous estimation minus the same columns.

$$X^{a}_{k-1} = \left[\hat{\mathrm{x}}^{a}_{k-1} \quad \hat{\mathrm{x}}^{a}_{k-1} + \sqrt{(n+\lambda)\hat{\mathrm{P}}^{a}_{k-1}} \quad \hat{\mathrm{x}}^{a}_{k-1} - \sqrt{(n+\lambda)\hat{\mathrm{P}}^{a}_{k-1}}\right]$$

The components of these sigma points can be divided into two groups: $X_{k-1}{}^{x}$ derived from the state *x* and $X_{k-1}{}^{v}$ from the state noise *v*.

The weights assigned to each sigma point are calculated in the same way as in the unscented transformation. Therefore, the weight 0 will be different to obtain the weights applied to mean $W_{\mathrm{i}}^{(\mathrm{m})}$ or the weights applied to covariance $W_{\mathrm{i}}^{(\mathrm{c})}$.

$$W_0^{(m)} = \lambda/(n+\lambda)$$

$$W_0^{(c)} = \lambda/(n+\lambda) + (1+\alpha^2+\beta)$$

$$W_i^{(m)} = W_i^{(c)} = 1/[2(n+\lambda)] \qquad i = 1, 2, \ldots n$$

where $\lambda = \alpha^2(n+\kappa) - n$ is a scale parameter. Constant α involves the spread of sigma point around the mean $\bar{x}$ which has a small positive value (usually $1 > \alpha > 10^{-4}$). Constant κ is a secondary scaling parameter, usually with values between *0* and *3-n*. Finally, β is used to incorporate a previous knowledge of the distribution of *x*.

In order to predict the sigma points in the k-th instant, knowing the previous points, the transition matrix *F* is firstly required.

$$\hat{\mathrm{x}}_{\mathrm{k|k\text{-}1}} = \mathrm{F} \cdot \hat{\mathrm{x}}_{\mathrm{k\text{-}1}}$$

Then, sigma points in the next time step can be calculated as follows:

$$X^{x}_{k|k-1} = \mathrm{F} \cdot X^{x}_{k-1} + X^{v}_{k-1}$$

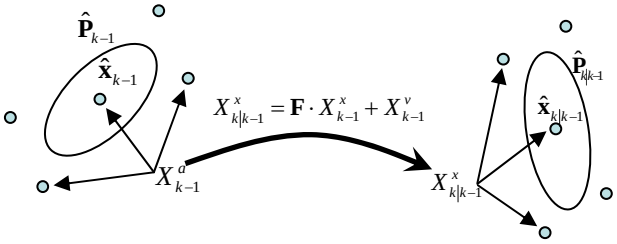


Figure 6. State prediction of mean and sigma points.

With these points and their weights, the predicted mean and covariance are given by

$$\hat{\mathrm{x}}_{\mathrm{k|k\text{-}1}} = \sum_{i=0}^{2n} W_i^{(m)} X^{x}_{i,k|k-1}$$

$$\hat{\mathrm{P}}_{\mathrm{k|k\text{-}1}} = \sum_{i=0}^{2n} W_i^{(c)} \left[X^{x}_{i,k|k-1} - \hat{\mathrm{x}}_{\mathrm{k|k\text{-}1}} \right] \left[X^{x}_{i,k|k-1} - \hat{\mathrm{x}}_{\mathrm{k|k\text{-}1}} \right]^{\mathrm{T}}$$

A graphic representation of this process is depicted in Figure 6.

4.1.2 Measurement prediction stage. The second contribution to original UKF consists in obtaining the measurement prediction taking into account measurements and measurement noises of each camera. In the measurement prediction stage, the first step consists of calculating the state predictions, and the measurement-tracker matching. Next, using the predictions and the measurement noise, both the extended measurement prediction state $\hat{\mathrm{x}}'^{a}_{k-1}$ and covariance $\hat{\mathrm{P}}'^{a}_{k-1}$ can be calculated. The concatenated measurement noise matrix R^{n} is built from measurement noise matrices of each camera R^{n}_{i} with $r \times r$ components, being $i = 1,2,...,S$.

$$\hat{\mathrm{x}}'^{a}_{k} = \left[\hat{\mathrm{x}}_{k|k-1} \; 0 \; 0 \; ... \right]^{\mathrm{T}} \quad \hat{\mathrm{P}}'^{a}_{k} = \begin{bmatrix} \hat{\mathrm{P}}_{k|k-1} & 0 \\ 0 & \mathrm{R}^{n} \end{bmatrix} \quad \mathrm{R}^{n} = \begin{bmatrix} \mathrm{R}^{n}_{1} & 0 & \\ 0 & \mathrm{R}^{n}_{2} & \\ & & \ddots \end{bmatrix}$$

In such a case, a tracker with *S* measurements, one of each camera, will have a state vector with $n' = r \cdot S + e$ components, and $2(r \cdot S + e) + 1$ sigma points.

$$X'^{a}_{k-1} = \left[\hat{\mathrm{x}}'^{a}_{k-1} \quad \hat{\mathrm{x}}'^{a}_{k-1} + \sqrt{(n+\lambda)\hat{\mathrm{P}}'^{a}_{k-1}} \quad \hat{\mathrm{x}}'^{a}_{k-1} - \sqrt{(n+\lambda)\hat{\mathrm{P}}'^{a}_{k-1}} \right]$$

$$W'^{(m)}_{0} = \lambda' / (n' + \lambda')$$

$$W'^{(c)}_{0} = \lambda' / (n' + \lambda') + (1 + \alpha'^{2} + \beta')$$

$$W'^{(m)}_{i} = W'^{(c)}_{i} = 1/[2(n' + \lambda')], \qquad i = 1,2,...n'$$

The sigma point components can be divided into components derived from the state X'^{x}_{k} and components derived from the measure noise X'^{n}_{k}, which can be separated again, according to its measure $i = 1,2,...,S$:

$$X'^{n(1)}_{k}, X'^{n(2)}_{k}, ..., X'^{n(S)}_{k}$$

The measurement matrix H, which converts state coordinates into measurement coordinates, is applied to obtain the measurement prediction sigma points $Y^{(s)}_{k|k-1}$ from sigma points for each camera.

$$Y^{(s)}_{k|k-1} = \mathrm{H} \cdot X'^{x}_{k} + X'^{n(s)}_{k}, \quad s = 1,2,...,S$$

Using these *S* sets of sigma points, we can obtain, for each measurement, the measurement prediction, the covariance prediction and the measurement-state cross-covariance.

$$\hat{\mathrm{y}}^{(s)}_{\mathrm{k|k\text{-}1}} = \sum_{i=0}^{2n'} W'^{(m)}_{i} Y^{(s)}_{i,k|k-1}$$

$$\mathrm{P}^{(s)}_{\hat{y}_k \hat{y}_k} = \sum_{i=0}^{2n'} W'^{(c)}_{i} \left[Y^{(s)}_{i,k|k-1} - \hat{\mathrm{y}}^{(s)}_{\mathrm{k|k\text{-}1}} \right] \left[Y^{(s)}_{i,k|k-1} - \hat{\mathrm{y}}^{(s)}_{\mathrm{k|k\text{-}1}} \right]^{\mathrm{T}}$$

$$\mathrm{P}^{(s)}_{\hat{x}_k \hat{y}_k} = \sum_{i=0}^{2n'} W'^{(c)}_{i} \left[X^{(s)}_{i,k|k-1} - \hat{\mathrm{x}}_{\mathrm{k|k\text{-}1}} \right] \left[Y^{(s)}_{i,k|k-1} - \hat{\mathrm{y}}^{(s)}_{\mathrm{k|k\text{-}1}} \right]^{\mathrm{T}}$$

These equations are depicted in Figure 7, with a two camera example.

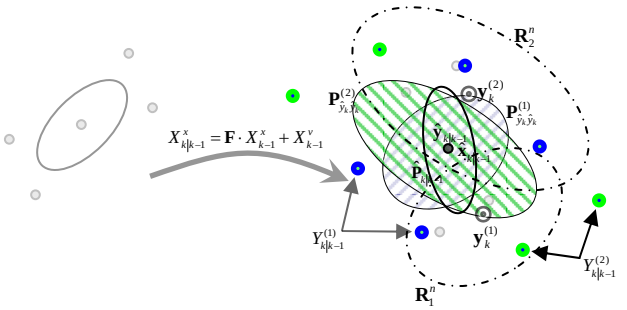


Figure 7. Hypothetic sigma point distribution for measurements of two different cameras. These points adjust their positions to represent the measurement covariance placed on the prediction.

4.1.3 Estimation stage. First, a gain matrix for each measurement associated to the tracker is calculated.

$$\mathrm{K}^{(s)}_{k} = \mathrm{P}^{(s)}_{\hat{x}_k \hat{y}_k} / \mathrm{P}^{(s)}_{\hat{y}_k \hat{y}_k}$$

After that, measurements from different cameras must be combined to obtain a shared estimation (Figure 8). Weights $\beta(s)$ play the role of combining the different measurements depending on their reliabilities. It is considered that weights are composed of 2 factors: the distance to the prediction, and the covariance of each measurement. Both factors are combined depending on the importance given to each one. Weights $\beta(s)$ can be interpreted as a priori probability of each measure.

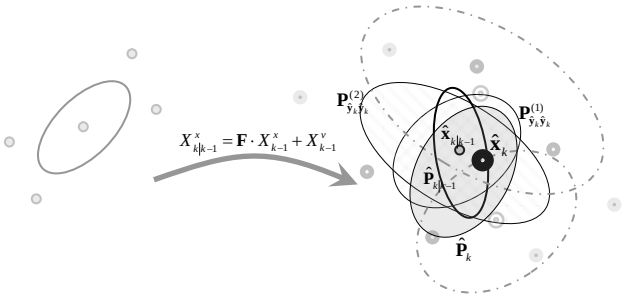


Figure 8. Graphic scheme which shows the estimation.

The set of weights will be normalized, since the sum of the weights must be 1. Mean and covariance estimations will be:

$$\hat{\mathrm{x}}_{k} = \hat{\mathrm{x}}_{k|k-1} + \sum_{s=1}^{S} \beta^{(s)} \mathrm{K}^{(s)}_{k} \left(\mathrm{y}^{(s)}_{k} - \hat{\mathrm{y}}^{(s)}_{k|k-1} \right)$$

$$\hat{\mathrm{P}}_{k} = \hat{\mathrm{P}}_{k|k-1} + \sum_{s=1}^{S} \beta^{(s)} \mathrm{K}^{(s)}_{k} \mathrm{P}^{(s)}_{\hat{y}_k \hat{y}_k} \left(\mathrm{K}^{(s)}_{k} \right)^{\mathrm{T}}$$

5. Results

Our system has been tested using several sequences of PETS 2006 database among which there are different situations, a diverse number of people and different types of luggage.

Once we know the positions of the person and the abandoned baggage in the plan's coordinates, we can measure the distance between both objects easily and act accordingly.

This application established two circular zones around the static object with a radius *a* and *b*. The luggage is attended to by the owner when they are within a distance *a* meters from the luggage. A luggage item is unattended when the owner is further than *b* meters from the luggage. In this moment, an alarm event is set up, and a time counter is activated. When an item of luggage has been left unattended by the owner for a period of *t* consecutive seconds (in which time the owner has not re-attended to the luggage and nor has the luggage been attended to by a second party), the alarm event is triggered. The distance between *a* and *b* is determined to be a warning zone where the luggage is neither attended to, nor left unattended. This zone is defined to separate the detection points of the two states, reducing uncertainties introduced due to calibration and detection errors in the sensor system. When the owner crosses the line at *a* meters, a warning event is set up to trigger the event after a time *t*.

Configurations parameters are a=2m, b=3m, t=30sg. Cameras 1, 3 and 4 have been utilized. Camera 2 has been rejected due to its poor resolution.

Object detection thresholds (Figures 2 & 3): *T1*=10, *T2* = *T4* = *T5*= 30, *T3*=230, *Tmin*=100, *A_min*=150, *r_H/W*= ±5%, *r_std_H* = ±25%.

We use a constant acceleration model and a motion dynamic which permits objects with variable trajectories. By introducing x and y velocities in the state vector we can solve occlusions between tracked objects

$$\hat{x}_k = [x \quad v_x \quad y \quad v_y] \qquad \hat{x}_{k|k-1} = F \cdot \hat{x}_{k-1}$$

and the dynamic matrix will be given by

$$\begin{bmatrix} x_k \\ v_k^x \\ y_k \\ v_k^y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ v_k^x \\ y_k \\ v_k^y \end{bmatrix} \quad R^v = \begin{bmatrix} t^3/3 & t^2/2 & 0 & 0 \\ t^2/2 & t & 0 & 0 \\ 0 & 0 & t^3/3 & t^2/2 \\ 0 & 0 & t^2/2 & t \end{bmatrix}$$

With this configuration parameters, the measurement matrix H is given by

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

We have used a simple matching algorithm: the filter selects the nearest measurement of each camera when there are several measurements for each camera. However, if the distance between the nearest measurement and the tracker is higher than 5.99 (chi square test) or two meters, anyone will be selected for this camera.

Results for sequences 1, 2, 3 and 7 can be observed in Table 2.

Sequence	Luggage item (x,y) [meters]	Warning trigger [sec (frame)]	Alarm trigger [sec (frame)]
S1-T1-C	**(0.161,-0.338)**	**113.0 (2825)**	**113.6 (2840)**
S2-T3-C	**(0.458,-0.431)**	**91.32 (2283)**	**91.84 (2296)**
S3-T7-A	**(1.041,-0.458)**	-	-
S7-T6-B	**(0.336,-0.391)**	**92.32 (2308)**	**93.96 (2349)**

Sequence	Distance Error [meters]	Time Error [sec (frames)]
S1-T1-C	**0.12202**	**0.06 (1.5)**
S2-T3-C	**0.14815**	**0.06 (1,5)**
S3-T7-A	**0.20297**	-
S7-T6-B	**0.18444**	**0.18 (4.5)**
Mean Error	**0.1644**	**0.1 (2.5)**

Table 2. Numerical results test sequences.

6. Conclusions

In this paper we have proposed a system capable of detecting abandoned or left objects. When this happens, the owner is found and tracked until the static object moves or the owner goes out of the observed region. If the owner abandons the object for a time greater than a specified time, an alarm event is triggered.

The present approach is not based on tracking all people at the scene, which does not constitute the main goal of this research and would demand unnecessary computational resources. Instead, we identify when an object has been unattended and proceed to track the person closest to this object.

We have developed a static object detector based on a double-background subtraction, which can detect left luggage or other static objects in any scene. Furthermore, we have presented a new tracking method which enables us to manage several sensors to track the same object.

Preliminary results applied to a real scenario monitored by four distributed cameras show an accurate detection method once threshold parameters have been tuned for the scenario under consideration. In addition, the tracking algorithm is robust to distracters, and allows for dealing with occlusions provided that the tracked object is isolated in at least one of the views.

Acknowledgments

This work is partially supported by grants TIC2003-08382-C05-05 from Spanish Ministry of Education.
J. Martínez-del-Rincón is supported by a FPI grant BES-2004-3741 from the Spanish Ministry of Education.

References

[1] I. Haritaoglu, R. Cutler, D. Harwood and L. S. Davis, "Backpack: Detection of People Carrying Objects Using Silhouettes". *IEEE International Conference on Computer Vision*, vol. 1, pp.102-107, Kerkyra, Greece, September 1999.

[3] M. Spengler and B. Schiele, "Automatic Detection and Tracking of Abandoned Objects". *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, Nice, October 2003.

[2] S. Hongeng and R. Nevatia. "Multi-agent event recognition", *IEEE International Conference on Computer Vision*, vol.2, pp.84-91, Vancouver, Canada, 2001.

[4] G. L. Foresti, L. Marcenaro and C. S. Regazzoni, "Automatic Detection and Indexing of Video-Event Shots for Surveillance Applications", *IEEE Transactions on Multimedia*, vol. 4(4), pp.459- 471, December 2002

[5] C. Sacchi and C. S. Regazzoni, "A Distributed Surveillance System for Detection of Abandoned Objects in Unmanned Railway Environments". *IEEE Transactions on Vehicular Technology*, vol. 49(5), pp.2013-2026, September 2000.

[6] G. L. Foresti, "Object Recognition and Tracking for Remote Video Surveillance", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9(7), pp.1045-1062, October 1999.

[7] E. Stringa, and C. S. Regazzoni, "Real-Time Video-Shot Detection for Scene Surveillance Applications". *IEEE Transactions on Image Processing*, vol. 9(1), pp.69-79, January 2000.

[8] D. Thirde, M. Borg, J. Ferryman, J.Aguilera, M. Kampel, and G. Fernandez, "Multi-Camera Tracking for Visual Surveillance Applications". *11th Computer Vision Winter Workshop 2006*, Czech Republic, February 6–8 *2006.*

[9] J. Aguilera, D. Thirde, M. Kampel, M. Borg, G. Fernandez, and J. Ferryman, "Visual Surveillance for Airport Monitoring Applications", *11th Computer Vision Winter Workshop 2006*, Czech Republic, February 6–8 *2006.*

[10] Elías Herrero, Carlos Orrite, and Jesús Senar, "Detected motion classification with a double-background and a neighborhood-based difference". *Pattern Recognition Letters*, vol.24, pp.2079-2092, 2003.

[11] RI. Hartley and A. Zisserman, "Multiple view geometry in computer vision", *Second edition Cambridge University press*, chapter 7.8, 2004.

[12] A. Criminisi, I. Reid and A. Zisserman, "Single View Metrology", *International Journal of Computer Vision*, vol. 40(2), pp. 123 – 148, November 2000.

[13] S. J. Julier, J. K. U. "A new extension of the kalman filter to nonlinear systems". *Proc. of AeroSense: The 11th Int. Symp. On Aerospace/Defence Sensing Simulation and Controls*, 1997.

[14] E. A. Wan, R. v. d. M., "The Unscented Kalman Filter". chapter 7, 2001.

[15] J. Black and T. Ellis, "Multi Camera Image Measurement and Correspondence". *Measurement - Journal of the International Measurement Confederation*, 35(1) Elsevier Science, pp. 61-71, 2002.

[16] M. Meyer, T. Ohmacht, R. Bosch and M. Hotter, "Video Surveillance Applications using Multiple Views of a Scene", *32nd Annual 1998 International Carnahan Conference on Security Technology Proceedings*, pp. 216-219, Alexandria (VA, USA), Oct 12-14 1998.

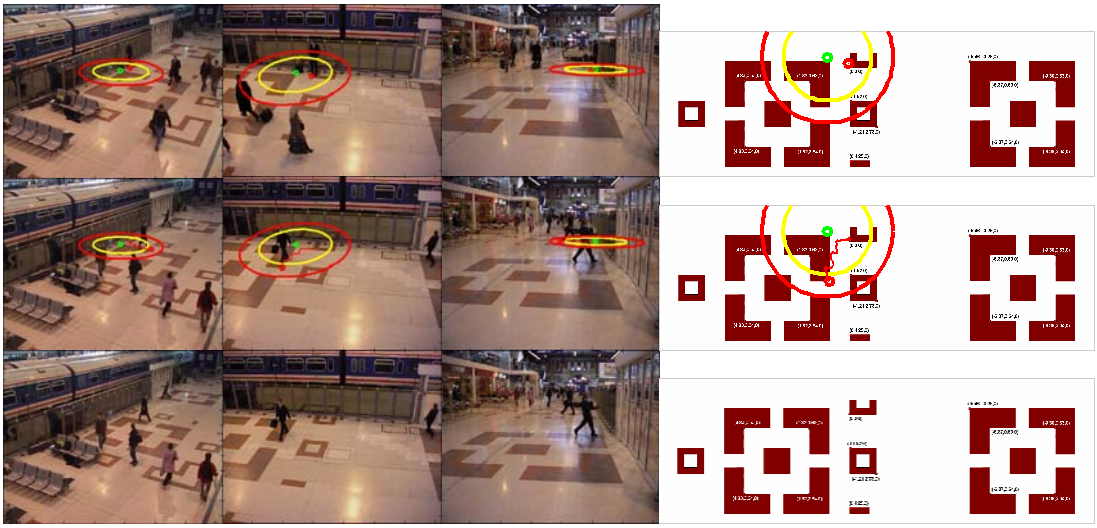


Figure 9. Results for sequence 3 (S3-T7-A).

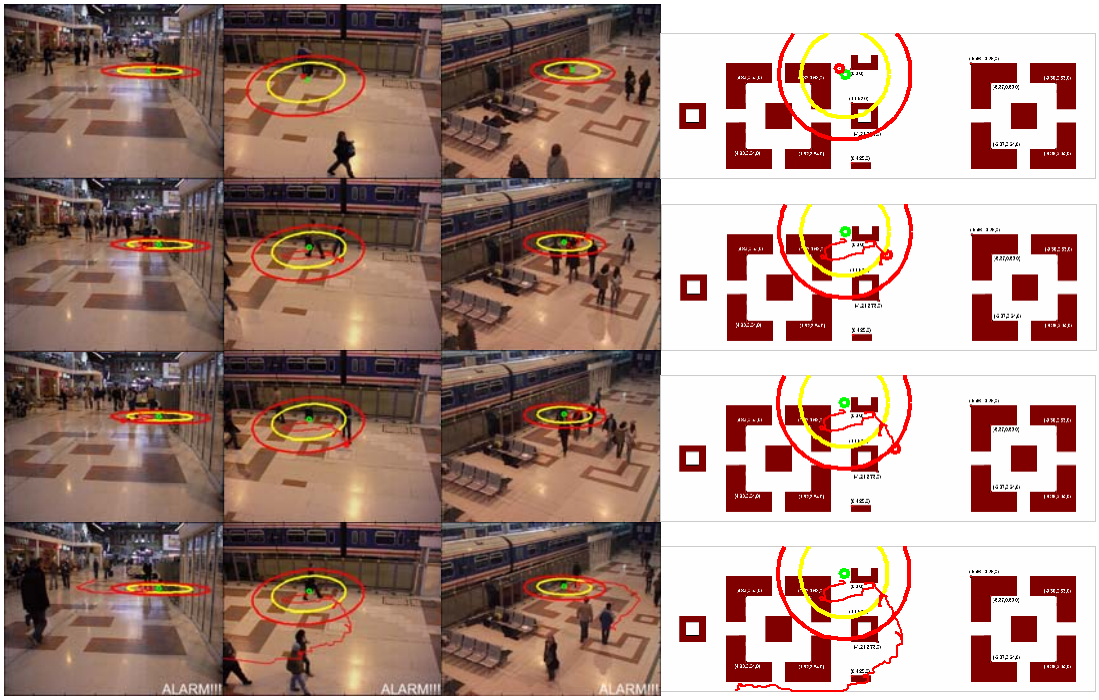


Figure 10. Results for sequence 7 (S7-T6-B).

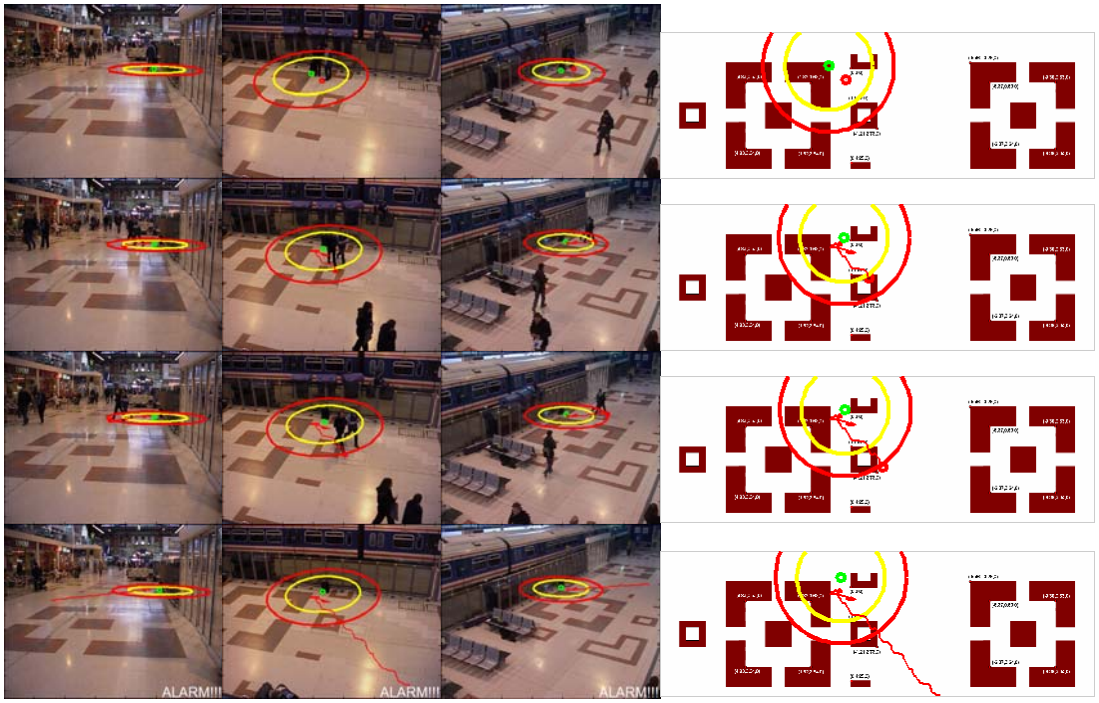


Figure 11. Results for sequence 2 (S2-T3-C).

Multi-View Detection and Tracking of Travelers and Luggage in Mass Transit Environments

N. Krahnstoever P. Tu T. Sebastian A. Perera R. Collins

General Electric Global Research, One Research Circle, Niskayuna, NY 12309
E-Mail: {krahnsto|tu|sebastia|perera|collins}@crd.ge.com

Abstract

A real-time multi-view surveillance system for detecting and tracking people, luggage and related events in mass-transit environments is presented. The system operates in a calibrated camera environment and is designed for ef cienc y and scalability. The scalability is achieved by separating target detection in the imagery and target tracking in the calibrated ground plane, which allows the use of advanced track formation and maintenance algorithms. An ef cient geometry-driven approach is used to detect targets (people and objects) in each camera view. Detected targets are integrated into a centralized ground-plane based tracking system. The complete system is evaluated using the PETS 2006 dataset, and the experiments demonstrate that the system is able to reliably detect unattended and abandoned luggage in challenging environments.

1 Introduction

Visual surveillance in mass transit environments can greatly aid the prevention, detection and investigation of crimes and accidents. Unfortunately, the traditional operator-based surveillance of such environments is labor-intensive due to the size of the sites and the number of people involved. Hence, surveillance in mass-transit environments is typically used only as a forensic tool rather than a preventive or interceptive tool. Automatic visual surveillance using computer vision techniques has the potential to provide continuous and proactive protection in mass transit environments. This, however, is not an easy task due to the inherently large scale of mass transit networks. For example a large metro surveillance system typically consists of thousands of cameras, and observes millions of travelers in a short period of time. In addition, an automated system typically has to deal with large crowds resulting in severe occlusion, poor lighting conditions, and views that are too narrow or too wide.

This paper focuses on luggage-related events such as abandoned and unattended luggage. This is an important problem for law enforcement of cials because unattended or abandoned luggage can be used hide bombs and chemical, biological or nuclear threats. The computer vision community has already presented numerous algorithms and systems for related tasks including detection of luggage [6], events such as luggage theft and abandonment [7, 14], violence [5], and other suspicious activities.

This paper makes two main contributions. First, it presents a detection and tracking framework that is particularly suited for operating in large scale environments. The scalability of the system is achieved by separating the tasks of local target detection, which is performed in each camera view, and tracking, which is performed centrally in the ground plane. Second, an ef cient person and luggage detection algorithm is presented. We take a model-based approach to explaining the foreground imagery similar to [8, 14]. Our approach differs from existing work in that it utilizes a computationally more ef cient target model and in that it utilizes a greedy search strategy rather than Monte Carlo methods to support real-time performance. Also, in contrast to [14], it performs detection of people and objects in a single uni ed framework.

The remainder of this paper is organized as follows. In Section 2 we outline the main components of our tracking system. In Section 3, we present our approach to person and luggage detection. This paper is aimed at solving the luggage event detection tasks in the PETS 2006 benchmark data and Section 4 outlines our reasoning framework that is used to solve this task. Results are presented and discussed in Sections 5 and 6. Section 7 concludes the paper.

2 Multi Camera Tracking

Many tracking systems, most notably template based approaches, separate the tracking problem into two tasks: track initialization followed by a track update step that involve local search in the prediction gate of the tracks. In contrast, many traditional tracking systems outside of the computer vision community, due to the nature of the sensors used, separate tracking problem into three steps: detection, data association and track state update. In such approaches, tracks only utilize the data that is associated to them to update their internal state. The robust detection of targets is necessary for such traditional tracking approaches to succeed. Our approach to target detection, and the recent progress in the real-time detection of faces [15] and people [4] as well as the robust detection of people in crowded scenes [11, 13, 16] have facilitated the use of such traditional tracking approaches. This approach is pursued in this paper, as well.

In the traditional tracking framework, the detection algorithm has to supply the tracking module with a list of detections at every frame. Each detection contains information about the class of the target (in our case 'adult', 'child' and 'luggage'), its location and location uncertainty in the image. In addition, we assume that the detector provides suf ci ent information to (i) project the location information into the ground plane and (ii) to recover information about the physical height and width of targets. An approach to do this is outlined in [10], where a person detector supplies bounding boxes for people in the image, based on which foot and head location estimates are obtained. This information, in conjunction with the metric projection matrix of the camera is used to project the location onto the ground plane as well as to obtain the physical dimensions for each detection. This approach works well when people occur in isolation, but breaks down in situations where people occur in groups or are only partially visible in the image (due to the image borders). This is a major challenge in many practical applications. Hence, we utilize a different approach, which is outlined in the next section.

At every step, detections are projected into the ground plane and supplied to a centralized tracker that processes the locations of these detections from all camera views. At this stage the tracking of extended targets in the imagery has been reduced to tracking 2D point locations in the ground plane, which can be performed very ef ciently . The system assumes that the central tracker operates on a physically separate processing node. Hence, it may receive detections that are out of order from the different camera views due to network delays. Therefore, all detections are time stamped according to a synchronous clock, buffered and time re-ordered by the central tracker before processing. The sets of time ordered detections are processed by the following standard stages:

Track Prediction - The location for each track is predicted forward in time according to its current state and its dynamical model. The time stamp of the currently processed detection batch determines how far forward in time the prediction is performed.

Data Association - Each track in the set of currently active tracks is assigned to at most one detection using the Munkres algorithm [2]. The distance between tracks and detections is measured using the Mahalanobis distance where the covariance given by the sum of the current track gate, the uncertainty of the detection and a base uncertainty. The Munkres algorithm obtains the optimal assignment between tracks and detections under this distance measure. Tracks that are too far away from their assigned detections are considered to be non-associated. The success of this general nearest neighbor approach to tracking [1] depends on the performance of the detector and the clutter probability, but was shown to be suf cient for the provided benchmark data.

Track Update - After association, tracks are updated according to their assigned observations. If a track was assigned to any observation, the update step is performed with a virtual observation that has in nite uncertainty, amounting to an update that does not correct the predicted location but increases the uncertainty of the state estimate. Track states are maintained using extended Kalman l-ters. As dynamical models we utilize a constant velocity turn model described in [1].

Track Maintenance - Tracks are marked for deletion if the state uncertainty becomes too large, if the track goes out of view (of the entire camera network) or if it has not been associated with a detection within a certain time window. Upon deletion, a determination is made as to whether the track was a false alarm, based on several criteria involving the lifetime of the track and its motion pattern.

Track Formation - Each detection that is not associ-

ated with an existing track leads to the formation of a new track if its spatial dimensions (height, width) passes a number of tests designed to limit the number of spurious tracks that are created. Steps are in place for detecting and patching 'ghosts' in the foreground image, created by targets that have been assimilated or initialized into the background.

The above described tracking system constitutes a generalized nearest neighbor tracker [1]. It is computationally very efficient and hence suited for tracking a large number of targets in many camera views simultaneously. The system has proven to be adequate for the task of detecting abandoned luggage, which requires accurate tracking of the people interacting with luggage, but not necessarily for accurately tracking people that move in groups or crowds. If accurate and persistent target tracking (in dense groups and crowds) is desired, more sophisticated and computationally more expensive approaches such as JPDAF [12], MHT [3] or Bayesian multi-target trackers [8] should be employed.

Figure 1 shows the tracker in operation on sequence S1 of the PETS 2006 dataset. The tracker deals well with isolated targets as well as with crowds, but if the scene becomes too crowded, the number of targets might be estimated incorrectly. This has little impact on the luggage detection task that is the focus of this work. It should be stressed that the availability of multiple calibrated camera views helps greatly in constraining the target tracking process in this work.

3 Person and Object Detection

Our approach to detecting people and objects (henceforth collectively called targets) operates on the foreground image, in which each pixel denotes the discretized probability of seeing foreground. The algorithm determines at every frame an estimate of the most likely configuration of targets that could have generated the given foreground image. We define $\mathbf{X} = \{\mathbf{X}_j = (x_j, y_j, c_j), j = 0, \ldots, N_t\}$ to be a configuration of targets with ground plane locations (x_j, y_j) and target class labels $c_j \in C$. Each target class in C is associated with size and height information. In addition, we assume that each target is composed of several *parts*. Let $O_{c,k}$ denote the part k of target class c. When a target configuration $\mathbf{X}$ is projected into the image, a label image $O[i] = (c_i, k_i)$ can be generated where at each image location i part k_i of class c_i is visible. If no part is visible we assume $O[i] = \mathrm{BG}$, a special background label. We now define the probability of the foreground image F at time t as

$$p(F_t|\mathbf{X}) = \prod_{\text{valid } (c,k)} \left[\prod_{\{i|\mathrm{O}[i]=(c,k)\}} p(F_t[i]|O[i]) \right] \prod_{\{i|i\in\mathrm{BG}\}} p(F_t[i]|i \in \mathrm{BG}), \quad (1)$$

where $F_t[i]$ is a discretized probability of seeing foreground at image location i. Note that the above product has as many $p(F_t|...)$ terms as we have pixels in the foreground image F_t. The above probability can be rewritten as a log likelihood

$$L(F_t|\mathbf{X}) = \sum_{\text{valid } (c,k)} \left[\sum_{\{i|\mathrm{O}[i]=(c,k)\}} \log \frac{p(F_t[i]|O[i])}{p(F_t[i]|i \in \mathrm{BG})} \right] \sum_{\text{all } i} \log p(F_t[i]|i \in \mathrm{BG}), \quad (2)$$

where the background BG and hence the last term in the above equation does not depend on $\mathbf{X}$, unlike $O[i]$, which is dependent on $\mathbf{X}$. Hence the above term can be simplified to

$$L(F_t|\mathbf{X}) = \sum_{\{i|O[i]\neq\mathrm{BG}\}} h_{O[i]}(F_t[i]). \quad (3)$$

where $h_{(c,k)}(p)$ is a histogram of likelihood ratios for part k of target class c given foreground pixel probabilities p. The goal of the person and object detection task is to find the most likely target configuration $\mathbf{X}$ that maximizes Eq. (3). However, to allow real-time execution, several simplifications and optimizations are made.

We now specify the generative model that produces the label images $O[i]$, given a configuration $\mathbf{X}$. We assume that people and objects are vertical ellipsoids with class specific heights and widths located on the ground plane. The bounding boxes of these ellipsoids constitute a simple approximation of the target silhouettes in the image. Depending on the class label, bounding boxes are subdivided into one or several parts. For example, a person silhouette

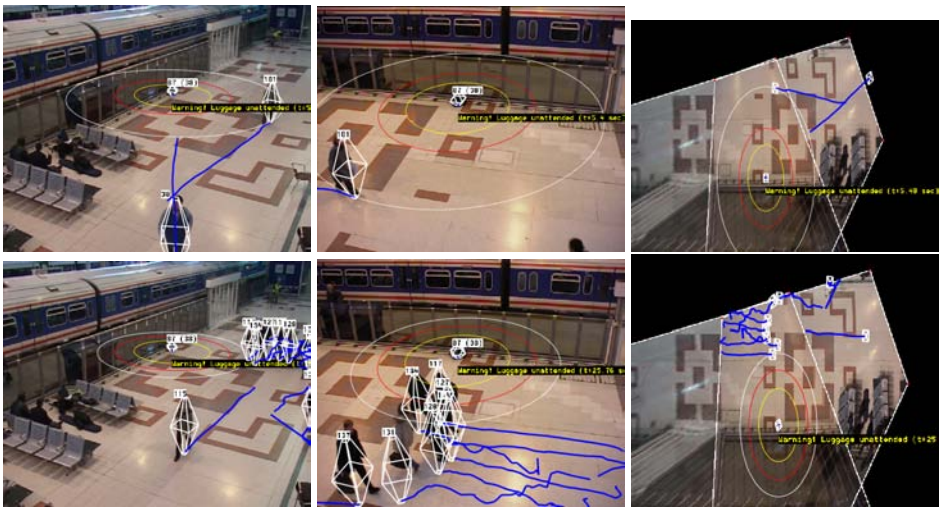


Figure 1: **Tracker.** *Top Left and middle: Two frames from sequence S1 showing two people and one luggage track. Right: Top view mosaic of the two camera views. The mosaic is generated by assuming all image content is located on the ground plane. The top view shows the two person tracks and the luggage location. Tracks are assigned identi er s. In addition, the luggage track (id=87) is associated with its owner (id=38). The bottom images shows tracking in a crowd where the number of targets have been miscalculated by the system.*

is split into three equal parts with separate body part labels assigned to the top, middle and bottom third of the bounding box.

We assume that targets can only be located at discrete ground plane locations in the camera view, which allows us to precompute the bounding boxes required for the evaluation of (3). Even with the above approximations, the maximum of (3) can not be found using exhaustive search since it is exponentially expensive in the number of visible targets, which is furthermore unknown. One could employ MCMC approaches such as [8, 16], but since these are computationally too expensive for real-time applications, we adopt a greedy approximation in this paper. We start with the empty scene and iteratively add targets to the ground plane in a way that yields the greatest increase in the data likelihood at every step. To achieve real-time performance, we make the following simplifying assumptions. First, we assume that the class and part speci c histograms $h_{(c,k)}(p)$ can be approximated as $h(p) + o_{(c,k)}$, i.e., a general model plus an offset. This, along with the previously described use of rectangular model projections, allows us to rewrite (3) as sums evaluated in a single histogram image

$$L(F_t|\mathbf{X}) \approx \sum_j \sum_{\text{parts } k \text{ of } \mathbf{X}_j} (\text{IIS}(H, B^k(\mathbf{X}_j)) + |B^k| o_{(c,k)}), \tag{4}$$

where $B^k(\mathbf{X}_j)$ is the bounding box of part k of target $\mathbf{X}_j$, $\text{IIS}(H, B)$ the integral image of H evaluated over the bounding box B, $H[i] = h(F_t[i])$, and $|B|$ the area of B. Since the above equation, unlike Eq. (3), disregards depth ordering and occlusion we enforce spatial and occlusion constraints by pruning the set of possible ground plane target locations after each greedy target location choice has been made. The approach outlined here allows the precomputation of all relevant quantities. Hence, the bulk of the target detection algorithm is spent on selecting locally optimal targets from the set of possible ground locations followed by a spatial pruning of non-selected neighboring targets that are affected by the local choice. Further computational ef cienc y is achieved by immediately discarding target choices with negative likelihood from the set of possible choices. As a consequence of the above

approximations, the average target class and location hypothesis evaluation takes a little more than four memory lookups and four additions.

Figure 2 shows some example person and object detection results using the described system.

3.1 Re nement

After the initial set of people and object detections have been obtained, the ground plane locations of person detections are re ned based on a more detailed analysis of the foreground and image content. We search for the head location of people in the scene, using an approach that is similar to the one described in [10]. If no such re ned location can be found, the original detection location is used.

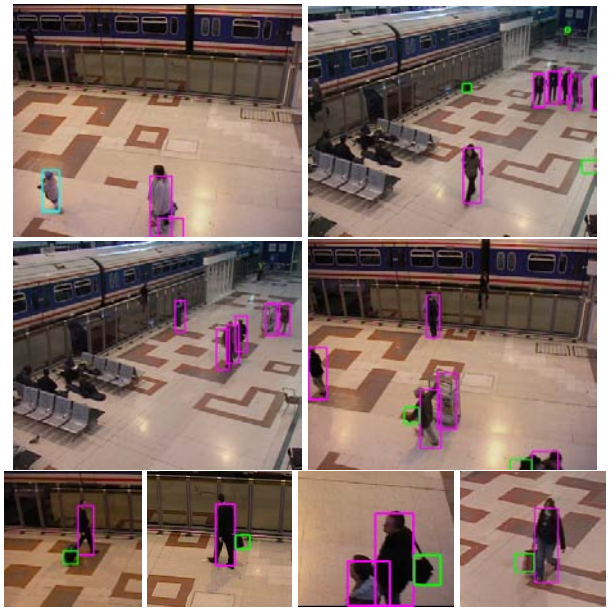


Figure 2: **Person and Luggage Detection.** *These frames show detections of people (red), a child (cyan) and luggage (green). The algorithm is designed for ef ciency rather than accuracy and spurious detections occur occasionally. However, these are handled through the use of a multi-view tracker and the track-formation process. The bottom four images show examples of successful detection of luggage even when carried by people.*

4 Luggage Event Detection

The system outlined in the previous two sections provides detections and tracks of people and luggage in all available camera views. Our ground plane-based centralized tracker operates in the fully calibrated camera views. This makes it straightforward to reason about spatial properties of tracks. Furthermore, the use of multiple views ensures that luggage that might be occluded in one view, is still detected and tracked by the system.

Every new luggage track is associated with an owner, which is de ned to be the person whose track has the shortest ground plane distance from the luggage. The system does not allow spontaneous discovery of abandoned luggage. If no person is present within a radius of $r_o = 1\mathrm{m}$, the luggage track is deemed invalid. This rule prevents the creation of spurious tracks. For example, it lters out false alarms caused by heads of tall people, since the projection of a person's head onto the ground plane is far away from the projection of that person's feet. A track linking stage [9] ensures that the identity of tracks are preserved for long periods of time such that the system can robustly reason over the spatiotemporal rules of unattended and abandoned luggage.

The luggage event detection system is triggered when a luggage track becomes stationary, which is de ned by the location covariance over a time window of $\tau_s = 3\mathrm{s}$ falling below a threshold of $r_s = 0.2\mathrm{m}$. The event detection system maintains for each luggage track the ground plane distance to its owner as well as the distance to the closest person that is not the owner, if such a person exists. As speci ed by the PETS rules, the system raises a warning if owners are more than $a = 2.0\mathrm{m}$ away from their luggage. It raises a second warning when the owner is at a distance of more than $b = 3.0\mathrm{m}$, by which time the system considers the luggage as being *unattended*. The owner has to come back to within radius a of the luggage to be considered attending his luggage again. If a piece of luggage is unattended for more than $\tau_u = 30\mathrm{s}$, the luggage is considered *abandoned* and the system raises an alarm.

Luggage pickup, by an owner or otherwise, is considered to have occurred when a luggage track disappears or becomes non-stationary when it is within a distance of $r_p = 1.0\mathrm{m}$ of a person. If the owner of the luggage is not within a radius of r_p during pickup, the person closest to

the luggage is considered to be the one that picked up the luggage. If somebody other than the owner picks up the luggage, a theft alarm is raised.

5 Results and Discussion

The surveillance system described in the previous sections was evaluated on the PETS 2006 datasets. This dataset contains sequences taken from a moderately busy railway station with people walking in isolation or as part of larger groups. Several actors perform luggage drop-off and pick-up scenarios with different levels of complexity. The enacted events are performed in front of a tinted glass wall that separates the main terminal from a train boarding platform. Since the focus of this study is to detect left luggage events (and not person detection and tracking through tinted walls), all detections that occur in ground plane locations beyond the wall are suppressed by the system.

For this paper we processed two out of four available views (see Figure 2) and focused on sequences S1-S4 and S6 that featured events involving *small* luggage items. The two other views were not considered for processing since one view was shot through a glass wall and the other had a viewing geometry that placed the horizon in the middle of the frame, which makes activity in the far eld dif cult to interpret from clutter in the near eld. The system was run on MPEG4 video streams obtained from transcoding the JPEG images that make up the PETS 2006 dataset. The algorithm processed two 720×576 video streams at a rate of 15 frames per seconds on a 3GHz single core Pentium 4 platform.

Although calibration information was provided as part of the PETS 2006 dataset, the calibration was repeated using a combination of the autocalibration approach described in [10] and information from ground plane-based scene landmarks, that were supplied to the system interactively. The calibration was considered accurate enough without compensating for lens distortion.

Three classes of targets were processed by the system: two classes of person targets ('adult' and 'child') and one class of 'luggage'. The geometric properties of the three target classes are listed in Table 1. At every frame the system evaluates the likelihood of approximately 161000 person and luggage locations in camera 4 and 82000 locations in camera 3.

Class	Diameter	Height	Min Img Width	Min Img Height
Adult	0.5 m	1.8 m	20 pix	20 pix
Child	0.4 m	1.3 m	20 pix	20 pix
Luggage	0.5 m	0.5 m	25 pix	25 pix

Table 1: **Target Classes.** *This table lists the three target classes processed by the system. This study focuses on the detection of small luggage items with a size of around* 0.5 *m.*

The processed sequences contained several scenarios and challenges. Sequence S1 features an "abandoned luggage event" where a person drops off a backpack in frame 1875 and walks away in frame 2059. Challenges include a sitting man that leaves around frame 609 and a chair that gets moved at the border of the image at frame 2628. The presented system raises an unattended luggage warning at frame 2088 and raises an abandoned luggage event at frame 2854 (31.8 seconds after the man walks away). See Table 2 for details. The sitting man and the moved chair are successfully handled by the 'ghost' detection algorithm.

Sequence S2 also features an "abandoned luggage event" where two men meet in frame 904. One person carries a suitcase, which he sets down in frame 1218. The two men start to leave around frame 1512 and leave the luggage behind. In the meantime, a worker directly behind the men, on the other side of the tinted wall, moves three garbage bins away with a small tractor. The proposed system raises an unattended luggage warning at frame 1542 and an abandoned luggage alarm at frame 2308 (31.84 seconds after the men leave). The system has no difficulty in detecting the dropped off luggage due to the use of multiple views. The garbage removal in the background causes severe temporary clutter in one camera view, but the abandoned luggage remains visible in the second and the tracker is continuously locked on the luggage.

Sequence S3 features a man with a carry-on stopping in the scene at frame 877 and leaving around frame 1316. The carry-on gets detected by the system, but no alarm is raised since the luggage leaves the scene with the owner.

Sequence S4 features a man with a carry-on stopping in

the scene (at frame 779). A second man joins him at frame 1249. Distracted, the first man leaves in frame 1802 and forgets his carry-on next to the second man. The proposed system correctly detects this sequence of events. The first man is correctly associated as the owner of the luggage before the arrival of the second. When the two men are talking, the detection system triggers only occasionally on the luggage that sits between the two men. The track linking component correctly maintains a long duration track on the luggage and the abandonment alarm is raised in frame 2611 (32.36 seconds after the owner leaves). See Figure 3 for images of the scene.

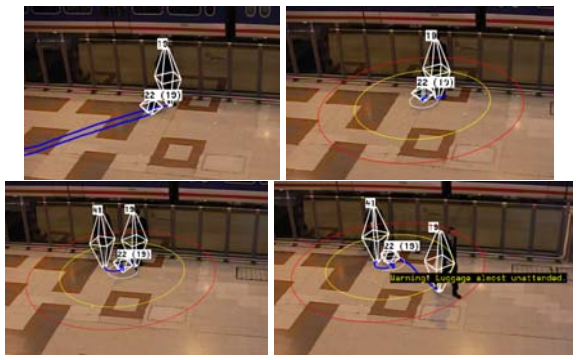


Figure 3: **Events in Sequence S4.** *Top-Left: The first man arrives with a carry-on. Note that both the man as well as the carry-on are tracked by the system. Top-Right: The man and his luggage are now stationary. Bottom-Left: A second man has arrived. Bottom-Right: The owner of the luggage is leaving without the luggage.*

Sequence S6 features two men somewhat covertly dropping of a backpack in frame 1583 and leaving in frame 1637. The system raises an alarm in frame 2455, 32.72 seconds after they leave the backpack. A man sitting on a bench, reading a newspaper also triggers an abandoned luggage alarm in frame 1336, which is an initialization phenomenon. The man never leaves his seat and is hence not detected and tracked as a person. However, arm movements cause spurious object detections that get associated with a person in the background, who subsequently leaves the area triggering the false alarm (see Figure 4).

The results of all sequences are tabulated in Table 2.

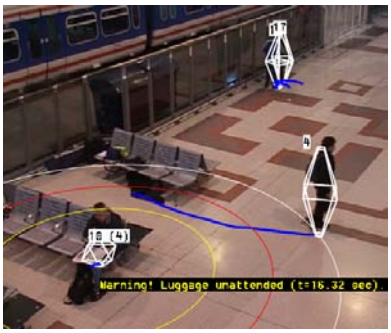


Figure 4: **False Alarm in S6.** *A man sitting on a bench reading a magazine through the entire duration of the sequence is never detected as a person. Movements of the man are detected as a foreground object, and are associated with a person standing behind him. When the standing person leaves, an alarm is raised.*

6 Discussion

The experiments described in the previous section show that robust multi-view detection and tracking of people and objects can successfully be used for detecting unattended and abandoned luggage. The proposed system has some success in tackling the challenges presented in PETS 2006 data. However, several issues should be raised regarding the problem of left luggage detection. First, much larger datasets are needed in order to accurately determine the performance characteristics of systems such as the one outlined here. The PETS 2006 dataset is only moderately complex when compared to, for example, a crowded gate in an airport. Second, the problem of long-duration stationary (e.g., sitting) people is one of the major challenges for detecting left luggage. If, for example, through adaptation or initialization on non-empty scenes, the system is unaware of the presence of such a person, reasoning about the relationship between this person and a potential piece of luggage becomes very difficult, as we observed in sequence S6. We believe that the use of strong non-adaptive but lighting invariant background models are necessary in order to make persistent foreground object detection possible in more challenging scenes. The system needs to have a strong notion of what the empty scene is, in order to make accurate judgments about subtle spatiotemporal events occurring in the non-empty scene.

Seq	Owner Leaves	Warning Raised	Alarm Raised	Comment
S1	2059	2088 (1.1 s)	2854 (31.8 s)	Person abandons backpack.
S2	1512	1542 (1.2 s)	2308 (31.8 s)	Two men leave suitcase.
S3	n.a.	n.a.	n.a.	A man attends his luggage and leaves with luggage.
S4	1802	1845 (1.7 s)	2611 (32.4 s)	Man forgets carry-on.
S6	n.a. 1637	 1689 (2.1 s)	1336 2455 (32.7 s)	False alarm. Two men leave backpack.

Table 2: **Luggage Events.** *This table lists the luggage events that were detected by the system. In columns 2-4 the r st values denote frame numbers. The values in parenthesis denotes the time that elapsed since the frame number in column 2.*

7 Conclusion

We described a multi-view surveillance system for detecting unattended and abandoned luggage. The system detects and tracks all people as well as objects in the scene and reasons about the spatiotemporal relationships between them. The system performs centralized tracking in a calibrated metric world coordinate system, which constrains the tracker and allows for accurate metrology regarding the size of targets and distances between them. When applied to the PETS 2006 video data, the system managed to detect all abandoned luggage events with only one false alarm, which, should be considered an initialization artifact of the supplied video sequence.

References

[1] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Publishers, 1999.

[2] F. Burgeois and J. C. Lassalle. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14:802–806, December 1971.

[3] I. J. Cox and S. L. Hingorani. An efficient implementation and evaluation of Reid's multiple hypothesis tracking algorithm for visual tracking. In *Intl. Conference on Pattern Recognition*, pages 138–150, 1994.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, pages 886–893, 2005.

[5] A. Datta, M. Shah, and N. D. V. Lobo. Person-on-person violence detection in video data. In *Proc. of the 16th International Conference on Pattern Recognition*, volume 1, pages 433–438, August 2002.

[6] I. Haritaoglu, R. Cutler, D. Harwood, and L. S. Davis. Backpack: Detection of people carrying objects using silhouettes. In *In Proc. IEEE International Conference on Computer Vision*, page 102107, 1999.

[7] S. Hongeng, R. Nevatia, and F. Bremond. Video-based event recognition: activity representation and probabilistic recognition methods. *Computer Vision and Image Understanding*, 96:129162, 2004.

[8] M. Isard and J. MacCormick. BraMBLe: A bayesian multiple-blob tracker. In *IEEE Proc Int. Conf. Computer Vision*, volume 2, pages 34–41, 2001.

[9] R. Kaucic, A. G. A. Perera, G. Brooksby, J. P. Kaufhold, and A. Hoogs. A unified framework for tracking through occlusions and across sensor gaps. In *Proc. Intl. Conference on Computer Vision and Pattern Recognition*, 2005.

[10] N. Krahnstoever and P. Mendonca. Bayesian autocalibration for surveillance. In *Proc. of IEEE International Conference on Computer Vision (ICCV'05), Beijing, China*, October 2005.

[11] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *Proc. CVPR, San Diego, CA*, 2005.

[12] C. Rasmussen and G. Hager. Joint probabilistic techniques for tracking multi-part objects. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 16–21, 1998.

[13] J. Rittscher, P. Tu, and N. Krahnstoever. Simultaneous estimation of segmentation and shape. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 486 – 493, 2005.

[14] M. Spengler and B. Schiele. Automatic detection and tracking of abandoned objects. In *VSPETS*, 2003.

[15] P. Viola and M. Jones. Robust real-time face detection. In *International Conference on Computer Vision*, volume 2, page 747, Vancouver, Canada, 2001.

[16] T. Zhao and R. R. Nevatia. Bayesian human segmentation in crowded situations. In *IEEE Computer Vision and Pattern Recognition, Madison, Wisconsin*, volume 2, pages 459–466, 2003.

Detecting Abandoned Luggage Items in a Public Space

Kevin Smith, Pedro Quelhas, and Daniel Gatica-Perez
IDIAP Research Institute & École Polytechnique Fédérale de Lausanne (EPFL)
Switzerland
{smith, quelhas, gatica}@idiap.ch

Abstract

Visual surveillance is an important computer vision research problem. As more and more surveillance cameras appear around us, the demand for automatic methods for video analysis is increasing. Such methods have broad applications including surveillance for safety in public transportation, public areas, and in schools and hospitals. Automatic surveillance is also essential in the fight against terrorism. In this light, the PETS 2006 data corpus contains seven left-luggage scenarios with increasing scene complexity. The challenge is to automatically determine when pieces of luggage have been abandoned by their owners using video data, and set an alarm. In this paper, we present a solution to this problem using a two-tiered approach. The first step is to track objects in the scene using a trans-dimensional Markov Chain Monte Carlo tracking model suited for use in generic blob tracking tasks. The tracker uses a single camera view, and it does not differentiate between people and luggage. The problem of determining if a luggage item is left unattended is solved by analyzing the output of the tracking system in a detection process. Our model was evaluated over the entire data set, and successfully detected the left-luggage in all but one of the seven scenarios.

1. Introduction

In recent years the number of video surveillance cameras has increased dramatically. Typically, the purpose of these cameras is to aid in keeping public areas such as subway systems, town centers, schools, hospitals, financial institutions and sporting arenas safe. With the increase in cameras comes an increased demand for automatic methods for interpreting the video data.

The PETS 2006 data set presents a typical security problem: detecting items of luggage left unattended at a busy train station in the UK. In this scenario, if an item of luggage is left unattended for more than 30s, an alarm should be raised. This is a challenging problem for automatic systems, as it requires two key elements: the ability to reliably detect luggage items, and the ability to reliably determine the owner of the luggage and if they have left the item unattended.

Figure 1. *Experimental Setup.* An example from the *PETS 2006* data set, sequence S1 camera 3. A man sets his bag on the ground and leaves it unattended.

Our approach to this problem is two-tiered. In the first stage, we apply a probabilistic tracking model to one of the camera views (though four views were provided, we restrict ourselves to camera 3). Our tracking model uses a mixed-state Dynamic Bayesian Network to jointly represent the number of people in the scene and their locations and size. It automatically infers the number of objects in the scene and their positions by estimating the mean configuration of a trans-dimensional Markov Chain Monte Carlo (MCMC) sample chain.

In the second stage, the results of the tracking model are passed to a bag detection process, which uses the object identities and locations from the tracker to attempt to solve the left-luggage problem. The process first searches for potential bag objects, evaluating the likelihood that they are indeed a bag. It then verifies the candidate bags, and searches the sequences for the owners of the bags. Finally,

Table 1. Challenges in the PETS 2006 data corpus.

Seq.	length (s)	luggage items	num people nearby	abandoned ?	difficulty (rated by PETS)
S1	121	1 backpack	1	yes	1/5
S2	102	1 suitcase	2	yes	3/5
S3	94	1 briefcase	1	no	1/5
S4	122	1 suitcase	2	yes	4/5
S5	136	1 ski equipment	1	yes	2/5
S6	112	1 backpack	2	yes	3/5
S7	136	1 suitcase	6	yes	5/5

once the bags and owners have been identified, it checks to see if the alarm criteria has been met.

The remainder of the paper is organized as follows. We discuss the data in Section 2. The tracking model is presented in Section 3. The process for detecting bags is described in Section 4. We present results in Section 5 and finish with some concluding remarks in Section 6.

2. The Left Luggage Problem

In public places such as mass transit stations, the detection of abandoned or left-luggage items has very strong safety implications. The aim of the PETS 2006 workshop is to evaluate existing systems performing this task in a real-world environment. Previous work in detecting baggage includes e.g. [3], where still bags are detected in public transport vehicles, and [2], where motion cues were used to detect suspicious background changes. Other work has focused on attempting to detect people carrying objects using silhouettes, e.g. [4]. Additionally, there has been previous work done on other real-world tracking and behavior recognition tasks (including work done for PETS), such as detecting people passing by a shop window [8, 5].

The PETS data corpus contains seven sequences (labeled S1 to S7) of varying difficulty in which actors (sometimes) abandon their piece of luggage within the view of a set of four cameras. An example from sequence S1 can be seen in Figure 1. A brief qualitative description of the sequences appears in Table 1.

An item of luggage is owned by the person who enters the scene with that piece of luggage. It is *attended* to as long as it is in physical contact with the person, or within two meters of the person (as measured on the floor plane). The item becomes *unattended* once the owner is further than two meters from the bag. The item becomes *abandoned* if the owner moves more than three meters from the bag (see Figure 2). The PETS task is to recognize these events, to trigger a warning 30s after the item is unattended, and to trigger an alarm 30s after it is abandoned.

The data set contains several challenges. The bags vary in size; they are typically small (suitcases and backpacks) but also include large items like ski equipment. The activities of the actors also create challenges for detecting left-luggage items by attempting to confuse ownership of the item of luggage. In sequence S4, the luggage owner sets down his suitcase, is joined by another actor, and leaves (with the second actor still in close proximity to the suitcase). In sequence S7, the luggage owner leaves his suitcase and walks away, after which five other people move in close proximity to the the suitcase.

A shortcoming of the PETS 2006 data corpus is that no training data is provided, only the test sequences. We refrained from learning on the test set as much as possible, but a small amount of tuning was unavoidable. Any parameters of our model learned directly from the data corpus are mentioned in the following sections.

3. Trans-Dimensional MCMC Tracking

The first stage of left-luggage detection is tracking. Our approach jointly models the number of objects in the scene, their locations, and their size in a mixed-state Dynamic Bayesian Network. With this model and foreground segmentation features, we infer a solution to the tracking problem using trans-dimensional MCMC sampling.

Solving the multi-object tracking problem with particle filters (PF) is a well studied topic, and many previous efforts have adopted a rigorous joint state-space formulation to the problem [6, 7, 9]. However, sampling on a joint state-space quickly becomes inefficient as the dimensionality increases when objects are added. Recently, work has concentrated on using MCMC sampling to track multiple objects more efficiently [7, 9, 11]. The model in [7] tracked a fixed number of interacting objects using MCMC sampling while [9] extended this model to handle varying number of objects via reversible-jump MCMC sampling.

In a Bayesian approach, tracking can be seen as the estimation of the filtering distribution of a state $\mathbf{X}_t$ given a sequence of observations $\mathbf{Z}_{1:t} = (\mathbf{Z}_1, ..., \mathbf{Z}_t)$, $p(\mathbf{X}_t|\mathbf{Z}_{1:t})$. In our model, the state is a joint multi-object configuration

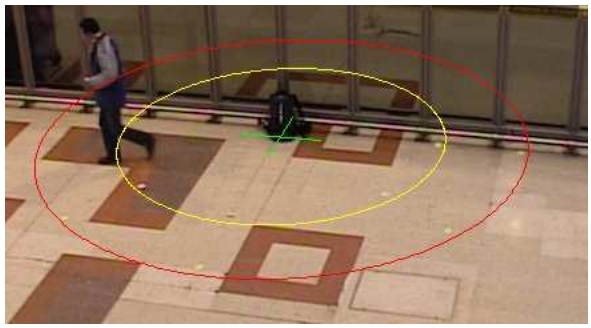


Figure 2. *Alarm Conditions.* The green cross indicates the position of the bag on the floor plane. Owners inside the area of the yellow ring (2 meters) are considered to be *attending* to their luggage. Owners between the yellow ring and red ring (3 meters) left their luggage *unattended*. Owners outside the red ring have *abandoned* their luggage. A warning should be triggered if a bag is unattended for 30s or more, and an alarm should be triggered if a bag is abandoned for 30s or more.

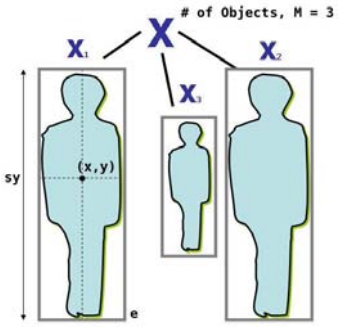


Figure 3. The state model for multiple objects.

and the observations consist of information extracted from the image sequence. The filtering distribution is recursively computed by

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) = C^{-1}p(\mathbf{Z}_t|\mathbf{X}_t) \times \int_{\mathbf{X}_{t-1}} p(\mathbf{X}_t|\mathbf{X}_{t-1})p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})d\mathbf{X}_{t-1}, \quad (1)$$

where $p(\mathbf{X}_t|\mathbf{X}_{t-1})$ is a dynamic model governing the predictive temporal evolution of the state, $p(\mathbf{Z}_t|\mathbf{X}_t)$ is the observation likelihood (measuring how the predictions fit the observations), and C is a normalization constant.

Under the assumption that the distribution $p(\mathbf{X}_{t-1}|\mathbf{Z}_{1:t-1})$ can be approximated by a set of unweighted particles $\{\mathbf{X}_t^{(n)}|n = 1,...,N\}$, where $\mathbf{X}_t^{(n)}$ denotes the n-th sample, the Monte Carlo approximation of Eq. 1 becomes

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx C^{-1}p(\mathbf{Z}_t|\mathbf{X}_t)\sum_n p(\mathbf{X}_t|\mathbf{X}_{t-1}^{(n)}). \quad (2)$$

The filtering distribution in Eq. 2 can be inferred using MCMC sampling as outlined in Section 3.4.

3.1. State Model for Varying Numbers of Objects

The dimension of the state vector must be able to vary along with the number of objects in the scene in order to model them correctly. The state at time t contains multiple objects, and is defined by $\mathbf{X}_t = \{X_{i,t}|i \in \mathcal{I}_t\}$, where $\mathcal{I}_t$ is the set of object indexes, $m_t = |\mathcal{I}_t|$ denotes the number of objects and $|\cdot|$ indicates set cardinality. The special case of zero objects in the scene is denoted by $\mathbf{X}_t = \emptyset$.

The state of a single object is defined as a bounding box (see Figure 3) and denoted by $\mathbf{X}_{i,t} = (x_{i,t}, y_{i,t}, sy_{i,t}, e_{i,t})$ where $x_{i,t}, y_{i,t}$ is the location in the image, $sy_{i,t}$ is the height scale factor, and $e_{i,t}$ is the eccentricity defined by the ratio of the width over the height.

3.2. Dynamics and Interaction

Our dynamic model for a variable number of objects is

$$p(\mathbf{X}_t|\mathbf{X}_{t-1}) \propto \prod_{i\in\mathcal{I}_t} p(\mathbf{X}_{i,t}|\mathbf{X}_{i,t-1})p_0(\mathbf{X}_t) \quad (3)$$

$$\stackrel{def}{=} p_V(\mathbf{X}_t|\mathbf{X}_{t-1})p_0(\mathbf{X}_t), \quad (4)$$

where p_V is the predictive distribution. Following [9], we define p_V as $p_V(\mathbf{X}_t|\mathbf{X}_{t-1}) = \prod_{i\in\mathcal{I}_t} p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1})$ if $\mathbf{X}_t \neq \emptyset$, and $p_V(\mathbf{X}_t|\mathbf{X}_{t-1}) = C$ otherwise. Additionally, we define $p(\mathbf{X}_{i,t}|\mathbf{X}_{t-1})$ either as the object dynamics $p(\mathbf{X}_{i,t}|\mathbf{X}_{i,t-1})$ if object i existed in the previous frame, or as a distribution $p_{init}(\mathbf{X}_{i,t})$ over potential initial object birth positions otherwise. The single object dynamics is given by $p(\mathbf{X}_{i,t}|\mathbf{X}_{i,t-1})$, where the dynamics of the body state $\mathbf{X}_{i,t}$ is modeled as a 2nd order auto-regressive (AR) process.

As in [7, 9], the interaction model $p_0(\mathbf{X}_t)$ prevents two trackers from fitting the same object. This is achieved by exploiting a pairwise Markov Random Field (MRF) whose graph nodes are defined at each time step by the objects and the links by the set $\mathcal{C}$ of pairs of proximate objects. By defining an appropriate potential function $\phi(\mathbf{X}_{i,t}, \mathbf{X}_{j,t})$, the interaction model, $p_0(\mathbf{X}_t) = \prod_{ij\in\mathcal{C}} \phi(\mathbf{X}_{i,t}, \mathbf{X}_{j,t})$, enforces constraints in the dynamic model of objects based on the locations of the object's neighbors.

With these terms defined, the Monte Carlo approximation of the filtering distribution in Eq. 2 becomes

$$p(\mathbf{X}_t|\mathbf{Z}_{1:t}) \approx C^{-1}p(\mathbf{Z}_t|\mathbf{X}_t) \prod_{ij\in\mathcal{C}} \phi(\mathbf{X}_{i,t}, \mathbf{X}_{j,t}) \times \sum_n p_V(\mathbf{X}_t|\mathbf{X}_{t-1}^{(n)}). \quad (5)$$

3.3. Observation Model

The observation model makes use of a single *foreground segmentation* observation source, $\mathbf{Z}_t$, as described in [10], from which two features are constructed. These features form the *zero-object likelihood* $p(\mathbf{Z}_t^{zero}|\mathbf{X}_{i,t})$ and the *multi-object likelihood* $p(\mathbf{Z}_{i,t}^{multi}|\mathbf{X}_{i,t})$. The *multi-object likelihood* is responsible for fitting the bounding boxes to foreground blobs and is defined for each object i present in the scene. The *zero-object likelihood* does not depend on the current number of objects, and is responsible for detecting new objects appearing in the scene. These terms are combined to form the overall likelihood,

$$p(\mathbf{Z}_t|\mathbf{X}_t) = \left[\prod_{i\in\mathcal{I}_t} p(\mathbf{Z}_{i,t}^{multi}|\mathbf{X}_{i,t})\right]^{\frac{1}{m_t}} p(\mathbf{Z}_t^{zero}|\mathbf{X}_t). \quad (6)$$

The *multi-object likelihood* for a given object i is defined by the response of a 2-D Gaussian centered at a learned position in precision-recall space (ν_l, ρ_l) to the values given by that objects current state (ν_t^i, ρ_t^i) (where ν and ρ are precision and recall, respectively) [9]. The *multi-object likelihood* terms in Eq. 6 are normalized by m_t to be invariant to changing numbers of objects. The precision for object i is defined as the area given by the intersection of the spatial support of object i and the foreground F, over the spatial support of object i. The recall of object i is defined as the

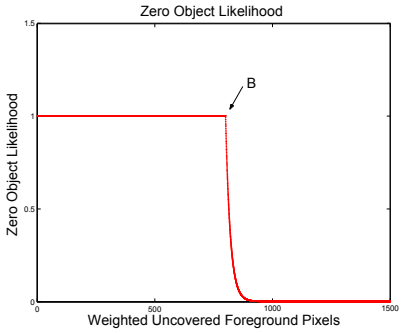


Figure 4. The Zero-Object likelihood defined over the number of weighted uncovered pixels.

area given by the intersection of the spatial support of object i and the dominant foreground blob it covers F_p, over the size of the foreground blob it is covering. A special case for recall occurs when multiple objects overlap the same foreground blob. In such situations, the upper term of the recall for each of the affected objects is computed as the intersection of the combined spatial supports with the foreground blob.

The *zero-object likelihood* gives low likelihoods to large areas of uncovered pixels, thus encouraging the model to place a tracker over all large-enough foreground patches. This is done by computing a weighted *uncovered pixel count* between the foreground segmentation and the current state $\mathbf{X}_t^i$. Pixels from blobs unassociated with any tracker i receive b times the weight of normal pixels. If U is the number of weighted uncovered foreground pixels, the zero-object likelihood is computed as

$$p(\mathbf{Z}_t^{zero}|\mathbf{X}_{i,t}) \propto exp(-\lambda \ \ max(0, U - B)) \qquad (7)$$

where λ is a hyper-parameter and B is the amount of weighted uncovered foreground pixels to ignore before penalization begins (see Figure 4).

3.4. Inference with Trans-Dimensional MCMC

To solve the inference issue in large dimensional state-spaces, we have adopted the Reversible-Jump MCMC (RJMCMC) sampling scheme proposed by several authors [11, 9] to efficiently sample over the posterior distribution, which has been shown to be superior to a Sequential Importance Resampling (SIR) PF for joint distributions over multiple objects.

In RJMCMC, a Markov Chain is defined such that its stationary distribution is equal to the target distribution, Eq. 5 in our case. The Markov Chain must be defined over a variable-dimensional space to accommodate the varying number of objects, and is sampled using the Metropolis-Hastings (MH) algorithm. Starting from an arbitrary configuration, the algorithm proceeds by repetitively selecting a *move type*, υ from a set of moves Υ with prior probability p_υ and sampling a new configuration $\mathbf{X}^*$ from a proposal distribution $q(\mathbf{X}^*|\mathbf{X})$. The move can either change the dimensionality of the state (as in birth or death) or keep it fixed. The proposed configuration is then added to the Markov Chain with probability

$$\alpha = \min\left(1, \frac{p(\mathbf{X}^*)q(\mathbf{X}|\mathbf{X}^*)}{p(\mathbf{X})q(\mathbf{X}^*|\mathbf{X})}\right) \qquad (8)$$

or the current configuration otherwise. The acceptance ratio α can be re-expressed through *dimension-matching* as

$$\alpha = \min\left(1, \frac{p(\mathbf{X}^*)p_\upsilon q_\upsilon(\mathbf{X})}{p(\mathbf{X})p_{\upsilon^*} q_{\upsilon^*}(\mathbf{X}^*)}\right) \qquad (9)$$

where q_υ is a move-specific distribution and p_υ is the prior probability of choosing a particular move type.

We define three different move types in our model: birth, death, and update:

- **Birth** of a new object, implying a dimension increase, from m_t to $m_t + 1$.
- **Death** of an existing object, implying a dimension decrease, from m_t to $m_t - 1$.
- **Update** of the state parameters of an existing object according to the dynamic process described in Section 3.2.

The tracking solution at time t is determined by computing the mean estimate of the MCMC chain at time t.

4. Left-Luggage Detection Process

The second stage of our model is the *left-luggage detection process*. It is necessary to search for bags separately because the tracking model does not differentiate between people and bags. Also, the left-luggage detection process is necessary to overcome failures of the tracking model to retain consistent identities of people and bags over time.

The left-luggage detection process uses the output of the tracking model and the foreground segmentation, F, for each frame as input, identifies the luggage items, and determines if/when they are abandoned. The output of the tracking model contains the number of objects, their identities and locations, and parameters of the bounding boxes.

The left-luggage detection process relies on three critical assumptions about the properties of a left-luggage item:

1. Left-luggage items probably don't move.
2. Left-luggage items probably appear smaller than people.
3. Left-luggage items must have an owner.

The first assumption is made with the understanding that we are only searching for unattended (stationary) luggage. For this case it is a valid assumption. The more difficult task

Figure 5. Object blobs (yellow contour on the right) are constructed from bounding boxes (left) and foreground segmentation (right).

of detecting luggage moving with its owner requires more information than is provided by the tracker.

To tackle the left-luggage problem, the detection process breaks it into the following steps:

- **Step 1**: Identify the luggage item(s).
- **Step 2**: Identify the owners(s).
- **Step 3**: Test for alarm conditions.

Step 1. To identify the bags, we start by processing the tracker bounding boxes ($\overline{\mathbf{X}}_t$) and the foreground segmentation F to form object blobs by taking the intersection of the areas in each frame, $\overline{\mathbf{X}}_t^i \cap F$, as seen in Figure 5. The mean x and y positions of the blobs are computed, and the size of the blobs are recorded. A 5-frame sliding window is then used to calculate the blob velocities at each instant. Examples of blob size and velocity for sequence S1 can be seen in Figure 6. Following the intuition of our assumptions, likelihoods are defined such that small and slow moving blobs are more likely to be items of luggage:

$$p_s(B^i = 1|\overline{\mathbf{X}}_{1:t}^i) \propto \mathcal{N}(s_t^i, \mu_s, \sigma_s) \quad (10)$$

$$p_v(B^i = 1|\overline{\mathbf{X}}_{1:t}^i) \propto exp(-\lambda v_t^i) \quad (11)$$

where p_s is the size likelihood, p_v is the velocity likelihood, $B^i = 1$ indicates that blob i is a bag, s_t^i is the size of blob i

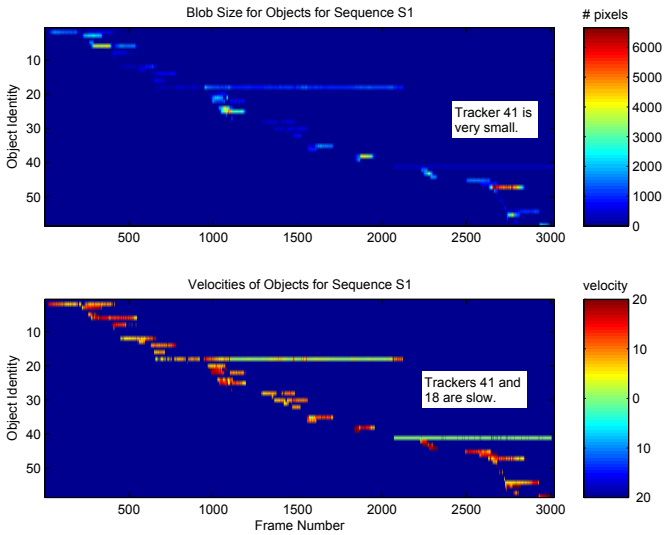


Figure 6. *(top)* Size and *(bottom)* velocity for each object computed over the course of sequence S1. In this example, blob 41 was tracking the bag, and blob 18 was tracking the owner of the bag.

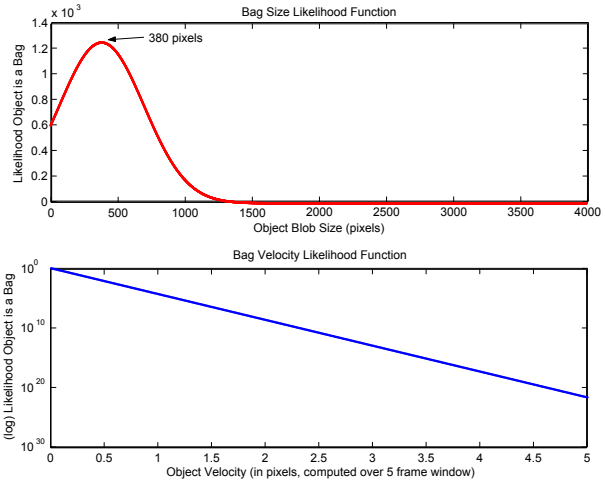


Figure 7. *(top)* Likelihood that a blob is a bag based on size (p_s). *(bottom)* Likelihood that a blob is a bag based on velocity (p_v).

and time t, μ_s is the mean bag blob size, σ_s is the bag blob variance, v_t^i is the blob velocity and λ is a hyper-parameter. These parameters were hand picked with knowledge of the data, but not tuned (see Section 5). The velocity and size likelihood terms can be seen in Figure 7. Because long-living blobs are more likely to be pieces of left-luggage, we sum the frame-wise likelihoods without normalizing by blob lifetime. The overall likelihood that a blob is a left-luggage item combines p_v and p_s,

$$p(B^i = 1|\overline{\mathbf{X}}_{1:t}^i) \propto \sum_{t=1:T} \mathcal{N}(s_t^i, \mu_s, \sigma_s) exp(-\lambda v_t^i). \quad (12)$$

An example of the overall likelihoods for each blob can be seen in the top panel of Figure 8.

The bag likelihood term $p(B^i = 1|\overline{\mathbf{X}}_{1:t}^i)$ gives preference to long-lasting, slow, small objects as seen in the top of Figure 8. Bag candidates are selected by thresholding the likelihood, $p(B^i = 1|\overline{\mathbf{X}}_{1:t}^i) > T_b$. In the example case of S1, this means blobs 41 (which tracked the actual bag) and 18 (which tracked the owner of the bag) will be selected as bag candidates. Because of errors in tracking, there could be several unreliable bag candidates. Thus, in order to be identified as a bag, the candidates must pass the following additional criteria: (1) they must not lie at the borders of the image (preventing border artifacts), and (2) their stationary position must not lie on top of other bag candidates (this eliminates the problem of repeated trackers following the same bag).

The next part of the detection process is to determine the lifespan of the bag. The identities of the tracker are too unreliable to perform this alone, as they are prone to swapping and dying. But as we have assumed that items of left luggage do not move, we can use the segmented foreground image to reconstruct the lifespan of the bag. A shape template $\mathcal{T}^i$ is constructed from the longest segment of frames below a low velocity threshold, T_v, to model what the bag looks like when it is stationary. The template is a normal-

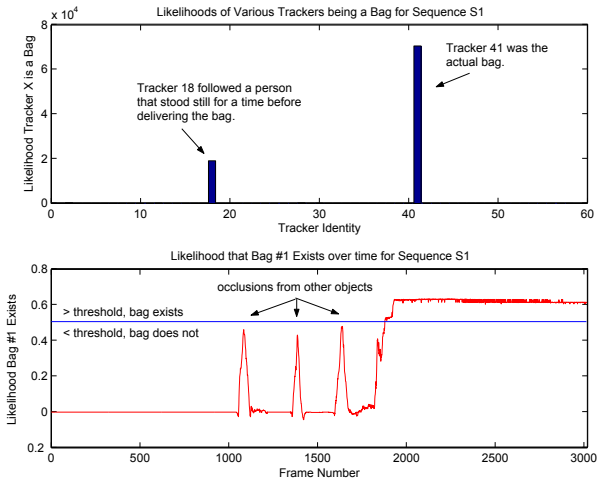


Figure 8. *(top)* The overall likelihoods of various tracked blobs in sequence S1. *(bottom)* The likelihood that bag candidate 1 exists for the length of sequence S1.

ized sum of binary image patches taken from the segmented foreground image around the boundaries of the bag candidate blob with the background pixels values changed from 0 to -1.

A *bag existence* likelihood in a given frame is defined for the blob candidate by extracting image patches from the binary image at the stationary bag location $\mathcal{I}_t$ and performing an element-wise multiplication

$$p(E_t = 1|B^i) \propto \sum_u \sum_v \mathcal{T}^i(u,v) \times \mathcal{I}_t(u,v) \qquad (13)$$

where $E_t = 1$ indicates that a bag exists at time t, and u and v are pixel indices. The bag existence likelihood is computed over the entire sequence for each bag candidate. An example of the existence likelihood for the bag found from blob 41 in the example can be seen in Figure 8 (bottom). A threshold, T_e, is defined as 80% of of the maximal likelihood value. The bag is defined as existing in frames with existence likelihoods above the threshold.

Step 2. In step 1, the existence and location of bags in the sequence were determined; in step 2 we must identify the owner of the bag. Unlike left-luggage items, we cannot assume the owner will remain stationary, so we must rely on the results of the tracker to identify the owner.

Typically, when a piece of luggage is set down, the tracking results in a single bounding box contain both the owner and the bag. Separate bounding boxes only result when the owner moves away from the bag, in which case, one of two cases can occur: (1) the original bounding box follows the owner and a new box is born to track the bag, or (2) the original bounding box stays with the bag, and a new bounding box is born and follows the owner. Thus, to identify the owner of the bag, we inspect the history of the tracker present when the bag first

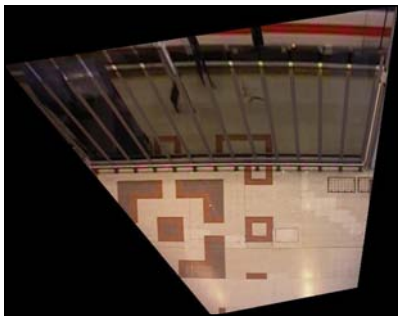


Figure 9. Image from S1 (camera 3) transformed by the DLT. In this plane, floor distance can be directly calculated between the bag and owner.

appeared as determined by the bag existence likelihood. If that tracker moves away and dies while the bag remains stationary, it must be the one identifying the owner. In this case, we designate the blob results computed from the estimate of the tracker and the foreground segmentation as the owner. If the tracker remains with the bag and dies, we begin a search for nearby births of new trackers within radius r pixels. The first nearby birth is deemed the owner. If no nearby births are found, the bag has no owner, and violates assumption 3, so it is thrown out.

Step 3. With the bag and owner identified, and knowledge of their location in a given frame, the last task is straightforward: determining if/when the bag is left unattended and sounding the alarm (with one slight complication). Thus far, we have been working within the camera image plane. To transform our image coordinates to the world coordinate system, we computed the 2D homography between the image plane and the floor of the train station using calibration information from the floor pattern provided by PETS [1]. Using the homography matrix H, a set of coordinates in the image γ_1 is transformed to the floor plane of the train station γ_2 by the discrete linear transform (DLT) $\gamma_2 = H\gamma_1$ This can even be done for the image itself (see Figure 9).

However, the blob centroids of objects in the image do not lay on the floor plane, so using the DLT on these coordinates will yield incorrect locations in the world coordinate system. Thus, we estimate the foot position of each blob by taking its bottommost y value and the mean x value, and estimate its world location by passing this point to the DLT. Now triggering warnings and alarms for unattended luggage can be performed by computing the distance between the bag and owner and counting frames.

It should be noted that the left-luggage detection process can be performed *online* using the 30s alarm window to search for bag items and their owners.

Table 2. Luggage Detection Results (for bags set on the floor, *even if never left unattended*). Mean results are computed over 5 runs for each sequence.

seq		# luggage items	x location	y location
S1	ground truth	1	.22	-.44
	mean result	1.0	.22	-.29
	error	0%	0.16 meters	
S2	ground truth	1	.34	-.52
	mean result	1.2	.23	-.31
	error	20%	0.22 meters	
S3	ground truth	1	.86	-.54
	mean result	0.0	-	-
	error	100%	N/A	
S4	ground truth	1	.24	-.27
	mean result	1.0	.13	.03
	error	0%	0.32 meters	
S5	ground truth	1	.34	-.56
	mean result	1.0	.24	-.49
	error	0%	0.13 meters	
S6	ground truth	1	.80	-.78
	mean result	1.0	.65	-.41
	error	0%	0.40 meters	
S7	ground truth	1	.35	-.57
	mean result	1.0	.32	-.39
	error	0%	0.19 meters	

5. Results

To evaluate the performance of our model, a series of experiments was performed over the entire data corpus. Because the MCMC tracker is a stochastic process, five experimental runs were performed over each sequence, and the mean values computed over these runs. To speed up computation time, the size of the images was reduced to half resolution (360×288).

As previously mentioned, because no training set was provided, some amount of training was done on the test set. Specifically, the foreground precision parameters of the foreground model for the tracker were learned (by annotating 41 bounding boxes from sequences S1 and S3, computing the foreground precision, and simulating more data points by perturbing these annotations). Several other parameters were hand-selected including the e and sy limits of the bounding boxes and the parameters of the size and velocity models, but these values were not extensively tuned, and remained constant for all seven sequences. Specific parameter values used in our experiments were: $\mu_s = 380$, $\sigma_s = 10000$, $\lambda = 10$, $T_v = 1.5$, $T_b = 5000$, $r = 100$, $b = 3$, $B = 800$.

We separated the evaluation into two tasks: luggage detection (Table 2) and alarm detection (Table 3). Luggage detection refers to finding the correct number of pieces of luggage set on the floor and their locations, *even if they are never left unattended* (as is the case in S3). Alarm detection refers to the ability of the model to trigger an alarm or warning event when the conditions are met.

The error values reported for *# luggage items*, *# alarms*, and *# warnings* are computed similarly to the word error rate, often used in speech recognition:

$$\text{error rate} = \frac{\text{deletions} + \text{insertions}}{\text{events to detect}} \times 100 \qquad (14)$$

Table 3. Alarm Detection Results (Mean results are computed over 5 runs for each sequence).

seq		# Alarms	# Warnings	Alarm time	Warning time
S1	ground truth	1	1	113.7s	113.0s
	mean result	1.0	1.0	112.9s	112.8s
	error	0%	0%	0.78s	0.18s
S2	ground truth	1	1	91.8s	91.2s
	mean result	1.0	1.2	90.8s	90.2s
	error	0%	20%	1.08s	1.05s
S3	ground truth	0	0	-	-
	mean result	0.0	0.0	-	-
	error	0%	0%	-	-
S4	ground truth	1	1	104.1s	103.4s
	mean result	0.0	0.0	-	-
	error	100%	100%	-	-
S5	ground truth	1	1	110.6s	110.4s
	mean result	1.0	1.0	110.6s	110.5s
	error	0%	0%	0.04s	0.45s
S6	ground truth	1	1	96.9s	96.3s
	mean result	1.0	1.0	96.9s	96.1s
	error	0%	0%	0.08s	0.18s
S7	ground truth	1	1	94.0s	92.7s
	mean result	1.0	1.0	90.4s	90.3s
	error	0%	0%	3.56s	2.38s

As shown in Table 2, our model consistently detected each item of luggage in sequences S1, S2, S4, S5, S6, and S7. A false positive (FP) bag was detected in one run of sequence S2 as a result of trash bins being moved and disrupting the foreground segmentation (the tracker mistook a trash bin for a piece of luggage). We report 100% error for detecting luggage items in S3, which is due to the fact that the owner never moves away from the bag, and takes the bag with him as he leaves the scene (never generating a very bag-like blob). However, it should be noted that for this sequence, our system correctly predicted 0 alarms and 0 warnings.

The spatial errors were typically small (ranging from 0.13 meters to 0.40 meters), though they could be improved by using multiple camera views to localize the objects, or by using the full resolution images. The standard deviation in x ranged from 0.006 to 0.04 meters, and in y from 0.009 to .09 meters (not shown in Table 2).

As seen in Table 3, our model successfully predicted alarm events in all sequences but S4, with the exception of a FP warning in S2. Of these sequences, the alarms and warnings were generated within 1.1s of the ground truth with the exception of S7 (the most difficult sequence). Standard deviation in alarm events was typically less than 1s, but approximately 2s for S2 and S7 (not shown in Table 3).

Our model reported a 100% error rate for detecting warnings and alarms in S4. In this sequence, the bag owner sets down his bag, another actor joins him, and the owner leaves. The second actor stays in close proximity to the bag for the

duration of the sequence. In this case, our model repeatedly mistook the second actor as the bag owner, and erroneously did not trigger any alarms. This situation could have been avoided with better identity recognition in the tracker (perhaps by modeling object color).

In Figure 10, we present the tracker outputs for one of the runs on sequence S5. Colored contours are drawn around detected objects, and the item of luggage is highlighted after it is detected. Videos showing typical results for each of the sequences are available at http://www.idiap.ch/~smith.

6. Conclusion and Future Work

In this paper, we have presented a two-tiered solution to the left-luggage problem, wherein a detection process uses the output of an RJMCMC tracker to find abandoned pieces of luggage. We evaluated the model on the PETS 2006 data corpus which consisted of seven scenarios, and correctly predicted the alarm events in six of the seven scenarios with good accuracy. Despite less-than-perfect tracking results for a single camera view, the bag detection process was able to perform well for high-level tasks. Possible avenues for future work include using multiple camera views and investigating methods for maintaining object identities in the tracker better.

Acknowledgements: This work was partly supported by the Swiss National Center of Competence in Research on Interactive Multimodal Information Management (IM2), the European Union 6th FWP IST Integrated Project AMI (Augmented Multi-party Interaction, FP6-506811, AMI-181), and the IST project CARETAKER.

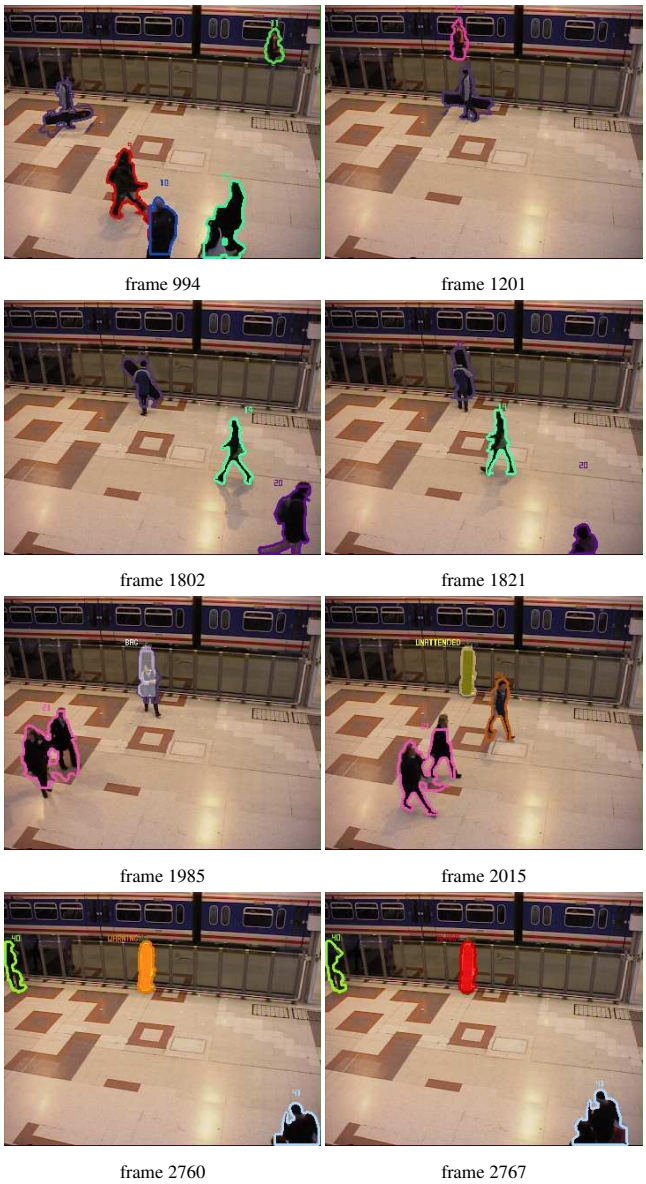


Figure 10. *Results: detecting left luggage items for sequence S5 from the PETS 2006 data corpus.* The luggage owner is seen arriving in frame 994, loiters for a bit in frame 1201, and places his ski equipment against the wall in frames 1802 and 1821. In frame 1985 the bag is detected, but the owner is still within the *attending* limits. In frame 2015, the model has determined he is more than 3 meters from the bag, and marks the bag as *unattended*. In frame 2760, 30s have passed since the owner left the 2 meter limit, and a warning is triggered (ground truth warning occurs in frame 2749). In frame 2767, 30s have passed since the owner left the 3 meter limit, and an alarm is triggered (ground truth alarm occurs in frame 2764).

References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry*. Cambridge University Press, 2000. 6

[2] D. Gibbons *et al*, Detecting Suspicious Background Changes in Video Surveillance of Busy Scenes. In *IEEE Proc. WACV-96*, Sarasota FL, USA, 1996. 2

[3] Milcent, G. and Cai, Y, Location Based Baggage Detection for Transit Vehicles. *Technical Report, CMU-CyLab-05-008, Carnegie Mellon University.* 2

[4] Haritaolu, I. *et al*, Backpack: Detection of People Carrying Objects Using Silhouettes. In *IEEE Proc. ICCV*, 1999. 2

[5] I. Haritaoglu and M. Flickner, Detection and tracking of shopping groups in stores. In *IEEE Proc. CVPR*, 2001. 2

[6] M. Isard and J. MacCormick, BRAMBLE: A Bayesian multi-blob tracker. In *IEEE Proc. ICCV*, Vancouver, July 2001. 2

[7] Z. Khan, T. Balch, and F. Dellaert, An MCMC-based particle filter for tracking multiple interacting targets. In *IEEE Proc. ECCV*, Prague, May 2004. 2, 3

[8] A. E. C. Pece, From cluster tracking to people counting. In *3rd PETS Workshop*, June 2002. 2

[9] K. Smith, D. Gatica-Perez, J.-M. Odobez, Using particles to track varying numbers of objects. In *IEEE Proc. CVPR*, June 2005. 2, 3, 4

[10] C. Stauffer and E. Grimson, Adaptive background mixture models for real-time tracking. In *IEEE Proc. CVPR*, June 1999. 3

[11] T. Zhao and R. Nevatia, Tracking multiple humans in crowded environment. In *IEEE Proc. CVPR*, Washington DC, June 2004. 2, 4

Left-Luggage Detection using Bayesian Inference

Fengjun Lv Xuefeng Song Bo Wu Vivek Kumar Singh Ramakant Nevatia

Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, CA 90089-0273
{flv|xsong|bowu|viveksin|nevatia}@usc.edu

Abstract

This paper presents a system that incorporates low-level object tracking and high-level event inference to solve the video event recognition problem. First, the tracking module combines the results of block tracking and human tracking to provide the trajectory as well as the basic type (human or non-human) of each detected object. The trajectories are then mapped to 3D world coordinates, given the camera model. Useful features such as speed, direction and distance between objects are computed and used as evidence. Events are represented as hypotheses and recognized in a Bayesian inference framework. The proposed system has been successfully applied to many event recognition tasks in the real world environment. In particular, we show results of detecting the left-luggage event on the PETS 2006 dataset.

1. Introduction

Video event recognition has been an active topic in computer vision and artificial intelligence community recently. It is a difficult task because it consists of not only low-level vision processing tasks such as tracking and object detection, which by themselves are difficult problems, but also high-level event inference, which has to handle the ambiguity in event definition, the variations in execution style and duration and the fusion of multiple source of information.

Another major difficulty is the lack of event data and evaluation criteria. This is because the acquisition of event data is hard either because events of interest (especially unusual events) rarely happen in real life or because recording of such events is prohibited in many situations. So many researchers have to rely on staged data, which requires a lot of effort to make it look realistic. This difficulty impedes the evaluation and comparison of different event recognition algorithms.

The PETS 2006 workshop serves the need of event data by providing a publicly available dataset that contains event scenarios in a real-world environment. The workshop focuses on detecting the left-luggage event in a railway station. The event has a clear definition consisting of the following three rules: **(1) Contextual rule**: A luggage is owned and attended by a person who enters the scene with the luggage until such point that the luggage is not in physical contact with the person. **(2) Spatial rule**: A luggage is unattended when the owner is further than 3 meters from the luggage. **(3) Temporal rule**: If a luggage has been left unattended by the owner for a period of 30 consecutive seconds in which time the owner has not re-attended to the luggage, nor has the luggage been attended to by a second party, the alarm event is triggered.

Although detection of left-luggage is the only considered task of PETS 2006, the event contains large variations in terms of the types of luggage and the complexity of scenarios. The dataset includes five types of luggage: {briefcase, suitcase, 25 liter rucksack, 70 liter backpack, ski gear carrier}. The size and the shape of these different types are different, which increases the difficulty in luggage detection and tracking. The dataset considers the following seven scenarios with increasing complexity. **(1)** Person 1 with a rucksack loiters before leaving the rucksack unattended. **(2)** Person 1 and 2 enter the scene from opposite directions. Person 1 places a suitcase on the ground, before person 1 and 2 leave together without the suitcase. **(3)** Person 1 temporarily places his briefcase on the ground before picking it up again and leaving. **(4)** Person 1 places a suitcase on the ground. Person 2 arrives and meets with person 1. Person 1 leaves the scene without his suitcase. **(5)** Person 1 with a ski gear carrier loiters before leaving the luggage unattended. **(6)** Person 1 and 2 enter the scene together. Person 1 places a rucksack on the ground, before person 1 and 2 leave together without the rucksack. **(7)** Person 1 with a suitcase loiters before leaving the suitcase unattended. During this event five other persons move in close proximity to the suitcase. Accordingly, the event recognition system should have the flexibility to accommodate these variations.

We present an event recognition system that tackles the challenges in tracking and event recognition problems. For

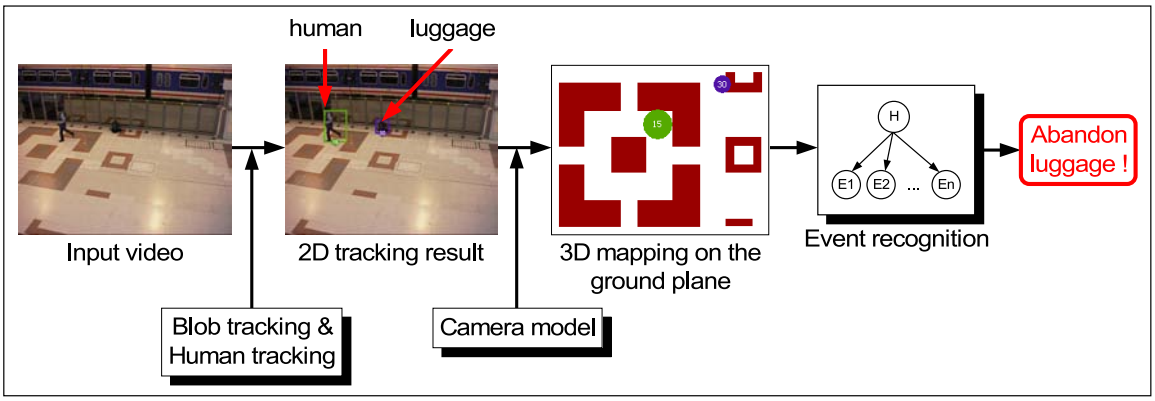


Figure 1: The overview diagram of our system

tracking, a combination of a blob tracker (see Section 2.1) and a human tracker (see Section 2.2) is used to provide the trajectory of each human and carried object. For event recognition, each event is represented and recognized in a Bayesian inference framework (see Section 3.1). In particular, the algorithm is applied to detect the left-luggage event on the PETS 2006 dataset (see Section 3.2). The overview diagram of our system is shown in Figure 1.

Note that in PETS 2006, each scenario is filmed from multiple cameras. We only use the third camera because it provides the best view point and the results based on this single view point is satisfactory.

Many efforts have been made to solve the human tracking problem. As it is hard to provide an adequate description of the whole literature, we only list some that are closely related to this work. The observations of human hypotheses may come from various cues. Some use the result of background subtraction as observations and try to fit multiple human hypotheses to explain the foreground blobs [12]. As the hypotheses space is usually of high dimension, sampling algorithms such as MCMC [12], or dynamic programming algorithms such as EM [7] are applied. Other methods, *e.g.* [2], build deformable silhouette models based on edge features to detect pedestrians and associate the detection responses to form tracks. These two types of approaches are complementary.

Proposed methods for video event recognition can be divided into three sub-categories based on the underlying features they use: **(1)** those based on 2-D image features such as optical flow [3] , body shape/contour [11], space-time interest point [4], **(2)** those based on 2-D trajectories of tracked objects such as hands, legs or the whole body [8] and **(3)** those based on 3-D body pose acquired from motion capture or 3-D pose tracking system [5]. None of above methods, however, involves the manipulation of objects such as luggage.

2. The Tracking Module

We use two components for the tracking task. A Kalman filter based blob tracker, which provides trajectories of all foreground objects including humans and carried objects, is described in Section 2.1. A human body detection based human tracker is described in Section 2.2. We describe the combination of both trackers in Section 2.3.

2.1 Kalman Filter based Blob Tracking

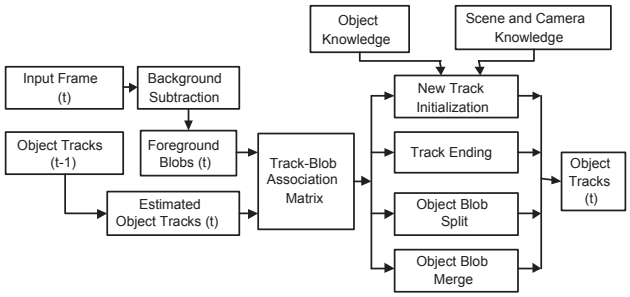


Figure 2: An overview of the blob tracker

Figure 2 shows an overview of our blob tracking algorithm. We learn the background model from the first five hundred frames and apply it to detect the foreground pixels at each frame. Connected foreground pixels form foreground blobs. At each frame t, a blob B_t^i contains the following information:

- Size and location: for simplicity, we use a rectangular bounding box $(x_t^i, y_t^i, width_t^i, height_t^i)$
- Appearance: we use the color histogram of the blob

Ideally, one blob corresponds to one real object and vice versa. But we have to consider the situations such as one object splits into several blobs or multiple objects merge into

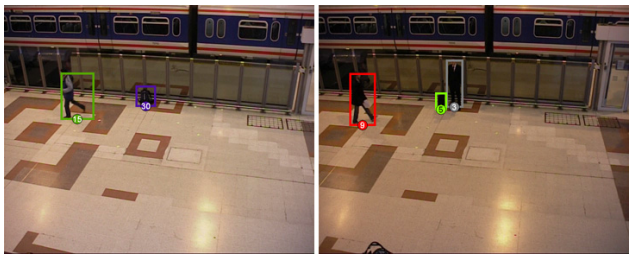


Figure 3: Blob tracking results

Figure 4: Human tracking results

one blob. For a blob B_i^t and an object O_j^t, we assign an association value based on the overlap between the bounding box of B_i^t and O_j^t:

$$M(B_i^t, O_j^t) = \begin{cases} 1, & if \quad \frac{Area(B_i^t \cap \hat{O}_j^t)}{min(Area(B_i^t), Area(\hat{O}_j^t))} > \tau \\ 0, & otherwise \end{cases} \tag{1}$$

where $\hat{O}_i^t$ is the predicted bounding box of O_j^t from the previous frame using the Kalman filter and $B_i^t \cap \hat{O}_j^t$ is the intersection of the two bounding boxes and τ is a threshold between 0 and 1. If $M(B_i^t, O_j^t)$ is one, we call B_i^t and O_j^t are matched.

At time t, assume we have m blobs and n predicted objects. For all pairs of blob and object, we construct an $m \times n$ association matrix. The following are possible situations that we need to consider: **(1)** If the blob-object match is one-to-one, we simply update the object with the blob. **(2)** If a blob has no matched object, a new object is created. **(3)** If an object has no match blobs for several frames, the object is removed. **(4)** If an object has multiple matched blobs, the blobs are merged into a bounding box of the object. **(5)** If a blob has multiple matched objects, the blob is segmented based on the appearance (color histogram) of each matched object.

A luggage is detected based on its mobility: It seldom moves after the start of its track. Some detected humans and luggage are shown in Figure 3.

2.2 Detection based Human Tracking

For shape based human tracking, we take the single frame human detection responses as the observations of human hypotheses. Tracking is done in 2D with a data association style method. The detector is a generalization of the body part detector proposed by Wu et al. in [9]. We do not rely on skin color, or face, hence our method can deal with rear views. The algorithm of [9] has been extended in [10] to track multiple humans based on the detectors. Our tracking algorithm is a simplified version of [10].

2.2.1 Multi-View Human Body Detection

In [9], four part detectors are learned for detecting full-body, head-shoulder, torso, and legs. We only use the full-body detector here. Two detectors are learned: one for the left profile view, and one for the frontal/rear view. The third detector is for right profile view, which is generated by flipping the left profile view horizontally. Nested cascade detectors are learned by boosting edgelet feature based weak classifiers, as in [9]. The training set contains 1,700 positive samples for frontal/rear view, 1,120 for left profile view, and 7,000 negative images. The positive samples are collected from the Internet and the MIT pedestrian set [6]. The negative images are all from the Internet. The training set is fully independent of the test sequences, and the detectors learned are for generic situations. For detection, the input image is scanned by all three detectors and the union of their responses is taken as the multi-view detection result.

2.2.2 Multiple Human Tracking

Humans are tracked in 2D by associating the frame detection responses. This 2D tracking method is a simplified version of [10]. In [10], the detection responses come from four part detectors and a combined detector. To start a trajectory, an initialization confidence $InitConf$ is calculated from T consecutive responses, which correspond to one human hypothesis, based on the cues from color, shape, and position. If $InitConf$ is larger than a threshold θ_{init}, a trajectory is started. To track the human, first data association with the combined detection responses is attempted; if this fails, data association with the part detection responses is attempted; if this fails again, a color based meanshift tracker [1] is used to follow the person. The strategy of trajectory termination is similar to that of initialization. A termination confidence $EndConf$ is calculated when an existing trajectory has been lost by the detector for T time steps. If $EndConf$ is larger than a threshold θ_{end}, the trajectory is terminated.

In this work, we do not use the combined detection method described in [9]. Since the occlusions are not strong in this data set, a local feature based full-body detector gives

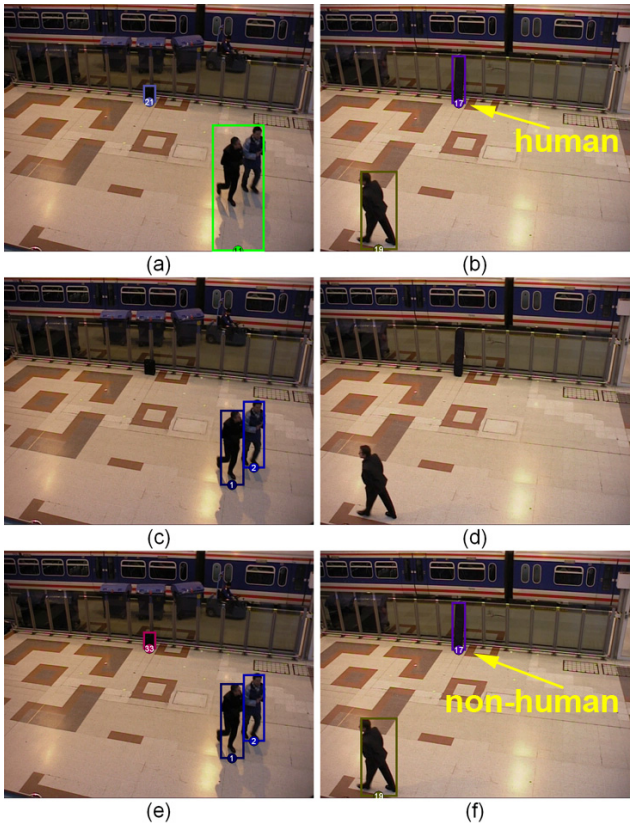


Figure 5: Tracking results of individual trackers and combined tracker. (a),(c),(e): The tracking results on the same frame using the blob tracker alone, the human tracker alone and the combined tracker. The combined tracker can separate two persons from a merge blob and locate the luggage as well. (b),(d),(f): Another example. The combined tracker confirms that the object in the middle is not human. It also tracks the person that is missed by the human tracker due to the large tilt angle.

satisfactory results. Fig.4 shows some sample frames of the tracking results.

2.3 The Combination of the Blob Tracker and the Human Tracker

In most cases, the blob tracker does a good job of tracking objects and locating luggage. However, when multiple humans appear merged from the beginning, the tracker can not separate them well and sometimes the tracker mis-identifies the type of the tracked objects based only on the mobility of the objects. See Figure 5(a) and (b) for examples of these issues.

On the other hand, the human tracker does not track objects other than humans. Also, although the learned detectors work with a large range of tilt angles (within about $[0°, 45°]$), when the tile angle is too large, the detectors can not find the human reliably. These issues can be clearly seen in Figure 5(c) and (d).

We can overcome the above limitations of the blob tracker and the human tracker by combining their results so that if one trajectory from the human tracker largely overlaps one trajectory from the blob tracker, the first trajectory supersedes the second one. This is because the human tracker is based on human shape so it is more reliable. The blob tracker, on the other hand, enhances the results by including objects that are missed by the human tracker. The combined results of the above examples are shown in Figure 5(e) and (f).

3. The Event Recognition Module

We first describe our event recognition algorithm, followed by the details of how this algorithm is applied to detect the left-luggage event on the PETS 2006 dataset.

3.1 Event Recognition using Bayesian Inference

Our event recognition algorithm takes object trajectory (provided by the underlying detection/tracking modules) as the only observation. The algorithm assumes that the type of each object is known. The basic object types include human, carried object, vehicle and scene object (*e.g.* door, stairs). This information is either automatically provided by the detection/tracking modules or specified by system users.

The trajectory is first smoothed using median filter to remove abrupt changes. Useful physical properties such as the position, speed and direction of a moving object and the distance between two objects are inferred from object trajectories. Since these physical properties are measured in terms of world coordinates, 2D object trajectories need to be mapped to 3D world coordinates first. For an object on the ground plane, we assume that the vertical coordinate of its lowest point is zero. Given the camera model, we use the center of the bottom of the image bounding box as the object's lowest point and map this point to 3D position on the ground plane.

System users have to provide an event model for each event in advance. A successful event model has to handle the ambiguity in the event definition (which implies that we should define an event in a probabilistic domain instead of a logical one) and incorporate multiple cues into a computational framework.

Due to these considerations, we use Bayesian inference as the event modeling and recognition framework: events and related cues are considered as hypotheses and evidence, respectively; Competing events are treated as competing hypotheses. For example, regarding the relationship between

a moving object A and a zone B in terms of position of A and B, there are four possibilities (hypotheses), corresponding to four competing events: $\{H_1$:A is outside of B, H_2:A is entering B, H_3:A is inside B, H_4:A is exiting B$\}$. We consider the following two pieces of evidence: $\{E_1$:the distance between A and B sometime (e.g. 0.5 second) ago, E_2:the distance between A and B now$\}$. The complete model also includes the prior probabilities of each hypothesis and the conditional probabilities of each evidence given the hypotheses. In the above example, we can set the prior probabilities of all hypotheses to be equal ($P(H_i) = 1/4$, $i = 1, 2, 3, 4$) and use the following conditional probabilities.

$P(E_1 = 0\|H_i)$:{0,0,1,1}
$P(E_1 > 0\|H_i)$:{1,1,0,0}
$P(E_2 = 0\|H_i)$:{0,1,1,0}
$P(E_2 > 0\|H_i)$:{1,0,0,1}

The meaning of this table is clear. Take the hypothesis H_2 "A is entering B" for example, $E_1 > 0$ and $E_2 = 0$ means A was outside of B sometime ago and A is inside B now. Given the evidence, the probability of H_2 is computed using the Bayesian rule:

$$P(H_2|E_1, E_2) = \frac{P(E_1, E_2|H_2)P(H_2)}{\sum_{i=1}^{4} P(E_1, E_2|H_i)P(H_i)} \quad (2)$$

If we assume each evidence is conditionally independent given the hypothesis, Eq.2 can be rewritten as:

$$P(H_2|E_1, E_2) = \frac{P(E_1|H_2)P(E_2|H_2)P(H_2)}{\sum_{i=1}^{4} P(E_1|H_i)P(E_2|H_i)P(H_i)} \quad (3)$$

Note that the above example has the simplest form of a probability distribution function: a threshold function (*i.e.* the distance is quantified to only two values "=0" and ">0") and the probability is either 0 or 1 (a deterministic case). For more complex cases, the following distribution functions are provided in our system: {Histogram, Threshold, Uniform, Gaussian, Rayleigh, Maxwell}. The parameters of the distribution functions are learned from training data. If such data are not available, the parameters are specified based on user's knowledge of the event.

If a hypothesis H has no competing counterparts, we consider $\neg H$, the opposite of the hypothesis, and use the following equation to compute the posterior probability of hypothesis given evidence.

$$P(H|E_1, \cdots, E_n) = \frac{\prod_{i=1}^{n} P(E_i|H)P(H)}{\prod_{i=1}^{n} P(E_i|H)P(H) + \prod_{i=1}^{n} P(E_i|\neg H)P(\neg H)} \quad (4)$$

When $P(E_i|\neg H)$ is unknown or hard to estimate, we use a default value 0.5.

3.2 Left-Luggage Detection

For the specific task of detecting left-luggage, as described in the introduction, the following events are modeled.

◇ Drop off luggage (denoted as D): Three pieces of evidence are used.

- E_1: The luggage did not appear -0.1 (or other small negative value) second ago.
- E_2: The luggage shows up now.
- E_3: The distance between the person and the luggage now is less than some threshold d_{drop_off} (*e.g.* 1.5 meters).

E_1 and E_2 together impose a temporal constraint: the luggage was carried by the human before the *drop-off* so the separate track of the luggage has not started yet. Once it appears, we know the *drop-off* has just happened.

E_3 is a spatial constraint: The owner has to be close to the luggage when *drop-off* just happened. This constraint eliminates irrelevant persons who are just passing by when *drop-off* takes place. In case that multiple persons are close to the luggage when *drop-off* takes place, the closest person is considered as the owner.

The prior and the conditional probabilities are listed as follows:

- $P(D) = P(\neg D) = 0.5$
- $P(E_i|D) = 0.99, i = 1, 2, 3$
- $P(E_i|\neg D) = 0.5, i = 1, 2, 3$

◇ Abandon luggage (denoted as A): Two pieces of evidence are used.

- E_1: The distance between H and L -0.1 (or other small negative value) second ago is less than the alarm distance d_{alarm} (=3 meters).
- E_2: The distance now is larger than d_{alarm}.

E_1 and E_2 together incorporate the **spatial rule** of PETS 2006 that is described in the introduction.

The prior and the conditional probabilities are listed as follows:

- $P(A) = P(\neg A) = 0.5$
- $P(E_i|A) = 0.99, i = 1, 2$
- $P(E_i|\neg A) = 0.5, i = 1, 2$

Sequence	1	2	3	4	5	6	7
# of persons	1	2	1	2	1	2	6
# of persons	1	2	1	2	1	2	5
# of luggage items	1	1	1	1	1	1	1
# of luggage items	1	1	1	1	1	1	1
Luggage location	(0.22,-0.44)	(0.34,-0.52)	(0.86,-0.54)	(0.24,-0.27)	(0.34,-0.56)	(0.80,-0.78)	(0.35,-0.58)
Luggage location	(0.11,-0.22)	(0.02,-0.44)	(1.10,-0.31)	(0.40,-0.10)	(0.27,-0.30)	(0.45,-0.53)	(0.2,-0.3)
Warning triggered time	2825 (113.0)	2280 (91.2)	none	2585 (103.4)	2749 (110.0)	2403 (96.1)	2317 (92.7)
Warning triggered time	2834 (113.4)	2278 (91.1)	none	2577 (103.1)	2743 (109.7)	2407 (96.3)	2310 (92.4)
Alarm triggered time	2843 (113.7)	2296 (91.8)	none	2602 (104.1)	2764 (110.6)	2422 (96.9)	2349 (94.0)
Alarm triggered time	2848 (113.9)	2298 (92.0)	none	2599 (104.0)	2757 (110.3)	2423 (96.9)	2350 (94.0)

Table 1: The ground-truth (with dark background) and our result (with white background). Locations are measured in meter. The last four rows show the frame number and time (in second) when an even is triggered.

We first detect the *drop-off* event on all combination of the human-luggage pair and only those pairs with high probability are considered further by the $abandon$ event. This is how the **contextual rule** is applied.

For the **temporal rule**, we require that if current frame i gets a high probability of the $abandon$ event and if any previous frame $i-t$ (t within the range of 30 seconds) also gets high probability, the high probability at frame $i-t$ will be discarded. Finally, alarm events are triggered at the frames with high probability. PETS 2006 also defines a warning event, which is treated exactly the same here except for a smaller distance threshold (=2 meters).

4. Results and Evaluation

The following information is provided in the ground-truth data of PETS 2006 dataset.

- ◇ The number of persons and luggage involved in the alarm event
- ◇ The location of the involved luggage
- ◇ The frame (time) when the warning event is triggered
- ◇ The frame (time) when the alarm event is triggered

We compare our results with the ground-truth in Table 1. The rows with dark background are the ground-truth data.

Figure 6 shows the key frames (with tracking results) in which the *drop off luggage* and the *abandon luggage* event are detected. The 3D mapping on the ground plane is shown at the right side of each frame. The floor pattern is also provided as reference. Note that the frame number shown in the *abandon luggage* event is the frame number of the *alarm* event minus 750 (equivalent to 30 seconds in a video with 25 fps frame rate).

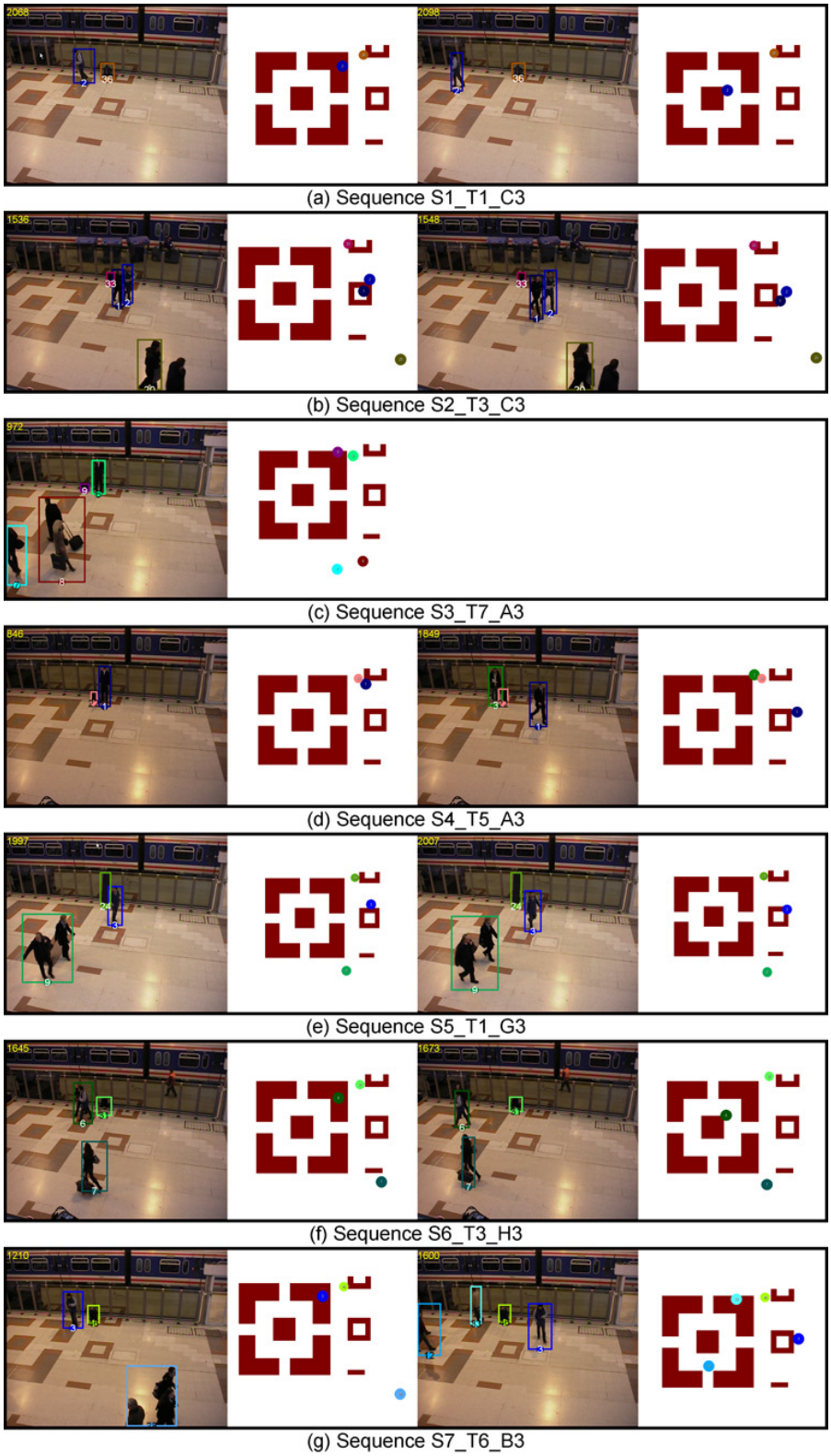
We can see in Figure 6, some errors on the ground plane are visible because the size and position of bounding boxes are not perfect and a small offset of a point in 2D can result in a significant error in 3D when the object is far away from the camera. But nonetheless, our tracking algorithms work very well. Note that in Figure 6(e), two persons on the left are not separated because they are merged in a single moving blob and the human detector fails to respond due to a large tilt angle.

The event recognition algorithm detects all (warning and alarm) events within an error of 9 frames (0.36 second). The involved persons include anyone who is close enough to the involved luggage when and after the *abandon luggage* event takes place. The correct number of involved persons in all sequences are detected, except for the last sequence. In that sequence, three persons are walking closely as a group. One of them is severely occluded by the other two and thus missed by the tracker.

The clear definition of the *alarm* event helps contribute to the good performance of our event recognition system. The contextual rule is crucial because detecting the *drop-off* event provides a filtering mechanism to disregard the irrelevant persons and luggage before being considered for the *abandon* event. Furthermore, sometimes our tracker misidentifies the type of an object based only on the mobility of the object, which can result in many false alarms. Due to the contextual rule, the possibility of such false alarms is significantly reduced. To illustrate the importance of the contextual rule, we list in Table 2 the number of triggered events without applying the contextual rule. This demonstrates that the high-level reasoning can eliminates the errors of low-level processing.

5. Conclusions and Future Work

We have presented an object tracking and event recognition system that has successfully tackled the challenge proposed by the PETS 2006 workshop. For tracking, we combine the results of a blob tracker and a human tracker, which provides not only the trajectory but also the type of each



(a) Sequence S1_T1_C3

(b) Sequence S2_T3_C3

(c) Sequence S3_T7_A3

(d) Sequence S4_T5_A3

(e) Sequence S5_T1_G3

(f) Sequence S6_T3_H3

(g) Sequence S7_T6_B3

Figure 6: The key frames (with tracking results) in which the *drop off luggage* and the *abandon luggage* event are detected. Figures shown here are, from left to right in each row, the key frame of *drop off luggage* in 2D and 3D, the key frame of *abandon luggage* in 2D and 3D, respectively.

Sequence	1	2	3	4	5	6	7
Total tracked persons	35	30	18	21	23	28	45
Total tracked luggage	1	8	1	1	4	1	1
Total drop off events	1	2	1	1	1	1	1
Total warning events	1	2	0	2	3	1	10
Total alarm events	2	4	0	1	2	1	10

Table 2: Without the contextual rule, the results contain many false alarms.

tracked object. For event inference, events are modeled and recognized in a Bayesian inference framework, which gives perfect event recognition result even though the tracking is not perfect.

Note that our system is designed as a framework for general event recognition task and is capable (and has already been applied) of recognizing a much broader range of events. For this specific (*left-luggage*) task, we just used our existing system and did not make any additional effort except that we specified the event models (as described in Section 3.2). This was done immediately. In conclusion, our system provides quick solutions to event recognition tasks such as the *left-luggage* detection.

Real-world scenarios usually are more complicated than the one presented here, in terms of the number of involved persons and objects and the variation in execution style and duration. For future work, we need to consider more sophisticated algorithms to handle such complexities.

References

[1] D. Comaniciu, V. Ramesh and P. Meer. The Variable Bandwidth Mean Shift and Data-Driven Scale Selection. In ICCV 2001, Vol I: 438-445.

[2] L. Davis, V. Philomin and R. Duraiswami. Tracking humans from a moving platform. In ICPR 2000, Vol IV: 171-178.

[3] A. A. Efros, A. C. Berg, G. Mori and J. Malik. Recognizing Action at a Distance. In ICCV 2003, 726-733.

[4] I. Laptev and T. Lindeberg. Space-time interest points. In ICCV 2003, 432-439.

[5] F. Lv and R. Nevatia. Recognition and Segmentation of 3-D Human Action using HMM and Multi-Class AdaBoost. In ECCV 2006, 359-372.

[6] C. Papageorgiou, T. Evgeniou and T. Poggio. A Trainable Pedestrian Detection System. In Proc. of Intelligent Vehicles, 1998, 241-246.

[7] J. R. Peter, H. Tu and N. Krahnstoever. Simultaneous Estimation of Segmentation and Shape. In CVPR 2005, Vol II: 486-493.

[8] C. Rao, A. Yilmaz and M. Shah. View-Invariant Representation and Recognition of Actions. In IJCV 50(2), Nov. 2002, 203-226.

[9] B. Wu and R. Nevatia. Detection of Multiple, Partially Occluded Humans in a Single Image by Bayesian Combination of Edgelet Part Detectors. In ICCV 2005, Vol I:90-97.

[10] B. Wu and R. Nevatia. Tracking of Multiple, Partially Occluded Humans based on Static Body Part Detection. To appear in CVPR 2006.

[11] A. Yilmaz and M.Shah. Actions Sketch: A Novel Action Representation. In CVPR 2005, 984-989.

[12] T. Zhao and R. Nevatia. Tracking multiple humans in crowded environment. In CVPR 2004, Vol II: 406-413.

Evaluation of An IVS System for Abandoned Object Detection on PETS 2006 Datasets

Liyuan Li, Ruijiang Luo, Ruihua Ma*, Weimin Huang, and Karianto Leman
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore, 119613
{lyli,rjluo,wmhuang,karianto}@i2r.a-star.edu.sg, *Ruihua.Ma@sophia.inria.fr

Abstract

This paper describes a visual surveillance system for unusual event detection and its performance for abandoned object detection on PETS 2006 Benchmark Data. The system is composed of four major modules, i.e., Foreground Segmentation, Moving Object Tracking, Stationary Object Tracking, and Event Detection. The Foreground Segmentation comprises robust background subtraction and context-controlled background maintenance. In Moving Object Tracking, multiple moving individuals are tracked through crowds according to their dominant colors. When a stationary object is detected, a layer, or template, is built to track it through complete occlusions. Finite State Machines are implemented to detect unusual events. The test of the system for abandoned object detection on PETS 2006 Benchmark Data is described. The performance of left-luggage detection and object tracking are then evaluated.

1. Introduction

This paper describes a visual surveillance system and its performance for abandoned object detection on PETS 2006 Benchmark Data. The Intelligent Visual Surveillance (IVS) system is developed for real-time detection of unusual events in public places, as for example, airports and railway stations. The interesting events include unattended objects, possible theft, loitering, and unusual gathering.

In this decade, several smart visual surveillance systems have been developed, such as W4 [6], VSAM [1], SensorForest [4], and PeopleVision [5] developed in US and ADVISOR [11] developed in Europe. The basic technologies for video-based surveillance are object detection, tracking, classification and event recognition.

Foreground object detection is normally the first step in visual surveillance and adaptive background subtraction (ABS) is widely used for this purpose [3, 12, 14]. These methods build and maintain a statistical model of background at each pixel. This model is subsequently applied to detect the foreground pixels in the incoming frames. In this system, a robust adaptive background subtraction method based on Principal Feature Representation (PFR) [7] is used. To run a surveillance system around the clock in a real public places, the background model has to be updated regularly over time. When some parts of background are frequently occluded by foreground objects, some foreground features will be learned into background model, and this will result in degradation of foreground detection subsequently. In this system, an approach to control the updating of pixel-level background model according to the perception of global contextual background features is used to deal with such problems [9].

Object tracking plays an important role in achieving semantic understanding of events for video surveillance. The challenges for tracking multiple objects in busy public places are the frequent overlaps of multiple targets. Most existing surveillance systems employ a single model, e.g. template [6], to track each object in the scene. Kalman filter or particle filters are helpful to deal with short-term occlusion [15]. Mean-shift is another efficient method to tackle partial occlusion [2]. In our proposed system, two models of different details are used to track moving objects and stationary objects separately. For a moving object with constant changes of position, size, pose, and shape, a global color distribution model, i.e. Principal Color Representation (PCR), is use to characterize its appearance, whereas for a stationary object, e.g., a bag, suitcase, or backpack, a layer model (or a template) is used to represent the 2D image of the object. A deterministic method for multi-object tracking based on PCR is employed to track moving objects through the crowds [8]. Meanwhile, for a stationary object, a layer model with a fixed position and complete image can be used to estimate the state of the object, e.g., exposed, occluded, moving again, or removed. This is helpful to track stationary objects through complete occlusions.

From the tracking result, several formal languages can be

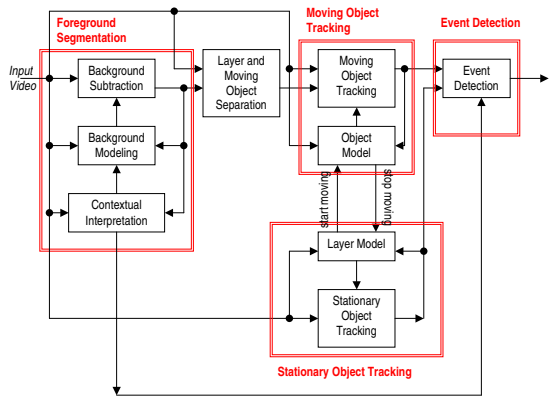


Figure 1. The block diagram of the system.

used to recognize events in the scene [10]. In this system, a set of Finite State Machines (FSM) are implemented to recognize the interesting events. FSM is chosen because of its efficiency, flexibility, and ease of implementation [13].

The goal of this paper is to evaluate the performance of an IVS system for detecting left-luggage on PETS 2006 Benchmark Data. The rest of the paper is organized as follows. First, the structure of the proposed system is briefly described in Section 2. The algorithms for foreground segmentation are described in Section 3. The tracking of multiple moving objects is described in Section 4, and the tracking of stationary objects is described in Section 5. Section 6 describes the implementation of FSMs for event detection. Experimental results on PETS 2006 Benchmark Data is presented in Section 7 and conclusions are given in Section 8.

2. System Overview

The IVS system is composed of four major modules, i.e., Foreground Segmentation, Moving Object Tracking, Stationary Object Tracking, and Event Detection. The block diagram is shown in Figure 1. By Foreground Segmentation, all the foreground objects, both moving and stationary objects, are extracted. The moving and stationary objects are then separated according to existing layer models. Then, the moving objects are tracked based on their global color distribution models and the stationary objects are tracked based on template comparison. The tracking results are fed to the Event Detection module to find interesting events. Details are described in the following sections.

3. Foreground Segmentation

This module contains adaptive background subtraction and context-controlled background maintenance.

PFR-Based Adaptive Background Subtraction

In this algorithm, the Principal Feature Representation (PFR) at each pixel is used to characterize the background appearances. The principal features are the most significant and frequent features observed at a pixel. Three types of features, i.e., spectral, spatial, and temporal features, are employed for background modelling. For each type of feature, a table which records the principal feature vectors and their statistics at a pixel $\mathbf{x}$

$$T_{\mathbf{v}}(\mathbf{x}) = \{p^t_{\mathbf{v}}(b), \{S^t_{\mathbf{v}}(i)\}^{N_{\mathbf{v}}}_{i=1}\} \quad (1)$$

is built, where $p^t_{\mathbf{v}}(b)$ is the prior probability of $\mathbf{x}$ being background with feature $\mathbf{v}$ and $S^t_{\mathbf{v}}(i)$ records the statistics of the $N_{\mathbf{v}}$ most significant feature vectors at $\mathbf{x}$, Each $S^t_{\mathbf{v}}(i)$ contains three components: $S^t_{\mathbf{v}}(i) = \{\mathbf{v}_i, p^t_{\mathbf{v}_i}, p^t_{\mathbf{v}_i|b}\}$, where $p^t_{\mathbf{v}_i} = P_{\mathbf{x}}(\mathbf{v}_i)$ is the prior probability of observing the feature vector $\mathbf{v}_i$ at pixel $\mathbf{x}$ and $p^t_{\mathbf{v}_i|b} = P_{\mathbf{x}}(\mathbf{v}_i|b)$ is the conditional probability of observing $\mathbf{v}_i$ as the background. The $S^t_{\mathbf{v}}(i)$ in the table are sorted in descending order with respect to the value $p^t_{\mathbf{v}_i}$. Hence, the first $N_{\mathbf{v}}$ elements are used as principal features. Three types of features are used, i.e., spectral feature (color), spatial feature (gradient), and temporal feature (color co-occurrence). Among them, the first two are used for static background pixels and the last is used for dynamic background pixels. The color vector is $\mathbf{c}_t = (R_t, G_t, B_t)$ from the input color frame. The gradient vector is $\mathbf{e}_t = (g^t_x, g^t_y)$ obtained by the Sobel operator. The color co-occurrence vector is $\mathbf{cc}_t = (R_{t-1}, G_{t-1}, B_{t-1}, R_t, G_t, B_t)$ with 32 levels for each color component.

The pixel is classified as background or foreground according to a Bayesian rule. Let the feature $\mathbf{v}_t$ be observed at $\mathbf{x}$ and time t. For a stationary pixel, the Bayesian rule is

$$2P_{\mathbf{x}}(\mathbf{c}_t|b)P_{\mathbf{x}}(\mathbf{e}_t|b)P_{\mathbf{x}}(b) > P_{\mathbf{x}}(\mathbf{c}_t)P_{\mathbf{x}}(\mathbf{e}_t) \quad (2)$$

For a dynamic pixel, the Bayesian rule is

$$2P_{\mathbf{x}}(\mathbf{cc}_t|b)P_{\mathbf{x}}(b) > P_{\mathbf{x}}(\mathbf{cc}_t) \quad (3)$$

The background model is updated with two operations. First, the principal features and their statistics at each pixel are updated gradually. The updated elements in each table are resorted in a descending order with respect to $p^{t+1}_{\mathbf{v}_i}$ to ensure the table always keep the first $N_{\mathbf{v}}$ most significant features observed at the pixel $\mathbf{x}$. When some new features become very significant at pixel $\mathbf{x}$, they will be tuned as new background features. Meanwhile, a reference background image is also maintained according to the segmentation result. This image is used to remove non-change pixels with simple image difference.

Context-Controlled Background Maintenance

In this system, the distinctive global features of some contextual background regions (e.g. the ground surface, an

ATM machine, etc.) are exploited to reason whether some background parts are exposed, and this perception result is used to control the maintenance of pixel-level background models [9]. Two types of contextual background regions (CBR) are exploited.

Type-1 CBR: A type-1 contextual background object is a fixed facility for the public, like an ATM machine, a counter, or a fixed chair. Its appearance model is extracted from the whole region in a snapshot. Two appearance descriptors are used for it. One is the *Orientation Histogram Representation* (OHR) to describe the structural features of a region and the other is *Principal Color Representation* (PCR) to describe its distinctive colors. Let H^i_{b1} be the OHR of the type-1 CBR R^i_{b1} and H_t be the OHR from the corresponding region in current frame. The likelihood of observing R^i_{b1} at current frame based on OHR is

$$P_L(H_t|H^i_{b1}) = (2 < H^i_{b1}, H_t >)/(\|H^i_{b1}\|^2 + \|H_t\|^2) \quad (4)$$

Let T^i_{b1} be the PCR of R^i_{b1} and T_t the PCR from the corresponding region in current frame. The likelihood of observing R^i_{b1} at current frame based on PCR is

$$P_L(T_t|T^i_{b1}) = \min\{P(T_t|T^i_{b1}), P(T^i_{b1}|T_t)\} \quad (5)$$

where $P(T_t|T^i_{b1})$ and $P(T^i_{b1}|T_t)$ are calculated according to (9). Fusing OHR and PCR, the log likelihood of observing R^i_{b1} at time t is

$$L^{i,t}_{b1} = \omega_s \log P_L(H_t|H^i_{b1}) + (1-\omega_s)\log P_L(T_t|T^i_{b1}) \quad (6)$$

where $\omega_s = 0.6$ is chosen empirically. If $L^{i,t}_{b1}$ is smaller than a threshold T_{L1}, it is occluded at the time.

Type-2 CBR: A type-2 CBR is a large homogeneous region in the scene, such as ground or wall surfaces. Only PCR descriptor is used for each of them. The likelihood of observing a pixel $\mathbf{x}$ as a point of type-2 CBR R^i_{b2} is evaluated on a 5×5 window $R_t(\mathbf{x})$ as

$$P(R_t(\mathbf{x})|R^i_{b2}) = \frac{1}{|R_t(\mathbf{x})|}\sum_{\mathbf{s}\in R_t(\mathbf{x})} \mathcal{B}(I_t(\mathbf{s})|R^i_{b2}) \quad (7)$$

where $\mathcal{B}(I_t(\mathbf{s})|R^i_{b2})$ equals 1 if the color belongs to the PCR. The log likelihood is $L^{i,t}_{b2}(\mathbf{x}) = \log P(R_t(\mathbf{x})|R^i_{b2})$. If it is larger than a threshold T_{L2}, $I_t(\mathbf{x})$ belongs to R^i_{b2}.

Control the pixel-level background updating: Three learning rates, i.e., high, normal, and low rates, are used to control the pixel-level background updating. If a type-1 CBR or part of type-2 CBR is occluded, a low learning rate $\alpha_t = 0$ is used for the all foreground regions which occluding the CBRs. If the CBRs are visible and no foreground region is detected by background subtraction, a normal learning rate ($\alpha_t = \alpha$) is used for the points. If a CBR or part of it is visible but there are foreground regions overlapp it, a high learning rate ($\alpha_t = 2\alpha$) is used to recover the pixel-level background models.

4. Moving Object Tracking

The distinctive colors are used to characterize the appearance of a moving object. The PCR model for the nth foreground region R^n_t at time t is

$$T^n_t = \{s_n, \{E^i_n = (\mathbf{c}^i_n, s^i_n)\}^N_{i=1}\} \quad (8)$$

where s_n is the size of R^n_t, $\mathbf{c}^i_n = (r^i_n\ g^i_n\ b^i_n)^T$ is the RGB values of the ith most significant color, and s^i_n is the significance value of $\mathbf{c}^i_n$ for the region. The components E^i_n are sorted out in descending order of significance values.

Let O^m_{t-1} be the mth tracked object described by its PCR $\mathcal{T}^m_{t-1} = \{s_m, \{E^i_m = (\mathbf{c}^i_m, s^i_m)\}^N_{i=1}\}$ obtained previously when it was an isolated object. Then, the likelihood of observing object O^m_{t-1} in region R^n_t can be derived as

$$P(R^n_t|O^m_{t-1}) = \frac{1}{s_m}\sum_{i=1}^{N}\min\left[s^i_m, \sum_{j=1}^{N}\delta(\mathbf{c}^i_m, \mathbf{c}^j_n)s^j_n\right] \quad (9)$$

To adapt to scale variations, the likelihood based on normalized PCRs is also computed as $\widetilde{P}(R^n_t|O^m_{t-1})$. Then, the scale-adaptive likelihood is defined as

$$P^*(R^n_t|O^m_{t-1}) = \max\{P(R^n_t|O^m_{t-1}), \widetilde{P}(R^n_t|O^m_{t-1})\} \quad (10)$$

With the foreground regions extracted by background subtraction, the directed acyclic graphes (DAGs) can be applied to connect the regions in the consecutive frames. Let the regions be denoted as the nodes and be laid in two layers: the parent layer and the child layer. The parent layer consists of the regions $\{R^j_{t-1}\}^{N_{t-1}}_{j=1}$ in the previous frame, and the child layer consists of regions $\{R^k_t\}^{N_t}_{k=1}$ in the current frame. If there is overlap between R^j_{t-1} and R^k_t, a directional link $l_{jk} = 1$ is built. A directed acyclic graph (DAG) is formed by a set of nodes in which every node connects to one or more nodes in the same group.

If a DAG contains a group of foreground objects, multi-object tracking has to be performed. This can be formulated as a Maximum A Posterior (MAP) problem and heuristically decomposed as two sequential steps from coarse to fine. The coarse *assignment* process assigns each object in a parent node to one of its child nodes while the fine *location* process determines the new positions of the objects assigned to each child node.

According to Bayesian estimation, the assignment of objects from a parent node to its child nodes can be performed one-by-one sequentially from the most visible object to the least visible object. First, the depth order of the objects in a parent region (a group) is estimated. Then objects are assigned iteratively. The assignment is determined according to the PCR-based likelihood and the shift of position. At the end of each iteration, the exclusion is performed to the

child node to which the object has been assigned to remove the visual evidence of the assigned object from the PCR of the region.

If more than one object have been assigned to one child region, the location process is performed to find their new positions in the group. Again, according to Bayesian estimation, the objects in a group are located one-by-one sequentially from the most visible to the least visible ones. First, the depth order of the objects is estimated. Then, a PCR-based mean-shift algorithm is performed to find the new position of it. At the end of each iteration, the exclusion is performed to remove the visual evidence of the located object from the corresponding position in the region.

5. Stationary Object Tracking

A deposited object is usually smaller than persons and groups. Hence, it is easy to be lost by tracking algorithm when it is almost completely occluded. However, once it is deposited, it becomes stationary completely. A layer model, or template, is used to track small stationary objects.

Once a small stationary object is detected, a layer model, i.e. an image of the object, is built with the fixed position. In the subsequent frames, the pixel-level color and gradient differences are calculated and linearly transformed to fuzzy measures with $2\hat{\sigma}$ as moderate point (0.5), where $\hat{\sigma}$ is the estimated noise variances of colors or gradients. Meanwhile, the foreground regions overlapping the layer are also examined. If the difference measure of a layer object is low, it is exposed. The layer model, or the template is also updated with a very small learning rate. If the difference measure of a layer object is high and no foreground region overlaps it, it is detected as being removed from the position, otherwise, it is almost completely occluded. If the difference measure of a layer object is moderate and it is overlapped partially by large foreground regions, it is partially occluded, otherwise, it might start moving again. Once it is detected as being removed, its layer model will be deleted. If it is confirmed as moving again, its layer model will be deleted and it will be transformed as the moving foreground object in the same position with similar size and color features. With these estimated states, the layer object can be tracked through frequent occlusions.

6. Event Detection

An "event" is an abstract symbolic concept of what has happened in the scene. It is the semantic level description of the spatio-temporal concatenation of movements and actions of interesting objects in the scene. In this system, Finite State Machines (FSM) are used for event recognition. The input to the FSMs are the perceptive features generated by foreground detection and object tracking. For each valid object, the visual cues include size, shape, position, motion, class, stay-time, and relations with other objects.

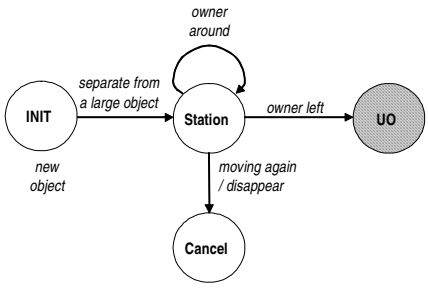


Figure 2. The FSM for "Unattended Object".

The FSM is designed for each specific event. As an example, the graph representation of the FSM for "Unattended Object" is shown in Figure. 2. This event involves two objects. The FSM is initialized for each new small object when it is separated from another large moving object. The ownership is established between the two objects. The FSM transits from state "INIT" to state "Station" once the small object becomes stationary. In the state, the object is associated with its owner. If the owner leaves the scene, the FSM transits from state "Station" to the goal state "UO" and an "Unattended Object" is detected.

7. Experimental Results

The IVS system has been run around the clock for three months in several busy public sites. At a resolution of 176×144, it ran at 10fps on average on a 2.8GHz standard PC. In this section, its performance for abandoned object detection on PETS 2006 Benchmark Data is presented. The parameter set used in this evaluation is the same as used for the testing in real scenes.

The PETS 2006 Benchmark Datasets are sequences containing left-luggage scenarios with increasing scene complexity in a real-world environment. The scenarios are captured by four cameras from different view points. To evaluate our system in a condition similar to that of the real tests, the sequences from camera 3 are chosen and the system takes one frame every three frames as the input (about 8.33fps). In the sequences from other three cameras, the left objects are too small to be distinguished from segmentation noise on 176×144 image resolution. Besides the real-time requirement, another reason to run on small image resolution is that less fragmentation would occur in the segmentation result. This is crucial to reduce subsequent tracking errors. In this evaluation the abandoned object is detected only when the owner has gone out of the scene. Using this

definition can reduce the false alarms significantly in busy public sites compared to the definition based on the distance between the owner and the left-luggage. The details of the test on each sequence are described as follows.

Dataset S1: This scenario contains a single person with a rucksack who loiters before leaving the item of luggage unattended. The subjective difficulty (SD) of this sequence is among the lowest (1-star). Four sample frames are shown in Figure 3 in which each tracked object is overlapped by a bounding box with an ID number over it. The 1st image shows the frame S1-T1-C.01180 in which a person (ID=14) enters the scene with a rucksack. The 2nd image, i.e. S1-T1-C.01288, shows that the person (ID=14) loiters in front of the fence. In the 3rd image (S1-T1-C.02242), the person (ID=14) left his rucksack (ID=43) beside the fence and is leaving the scene. The last image shows the frame S1-T1-C.02284 at the time the owner just leaving the scene. The rucksack is detected as an unattended object.

Dataset S2: This scenario contains two people who entering the scene from opposite directions. They meet, stay there, talk to each other and then leave together. One person places a suitcase on the ground before leaving. The SD of this sequence is moderate (3-star). Four sample frames are shown in Figure 4. In the 1st image (frame S2-T3-C.00931), one person (ID=22) enters the scene with a suitcase. The 2nd image, i.e. S2-T3-C.01048, shows the scene when the second person (ID=29) enters the scene and the two persons meet each other. The two persons stay there and talk to each other as shown in the 3rd image, i.e. S2-T3-C.01375. The last image (S2-T3-C.01699) shows the instant of the two people leaving together with the suitcase left in the place where they once stood. It can be seen that about 10s before the two people moving away, the background starts to change due to the moving of the trolley coach. The cluttered foreground regions result in tracking errors. Hence, the system fails to detect the abandoned object.

Dataset S3: This scenario contains a person waiting for a train. The person temporarily places his briefcase on the ground. He then picks it up again and moves to a nearby shop. The SD of this sequence is among the lowest (1-star). Four sample frames are shown in Figure 5. In the 1st image (frame S3-T7-A.00946), the person (ID=5) enters the scene with a briefcase. The 2nd image (S3-T7-A.01354) shows the scene of the person stands in front of the fence with the briefcase (ID=13) placed beside him on the ground. The 3rd image (S3-T7-A.01450) shows the instant of the person picking up the briefcase again and the last image (S3-T7-A.01510) shows the moment that the person is leaving the scene. In this scenario, even a static object (ID=13) is detected. Since its owner is always in side, no false alarm of unattended object is triggered.

Dataset S4: This scenario contains a person placing a suitcase on the ground. Following this a second person arrives and talks with the first person. The first person leaves the scene without his luggage. Distracted by a newspaper, the second person does not notice that the first person's luggage is left unattended. The SD of this sequence is high (4-star). Four representative frames from the sequence are shown in Figure 6. The 1st image (frame S4-T5-A.00988) shows the first person (ID=2) remaining with his luggage. In the 2nd image, i.e. S4-T5-A.01672, another person (ID=6) meets him and talks with him. The 3rd image (S4-T5-A.01984) shows the instant that the first person (ID=2) is leaving without his luggage (ID=16). In the last image (S4-T5-A.02074), the unattended object is detected when the first person has left the scene.

Dataset S5: This scenario contains a single person with ski equipment who loiters before abandoning the item of luggage. The SD of this sequence is low (2-star). Four sample frames from the sequence are shown in Figure 7. The first two images, i.e. frames S5-T1-G.01150 and S5-T1-G.01405, show the person (ID=2) entering and loitering in the scene. The 3rd image shows the moment that the person leaves the scene while his luggage (ID=16) is placed against the fence. In the last image, the luggage is still detected and the owner has gone, but no alarm of unattended object is triggered. This is because the luggage looks like a side view of a standing person. This scenario is very similar to that two persons stand together, one is almost completely occluded by another, and then, two persons separate, one stands there and another leaves. Allowing the detection of this event would result in many false alarms in real-world public places.

Dataset S6: This scenario contains two people who entering the scene together. They stay there together and talk to each other for a while. Before they leave the scene together, one person places a rucksack on the ground. The SD of this sequence is moderate (3-star). Eight sample frames from the sequence are shown in Figure 8. The first two images of the upper row (frames S6-T3-H.00988 and S6-T3-H.01087) show the two people (ID=6) entering the scene and staying there together as a group. In the next image (S6-T3-H.01384), they are separated as two persons, one having the original ID (ID=6) and the other having a new ID (ID=15). Then, they are involved in an overlapping of five persons (IDs:6,13,14,15,16) as shown in the last image of the upper row (S6-T3-H.01420) and the first image of the lower row (S6-T3-H.01438). When separated from the group, the two people are detected as a group and the ID of the front person (ID=15) is used for them, as shown in the first two images of the lower row (S6-T3-H.01438 and S7-T6-B.01873). The last two images of the lower row show the two people (ID=15) leaving the scene with a rucksack (ID=21) left (S7-T6-B.0194) and an alarm of unattended object is triggered when they go out of the scene (S7-T6-B.02029).

Figure 3. Dataset S1 (Take 1-C, Subjective Dif culty : 1): In this scenario, a person (ID=14) enters the scene, loiters, leaves a rucksack (ID=43) and goes away.

Figure 4. Dataset S2 (Take 3-C, Subjective Dif culty: 3-star): In this scenario, two persons (ID=22 and ID=29) meet, stay there and talk, leave a suitcase and go away.

Figure 5. Dataset S3 (Take 7-A, Subjective Dif culty: 1-star): In this scenario, a person (ID=5) enters, puts a briefcase (ID=13) on the ground and stays there, then picks up the briefcase and goes away.

Figure 6. Dataset S4 (Take 5-A, Subjective Dif culty: 4-star): In this scenario, a person (ID=2) enters with his luggage, another person (ID=6) meets him, then he leaves without his luggage (ID=16).

Figure 7. Dataset S5 (Take 1-G, SD: 2-star): A person (ID=2) leaves a ski equipment (ID=16).

Figure 8. Dataset S6 (Take 3-H, SD: 3-star): Two people (ID=6, ID=15) leave a rucksack (ID=21).

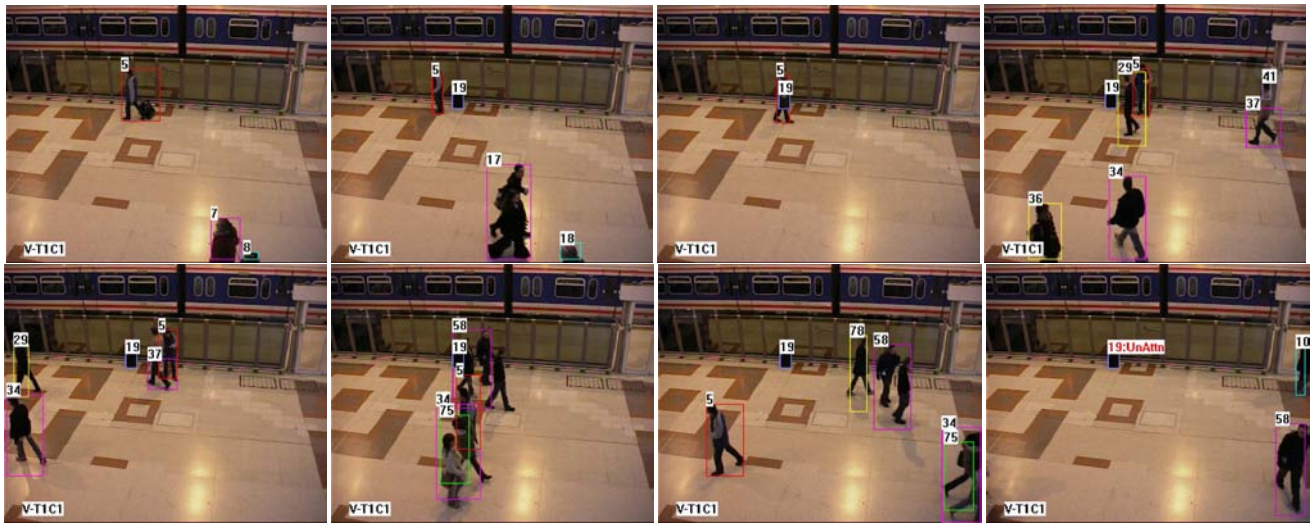


Figure 9. Dataset S7 (Take 6-B, SD: 5-star): One person (ID=5) leaves a luggage (ID=19), loiters and then goes away through crowds.

Dataset S7: This scenario contains a single person with a suitcase. The person enters the scene, leaves the luggage, loiters and then leaves the scene. During the time five other people move in close proximity to the item of the luggage. Hence, this scenario is considered as the most difficult one in this Benchmark dataset, i.e., the SD is 5-star. Eight sample frames from the sequence are shown in Figure 9. In the upper row, the 1st image (frame S7-T6-B.00760) shows that the actor (ID=5) is entering the scene with a suitcase. The actor then places the suitcase (ID=19) on the ground and loiters it, as shown in the 2nd and 3rd images (S7-T6-B.01339 and S7-T6-B.01498). During the time, the actor and the suitcase have been occluded by two persons (ID=29 and ID=37) in close proximity, as shown in the last image of the upper row (S7-T6-B.01630) and the first image of the lower row (S7-T6-B.01702). After that, they are involved in merging and separation of 6 people, i.e., a group of 3 people (ID=58) and a group of 2 people (ID=34 and ID=75), as shown in the 2nd image of the lower row (S7-T6-B.01873). The 3rd image in the lower row (S7-T6-B-.01942) displays the moment when the actor moves past the crowds and is leaving the scene. The last image (S7-T6-B.02029) shows the instant of the alarm being triggered when the actor has just gone out of the scene.

Summary: The accuracy rate of left-luggage detection on the dataset of camera 3 from PETS 2006 Benchmark Data is 5/7=71.4% and no false alarm is given. Among the two missed events, one is caused by the significant background changes just behind the actors, and the other is due to that the object shape looks like human beings. It is worthy to note that this result is achieved without any specified parameter about the scene. If some specified information about the scene is used, better result can be obtained. As an example, if all the foreground objects behind the fence are removed according to the calibration information and a human-like object without motion over 30 seconds will be classified as a non-human object, all the seven events are detected correctly without any false alarm (i.e. 100% accuracy rate). The demo videos obtained without and with specified parameters are all available at http://perception.i2r.a-star.edu.sg/PETS2006/UnAttnObj_C3.htm.

We also evaluated the performance of object tracking for the system. In this evaluation, only valid objects are counted. The invalid objects include those behind the fence, the persons who do not move into the scene, the separated shadows and small random clutters. If the ID number of a valid object has changed n times during the time of its appearance in the scene, n times of tracking errors are observed. For the 7 sequences from camera 3, 123 valid objects have been observed, and 9 tracking errors have been observed. Hence, the error rate of tracking is 9/123=7.32%. It is noted that there is no top of this rate since the ID number of a valid object might change more than one time.

The performance of both event detection and object tracking agrees with what we obtained from real tests on natural public places.

8. Conclusions

In this paper, a distinctive visual surveillance system for unusual event detection in natural public places is described. Its performance for abandoned object detection on PETS 2006 Benchmark Data is evaluated. The result is encouraging. The future work is to improve the efficiency of the system to be able to work on high resolution images in real-time.

References

[1] R. Collins and et al. A system for video surveillance and monitoring. *VSAM Final Report*, 2000.

[2] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Trans. PAMI*, 25:564–577, 2003.

[3] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. *Proc. ECCV*, 2:751–767, 2000.

[4] W. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. *Proc. CVPR*, pages 22–29, 1998.

[5] A. Hampapur and et al. Smart video surveillance: Exploring the concept of multiscale spatiotemporal tracking. *IEEE Signal Processing Magazine*, 22(2):38–51, 2005.

[6] I. Haritaoglu, D. Harwood, and L. Davis. W^4: Real-time surveillance of people and their activities. *IEEE Trans. PAMI*, 22(8):809–830, 2000.

[7] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian. Statistical modeling of complex background for foreground object detection. *IEEE Trans. IP*, 13(11):1459–1472, 2004.

[8] L. Li, R. Luo, I. Y. H. Gu, W. Huang, and Q. Tian. Pcr-based multi-object tracking for video surveillance. *IEEE Workshop Visual Surveillance*, pages 137–144, 2006.

[9] L. Li, R. Luo, W. Huang, and H. Eng. Context-controlled adaptive background subtraction. *submitted to IEEE Workshop PETS*, 2006.

[10] N. Rota and M. Thonnat. Video sequence interpretation for visual surveillance. *IEEE Workshop Visual Surveillance*, pages 59–68, 2000.

[11] N. Siebel and S. Maybank. The advisor visual surveillance system. *Proc. ECCV Workshop on Applications of Computer Vision*, pages 103–111, 2004.

[12] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. PAMI*, 22(8):747–757, 2000.

[13] P. Winston. *Arti cial Intelligence*. Addison-Wesley, 1993.

[14] C. Wren, A. Azarbaygaui, T. Darrell, and A. Pentland. *P nder* : Real-time tracking of the human body. *IEEE Trans. PAMI*, 19(7):780–785, 1997.

[15] C. Yang, R. Duraiswami, and L. Davis. Fast multiple object tracking via a hierarchical particle filter. *Proc. ICCV*, 1:212–219, 2005.

Abandoned Object Detection in Crowded Places

Sadiye Guler *Member IEEE* and Matthew K. Farrow
intuVision, Inc.
100-F Tower Office Park Woburn, Massachusetts 01801
{sadiye,matt }@intuvisiontech.com

Abstract— **This paper presents a robust and practical method for automatic detection of abandoned objects from surveillance video in crowded areas. Our method introduces a unique combination of a moving object tracker focusing on object splits that potentially lead to a "drop-off" event and a novel stationary object detector for quickly spotting and persistently maintaining the unattended objects even with occlusions of the object by multiple people. We use a background subtraction based tracker and mark the projection point of the center of mass of each object on the ground. The tracker identifies object splits and qualifies them for possibility of a "drop-off" event. Our stationary object detector running in parallel with the tracker quickly identifies potential stationary foreground objects by watching for pixel regions, which consistently deviate from the background for a set duration of time. The stationary objects detected are correlated with drop-off events and the distance of the owner from the object is used to determine warnings and alerts for each camera view. The abandoned objects are correlated within multiple camera views using the location information and a time weighted voting scheme between the camera views is used to issue the final alarms to eliminate the effects of view dependencies and achieve the best results.**

1. Introduction

The recent threats to security of public facilities have increased the need for automatic detection of certain classes of events such as left or abandoned objects from video surveillance data in real-time. Detection of abandoned objects has been one of the most sought after events to be detected automatically since the early days of smart video surveillance systems. Several algorithms have been developed from research prototypes as in [3] to commercial product modules [2] for this purpose.

This material is based upon work funded in part by the U. S. Government. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Government.

Most existing algorithms tend work well with "blocks world" type scenarios where scenes include only a few people, the objects are abandoned in clear view of the camera and demonstrative alarms are generated within seconds of object being put down. In practice however, especially when there is harmful intent, these events do not happen in an obvious manner, rendering these algorithms of little use in crowded areas.

In [6] a Bayesian multi people tracker is used together with a blob-based object detector to detect abandoned objects and provide warnings and alerts to assist to a human operator. In [8] an event detection system for indoors surveillance applications consisting of object extraction and event detection modules is presented. The event detection module classifies objects including abandoned objects using a neural network. The system is limited to detecting one abandoned object event in unattended environments. These methods are developed considering only the algorithmic aspect of abandoned object detection expecting these events to reasonably follow a model.

Recently event detection methods that deal with the crowded nature of the public transport system surveillance applications is presented in [1], and more refined analysis of foreground-background regions and layers to identify abandoned objects have been described in [9]. In [1] an event detection algorithm based on object trajectories, designed for CCTV surveillance systems is presented. Following the foreground segmentation, basic characteristics of blob and scenes such as blob position or speed and people density are used to create low-level descriptions of predefined events including abandoned objects.

The focus of this paper is those applications in which bags and suitcases are abandoned in crowded places when there are several people around the immediate area. The owners usually move around the object they are going to leave behind pretending checking schedules, or asking someone a question. The existence of several people around not only occludes the object but also makes it difficult to maintain the object to owner correspondence for video trackers.

Motivated by the idea of humans' use of dedicated processing paths for frequently needed visual tasks we

developed the concept of dedicated simple event component detectors for specific video understanding tasks. To this end, for detecting abandoned objects we propose to use a stationary object detector in conjunction with an object tracker. The object tracking and stationary object detection are conducted in parallel and potential object drop-off events are correlated with candidate-abandoned objects. We use a background subtraction based video tracker. Although a crucial component, tracking is not the novel part of our proposed approach. Our proposed stationary object detection algorithm operates by watching for foreground regions that consistently deviate from the background for duration of time.

The remainder of the paper is organized as follows. After this introduction we provide a high-level description of our approach in Section 2. The moving object tracking, the stationary object detection and abandoned object event detection algorithms are described in Section 3. Section 4 discusses the experimental results followed by the concluding remarks and future directions in Section 5.

2. Abandoned Object Detection Method

Our method consists of two major components: a video tracker to detect and track moving objects, and a stationary object detector for quickly spotting and persistently detecting the abandoned objects.

As illustrated in Figure 1, the tracker keeps track of all object-split events that may lead to an object "drop-off" event. The tracker qualifies object splits for possibility of a "drop-off" event by using the parent/child object association and the distance between the parent and spawned objects. A stationary object detector running in conjunction with the moving object tracker quickly identifies potential stationary foreground objects by using background statistics with a duration parameter to persistently detect the stationary objects through multiple occlusions.

The detected stationary objects are then correlated with drop-off events and the distance of the owner from the object is used to declare warnings and alerts for each camera view. Final abandoned object detection results are obtained by fusing the information from these detectors and over multiple cameras. The left behind objects are correlated among multiple camera views using the location information and a time and object location based voting scheme as described in Section 3.3 to issue the final alarms. As shown in Figure 1 the abandoned object warnings and alarms generated for each camera contributes to overall decision using the voting scheme.

The object tracking, stationary object detection and tracked and stationary object fusing algorithms for abandoned object detection are discussed in the next section.

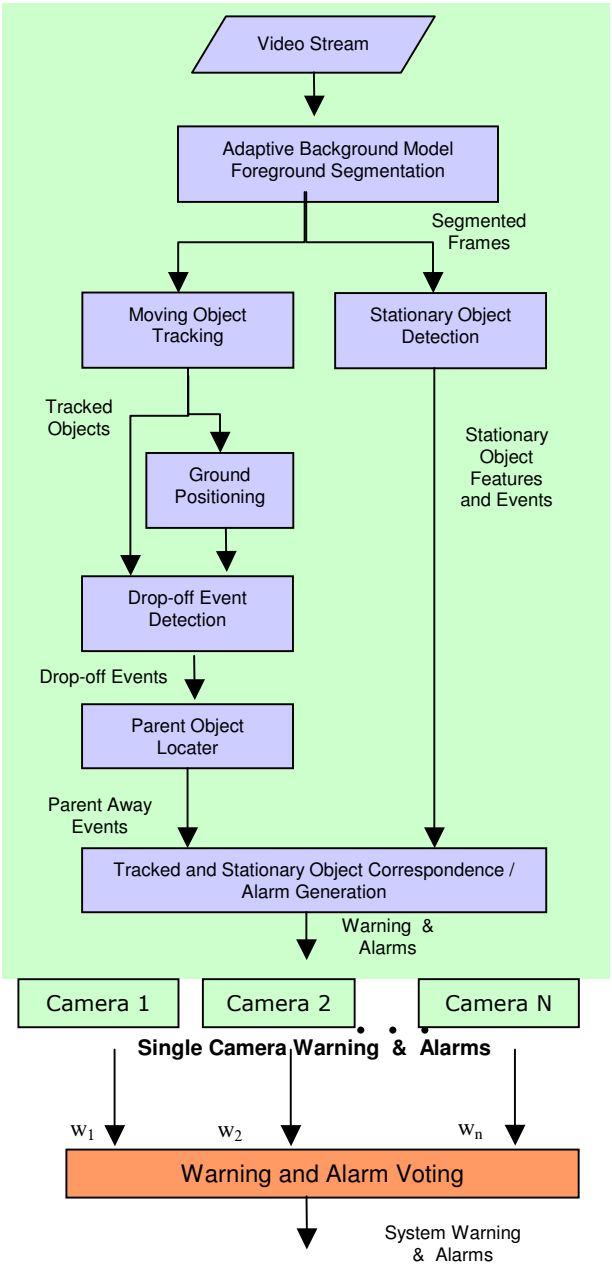


Figure 1. Abandoned Object Detection Method.

3. Object Tracking and Stationary Object Detection

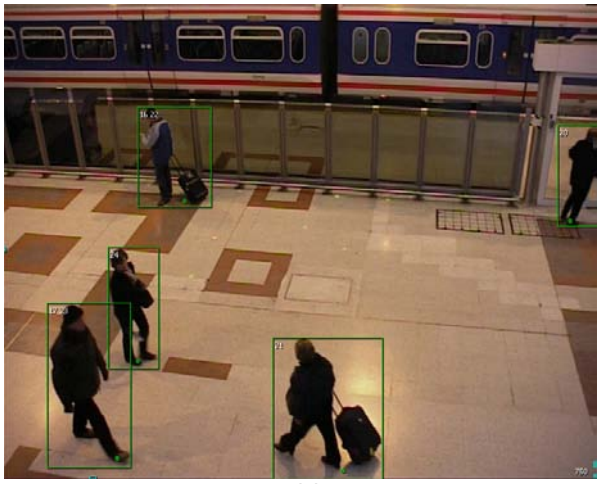
To detect the split and drop-off events in a video sequence the tracking of objects has to be performed first. A high level description of the tracker is provided next in Section 3.1 for completeness. Our proposed stationary object detection algorithm is explained in Section 3.2. The fusing of the information from these detectors is discussed in Section 3.3.

3.1. Object Tracking and Drop-off Event Detection

We employ a common background subtraction based tracker with an adaptive background model similar to that of [7]. Each pixel is assumed to be a sample from a mixture of Gaussians in R, G and B channels. The background subtraction step produces the candidate foreground pixels that are then grouped together into foreground regions using a distance and value based connected components algorithm. The shadow pixels are identified using an algorithm based on separating brightness from the chromaticity as in [4, 5]. Segmented foreground regions are then declared as objects and tracked from frame to frame, using motion, position and region based correspondence. An appearance model with features based on geometrical properties (such as size and aspect ratio) and color distribution is also used in object correspondence. Each detected object is assigned a unique ID number that should stay persistent during the life of the tracked object through splits and merges with other objects.
To accurately determine the position of tracked objects in each frame we project the object center of mass onto the ground plane and use this point depicted by the green circular marker in the lower portion of the bounding boxes in Figure 2 for calculating distances between objects.

The objects' tracks are then analyzed to detect object splits where one tracked object becomes two or more objects and identify the "drop-off" events. For example in Figure 2 (a) a person carrying a suitcase is tracked as a single object and in Figure 2 (b) an object split is observed resulting in the suitcase being spawned from the person.

Each object split is a candidate for a "drop-off" event and has to be qualified for such a possibility. In general object splits may occur due to a number of reasons such as people walking together may separate, a single tracked object may break into two or more regions due to tracking challenges or a person may leave a carried object as in "drop-off". To identify the drop-off events the sizes and the motion properties of the spawned object in each detected object split are examined. List of all potential drop-off events is created for further verification of parent object's distance from the dropped-off object. The distance between the parent and spawned object is determined using the ground plane calibration. The dropped off objects detected are to be matched with the list of stationary objects detected with the algorithm described next in Section 3.2.



(a)

(b)

Figure 2. Object Detection and Tracking illustrating the drop-off event: (a) Person with the suitcase before the split, (b) The suitcase is identified as a new object after the split.

3.2. Stationary Object Detection

While moving objects and potential drop-off events are detected and kept track of by the object tracker a stationary object detector runs in parallel using the same background model as the tracker and looks for stationary objects. Our stationary object detection algorithm looks for foreground pixels that deviate from the background for a duration of time. The algorithm produces an intermediate image called a *stationary object confidence image*, *S*, in which each pixel value represents the confidence in that pixel belonging to a stationary object. For display purposes the confidence values are mapped to [0, 255], 0 indicating a background pixel and 255 indicating a stationary object pixel. At the start of processing all pixels *s(x,y)* of the stationary object confidence image are initialized to 0. In the following *I* denotes the original frame image and *B* denotes the background model, both defined on 2-dimensional grid positions *(x,y)*. For each new frame, every pixel *s(x,y)* is updated based on a comparison of the corresponding band of the original frame image pixel *i(x,y)* with the corresponding background model $b_{(x,y)}$ (μ,σ) at pixel *(x,y)* and a duration based update function. An increment counter *C(x,y)* at each pixel keeps track of number of consecutive *i(x,y)* observations that do not fit the background model at that pixel. This counter is reset to zero when observed *i(x,y)* falls within the expected background model at *(x,y)*. Similarly a decrement counter *D(x,y)* is started each time *C(x,y)* is reset to keep track of number of consecutive *i(x,y)* observations that fit the background model at *(x,y)*. The following update functions truncated into [0, 255] are used to generate S:

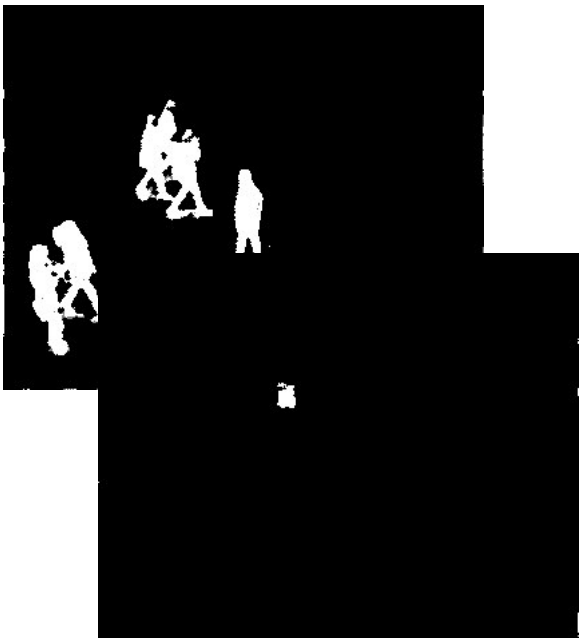
$$s(x,y) = s(x,y) + C(x,y)\ (\ 255 / \boldsymbol{t} \times FrameRate) \quad (1)$$
$$\text{if } i(x,y) \text{ does not fit } b_{(x,y)}\ (\mu,\sigma)$$

$$s(x,y) = s(x,y) - \boldsymbol{r}D(x,y)\ (\ 255 / \boldsymbol{t} \times FrameRate) \quad (2)$$
$$\text{if } i(x,y) \text{ fits } b_{(x,y)}\ (\mu,\sigma)$$

where $\boldsymbol{t}$ is the "drop-off event wait time" parameter in seconds for declaring a pixel stationary and $\boldsymbol{r}$ is a positive number to generate a fraction of time t for declaring a pixel non-stationary.

Pixels fall on a stationary object will not fit the background model and remain different from the background for as long as that object stays in place, causing the corresponding *s(x,y)* pixels to be incremented at every frame and to reach the maximum confidence value (255) by the time the object has been stationary for $\boldsymbol{t}$ seconds. While building the confidence for the stationary object pixels, color values for *i(x,y)* can be used to build a color distribution for the stationary object to distinguish the actual stationary object from the other occluding foreground objects. Moving objects occluding an abandoned object will not interfere with the detection of the stationary object as depicted with the blue bounding box in Figure 3(a) below.

(a)



(b)

Figure 3. Stationary Object Detection (a) Stationary object is preserved through occlusions caused by the people moving in front, (b) Processed image views with all foreground object (top) and the stationary object confidence image (bottom).

The pixels on the abandoned object will remain different from the background, regardless of which object they belong: a stationary object or an occluding moving object in the foreground. In Figure 3(a) the stationary object is almost fully occluded in that particular frame by the group of people passing in the front.

When the abandoned object is removed, the corresponding stationary object confidence image pixels will be updated so that they are turned "off" in t/r seconds. The update functions chosen above facilitate increasing rate of confidence building while the object is consistently stationary for a number of frames and not penalizing the confidence values severely due to intermittent observations that fall in the background model while the stationary object is still in the scene.

3.3. Abandoned Object Detection

The results from moving object tracking, object drop-off event detection need to be corresponded with stationary object detection for determining abandoned object warnings and alarms. The list of candidate drop-off events is compared and validated against the list of stationary objects detected. This dual detection mechanism helps reduce the effect of the occlusions (and hence the view dependence) by strengthening the abandoned object hypothesis using cues from both moving and stationary object analysis. A warning or alarm decision is made based on the owner and dropped object distance and time parameters using the PETS task definitions. We intend to use voting across the camera views for fusing alarm information from multiple cameras using a voting scheme based on abandoned object's location and warning and alarm frames. As new alarms and warnings are generated in each camera view, they are sent to a real time voting and correlation module.

Warning and alarm notifications will be identified as correlated if:

i. The corresponding warnings within t seconds of each other, and
ii. The abandoned object coordinates for the corresponding alarms/warnings are within d meters of each other.

When correlated notifications are received from two cameras, a global alarm or warning will be issued. If these two notifications are warnings then a global warning message is reported and the first subsequent alarm message will be reported as a global alarm. The results of using this voting scheme are illustrated in the next section.

4. Experimental Work

The methodology described in the previous sections has been implemented as real-time video analysis modules and experiments have been run on all the PETS 06 sequences with same parameter settings except for one time parameter in one of the camera views. The data set includes 28 video sequences from 4 camera views and 4 scenarios. We would like to stress that, in our experiments, the parameter settings remained constant for three of the camera views for all the 7 scenarios in each view (i.e., same settings were used for the 21 out of 28 experiments and the "dropped object wait time" parameter t in equations (1) and (2) was changed for the other 7). We have used the provided ground plane information and Tsai [10] camera calibration algorithm for each single camera view. The calibration results showed an error margin of 0.08 m. for cameras 3 and 4 and 0.2m. for Cameras 1 and 2.

The results for single camera detections have been summarized in Table 1. As seen from the results, we have achieved better results with Camera views 3 and 4. These cameras have been positioned high with more top-down views making object splits and drop-off events easier to observe. Cameras 1 and 2 are positioned at lower heights causing difficulties in tracking due to multiple occlusions or limited field of views. In all cases the stationary object detection algorithm improved the performance of abandoned object detection based solely on tracking. As indicated in Table 1, we have achieved good results for the favorable camera views in all except one scenario and achieved very promising results for the less than ideal camera views. Note that in Scenario 3 the object was never abandoned as indicated by N/A in the Alarm and Warning Time Differences columns of Table 1. For this scenario in the camera-3 view the object that was put down, was detected by the stationary object detector but not confirmed as an abandoned object since the owner did not leave.

4.1. Camera 1 and 2 Views

Our algorithm showed promising results for camera 1, which presented challenging view of the abandoned object scenarios. The low camera view caused multiple occlusions for just about every object in the view, multiple shadow and reflection problems, and constant movement around the abandoned object. In all scenarios except for scenario-4 we were able to detect the abandoned object with an average of 2.14 seconds from the ground truth alarm time and an average of 0.40 meters from the ground truth object location. The major cause of the errors was the difficulty in maintaining object-ID's due to the depth of the view and distinguishing people in the groups as shown in Figure 4.

Also the reflections in the scene caused the algorithm to incorrectly locate the ground position of each object, causing the objects to look closer or further away from the abandoned object. False alarms also occurred in this camera view due to constant movement towards the back of the field of view.

Table.1: Summary of Results

Scenario / Camera	Alarm Time Δ (seconds)	Warning Time Δ (seconds)	Abandoned Object Distance Δ (meters)	False Alarm
1 / 1	1.84	1.04	0.241	0
1 / 2	0.48	0.12	0.107	0
1 / 3	0.16	0.2	0.103	0
1 / 4	3.27	4.64	0.276	0
2 / 1	0.72	1.36	0.406	0
2 / 2	1.16	1.8	0.044	0
2 / 3	1.24	1.88	0.071	0
2 / 4	0.04	0.04	0.177	0
3 / 1	N/A	N/A	--	0
3 / 2	N/A	N/A	--	0
3 / 3	N/A	N/A	0.629	0
3 / 4	N/A	N/A	--	0
4 / 1	--	--	--	0
4 / 2	--	--	--	0
4 / 3	--	--	0.284	0
4 / 4	--	--	0.138	0
5 / 1	1.92	1.32	0.623	0
5 / 2	--	--	--	0
5 / 3	0.6	1.2	0.048	0
5 / 4	0.2	0.12	0.086	0
6 / 1	1.04	1.8	0.624	1
6 / 2	0.08	0.44	0.044	0
6 / 3	0.04	0.44	0.278	0
6 / 4	4.64	5.4	0.009	0
7 / 1	7.36	8.64	0.121	2
7 / 2	--	--	--	0
7 / 3	0.68	0.2	0.211	0
7 / 4	1.04	1.88	0.149	0

Although located at a high position, in the Camera 2 view the scenario occurs in a small footprint within the image limiting the number of pixels on each object making it difficult to detect the abandoned objects. In this camera view we detected the abandoned object event in scenarios 1, 2, and 6 with an average of 0.573 seconds from documented alarm time. The small abandoned object in scenario 5 closely matches the background and segmentation algorithm was unable to detect the object. In scenario 7, the abandoned object's border pixels dropped in and out of the background model. This prevented these few border pixels from being incorporated in the stationary object, as controlled by the *stationary object overlap* parameter in Table 3. Changing this parameter would make it possible to detect the abandoned object in this scenario but would have caused some spurious detections in other scenarios.

Figure 4. The undesired effects of field depth, shadows and reflections interfere with accurate tracking. Abandoned object is detected but parent object's ID could not be maintained.

4.2. Camera 3 and 4 Views

We achieved reasonable results in all of the 7 scenarios for camera views 3 and 4 except for Scenario 1 as explained above. In each case we detected the split event and validated the object drop-off. For this set of data our alarm times produced an average difference .8 seconds from the alarm and warning times documented in the ground truth data. Due to the use of 2D analysis in single camera views, our detection is inevitably view dependent especially for objects that are moving directly towards the camera as in camera 4 view of the Scenario 1. Our worst result occurred in this case, where the parent object walked toward the camera causing the detection of the drop-off event after the owner crossed the 3 meter boundary, issuing an alarm 3.27 seconds late. Also, the Scenario 4 presented a setting that emphasizes the limitation of object split based analysis. In this scenario the owner of the bag did not leave it until another person stood next to the bag. Although no alarm was generated for these views, we were able to correctly identify the bag's location in the stationary image to within .21 m. error.

4.3. Camera Voting

As explained in Section 3, we used a voting scheme to fuse the warning and alarm information from multiple camera views to lessen the effects processing limitations and to produce accurate results. For example as mentioned above Camera 4 view of the Scenario 1 produced a bad single camera result due to the view dependency limitation, but as seen in Table 2, the camera-voting algorithm produced an accurate overall outcome despite the individual inaccurate camera result. Similarly in Camera 4 view of Scenario 6 and Camera 1 view of Scenario 7 the erroneous alarm time results due to view dependent tracking limitations are eliminated from the final outcome.

Table.2: Results From Camera Voting for each Scenario

Scenario	Alarm Time Δ (seconds)	Warning Time Δ (seconds)	Distance Δ (meters)
1	0.16	0.24	0.057
2	0.04	0.68	0.252
5	0.20	0.12	0.272
6	0.04	0.44	0.139
7	1.04	0.20	0.138

4.4. Parameter Settings

The parameters specific to the algorithm described in this paper have been listed in Table 3. For each of these parameters we have determined the optimal default settings and for all except one have remained constant through all views and all scenarios. The "drop event wait time" parameter has been highlighted in yellow in Table 3 was set differently for one of camera views only. In our experiments all the tracking parameters remained constant for each of the camera views through each scenario, including tracking algorithm parameters such as background thresholds, minimum object size and image masks used. The parameters in the last three rows of Table 3 including the "parent object warning distance", "parent object alarm distance", and the "parent object time away from stationary object" are specified by the PETS evaluation task.

Table.3: Stationary Object Detection Algorithm Parameters

NAME	DESCRIPTION	Value
Drop-off Event Wait Time	Parameter controls the time it takes for the dropped object to be stationary.	Cameras 1,2,4: 3 seconds Camera 3: 2 seconds
Drop-off Event Idle Distance	Parameter specifies the maximum distance In pixels that the center point of the dropped object can move to be classified as stationary.	3 pixels
Stationary Object Overlap	Percent pixel of overlap between the dropped-off object and the object detected in the stationary view.	80%
Parent Object warning distance	Defined by PETS. The distance that the parent object has to move away from left object to issue a warning.	2 meters
Parent Object alarm distance	Defined by PETS. The distance that the parent object has to move away from left object to issue a alarm.	3 meters
Parent Object time away from stationary object	Defined by PETS. The time the parent object has to be away from the warning and alarm zone for a warning or alert to be issued.	30 Seconds

5. Conclusions and Future Work

We presented a new method combining moving object tracking with a novel stationary object detection algorithm to detect abandoned objects even in the existence of multiple occlusions.

The method we propose here addresses not only algorithmic but also functional and implementation aspects of abandoned object detection to provide a robust and practical solution. While the algorithmic aspect of an event outlines the steps of the process, the functional aspect deals with the role of significant components or a group of steps and their intermediate impact. Finally the implementation aspect focuses on the feasibility and applicability of the solution to real world situations to be analyzed in near-real time to identify and respond to security violations in time to mitigate their impact. The implementation feasibility of any method is

of utmost importance as the number of cameras in the facilities to be protected increase; the automatic video data analysis problem grows exponentially.

The tools and technologies described here are implemented as parts of a real time video analysis system while additional capabilities are currently under development at intuVision.

Our extensive experiments, as discussed in the previous section, illustrate the potential and the limitations of the proposed approach. Guided by these results we identified the areas of improvements as further eliminating the view dependency by and extracting the owner object for the drop-off event detection. If the drop-off event as defined in this paper is unobservable, a real-time historic search mechanism can be used to look through data extracted from previous frames of video and locate the frame of video where the abandoned object first appeared. Then the algorithm could locate the object closest to that abandoned object and designate that as the owner of the object. Finally the tracking information can be used to locate the owner in the current frame to determine the alarm time. We are also developing a classification algorithm to better identify the stationary image objects to distinguish between valid objects and regions created by other elements in the scene such as pixels occupied by multiple moving objects over a period due to the viewing conditions.

Acknowledgements

The authors would like to thank Mr. Ian Pushee and Mr. Jason Silverstein for implementing parts of the abandoned object detection algorithms and running experiments on PETS 06 sequences.

References

[1] L.M. Fuentes, S.A. Velastin, "Tracking-based Event Detection for CCTV Systems" *in Pattern Analysis and Applications', 7(4) Springer, ISBN/ISSN 1433-755X (2005*).

[2] S. Guler, T. Freyman and R.F. Brammer "Total Security Solutions", *in Proc. IEEE Conf. on Technologies for HLS, pp. 153-158 Boston, April 2004.*

[3] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: Realtime Surveillance of People and Their Activities," *IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 809-830, Aug. 2000*.

[4] T. Horpraset, D. Harwood, and L.S. Davis, "A Statistical Approach for Realtime Robust Background Subtraction and Shadow Detection," *in Proc. IEEE Frame Rate Workshop, pp. 1-19, Kerkya, Greece 1999*.

[5] P. KaewTraKulPong and R. Bowden, "An Improved Adaptive Background Mixture Model for Realtime Tracking with Shadow detection," *Video Based Surveillance Systems, pp. 135-144, Kluwer Academic Publishers, Boston, 2002*.

[6] K. Kyungam, T.H. Chalidabhongse, D. Harwood, L. Davis, "Real-time foreground –background Segmentation Using Codebook Model," *in Real-Time Imaging, Elsevier Publishers, 2005*.

[7] M. Spengler and B. Schiele, "Automatic Detection and Tracking of Abandoned Objects" *Perceptual Computing and Computer Vision, Computer Science Dept. Report, ETH Zurich, Switzerland*.

[8] C. Stauffer and W.E.L. Grimson, "Learning Patterns of Activity Using Real-time tracking', *IEEE Trans. Pattern. Anal. Mach. Intell. vol. 22, no. 8, Aug. 2000, pp. 831-843*.

[9] E. Stringa and C. Regazzoni, "Content-Based Retrieval And Real Time Detection From Video Sequences Acquired By Surveillance Systems," *in Proc. IEEE Int. Conf. Image Processing, pp. 138-142, (Chicago, IL), Oct. 1998*.

[10] Y. L. Tian, M. Lu, and A. Hampapur, "Robust and Efficient Foreground Analysis for Real-time Video Surveillance," *IBM T.J. Watson Research Center Tech Report*.

[11] R. Y. Tsai, "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision", *in Proc. IEEE Conf on CVPR, Miami Beach FL, pp 364-374, 1986*.

Index of Authors