



CAVIAR

Context Aware Vision using Image-based Active Recognition



CAVIAR : D 11 – Network Activity

Date: 31 October 2004

Author(s): James L. Crowley

Work package WP 7 - Dissemination

Document status: Version 1.0

Usage: internal

Keywords: Dissemination Workshop: Performance Evaluation for
Tracking and Surveillance

Abstract

This report describes the Sixth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2004) held in Prague, 10 May 2004. It reviews the objectives for the workshop, presents the program, lists the registered participants, and provides the papers published in the proceedings.

DOCUMENT HISTORY

<i>Version</i>	<i>Date</i>	<i>Reason of change</i>
	15 Nov 2003	Published data sets and call for papers
	15 Feb 2004	Paper Submission Deadline
	1 April 2004	Returned paper reviews
	23 April 2004	Edited proceedings for workshop
	10 May 2004	Workshop held in Prague
0.9	30 October 2004	Report constructed based on workshop
1.0	31 October 2004	Edited report to improve presentation

Table of Contents

1. Introduction.....	1
2. The PETS '04 Programme.....	2
3. Registered Participants	4
4. PETS 2004 Workshop Proceedings.....	5

1. Introduction

The transition from laboratory demonstrations to commercial products requires methods and metrics to specify and evaluate the performance of competing systems. The publication of ground-truth labeled benchmark sequences is particularly important for such development. The CAVIAR consortium has organized an international workshop on Performance Evaluation for Tracking and Surveillance, PETS '04, on 10 May 2004, in Prague in conjunction with. The theme for PETS '04 has been the automatic observation of human activity in public places.

Six scenarios were recorded with a wide-angle camera lens in the entrance lobby of the *INRIA Rhône-Alpes* research laboratory in Montbonnot France, using members of the CAVIAR project as actors. Activities included a person walking in a straight line (3 sequences), a person browsing at information displays (5 sequences), behaviours while seated in a chair (3 sequences), persons abandoning packages (5 sequences), groups of people encountering (6 sequences), and people fighting (4 sequences). For each scenario, a ground-truth file has been constructed to indicate a bounding box for each individual, activity labels for each individual (appear, disappear, occluded, inactive, active, walking, running), a scenario label for each individual (fighter role, browser role, left victim role, leaving group role, walker role, left object role) and a situation label for each frame: (moving, inactive, browsing) and a scenario label for each frame (browsing, immobile, walking, drop down). These ground truth files have been made public for half of the sequences for use as training data. The PETS challenge is to demonstrate automatic labeling for the remaining sequences.

Authors were asked to describe the tracking and recognition methods, estimate or measure computational costs, and present error rates obtained with the published ground truth. Authors are also invited to propose new performance evaluation metrics that might be of interest.

The workshop was attended by 27 registered participants, assisted by several unregistered “walk-ins”. The program was composed of 11 presentations, selected from 12 papers submitted to the review process. The PETS '04 workshop has helped to establish the CAVIAR labeled data sets as a known and widely used benchmark data set for the community.

2. The PETS '04 Programme

Session 1	Welcome and Introduction to PETS ECCV'04
9:00 - 9:15	Introduction - James L. Crowley
9:15 - 9:30	The PETS04 Surveillance Ground-Truth Data Sets <i>Bob Fisher</i>
Session 2	Metrics
9:30 - 10:00	New Performance Evaluation Metrics for Object Detection Algorithms <i>Jacinto Nascimento Jorge S. Marques</i>
10:00 - 10:30	Trajectory Distance Metric using Hidden Markov Model based Representation <i>Fatih Porikli</i>
10:30 - 11:00	Coffee
Session 3	Automatic Regulation
11:00 - 11:30	Visual Tracking in Video Sequences <i>Alban Caporossi, Daniela Hall, Patrick Reignier and James L.Crowley</i>
11:30-12:00	Automatic Tracking and Labeling of Human Activities in a video sequence <i>Fengjun Lv, Jinman Kang, Ram Nevatia, Isaac Cohen, Gerard Medioni</i>
12:00 - 12:30	Joint Appearance and Trajectory based Data Association for Multi-object Tracking <i>Arvind Lakshmikumar, Michael Burl</i>
12:30 - 14:00	Lunch

- Session 4 **Semantics**
- 14:00 - 15:30 **Is it Interesting? Comparing human and machine judgements on the PETS dataset**
Hannah Dee, David Hogg
- 14:00 - 15:30 **Ontology-guided Training of Bayesian Networks for High Level Analysis in Visual Surveillance**
Christopher Town
- 14:00 - 15:30 **On-line Tracking Groups of Pedestrians with Bayesian Networks**
Pedro M. Jorge, Jorge S. Marques, Arnaldo J. Abrantes
- 15:30 - 16:00 **Coffee**
- Session 5 **Tracking Groups**
- 16:00 - 17:30 **A Probabilistic model for an EM-like Object Tracking Algorithm using color histograms**
Z. Zivkovic Ben Krose
- 16:00 - 17:30 **Tracking Pedestrians in a Multiple Camera System with Trajectory Prediction and Occlusion Modeling**
Jorge P. Batista
- 16:00 - 17:30 **Discussion: tracking algorithms and performance metrics**

3. Registered Participants

A list of 27 participants registered to participate in the workshop. These are:

Tian Qi, Institute for Infocomm Research, Media Division

Timo Kohlberger, University of Mannheim, Dept. of Math. & Computer Science

Dr. Lucas Paletta, Joanneum Research, Institute of Digital Image Processing

Mr. Kenji Okuma, University of British Columbia, Computer Science

Mr. Faisal Qureshi, University of Toronto, Department of Computer Science

Dr. Vincent Devin, University of Leeds, School of Computing

Ms. Hannah Dee, University of Leeds, School of Computing

Mr. Christopher Town, University of Cambridge, Computer Laboratory

Prof. Gerard Medioni, University of Southern California, Computer Science

Dr. Andy Thean, TNO TPD, Imaging and Data Interpretation

Dr. Wolfgang Niem, Robert Bosch GmbH, Corporate Research / Image proc. Systems

Dr. Fatih Porikli, MERL,

Dr. Zoran Zivkovic, University of Amsterdam, Intelligent Autonomous Systems

Mr. Jonathan Rymel, Computer Recognition Systems Ltd.

Prof. James CROWLEY, INRIA Rhone Alpes, Laboratoire GRAVIR

Mr. Peter Gemeiner, Vienna University of Technology

Dr. Jacinto Nascimento, IST, Eng. Electrotécnica e de Computadores

Mr. Karel Zimmerman, Czech Technical University Prague, Dept. of Cybernetics

Dr. Jorge Batista, ISR-institute of Systems and Robotics, DEEC-FCTUC

Mr. Pedro Mendes Jorge, ISEL, DEETC

Ms. Barbara Cenitch, USG, Engineering Division

Ms. Jodi Miller, DOD, Army

Prof. Robert Fisher, Univ. of Edinburgh, School of Informatics

Mr. David Longbottom, Vision Base Ltd, Research & Development

Prof. José Santos-Victor, Instituto Superior Técnico, ISR

Dr. Matthew Antone, Alphatech, IncComputer Vision Group

Proceedings

PETS ECCV 2004

6th International Workshop on
Performance Evaluation for Tracking and Surveillance

10 May 2004 • Prague, Czech Republic

Sponsored by Project

IST 2001 37540 CAVIAR: Context Aware Vision using Image-based Active Recognition

Proceedings of 6th International Workshop on Performance Evaluation for Tracking and Surveillance—
PETS ECCV 2004

Edited by James L. Crowley.

Published by Czech Technical University, Faculty of Electrical Engineering
Prague, Czech Republic, Prague May 2004
iii+92 pages, 30 copies.

Typeset by authors.

© Copyright to all papers are retained by the authors.

Additional copies may be ordered from:
James L. Crowley,
INRIA Rhone Alpes, 655 Ave de l'Europe, 38330 Montbonnot, France.
James.Crowley@inrialpes.fr

PETS 2004 took place in conjunction with European Conference on Computer Vision 2004.

ISBN 80-01-02979-4

PETS 2004 Workshop Proceedings

PETS STEERING COMMITTEE

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

PETS '04 Program Co-Chaimen:

James L. Crowley I.N.P. Grenoble, France
Robert B. Fisher University of Edinburgh, UK
José Santos-Victor IST-ISR Lisbon, Portugal

Workshop webmaster:

Daniela Hall INRIA Rhône Alpes, France

Program Committee:

James L. Crowley I.N.P. Grenoble, France
Robert F. Fisher University of Edinburgh
José Santos-Victor IST-ISR Lisbon, Portugal
Fatih Porikli Mitsubishi Research Laboratories, USA
Arvind Lakshmikuma Honeywell Labs, USA
Thomas Moeslund Aalborg University, Denmark
James Ferryman Reading Univeristy, UK
Daniela Hall INRIA Rhône Alpes, France

Proceedings edited by : James L. Crowley
Professeur, I.N.P. Grenoble
INRIA Rhône Alpes
655 Ave de l'Europe
38330 Montbonnot
France

Published by Czech Technical University
Faculty of Electrical Engineering
Prague, Czech Republic
May 2004

ISBN 80•01•02979•4

Copyright to all papers are retained by the authors.

Contents

The PETS04 Surveillance Ground-Truth Data Sets.....	1
<i>Robert B. Fisher</i>	
New Performance Evaluation Metrics for Object Detection Algorithms.....	7
<i>Jacinto Nascimento Jorge S. Marques</i>	
Trajectory Distance Metric using Hidden Markov Model based Representation.....	15
<i>Fatih Porikli</i>	
Visual Tracking in Video Sequences.....	23
<i>Alban Caporossi, Daniela Hall, Patrick Reignier and James L. Crowley</i>	
Automatic Tracking and Labeling of Human Activities in a video sequence.....	33
<i>Fengjun Lv, Jinman Kang, Ram Nevatia, Isaac Cohen, Gerard Medioni</i>	
Joint Appearance and Trajectory based Data Association for Multi-object Tracking.....	41
<i>Arvind Lakshmikumar, Michael Burl</i>	
Is it Interesting? Comparing human and machine judgements on the PETS dataset.....	49
<i>Hannah Dee, David Hogg</i>	
Ontology-guided Training of Bayesian Networks for High Level Analysis in Visual Surveillance....	57
<i>Christopher Town</i>	
On-line Tracking Groups of Pedestrians with Bayesian Networks.....	65
<i>Pedro M. Jorge, Jorge S. Marques, Arnaldo J. Abrantes</i>	
A Probabilistic model for an EM-like Object Tracking Algorithm using color histograms.....	73
<i>Zoran Zivkovic and Ben Krose</i>	
Tracking Pedestrians in a Multiple Camera System with Trajectory Prediction and Occlusion Modeling.....	83
<i>Jorge P. Batista</i>	

Introduction to PETS 2004

James L. Crowley, Robert B. Fisher and José Santos Victor

Recent international events have created a strong economic demand for computer vision technologies for observing human activity. The transition from laboratory demonstrations to commercial products requires methods and metrics to specify and evaluate the performance of competing systems. The goal of the PETS workshop is to foster the emergence of such methods and metrics.

The 6th International Workshop on Performance Evaluation for Tracking and Surveillance, PETS '04 continues the recent series of PETS workshops held at FG 00, CVPR '01, ECCV '02, ICVS '03 and ICCV '03. The theme for PETS '04 is observing human activity in public places. PETS '04 has been organised by the IST CAVIAR project. Six scenarios were recorded with a wide-angle camera lens in the entrance lobby of the *INRIA Rhône-Alpes* research laboratory in Montbonnot France, using members of the CAVIAR project as actors. Activities included a person walking in a straight line (3 sequences), a person browsing at information displays (5 sequences), behaviours while seated in a chair (3 sequences), persons abandoning packages (5 sequences), groups of people encountering (6 sequences), and people fighting (4 sequences). For each scenario, a ground-truth file has been constructed to indicate a bounding box for each individual, activity labels for each individual (appear, disappear, occluded, inactive, active, walking, running), a scenario label for each individual (fighter role, browser role, left victim role, leaving group role, walker role, left object role) and a situation label for each frame: (moving, inactive, browsing) and a scenario label for each frame (browsing, immobile, walking, drop down). These ground truth files have been made public for half of the sequences for use as training data. The PETS challenge is to demonstrate automatic labeling for the remaining sequences.

Authors were asked to describe the tracking and recognition methods, estimate or measure computational costs, and present error rates obtained with the published ground truth. Authors are also invited to propose new performance evaluation metrics that might be of interest.

The PETS04 Surveillance Ground-Truth Data Sets

Robert B. Fisher
School of Informatics, University of Edinburgh
rbf@inf.ed.ac.uk

Abstract

This paper summarizes the 28 video sequences available for result comparison in the PETS04 workshop. The sequences are from about 500 to 1400 frames in length, for a total of about 26500 frames. The sequences are annotated with both target position and activities by the CAVIAR research team members.

1. Introduction

This paper describes the video sequences used in the PETS04 workshop competition. The sequences are oriented about a public space surveillance task, and are ground truth labeled frame-by-frame with bounding boxes and also a semantic description of the activity in each frame. Altogether, there are 28 video sequences containing about 26500 labeled frames, grouped into 6 different activity scenaria.

The first group of videos was acquired at INRIA in July 2003. The sequences contained scripted activities by the research team members. The intended test scenaria are:

Scenario	Number of Sequences	Number of Frames
Walking	3	3045
Browsing	6	6665
Collapse	4	4227
Leaving object	5	5848
Meeting	6	4135
Fighting	4	2499
Total	28	26419

However, almost all sequences also contained both an introductory activity by one of the researchers, as well as unscripted activity (usually walking or meetings by other employees at INRIA).

These sequences are publicly accessible at URL:
homepages.inf.ed.ac.uk/rbf/CAVIARDATA1

1.1 Ground Truth Labeling

Based on the CAVIAR activity representation model, each video frame has been labeled with a set of ground truth descriptions.

Each individual person was described by a bounding box (id, centre coordinates, width, height, orientation of main axis of individual), plus a description of his/her movement (inactive, active, walking, running). Individuals are only labeled once they start moving; otherwise they are effectively background. Based on the proposed semantics of the activity interpretation, each box is usually labeled with a role (fighter, browser, left victim, leaving group, walker, left object), is a participant in a situation (browsing, moving, inactive), which is a component of a scenario (Walking, Idleness, Browse, Collapse, Leaving object, Meeting, Fighting). Each box is labeled with some of the above labels in each frame.

The semantics of activity labeling were constrained by a finite-state model of the allowable behaviors. These are summarized in Section 2, which shows the allowable sequences of situations in a given scenario. In each scenario, the individual or group is observed in a sequence of situations determined by the finite state model for that scenario. When in a situation, the actor must fulfill a specific role linked to that situation. As well as the role, the ground truth labeling for the box has a qualitative assessment of the motion of the individual or group, *i.e.* whether they are running, walking, stationary but active (*e.g.* moving arms), or inactive.

Each video frame contains zero or more labeled individual or group boxes. The boxes are labeled with an identifier, which persists as long as the individual is visible. If a person disappears and then later reappears, then the individual obtains a new identity. If the person is obscured/occluded for only a few frames, then the same identity is maintained.

Similarly, groups of interacting individuals also are described by bounding boxes (id, centre coordinates, width, height, orientation of main axis of individual, list of component individual boxes), plus a description of the group's movement (inactive, active, moving). Based on the pro-

posed semantics of the activity interpretation, each group box is usually labeled with a role (meeters, fighters, walkers), is a participant in a situation (fighting, moving, meeting, split up, inactive, leaving victim, leaving object), which is a component of a scenario.

The grammar of the ground truth file can be seen in appendix A. The web site will also provide the ground truth labels in XML shortly. An example of the current ground-truth entry for frame 517 of sequence LeftBag is:

```

frame      LeftBag      517      ibl
  ib      2
          210  247  55  39  10  wk
          wr  1.0  m  1.0  im  1.0
  eib
eibl      gbl      egbl      eframe

```

The description says: there is only individual box 2, with center at column 210 and row 247. The bounding box width is 55 pixels wide and 39 pixels tall, and the dominant orientation is 10 degrees. The target is walking (wk), fulfills the walker role (wr) with certainty 1.0, is in a moving situation (m) with certainty 1.0, which is part of the immobile scenario (im) with certainty 1.0.

1.2 Open issues

The labeling has highlighted some issues:

1. Variability of the ground truth

Since the labeling was done by humans, there is a natural variation in both the parameters and occurrence of the labels, *e.g.* the positions and sizes of the bounding boxes, or when the box or activity starts. Knowing the range of human variation will help with comparison to automatic calculations of the statistics.

To help assess this question, one of the datasets has three labelings by different individuals. As the statistics package is still being developed, we do not yet have data on the variation.

2. Nature of the behaviour labeling

We have taken the position of an omniscient labeler, so all scenaria are labeled as they actually are, although the system may not be able to correctly label the scenario until many frames in the future.

The main labeling difficulty is one of timing - when does one situation or scenario change into another. We have assumed that differences in this will be the sort of natural variation assessed as described above.

The labeling of the roles/situations/scenaria was problematic. It was often unclear how each of the labels

was to be used. We attempted to maintain at least consistent labeling by coordinating and reviewing of labels by one person. Therefore, the symbolic labeling is based on a best-guess representation of the final activity model.

3. What is a group?

We have attempted to define a group as a set of individuals that are reacting to each other. This means that individuals may pass each other, *e.g.* one behind the other, without interacting and thus not forming a group. The human labelers can usually make this judgment, but it is less likely that an automatic labeler will be able to distinguish all instances of interaction. Thus, there is probably going to be a lot of false alarms on group box detection (*i.e.* individuals who are really not interacting, but just passing closely).

Similarly, we grouped individuals that were interacting independently of the distance between the individuals, starting from the frame in which they first seemed to react to each other. For example, if two people wave while still quite distant and then turn to approach each other, the group box and labeling starts in the frame where the two noticed each other and initiated the waving.

4. Multiple *versus* unique labels

Should an individual (or a group) have more than one role label, and participate in more than one situation and scenario at the same time? In labeling, we have decided only single classifications apply in each frame.

2. Semantic labeling

The modeled scenaria, their constituent situations, the participant roles allowed in each situation and the movement description for each role are summarized here.

The models are currently expressed as finite state automata, with the states as individual situations.

2.1 Plaza Observation Setting

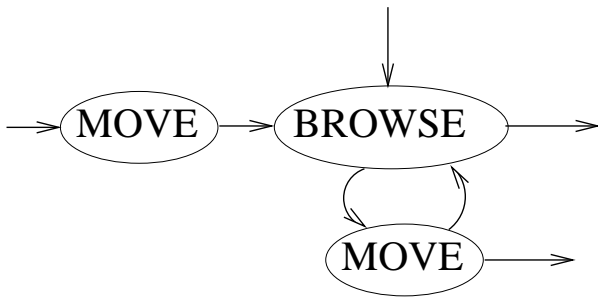
The different contexts that can give rise to scenaria are: Browse, Idleness, Drop-Dead, Walk, Fight, Meet, Leave-Object.

Solid ovals are individual situations, dashed ovals are group situations. Vertical bars are when two situations need to end at the same time.

For each scenario, there is a set of situations. Each situation (*e.g.* "Browse") has listed the allowable Roles (*e.g.* "Browser") and allowable Movements (*e.g.* "Inactive"): `BROWSE:Browser/{Inactive}`.

2.1.1 Browse Context

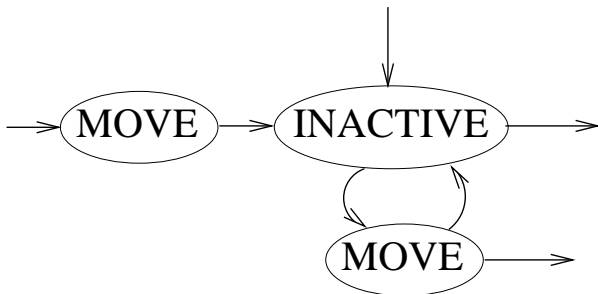
Actually looking at some information display:



MOVE: {Walker,Browser}/{Walking}
 BROWSE: Browser/{Active,Inactive}

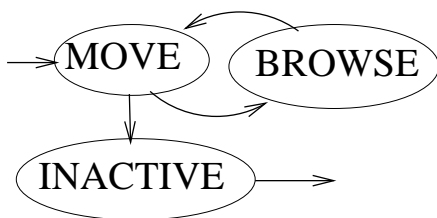
2.1.2 Idleness Context

Just standing around:



MOVE: Walker/{Walking}
 INACTIVE: Walker/{Active,Inactive}

2.1.3 Drop Dead Context



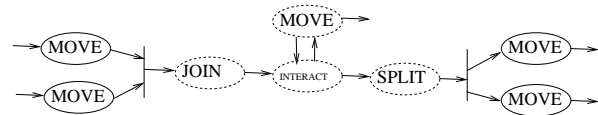
MOVE: {Walker,Browser}/{Walking}
 INACTIVE: Walker/{Inactive}
 BROWSE: Browser/{Active,Inactive}

2.1.4 Walk Context



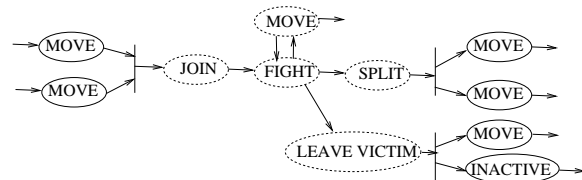
MOVE: Walker/{Walking}

2.1.5 Meet



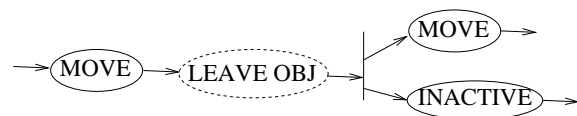
MOVE (individual): Walker/{Walking}
 MOVE (group): Walkers/{Movement}
 JOIN: Meeters/{Movement}
 INTERACT: Meeters/{Active,Inactive}
 SPLIT: Meeters/{Movement}

2.1.6 Fight



MOVE (individual): Walker/{Walking,Running}
 MOVE (group): Walkers/{Movement}
 JOIN: Fighters/{Movement}
 FIGHT: Fighters/{Active,Movement}
 SPLIT: Fighters/{Movement}
 LEAVE VICTIM: Fighters/{Active,Movement}
 INACTIVE: Left Victim/{Active,Inactive}

2.1.7 Leave-Object



MOVE (individual): Walker/{Walking}
 INACTIVE: Left Object/{Inactive}
 LEAVE OBJ: Walkers/{Inactive}

3. Shop observation scenario datasets

The web site given above will also eventually contain about 50 additional ground-truth labeled video sequences observing scenaria that occur in a shopping center, containing about 60000 labeled frames. This is expected to be complete in the summer of 2004.

Acknowledgements

Funding for this project was given under the EC's Information Society Technology's programme project IST 2001 37540. We'd like to thank many people for help with the ground-truth labeling: Toby Breckon, Helmut Cantzler, Ignasi Cos Aguilera, Jose Manuel Vazquez Diosdado, Dina Kronhaus, Gregor Miller and Donald Nairn.

A. Ground truth label grammar

The grammar and meaning of the ground truth files is as follows:

```
% the whole file
FILE -> FRAMELIST
```

```
% a list of frame descriptions
FRAMELIST -> FRAME
FRAMELIST -> FRAMELIST FRAME
```

```
% a frame description
FRAME -> frame NAME FID ibl IBLIST eibl gbl
      GBLIST egbl eframe
```

```
% a video sequence name
NAME -> character string with no blanks
```

```
% the frame number
FID -> an integer
```

```
% a list of individual boxes
IBLIST ->
IBLIST -> IBLIST IB
```

```
% a list of group boxes
GBLIST ->
GBLIST -> GBLIST GB
```

```
% an individual box description
IB -> ib IID IC IR IW IH IO IASL IFLAGL eib
```

```
% individual box ID
IID -> an integer
```

```
% IR, IC - row and column of center of
% individual box
IC -> an integer
IR -> an integer

% IH, IW - height and width of individual
% box
IW -> an integer
IH -> an integer
```

```
% IO - main axis orientation [0..179]
% degrees
IO -> an integer
```

```
% IASL - state flag list
IASL ->
IASL -> IASL IAS
IAS ->
      ap          % appear
      | di        % disappear
      | o         % occluded
      | in        % inactive
      | ac        % active
      | wk        % walking
      | r         % running
```

```
% IFLAGL - scenario flag list
PROB -> a floating point probability
      in [0.0...1.0]
```

```
IFLAGL ->
IFLAGL -> IFLAGL IFLAG
IFLAG ->
      f  PROB    % fighter role
      | br PROB  % browser role
      | lv PROB  % left victim role
      | lg PROB  % leaving group role
      | wr PROB  % walker role
      | lo PROB  % left object role
      | m  PROB  % moving situation
      | is PROB  % insactive situation
      | bsi PROB % browsing situation
      | bsc PROB % browsing scenario
      | im PROB  % immobile scenario
      | wg PROB  % walking scenario
      | dd PROB  % drop down scenario
      | pi PROB  % immobile event
```

```
% a group box description
GB -> gb GID GC GR GW GH GO gibl GMEML egibl
      GASL GFLAGL egb
```

```
% group box ID
```

```

GID -> an integer

% GR, GC - row and column of center of
% group box
GC -> an integer
GR -> an integer

% GH, GW - height and width of group box
GW -> an integer
GH -> an integer

% GO - main axis orientation [0..179]
% degrees
GO -> an integer

% GMEML - List of group members
GMEML ->
GMEML -> GMEML IID

% GASL - group state flag list
GASL ->
GASL -> GASL GAS
GAS ->
    ap          % appear
    | d         % disappear
    | i         % inactive
    | ac        % active
    | mo        % movement

% GFLAGL - scenario flag list
PROB -> a floating point probability
      in [0.0...1.0]
GFLAGL ->
GFLAGL -> GFLAGL GFLAG
GFLAG ->
    f  PROB    % fighters role
    | me PROB  % meeters role
    | w  PROB  % walkers role
    | gf PROB  % fighting situation
    | gmo PROB % moving situation
    | gme PROB % meeting situation
    | s  PROB  % split up situation
    | gi PROB  % inactive situation
    | glv PROB % leaving victim situation
    | glo PROB % leaving object situation
    | fsc PROB % fighting scenario
    | mes PROB % meeting scenario
    | ls  PROB % leaving object scenario
    | fst PROB % fight start event
    | fe  PROB % fight end event
    | fv  PROB % left victim event

```


New Performance Evaluation Metrics for Object Detection Algorithms*

Jacinto Nascimento

Jorge S. Marques

ISR/IST

{jan,jsm}@isr.ist.utl.pt

Av. Rovisco Pais, Torre Norte, 10049-001, Lisboa Portugal

Abstract

This paper proposes novel metrics to evaluate the performance of object detection algorithms in video sequences. The proposed metrics allow to characterize the methods being used and classify the types of errors into region splitting, merging or merge-split, detection failures and false alarms. This methodology is applied to characterize the performance of five segmentation algorithms. These tests are performed in the context of object detection in outdoor scenes with a fixed camera.

1. Introduction

Video surveillance systems rely on the ability to detect moving objects in the video stream. Each image is segmented by image analysis techniques. This should be done in a robust way in order to cope with unconstrained environments, non stationary background and different object motion patterns. Furthermore, different types of objects have to be considered e.g., persons, vehicles or groups of people.

Many algorithms have been proposed for object detection in video surveillance. They rely on different assumptions e.g., statistical models of the background [12, 10] frame differences [5] or a combination of both [3]. However, few information is available on the performance of these algorithms in different operating conditions.

Object detection assessment has been recently considered in [8] assuming it is a binary detection problem. Standard measures used in Communication theory such as misdetection rate, false alarm rate and receiver operating characteristics (ROC) were used [11]. However, this approach has several limitations. Object detection is not a binary detection problem. Several types of errors should be considered (not just misdetections and false alarms). Second, the proposed test in [8] is based on the selection of rectangular regions with and without persons. This is an unrealistic

assumption since practical algorithms have to segment the image into background and foreground and do not have to classify rectangular regions selected by the user.

In this paper, we propose objective metrics to evaluate the performance of object detection methods by comparing the output of the video detector with the ground truth obtained by manual edition of the video frames. The main features of the proposed method are the following. Given the correct segmentation of the video sequence we detect several types of errors *i*) splits of foreground regions, *ii*) merges of foreground regions, *iii*) simultaneous split and merge of foreground regions, *iv*) false alarms (detection of false objects), *v*) the detection failures (missing active regions). They all influence the performance of the video surveillance system in different ways. Furthermore, ambiguous segmentations have also been explicitly considered. For example, it is not always possible to know if two close objects correspond to a single group or a pair of disjoint regions. Both interpretations are adopted in such cases.

Five segmentation algorithms are evaluated in this paper. The first is denoted as basic background subtraction (BBS) algorithm. It computes the absolute difference between the current image and a static background and compares each pixel to a threshold. All the connected components are computed and they are considered as active regions if their area exceeds a given threshold. The second method is the object detection algorithm proposed in the W4 system [6]. Three features are used to characterize each pixel of the background image: minimum intensity, maximum intensity and maximum absolute difference in consecutive frames. The third method assumes that each pixel of the background is a realization of a random variable with Gaussian distribution (SGM - Single Gaussian Model) [12]. The mean and covariance of the Gaussian distribution are independently estimated for each pixel. The fourth algorithm models each pixel as a mixture of Gaussians [10], determining which mode corresponds to the background and which describe active regions (MGM - Multiple Gaussian Model). The fifth method is the one used in the *Lehigh Omnidirectional Tracking System* (LOTS [2]).

*This work was partially supported by FEDER and FCT under the project LTT and by EU project CAVIAR (IST-2001-37540).

The tests presented in this work were performed with PETS2001 using the metrics proposed in this paper and PETS2004 sequences and evaluated using the metrics adopted in the CAVIAR project.

The structure of the paper is as follows. Section 2 briefly reviews previous work. Section 3 describes the segmentation algorithms. Section 4 describes the performance metrics proposed in the paper. Experimental tests are discussed in section 5 and section 6 presents the conclusions.

2. Related Work

Surveillance and monitoring systems often require the segmentation of all the moving objects in the video sequence. Segmentation is a key step since it influences the performance of the other modules, e.g., object tracking, classification or recognition. For instance if object classification is required, an accurate detection is often needed to obtain a correct classification of the object.

Background subtraction is a simple approach to detect moving regions. This can be done in several ways. In [6] each pixel of the background image is represented by three features: minimum intensity, maximum intensity, and the maximum rate of change at consecutive frames [6], or the median of largest inter-frames absolute difference [5]. Background subtraction can be performed by combining different types of features. Other methods rely on the use of statistical models of the background image. The Pfinder (“Person Finder”) [12] assumes that the background is modeled by a Gaussian distribution: each pixel is described by its mean and covariance matrix. The object detection is based on blob detection. A blob is a connected region, obtained by clustering the pixels with similar color and image coordinates. A multiclass statistical model of color is used to monitor the activity of person in indoor environments. In [10] segmentation is based on an adaptive background subtraction method which models each pixel as a mixture of Gaussians. In [7] it is proposed a background subtraction method which combines color and gradient information. Another approach is described in [2]. This system includes multiple background modelling, combining various techniques: adaptive thresholding with hysteresis and spatio-temporal grouping of active pixels, denoted quasi-connected components.

A problem related to the background subtraction approach are the false active regions, the so-called “negative” or ghosts [9]. These regions are caused by static objects belonging to the background image (e.g., cars) which start to move. This gives rise to a false active region located where the object was placed, due to the difference between the current frame and the background model computed using past information. This problem can be overcome by high level techniques [4] or by modeling the background with a mix-

ture of Gaussians [10].

3 Segmentation Algorithms

This section describes object detection algorithms used in this work: *BBS*, *W4*, *SGM*, *MGM*, *LOTS*. The *BBS*, *SGM*, *MGM* algorithms use color while *W4* and *LOTS* use gray scale images. The *BBS* algorithm, detects moving objects by computing the difference between the current frame and the background image. A thresholding operation is performed to classify each pixel as foreground or background. Ideally, pixels associated with the same object should have the same label. This is accomplished by morphological filtering (dilation and erosion) to eliminate isolated pixels and small regions followed by a connected component analysis (e.g., using 8 - connectivity criterion).

The second algorithm is denoted here as *W4* since it is used in the *W4* system to compute moving objects [6]. This algorithm is designed for grayscale images. The background model is built using a training sequence without persons or vehicles. During this period three values are estimated for each pixel: minimum intensity (Min), maximum intensity (Max), and the maximum intensity difference between consecutive frames (D). Foreground objects are computed in four steps: *i*) thresholding, *ii*) noise cleaning by erosion, *iii*) fast binary component analysis, *iv*) elimination of small regions.

The thresholding step proposed herein is given by

$$\begin{aligned} & (|I^t(x, y) < \text{Min}(x, y)| \vee |I^t(x, y) > \text{Max}(x, y)|) \\ & \wedge |I^t(x, y) - I^{t-1}(x, y)| > D(x, y) \end{aligned} \quad (1)$$

which leads to a less level of misdetections comparing with the one described in [6]. The third algorithm considered in this study is the *SGM* (Single Gaussian Model) algorithm. Color information is used in this method. Each pixel is represented by a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ where the mean μ and covariance Σ are recursively updated as follows

$$\mu^t = \alpha I^t(x, y) + (1 - \alpha)\mu^{t-1}, \quad (2)$$

$$\Sigma^t = (1 - \alpha)\Sigma^{t-1} + \alpha(I^t(x, y) - \mu^t)(I^t(x, y) - \mu^t)^T \quad (3)$$

The *SGM* performs a binary classification of the pixels into foreground or background and tries to cluster foreground pixels into blobs.

The fourth algorithm (*MGM*) models each pixel $x = (x, y)$ as a mixture of three Gaussians distributions, i.e.

$$p(I(x)) = \sum_{k=1}^N \omega_k \mathcal{N}(I(x), \mu_k, \Sigma_k), \quad (4)$$

where $\mathcal{N}(I(x), \boldsymbol{\mu}_k, \Sigma_k)$ is a multivariate normal distribution, $N = 3$ and ω_k is the weight of k th normal,

$$\mathcal{N}(I(x), \boldsymbol{\mu}_k, \Sigma_k) = c \exp\left\{-\frac{1}{2}\left(I(x)-\boldsymbol{\mu}_k\right)^T \Sigma_k^{-1}\left(I(x)-\boldsymbol{\mu}_k\right)\right\} \quad (5)$$

with $c = \frac{1}{(2\pi)^{n/2}|\Sigma_k|^{1/2}}$. Note that each pixel $I(x)$ is a 3×1 vector with three component colors (red, green and blue), i.e., $I(x) = [I(x)^R I(x)^G I(x)^B]^T$. To avoid an excessive computational cost, the covariance matrix is assumed to be diagonal [10].

The mixture model is updated dynamically. *i)* The algorithm checks if the pixel value x can be ascribed to a given mode of the mixture (match)¹. *ii)* If a distribution matches the new observation the parameters are updated according to (2), (3) where α is replaced by

$$\lambda_k = \alpha \mathcal{N}(I(x), \boldsymbol{\mu}_k, \Sigma_k) \quad (6)$$

The weights in (4) are updated by

$$\begin{aligned} \omega_k^t &= (1 - \alpha)\omega_k^{t-1} + \alpha(M_k^t), \quad \text{with} \\ M_k^t &= \begin{cases} 1 & \text{matched model} \\ 0 & \text{remaining models} \end{cases} \end{aligned} \quad (7)$$

α is the learning rate. The non matched components of the mixture remain the same. If none of the existing components match the incoming observation, the least probable distribution is replaced with the current value (as its mean), a large covariance matrix and a low weight. This distribution should contain a high variance and mean equal to the current value of the frame, a low prior weight should be assigned in this situation. *iii)* The distributions are sorted in the descending order of $\omega_k/|\Sigma_k|$. *iv)* The algorithm selects the first B Gaussians distributions as belonging to the background. B is chosen as follows: B is the smallest integer such that

$$\sum_{k=1}^B \omega_k > T \quad (8)$$

where T is a threshold that accounts for a certain quantity of data that should belong to the background.

The fifth algorithm [2] is tailored to the detection of non cooperative targets (e.g., snipers) under non stationary environments. This algorithm uses two gray level background images B_1, B_2 . This allows the algorithm to cope with intensity variation due to noise or fluttering objects which move in the scene. Each pixel of the input frame is compared to the closest background value and classified as active if the difference exceeds a given threshold $T_L(x, y)$. A quasi connected component analysis is then performed using a second threshold $T_H(x, y)$ in order to select groups of

¹A match occurs if the pixel is inside the confidence interval with ± 2.5 standard deviation.

active pixels and to classify them as targets. It is assumed that $T_H(x, y) = T_L(x, y) + c_S$, c_S being defined by the user.

These images are updated as follows

$$B_1(x, y) = \min\{I^n(x, y), n = 1, \dots, T\} \quad (9)$$

$$B_2(x, y) = \max\{I^n(x, y), n = 1, \dots, T\} \quad (10)$$

where $n \in 1, 2, \dots, T$ denotes the different time instants during the adaptation period.

The background images are updated as follows. We compute the background image closest to the image intensity $I^t(x, y)$ and update it

$$B_i^{t+1}(x, y) = \begin{cases} (1 - \alpha')B_i^t(x, y) + \alpha'I^t(x, y) & \text{if } (x, y) \in T \\ (1 - \alpha)B_i^t(x, y) + \alpha I^t(x, y) & \text{if } (x, y) \in N \end{cases} \quad (11)$$

where α, α' are update gains ($\alpha' < \alpha$), T is a target set, N is non-target set. α is usually small to enforce a slow adaptation. This is important to avoid the integration of active regions in the background image.

When updating the background image, the thresholds for pixels are also updated. Each pixel is first classified as false alarm, detection failure or target. Then the thresholds are updated

$$T_L^{t+1} = \begin{cases} T_L^t(x, y) + c_{FA} & \text{if false alarm} \\ T_L^t(x, y) - c_{DF} & \text{if detection failure} \\ T_L^t(x, y) & \text{if target} \end{cases} \quad (12)$$

In this paper we choose $c_{FA} = 10$, $c_{DF} = 1$. If the pixel is correctly classified as target the threshold remains the same. If the pixel is a false alarm the threshold is increased. If there is a detection failure the threshold is decreased.

4 Proposed Framework

In order to evaluate the performance of object detection algorithms we propose a procedure based on the following principles:

- A set of test sequences is selected. All moving objects are then detected and manually corrected if necessary to obtain the ground truth, one frame per second.
- The output of the automatic detector is compared with the ground truth.
- The output is then classified in one of the following classes: correct detection; false alarm; detection failure; merge; split; split-merge.

To perform the first step we made a user friendly interface which allows the user to define the foreground regions in the test sequence in a semi-automatic way. Fig. 1 shows

the interface used to generate the ground truth. A set of frames is extracted from the test sequence (one per second). An automatic object detection algorithm is then used to provide a tentative segmentation of the test images. Finally, the automatic segmentation is corrected by the user, by merging, splitting, removing or creating active regions.

In the case depicted in the Fig. 1, there are four active regions: a car, a lorry and two groups of persons. The segmentation algorithm also detects regions due to lighting changes, leading to a number of false alarms (four). The user can easily edit the image by adding and removing regions until a correct segmentation is obtained.



Figure 1. User interface used to create the ground truth from automatic segmentation results.

The test images are used to evaluate the performance of object detection algorithms. In order to compare the output of the algorithm with the ground truth segmentation, a region matching procedure is adopted which allows to establish a correspondence between the detected objects and the ground truth. Several cases are considered:

1. **Correct Detection (CD) or 1-1 match:** the detected region matches one and only one region.
2. **False Alarm (FA):** the detected region has no correspondence.
3. **Detection Failure (DF):** the test region has no correspondence.
4. **Merge Region (M):** the detected region is associated to several test regions.
5. **Split Region (S):** the test region is associated to several detected regions.
6. **Split-Merge Region (SM):** when the conditions 4, 5 are simultaneously satisfied.

4.1 Region Matching

Object matching is performed by computing a binary correspondence matrix \mathcal{C} which defines the correspondence between the active regions of a pair of images. Let us assume that we have N ground truth regions \tilde{R}_i and M detected regions R_j at time t . Under these conditions \mathcal{C} will be a $N \times M$ matrix, defined as follows

$$\mathcal{C}(i, j) = \begin{cases} 1 & \text{if } \tilde{R}_i \cap R_j \neq 0 \\ 0 & \text{if } \tilde{R}_i \cup R_j = 0 \end{cases} \quad \forall i \in \{1, \dots, N\}, j \in \{1, \dots, M\} \quad (13)$$

It is also useful to add the number of ones in each line or column, defining two auxiliary vectors

$$L(i) = \sum_{j=1}^M \mathcal{C}(i, j) \quad i \in \{1, \dots, N\} \quad (14)$$

$$C(j) = \sum_{i=1}^N \mathcal{C}(i, j) \quad j \in \{1, \dots, M\} \quad (15)$$

A match $\mathcal{C}(i, j) = 1$ can be classified as:

$$\begin{aligned} \text{correct detection: } & \text{if } L(i) = C(j) = 1 \\ \text{merge: } & \text{if } C(j) > 1 \\ \text{split: } & \text{if } L(i) > 1 \\ \text{split-merge: } & \text{if } L(i) > 1 \wedge C(j) > 1 \end{aligned} \quad (16)$$

The false alarms and detection failures can be formed detecting empty columns or lines in matrix \mathcal{C} . It is therefore easy to compute the statistics of each type of errors from matrix \mathcal{C} . Fig. 2 illustrates six situations considered in this analysis, by showing synthetic examples. Two images are shown in each case, corresponding to the ground truth (left) and detected regions (right). It is also depicted the correspondence matrix \mathcal{C} . For each case, the left image (\tilde{I}^t) contains the regions defined by the user (ground truth), the right image (I^t) contains the regions detected by the segmentation algorithm. Each region is represented by an white region containing a visual label. Fig. 2 (a) shows an ideal situation, in which each test region matches only one detected region (correct detection). In Fig. 2 (b) the “square-region” has no correspondence with the detected regions, thus it corresponds to a detection failure. In Fig. 2 (c) the algorithm detects regions which do not match any region of I^t generating a false alarm. In Fig. 2 (d) shows a merge of two regions since two different regions (“square” and “dot” regions in I^t) correspond to the “square region” in \tilde{I}^t . The remaining examples in this figure are self explaining, illustrating the split (e) and split-merge (f) situations.

Sometimes the segmentation procedure is subjective, since each active region may contain several objects and it

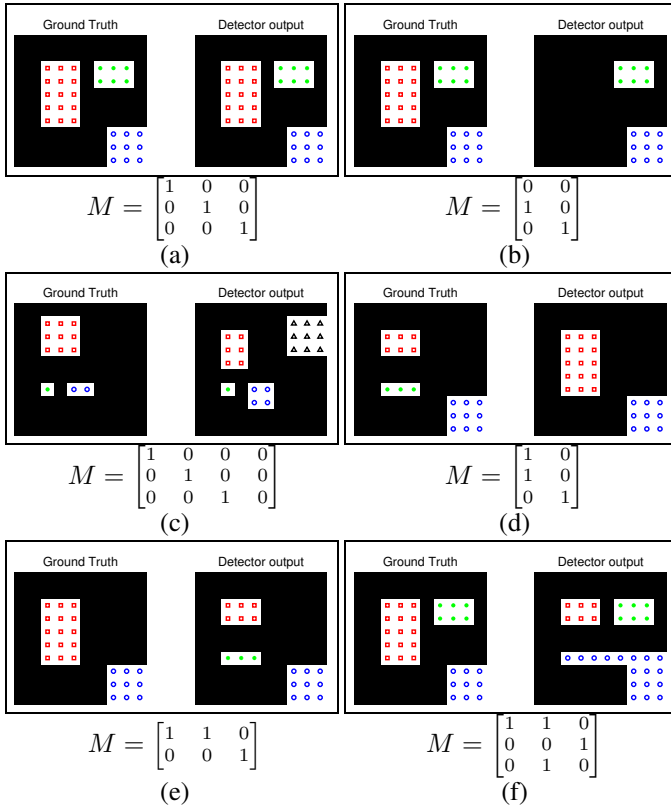


Figure 2. Different matching cases: (a) Correct detection; (b) Detection Failure; (c) False alarm; (d) Merge; (e) Split; (f) Split Merge.

is not always easy to determine if it is a single connected region or several disjoint regions. For instance, Fig. 3(left) shows an input image with a manual segmentation. Three active regions were considered: person, lorry and group of people. Fig. 3 (right) shows the segmentation results provided by the SGM algorithm. This algorithm splits the group into three individuals which can also be considered as a valid solution since there is very little overlap. This segmentation should be considered as an alternative ground truth. On the contrary, the situations depicted in Fig. 4 should be considered as errors. Fig. 4 shows the ground truth (left) and the segmentation provided by the W4 algorithm (right). The W4 algorithm makes a wrong split of the vehicle.

Another ambiguous example is shown in Fig. 5. This suggests the use of multiple interpretations for the segmentation. To accomplish this the evaluation setup takes into account all possible merges of single regions belonging to the same group whenever multiple segmentation hypothesis may occur in the frame, i.e., when there is a small overlap among the group members.

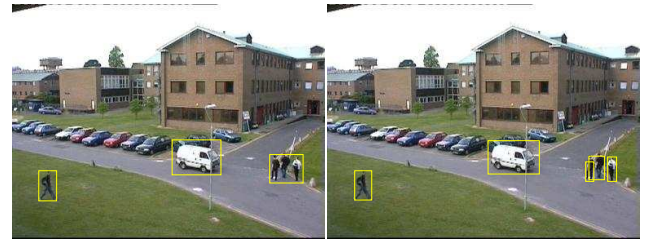


Figure 3. Correct split example, supervised segmentation and SGM segmentation.

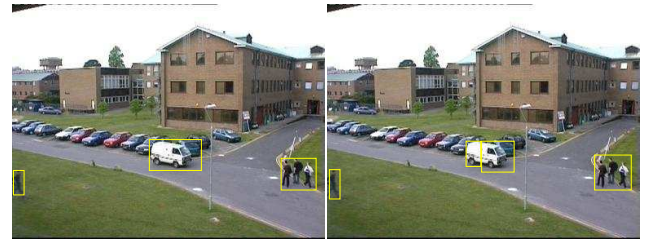


Figure 4. Wrong split example, supervised segmentation and W4 segmentation.

The number of merges depends on the relative position of single regions. Fig. 6 shows different merged regions corresponding to a group of three objects (each one representing a person in the group) when the relative position is different. In Fig. 6 (a) it is not necessary to consider the merge of the first and the third regions since the second region is in the middle. However, if the positions of the regions change (see Fig. 6 (b)) the number of links may be different. It is reasonable to assume that each region can be merged with the all others. In the automatic evaluation process, it is enough for the user, to give single regions and the interpretations are generated automatically. Figs. 7,8 illustrate this situation. Fig. 7(a) shows the input frame, Fig. 7(b) shows the hand segmented image, and Fig. 7(c) illustrates the output of the SGM. Fig. 8 shows all possible merges of individual regions. All of them are considered as correct. Remain to know which segmentation should be selected to appraise the performance. In this paper we choose the best segmentation, which is the one that provides the highest number of correct detections. In the present example the segmentation illustrated in Fig. 8 (g) is selected. In this way we overcome the segmentation ambiguities that may appear without penalizing the algorithm.

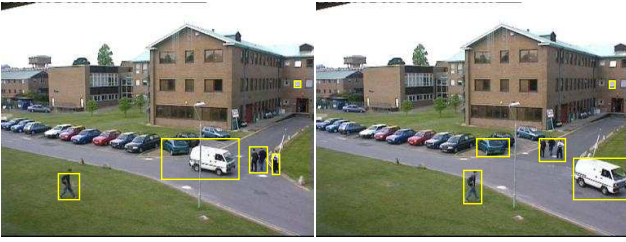


Figure 5. Output of the SGM method at two time instants.

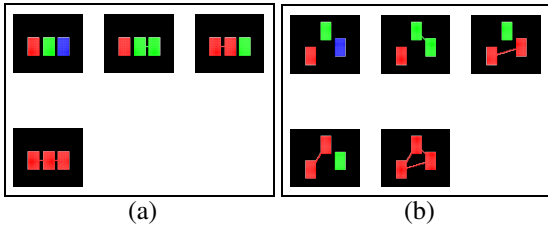


Figure 6. Regions linking procedure. The same number of foreground regions may have different interpretations: three possible configurations (a), or four configurations (b). Each color represent a different region.

5 Tests on PETS2001 dataset

The segmentation algorithms described in this paper were evaluated using PETS2001 and PETS2004 dataset with resolution 384×288 pixels. The ground truth was generated by segmenting one image per second and correcting the automatic segmentation results using the graphical editor described before. The output of the algorithms was then compared with the ground truth. The active regions with less than 25 pixels were eliminated.

The segmentation algorithms described herein depend on a set of parameters: thresholds and learning rate α . To find appropriate values for these parameters, we produced ROC curves which display the performance of each method as a function of the parameters. Each ROC is built by keeping all the parameters constant but one. This requires a considerable amount of tests, which were done using a training sequence of the PETS2001 data set. Once the parameters are set, we use these values to evaluate the algorithms in a test sequence of PETS2001.

ROC curves describe the evolution of the false alarms (FA) and detection failures (DF) as T varies. An ideal ROC should be close to the origin, i.e., with small area.

Fig. 9 shows the receiver operating curves (ROC) for the

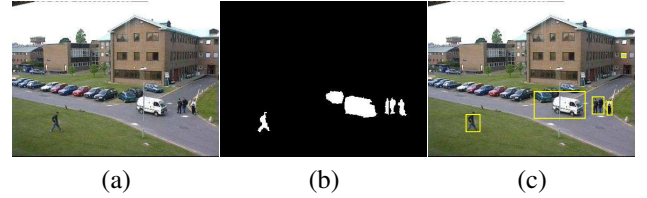


Figure 7. Input frame (a), segmented image by the user (b), output of SGM (c).

best value of α for different values of the threshold. These tests were performed on the training sequence. The BBS algorithm is sensitive to the threshold (see 9(a)). Small changes of T leads to large variations of false alarms and detection failures. The best value is $T = 0.2$. Fig. 9(b) shows the ROC of the SGM method, for $\alpha = 0.005$. This method is more robust than the BBS algorithm with respect to the threshold. A smooth variation of FA and DF is obtained for $-400 < T < -150$. We choose $T = -400$. Fig. 9(c) depicts the results of the MGM method. We notice that the algorithm strongly depends on the value of T , since for small variations of T there are significant changes of FA and DF. The best performance is obtained for $T > 0.9$.

Fig. 9(d) displays the results of the LOTS algorithm for a variation of the sensitivity from 10% to 110%. We use a small blending parameter. LOTS does not update the background image in every single frame to avoid a high computational cost. This algorithm only updates the background every N frames instead. We used an integration factor $\alpha = 6.1 \times 10^{-5}$ which corresponds to add 0.0625 of the current frame to the background in every 1024th frame.

All methods have a large number of false alarms. In this sequence there is a static car which suddenly starts to move. Since the background is slowly updated, this event produces a ghost active object which is detected in a large number of frames.

Table 1 shows the results obtained on the test sequence using the parameters obtained from the ROC curves. In terms of false alarms the BBS method is the worst and the W4 is the best one. The main characteristics of the BBS method is that it tends to detect everything that moves in the scene. As a consequence it has a high percentage of correct detections but high false alarms rate.

The highest percentage of correct detections is achieved by LOTS followed by SGM. In the detection failures the W4 outperforms the others. W4 exhibits perhaps the best tradeoff between CD and FA. However, this method tends to split regions. This happens in situations in which the objects have a slow motion or when they stop, since the method is not able to remove the corresponding regions from the background model. This is the main drawback of the method. In

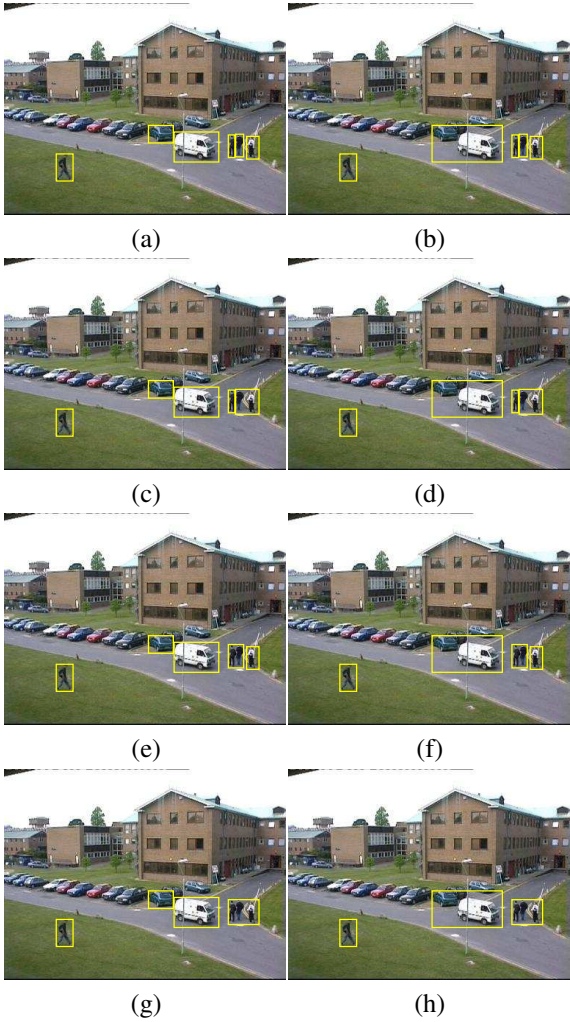


Figure 8. Multiple interpretations given by the application. The segmentation illustrated in (f) is selected for the current frame.

term of merges, none of the algorithms studied here has a tendency to merge regions.

Comparing the results of false alarms between the Fig. 9 and the table 1, we notice that all the algorithms exhibit a larger percentage of false alarms in the training sequence (see Fig. 9) than in the test sequence (see table 1). For comparison purposes we also computed the ROC curves using the test sequence. These results are shown in Fig. 10.

The choice of the method depends on the application. For instance the LOTS, SGM and W4 are well suited for tracking applications. Although the W4 method is sensible to splits, this is not a serious drawback in tracking since the system can always track one of the detected regions. However, if the application involves object recognition with

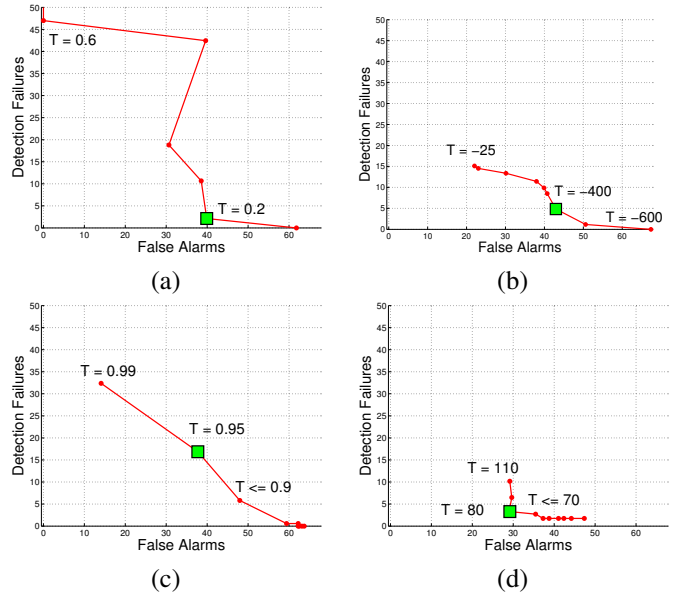


Figure 9. Receiver Operation Characteristic for different values of α (training sequence). First row: (a) BBS, $\alpha = 0.15$, (b) SGM, $\alpha = 0.15$, second row (c) MGM, $\alpha = 0.008$, (d) LOTS, $\alpha = 6.1 \times 10^{-5}$.

global features (e.g., histograms) W4 is not suitable.

A second set of tests were performed using different sequences and metrics. The five algorithms were evaluated using the sequence Walk1 from PETS2004 data set and evaluated using CAVIAR metrics [1]. These metrics compare the bounding boxes of the detected regions with the bounding boxes of the ground truth and compute the statistics for true detections, misdetections and false alarms. These results are shown in Table 2 for an overlap requirement of 20%. These results are compatible with the probability of correct detections previously obtained with the metrics proposed in this paper (the LOTS and SGM provide the best set of results as before). However, they are not enough to understand what types of errors are made by the algorithms and if they are relevant or not for the other processing blocks. This kind of information can be obtained from the metrics proposed in this paper.

6 Conclusions and future work

This paper proposes new metrics for the evaluation of object detection algorithms in surveillance applications. The proposed methodology is based on the comparison of the detector output with the ground truth segmentation of test sequences followed by a classification of the errors.

	<i>BBS</i>	<i>W4</i>	<i>SGM</i>	<i>MGM</i>	<i>LOTS</i>
Correct Detections	83.5	76.7	87.9	74.8	92.4
Detection Failures	8.6	4.8	9.2	15.9	5.9
Splits	2.1	10.1	0.2	5.5	0.2
Merges	0	0.6	0	0	0.9
Split/Merges	6.2	6.9	3.0	3.4	0.2
False Alarms	21.7	1.1	11.1	10.7	7.9

Table 1. Performance of five object detection algorithms.

	<i>BBS</i>	<i>W4</i>	<i>SGM</i>	<i>MGM</i>	<i>LOTS</i>
True Detections	94.4	51.3	94.9	89.7	94.9
Missed Detections	5.5	48.6	5.0	10.2	5.0
False Detections	38.2	0.1	5.9	150.1	5.2

Table 2. Results using the statistics of the Caviar Project with an overlap requirement of 20%.

The performance evaluation is made in terms of achieving the best tradeoff between correct detection and false alarms. Although we find LOTS and SGM the most suitable algorithms to perform region segmentation, the choice depends on the context of the application.

These tests should be further extended to consider other sequences. It would also be interesting to enlarge the set of methods and to characterize the effect of each type of error on the performance of the overall system. The measurements proposed herein are important to characterize the performance of object detection algorithms.

Another important issue is that the proposed framework can also be used to evaluate tracking algorithms. To accomplish this, it is enough to record the trajectory of the detected regions using the ground truth.

Acknowledgement: We thank Dr. Thor List of Edinburgh University for performing the the statistics of Table 2. We also thank Prof. José Santos Victor for providing valuable information about the CAVIAR activities.

References

- [1] <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>.
- [2] T. Boulton, R. Micheals, X. Gao, and M. Eckmann. Into the woods: Visual surveillance of non-cooperative camouflaged targets in complex outdoor settings. pages 1382–1402, 2001.

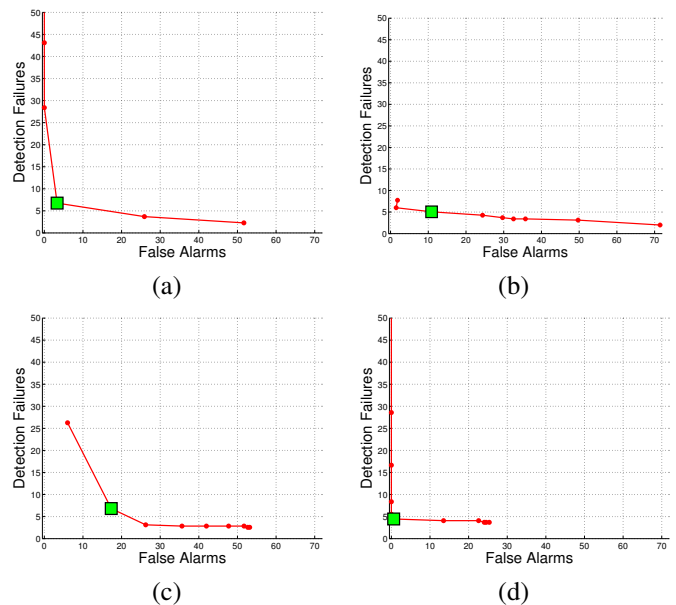


Figure 10. Receiver Operation Characteristic on the test sequence using the same values of α : (a) BBS, (b) SGM, (c) MGM, (d) LOTS.

- [3] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa. A system for video surveillance and monitoring. tech. report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, May 2000.
- [4] R. Cucchiara, C. Grana, and A. Prati. Detecting moving objects and their shadows: an evaluation with the pets2002 dataset. Proceedings of Third IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS 2002) in conj. with ECCV 2002, pages 18–25, May 2002.
- [5] I. Haritaoglu, D. Harwood, and L. S. Davis. w^4 : Who? when? where? what? a real time system for detecting and tracking people. IEEE International Conference on Automatic Face and Gesture Recognition, pages 222–227, April 1998.
- [6] I. Haritaoglu, D. Harwood, and L. S. Davis. w^4 : real-time surveillance of people and their activities. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 22(8):809–830, August 2000.
- [7] S. J. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80(1), 2000.
- [8] C. Reggazzoni, G. Foresti, and A. Venetsanopoulos. Shape representation from image sequences by using binary statistical morphology. volume II of *Proceedings ICIP-94*, pages 106–110, Los Alamitos, 1994. IEEE Computer Society Press.
- [9] N. T. Siebel and S. J. Maybank. Real-time tracking of pedestrians and vehicles. Proc. of IEEE workshop on Performance Evaluation of tracking and surveillance, 2001.
- [10] C. Stauffer, W. Eric, and L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 22(8):747–757, August 2000.
- [11] H. V. Trees. *Detection, Estimation, and Modulation Theory*. John Wiley and Sons, 1968.
- [12] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfunder: Real-time tracking of the human body. *IEEE Trans. Pattern Anal. Machine Intell.*, 19(7):780–785, July 1997.

Trajectory Distance Metric Using Hidden Markov Model based Representation

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

Abstract

In this paper, we introduce a set of novel distance metrics that use model based representations for trajectories. We determine the similarity of trajectories using the conformity of the corresponding HMM models. These metrics enable the comparison of trajectories without any limitations of the conventional measures. They accurately identify the coordinate, orientation, and speed affinity. The proposed HMM based distance metrics can be used not only for ground truth comparisons but for clustering as well. Our experiments prove that they have superior discriminative properties.

1. Introduction

Recent advances in object tracking made it possible to obtain spatiotemporal motion trajectories for further analysis of concealed information. Although the extraction of trajectories is well understood and studied, relatively little investigation on the precise comparison of the trajectories and the secondary outputs of the tracking process is presented in the literature.

A key issue in performance evaluation of tracking results is the distance metric that determines the similarity of the trajectories. Any additional analysis, such as action recognition, event detection, etc., highly depends on the accuracy of the similarity assessment. Most existing measures [2], [6] compute a mean distance of the corresponding positions of two equal duration trajectories. Supplementary statistics such as variance, median, minimum, and maximum distances are also suggested to extend the description of similarity. In a recent work, Needham [4] proposed an alignment based distance metric that reveals the spatial translation and temporal shift between the given trajectories, and introduced an area based metric that calculates the total enclosed area between the trajectories using trajectory intersections. Similarly, Ellis [1] characterized several statistical properties of the tracking performance using the compensated means and standard deviations.

One main disadvantage of the existing approaches is that they are all limited to the equal duration (lifetime) trajectories. By duration we refer the number of coordinate points that constitute the trajectory. These coordinates are sampled at different time instances. Since the existing measures depend on the mutual coordinate correspondences, they cannot be applied to trajectories that have different durations unless the trajectory duration is normalized or parameterized first. However, such a normalization destroys the temporal properties of the trajectory.

Conventional distance measures assume that the temporal sampling rates of the trajectories are equal. For instance, a ground truth trajectory labeled at a certain frame rate can be compared only with the trajectory generated by a tracker working at the same frame rate. These approaches do not cope with the uneven sampling instances, i.e. varying temporal distance between the coordinates, either. This is a common case especially for the real-time object trackers that process streaming video data. A real time tracker works on the next available frame, which may not be the immediate temporal successor of the current, whenever the current frame is processed. Thus, the obtained trajectory coordinates have varying temporal distance.

Therefore, there is a need to develop other alternatives that can effectively measure the difference between unrestricted trajectories. In this paper, we introduce a set of novel distance metrics that use model based representations. We determine the similarity of trajectories using the conformity of the corresponding models. We construct a mixture of continuous Hidden Markov Models (HMM) that capture the dynamic properties of trajectory within a state transition matrix. The HMM based metrics enable the comparison of trajectories without any limitations of the previous measures. We can use the proposed metrics not only for ground truth comparisons but for clustering as well. We measure the similarity of trajectories that have different durations, sampling rates, and temporal properties. We accurately identify coordinate, orientation, and speed affinity. We also propose additional features that are extracted from the trajectories such as object-wise histograms of aspect-

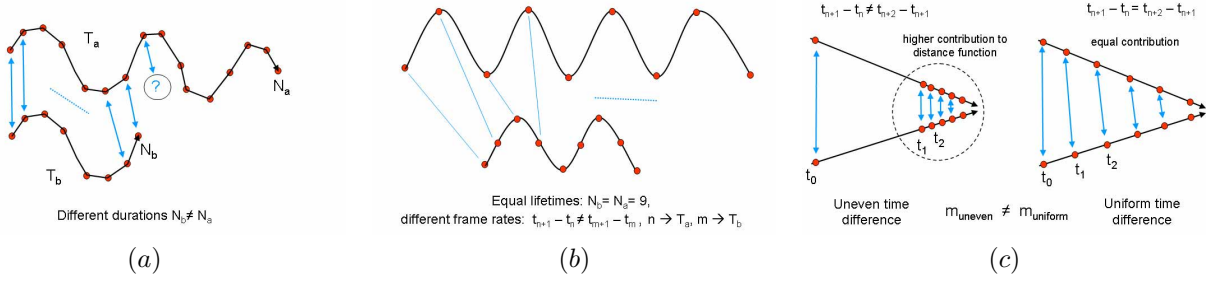


Figure 1. Ambiguous cases for conventional metrics; (a) unequal durations, (b) equal durations but different frame rates. (c) Effect of uneven frame rates.

ratio, location, orientation, speed, size, etc. to improve the available features.

In section 2, we discuss the existing trajectory distance metrics. In section 3 we introduce the additional features. In section 4 we present the HMM based distance metrics, and in the last section we discuss the experimental results.

2. Trajectory Distance Measures

A trajectory is a time sequence of coordinates representing the motion path of an object over the duration (lifetime), i.e. number of frames that object exists. These coordinates correspond to marked positions of object shape in consecutive frames. A marked position often indicates the center-of-mass (for pixel model), the intersection of main diagonals (for ellipsoid model), and the average of minimum and maximum on perpendicular axes (for bounding box model) of object region. It is, therefore, possible to view the trajectory as a collection of frame-wise abstractions of object shape. We will adopt the following notation $T : \{p_n\} : \{(x_1, y_1, t_1), (x_2, y_2, t_2), \dots, (x_N, y_N, t_N)\}$ where N is the duration.

The simplest metric used for computing the distance between a pair of trajectories is the mean of coordinate distances, which is given as

$$m_1(T^a, T^b) = \frac{1}{N} \sum_{n=1}^{n=N} d_n^2 \quad (1)$$

where the displacement between the positions is calculated using the Cartesian distance

$$d_n^2 = [(x_n^a - x_n^b)^2 - (y_n^a - y_n^b)^2]^{\frac{1}{2}}, \quad (2)$$

or using other L-norm formulations

$$d_n^L = [(x_n^a - x_n^b)^L - (y_n^a - y_n^b)^L]^{\frac{1}{L}}. \quad (3)$$

Note that, the mean distance metric makes three critical assumptions; 1) the durations of the both trajectories are same

$N^a = N^b = N$ (fig. 1-a), 2) the coordinates are synchronized $t_n^a = t_n^b$ (fig. 1-b), and 3) the time sampling rate is constant $t_n^a - t_{n+1}^a = t_m^b - t_{m+1}^b$ since the contribution of each distance component d_n in equation 1 is same as illustrated in fig. 1-c. It is evident that the mean of distances is very sensitive to the partial mismatches and cannot deal with the distortions in time.

To provide more descriptive information, the second order statistics such as median, variance, minimum and maximum distance may be incorporated. The variance is defined as

$$m_2(T^a, T^b) = \frac{1}{N} \sum_{n=1}^N (d_n - m_1(T^a, T^b))^2. \quad (4)$$

To find the median, the displacements d_n are ordered with respect to their magnitudes as $d_n \rightarrow d_m$, then the value of the halfway component of the list is assigned

$$m_3(T^a, T^b) = \begin{cases} d_{\frac{N+1}{2}} & N \text{ odd} \\ \frac{1}{2}(d_{\frac{N}{2}} + d_{\frac{N+1}{2}}) & N \text{ even} \end{cases}$$

The minimum and maximum distances are simply defined as

$$m_4(T^a, T^b) = \min d_n \quad (5)$$

$$m_5(T^a, T^b) = \max d_n \quad (6)$$

Although these statistics supply extra information, they inherit (even amplify) the shortcomings of the ordinary mean of distances metric m_1 . Besides, none of the above metrics is sufficient enough by itself to make an accurate assessment of the similarity.

An area based distance metric is proposed in [4]. The crossing points $q : T^a(p_i) = T^b(p_j)$ of two paths are used to define regions Q_j $j = 1, \dots, J$ between the trajectories. For each region, a polygon model is generated and the enclosed area is found by tracing the parameterized shape

$$m_6(T^a, T^b) = \sum_{j=1}^J \text{area}(Q_j) \quad (7)$$

This metric can handle more complex trajectories, however it is sensitive to entanglements of the trajectory, it discards the time continuity, and fails to distinguish two trajectories in opposite directions. Although the area between a pair of trajectories is easily apprehended, its computation may demand case-specific analytic solutions that are not always straightforward to formulate.

It is possible to compute an optimal spatiotemporal alignment $(\delta x, \delta y, \delta t)$ for which the mean distance is minimized

$$(\delta x, \delta y, \delta t) = \arg \min m_1(T^a, T^b + (\delta x, \delta y, \delta t)) \quad (8)$$

and use this alignment to compute a compensated distance

$$m_7(T^a, T^b) = m_1(T^a, T^b + (\delta x, \delta y, \delta t)). \quad (9)$$

Ellis [1] proposed several statistical measures such as true detection rate, false alarm rate, etc. using the aligned trajectories for comparison of a trajectory with the ground truth. However, not all trajectory distance tasks involve ground truth comparison, i.e. clustering.

In the following sections, we introduce an extended set of trajectory based features, and then we present the details of the HMM based metrics.

3. Trajectory Based Features

A set of coordinates is not the only available trajectory feature. In spite of its simplicity, duration (lifetime) is a distinctive feature. For instance, at a hallway camera in a surveillance setting the suspicious event may be a left behind unattended bag, which can be easily detected since human objects do not stay still for extended periods of time. The total length of the trajectory is defined as $\sum_{n=2}^N |T(p_n) - T(p_{n-1})|$. This is different from the total displacement of the object, which is equal to $|T(p_N) - T(p_1)|$. By assuming a ground plane of the camera imaging system is available, the trajectory may be projected to obtain the respective 3D length. A total orientation descriptor keeps the global direction of the object. Depending on the camera setup, the length related descriptors may be used to differentiate unusual paths. The length/duration ratio gives the average speed.

Dynamic properties of an object such as orientation, aspect ratio, size, instantaneous speed, and location are represented by histograms. The location histogram keeps track of the image coordinates where object stays most. Using the size histogram, dynamic properties of the object size are captured, e.g. we can separate an object moving towards the camera (assuming the size will get larger) from another object moving parallel. An object moves at different speeds during tracking, therefore the instantaneous speed of an object is accumulated into a histogram. Speed is the key as-

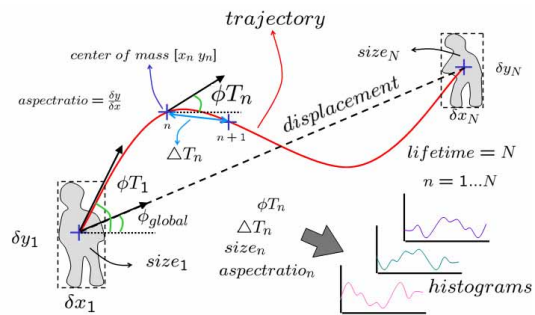


Figure 2. Additional trajectory features.

pect of some events, e.g. a running person where everybody walks. The speed histogram may be used to interpret the regularity of the movement such as erratically moving objects. An accident can be detected using the speed histogram; the speed components will accumulate around zero and high velocities rather than being distributed uniformly.

The orientation histogram is one of the important descriptors. For instance, it becomes possible to distinguish objects moving on a certain path, making circular movements, etc. It is possible to find a vehicle backing up on a wrong lane then driving correctly again, which may not be detected using a global orientation. The aspect ratio is a good descriptor to distinguish between human objects and vehicles. The aspect ratio histogram can capture whether a person crouches and stands up during its lifetime. Figure 2 illustrates some of the object features.

4. Hidden Markov Model Based Metric

Due to the shortcomings of the existing metrics, we propose a model based representation that captures the dynamic properties of trajectories. We project each trajectory T into a parameter space λ that is characterized by a set of HMM parameters.

An HMM is a probabilistic model composed of a number of interconnected states a directed graph, each of which emits an observable output. Each state is characterized by two probability distributions: the transition distribution over states and the emission distribution over the output symbols. A random source described by such a model generates a sequence of output symbols as follows: at each time step the source is in one state, and after emitting an output symbol according to the emission distribution of the current state, the source jumps to a next state according to the transition distribution of its current state. Since the activity of the source is observed indirectly, through the sequence of output symbols, and the sequence of states is not directly observable, the states are said to be hidden.

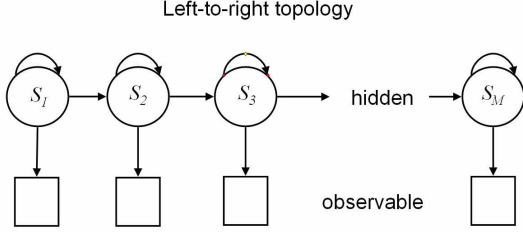


Figure 3. Left-to-right topology.

In our case, we replace the trajectory information as the emitted observable output for the above directed graph. The hidden states then capture the transitive properties of the consecutive coordinates of the spatiotemporal trajectory. The state sequence that maximizes the probability becomes the corresponding model for the given trajectory.

A simple specification of an K -state $\{S_1, S_2, \dots, S_K\}$ continuous HMM with a Gaussian observation is given by:

1. A set of prior probabilities $\pi = \{\pi_i\}$ where $\pi_i = P(q_1 = S_i)$ and $1 \leq i \leq K$.
2. A set of state transition probabilities $B = \{b_{ij}\}$, where $b_{ij} = P(q_{t+1} = S_j | q_t = S_i)$ and $1 \leq i, j \leq K$.
3. Mean, variance and weights of mixture models $\mathcal{N}(O_t; \mu_j, \Sigma_j)$ where μ_j and Σ_j are the mean and covariance of the state j .

Above, q_t and O_t are the state and observation at time t , respectively.

For each trajectory T^a , we fit an M -mixture HMM $\lambda^a = (\pi, B, \mu, \Sigma)^a$ that has left-to-right topology using the Baum-Welch algorithm. We chose the left-to-right topology since it can efficiently describe continuous processes.

We train a HMM model using the trajectory as the training data after we initialize the state transition and prior probability matrices with random variables, thus we make no assumptions on the trajectory. Initialization can be adapted for specific applications as well. Finally, each trajectory is assigned to a separate model.

The optimum number of states and mixtures depend on the complexity and duration of the trajectories. To provide sufficient evidence to every Gaussian of every state in the training stage, the lifetime of the trajectory should be much larger than the number of mixtures times number of states $N \gg M \times K$. A state can be viewed as a basic pattern of the trajectory, thus depending the trajectory, the number of states should be large enough to conveniently characterize such distinct patterns but small enough to prevent from overfitting.

A priori knowledge about tracking scenario may be used to impose a structure on an HMM and a meaning for the

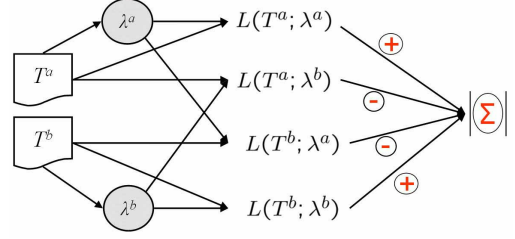


Figure 4. Cross-fitness distance.

values of the state variable. It is known that each state may be associated with a certain label. Furthermore, the topology of the HMM can be strongly constrained: most transition probabilities are forced to be zero. Since the number of free parameters and the amount of computation are directly dependent on the number of non-zero transition probabilities, imposing such structure is very useful when it is appropriate. The most basic structure that is often imposed on HMM's is the left-to-right structure: states are ordered sequentially and transitions go from the "left" to the "right", or from a state to consecutive state or itself, as in fig. 3.

We search an optimal number of states of the HMM network for the given trajectory while repeating the generation and evaluation of the topology. At the beginning of the search, possible HMM's up to a maximum number of states are generated randomly. In general, the likelihood of HMM increases with the complexity of the topology. However, it is known that over representation is frequently observed as the complexity increases. Therefore, in order to balance the likelihood and the complexity, we have adopted a score [3] as

$$v_i = [-\log L(T; \lambda_i) + i\alpha]^{-1} \quad (10)$$

where $i = 2, \dots, M_{max}$ is the number of states, $L(T; \lambda_i) = P(T|\lambda_i)$ is the likelihood obtained for HMM with i -states, and α is a constant balancing factor. The number of states is then chosen as the one that given the highest score.

We define the distance between two trajectories in terms of their HMM parameterizations:

$$m_8(T^a, T^b) = |L(T^a; \lambda^a) + L(T^b; \lambda^b) - L(T^a; \lambda^b) - L(T^b; \lambda^a)| \quad (11)$$

which corresponds the cross-fitness of the trajectories to each other's models as illustrated in fig. 4. The $L(T^a; \lambda^a)$ and $L(T^b; \lambda^b)$ terms indicate the likelihood of the trajectories to their own fitted model, i.e. we obtain the maximum likelihood response for the models. The cross terms $L(T^a; \lambda^b)$, $L(T^b; \lambda^a)$ reveal the likelihood of a trajectory generated by the other trajectories model. In other words, if two trajectories are identical, the cross terms will have a

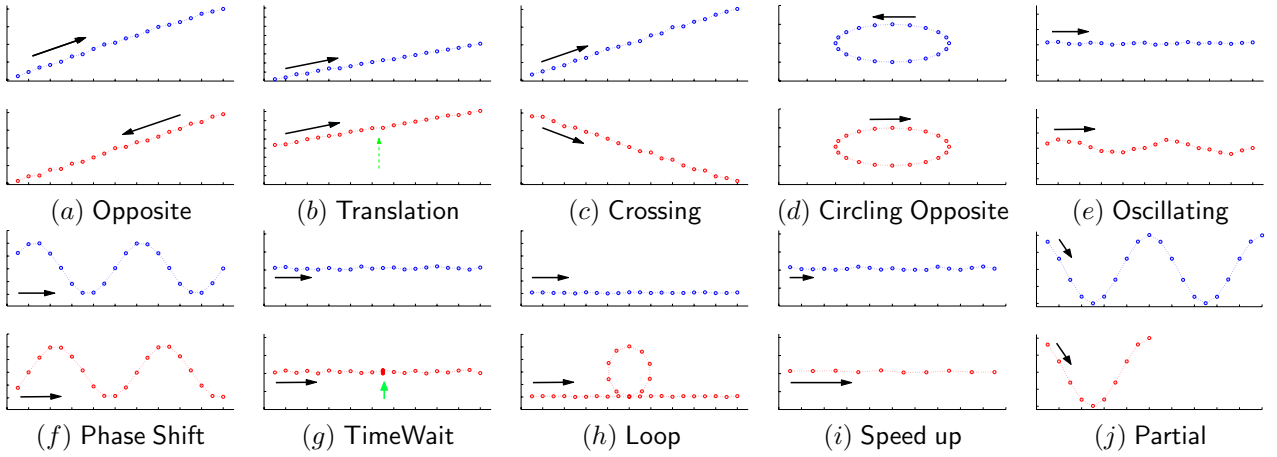


Figure 5. Different trajectory pattern pairs.

maximum value, thus eq. 12 will be equal to zero. On the other hand, if trajectories are different, their likelihood of being generated from each others model will be small, thus the distance will be high.

Up to now, we employed trajectory coordinates as our feature sequence. Using coordinates reveals spatial correlation between trajectories, however in some situations it is more important to distinguish shape similarity of the trajectories independent of the spatial coordinates. The instantaneous speed and orientation sequences are potential features that establish shape similarity even if there is a spatial translation. Thus, we define two other sequential features and corresponding distances; the orientation sequence as

$$\begin{aligned}\phi T(p_n) &= \tan^{-1} \frac{y_n - y_{n-1}}{x_n - x_{n-1}} \\ m_9(T^a, T^b) &= m_8(\phi T^a, \phi T^b)\end{aligned}\quad (12)$$

and the speed sequence

$$\begin{aligned}\Delta T(p_n) &= [(x_n - x_{n-1})^2 + (y_n - y_{n-1})^2]^{\frac{1}{2}} \\ m_{10}(T^a, T^b) &= m_8(\Delta T^a, \Delta T^b).\end{aligned}\quad (13)$$

The mentioned HMM distance is also applicable to histogram features such as orientation histogram, speed histogram, etc. However, since these features discard the temporal ordering of the points, they are more suitable to evaluate the statistical properties of trajectories rather than measuring the similarity of their shape and coordinates.

5. Comparisons

To compare the proposed metrics m_8, m_9, m_{10} and the referenced conventional metrics m_1, \dots, m_7 , we computed the distances of several distinct trajectory pattern pairs as

presented in fig. 5. Equal (a-f) and unequal (g-j) duration trajectories are among these patterns. Each equal duration trajectory consists of 100 points. To make the comparison more realistic, we added a random white noise to all patterns. The first set of equal duration patterns include the trajectory pairs that are in opposite direction, spatially shifted trajectories, trajectories that are crossing each other, trajectories that have the same circling path but in opposite direction, trajectories that their global orientation is same but their paths have small perturbations, and trajectories that the form is same except a time shift. The second set of trajectory pairs have different durations. For instance, fig. 5-g shows a pair that have same spatial path but one of the trajectory has a several frames long waiting period as shown with the green arrow. Fig. 5-h shows a pair that are same spatial form except one trajectory has a loop. In fig. 5-i the second trajectory has the same form but its duration is half of the first one. In fig. 5-j a partially matching pair is given.

After we computed the distances of all pairs for a given metric, we normalized the distances using the maximum distance obtained for that metric since there is no a common normalization factor that can applied to all the metrics. For instance, the numerical values of the variance (m_3) and the area (m_6) metrics are clearly incommensurate. Thus, we evaluate the sensitivity based on the given pattern set. We listed the normalized responses of all metrics in table 1. The highest score at each column indicates the pattern that the metric is most sensitive. Note that, an ideal metric should be applicable to all diverse patterns regardless of the trajectory duration, frame-rate, and other limitations.

From the table, it is evident that the sum of coordinate distances m_1 , the variance of coordinate distances m_2 , and the median coordinate distance m_3 have all similar properties. Their fusion would not improve the overall discriminative capability. The maximum distance m_4 and minimum

Table 1. Comparison of Distance Metrics

	m_8	m_9	m_{10}	m_1	m_2	m_3	m_4	m_5	m_6	m_7
Opposite (ED)	0.123	1.000	0.001	1.000	1.000	1.000	0.055	1.000	1.000	0.001
Translation (ED)	0.356	0.001	0.006	0.283	0.001	0.287	1.000	0.148	0.002	0.573
Crossing (ED)	1.000	0.370	0.002	0.707	0.502	0.721	0.016	0.707	0.677	1.000
Circling (ED)	0.008	0.105	0.000	0.449	0.143	0.491	0.000	0.355	0.403	0.000
Perturbation (ED)	0.001	0.027	0.012	0.017	0.000	0.018	0.001	0.014	0.417	0.139
Phase shift (ED)	0.073	0.001	0.002	0.107	0.008	0.123	0.029	0.085	0.020	0.226
Wait (VD)	0.069	0.071	0.316	-	-	-	-	-	-	0.001
Loop (VD)	0.389	0.529	0.775	-	-	-	-	-	-	0.001
Speed up(VD)	0.001	0.214	1.000	-	-	-	-	-	-	0.003
Partial (VD)	0.198	0.001	0.002	-	-	-	-	-	-	0.000

(Each column is normalized within itself, ED: equal duration, VD: variable duration)

distance m_5 are very sensitive to singularities, for instance the maximum distance can be very high even a the trajectories have matching well except a single coordinate. The minimum distance fails if a single crossing exists. The spatiotemporal alignment metric m_7 is insensitive to shifting, otherwise it is similar to m_1 . These metrics cannot handle different duration trajectories. The area metric m_6 fails for patterns that have same path but opposite direction. It cannot distinguish the temporal deformations either.

On the other hand, the HMM based metrics are applicable to trajectories that have different durations. It is shown that these metrics can successfully identify various temporal deformations including the time waiting, partial match, different speed, time loop, etc. Each topology has 3 states and 3 Gaussian models. The coordinate based HMM metric m_8 is sensitive towards the spatial positioning of the trajectories, and it can identify the crossing, translation, phase shift, time loop, partial, and opposite directions. The orientation based HMM metric m_9 is responsive towards the orientation variances, i.e. it gave the highest score to opposite direction pattern, and it can recognize crossing, time loop, and circling patterns. The speed based HMM m_{10} detects the speed changes and time loops most effectively, and it can identify the uneven frame-rates as well.

We observed that the three possible HMM metrics are responsive towards the different patterns, thus their mixture is a perfect candidate for measuring the trajectory distance.

We conducted another experiment using the PETS-2004 benchmark sequences. For the sequences that the ground truth is not given, we obtained the trajectories by our mean-shift based tracker [5]. The trajectories, which have duration ranging from 30 to 800 points, are presented in fig. 6. We determined the most similar and most different trajectories to a given trajectory using the m_8 metric as shown in fig. 7. In the graphs, the red is the given trajectory. The blue is the most similar and green is the most different trajectory among the all trajectories. As visible, the proposed

HMM based metric accurately identified the most similar and dissimilar trajectories at each case.

6. Conclusion

We proposed a set of HMM based trajectory distance metrics that can accurately measure the coordinate, orient, and speed similarity of a pair of trajectories. These metrics measure different duration trajectories without destroying the temporal properties. They can be used not only for ground truth comparisons but also for further analysis of the tracking results, e.g. clustering and event analysis. Our experiments prove that the HMM distance metrics have superior discriminative properties than conventional metrics.

References

- [1] T. Ellis. Performance metrics and methods for tracking in surveillance. *Proc. of PETS*, pages 26–31, Copenhagen, Denmark, June 2002.
- [2] C. Jaynes, S. Webb, R. Steele, and Q. Xiong. An open development environment for evaluation of video surveillance systems. *Proc. of PETS*, Copenhagen, Denmark, June 2002.
- [3] A. Konagaya and Y. Kondo. Stereo person tracking with adaptive plan-view statistical templates. *Hawaii Int. Conf. on System Sciences*, pages 746–755, 1993.
- [4] C. Needham and R. Boyle. Performance evaluation metrics and statistics for positional tracker evaluation. *Proc. of ICVS*, pages 278–289, Graz, Austria, April 2003.
- [5] F. Porikli and O. Tuzel. Performance evaluation metrics and statistics for positional tracker evaluation. *Proc. of PETS*, pages 37–45, Graz, Austria, April 2003.
- [6] A. Senior, A. Hampapur, Y. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. *Proc. of PETS*, Hawaii, Kauai, December 2001.

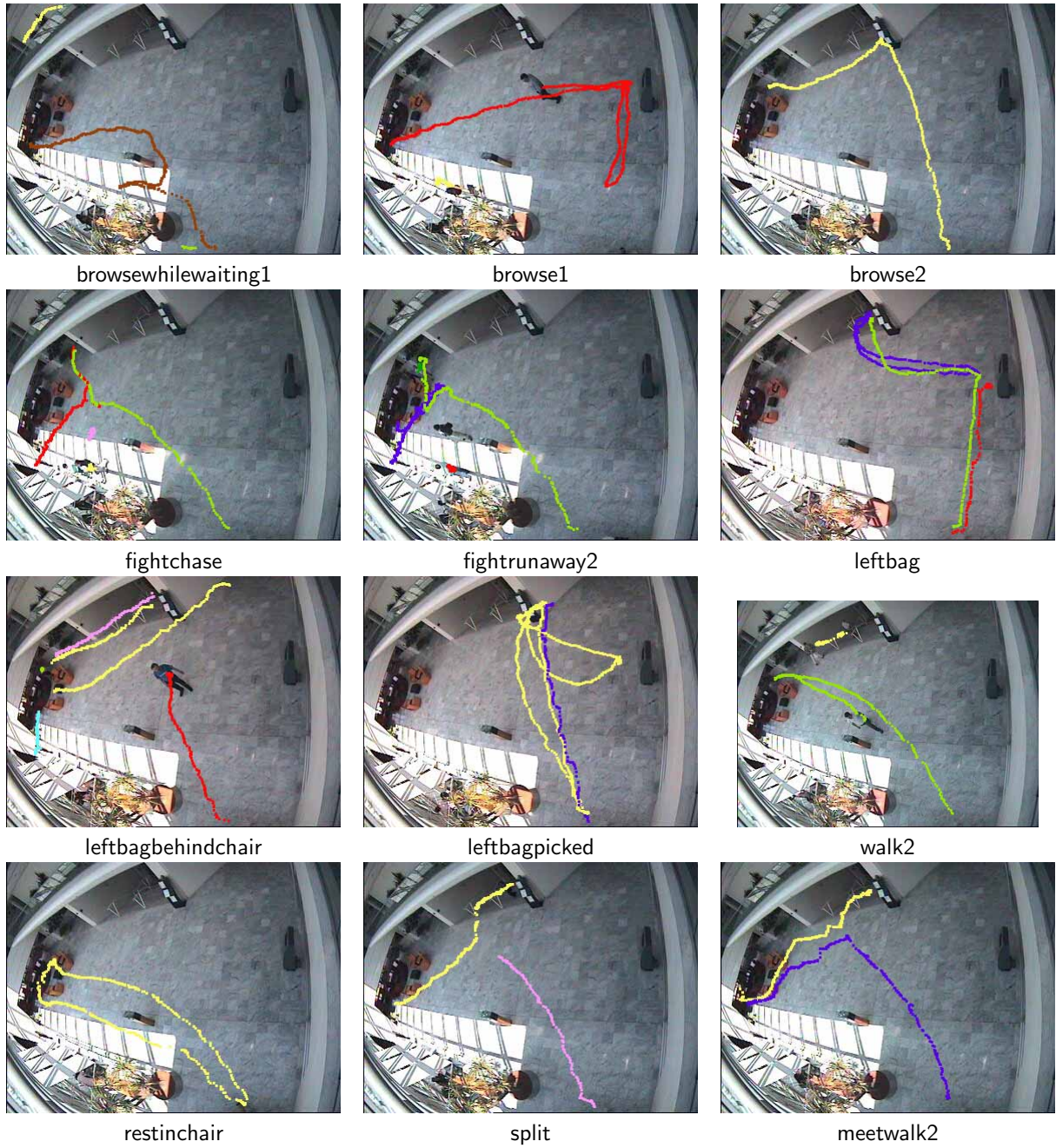


Figure 6. Detected trajectories for the PETS-2004 sequences.

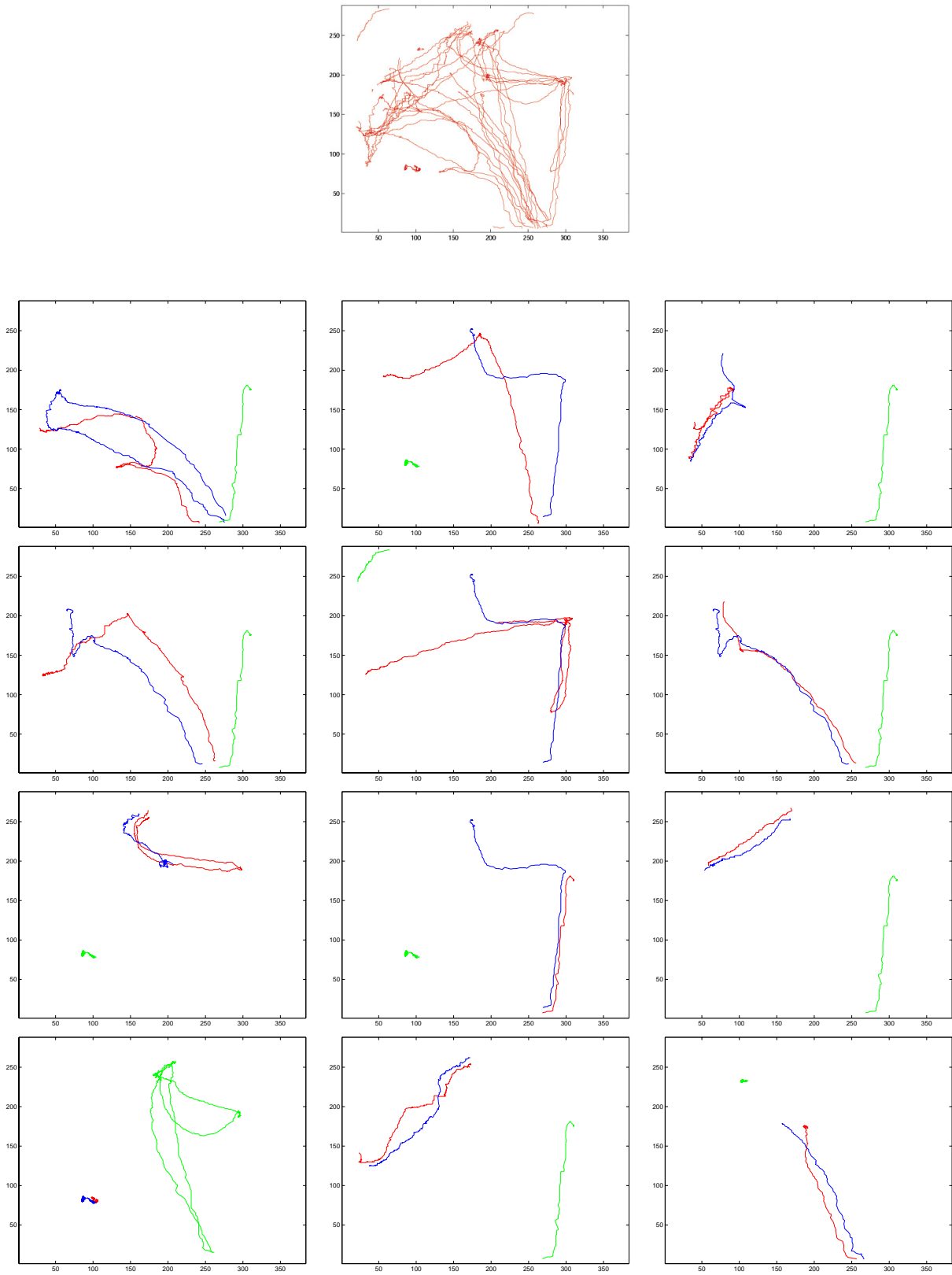


Figure 7. (Top graph) All trajectories mapped together. (Other graphs) Red: given trajectory, blue: most similar trajectory, green: most different trajectory obtained by the coordinate HMM metric (m_8) for the given trajectory.

Robust Visual Tracking from Dynamic Control of Processing

Alban Caporossi, Daniela Hall, Patrick Reignier and James L. Crowley*
PRIMA-GRAVIR, INRIA Rhône-Alpes, Montbonnot, France

Abstract

This paper presents a robust tracking system that employs a supervisory controller to dynamically control the selection of processing modules and the parameters used for processing. This system employs multiple pixel level detection operations to detect and track blobs at video rate. Groups of blobs can be interpreted as related components of objects during an interpretation phase. A central supervisor is used to adapt processing parameters so as to maintain reliable real time tracking. System performance is demonstrated on the PETS 04 data set.

1. Introduction

This paper presents an architecture for robust on-line tracking and interpretation of video streams. The system is based on a real time process managed by a supervisory controller. During each cycle, target blobs are observed and updated using simple pixel level detection processes. Detection procedures are then specified in a number of detection regions to detect new blobs. An evaluation phase is used to assess system performance and to adapt processing so as to maintain both reliability and real time (video rate) processing. An interpretation phase is then run to interpret groups of blobs as more abstract objects. Performance for this system is illustrated using the PETS 04 data set.

The paper starts with an overview of the system architecture. Section 3 describes the underlying principle of the core modules followed by technical details of the implementation. Section 4 describes a method for automatic adaption of the parameters necessary for the tracking system. The flexibility of the architecture is demonstrated in section 5. Section 6 evaluates the performance of this system on the PETS 04 data sets.

2. Architecture

Figure 1 shows the system architecture. The core of the tracking system is composed of a supervisor, a target initial-

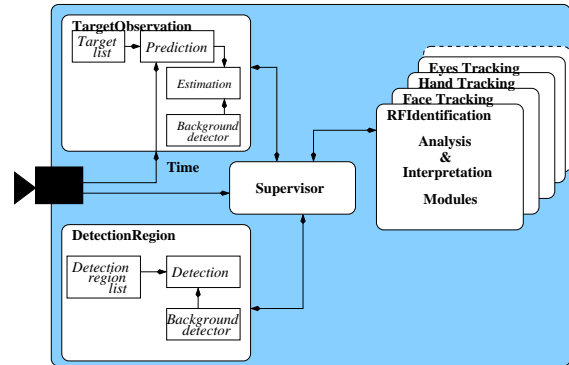


Figure 1. Visual tracking using a central supervisor architecture with core modules enables the flexible plug-in of higher level modules.

isation module (*Detection Region*) and a tracking module (*Target Observation*). These modules are detailed in section 3.

The supervisor acts as a process scheduler, sequentially executing modules in a cyclic process. Each cycle begins by acquiring the current image from an image buffering system (video demon). For each image, targets are tracked and new targets are detected. The supervisor enables a flexible integration of a several modules. During each cycle, for each target, the supervisor can call additional modules for analysis and interpretation as needed. During each cycle, the currently listed image processing operation for each target is applied to the target's region of interest. In this way, the appropriate image processing procedure can be changed and new image processing procedures can be added without changing the existing system architecture. Section 5 shows examples on this flexible architecture by adding modules for head and hand tracking, for eye detection and tracking and for general target identification.

*This research is supported by IST-CAVIAR 2001 37540



Figure 2. Target tracking by background differencing. The central person is tracked using all pixels whereas the two other persons are tracked using every second pixel.

3. The tracking system

In this section, we describe the theoretical aspects and the details on the actual implementation of the core tracking system.

3.1 Energy detection

Currently, targets can be detected by energy measurements based on background subtraction or intensity normalized color histograms. The background subtraction module computes a difference image I_d from the current frame $I = (I_{red}, I_{green}, I_{blue})$ and the background image $B = (B_{red}, B_{green}, B_{blue})$:

$$I_d = \frac{1}{3} (|I_{red} - B_{red}| + |I_{green} - B_{green}| + |I_{blue} - B_{blue}|)$$

The background image B is updated with each frame using a weighted averaging technique, with a strong weight applied to the previous background, and a small weight applied to the current image. This procedure constitutes a simple first order recursive filter along the time axis for each pixel. The background image is only updated for those pixels that do not belong to one of the target ROIs.

$$B_t(i, j) = \begin{cases} \alpha I_t(i, j) + (1 - \alpha) B_{t-1}(i, j), & (i, j) \in \text{bg} \\ B_{t-1}(i, j), & \text{else} \end{cases} \quad (1)$$

Figure 2 shows an example of target tracking by background subtraction. The right image represents the background difference image I_d after processing of three ROI's.

Three targets can be clearly identified. Notice that the center target appears as solid white, while the adjacent targets appear to be "hashed". This is the result of optimization that allows the processing to be applied to every N th pixel. In this example, the two adjacent regions were processed with $N = 2$, while the center target was processed with $N = 1$. N is determined dynamically during each cycle by the process supervisor.

The position and extent of a target are determined by the moments of the detected pixels in the difference image I_d within the ROI. The center of gravity (or first moment) gives the position of a target. The covariance (or second moment) determines the spatial extent, and can be used to determine width, height, and slant of a target. These parameters also provide the target's search region in the next image.

Chrominance information can be used to provide probabilistic detection of targets. The intensity for each RGB color pixel within a ROI is normalized to separate chrominance from luminance.

$$r = \frac{R}{R + G + B}, \quad g = \frac{G}{R + G + B} \quad (2)$$

These color components have the property to be robust to intensity variations [6].

The probability that a pixel takes on a particular color can be represented as a histogram of (r, g) values. The histogram h_T of chrominance values for a target, T , provides an estimate of the probability of a chrominance vector (r, g) given the target $p(r, g|T)$. The histogram of chrominance for all pixels h_{total} gives the global probability $p(r, g)$ of encountering a chrominance among the pixels. The probability of a target is the number of pixels of the target divided by the total number of pixels. Putting these values into Bayes rule shows that an estimate of the probability of the target for each pixel can be obtained by evaluating the ratio of the target histogram divided by the global histogram.

$$p(T|r, g) = \frac{p(r, g|T)p(T)}{p(r, g)} \approx \frac{h_T(r, g)}{h_{total}(r, g)} \quad (3)$$

For each image, a probability map, I_p , can be created by evaluating the ratio of histograms for each pixel in the image. Figure 3 shows an example of face detection using a ratio of chrominance histograms. The bottom image displays the probability map I_p . The probability map is only evaluated within the search region provided by the Kalman filter in order to increase processing speed.

A common problem in both background subtraction and histogram detection are spatial outliers. In order to increase the stability of target localization, we suppress the contribution of outliers using a method proposed by Schwerdt in [5]. With this method, the probability image I_p is multiplied by

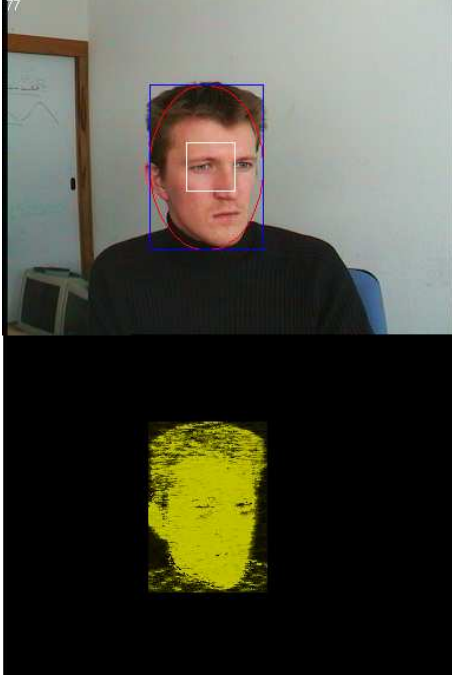


Figure 3. Target detection by normalized color histogram.

a Gaussian weighting function centered at the predicted target position. This corresponds to a filtering by a strong positional prior. The effect is that spatial outliers lose their influence on position and extent as a function of distance from the predicted Gaussian. In order to save computation time, this operation is performed only within the region of interest R of each target. Even for small regions of interest this operation stabilizes the estimated position and extent of targets.

$$I'_p = \begin{cases} I_p * G(\mu, \Sigma), & (i, j) \in R \\ 0, & \text{else} \end{cases} \quad (4)$$

where

$$G(\vec{x}; \mu, \Sigma) = e^{-\frac{1}{2}(\vec{x}-\mu)^T \Sigma^{-1}(\vec{x}-\mu)} \quad (5)$$

The center of gravity $\mu = [\hat{x}_t^-, \hat{y}_t^-]^T$ is the Kalman prediction of the target location. The spatial covariance Σ reflects the size of the target as well as the growing uncertainty about the current target size and location. The same principle can be applied to the background difference I_d .

3.2 Tracking process

The tracking system is a form of Kalman filter [7]. The state vector for each target is composed of position and velocity. The current target state vector \hat{x}_{t-1} is used to make

a new prediction according to :

$$\hat{x}_t^- = \Phi_t \hat{x}_{t-1}, \quad \text{with} \quad \Phi_t = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \quad (6)$$

and Δt the time difference between two iterations.

From the new position measurement z_t , estimation update is carried out.

$$\hat{x}_t = \hat{x}_t^- + K_t(z_t - H_t \hat{x}_t^-) \quad (7)$$

This relation is important for balancing the estimation between measurement and prediction with the Kalman gain K_t . The estimated precision is a diagonal covariance matrix

$$P_t^- = \begin{bmatrix} \hat{\sigma}_{xx}^2 & 0 & 0 & 0 \\ 0 & \hat{\sigma}_{yy}^2 & 0 & 0 \\ 0 & 0 & \hat{\sigma}_{v_{xx}}^2 & 0 \\ 0 & 0 & 0 & \hat{\sigma}_{v_{yy}}^2 \end{bmatrix} \quad (8)$$

and is predicted by:

$$P_t^- = \Phi_{t-1} P_{t-1} \Phi_{t-1}^T + Q_{t-1} \quad (9)$$

where Q_{t-1} is the covariance matrix of the prediction error which represents the growth of the uncertainty in the current target parameters.

3.3 The core modules

The tracking process has been implemented in the ImaLab environment [4]. This environment allows real-time processing of frames extracted from the video stream. The basic tracking system is composed of two modules:

- *TargetObservation* predicts for each target the position in the current frame by a Kalman filter and then computes its real position by background subtraction or color histogram detection.
- *DetectionRegion* detects new targets by analysing the energy (background differencing or color histogram) within several manually defined detection regions.

Figure 1 shows the system architecture. Both core modules can be instantiated to use either background differencing or color histogram. For the PETS 04 experiments, we use tracking based on background subtraction.

3.4 Target initialization module

Detection regions are image regions where new targets can appear. Restricting detection of new targets to such regions allows the system to reduce the overall computing time. As a side effect, the use of detection regions also provides a reduction in the number of spurious false detections

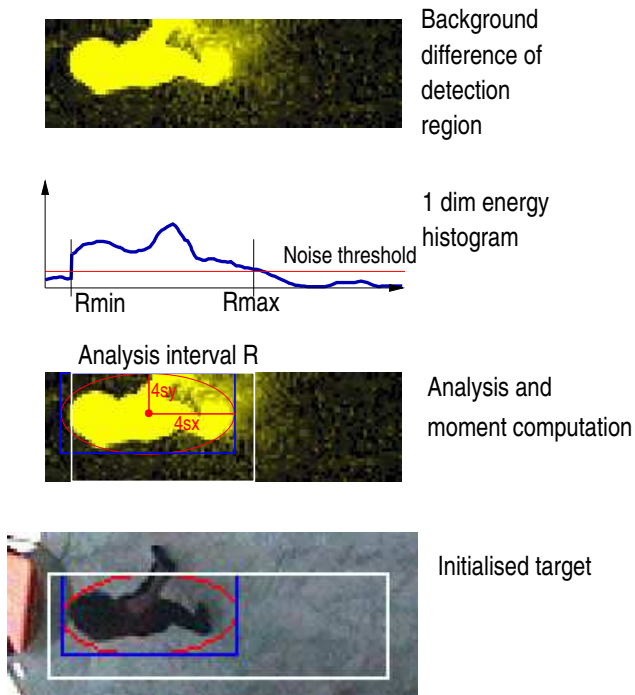


Figure 4. Initialisation of new target.

by avoiding detection in unlikely regions, but targets might be missed when the detection regions are not chosen appropriately.

For each scenario a different set of detection regions is determined. Currently, these regions are selected by hand. An automatic algorithm appears to be relatively easy to imagine. New targets are initialized automatically by analysing the detection regions in each tracking cycle. This analysis is done in two steps. In the first step, the subregion which is occupied by the new target is determined by creating a 1 dimensional histogram along the long axis of the detection region. The limits of the target subregion are characterized by an interval, R_{min} , R_{max} , whose values of the one dimensional histogram are above a noise threshold (see Figure 4). In the second phase, the energy density within the so specified subregion R is computed as

$$e_R = \frac{1}{|R|} \sum_{(i,j) \in R} I_d(i,j) \quad (10)$$

with $|R|$ number of pixels of R . A new target with mean μ_R and covariance Σ_R is initialised when the measured energy density e_R exceeds a threshold. This approach has the advantage, that targets can be detected independently of the size of the detection region.

3.5 Tracking module

The module *TargetObservation* implements the target tracking. The supervisor maintains a list of current targets. Targets of this list are sequentially updated by the supervisor depending on the feedback of the modules. For each target, a new position is predicted by a first order Kalman filter. This prediction determines a search region within which the target is expected to be found. A target is found by applying the specified detection operation to the search region. If the average target detection energy is above a threshold, the target observation vector is updated. This module depends on following parameters:

- Detection energy threshold: this represents the average energy threshold validating the target existence.
- Sensitivity threshold : this parameter thresholds the energy image (I_d in case of background differencing or I_p in case of chrominance detection). If the value is 0, the raw data of the energy image is used.
- Target area threshold: A step size parameter N enables faster processing for large targets by processing only 1 out of N pixels. When the target surface is larger than a threshold, N is increased. This temporary measure will be replaced by a more sophisticated control logic based on computing time. Figure 2 illustrates the use of this parameter.

3.6 Split and merge of targets

In real world video sequences, especially in the domain of video surveillance, it often happens that targets come together, move in the same direction for a while and then separate. It can also occur that close targets occlude each other. In that case only one target is visible at the time, but both targets are still present in the scene. To solve such problems, we use a method that allows merging and splitting of targets. This method enables to keep track of occluded targets and also to model common behavior of a target group. The PETS 04 sequences contain many examples of such group behavior.

A straight forward approach is applied for the detection of target split and merge. Merging of two targets that are within a certain distance from each other is detected by evaluating following equation:

$$c/(a+b) < threshold \quad (11)$$

where c is the distance between the gravity centers of both targets, a and b are the distances between the center of gravity and the boundary of the ellipse defined by the covariance of the respective target(see Figure 5 (left)). In our implementation we use a $threshold = 0.8$.

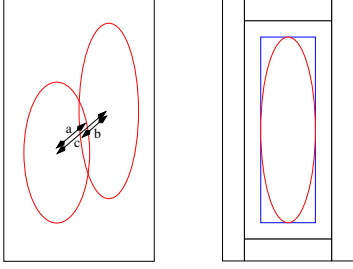


Figure 5. (left) Merging of targets as a function of the target relative position and size. (right) Splitting detectors are defined proportionally to the target size.

Splitting of targets is implemented by placing detection regions around the target as shown in Figure 5 (right). The size and location of the split detection regions are proportional to the target size. Within each split detection region, the average energy is evaluated in the same way as in the target initialisation module. A new target is created if this average energy is greater than the threshold $u = \text{energy density} * \text{split coefficient}$. The parameter *split coefficient* controls the constraints for target splitting.

4. Automatic parameter adaption

Target initialization and tracking by background differencing or histogram detection requires a certain number of parameters, as mentioned in the previous sections (detection energy threshold, sensitivity, density energy threshold, α , split coefficient, area threshold).

In order to preserve the re-usability of the tracking module and guarantee good performance in a wide range of different tracking scenarios, it is crucial to have a good parameter setting at hand. Up to now, parameter adaption is done manually. This is a very tedious job which might need frequent repetition when the scene setup has changed.

In this section we propose a first attempt of a module that automatically finds a good parameter setting. As a first step, we consider the tracker as a classical system with control parameters and noise perturbations (see Figure 6). The system produces an output $y(t)$ that depends on the input $r(t)$, some noise $d(t)$, and a set of parameters that affect the control module K [1].

4.1 Algorithm

First we need to explore the effect of particular parameters on the system. The goal of this step is to identify the important parameters, their relation and eventually discard

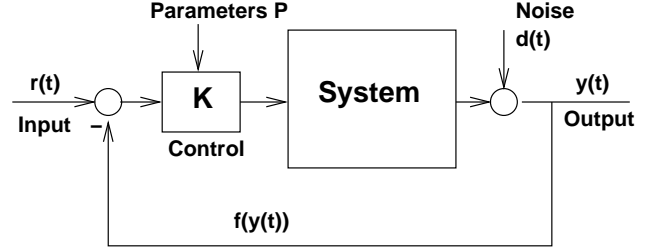


Figure 6. A controlled system

parameters with little effect. For a sequence for which the ground truth $r(t)$ is available we vary the parameters systematically and measure the output of the system, $y_{P_k}(t)$ for a particular parameter setting P_k in the parameter space P . $y_{P_k}(t)$ and $r(t)$ are split in m sections according to m intervals $s_i = [t_{i-1}, t_i], i = 1, \dots, m$.

For each parameter setting P_k and each interval $r(s_i)$ and $y_{P_k}(s_i)$ are known. From these input/output correspondences we can compute the transfer function $f(y_{P_k}(s_i)) = r(s_i)$ by a least squares approximation. The overall error of the transfer function on the sequence is computed as follows:

$$\epsilon = \|r(t) - f(y_{P_k}(t))\| = \sum_{s_i} \|r(s_i) - f(y_{P_k}(s_i))\| \quad (12)$$

For each P_k , we determine the transfer function that minimizes this error. The average error ($\bar{\epsilon} = \epsilon/n$, n number of frames) is used to characterize the performance of the system with the current parameter setting. This is a very coarse approximation, but as we will see, the average error evolves smoothly over the parameter space.

We consider polynomial transfer functions of first and second order (linear and quadratic) of the following form

$$\vec{r}(t_k) = A_0 \vec{y}(t_k) + \vec{b} \quad (13)$$

$$\vec{r}(t_k) = A_2 (\vec{y}(t_k))^2 + A_1 \vec{y}(t_k) + \vec{b} \quad (14)$$

with transfer matrices A_i and offset \vec{b} .

The measurements have either two or four dimensions. In the two dimensional case, the measurements contain the coordinates of the center of gravity of the target. The four dimensional case also contains the height and width of the target bounding box. We could have considered an additional dimension for the target slant, but we discarded this possibility due to the discontinuity of the slant measurement at 180° .

The linear transfer function estimated from the data of the sequences *Walk1.mpeg* and *Walk3.mpeg* produce good results. We observe a transfer matrix A_0 that is close to identity. The quadratic transfer function has a smaller $\bar{\epsilon}$, but the transfer matrix A_2 has very low values and is therefore

not significant. This means that the linear transfer function is a good model for our system.

4.2 Exploration of the parameter space

The average error of the best transfer function evaluated on the entire test sequence is used to characterize the performance of the controlled system. The parameter space can be very high dimensional. Therefore, exploring the entire space can be time consuming. To cope with this problem we assume that some parameters evolve independently from each other. This allows to restrict the search of an optimal parameter value to a low dimensional hyperspace. In the experiment we use following default values for the constant parameters of the hyperspace: detection energy = 10, density = 15, sensitivity = 20, split coefficient = 2.0, $\alpha = 0.001$, area threshold = 1500. We experiment on sequence *Walk1.mpeg* except for figure 7.

Figure 7 shows the surface produced by varying the detection energy threshold and the sensitivity threshold simultaneously. Figure 8 shows the error evolution by varying the split coefficient and the sensitivity. The optimal parameter value is different for each sequence. This means that the parameters are sequence dependent. In all cases the error evolves smoothly. This means that we are dealing with a controlled system and not with a system following chaotic or arbitrary rules.

Figure 9 (left) provides evidence to set $\alpha = 0.1$. Figure 9 (right) shows that the density threshold has no effect on the average error. This parameter is therefore a candidate that needs not be considered for further exploration of the parameter space.

Figure 10 shows the effect of the parameter area threshold. This parameter treats one pixel out of two for targets that are larger than area threshold pixels. This explains the increase of the error for small thresholds and the speed up in processing time. It is interesting to see, that the error increase is very small, less than 4% error increase for a 25% gain in processing time. Our method allows to identify this kind of relations between parameters.

4.3 Summary

We have shown a method to evaluate the performance of a system controlled by a set of parameters. The average error is used to understand the effect of single parameters and parameter pairs. This method allows to verify that our tracking system has a controlled behavior. We identified that the density parameter has no effect on the error performance and it can be removed from the parameter space. The area threshold parameter influences the overall processing time and the average error. With our method, we found that the increase in error is small with respect to the gain in

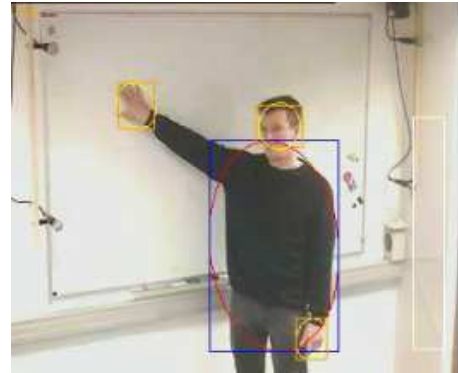


Figure 11. Modules for face and hand observation are plugged into tracking system.

processing time. This is an interesting result which a dynamic control system should take into account. The experiments show that the optimal parameter setting estimated from one sequence scenario must not be optimal for another sequence. This needs to be explored by evaluating more data sequences. Another important point is that the approach requires ground truth labelling. This means that our method can not find the optimal parameters when the ground truth is unknown. Likelihood may be appropriate in some cases to replace the ground truth, but the results will be inferior since the likelihood increases the noise perturbations.

5. Tracking : optional higher level modules

In this section we demonstrate the flexibility of our tracking system. The proposed architecture enables easy plug in of higher level modules which enables the system to solve quite different tasks.

5.1. Face and hand tracking for human computer interaction

Modules for face and hand tracking use color histogram detection. Face and hands are initialised automatically with respect to a body detected by background differencing. This means that the same tracking principle is applied to faces and hands at a higher level. An example is shown in Figure 11.

5.2. Eye detection for head pose estimation

This module detects facial features by evaluating the response to receptive field clusters [2]. The method detects facial features robust to scale, lighting variation, person and

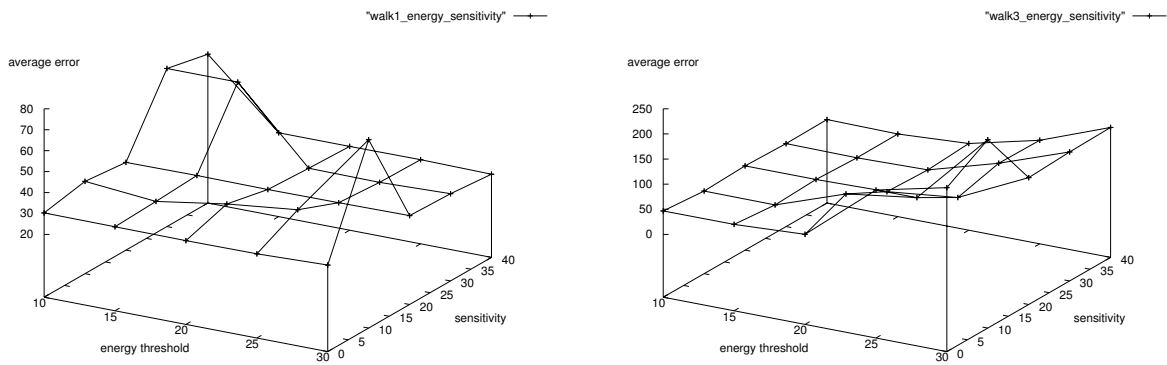


Figure 7. Evolution of the average error over detection energy threshold and sensitivity threshold (sequence Walk1.mpeg (left) and Walk3.mpg (right) and default values for free parameters).

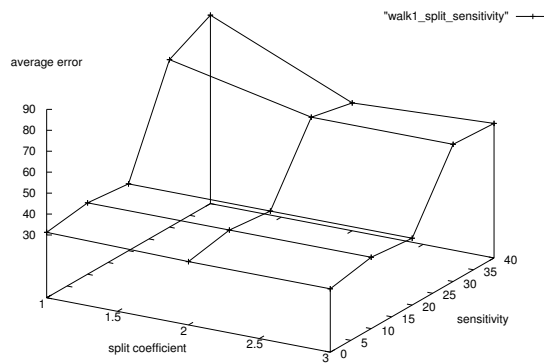


Figure 8. Evolution of the average error over split coefficient and sensitivity threshold.

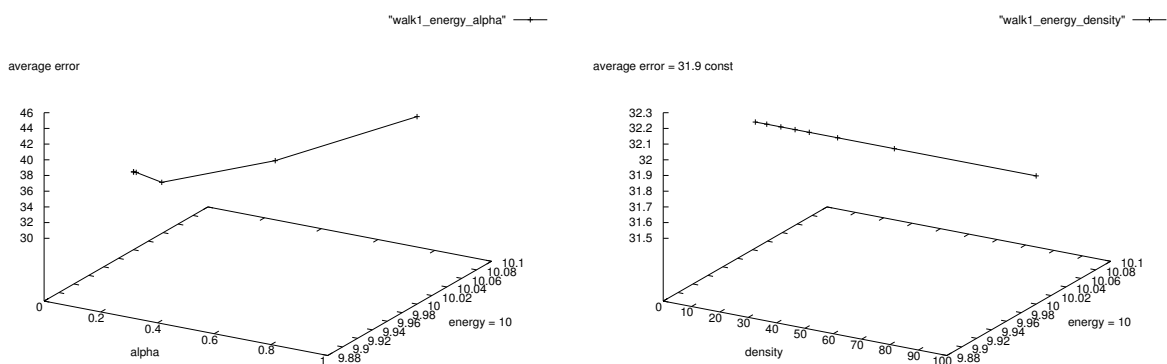


Figure 9. Evolution with varying alpha (left) and varying density (right). We can identify an optimal value for alpha ($\alpha = 0.1$), but the error is constant for all density values.

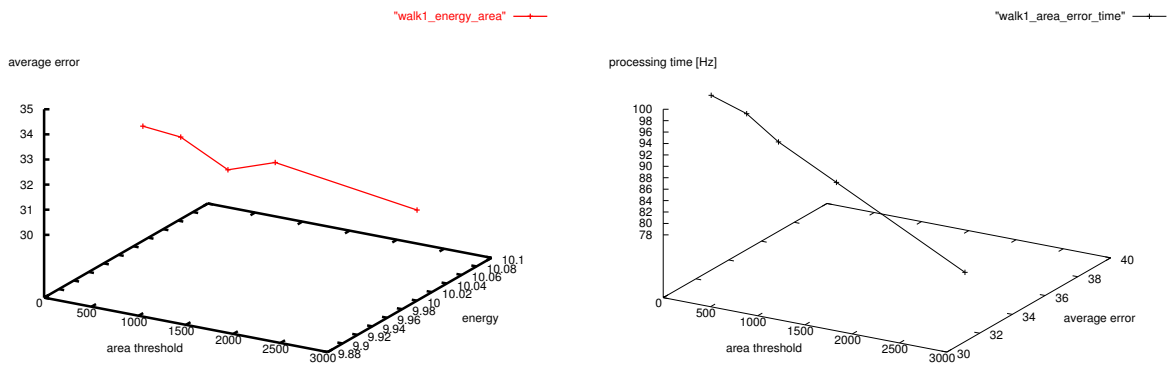


Figure 10. Evolution with varying area threshold (left). The error increases slightly with decreasing area threshold. The area threshold has a significant impact on the processing time (right).



Figure 12. Real-time head pose estimation.

head pose. The tracking system provides the precise face location which allows the combined system to run in real time. Figure 12 shows an example of the eye tracking module.

5.3. Agent identification

The agent identification module provides an association between individual features and tracked targets by background subtraction. Identification of each tracked blob is carried out by elastic matching of labelled graphs where the labels are receptive field responses [2]. The degree of correspondence between the model and the observations extracted from the ROI provided by the tracking system is computed by evaluating a cost function. The cost function is a weighted sum of the spatial similarity and the appearance similarity [3, 8]. Figure 13 shows a successful identity recovery after a target occlusion. The system currently processes 10 frames/s.

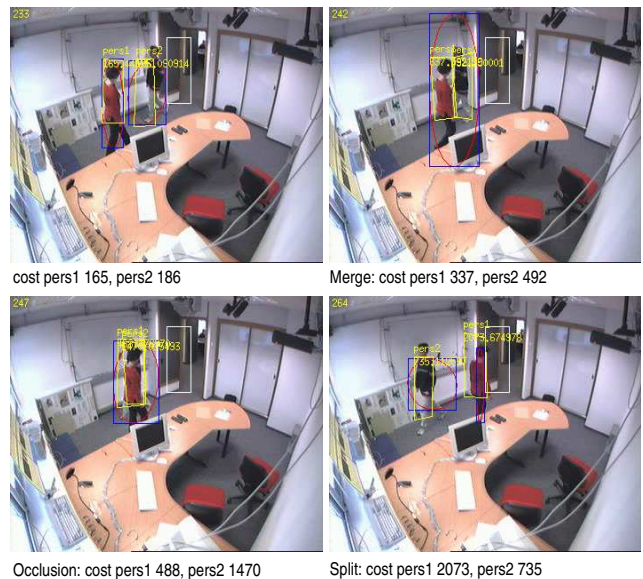


Figure 13. Example of a split and merge event with successful identity recovery.

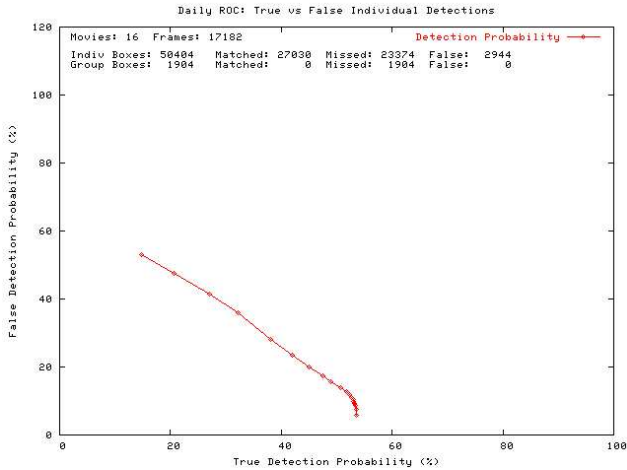


Figure 14. True versus False detections for individuals

6. Tracking performance of the core modules

In order to evaluate the performance of our tracking system, we have tested the core modules on 16 of the PETS 04 sequences (17182 frames containing 50404 targets marked by bounding boxes)¹. In this section we give a brief summary of the tracking results.

Figure 14 shows the receiver operator curve for all 16 sequences. Our system has a low false detection probability of 9.8% and a true detection probability of 53.6%. This translates to a recall of 53.6% (27030 correct positives out of 50404 total positives) and a precision of 90.2% (27030 correct positives out of 29974 detections). The reason for the relatively low recall is the fact that the ground truth labeling takes into account targets that are already present in the scene and targets that pass on the gallery at the first floor. Our tracking system relies on the method of detection region for target initialization. Both type of targets are not detected by our tracking system, because they are not initialized.

The tracking results are evaluated with respect to other parameters such as errors in detected position, size, and orientation, the time lag of entry and exit. The performance of our system with respect to these parameters is summarized in Table 1. Our system performs very well in position detection, orientation estimation and exit time lag. The bounding box produced by the tracking system is significantly smaller than the bounding box of the ground truth. This is due to the fact that the tracking system estimates the bounding box from the covariance of the pixels with high energy whereas

¹The sequences as well as the statistics are available at the CAVIAR home page <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/caviar.htm>

Average error in	average value	maximum value
Position	6 - 7 pixels	13 - 15 pixels
Size	-160% to -240%	-240%
Orientation	$\pm 0.5\%$	$\pm 30\%$
Entry time lag	50 to 80 frames	100 to 160 frames
Exit time lag	1 frame	1 frame

Table 1. Evaluation of the tracking results with respect to measurement precision.

a human draw a bounding box that includes all pixels that belong to the target. The tracking system can produce a similar output by computing the connected components of the energy image. This is a costly operation. In the case where the connected components bounding box is used for position computation, the position become more unstable. For this reason we decided to use the first and second moments of energy pixels for target specification. The entry time lag is a problem related to the detection region. A human observer marks a new target as soon as it occurs. The detection region requires that the observed energy is above the energy density threshold.

7. Conclusion

We have presented an architecture for a tracking system that consists of a central supervisor, a tracking module based on background subtraction or color histogram detection combined with Kalman filtering and an automatic target initialization module restricted to detection regions. These three modules form the core system. The central supervisor architecture has the advantage that additional modules can be plugged in very easily. New tracking systems can be created in this way that can solve different tasks.

The tracking system depends on a number of parameters that influence the performance of the system. Therefore, finding a good parameter setting for a particular scenario is essential. We have proposed to consider the tracking system as a classical controlled system and identified a method to evaluate the quality of a particular parameter setting. The preliminary experiments show that small variations of the parameters produce smooth changes of the average error function. Using this behavior, we can improve the performance of our tracking system by finding a good parameter setting using gradient descend in the parameter space. Unfortunately, the experiments on the automatic parameter adaption are preliminary and could not yet be integrated in the performance evaluation of the system.

References

- [1] P. de Larminat. *Automatique commande des systèmes lineaires*. Hermes Science Publications, 2nd edition, 1996.
- [2] D. Hall and J.L. Crowley. Détection du visage par caractéristiques génériques calculées à partir des images de luminance. In *Congrès Francophone de Reconnaissance des Formes et Intelligence Artificielle*, pages 1365–1373, Toulouse, France, 2004.
- [3] M. Lades, J.C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Mahlsburg, R.P. Würz, and W. Konen. Distortion invariant object recognition in the dynamic link architecture. *Transactions on Computers*, 42(3):300–311, March 1993.
- [4] A. Lux. The imalab method for vision systems. In *International Conference on Vision Systems*, pages 319–327, Graz, Austria, April 2003.
- [5] K. Schwerdt and J.L. Crowley. Robust face tracking using color. In *International Conference on Automatic Face and Gesture Recognition*, pages 90–95, Grenoble, France, March 2000.
- [6] M.J. Swain and D.H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
- [7] G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, 2004.
- [8] L. Wiskott, J.M. Fellous, N. Krüger, and C. von der Mahlsburg. *Face Recognition by Elastic Bunch Graph Matching*, chapter 11, pages 355–396. *Intelligent Biometric Techniques in Fingerprint and Face Recognition*. CRC Press, 1999.

Automatic Tracking and Labeling of Human Activities in a Video Sequence

Fengjun Lv Jinman Kang Ram Nevatia Isaac Cohen Gérard Medioni
University of Southern California
Institute for Robotics and Intelligent Systems
Los Angeles, CA 90089-0273
{flv|jinmanka|nevatia|icohen|medioni}@iris.usc.edu

Abstract

This paper presents a novel approach for tracking multiple objects and a statistical learning approach for detection of human activities in a video sequence. For the tracking, a 2D rigid transformation invariant appearance model combining color and edge information of the detected blob is proposed. For the activity detection, each activity label is regarded as a hypothesis. Given some labeled sequences, a group of features are first extracted from motion trajectories of each detected object and the likelihood of each feature under that hypothesis is calculated. A dynamic programming-based training algorithm is applied to get an optimal classifier for each feature. Then it selects the classifiers with the most discriminative power and combines them to form a stronger classifier. This algorithm complies with Neyman-Pearson criterion so that it is guaranteed to achieve a specified detection rate as well as a minimized false alarm rate. Results on PET S'04 dataset¹ show the effectiveness of the proposed algorithm.

1 Introduction and Related Work

Developing an automatic activity recognition system is becoming of increasing interest to many researchers in recent years. Most proposed systems try to interpret human activities based on a well-structured interaction model (*HMM*[11] and its extension such as *Semi-HMM*[8], or recently, Propagation Network[13]). Due to the uncertain nature of the activity instances, such systems can only deal with either very simple activities or complex ones but in a tightly constrained environment.

In many cases, however, instead of telling people what has happened and how did that happen, it is sufficient for a computer system to alert people when an activity

is (probably) happening. This can have many applications such as helping people with the tedious monitoring work in video surveillance and content based video retrieval. The monitored activities have one characteristic in common that they have a relatively short duration and thus the presence of that activity can be inferred with no or only little contextual information. We term these *primitive* activities and they are the focus of this paper.

Detecting (or labeling) a primitive event in nature is a classification (YES or NO) problem and this fits very well in a statistical Hypothesis-Testing approach. Unlike model-based methods, in which many assumptions need to be made about the underlying activity models, all that we need are some training data which have been labeled manually or by other means. In the training phase, after a set of features are extracted, the *prior* probability of each hypothesis(activity label) and the likelihood of each feature under that hypothesis are calculated. Then in the testing phase, given the values of the same set of features, the activity is labeled as YES or NO based on its *posterior* probability.

Bayesian Network is a suitable tool for this purpose [1] [9]. The use of Bayesian networks in these approaches differs in features that they used as evidences and structures of the networks, which are usually derived by heuristic knowledge. Our approach is different in that instead of using a pre-defined fixed feature set and network structure, we learn the optimal feature set dynamically from the training data and compute the classifiers based on the joint distribution of the feature set.

We present an algorithm for optimal classifier training for each feature. The algorithm can minimize false alarm rate while maintaining certain detection rate by mapping the problem to the classical 0-1 *Knapsack* problem which has a well-known dynamic programming-based optimal solution. The algorithm works in an iterative manner such that at each time it selects features with the

¹Data come from the EC Funded CAVIAR project/IST 2001 37540 at <http://www.dai.ed.ac.uk/homes/rbf/CAVIAR/>.

most discriminative power and combines them to form a stronger classifier until the false alarm rate falls below a specified value or the upper limit of number of iterations has been reached.

Since our system is trained and tested primarily on the *PETS'04* dataset, before describing the details of the algorithm, some insights into the data may be worth pointing out. In this dataset, 28 sequences were filmed with a wide angle camera lens in the entrance lobby of a building, of which 14 are provided with ground-truth data. The dataset contains 6 staged scenarios including people walking, browsing, resting, leaving a bag, group meeting and fighting. There are in total 42 labels of activities including individual activities such as walking and browsing and group activities such as fighting and meeting. Fig.1 lists the name and meaning of each label (Labels of group box are marked with a “g_” in order to distinguish them from labels of individual box). The dataset provides an ideal test-bed for activity labeling algorithms in terms of the variety as well as the complexity of the contents of the video sequences.

individual box			group boxes		
category	label	meaning	category	label	meaning
state flag list (7)	ap	appear	state flag list (5)	g_ap	appear
	di	disappear		g_d	disappear
	o	occluded		g_i	inactive
	in	inactive		g_ac	active
	ac	active		g_mo	movement
	wk	walking	g_f	fighters	
	r	running	g_me	meters	
scenario flag list(14)	f	fighter	g_w	walkers	
	br	browser	g_gf	fighting	
	lv	left victim	g_gmo	moving	
	lg	leaving group	g_gme	meeting	
	wr	walker	g_s	split up	
	lo	left object	g_gi	inactive	
	m	moving	g_glv	leaving victim	
	is	inactive	g_glo	leaving object	
	bsi	browsing	g_fsc	fighting	
	bsc	browsing	g_mes	meeting	
	im	immobile	g_ls	leaving object	
	wg	walking	g_fst	fight start	
	dd	drop down	g_fe	fight end	
	pi	immobile	g_fv	left victim	
scenario flag list(16)			role (3)		
			situation (7)		
			scenar -io(3)		
			event (3)		

Figure 1: Name and meaning of each label in *PETS'04* dataset

2 Detection and Description of Moving Objects

Object detection and tracking are fundamental elements for an activity recognition system. A persistent appearance model is usually needed in order to track objects robustly in a scene containing large dynamics (occlusions, disappearing-reappearing of moving objects, etc.). An object appearance model is represented

by a set of distinctive features such as color, shape or texture. In [2], active contour is used as the shape-based appearance model. The active contour based method, however, usually requires initializing the contour manually and only handles small non-rigid motion. Various color-based methods have been proposed, of which many use only one color histogram for each object. In [12], an appearance model based on temporal color is used but this approach is not invariant to arbitrary rigid motion. Multiple color models and their relative localization have been considered for an efficient use of the color in object tracking. In [6], a multiple color model was proposed for human detection, but it required a segmentation of detected blob into the head, torso, and legs.

Appearance changes are expected when objects are moving. For example, the limb motion of a walking human will create localized shape variations and self-occlusions. Therefore, object appearance models have to be continuous in the sense that a small localized change of the object color and shape should create a small localized variation in its signature. The object description should also be invariant to 2D rigid transformation and scale change in order to accommodate change of perspective. The appearance descriptor we proposed here complies with these requirements, as described in Sec.3.

The color distribution model is obtained by mapping the blob into multiple polar representations. Several shape or color distribution models using a polar representation have been proposed [15] [10] [3]. The approach in [15] focuses on the object’s shape description (edge) and is limited to represent local shape properties. In [10] the proposed model measures color distribution using a similar polar representation, but focuses on characterizing a global appearance signature of the object. Since this model is not 2-D rotation-invariant, we combine it with the shape description model proposed in [3] to guarantee the invariance to 2-D rigid transformation and scale change.

Given a detected moving blob, the smallest circle (C_R) containing the blob, is the region of our interest. A set of control points P_i are uniformly sampled along C_R . For each P_i , a set of concentric circles of various radii are used to define the radial bins of the appearance model. Inside each bin, a Gaussian distribution (for each channel of R,G,B) is computed for modeling the color properties of pixels falling into that bin. Therefore, for a given control point P_i we have a 1-D distribution $\gamma_i(P_i)$. The normalized combination of $\gamma_i(P_i)$ defines the color model of the detected blob: $\Lambda = \sum \gamma_i(P_i)$.

The shape description ($E(r)$) of each bin is obtained similarly by counting the number of edge pixels falling into that bin. Then $E(r)$ is normalized as follows:

$$E(r) = \sum_i E_r(P_i) / \max_r \left(\sum_i E_r(P_i) \right) \quad (1)$$

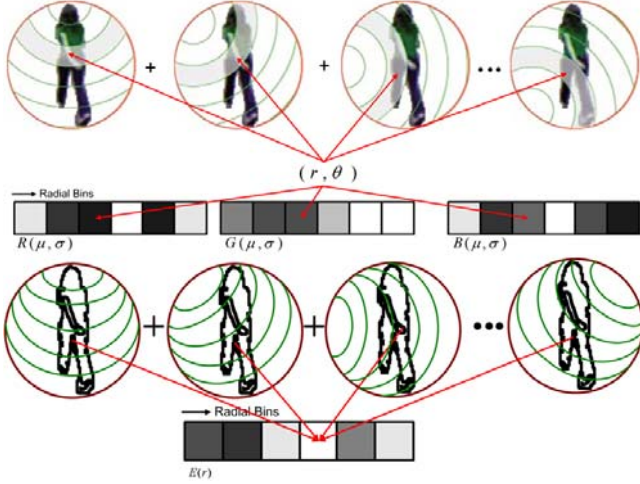


Figure 2: Computation of the color and shape based appearance model of detected moving blobs

An illustration of the definition of the appearance model is shown in Fig.2 where we sampled the reference circle with 8 control points. This model is inherently translation invariant. Rotation invariance is obtained by taking a larger number of control points along the reference circle. Finally, normalizing C_R to unit circle guarantees scale invariance.

3 Tracking Using Invariant Appearance Probability Model

The appearance probability model is defined as a similarity measurement among detected blobs in successive frames. In this paper, we employ Kullback-Leibler distance for measuring the similarity of the computed appearance models.

The similarity measurement of the color model is derived in terms of mean and variance of the Gaussians in each bin and given by Eq.2.

$$Dist_{KL_Color} = \sum_{N_{rgb}} \frac{1}{2} \left\{ (\mu_t - \mu_{t+1})^2 \cdot \left(\frac{1}{\sigma_{t+1}^2} + \frac{1}{\sigma_t^2} \right) + \frac{\sigma_t^2}{\sigma_{t+1}^2} + \frac{\sigma_{t+1}^2}{\sigma_t^2} \right\} \quad (2)$$

The similarity measurement of the shape model is given by Eq.3.

$$Dist_{KL_Shape} = \frac{1}{2} \sum (E(r)_t - E(r)_{t+1}) \log \frac{E(r)_t}{E(r)_{t+1}} \quad (3)$$

These two are combined by Eq.4.

$$P_{App_KL} = 1 / \sqrt{Dist_{KL_Color}^2 + Dist_{KL_Shape}^2} \quad (4)$$

Fig.3 illustrates the capability of the proposed tracking approach. As we can see, the method allows to track moving objects continuously with occlusion and large non-rigid deformation.

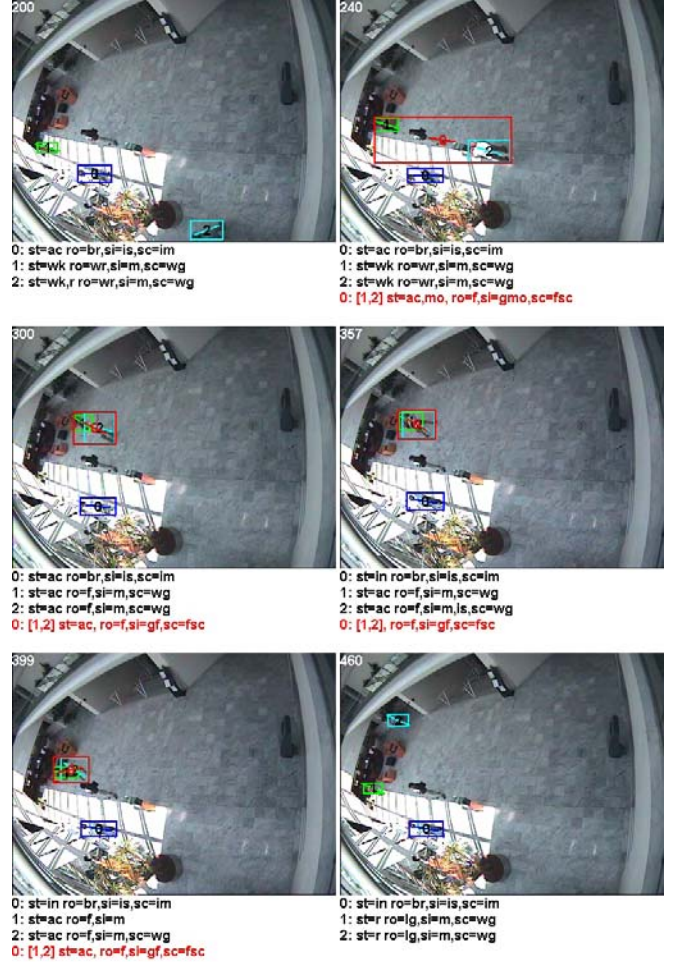


Figure 3: Some tracking as well as labeling results of sequence *Fight_RunAway2*

4 Activity Labeling Algorithm

For a statistical learning-based training algorithm, two questions need to be addressed: (1) What kind of error criterion do we use? (2) What kind of features (evidences) are suitable for training classifiers? These two questions are answered in the following two sub sections, followed by a detailed description of the training and labeling algorithm.

4.1 The Neyman-Pearson Criterion

Bayesian classifier is a powerful, yet simple tool to infer *posterior* probability of H (hypothesis), based on the *prior* probability of H and the likelihood of E (evidence) under H . The general form of Bayesian rule is shown in Eq.5.

$$P(H|E) = \frac{P(E|H)P(H)}{P(E|H)P(H)+P(E|\bar{H})P(\bar{H})}, \quad P(\bar{H}|E) = 1 - P(H|E) \quad (5)$$

Based on Eq.5, a decision for one-class classification problem can be made as: If $P(H|E) > P(\bar{H}|E)$, then accept; otherwise, reject.

Two types of error can occur: One is false negative (or missing detection) and the other is false positive (or false alarm). If the decision is probabilistic rather than deterministic, the difference between the real and computed *posterior* probability should also be considered as a part of false negative error. As in our training dataset, the probability of each label is either 0 or 1 (thus easier to label ground-truth data by human), we only consider the deterministic case here. If the number of positive samples, negative samples, false negative and false positive are cnt^+ , cnt^- , cnt_{fn} , cnt_{fp} , respectively, the error rates of the two types are given by:

$$error_{fn} = \frac{cnt_{fn}}{cnt^+} = 1 - detection_rate, \quad error_{fp} = \frac{cnt_{fp}}{cnt^-} \quad (6)$$

Eq.5 treats these two types of error equally so that the probability of overall error in terms of $\frac{cnt_{fn}+cnt_{fp}}{cnt^++cnt^-}$ is minimized [7]. In some cases, different utility (cost, risk, etc.) functions are assigned to each type of error to accommodate the difference in impact of the errors. Here we train the optimal classifier according to the *Neyman-Pearson* criterion [14], which imposes constraint on one of the errors and optimize the other. Since we care more about the detection rate, we specify $error_{fn}$ and try to minimize $error_{fp}$.

4.2 Features

Feature design is of key importance for a classification problem. A good feature for classification purpose should have the following characteristics: (1) Capable of discriminating different classes; (2) Easy to acquire; (3) Intuitive so that human knowledge can be beneficial in feature design. Despite the size of the initial feature pool being large, the final classifier usually contains only a small number of critical features with the most discriminative power. In our system, a pool of as many as 50 features is used. Each feature is a scalar.

Based on our observation of *PET S'04* dataset, the 42 activity labels can be classified into the following three types: (I) Activities of a single person; (II) Activities of a group of persons and (III) Roles that an individual played in a group. We design different features for different category of activities according to the interactions among humans in the activities.

For activities of type *I*, the following 13 features are used. Most of them are (or can be derived directly from) motion trajectory of that person.

$$\begin{aligned} (1)x_{IB}^t, (2)y_{IB}^t, (3)o_{IB}^t, (4)w_{IB}^t, (5)h_{IB}^t, (6)size_{IB}^t &= w_{IB}^t \cdot h_{IB}^t \\ (7)speed_{IB}^t &= \sqrt{(x_{IB}^t - x_{IB}^{t-1})^2 + (y_{IB}^t - y_{IB}^{t-1})^2} \\ (8)direction_{IB}^t &= \cos^{-1}\left(\frac{x_{IB}^t - x_{IB}^{t-1}}{speed_{IB}^t}\right) \\ (9)change_w_{IB}^t &= \frac{w_{IB}^t - w_{IB}^{t-1}}{w_{IB}^{t-1}} \quad (10)change_h_{IB}^t = \frac{h_{IB}^t - h_{IB}^{t-1}}{h_{IB}^{t-1}} \\ (11)change_size_{IB}^t &= \frac{size_{IB}^t - size_{IB}^{t-1}}{size_{IB}^{t-1}} \quad (12)duration_{IB}^t, (13)end_{IB}^t \end{aligned}$$

Note that (1) Subscript IB stands for Individual Box and superscript t and $t-1$ stand for current frame and previous frame; (2) x, y, o, w and h denote, respectively, column and row of center, main axis orientation, width and height of the individual box, as described in the data format of *PET S'04*; (3) *duration* is the number of frames since the object appeared. If the object disappears for a while and appears again, the value is reset to one; (4) *end* is a boolean value. It is *true* if the object disappeared at the previous frame.

For activities of type *II*, the following 26 features are used. They contain information of the group box as well as some statistics of its members.

$$\begin{aligned} (1)x_{GB}^t, (2)y_{GB}^t, (3)o_{GB}^t, (4)w_{GB}^t, (5)h_{GB}^t, (6)size_{GB}^t \\ (7)change_w_{GB}^t, (8)change_h_{GB}^t, (9)change_size_{GB}^t \\ (10)duration_{GB}^t, (11)end_{GB}^t, (12)cnt_{GB}^t \\ (13)o_{\mu}^t, (14)o_{\sigma}^t, (15)dist_{\mu}^t, (16)dist_{\sigma}^t, (17)dist_{max}^t, (18)dist_{max}^t \\ (19)speed_{\mu}^t, (20)speed_{\sigma}^t, (21)direction_{\mu}^t, (22)direction_{\sigma}^t \\ (23)size_{\mu}^t, (24)size_{\sigma}^t, (25)size_{max}^t, (26)size_{min}^t \end{aligned}$$

Note that (1) Subscript GB stands for Group Box; (2) cnt is the number of individual boxes in the group; (3) Subscript μ and σ , denoted mean and standard deviation of the values of each member; (4) $dist_{max}$ is the maximal distance from center of group to center of each member; (5) $dist2_{max}$ is the maximal distance between any two members in the group.

Group detection itself can be regarded as a labeling problem. The feature set contains feature (12,13,...,26) listed above. The problem is that we don't have negative samples. Here, if a frame contains at least two individual boxes and no group box, we consider it as a negative sample and the features are computed based on the two

closest individual boxes.

Lastly, for activities of type *III*, features include the 13 features used for type *I* activities as well as the following 11 new features, which represent deviation of that person from the rest of the group:

$$\begin{aligned}
(1)dist_{IG}^t, (2)\frac{dist_{IG}^t - dist_{\mu}^t}{dist_{\sigma}^t}, (3)dist_{IG}^t, (4)o_{IB}^t - o_{\mu}^t, (5)\frac{o_{IB}^t - o_{\mu}^t}{o_{\sigma}^t} \\
(6)speed_{IB}^t - speed_{\mu}^t, (7)\frac{speed_{IB}^t - speed_{\mu}^t}{speed_{\sigma}^t} \\
(8)direction_{IB}^t - direction_{\mu}^t, (9)\frac{direction_{IB}^t - direction_{\mu}^t}{direction_{\sigma}^t} \\
(10)size_{IB}^t - size_{\mu}^t, (11)\frac{size_{IB}^t - size_{\mu}^t}{size_{\sigma}^t}
\end{aligned}$$

Note that (1) $dist_{IG}^t$ is the distance from the individual to the center of group; (2) $dist_{IG}^t$ is the maximal distance between the individual and any other member in the group.

4.3 The Training and Labeling Algorithm

For each activity, given the training data, the task of the training algorithm is to select the best features and train the optimal classifier on that basis. If a one-dimensional classifier based on a single feature can not separate all the positive and negative samples sufficiently well, features need to be combined to form a stronger multi-dimensional classifier.

Let us begin with the training of 1-*D* classifiers. Assume *A* is the activity we are interested and (F_1, F_2, \dots, F_n) is the feature set for *A*. The training data contain cnt^+ positive samples (frames with *A*) and cnt^- negative samples (frames without *A*). If *D*, the detection rate is specified, the upper bound of missing detection is $max_{fp} = (1 - D) \cdot cnt^+$.

For each feature F_i , $i = 1 \dots n$, we first compute feature values of both positive and negative samples $(f_{i,1}^+, f_{i,2}^+, \dots, f_{i,cnt^+}^+)$, $(f_{i,1}^-, f_{i,2}^-, \dots, f_{i,cnt^-}^-)$. We use histogram of $(f_{i,1}^+, f_{i,2}^+, \dots, f_{i,cnt^+}^+)$ to approximate the likelihood of $P(F_i|A)$. Usually, the more samples in the training data, the closer the approximation to the real distribution. Fig.4 shows the histograms of feature x_{IB}^t and y_{IB}^t computed from positive samples of the activity *bsc*. As we can see, bins with large number of polls correspond to the real browsing areas (e.g. bulletin and map) marked with a number.

Unfortunately, if we draw the histogram of $(f_{i,1}^-, f_{i,2}^-, \dots, f_{i,cnt^-}^-)$, we will probably find that the number of negative samples falling into the same bins is also large. Ideally, if one bin contains only positive samples, we can confidently mark it as *accept*. Or *vice versa*, we reject without hesitation those bins with only negative samples.

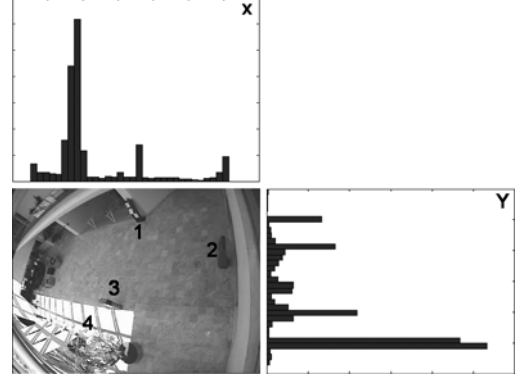


Figure 4: Histograms of feature x_{IB}^t and y_{IB}^t of activity *bsc*

For a bin with more than max_{fp} positive samples, we have no choice but to accept it. The bins left are of our interest.

The goal is obvious. We have to find out and accept the bins with most positive samples and least negative samples. In other words, we want to reject the bins with most negative samples and least positive samples, under the constraint that the total number of rejected positive samples can not exceed max_{fp} . This can be perfectly mapped to the classical 0-1 *Knapsack* problem [4].

The 0-1 *Knapsack* problem is stated as: “A thief robbing a store finds n items; the i th item is worth v_i dollars and weighs w_i pounds, where v_i and w_i are integers. He wants to take as valuable a load as possible, but he can carry at most W pounds in his knapsack for some integer W . Which items should he take?” The problem has an optimal solution based on the following dynamic programming approach: $Knapsack(i, w) = \max(Knapsack(i+1, w), v_i + Knapsack(i+1, w - w_i))$. The computational complexity of this algorithm in worst case is $O(W * n)$.

Here, the i th bin is mapped to the i th item. The count of positive and negative samples in the bin are mapped to w_i and v_i and max_{fp} is mapped to W . To reject the bin is equivalent to take the item.

The partition of histogram plays a critical role here. It is natural to choose bins with equal width. The problem is, how many bins should we use? A coarse histogram can not eliminate negative samples sufficiently: In the extreme case where there is only one bin, all samples are accepted. On the other hand, if the histogram is too fine, the over-fitting problem (*i.e.* prediction is too sample-dependent) becomes severe. Alternatively, a clustering-based histogram partition can do a better job in that the

result naturally reflects the distribution of the samples.

We choose *ISODATA* algorithm over *K-Means* because *ISODATA* allows for different number of clusters while the *K-Means* assumes that the number of clusters is known *a priori*. Initially, a fine histogram is built. The list of tuples (bin_{no} , $count_{samples\ in\ the\ bin}$) (if the bin is not empty) is the input of *ISODATA*. Then *ISODATA* can automatically determine the optimal number of bins by splitting and merging them. Another advantage of *ISODATA* is that if the decision is made based on a threshold, for example, “A group is formed when people are close enough to each other”, we simply adjust K , the number of bins desired, which is a parameter used in *ISODATA*, to two. For details of *ISODATA*, please refer to [5].

The final histogram is partitioned according to the output of the clustering algorithm. Please be aware that so far we are talking about histogram partition for the positive samples. To make this partition applicable to the negative samples as well, two special bins need to be included for values out of range. Furthermore, another special bin should be added for the samples (positive or negative) with an undefined value, *e.g.*, $direction_{IB}^t$ when $speed_{IB}^t$ is zero.

After we partition the histogram and mark each bin either as “accept” or “reject”, we can get the overall false positive and false negative rate of feature F_i for activity A . This is applied to all F_i , $i = 1 \dots n$. If F_{best_1} turns out to have the lowest false positive rate (False negative rate of each feature is almost the same, which is equal to or slightly smaller than the specified value. In case that two features have the same false positive rate, we choose the one with a smaller false negative rate), we select F_{best_1} as the best 1- D feature for A .

If F_{best_1} alone performs poorly in distinguishing positive samples from negative samples, we combine F_{best_1} with each of the rest features: Assume the current feature is F_j ($j = 1, 2, \dots, best_1-1, best_1+1, \dots, n$), we use the previously computed partitions of F_{best_1} and F_j to build 2- D histograms (joint distribution) of positive and negative samples. Once again, the 0-1 *Knapsack* algorithm is used to search for the optimal classifier for (F_{best_1}, F_j) .

Similarly, we select the pair with the lowest false positive rate as the best 2- D classifier. Assume the pair is (F_{best_1}, F_{best_2}) . If the result is still not good enough, we combine (F_{best_1}, F_{best_2}) with each of the rest features. So on and so forth, until either the false alarm rate is lower than a threshold, or the upper limit of number of iterations has been reached.

This greedy approach is not globally optimal. As a matter of fact, finding the optimal set of features needs an exhaustive search in the whole feature space and thus is not feasible due to the large dimensionality of the feature space.

The complexity (mostly on constructing joint histogram and recursive calling of *Knapsack*) of this algorithm grows exponentially with the number of features added to the classifier. But better data structure (using hash table instead of array) can reduce it to be polynomially bounded in the number of training samples. Experiments show that up to 3 features with at most 32 bins each can produce satisfactory results.

After the classifier for the activity is learned, the labeling process becomes straightforward: Step1. The values of the same set of features are computed; Step2. Find the bin number in the 1- D histogram of each feature; Step3. Compute the bin number in the joint histogram; Step4. If the bin marked with “accept”, then accept it; otherwise, reject it.

5 Experimental Results

We conducted experiments as follows: (1) In exp.1, all 14 ground-truth sequences were used for training and testing. The purpose is to find out if the result feature set is consistent with our perception and also to get an estimate of the computational cost; (2) In exp.2, in order to test the algorithm’s capability of making correct predictions on unknown data, we used 75% of ground-truth data for training and the rest 25% for testing; (3) In exp.3, we changed the ratio to 50%-50% to investigate the robustness of the algorithm in terms of the quantity of training data; (4) In exp.4, we adjusted the detection rate to obtain the ROC curve; (5) In exp.5, we tested our system on an unknown (*i.e.* without ground-truth) sequence *Fight_RunAway2*. Note that due to the limited time, we didn’t test our tracking algorithm in the first four experiments. Instead, we used the ground-truth tracking data (of individual box only) and focused on evaluating our labeling algorithm.

5.1 Exp.1: Training and testing on same data

The results of Exp.1 are shown in Fig.5. Eight columns of each row are the label, best feature set, cnt^+ , cnt^- , $error_{fn}$, $error_{fp}$, training time and testing time (in second), respectively. The group detection is the last row of the second category. Here we used 95% as the detection rate for training, thus $error_{fn} \leq 5\%$.

As we can see, the false positive rates of most labels are quite low (less than 3%) and the learned best fea-

ture sets are understandable. Take label *ap* for example, *duration* and *change_size* mean *ap* occurs mostly at the first several frames and the size of the bounding box changes significantly during this period of time. The position features (*x* and *y*) appear in many results, which is not surprising because all activities in the training data happened within a very limited space and due to the apparent perspective effect, many other features (*e.g.* speed, direction, *etc.*) are not independent of position. In case that we do believe the best feature set contains feature that contradicts our common sense, which usually implies an over-fitting problem, we can intentionally remove that feature before training and this is how human knowledge helps in feature design. For detection of group activities, the results confirm our hypothesis that the statistics of group members is playing an important role.

The results also show the efficiency of our algorithm. For a dataset with more than 27,000 samples in total, the training of each label takes less than 5 seconds and labeling less than 0.15 second on a desktop PC with dual P4 2.4GHz CPUs. The labeling process is significantly faster than real-time.

5.2 Exp.2&3: Training and testing on different data

Since each ground-truth sequence has different amount of labels (Some labels appear in only one sequence), we split the dataset based on individual boxes instead of sequences, *i.e.*, for each label, we mixed all individual boxes containing the label and randomly chose 75% for training and the rest 25% for testing. In order to get an unbiased result, we run Exp.2 and 3 five times and the mean (μ) and standard deviation (σ) of $error_{fn}$ and $error_{fp}$ are shown in Fig.6. We didn't show the best feature sets because they depend on the training data and may change each time, but we did find that the best feature sets are generally consistent with those of Exp.1. Again, we used 95% as the detection rate for training.

The overall results of both experiments are good. The average of ($error_{fn}, error_{fp}$) of all labels (We did not count those marked with *n/a* because they appeared only once or didn't appear at all in our dataset.) are $(5.14\%, 2.09\%)_{exp2}$ and $(6.40\%, 2.35\%)_{exp3}$, compared with $(2.89\%, 1.07\%)$ in Exp.1. The fact that Exp.2 performed better than Exp.3 confirms the common characteristic of learning-based algorithms: The more training data there are, (usually) the better testing result it can have. Despite this fact, the result of Exp.3 does not degenerate much, which indicates that the algorithm is robust regardless the amount of available training data. The

label	features	+	-	fn(%)	fp(%)	t_t(s)	t_l(s)
ap	duration,y,change_size	251	27143	4.781	0.309	2.375	0.11
di	y,size,end	50	27344	4	0.004	2.375	0.109
o	w,y,h	969	26425	0.929	0.091	2.391	0.125
in	y,h,duration	5945	21449	4.996	0.117	2.484	0.125
ac	y,x,duration	9092	18302	4.652	0.311	2.657	0.125
wk	y,x,o	11945	15449	4.311	1.819	2.734	0.125
r	duration,y,speed	357	27037	3.641	0.481	2.328	0.11
br	y,x,h	7793	19601	0.565	0.107	2.36	0.125
wr	y,h,o	16591	10803	1.947	0.111	2.875	0.109
m	y,h,duration	15060	12334	2.052	0.965	2.531	0.11
is	x,y,change_size	8700	18694	5	2.124	2.422	0.125
bsi	y,x,w	3556	23838	4.865	0.319	2.531	0.125
bsc	y,x,size	5776	21618	4.519	0.324	2.546	0.125
im	y,x,duration	9879	17515	4.97	0.337	2.718	0.141
wg	y,x,o	9720	17674	4.753	1.771	2.781	0.125
pi	x,w,hr	1166	26228	1.715	1.567	2.657	0.094
g_ap	change_size,duration	17	2724	0	0.037	4.094	0.015
g_d	dist_std,w,end	8	2733	0	0.037	4.14	0.016
g_i	y,duration,speed_mean	843	1898	0.949	3.267	4.187	0.016
g_ac	o,y,direction_std	686	2055	4.956	0.487	4.187	0.015
g_mo	o,duration,speed_mean	1212	1529	4.868	0.327	4.125	0.016
g_f	size,speed_mean	242	2499	2.893	1.521	4.156	0.016
g_me	o_mean,size_min,y	1488	1253	3.495	0.718	4.094	0.016
g_w	o_mean,size_max,dist_mean	717	2024	2.65	1.729	4.078	0.015
g_gf	o_mean,dist_mean,speed_std	176	2565	3.977	0.663	4.125	0.016
g_gmo	o_mean,w,speed_mean	1083	1658	3.416	3.016	4.219	0.015
g_gme	y,duration,size_min	1171	1570	1.196	2.866	4.235	0.015
g_s	o,duration,dist_mean	199	2542	1.508	2.872	4.297	0.016
g_gi	n/a	0	2741	n/a	n/a	n/a	n/a
g_glv	size_mean,dist_std	21	2720	4.762	1.103	4.25	0.016
g_glo	size_min,duration,o	91	2650	1.099	0.792	4.297	0.015
g_fsc	o_mean,h,dist_mean	383	2358	1.305	1.018	4.25	0.031
g_mes	o_mean,o,size_min	1740	1001	1.092	1.099	4.234	0.015
g_ls	size_min,y	91	2650	1.099	0.792	4.266	0.015
g_fst	dist_max,speed_mean	2	2739	0	0.694	4.078	0.016
g_fe	size_std,dist_max	2	2739	0	0.767	4.079	0.015
g_fv	size,dist_std	1	2740	0	0	4.078	0.016
g_det	dist_mean,dist2_max	2741	4941	3.685	3.845	4.233	0.016
f	x,o,size_size_mean	365	27029	4.932	1.325	2.375	0.125
lv	size,speed,	101	27293	1.98	1.412	2.407	0.109
lg	orientation,duration,speed	273	27121	4.762	1.427	2.328	0.141
lo	size,x,speed	728	26666	4.533	1.264	2.375	0.109
dd	x,y,orientation	1139	26255	4.565	1.261	2.437	0.125

Figure 5: Training and testing results on same ground-truth data

algorithm is also stable in terms of low standard deviation: The average of $(\sigma_{error_{fn}}, \sigma_{error_{fp}})$ of all labels are $(1.11\%, 0.38\%)_{exp2}$ and $(1.82\%, 0.68\%)_{exp3}$.

Fig.6 also shows that the false positive rates are in general lower than the false negative rates. This reveals the power of the algorithm in eliminating negative training samples, which somehow compromises detection of positive testing samples.

5.3 Exp.4: ROC Curve

We repeated Exp.2 but with different detection rates. The ROC curve is shown in Fig.7. Due to the limited space, here we only show several labels with large error rate in Exp.2.

As we have expected, there is a tradeoff between a high detection rate and a low false alarm rate. Please be aware that the detection rates we can directly control are for training, not for testing, but this can end up influencing the detection rate and false alarm rate of testing.

label	75% training, 25% testing				50% training, 50% testing			
	fn μ	fn σ	fp μ	fp σ	fn μ	fn σ	fp μ	fp σ
ap	9.176	1.279	0.692	0.154	11.096	1.965	0.687	0.199
di	6.717	1.251	0.009	0.001	10.729	2.643	0.011	0.002
o	1.96	0.425	0.153	0.047	3.04	0.277	0.251	0.05
in	7.92	1.304	0.238	0.036	11.15	3.48	0.304	0.042
ac	9.007	2.347	0.588	0.09	11.242	3.672	0.58	0.148
wk	7.026	2.547	4.35	0.829	10.89	3.509	3.893	1.573
r	6.116	1.58	0.67	0.2	7.066	2.665	1.28	0.322
br	0.757	0.184	0.118	0.028	1.298	0.325	0.229	0.094
wr	2.478	0.663	0.195	0.051	3.045	0.797	0.233	0.063
m	2.36	0.523	2.115	0.406	3.333	0.863	2.602	0.663
is	11.334	1.842	3.632	0.553	13.242	4.588	4.559	1.172
bsi	9.882	2.168	0.632	0.175	8.549	3.773	0.741	0.173
bsc	8.665	1.571	0.579	0.151	5.634	1.418	0.688	0.274
im	6.114	1.824	0.536	0.092	9.401	3.828	0.721	0.215
wg	7.041	2.34	2.872	0.641	14.687	1.673	3.125	1.075
pi	2.952	0.662	2.907	0.587	3.205	1.408	2.762	0.743
g_ap	0	0	0.068	0.013	0	0	0.09	0.022
g_d	0	0	0.068	0.015	0	0	0.088	0.014
g_j	2.129	0.34	7.482	1.146	2.185	0.58	8.32	1.636
g_ac	9.55	1.918	0.994	0.159	8.666	3.329	0.976	0.269
g_mo	10.228	1.683	0.368	0.106	7.415	2.541	0.872	0.239
g_f	5.448	1.329	3.089	0.53	4.463	2.062	2.917	1.258
g_me	6.537	1.741	1.498	0.17	9.533	2.138	2.015	0.573
g_w	3.999	0.534	2.821	0.283	6.453	1.641	4.012	1.41
g_gf	9.528	0.748	1.134	0.297	8.519	3.078	1.92	0.551
g_gmo	4.271	1.463	4.325	0.741	8.174	2.168	4.941	1.782
g_gme	1.75	0.51	6.094	0.73	3.321	0.821	4.168	2.044
g_s	3.108	0.792	6.084	0.639	3.693	0.816	7.297	0.345
g_gi	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
g_glv	6.862	1.754	2.242	0.406	9.379	2.928	2.356	0.732
g_glo	1.424	0.31	1.198	0.291	2.075	0.769	1.829	0.471
g_fsc	2.079	0.464	2.279	0.395	2.425	0.423	2.397	0.71
g_mes	1.706	0.569	2.321	0.384	3.198	0.484	2.974	0.491
g_ls	1.631	0.526	1.515	0.32	2.966	0.987	1.999	0.424
g_fst	0	0	1.226	0.284	0	0	1.329	0.246
g_fe	0	0	1.39	0.26	0	0	1.631	0.712
g_fv	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
g_det	9.058	0.858	8.104	2.02	6.985	3.026	8.638	2.487
f	7.202	2.116	1.947	0.629	10.312	3.13	1.78	0.861
lv	3.893	0.815	3.068	0.502	5.409	1.176	3.823	0.881
lg	8.335	0.804	3.19	0.201	13.433	2.303	3.044	0.886
lo	5.349	1.705	1.314	0.308	8.286	2.379	2.39	1.182
dd	7.245	2.089	1.765	0.572	8.105	1.117	1.916	0.795

Figure 6: Training and testing results on different ground-truth data

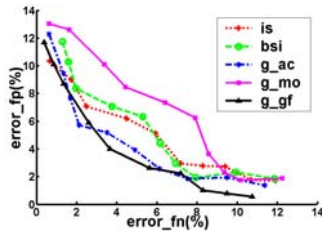


Figure 7: ROC curve

5.4 Exp.5: Results on sequence Fight_RunAway2

In the last experiment, we tested our labeling algorithm as well as our tracking algorithm on sequence Fight_RunAway2. Several frames are shown in Fig.3. We used training result from Exp.1 and we found that it could successfully detect all critical events such as fight start, fight end, leaving group, etc. Since no ground-truth data is provided for this sequence, we don't have a quantitative evaluation for the results.

6 Conclusion

We have addressed the challenging problem proposed by *PETS'04* for automatic labeling of human activi-

ties in a video sequence. The major contributions are: (1) We have presented a novel approach for tracking multiple objects using the appearance model which combines color and edge information of the detected blob, and in return is invariant to 2D rigid transformation and scaling; (2) We have proposed a statistical learning approach for detection of primitive activities which does not require any knowledge of activity models; (3) We have developed a training algorithm to select best features and combine them to form a stronger classifier. This algorithm is generic and thus applicable to other classification problems.

Acknowledgement

This research was supported, in part, by the Advanced Research and Development Activity of the U.S. Government under contract No. MDA904-03-C1786.

References

- [1] Hilary Buxton and Shaogang Gong. Advanced visual surveillance using Bayesian networks. In *Proc. of IEEE Workshop on Context-Based Vision*, 1995.
- [2] Y. Chen, Y. Rui and T. Huang. JPDAF Based HMM for Real-Time Contour Tracking. In *Proc. of IEEE CVRP*, pp. 543-550, Dec. 2001.
- [3] I. Cohen and H. Li. Inference of Human Postures by Classification of 3D Human Body Shape. In *Proc. of IEEE IWAMFG*, 2003.
- [4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms*, Second Edition, page 382, MIT Press, Sept. 2001.
- [5] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1972.
- [6] A. Elgammal and L. S. Davis. Probabilistic Framework for Segmenting People Under Occlusion. In *Proc. of IEEE ICCV*, 2001.
- [7] Hashlamoun, W.A., Varshney, P.K. and Samarsooriya, V.N.S. A tight upper bound on the Bayesian probability of error. In *Trans. of IEEE PAMI*, 16(2):220-224, Feb. 1994.
- [8] Somboon Hongeng, Ramakant Nevatia. Large Scale Event Detection Using Semi-Hidden Markov Models. In *Proc. of IEEE ICCV*, pp. 1455-1462, Oct. 2003.
- [9] S. Intille and A. Bobick. Recognizing planned multi-person action. In *Jour. of Computer Vision and Image Understanding*, vol. 3, pages 414-445, March 2001.
- [10] J. Kang, I. Cohen and G. Medioni. Continuous Tracking Within and Across Camera Streams, In *Proc. of IEEE CVPR*, 2003.
- [11] J. Ohya, J. Yamato and K. Ishii. Recognizing human action in time-sequential images using a hidden Markov model. In *Proc. of IEEE CVPR*, 1992.
- [12] H. Roh, S. Kang and S. Lee. Multiple People Tracking Using an Appearance Model Based on Temporal Color. In *Proc. of IEEE ICPR*, pp. 643-646, 2000.
- [13] Yifan Shi and Aaron Bobick. P-Net: A Representation for Partially-Sequenced, Multi-stream Activity. In *Proc. of IEEE Workshop on Event Mining*, 2003.
- [14] Van Trees, H. L. *Detection, Estimation and Modulation Theory, Part I*. John Wiley and Sons, New York, 1968.
- [15] H. Zhang and J. Malik. Learning a discriminative classifier using shape context distance. In *Proc. of IEEE CVPR*, Vol. 1, pp. 242-247, 2003.

Joint Appearance and Trajectory-based Data Association for Multi-object Tracking

Arvind Lakshmikummar, M.Shivram
Honeywell Technology Solutions Labs, Bangalore

Michael Burl, Gautam Apte
Department of Computer Science
University of Colorado, Boulder

1 Abstract

Automated video surveillance demands the ability to detect, track and analyze the behavior of multiple moving objects at high frame rates. The system should also be lenient towards temporary occlusions of the objects and also to missing frames.

In this paper, we begin with the assumption that moving objects in a scene have been detected using some reasonable motion detection algorithm. We proceed on to describe an algorithm that tracks these detected objects across frames, and builds a history of object state and appearance that can be used for doing higher-order inference about the activities in the video. The key tracking step, *Data association* is performed by integrating appearance and trajectory information. The appearance component exploits the color or gray-level composition of the detected objects, while the trajectory component exploits models of the object dynamics. Combining these two sources of information makes the system more robust to occlusions and allows it to handle multiple objects crossing each other. For every tracked object, we extract its bounding box and build the motion histogram of the object's motion history image. This motion histogram provides an insight to the kind of activity being performed by that particular object. We built motion histograms for a range of activities (walking, running, pushing, punching and kicking) and were successful in tracking the object performing these activities.

2 Introduction

The objective of an automated video surveillance system is to monitor a given environment with minimal intervention from an operator. Generally when monitoring a given scene for suspicious activity, we would like to answer the following questions.

- Is there any change in the scene?
- If so, what objects are moving about the environment?
- Where are they located?
- Where are they going?
- What are they doing?

A variety of motion detection algorithms, based on different methods have been proposed [4] that lend answers to the first three questions. In this paper, we assume the existence of a reasonable detection algorithm that can provide us with the segmented blobs that represent moving foreground objects. We propose a method that can robustly track these segmented blobs, label them and use this information to do some reasoning on their actions. This paper is organized as follows. Section 3 describes a filtering mechanism that helps a detection algorithm in handling shadows caused by illumination changes. Section 4 illustrates the data association methods adopted by us to associate the different detected blobs between frames. In Section 5, we describe a ranking assignment algorithm, based on *Bertsekas' auction algorithm* to find the optimal set of assignments between multiple targets. Experimental results are presented in Section 9. A discussion of the strengths and weaknesses of the current approach along with directions for future work are outlined in Section 10. Final conclusions are presented in Section 11.

3 Motion Detection

Detection of moving objects in video streams is a significant and difficult, research problem by itself. Apart from its intrinsic utility of segmenting video streams into moving and background components, detecting moving blobs provides a focus of attention for recognition, classification, and activity analysis, making these later processes more efficient since only moving pixels need be considered. There are three conventional approaches to moving object detection: temporal differencing, background subtraction and optical flow. Temporal differencing is very adaptive to dynamic environments, but generally does a poor job of extracting all relevant feature pixels. Background subtraction provides the most complete feature data, but is extremely

sensitive to dynamic scene changes due to lighting and extraneous events. Optical flow can be used to detect independently moving objects in the presence of camera motion; however, most optical flow computation methods are computationally complex, and cannot be applied to full-frame video streams in real-time without specialized hardware.

We also note some general concerns with background subtraction techniques that are not specific to the background estimation scheme:

- Stationary camera and stationary background assumptions.
- Fragmentation and disappearances. If an object passes over a region of similar gray-level (or color if using color images), the object may completely disappear. Similarly, if only part of an object, say a person’s pants, match the background then that part will disappear in the background subtraction leading to fragmentation of the person (the head and torso may appear disconnected from the shoes).
- Easily tricked by sudden, gross changes due to illumination/weather.

These issues can wreck havoc on a tracking system and make it impossible to uniquely associate objects. To keep the overhead of detection very low and maintain focus on the problem of tracking, we adopted a simple adaptive background subtraction technique. However to tackle the problems described above, we apply a preprocessing step prior to the background estimation.

The grouping or clustering process is a straightforward extension of the classical connected components algorithm [9]. Instead of requiring that a pixel be strictly connected to another (e.g., an 8-neighbor), there is a distance threshold that requires any two pixels within the specified spatial distance threshold from each other to be assigned to the same cluster. This rule is applied transitively as in the single linkage clustering algorithm [3].

4 Gating and Association

4.1 Trajectory-based

Given that we are tracking an object and have observations up to and including frame k , Equations 13 and 14 provide a means to predict where the next observation from this track should occur and how much uncertainty from the predicted position we might expect. By setting a bound on the squared Mahalanobis distance from $\hat{\mathbf{z}}_{k+1|k}$, i.e., on

$$d^2 = (\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k})^T \hat{\Sigma}_{k+1|k}^{(z)-1} (\mathbf{z}_{k+1} - \hat{\mathbf{z}}_{k+1|k}) \quad (1)$$

we can establish an elliptical *gate* into which the next measurement for a particular track should fall. Since we will be tracking multiple objects, we expect there to be multiple detected objects (observations) and more than one object may fall into the gate of a particular track. If we have n_t currently active tracks and n_z separate observations (“objects”) in the current frame then the process of gating amounts to determining a $(n_t \times n_z)$ trajectory association matrix \mathbf{A}^{traj} in which entry $A_{i,j}^{\text{traj}} = 0$ if measurement j does not fall inside gate i and will be non-zero if measurement j does fall in gate i . For convenience, we assign $A_{i,j}^{\text{traj}}$ to be the Mahalanobis distance squared of measurement j from the position predicted for the measurement in track i .

4.2 Appearance-based

For the tracking system to perform properly, the most likely measured potential target location should be used to update the target’s state estimator. The probability of the given measurement being correct is some distance measure between the predicted state of the target and the measured state. The state vector used in traditional data association methods, like the one above is normally the position vector of the object. Limiting ourselves to position alone gives rise to problems under conditions of occlusion. We propose an extension that utilizes color information from the scene and uses that as a feature in conjunction with the position. This allows objects to cross directly in front of one another without losing track of which is which after they separate. The color histogram is well suited to this task because of its ability to implicitly capture complex, multi-modal patterns of color. Moreover, because it disregards all geometric information, it remains invariant to non-rigid motions.

The motion detection module, followed by spatial clustering gives an estimate of a detected object’s bounding box. We construct a color histogram by filling buckets of a transformed discretized color space [R + 16 G + 256 B] with pixels inside the bounding box. This serves as a model histogram M . With every new frame, the bounding boxes are computed and the candidate histograms are computed. If we assume the color of the model has a density function q_z and the candidate object at location y has a density function distributed as $p_z(y)$, then [8]

$$\rho(y) = \rho[p(y), q] = \int \sqrt{p_z(y)q_z} dz \quad (2)$$

This is the general form of the Bhattacharya coefficient and its computation from sampled data involves estimating the densities p and q . In our case the densities are given by the color histograms of the model and the candidate targets. Based on the sampled estimates of the Bhattacharya coefficient, the distance between the two (model and candidate) color distributions is given by

$$d(y) = \sqrt{1 - \rho[p(y), q]} \quad (3)$$

This measure is a metric valid for arbitrary distributions and is invariant to scale.

4.3 Joint Association

In order to obtain a distance metric for data association that incorporates both the histogram intersection and position difference, we calculate the joint probability of the two measurements, appearance and position. Let us define $X_{i,j}$ as the event that a detected object i is actually the previous object j , $Y_{i,j}$ as the value of the histogram intersection between objects i and j , and $Z_{i,j}$ as the distance between the position of object i and the predicted position of object j . Since the color and position measurements are statistically independent, the conditional probability can be expressed as

$$P(X_{i,j}|Y_{i,j}, Z_{i,j}) = \frac{p(Y_{i,j}|X_{i,j})p(Z_{i,j}|X_{i,j})P(X_{i,j})}{p(Y_{i,j})p(Z_{i,j})} \quad (4)$$

If we assume equal prior probabilities for all $X_{i,j}$, the above equation can be simplified to maximize $F_y(Y_{i,j})F_z(Z_{i,j})$, where

$$F_y(Y_{i,j}) = \frac{p(Y_{i,j}|X_{i,j})}{p(Y_{i,j})} \quad (5)$$

$$F_z(Z_{i,j}) = \frac{p(Z_{i,j}|X_{i,j})}{p(Z_{i,j})} \quad (6)$$

where $F_y(Y_{i,j})$ and $F_z(Z_{i,j})$ are functions that represent the feature difference and position difference respectively. Using this naive Bayes approach, we have a joint association matrix A^{joint} that now has information from both color and position. This association matrix represents the probabilities that a new measurement belongs to an already detected blob, is a newly detected blob or is a false alarm.

5 Solving the joint association matrix and linking trajectories

Motion detection for each frame image A^t leaves us with a set of T (total number of frames) matrices $C^t \in \mathbb{R}^{N_t \times 2}$, where the total number of objects in frame t is given by N_t . The task is now to identify observations of the same object in subsequent frames in order to link $C^t_{t=1}^T$ into trajectories over time. The basic idea is to determine a set of associations between two sets of detected object states, such that the set of associations is optimal (minimizes a linear cost

functional). Based on the joint association computed from the previous section, we define an association matrix G with element $G(i, j) = g_{ij}$ equal to 1 if p_i in frame t and q_j in frame $t + 1$ are produced by the same object. In order to let the number of detected objects vary between frames, i.e. $N_t \neq N_{t+1}$, the association matrix is augmented with both a row g_{0j} and a column g_{i0} for dummy targets at times t and $t + 1$. Linking an object to a dummy means that the detected object has disappeared between frames t and $t + 1$ and linking the dummy to an object means that the object has freshly appeared in the scene. With this in place, the following topology constraint on G can be formulated

Every row $i > 0$ of G and every column $j > 0$ of G must contain exactly one entry with value , all others zero. Row 0 and column 0 can contain more than one entry

In order to find an optimal set of links, g_{ij} , we need to define the functional to be minimized. The only restriction in the definition of such a functional is that it needs to be linear in the association variables g_{ij} and may be written as

$$\phi = \sum_{i=0}^{N_t} \sum_{j=0}^{N_{t+1}} \phi_{ij} g_{ij} \quad (7)$$

where, ϕ_{ij} represents the cost of associating object p_i in frame t with object q_j in frame $t + 1$. The definition of ϕ encompasses information about the objects' states (position and color).

5.1 Initialization

The association matrix G is initialized by assigning each particle in frame t to its "nearest neighbor" (using the distance measure ϕ) in frame $t + 1$ that is not already assigned to some other object. This means that, for every $i = I, j = J$ is chosen such that ϕ_{IJ} is the minimum of all ϕ_{IJ} for which no other g_{iJ} is set to one. This g_{IJ} is then set to one. If no such minimum is found then the object is linked to a dummy, i.e. $g_{I0} = 1$. After having done this for all objects, p_i , every J for which no g_{iJ} is set is determined and the corresponding $g_{0J} = 1$. This initialization creates a matrix G that fulfills the above mentioned topology constraint. In practice, this solution will be very close to the optimal since only few conflicts will occur in practice.

6 Tracking

Object detection and spatial clustering identify moving regions and group these together into spatially consistent "objects". The next steps involve linking the objects from different frames together into tracks. For these steps, we have used the machinery of the Kalman filter [10, 1].

Underlying the Kalman filter is a linear dynamical model of the following form (following the notation of [1]):

$$\mathbf{x}_{k+1} = \mathbf{F}_k \mathbf{x}_k + \mathbf{G}_k \mathbf{w}_k \quad (8)$$

$$\mathbf{z}_k = \mathbf{H}_k^T \mathbf{x}_k + \mathbf{v}_k \quad (9)$$

where \mathbf{x}_k is an $(xr \times 1)$ vector describing the state of a particular object at discrete time-step k (frame number in our case), \mathbf{w}_k is a $(wr \times 1)$ noise process (assumed Gaussian with zero-mean and covariance \mathbf{Q}_k) that drives the object dynamics, and \mathbf{z}_k is the $(zr \times 1)$ measurement (observation) made at time-step k . Note that \mathbf{z}_k is a function of the system state and the noise process \mathbf{v}_k (assumed to be Gaussian with zero mean and covariance \mathbf{R}_k). The statistics of the initial state \mathbf{x}_0 are assumed to be Gaussian with mean $\bar{\mathbf{x}}_0$ and covariance \mathbf{P}_0 .

The dynamical equations above satisfy the discrete Gauss-Markov property so that the state at time step $k+1$ is Gaussian distributed and the density $p(\mathbf{x}_{k+1}|\mathbf{x}_k, \dots, \mathbf{x}_0) = p(\mathbf{x}_{k+1}|\mathbf{x}_k)$, i.e., knowing the most recent state \mathbf{x}_k tells us as much about \mathbf{x}_{k+1} as knowing the full state history. Since the various quantities are Gaussian distributed, we only need to keep track of means and covariances to know the full distributions.

Given the sequence of observations $\mathcal{Z}_k = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k\}$, we will define two estimators: one estimates the current state and the other predicts the next state (one-step prediction).

$$\hat{\mathbf{x}}_{k|k} = E[\mathbf{x}_k | \mathcal{Z}_k] \quad (10)$$

$$\hat{\mathbf{x}}_{k+1|k} = E[\mathbf{x}_{k+1} | \mathcal{Z}_k] \quad (11)$$

$$(12)$$

The notation $\hat{\mathbf{x}}_{k|k}$ means the estimate of the state at time step k given observations up to and including time step k . Similarly, the notation $\hat{\mathbf{x}}_{k+1|k}$ means the estimate of the state at time step $k+1$ given measurements up to and including time step k . In addition to the state estimators, we also keep track of the covariance of the estimates with $\hat{\Sigma}_{k|k}$ and $\hat{\Sigma}_{k+1|k}$. From the one-step state predictor and its covariance, we can determine the expected value for the observation and its covariance. Specifically,

$$\hat{\mathbf{z}}_{k+1|k} = \mathbf{H}_k^T \hat{\mathbf{x}}_{k+1|k} \quad (13)$$

$$\hat{\Sigma}_{k+1|k}^{(z)} = \mathbf{H}_k^T \hat{\Sigma}_{k+1|k}^{(x)} \mathbf{H}_k^T + \mathbf{R}_k \quad (14)$$

Given that we want to track both position and color, the Kalman filter equations described in Section 6 need to be modified accordingly. The state vector $\mathbf{x}_k = [p_k^T, \mu_k^T, \phi_k(C_k)]^T$, now contains both the centroid information, $p_k = [x_k, y_k]$, the mean $\mu_k = [\mu_{1k}, \mu_{2k}]^T$ of the updated histogram M_k in a transformed RGB color space and the value $\phi_k(C_k)$ corresponding to the histogram intersection between the histogram of the bounding region in the new frame (C_k) with the one in the previous frame (M_{k-1}).

7 Update and Prediction

7.1 Measurement Update (Correction)

Once a specific measurement is associated to a specific track, we can perform the *measurement update*. The trajectory update is based on the Kalman filter update equations, while the appearance update is simply a weighted average of the current histogram (from the measurement) and the previous histogram (from the associated track).

The trajectory update step amounts to computing an estimate of the current state using all of the observations up to and including the current time. The equations can be conveniently written in terms of the *innovation*, $\boldsymbol{\nu}_k$, which reflects the difference between the actual measurement and the predicted measurement.

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{L}_k \boldsymbol{\nu}_k \quad (15)$$

$$\hat{\Sigma}_{k|k} = \hat{\Sigma}_{k|k-1} - \mathbf{L}_k \mathbf{H}_k^T \hat{\Sigma}_{k|k-1} \quad (16)$$

$$\boldsymbol{\nu}_k = \mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1} \quad (17)$$

$$\mathbf{L}_k = \hat{\Sigma}_{k|k-1}^{(x)} \mathbf{H}_k \hat{\Sigma}_{k|k-1}^{(z)} \quad (18)$$

The matrix \mathbf{L}_k is known as the innovations gain matrix.

The Kalman filter is also able to track through situations in which a measurement is not acquired during a particular frame. In this situation, the state estimate and its covariance are just based on the predictions. Thus, if no measurement is acquired, the tracker uses:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} \quad (19)$$

$$\hat{\Sigma}_{k|k} = \hat{\Sigma}_{k|k-1} \quad (20)$$

When this happens, the uncertainty in the state estimate grows with each frame, which can eventually lead to problems with the gating process. In practice, if a measurement is not acquired for some limited number of frames the track is deactivated. If the object is later detected again, it will be assigned a new track.

The appearance update is given by:

$$\phi^{(NEW)} = (1 - \beta)\phi^{(OLD)} + \beta\phi^{(CURRENT)} \quad (21)$$

7.2 Time Update (Prediction)

For the trajectory, we need to predict ahead to the next frame to aid in locating the next measurement. For the appearance, we do not have a dynamical mode, so the appearance at the next frame is predicted to be the same as the appearance from the measurement update.

The time update step of the Kalman filter involves predicting the state in the next frame and the covariance of that

prediction. Basically, this step reduces to propagating the current estimates through the linear dynamical model.

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{F}_k \hat{\mathbf{x}}_{k|k} \quad (22)$$

$$\hat{\Sigma}_{k+1|k} = \mathbf{F}_k \hat{\Sigma}_{k|k} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T \quad (23)$$

Similarly for the measurements, we can predict ahead one step to find (same as Equations 13 and 14):

$$\hat{\mathbf{z}}_{k+1|k} = \mathbf{H}_k^T \hat{\mathbf{x}}_{k+1|k} \quad (24)$$

$$\hat{\Sigma}_{k+1|k}^{(z)} = \mathbf{H}_k^T \hat{\Sigma}_{k+1|k}^{(x)} \mathbf{H}_k^T + \mathbf{R}_k \quad (25)$$

If the predictions for a given track are outside of the visible image (taking into account the covariance and gate size), then the track is deactivated. If we fail to take this action, the number of tracks that have to be updated will grow linearly with the number of frames collected.

The time update and deactivation step completes all the work to be done on the current frame. The algorithm then returns to the top of the process and repeats on the next frame.

8 Activity Analysis

Once the targets have been detected and tagged with a unique label, we extract the bounding boxes of the tagged targets. The region within this bounding box forms the basis of our analysis. The Motion History Image (MHI) [13] focusses on accumulating and recognizing patterns of motion rather than structural features. The MHI is constructed by successively layering selected image regions over time using a simple update rule:

$$MHI_\delta(x, y) = \tau, \psi(I(x, y)) \neq 0 \quad (26)$$

$$MHI_\delta(x, y) = 0, MHI_\delta(x, y) < \tau - \delta \quad (27)$$

where each pixel (x, y) in the MHI is marked with a current timestamp τ if the function ψ signals object motion in the current video frame. The remaining timestamps in the MHI are removed if they are older than the decay value $\tau - \delta$. This update function is called for every new video frame.

The function ψ that selects a pixel location in the input image can be arbitrarily specified. As described in [12], we apply a fixed Sobel gradient mask to compute the motion vectors of the image. The convolution with the gradient mask will ensure consistency in the motion vectors. Prior to performing this analysis, we establish ground truth information for the various activities/roles under consideration. A priori, we apply the above described process and build a database of motion histograms of detected objects in a variety of roles. The activities we considered were walking,

running, punching, kicking, pushing and waving. For every frame, we extract the motion histogram and compute a measure of similarity/dissimilarity with the histograms in the database. The measure we adopt is the Bhattacharya coefficient and it provides a reasonable estimate of the correlation between different motion histograms.

9 Experiments and Results

The key claims we have made are that the set of algorithms described earlier can robustly detect and track multiple moving objects under varying indoor conditions. We further claim that comparisons of the motion histograms of the motion history images gives a fair idea on the kind of activity being performed. In this section, we illustrate some of the results obtained by us on the PETS datasets.



Figure 1. Walk1



Figure 2. Walk1

10 Discussion and future work

Once the moving targets have been detected and tracked (assigned unique labels), the activity analysis routine (as described in Section 8) comes into play and is applied to the tracked region. The data from the detection and tracking routines is collated with the results of activity analysis to build an information table similar to the PETS ground



Figure 3. Walk1



Figure 7. Fight1



Figure 4. Meet Crowd



Figure 8. Fight1



Figure 5. Meet Crowd



Figure 9. Fight1



Figure 6. Meet Crowd



Figure 10. Fight1

truth data. The results of our analysis depend very heavily on the kind of detection output obtained. When people in the scene are very close to each other, the detection module has difficulty in segmenting the people individually. This leads to the detected target being tracked together as a group (as shown in the results above). We tested our algorithm on the available datasets and the results of activity classification have been favorable. We compared our results with the PETS ground truth data and tabulated deviations on positions, bounding box coordinates, track appearance/disappearance, role and activity of the tracked object.

Some scope for future work in comes from the rationale behind combining a color histogram and a position tracker? There are a lot of existing algorithms to track multiple objects. Each of these come with their own advantages and disadvantages. Our experiments gave us empirical evidence that this combination is potent to track objects effectively. Our current system combines these two approaches in a simple unweighted fashion. If we would like give importance to either of these methods, we need a good mechanism to determine how they complement each other. We could think of these algorithms as models we could be applying to a tracking system. Let us take the case where we apply a position tracker. The measurable output of this tracker would be the centroids of the object(s) that is being tracked. We could treat this output as a random variable and define a distribution on how it evolves over all the frames. We could also construct a similar distribution for the color tracker. These two random variables have a joint probability distribution and also have their individual distributions. By measuring the amount of mutual information (KL-Divergence) in these distributions, we can come to a mutual agreement on how to weight these two models.

11 Conclusions

In this paper, we have developed a prototype system that can effectively detect and track multiple moving objects in a dynamic environment. We have proposed a method to combine two different tracking approaches and demonstrated a technique to integrate them and manage the tracks generated. We have also experimentally validated and justified the combination.

References

- [1] B.D.O. Anderson and J.B. Moore, *Optimal Filtering*, T. Kailath (ed), Prentice-Hall. (1979)
- J. Canny, "A Computational Approach to Edge Detection", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **8**(6), 679-698, (1986).
- [2] R.T. Collins, A.J. Lipton, H. Fujiyoshi, T. Kanade, "Algorithms for Cooperative Multisensor Surveillance", *Proc. of the IEEE*, Vol. 89, No. 10, (Oct 2001).
- [3] R.O. Duda, P.E. Hart, D.G. Stork, *Pattern Classification*, John Wiley and Sons, (2001).
- [4] R.T. Collins, A.J. Lipton, H. Fujiyoshi, T. Kanade, "Algorithms for Cooperative Multisensor Surveillance", *Proc. of the IEEE*, Vol. 89, No. 10, (Oct 2001).
- [5] H.G. Barrow and J. Tenenbaum, *Recovering intrinsic scene characteristics from images*, In A.R. Hanson and E.M. Riseman, editors, *Computer Vision Systems*, Academic Press, (1978)
- [6] E. H. Land and J. J. McCann, *Lightness and retinex theory*, *Journal of the Optical Society of America*, **61** 1-11, (1971).
- [7] Andrzej J. Kasinski and Alaa M. Hamdy, *Segmentation based on homomorphic filtering and improved seeded region growing for mobile robots tracking in image sequences*, *Machine Graphics & Vision International Journal*, **10**, 447-466, (2001).
- [8] Dorin Comaniciu, Peter Meer, *Mean Shift Analysis and Applications*, *ICCV* (2), 1197-1203, (1999)
- [9] B.K.P. Horn, *Robot Vision*, MIT Press, (1986).
- [10] R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", *J. Basic Eng.*, ASME, Series D, Vol 82, No. 1, pp. 33-45, (1960).
- [11] M. J. Swain ,D. H. Ballard, "Indexing via colour histograms", *ICCV'90*, pp. 390-393, (1990).
- [12] A. Bobick and J. Davis, "The Representation and Recognition of Action Using Temporal Templates", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 23, No. 3, 2001, pp. 257-267.
- [13] J.Davis, "Hierarchical Motion History Images for Recognizing Human Motion", *IEEE Workshop on Detection and Recognition of Events in Video*, Vancouver, Canada, July 8, 2001, pp. 39-46.
- [14] Stan Birchfield, "Elliptical head tracking using intensity gradients and color histograms", *Proceedings of CVPR*, pp 232-237, (1998).
- J. Wang, E.H. Adelson, "Representing Moving Images with Layers", *IEEE Trans. on Image Proc.*, Vol. 3, No. 5, pp. 625-638, (1994).

Is it interesting?

Comparing human and machine judgements on the PETS dataset

Hannah Dee & David Hogg
School of Computing
University of Leeds
Leeds LS2 9JT, United Kingdom
{hannah}{dch}@comp.leeds.ac.uk

Abstract

This paper presents a novel approach to evaluating the detection of unusual or interesting events in videos involving certain types of human behaviour, such as pedestrian scenes. The holy grail of computer vision for surveillance can be thought of as an interesting or unusual event detector which when given an input video stream, outputs some form of alarm whenever anything unusual happens inside its field of view. This paper addresses the question of how we would go about evaluating such a system, suggests one possible evaluative schema, and presents an example of this evaluative procedure in use on a prototype Interesting Event Detector.

1 Introduction

When you monitor a pedestrian scene, a number of different behaviour patterns can be observed. People walk along pathways and cars are driven along roads, and occasionally people will take shortcuts or get into a car or stop for a chat. Very occasionally, someone will do something different or *interesting* – something that does not fit our general understanding of what behaviour people exhibit in that scene. Humans are very good at detecting such events, but are not so good at articulating what exactly it is that makes such events unusual or interesting. It is worth making the distinction between events which are *interesting* and events which are *atypical* - most Computer Vision systems for surveillance attempt to detect the latter, whereas humans are much more interested (by definition) in the former. It is easy to imagine a system which would ring an alarm if a pedestrian strayed from a path. But if the path was blocked, this behaviour, although atypical, would not really be interesting.

A number of systems have been constructed which can

be portrayed as attempts at identifying such events. Many systems are actually designed to do something else, and atypicality detection emerges as a “bonus” feature: by modelling a particular feature of the environment (path usage, patterns of pedestrian motion over time etc.) and determining which instances *do not* fit the model, some form of interesting-event detector has magically been constructed.

One approach is exemplified in [4], in which the typicality or otherwise of pedestrian trajectories is assessed based upon learned models of absolute location and speed over time. In [6], a model of the paths within a scene is constructed based upon the behaviour of pedestrians, and this path model can subsequently be used to detect unusual trajectories. The relationships between objects can also be used to judge typicality [7]. In [9] patterns of activity are learned at a site, and unusual event detection is performed by spotting events which do not fit the pattern or co-occurrence data. In [3] “suspicious” behaviour is correlated with the rapid head movements.

Determining the overall effectiveness of algorithms of this type has historically been unsystematic. This is acknowledged by the authors of [9], who state they are working on methods of evaluating the unusual event detection aspect of their work.

Evaluative techniques at their simplest involve investigating the problematic cases by hand - looking at the outliers - and saying “Yes, that’s unusual” [9, 4]. One model, trained on pedestrians, had a major outlier which turned out to be a cyclist. This is, of course, confirmation that the model provides a reasonable basis for the detection of strange pedestrian activity, however, the confirmation such evidence provides is at best anecdotal. It is also completely self-justifying - if we look at the examples which do not fit the model, and find they are odd in some way, then of course they are interesting to us - they fit our frame of reference (or rather, they don’t fit our frame of reference) by definition.

Another means of evaluating such systems is through the

use of “actors”¹. These people are recorded behaving in an unusual fashion, and the system in question is evaluated on its ability to single out the sequences featuring the strangely behaving actors [3, 7, 4]. Problems with this approach are manifold, but all hinge upon the question of *whose idea of interesting or unusual we are dealing with*. If the decision as to what constitutes unusual behaviour is left up to the actors, questions about who the actors are, what their preconceptions of the project are and most importantly, their links to the software designers, become paramount. If the actors are lab-mates of the paper author, do they know how the algorithm in question works? The alternative case, where the actors are instructed by the system designer on the nature of unusual behaviour, could be even worse - it is easy to imagine a scenario in which the instruction “We need some footage of suspicious behaviour, like walking from car to car across the car park in a wavy line” is issued. This is not exactly *good science*.

Computer Vision systems for surveillance are generally model based. And things which do not fit the model can only be classed as unusual or interesting with respect to that model. We cannot really claim that events which fall outside the model are interesting or unusual - all we can really say about them is just that they don’t fit the model. Thus we really cannot claim any more or less for these interesting event detectors until we have a more principled way of evaluating their performance. This paper proposes a way out of this model-based trap - by providing a form of “ground-truth” for interestingness.

2 Am I interesting or not?

Within the surveillance domain, what we are interested in are events which might be associated with criminal or dangerous behaviour. A recent study [10] investigates whether such events can be predicted from CCTV footage - that is, whether it is possible to distinguish sequences where a crime was about to occur from neutral sequences. The authors conclude that not only is it possible, but that naïve observers perform as well as trained security guards. This suggests that there is no learned or innate ability to detect the type of events security guards detect.

Our central assumption is that benchmarking against a number of humans is an improvement over relying on the author, actors, or serendipity to provide some measure of the interestingness or otherwise of the data set.

The evaluative schema we propose involves requiring a number of volunteers (in this case, undergraduate and post-graduate students with no knowledge of the project being evaluated) to rank the behaviour of each agent in the scene

¹These actors often look suspiciously like computer vision postgraduates.

in question. To assist in this task, separate videos are produced for each agent containing only those frames of video encompassing the agent’s trajectory. A highlight indicates exactly the agent we are interested in - this makes the cognitive task of those evaluating much easier in scenes with multiple, occluded agents.

Volunteers are asked to rate the “interestingness” of these videos on a scale of 1 to 5. The instructions given to the volunteers were as follows:

“If you were a security guard, would you regard the behaviour of the agent highlighted in this video as interesting? Please indicate on the following questionnaire, with one being uninteresting and five being interesting.”

Volunteers were also invited to note down any comments they wished to make about any of the videos.

An average of the scores from the human rankers is then assumed to provide a simple measure of “interestingness”: we choose the median, as this is less sensitive to outliers. We can then compare it directly to the output of any machine generated indication of typicality, and if we want our system to output a binary decision (interesting, or not) we can use ROC graphs to assist in the determination of a threshold.

However, the median is just one statistic we can use: the advantage of having the opinions of a number of people is that there is a richness of information we can incorporate into our evaluations. We can, for example, calculate the correlation statistics - both within the human set (to determine consistency within the set of human rankers) and between the set of human rankings and the machine generated statistic. The correlation statistic applicable to this data is Spearman’s Rho [1], as the data is clearly non-parametric and on different scales - that is to say that any computer generated statistic is unlikely to map directly onto a 1-5 rating of interestingness. Nevertheless, if those videos rated highly by the computer are those videos rated highly by the human volunteers this is a positive result.

Spearman’s Rho is a similar calculation to the product-moment correlation (sometimes called Pearson’s), except Spearman’s operates on ranked data. Given ranked data, Spearman’s can be calculated using the following formula:

$$r_s = 1 - \frac{6 \sum_{i=1}^n d_i^2}{n(n^2 - 1)}$$

Where n is the number of videos, and d is the difference between the matched pairs of ranks. Spearman’s Rho can be tested for significance: for small values of n , r_s has a non standard distribution and specific tables must be used. For large ($n > 10$) values of n the following function of r_s follows approximately the distribution of a t-test statistic with $n - 2$ degrees of freedom:

$$t_s = \sqrt{\frac{n-2}{1-r_s^2}}$$

The resultant value t_s can be compared against any standard statistical tables for significance testing.

As well as the possibility of performing a range of statistical tests we have a wealth of qualitative information in the form of comments made by the subjects as they were ranking the dataset. These can help in instances where disagreement occurs - for example, in one outdoors scenario an object was reported as being highly interesting by several subjects, but the trajectory taken by that object was very dull. Inspection of their forms revealed that it was interesting because it was an ambulance.

3 The evaluative schema applied to the PETS dataset

A subset of the PETS2004 dataset was used in this study.² This consists of pedestrian footage filmed in a foyer situation, with actors performing various roles such as meeting, walking, fighting and browsing. Included in the dataset are various people we assume are bystanders. As we are only interested in evaluating high level classifications of behaviour (and not tracking), we only consider those videos for which ground truth has already been provided. We then exclude those agents whose trajectories are only partially covered by the video, and those agents who hover on the periphery. In short, we only analyse the main actors in each scene, and those bystanders whose trajectories are shown in full. This leaves us with a total of 23 agents from 12 movies. These are listed in detail, alongside an image showing the path of the trajectory, in Appendix A at the end of this paper.

The 12 original videos are used to produce 23 ($n = 23$) labelled videos. These were presented to 12 subjects ($n_s = 12$), who rated each on the 1-5 scale as detailed in Section 2. Spearman's Rho was calculated for each pair of human raters, giving a correlation matrix with 66 entries ($\frac{n_s^2 - n_s}{2}$). All of these correlations were positive, and 62 of the 66 were significantly positive at the 0.05 level. This means we can safely assume that the group of humans are in broad agreement about which clips are interesting.

It is interesting to take a closer look at the behaviour of those agents where the human rankers were in disagreement - where the standard deviation of the human scores is high. Some of these were due to partial trajectories, and to the inclusion of people such as ID1 from Walk3.mpg, who entered the scene then immediately turned around and left (we assume he was a passer-by, perhaps put off by the camera).

²This data comes from from the EC Funded CAVIAR project/IST 2001 37540

In particular, there are four cases in particular where the human rankings range from lowest (1) to highest (5) and it is worth investigating these in a little more detail:

- **ID 0 from Walk1.mpg:** Standard Deviation = 1.07. In this movie clip, the agent walks out and waves at the camera, then leaves the scene by the same door they came in from. The actor in this clip is presumably signalling to the camera person that they are ready to go, although this was not clear from context.
- **ID 0 from Rest_SlumpOnFloor.mpg:** Standard Deviation = 1.48. In this movie, the agent walks out of the scene (the clip clearly starts before the actor is ready) then re-enters, crosses to the object on the left, then sits on the floor for a short while before leaving. Some of the subjects think that sitting on the floor was uninteresting.
- **ID 1 from Meet_WalkSplit.mpg:** Standard Deviation = 1.62. This clip and the following feature agents entering the scene from different doors, meeting in the middle, and then leaving from different doors. Comments by those subjects who rated these clips highly indicate that they thought a package was passed between the two actors - which would be suspicious given the instructions to subjects.
- **ID 3 from Meet_WalkSplit.mpg:** Standard Deviation = 1.56. See above.

That there was disagreement between the human subjects on some of the clips should not be seen as a drawback to this evaluative schema - indeed, one of the reasons for including a number of subjects is to allow for such differences and disagreements. These help provide a richer framework against which to evaluate our software.

4 A prototype *interesting event* detector

The interesting event detector we will use as a demonstration is inspired Dennett's large body of work on intentional explanation (for example, [2]), which describes different ways of thinking about and explaining the behaviour of agents. It is hoped that this system, when completed, will provide a new way of thinking about the problem of behaviour modelling in the surveillance domain. However, it is still at the prototype stage and we will just sketch an overview of its operation here.

Our system tries to work out where the agent might be heading, and combines this hypothesis with a simple model of the way in which people intentionally navigate towards the geographical goals in a scene. In our original formulation, the hypothesis about where the agent may be heading

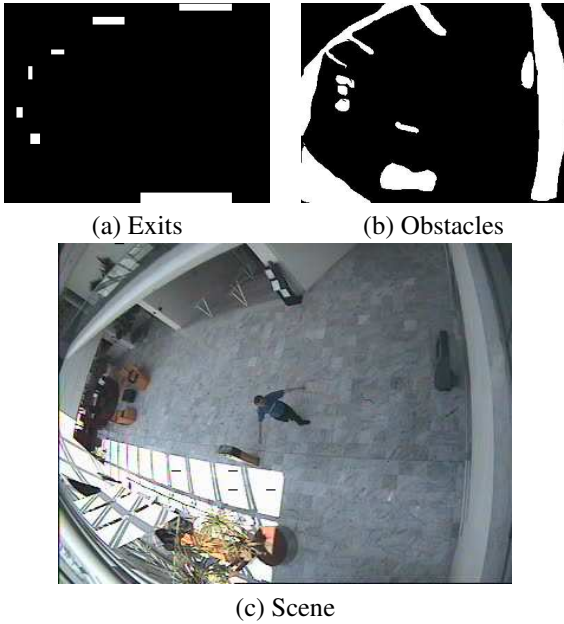


Figure 1. The exit model, obstacle model and scene.

is built up using information from a person tracker [5], an obstacle model and an exit model, but in the current implementation we simply use the ground truth information provided (specifically, the position of the object centroid) for the agent’s position (\mathbf{x}). We apply a Kalman filter to this and store the directional component to obtain an estimate of direction of travel θ .

All calculations are carried out in the image plane, and we make the simplifying assumption that the wide-angle lense used to capture the PETS2004 datasets will not have a significant effect on our calculations.

Central to our approach is the concept of a *goal*. We define goals as places where the agent can leave the scene - doors and exits - or to borrow terminology from Ellis and Xu [11] “Long-term Occlusions” or “Border Occlusions”. In the types of pedestrian scene typically subject to surveillance, these are the goals of the agents therein - in a car park, the pedestrian goals are either their cars or the door; in a general pedestrian scene like a shopping mall or the foyer of a research institute, the goals are the exits of the scene, and perhaps some other form of attraction such as an information desk or an ATM machine. The goals for any particular scene can be learned, if enough example footage is present. In the current experiment the PETS2004 dataset (which features a number of short videos) does not provide the body of data required for learning to take place, and so the exit model was hand crafted. This consists simply

of rectangular boxes representing each exit. The obstacle model is similarly hand crafted, but for computational reasons does not have to be regular and is simply stored as a bitmap. Figure 1 shows these models and an image of the scene.

Our central assumption is that people move consistently towards their goal. If there is an obstacle between the agent and their goal, virtual “sub-goals” are constructed in places where the agent might be able to see more of the scene than they currently can - thus, sub-goals are constructed on the edge of obstacles, in places where the agent would be able to see further around the obstacle. From each sub-goal, we compute which goals would be visible if the agent were at that point, and also any sub-sub-goals. And from each sub-sub-goal, we compute which further goals would be visible. Thus for each goal \mathbf{x}_g within the scene we can determine whether that goal is directly visible, or whether it would be visible by turning a corner, or whether it would be visible by turning two corners (in the current implementation we stop computation at two levels of sub-goal analysis). This takes the form of a label - $Label(\mathbf{x}_g)$ - which can have the values V , for those goals which are directly visible; N , for those goals which are not visible at all, and $S1$ or $S2$ for those goals which are accessible via a sub-goal or two.

Indeed, there are four possible relationships between an agent and each goal for each frame, which can be determined from the label of the pixel at the position of the goal $Label(\mathbf{x}_g)$, and the angle ϕ , which is the angle subtended by a line between the position of the goal \mathbf{x}_g , the position of the agent \mathbf{x} , and the agent’s current direction estimate θ . These are:

1. A : The goal is directly visible: $Label(\mathbf{x}_g) = V$; and the agent is heading towards it $1 > \phi > -1$. $g2$ is in this state in Figure 2.
2. D : The goal is directly visible to the agent: $Label(\mathbf{x}_g) = V$; but they are heading away from it: $\phi > 1$ or $\phi < -1$. $g4$ is in this state in Figure 2.
3. N : The goal is not visible to the agent: $Label(\mathbf{x}_g) = N$ (it is on the other side of an obstacle, and is not reachable by means of a sub-goal). $g3$ is in this state in Figure 2.
4. Sn : The goal is visible to the agent, but only via a sub-goal ($S1$) or a sub-sub-goal ($S2$): $Label(\mathbf{x}_g) = Sn$. $g1$ is in state $S1$ in Figure 2.

Given these frame-by-frame classifications for each goal, we can build an idea of how likely - or rather, unlikely - that goal is as an explanation for the trajectory as a whole. This is done by associating a cost with certain state transitions. The diagram in Figure 3 shows costs associated with transitions in the current model.

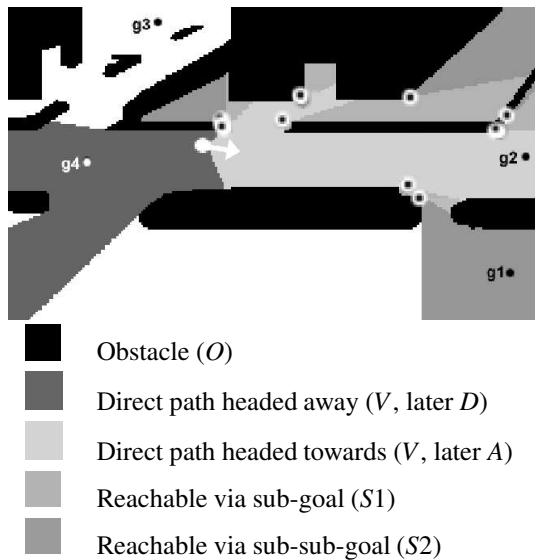


Figure 2. An example of the sub-goal algorithm in action in an outdoor pedestrian scene. The agent is represented by a white dot and a white arrow (corresponding to its velocity vector); white dots with black centres are sub-goals; the obstacle model is shown in black; areas which are not visible (either directly or via a sub-goal or two) in white; areas shaded very light grey represent areas directly visible and headed towards; darker shades of grey represent areas of the scene accessible only via sub-goals or sub-sub-goals; very dark grey represents areas directly visible but not within the angle of vision; g_1 , g_2 , g_3 and g_4 are example goals referred to in section 4.

Applying costs as laid out in Figure 3 provides us with a cost for each goal within the scene, and that cost can be thought of as representing the number of frames in which the agent’s behaviour is inconsistent with travel towards that particular goal. These costs are then divided by the total length of the trajectory to provide a statistic which is comparable across agents. Finally, we need to simplify matters and provide a single cost for each actor. If the system were fully recursive and we were able to work out the final exit for each agent, the cost associated with the final exit would be an alternative measure. However, in the current dataset there are some trajectories which finish whilst the agent is still in view of the camera, making this statistic unreliable. The highest cost or average cost would be inappropriate, as it is possible for perfectly uninteresting trajectories to avoid one or more exits completely; these would have very high

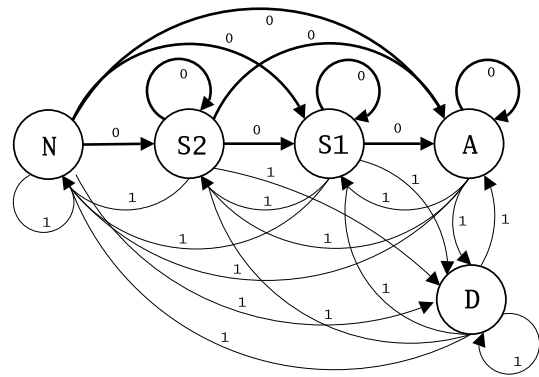


Figure 3. State transition diagram indicating the cost of each transition. Those transitions which are *free* (drawn with thick lines) are those associated with progress towards the particular goal; those with a cost are those associated with movement away from the goal

costs indeed - this would also affect any attempt to use the average cost or some other aggregate score over all goals. Therefore, in the current situation, the best choice for a single cost is the lowest cost.

We can think of the Cost% statistic as representing the percentage of frames in which the agents’ travel was inconsistent with motion towards *their most likely* goal. Cost% scores for the PETS2004 dataset are set out in full in Appendix A. The next section of this paper discusses ways in which our Cost% statistic can be compared with the “ground truth” scores discussed in section 3. It is worth noting that with simple scenes without obstacles, the algorithm just described simplifies to straightest path and the sub-goal mechanism does not make any difference to the output. For several of the simpler trajectories in the PETS2004 dataset this was indeed the case, and the power of the approach would be better demonstrated in a more complicated scene with multiple obstacles. That said, the results on the PETS2004 dataset are still promising and worth discussion.

5 The prototype evaluated

The question we now have to address is how well our prototype results agree with the results of the subjects detailed in Section 3. Firstly, we can calculate Spearman’s Rho - r_s - the correlation coefficient, between the computer generated Cost% statistic and the human subjects, and between the Cost% and the human mean and median (making the assumption that it is appropriate to reify the averages in this way). The correlation statistic r_s and the t-statistic t_s are

Correlation with	r_s	t_s
H1	0.639	3.807
H2	0.679	4.234
H3	0.408	2.05
H4	<i>0.353</i>	1.729
H5	0.507	2.692
H6	0.453	2.329
H7	0.277	1.319
H8	0.292	1.4
H9	<i>0.386</i>	1.917
H10	0.319	1.542
H11	0.47	2.439
H12	0.626	3.676
Median Human	0.607	3.499
Mean Human	0.639	3.810

Table 1. Correlation statistics for the Cost% score against each individual subject and the human averages. Those values which are statistically significant at the 0.05 level are highlighted in boldface, and those which are significant at the 0.1 level but not the 0.05 in italics.

set out in Table 1.

The significance levels for t_s with $n = 23$ are 1.721 at the 10% level and 2.080 at the 5% level. As is clear from Table 1 the correlation with the average human is statistically significant. In Figure 4 we have drawn the graph of Cost% and the median human score by video clip (sorting the video clips by median) it is clear that those clips rated highly by humans generally scored highly on the machine generated statistic as well. This graph also enables us to see the anomalous cases clearly.

The spike labelled A in Figure 4 corresponds to Meet_WalkTogether1.mpg id 2. In this clip, the agent enters from one side, meets someone, changes direction and heads towards an exit - he does not actually exit the scene, but turns around and comes a short way back into the foyer before the video cuts off. We assume that this is an artifact of video editing - it is definitely strange behaviour if not. Given that our system can be thought of as providing a measure of behaviour consistency, it is acceptable for it to pick up on such artifacts.

The spikes labelled B and C in Figure 4 correspond to Meet_WalkSplit.mpg. These trajectories are both quite complicated, involving first moving towards the other agent in the scene and then moving towards an exit (different exits in each case). This is an aspect of our software that we hope to address in future - specifically, making other tracked agents within the scene legitimate goals in themselves. This

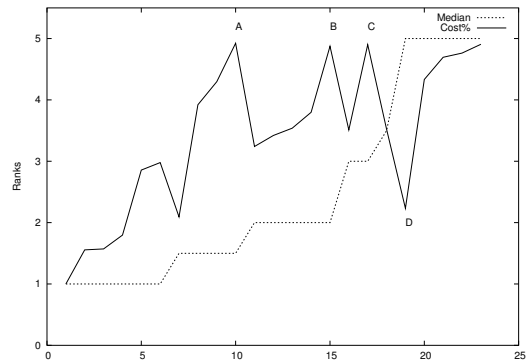


Figure 4. A comparison of the Cost% statistic and the median human rankings. Cost% has been scaled, in order to place both outputs in the same range (1-5). The x-axis values are ordered by median.

is one of the clips highlighted for attention in Section 3 as being a video with high variance amongst human rankers.

The trough labelled D in Figure 4 corresponds to id 6 in Fight_RunAway1.mpg. This agent enters the scene moving quite rapidly, has a play-fight with another agent (lasting just a few seconds), then runs across to the exit opposite. His trajectory is essentially a straight line with a slight kink in the middle, and as our software operates solely on individual trajectories does not pick up on this behaviour. An extension to our system which might cope with this would be to take into account the relative positions of other agents in the scene.

6 Conclusions

Evaluation in Computer Vision should be about more than merely x, y, t . And even when evaluating something as simple as x, y, t , it has been suggested [8] that reliance on just one estimate is unwise. As we produce more complicated systems, performing higher level cognitive tasks than “simple” classification or location, we need more complicated, higher level evaluative techniques. If a system is presented as a general-purpose surveillance system, or an “Interesting Event Detector”, then it should be evaluated as such. Specifically, it should be evaluated in such a way that the opinions and prejudices of the designers cannot affect the evaluation. Evaluation by accident - simply noting that the events detected seem to be odd - is not good enough. Evaluation by actor - by engineering test cases which involve people behaving strangely - is suspect, and evaluation based upon the opinion of the system author is also unsatisfactory. In this paper, we have presented a novel approach
























which uses a group of naïve subjects who together provide a rich background against which the performance of an algorithm can both be measured statistically and compared qualitatively.

The software outlined in this paper is also novel, in that it adopts a high level intentional analysis of what is essentially quite simple behaviour. Previous work consists of analyses of the resultant behaviour: the fact that people follow similar trajectories across a scene [4] is because they have similar goals; the fact that paths can be approximated by trajectory analysis [6] is because paths join two goals. This work attempts instead to analyse the cause of the behaviour – the goals – directly. The initial results presented here are promising, and show that in principle such an analysis could be used in a practical situation to provide a filter on surveillance data.

References

- [1] Clarke G.M. and Cooke D. *A basic course in statistics: third edition*. Edward Arnold, London, 1992, 3 edition.
- [2] Dennett D.C. ‘True believers: The intentional strategy and why it works.’ In: W.G. Lycan (editor), *Mind and Cognition: A Reader*, pp. 150–167. Blackwell, Cambridge, MA, 1990.
- [3] Jan T., Piccardi M. and Hintz T. ‘Detection of suspicious pedestrian behavior using modified probabilistic neural network.’ In: *Proc. of Image and Vision Computing*, pp. 237–241. Auckland, New Zealand, 2002.
- [4] Johnson N. and Hogg D.C. ‘Learning the distribution of object trajectories for event recognition.’ *Image and Vision Computing*, Vol 14(8), pp. 609–615, 1996.
- [5] Magee D.R. ‘Tracking multiple vehicles using foreground, background and shape models.’ *Image and Vision Computing*, Vol 22, pp. 143–155, 2004.
- [6] Makris D. and Ellis T. ‘Spatial and probabilistic modelling of pedestrian behaviour.’ In: *Proc. British Machine Vision Conference*, pp. 557–566. Cardiff, UK, 2002.
- [7] Morris R.J. and Hogg D.C. ‘Statistical models of object interaction.’ *International Journal of Computer Vision*, Vol 37(2), pp. 209–215, 2000.
- [8] Needham C.J. and Boyle R.D. ‘Performance evaluation metrics and statistics for positional tracker evaluation.’ In: *Proc. International Conference on Computer Vision Systems*. Austria, 2003.
- [9] Stauffer C. and Grimson E. ‘Learning patterns of activity using real-time tracking.’ *Pattern Analysis and Machine Intelligence*, Vol 22(8), pp. 747–757, 2000.
- [10] Troscianko T., Holmes A., Stillman J., Mirmehdi M., Wright D. and Wilson A. ‘What happens next? the predictability of natural behaviour viewed through CCTV cameras.’ *Perception*, Vol 33(1), pp. 87–101, 2004.
- [11] Xu M. and Ellis T. ‘Partial observation vs. blind tracking through occlusion.’ In: *Proc. British Machine Vision Conference*, pp. 777–786. Cardiff, UK, 2002.

A Table of agents and results

Filename	Id	Image	Description	Cost% (Scaled)	Human mean	Human SD	Human Median
Walk1.mpg	0		Walks in, waves at camera, goes back through same door	37.8 (3.52)	3.33	1.07	3.5
Walk1.mpg	1		Walks slowly across scene	8.57 (1.57)	1.25	0.45	1
Walk3.mpg	1		Walks out, turns around, walks back through same door	38.1 (3.54)	2.08	1.16	2
Walk3.mpg	2		Walks slowly across scene	16.4 (2.09)	1.58	0.67	1.5
Meet_WalkTogether1.mpg	1		Enters, meets, shakes hands, changes direction, exits	49.46 (4.3)	1.92	1.16	1.5
Meet_WalkTogether1.mpg	2		Enters, meets, shakes hands, changes direction, exits	43.82 (3.92)	1.92	1.16	1.5
Rest_FallOnFloor.mpg	2		Enters in a wobbly fashion, falls over, gets up and leaves	58.6 (4.91)	4.67	0.65	5
Rest_SlumpOnFloor.mpg	0		Leaves scene, re-enters, slumps on floor, leaves scene again	58.52 (4.9)	3	1.48	3
Meet_WalkSplit.mpg	1		Walks towards person, shakes hands, turns, leaves scene	58.13 (4.88)	2.5	1.62	2
Meet_WalkSplit.mpg	3		Walks towards person, shakes hands, turns, leaves scene	36.31 (3.42)	2.33	1.56	2
Meet_Crowd.mpg	0		Walks in straight line across scene	8.33 (1.56)	1.33	0.78	1
Meet_Crowd.mpg	1		Walks in straight line across scene	11.95 (1.8)	1.5	0.67	1
Meet_Crowd.mpg	2		Walks in relatively straight line across scene	27.86 (2.86)	1.5	1	1
Meet_Crowd.mpg	3		Walks in relatively straight line across scene	29.67 (2.98)	1.58	1.16	1
Fight_RunAway1.mpg	6		Walks in, fights, runs out	18.5 (2.23)	4.75	0.62	5
Fight_RunAway1.mpg	7		Hangs around, Walks in, fights, runs out	56.44 (4.76)	4.67	0.65	5
Fight_OneManDown.mpg	4		Walks in, fights, runs in circles, runs out	50 (4.33)	4.75	0.62	5
Fight_OneManDown.mpg	5		Enters, gets fought with and knocked over, leaves	55.43 (4.7)	4.33	1.15	5
Browse_WhileWaiting2.mpg	0		Wanders aimlessly	41.97 (3.8)	2.08	0.9	2
Browse4.mpg	1		Wanders aimlessly	33.62 (3.24)	1.92	0.9	2
Browse4.mpg	2		Walks directly across scene	0 (1)	1.17	0.39	1
Browse2.mpg	1		Walks in, waves at camera, leaves	37.66 (3.51)	2.75	0.97	3
Browse2.mpg	3		Wanders towards bookshelves, browses, leaves	58.87 (4.92)	1.67	0.78	1.5

Ontology-guided Training of Bayesian Networks for High-level Analysis in Visual Surveillance

Christopher Town
University of Cambridge Computer Laboratory
15 JJ Thomson Avenue, Cambridge CB3 0FD, UK
cpt23@cam.ac.uk

Abstract

Ontologies offer a convenient means of encoding hierarchical knowledge in terms of entities, attributes and relationships which may be used to characterise a given domain. This paper shows how modern techniques for structure and parameter learning in Bayesian networks can be applied to ground truth data to automatically generate effective high-level state and event recognition mechanisms for video analysis. The manual annotations are used to instantiate visual tracking appearance modelling modules in order to augment the ground truth with states which may be directly estimated by means of video object tracking. Both the structure and parameters of Bayesian networks are then trained to infer high-level object and scenario properties on the basis of the visual properties and an ontology of states, roles, situations and scenarios which is easily derived from the original ground truth schema.

1 Introduction and Overview

As visual surveillance applications become increasingly prevalent, automated techniques for the detection and analysis of objects and events in video data are gaining prominence. It is likely that an increased reliance on such methods will bring about important changes to the way that research and development in relevant fields of computer vision is conducted and assessed. The case of vision-based biometrics in recent years offers some insights into likely developments in other areas of computer vision which are pertinent to the booming security industry.

Increased commercial and government interest in automated visual surveillance is not only resulting in increased emphasis on performance analysis and evaluation standards, but also fundamentally affects the way such research is conducted. Rather than focussing on the particular merits and intellectual importance of particular vision algorithms and representations, developers of visual surveillance systems will be confronted with largely externally imposed specifications of what information such systems are

to extract from available video footage.

This paper presents a study of how the process of creating recognition systems for high-level analysis of surveillance data can be largely automated, provided sufficient quantities of ground truth data which has been annotated with descriptors from the desired analysis specification are available. Such a specification may usefully be regarded as an ontology which provides a prior description of the application domain in terms of those entities, states, events and relationships which are deemed to be of interest. The hierarchical organisation and relational constraints imposed by such an ontology may then be used to guide the design of a complete visual analysis system.

In this paper, video sequences and ground truth from the CAVIAR project ¹were used to define an ontology of visual content descriptors arranged in a hierarchy of scenarios, situations, roles, states, and visual properties. The latter category was defined by choosing object attributes such as translational speed and appearance change which could easily be extracted by means of a pre-existing blob tracking and appearance modelling framework. The CAVIAR training data was then re-labelled with this extended set of descriptors by instantiating the tracking framework with the individual objects in the ground truth and computing the selected visual attributes for all frames in the sequences.

The resulting data was then used to learn both the structure and parameters of Bayesian networks for high-level analysis. Evaluations were performed to assess how easily the categories of the ontology could be inferred on the basis of the chosen visual features and on the basis of preceding layers in the hierarchy. The former allows one to assess the (in)adequacies of a set of given visual content extraction and representation methods, which is an important tool in designing the computer vision components of a surveillance system in order to maximise their utility for high-level inference in light of the domain ontology. Conversely, one can use the probabilistic scoring methods applicable to Bayesian networks to evaluate how well-defined

¹EC Funded CAVIAR project/IST 2001 37540, see <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>

e.g. the pre-defined set of situation descriptors are in terms of the labels for object roles and states which appear in the ground truth. From this one can draw conclusions as to the semantic and syntactic self-consistency and completeness of the ground truth schema and the extent to which the manual annotations are consistent with the assumptions incorporated into the ontology. Such results then allow one to iteratively refine both the ontology and the underlying visual content extraction methods in order to arrive at a complete system which meets its requirements. They may also serve as a valuable basis for comparison of alternative approaches to solving a particular set of tasks.

2 Related Work

In recent years, Bayesian graphical models have played a prominent role in visual surveillance tasks [3, 2]. They offer many advantages for tracking tasks such as incorporation of prior knowledge and good modelling ability to represent the dynamic dependencies between parameters involved in a visual interpretation.

In [15] tracking of a person’s head and hands is performed using a Bayesian network which deduces the body part positions by fusing colour, motion and coarse intensity measurements with context dependent semantics. Tracking on the basis of multiple sources of information is also demonstrated by Choudhury et al [4] who present a system which fuses auditory and visual cues for speaker detection. As an improvement to earlier work by the same authors, the fusion is performed by a Dynamic Bayesian network whose structure was learned by means of a modified AdaBoost algorithm. [7] uses a relatively sophisticated mixture of Gaussians model for the background and a Bayesian network to reason about the state of tracked objects. Objects are represented by a combination of predictive features. A method for recognising complex multi-agent action is presented in [9]. Belief networks, including some automatically generated from the temporal structure descriptions of compound actions, are used to probabilistically represent and infer the goals of individual agents and integrate these in time from visual evidence.

Ontologies offer a way of incorporating structured syntactic and semantic knowledge into visual analysis frameworks [16]. [11] presents an ontology of actions represented as states and state transitions hierarchically organised from most general to most specific (atomic). The paper stresses the point that language-based description of video requires one to establish correspondences between concepts and features extracted from video images. Appropriate syntactic components such as verbs and object labels can then be determined to generate a natural language sentence from a video sequence. In [5], an architecture for perceptual computing is presented which integrates different visual pro-

cessing routines in the shape of a “federation of process” where bottom-up data is fused with top-down information about the user’s context and roles based on an ontology of roles and relations. The use of ontologies for visual content analysis also features in [13] which describes an event recognition language for video. Events can be hierarchical composites of simpler primitive events defined by various temporal relationships over object movements.

3 Visual Analysis and Tracking

This section provides an overview of the visual tracking and object modelling methods which were applied to the CAVIAR training sequences.

3.1 Background modelling and foreground detection

The system maintains a background model and foreground motion history (obtained by frame differencing) which are adapted over time using an exponential rate of decay to determine the decreasing influence of previous frames im_{i-1} in the history:

$$M_i = \beta * |im_i - im_{i-1}| + (1 - \beta) * M_{i-1} \quad (1)$$

$$\text{where } \beta = 1 - e^{-1/\lambda_M} \quad (2)$$

The motion history M_i is used to identify a background image bim_i of pixels undergoing sufficiently slow change (i.e. due to noise or gradual changes in lighting conditions) which can then be used to reliably update the background model B_i and estimate its variance:

$$B_i = \alpha * bim_i + (1 - \alpha) * B_{i-1}; \quad B_0 = im_0 \quad (3)$$

$$\text{where } bim_i = |im_i - M_i| < \tau; \quad \alpha = 1 - e^{-1/\lambda_B} \quad (4)$$

Pixels are deemed to be part of the non-static foreground if they exceed a difference threshold which is a multiple of the background variance σ_i^B . Furthermore, pixels which are thus classified as outliers $B_i^{outlier}$ from the background process are not labelled as foreground if they are likely to be part of (moving) shadows as determined by the DNMI algorithm described in [14]:

$$B_i^{outlier} = |im_i - B_{i-1}| > k * \sigma_i^B \quad (5)$$

$$F_i(x, y) = B_i^{outlier}(x, y) \wedge \neg shadow_i(x, y) \quad (6)$$

3.2 Blob tracking and adaptive appearance modelling

Foreground pixels are clustered using connected components analysis to identify moving regions (“blobs”). These



Figure 1: Tracking results (from left to right): Original frame; Model of the background variances; Results of background subtraction; Detected blobs after morphological operations; Resulting tracked objects (outlined in green) with ground truth data and results shown in yellow.



Figure 2: Sample frames from a surveillance sequence showing the objects tracked by the blobtracking framework outlined in green.

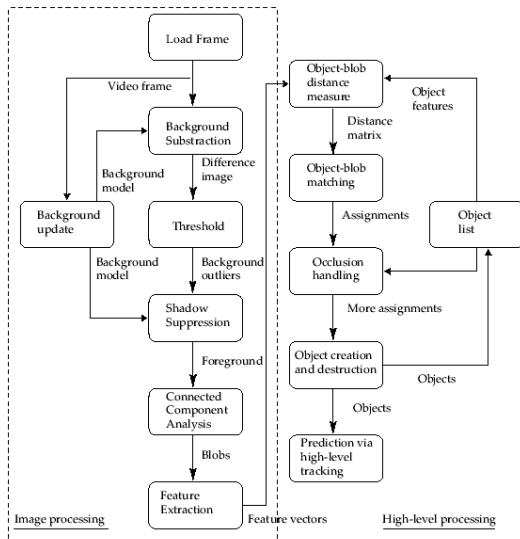


Figure 3: Overview of the blob tracking method.

are then parameterised using shape (bounding box, centre of gravity, major axis orientation) and colour measures. Colour appearance is modelled by means of both an RGB histogram and a Gaussian mixture model in hue-saturation space. In a similar manner to [12], re-estimation of the mixture parameters is performed selectively by weighting frame contributions with the blob's colour log-likelihood under the model.

Blob positions are tracked using a Kalman filter or Condensation tracker with a second order motion model.

Tracked objects are matched to detected blobs using a weighted dissimilarity metric which takes into account differences in predicted object location vs blob location and changes in shape and appearance. Histograms are compared using the EMD measure and provide a useful measure of short-term appearance variation while the Gaussian mixture models a more stable and long-term representation of appearance which is useful e.g. for identity maintenance across object occlusions. Figure 3 summarises the blob tracking framework.

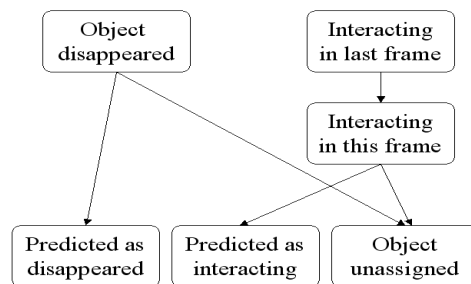


Figure 4: Bayesian network for occlusion reasoning and prediction of object interactions.

3.3 Occlusion reasoning

To make tracking more robust, the object to blob assignment stage features a Bayesian network for reasoning about occlusions and object interactions (see figure 4) based on observed or predicted overlap of object bounding boxes and failures of object assignment.

4 High-level Analysis

4.1 Data Set

The aforementioned CAVIAR data comprises 28 sequences taken by a surveillance camera in the entrance lobby of the INRIA Rhone-Alpes research laboratory in Montbonnot, France. They consist of six scenarios of actors performing different activities such as walking around, browsing information displays, sitting down, meeting one another and splitting apart, abandoning objects, fighting and running away.

Each sequence has been annotated with the spatial location, angle of rotation and extent of bounding boxes around individuals and groups of people. Each such box is assigned a numerical label to identify it in subsequent frames and a list of properties (see section 4.2). These consist of binary states and probabilities (which are effectively either 1 or 0 in the ground truth) for the events, scenarios, situations and roles which are deemed to best describe the behavioural and situational context of the given person or group. Groups have a different set of descriptors from individuals and are defined in terms of their constituent individuals and the smallest bounding box which encompasses them.

However, this paper limits its scope to the individuals and their descriptors. This is because high-level analysis on the basis of individual state and actions was found sufficient to investigate the use of the extended ground truth ontology for the creation and evaluation of Bayesian inference networks. Moreover, the criteria for grouping in the data were found to be somewhat ill-defined, for example often people are grouped together at a particular frame on the basis that they will interact in some way several seconds later in the sequence. Furthermore, the training of Bayesian networks was limited to the subset of the ground truth data made available for such purposes as part of the 2004 IEEE Workshop on Performance Evaluation in Tracking and Surveillance (PETS2004). This subset encompasses half the sequences for a total of 17374 annotations of individual people.

4.2 Domain Ontology

Ontologies encode the relational structure of concepts which one can use to describe and reason about aspects of the world. Ontology is the theory of objects in terms of the criteria which allow one to distinguish between different types of objects and the relations, dependencies, and properties through which they may be described.

The CAVIAR annotations can naturally be organised into a hierarchical ontology as shown in table 1. This arrangement offers guidance for the design of Bayesian inference networks. For example, one would expect an individual's

state to depend primarily on their current role, their current role to depend on the situation they are facing, and their situation to depend on the scenario in which they are participating. These broad hierarchical relationships can be used as a structural prior for the training of Bayesian networks as described in the next section.

In order to study the extent to which elements of the ontology may be inferred on the basis of automatically extracted visual information, one needs to augment the ontology with appropriate descriptors that can be computed from raw sequence data using computer vision techniques. Using the tracking and appearance modelling framework described in section 3, a set of such descriptors was defined in order to form the bottom layer of the ontology:

cvSpeed: Current object speed (as estimated from the tracker) in terms of the estimated displacement of the object's bounding box expressed in pixels per second (calculated per frame and normalised using the camera's frame rate)

cvFlow: Amortised flow measure representing a recent history of the object's motion:

$$fl_t = \gamma * (|cx_t - cx_{t-1}| + |cy_t - cy_{t-1}|) + (1 - \gamma) * fl_{t-1}$$

where $\gamma = 1 - e^{-1/3}$ and (cx_t, cy_t) = object centre of gravity at time t.

cvLifetime: Whether or not the object has been newly instantiated or is about to be terminated due to no longer being detectable in the image (in the absence of any other explanation offered by the occlusion reasoning).

cvHistdist: Measure of inter-frame appearance variation calculated as the weighted sum of histogram EMD measure and Gaussian mixture model likelihood. The weight given to histogram distance is increased if the object is moving rapidly.

cvOccstat: Whether the object is estimated to be unoccluded, occluded, or to have disappeared based on the occlusion reasoning network shown in figure 4.

These visual descriptors are not claimed to constitute the best choice for the analysis task at hand. They are merely properties of tracked objects which can be simply and robustly defined using the techniques described in section 3 and offer a reasonable basis for studying the requirements for low-level analysis mechanisms which result from the pre-defined ontology of higher-level terms. Additional object tracking and analysis modules could easily be integrated into the existing framework to provide additional information for terms which are currently hard or impossible to infer (e.g. the detection of left objects and other roles which require knowledge of multi-object interactions). The principle goal of the investigation was to study the suitability of the ontology and ground truth for automated construction of Bayesian inference networks independent of the

performance of any particular tracking methods. Thus the available annotations were used to initialise objects maintained by the visual tracking framework in order to then augment the ground truth for each individual with the resulting visual descriptors listed above.

<i>Scenario</i>	A description of an individual's overall context.
scBSC	Browsing scenario
scIM	Immobile scenario
scWG	Walking scenario
scDD	Drop-down scenario
<i>Situation</i>	The situation in which the individual is participating.
siM	Moving situation
siIS	Inactive situation
siBSI	Browsing situation
<i>Role</i>	The individual's role in the current situation.
rF	Fighter role
rBR	Browser role
rLV	Left victim role
rLG	Leaving group role
rWR	Walker role
rLO	Left object role
<i>State</i>	The individual's current attributes.
tAP	Appear
tDI	Disappear
tO	Occluded
tIN	Inactive: visible but not moving
tAC	Active: visible, moving but not translating across the image
tWK	Walking: visible, moving, translating across the image slowly
tR	Running: visible, moving, translating across the image quickly

Table 1: **Ontology**

4.3 Learning Bayesian Network Structure and Parameters

There are a variety of methods for learning both the parameters and structure of Bayesian networks from data, see [8, 10] for an overview and further references. In this paper, the goal was to learn the structure and parameters of a static directed Bayesian network given fully observed data, i.e. the values of all nodes are known in each case from the ground truth (augmented as required with the information gathered by the computer vision techniques). All nodes were represented as discrete states, with the nodes cvSpeed, cvFlow and cvHistdist quantised to 5 different values. Nodes cvLifetime and cvOccstat have 3 states while all other variables are binary.

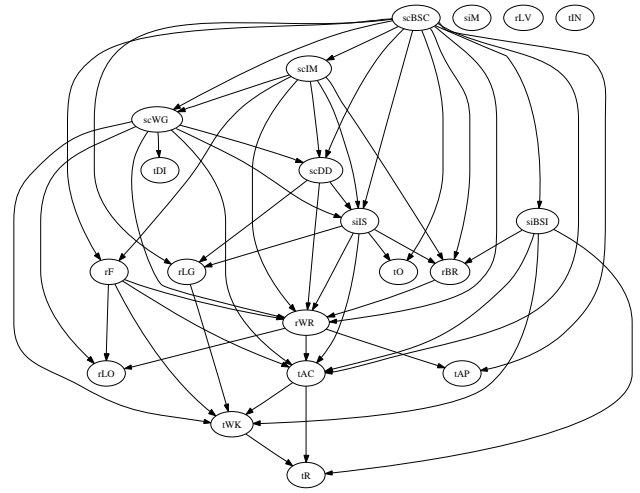


Figure 5: **Bayesian network structure trained using the K2 algorithm applied to the original ground truth schema.**

Learning of networks structure was performed using the K2 algorithm (Cooper and Herskovits, 1992). Other techniques such as Markov Chain Monte Carlo (MCMC) were largely found to provide inferior results and required many thousands of iterations to converge to a solution. Furthermore, the K2 method directly benefits from the prior structural information contained in the ontology. The K2 algorithm is a greedy search technique which starts from an empty network but with an initial ordering of the nodes. A Bayesian network is then created iteratively by adding a directed arc to a given node from that parent node whose addition most increases the score of the resulting graph structure. This process terminates as soon as none of the possible additions results in an increased score. Scoring was performed using the marginal likelihood of the model. Once the network structure has been trained, parameters can be estimated easily using maximum likelihood estimation using Dirichlet priors (pseudo-counts).

Figure 5 shows the Bayesian network which results from training a Bayesian network using the K2 method and the data from the CAVIAR ontology in table 1. As can be seen from the graph, several of the nodes are not definable in terms of the other nodes and the overall connectivity seems somewhat ad-hoc. By contrast, figure 6 shows a network which was trained using the full ontology and a structural prior specifying that nodes which are part of the same semantic level in the ontology (e.g. all situation labels) should be treated as equivalent in terms of the ordering of nodes. The resulting network structure encompasses many of the causal relationships one would expect from the semantics and shows that there are strong dependencies between the

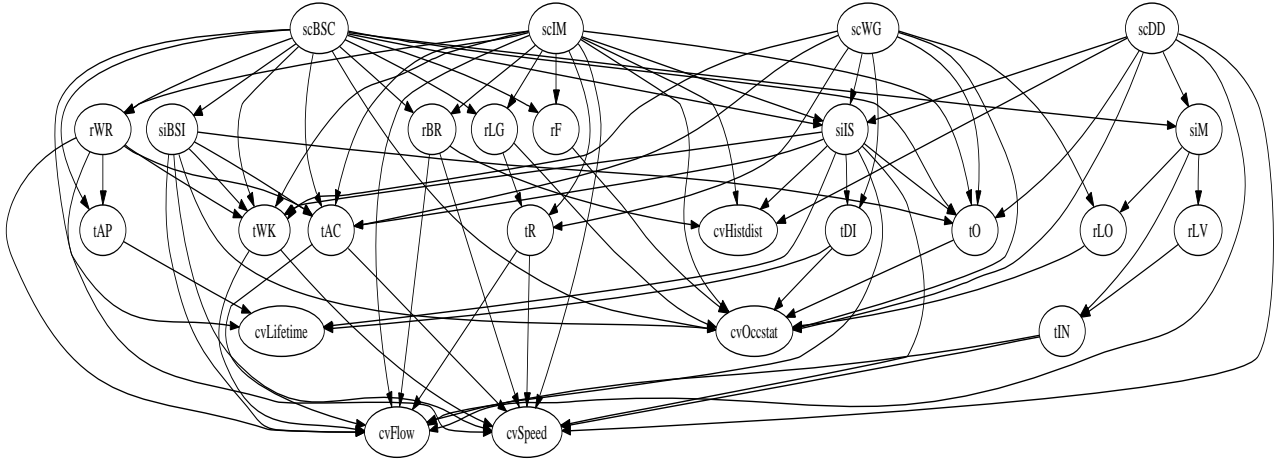


Figure 6: Bayesian network structure trained using the K2 algorithm with a structural prior and using the extended ontology.

computer vision derived terms and the states and roles in particular.

5 Performance Analysis

5.1 Visual Tracking Accuracy

Figure 1 shows results from the visual tracking for one frame of the CAVIAR sequences, while figure 2 shows results of vision-based tracking over several frames of a sequence. A range of performance evaluation metrics have been proposed to assess the quality of visual object tracking [6, 1]. Important factors include the number of *True positives* N_{tp} ('hits'): the number of visually tracked objects confirmed by the ground truth

False positives N_{fp} ('duds'): the number of objects not matching the ground truth

False negatives N_{fn} ('misses', 'false rejects'): the number of ground truth objects not matched by the tracking

True negatives N_{tn} ('true rejects'): the number of erroneous observations rejected by the visual tracking

It follows that the number of objects in the ground truth, N , can be expressed as $N = N_{tp} + N_{fn}$. One can then define metrics such as

Detection rate (sensitivity) $DR = N_{tp}/N$

False alarm rate $FR = N_{fp}/(N_{fp} + N)$

A mean distance-from-track measure TD is also computed. For each object in the ground truth, this is the average distance across the sequence of the normalised (in terms of the maximum possible distance across the image) Euclidean distance of object and matching observation centres of gravity.

Figure 7 shows results of DR , FR and TD calculated for the visual object tracking framework in section 3 over the approximately 13000 frames of CAVIAR sequences used purely for evaluation.

5.2 Performance of High-level Analysis

One useful measure of comparing Bayesian networks is the likelihood of the training data. Assuming independence of the training cases, the log-likelihood of the training set $D = \{D_1, \dots, D_M\}$ can be computed over all N nodes of the network:

$$LL = \log \prod_{m=1}^M \Pr(D_m|G) = \sum_{i=1}^N \sum_{m=1}^M \log P(X_i|Pa(X_i), D_m)$$

where $Pa(X_i)$ are the parents of node X_i . In order to arrive at a simple statistic for comparison of different networks, one can compute the average likelihood of the data per node:

$$L^{\sim} = e^{LL/(M*N)}$$

Using this measure, the network in figure 5 achieves a score of $L^{\sim} = 0.729$, while the network in figure 6 scores $L^{\sim} = 0.888$. Its likelihood scores for each node are shown in table 2.

These scores indicate how well the conditional distribution at each node represents the training data for that node. For example, the values derived using computer vision have no support in the ground truth and their derivation therefore involves a greater amount of uncertainty, resulting in a lower likelihood score. Even within the nodes corresponding to terms in the ground truth, some such as "rWR", "scWG", "scIM" and "scBSC" exhibit smaller likelihood

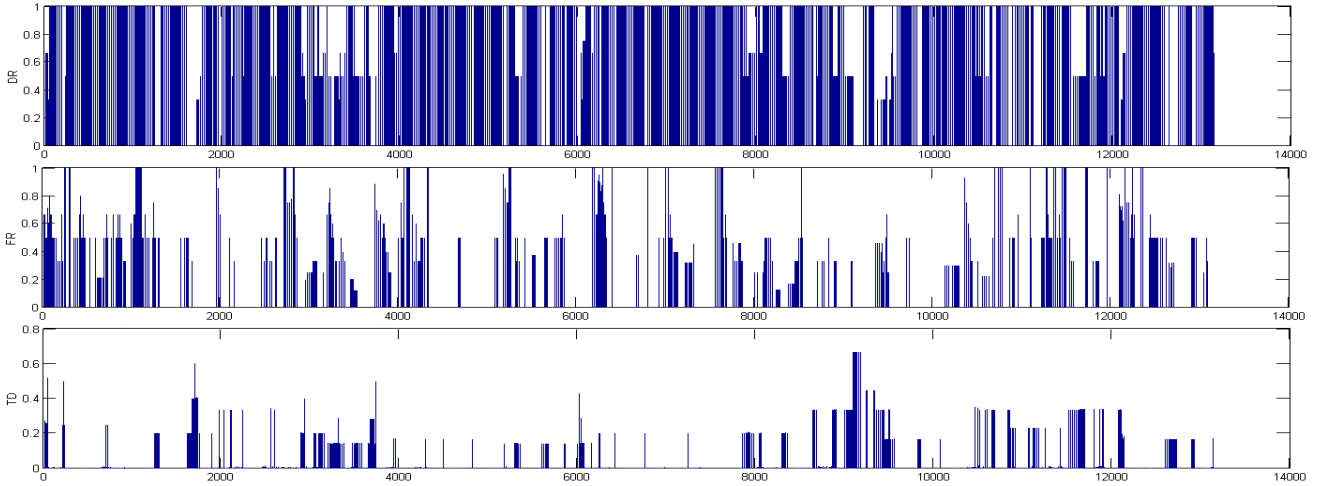


Figure 7: Performance results of the visual tracking for about 13000 frames from the CAVIAR sequences. Top: detection rate DR . Middle: false positive rate FR . Bottom: Average distance from track TD .

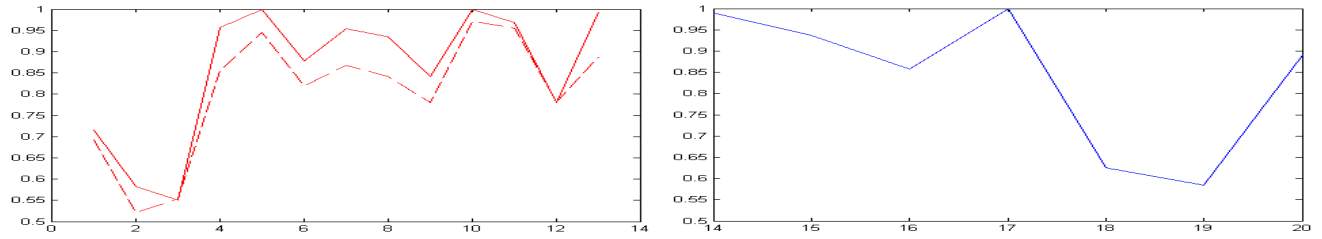


Figure 8: Expected detection rates for the nodes in the Bayesian network in figure 6 given different evidence (see section 5.2). The values on the x-axis correspond to nodes in the network when enumerated in topological order as in table 1 (i.e. $scBSC=1$, $scIM=2$, ..., $cvOccstat=25$).

values. This may be due to inconsistencies in the ground truth or failure of the Bayesian network to fully model their interdependencies.

Perhaps more useful conclusions can be drawn from an analysis of posterior probability scores. Using a suitable inference algorithm such as the Junction tree method, one can compute the marginal probabilities for each possible state of a given node for the available evidence. By computing the marginal probability of a given node for its “correct state” (i.e. that provided in the ground truth data), one can compute the expected detection rate for a given term in the ontology as the mean of these marginal scores over the evaluation data set. This allows one to quantify the value of adding and removing nodes and edges in the Bayesian network.

The dashed red line on the left of figure 8 indicates how well defined the upper-level terms are given only the value of the states (tAP, tDI, tO, tIN, tAC, tWK, tR) as evidence for the network in figure 6. By comparison, the solid red line shows expected correct detection rates if the Bayesian network is given both the values of the states in the ground

truth and that of the computer vision derived nodes as evidence. It can be seen that adding the latter improves the performance of higher-level inference. The blue line on the right side of figure 8 indicates the expected correct inferences for each state given the values of the computer vision nodes only.

6 Summary and Conclusions

This paper demonstrates the value of using ontologies to build working high level vision systems. It is shown how such ontologies can be derived from an existing ground truth schema and a set of visual tracking methods. Together with a set of annotations, such an ontology can then be used to derive training data and prior structural information for automated learning of both the connectivity and parameters of Bayesian networks for high-level inference. Provided sufficient amounts of such data are available, this process is reasonably robust to human errors in the annotations and inadequacies in the automatically extracted visual content descriptions. Performance analysis of the resulting networks

Node	Likelihood	Node	Likelihood
scBSC	0.604	tAP	0.959
scIM	0.534	tDI	0.980
scWG	0.505	tO	0.978
scDD	0.867	tIN	0.999
siM	0.999	tAC	0.728
siIS	0.832	tWK	0.718
siBSI	0.902	tR	0.937
rF	0.904	cvSpeed	0.266
rBR	0.845	cvFlow	0.278
rLV	0.999	cvLifetime	0.993
rLG	0.927	cvHistdist	0.392
rWR	0.695	cvOccstat	0.555
rLO	0.985		

Table 2: Likelihood scores

and the quality of the visual tracking provides a useful basis for comparison of alternative schemes and methods. It allows alternative ontologies to be compared for their self-consistency and realisability in terms of the different visual detection and tracking modules.

It was shown how this can be done using the CAVIAR project’s ground truth annotated sequences. Using the annotations and a robust but straightforward blob-based tracking framework, one can gather co-occurrence statistics for the terms in the ontology and use these together with the hierarchical relationships implied by it (event, scenario, situation, role, state) to build Bayesian nets for high-level analysis tasks in the chosen visual surveillance domain.

Acknowledgements

The author gratefully acknowledges financial assistance from AT&T Labs Research and the Royal Commission for the Exhibition of 1851.

References

- [1] J. Black, T. Ellis, and P. Rosin. A novel method for video tracking performance evaluation. In *Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2003.
- [2] H. Buxton. Learning and understanding dynamic scene activity. In *Proc. ECCV Generative Model Based Vision Workshop*, 2002.
- [3] H. Buxton and S. Gong. Advanced visual surveillance using bayesian networks. In *Proc. International Conference on Computer Vision*, 1995.
- [4] T. Choudhury, J. Rehg, V. Pavlovic, and A. Pentland. Boosting and structure learning in dynamic bayesian networks for

audio-visual speaker detection. In *Proc. Int. Conference on Pattern Recognition*, 2002.

- [5] J. Crowley, J. Coutaz, G. Rey, and P. Reignier. Perceptual components for context aware computing. In *Proc. Ubicomp 2002*, 2002.
- [6] T. Ellis. Performance evaluation of tracking and surveillance. In *IEEE Workshop on Performance Evaluation in Tracking and Surveillance*, 2002.
- [7] T. Ellis and M. Xu. Object detection and tracking in an open and dynamic world. In *IEEE Workshop on Performance Evaluation in Tracking and Surveillance*, 2001.
- [8] D. Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, 1996.
- [9] S. Intille and A. Bobick. Representation and visual recognition of complex, multi-agent actions using belief networks. In *IEEE Workshop on the Interpretation of Visual Motion*, 1998.
- [10] M. Jordan, editor. *Learning in Graphical Models*. MIT Press, 1999.
- [11] A. Kojima, T. Tamura, and K. Fukunaga. Natural language description of human activities from video images based on concept hierarchy of actions. *Int. Journal of Computer Vision* (to appear), 2002.
- [12] S. McKenna, Y. Raja, and S. Gong. Object tracking using adaptive color mixture models. In *Proc. Asian Conference on Computer Vision*, pages 615–622, 1998.
- [13] R. Nevatia, T. Zhao, and S. Hongeng. Hierarchical language-based representation of events in video streams. In *Proc. IEEE Workshop on Event Mining*, 2003.
- [14] A. Prati, I. Mikic, R. Cucchiara, and M. Trivedi. Comparative evaluation of moving shadow detection algorithms. In *Proc. Workshop on Empirical Evaluation Methods in Computer Vision*, 2001.
- [15] J. Sherrah and S. Gong. Tracking discontinuous motion using bayesian inference. In *Proc. European Conference on Computer Vision*, pages 150–166, 2000.
- [16] C.P. Town and D.A. Sinclair. A self-referential perceptual inference framework for video interpretation. In *Proc. Int. Conference on Vision Systems*, 2003.

On-line Tracking Groups of Pedestrians with Bayesian Networks^{*}

Pedro M. Jorge
ISEL / ISR
pmj@isel.ipl.pt

Jorge S. Marques
IST / ISR
jsm@isr.ist.utl.pt

Arnaldo J. Abrantes
ISEL
aja@isel.ipl.pt

Abstract

A video tracker should be able to track multiple objects in the presence of occlusions. This is a difficult task since there is not enough information during the occlusion time intervals. This paper proposes a tracking system which solves these difficulties, allowing a long term tracking of multiple interacting objects. First active regions are tracked using simple image analysis techniques. Then, a Bayesian network is used to label/recognize all the detected trajectories, taking into account the interaction among multiple objects. Experimental results are provided to assess the proposed algorithm with PETS video sequences.

1. Introduction

Video surveillance systems aim to detect, track and classify human activities from video sequences captured by single or multiple cameras. Several systems have been recently proposed to perform all or some of these tasks (e.g., see [13, 16, 6, 11, 14]).

The problem becomes difficult when there is an overlap of several objects in the image or the occlusion of some of the objects to be tracked. In such cases it is not possible to track each moving object all the time and inference strategies must be devised in order to recover tracking when enough information becomes available. Fig. 1 shows the superposition of multiple objects with partial occlusion of some of them and their separation into isolated active regions.

Several methods have been used to recover from object superposition and occlusion as well as detection errors (mis-detection and false alarms). Some of them are modified versions of the methods used in the tracking of point targets in clutter e.g., nearest neighbor tracker [4], the JPDAF [2], the multiple hypothesis tree or particle filtering [3, 8]. The two problems (target tracking and video objects tracking)

^{*}This work was supported by FEDER and FCT under project LTT (POSI 37844/01).

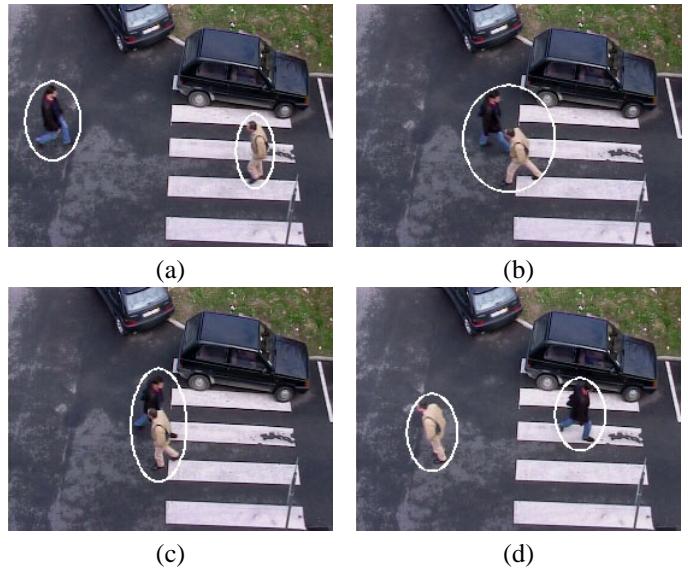


Figure 1. Occlusion example: merge & split

are very different however and they should be tackled with different techniques.

This paper describes a new method which has been developed by the authors which formulates object tracking in video sequences as a labeling problem. It is often simple to detect and track moving objects in video sequences when they are isolated. This can be efficiently done using simple image analysis techniques (e.g., background subtraction). When the object is occluded by other objects or by the background it is usually not possible to separately track. All we can expect to achieve most of the time is to track the group of objects. However, when the object becomes isolated again we should be able to recognize it and recover the track. How can we perform these tasks using all the available information (e.g., information about the interaction among multiple objects, visual characteristics of the objects to be tracked, physical laws)?

This paper described a solution based on Bayesian networks which addresses all these problems. Object tracking is decomposed in two steps: tracking of active regions and

labeling/recognition of detected trajectories. The labeling task is formulated as an inference problem which is solved by resorting to the use of Bayesian networks which provide useful models for objects interaction and occlusion.

This paper is organized as follow. Section 2 presents an overview of the Bayesian Network tracker. The low level processing is described in section 3 and the generation of the Bayesian network is presented in section 4. Section 5 deals with computation and implementation aspects of the proposed tracker. Section 6 described experimental results and section 7 presents the conclusions.

2. Bayesian Network Tracker

The Bayesian network (BN) tracker consists of two steps. The first step tries to track all the active regions in the video stream. These regions are either isolated objects or groups of objects. The output of the first step is a set of trajectories (see [1, 10] for details).

When the objects overlap in the image domain or when they are occluded, the methods used in first step are not able to reliably associate active regions detected in consecutive frames and the trajectories are broken. A labeling operation is then performed in the second step in order to recognize trajectories of the same object.

Furthermore, we wish to perform a consistent track of object groups i.e., we want to know if a given region is a group, to estimate the group trajectory and to know which objects are in the group.

The labeling operation is performed using a Bayesian network. The Bayesian network plays several roles. It models the interaction among the trajectories of different objects and with the background. Second it provides a consistent labeling which accounts for known restrictions (e.g., in object occlusions, group merging and splitting). Finally, it allows to update the labeling decisions every time new information is available. Fig. 2 shows the output of the two steps for the example of Fig. 1

Let $s_k, k = 1, \dots, N$ be the set of segments detected by the low level operations of step 1 (see Fig. 2a). In order to interpret this data, a label x_k is assigned to each segment s_k . Each label identifies all the objects in the segment i.e., if the segment corresponds to a single object, the label is the object identifier. If the segment corresponds to a group, the label is a set of identifiers of all the objects inside the group. The key issue is how to estimate the labels from the information available in the video stream?

Three information sources should be explored. First, labels should be compatible with physical restrictions (e.g., the same object can not be in two places at the same time, the objects velocities are bounded). Second there is prior information which should be used e.g., if the trajectories of two isolated meet a given point and a new trajectory is cre-

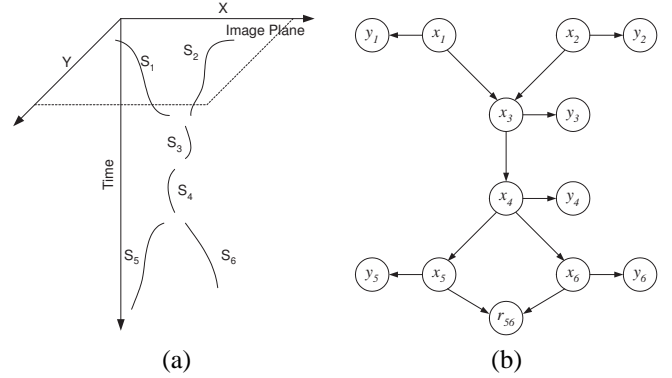


Figure 2. BN tracker: a) object trajectories b) Bayesian network.

ated, then the new trajectory is probably a group with the two previous objects. Finally, visual features can be easily extracted from the video stream (e.g., color histogram) which aid to recognize the objects especially in the case of isolated objects.

A Bayesian network is used to represent the joint distribution of the labels $x = (x_1, \dots, x_N)$ and visual features $y = (y_1, \dots, y_N)$ detected in the video stream. Additional variables r denoted as restriction variables are also used to guarantee that the physical restrictions are verified (details are given in section 4). Fig. 2.b shows the Bayesian network associated with the example of Fig. 2a. The labeling problem is solved if we manage to obtain the most probable configuration given the observations,

$$\hat{x} = \arg \max_x p(x/y, r) \quad (1)$$

where x is the label configuration, y the visual features and r the restriction variables. Each variable corresponds to a node of the BN. Object interaction (trajectory geometry) is encoded in the network topology. Two nodes x_i, x_j are connected if the j -th segment starts after the end of the i -th segment. Additional restrictions are used to reduce the number of connections as discussed in Section 4.

Three issues have to be considered in order to specify a Bayesian network for a tracking problem: i) computation of the network architecture: nodes and links; ii) choice of the admissible labels L_i associated to each hidden node; iii) the conditional distribution of each variable given its parents.

The last two items depend on the type of application. Different solutions must be adopted if one wants to track isolated objects or groups of objects. Group tracking leads to more complex networks since each segment represents multiple objects. These topics are addressed in the next sections. Section 3 describes low level processing and section 4 describes the network architecture.

Since the network represents all the trajectories detected during the operation, the number of nodes increases with time without bound. As mentioned before, this approach can only be used for off-line analysis of short video sequences with few tens of objects. Section 5 describes the extension of this method for on-line operation.

3. Low Level processing

The algorithm described in this paper was used for long term tracking of groups of pedestrians in the presence of occlusions. The video sequence is first pre-processed to detect the active regions in every new frame. A background subtraction method is used to perform this task followed by morphological operations to remove small regions [14].

Then region linking is performed to associate corresponding regions in consecutive frames. A simple method is used in this step: two regions are associated if each of them selects the other as the best candidate for matching [15]. The output of this step is a set of strokes in the spatial/temporal domain describing the evolution of the region centroids during the observation interval.

Every time there is a conflict between two neighboring regions in the image domain the low level matcher is not able to perform a reliable association of the regions and the corresponding strokes end. A similar effect is observed when a region is occluded by the background. Both cases lead to discontinuities and the creation of new strokes.

The role of the Bayesian network is to perform a consistent labeling of the strokes detected in the image i.e., to associate strokes using high level information when the simple heuristic methods fail. Every time a stroke begins a new node is created and the inference procedure is applied to determine the most probable label configuration as well as the associated uncertainty.

4. Network Architecture

The network architecture is specified by a graph, i.e., a set of nodes and corresponding links. Three types of nodes are used in this paper: the hidden nodes x_i representing the label of the i -th segment, the observation nodes y_i which represent the features extracted from the i -th segment and binary restriction nodes $r_{i,j}$ which are used to avoid labeling conflicts. The restriction node $r_{i,j}$ is created only if x_i and x_j share a common parent. A link is created from a hidden node x_i to x_j if x_j can inherit the label of x_i . Physical constraints are used to determine if two nodes are linked (e.g., the second segment must start after the end of the first and the average speed during the occlusion gap is smaller than the maximum velocity specified by the user). Furthermore, we assume that the number of parents as well as the num-

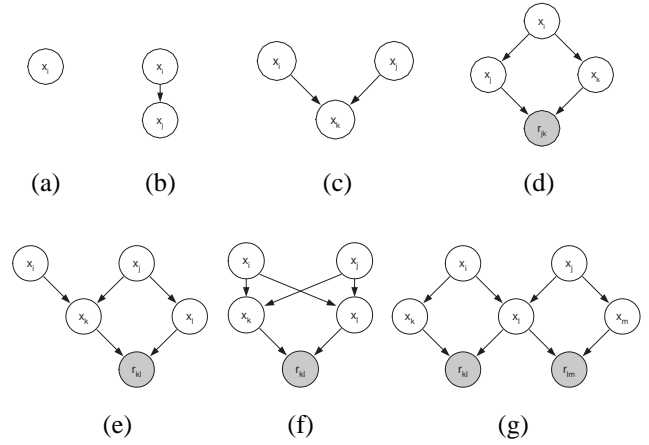


Figure 3. Basic structures (grey circles represent restriction nodes).

ber of hidden children of each node is limited to 2. Therefore, seven basic structures must be considered (see Fig. 3). These structures show the restriction nodes $r_{i,j}$ but the visible nodes y_i are omitted for the sake of simplicity. When the number of parents or children is higher than two, the network is pruned using link elimination techniques. Simple criteria are used to perform this task. We prefer the connections which correspond to small spatial gaps.

4.1. Tracking Isolated Objects

A stroke s_i is either the continuation of a previous stroke or it is a new object. The set of admissible labels L_i is then the union of the admissible labels L_j of all previous strokes which can be assigned to s_i plus a new label corresponding to the appearance of a new object in the field of view. Therefore,

$$L_i = \left[\bigcup_{j \in I_i} L_j \right] \cup \{l_{new}\} \quad (2)$$

where I_i denotes the set of indices of parents of x_i . See Table 1 which shows the labels associated to the hidden nodes of the Bayesian network of Fig. 2. The Bayesian network becomes defined once we know the graph and the conditional distributions $p(x_i|p_i)$ for all the nodes, where p_i are the parents of x_i . As mentioned before, seven cases have to be considered (see Fig. 3). The distribution $p(x_i|p_i)$ for each of these cases are defined following a few rules. It is assumed that the probability of assigning a new label to x_i is a constant P_{new} defined by the user. Therefore,

$$p(x_i = l_{new} | x_j = k) = P_{new} \quad (3)$$

All the other cases are treated on the basis of a uniform probability assignment. For example in the case of Fig. 3c,

k	L_k
1	1
2	2
3	1 2 3
4	1 2 3 4
5	1 2 3 4 5
6	1 2 3 4 6

Table 1. Admissible labels (isolated objects).

x_i inherits the label of each parent with equal probability

$$p(x_i|x_p, x_q) = (1 - P_{new})/2 \quad (4)$$

for $x_i = x_p$ or $x_i = x_q$. Every time two nodes x_i, x_j have a common parent, a binary node r_{ij} is included to avoid conflicts i.e., to avoid assigning common labels to both nodes. The conditional probability table of the restriction node is defined by

$$\begin{aligned} p(r_{ij} = 1/x_i \cap x_j = \emptyset) &= 1 \\ p(r_{ij} = 0/x_i \cap x_j \neq \emptyset) &= 0 \end{aligned} \quad (5)$$

It is assumed that $r_{ij} = 0$ if there is a labeling conflict i.e., if the children nodes x_i, x_j have a common label; $r_{ij} = 1$ otherwise. To avoid conflicts we assume that r_{ij} is observed and equal to 1. Inference methods are used to compute the most probable configuration (label assignment) as well as the probability of the admissible labels associated with each node. This task is performed using the Bayes Net Matlab toolbox [12]. Each stroke detected in the image is characterized by a vector of measurements y_j . In this paper y_j is a set of dominant colors. The dominant colors are computed applying the LBG algorithm to the pixels of the active region being tracked in each segment. A probabilistic model of the active colors is used to provide soft evidence about each node [9]. Each label is also characterized by a set of dominant colors. This information is computed as follows. The first time a new label is created and associated to a segment, a set of dominant colors is assigned to the label. The probability of label $x_j \in L_j$ given the observation y_j is defined by

$$P(x_j/y_j) = \binom{N}{n} P^n (1 - P)^{N-n} \quad (6)$$

where n is the number of matched colors, N is the total number of colors ($N = 5$ in this paper) and P is the matching probability for one color.

4.2. Group Model

This section addresses group modeling. Three cases have to be considered: group occlusions, merging and splitting. Fig. 2 shows a simple example in which two persons

k	L_k
1	1
2	2
3	1 2 (1,2) 3
4	1 2 (1,2) 3 4
5	1 2 (1,2) 3 4 5
6	1 2 (1,2) 3 4 6

Table 2. Admissible labels (groups of objects).

meet, walk together for a while and separate. This example shows three basic mechanisms: group merging, occlusion and group splitting. These mechanisms allow us to model more complex situations in which a large number of objects interact forming groups. After detecting the segments using image processing operations each segment is characterized by a group label x_i . A group label is a sequence of labels of the objects present in the group. A Bayesian network is then built using the seven basic structures of Fig. 3. Let us now consider the computation of the admissible labels. The set of admissible labels L_k of the k -th node is recursively computed from the sets of admissible labels of its parents L_i, L_j , starting from the root nodes. This operation depends on the type of connections as follows:

occlusion

$$L_k = L_i \cup l_{new} \quad (7)$$

merging

$$\begin{aligned} L_k &= L_i \cup L_j \cup L_{merge} \cup L_{new} \\ L_{merge} &= \{a \cup b : a \subset L_i, b \subset L_j, a \cap b = \emptyset\} \end{aligned} \quad (8)$$

splitting

$$L_k = L_j = \mathcal{P}(L_i) \cup l_{new} \quad (9)$$

where $\mathcal{P}(L_i)$ is the partition of the set L_i , excluding the empty set. In all these examples, l_{new} stands for a new label, corresponding to a new track. Table 2 shows the set of admissible labels for the example of Fig. 2. Labels 1,2 correspond to the objects detected in the first frame and labels 3-6 correspond to new objects which may have appeared. Conditional probability distributions must be defined for all the network nodes, assuming that the parents labels are known. Simple expressions for these distributions are used based on four parameters chosen by the user:

- P_{occl} - occlusion probability
- P_{merge} - merging probability
- P_{split} - splitting probability
- P_{new} - probability of a new track

These parameters are free except in the case of the occlusion (Fig. 3b). In this case, the conditional probability of x_k given x_i is given by

$$P(x_k/x_i) = \begin{cases} 1 - P_{new} & x_k = x_i \\ P_{new} & x_k = l_{new} \end{cases} \quad (10)$$

The computation of all conditional distributions for the basic structures are detailed in [10].

The probabilistic models for the observations is the same used in the previous section (see (6))

Since the network represents all the trajectories detected during the operation, the number of nodes increases with time without bound. As mentioned before, this approach can only be used for off-line analysis of short video sequences with few tens of objects. The following section describes the extension of this method for on-line operation.

5. On-line Operation

A tracking system should provide labeling results in real time, with a small delay. Therefore it is not possible to analyse the video sequence in a batch mode i.e., performing inference after detecting the object trajectories. Furthermore, the model complexity must be bounded since it is not possible to deal with very large networks in practice.

To avoid these difficulties two strategies are proposed in the paper: periodic inference and network simplification. The first strategy consists of incrementally building the network and performing the inference every T seconds. If we denote by $x_0^{kT}, y_0^{kT}, r_0^{kT}$ the variables of the video signal in the interval $[0, kT]$, then the inference problem is given by

$$\hat{x}_0^{kT} = \arg \max_{x_0^{kT}} p(x_0^{kT}/y_0^{kT}, r_0^{kT}) \quad (11)$$

The network grows as before but the labeling delay is reduced to less than T seconds. The solution of (11) can be obtained by several methods e.g., by the junction tree algorithm. The Bayes net toolbox was used in this paper [12].

In practice we wish to have an instantaneous labeling of all the objects i.e., we do not wish to wait T seconds for a new global inference. To obtain on-line labeling a sub-optimal approach can be devised which combines the optimal decision obtained at the instant kT with the new information. Let x_i be a hidden node associated to a trajectory active in the interval $[kT, t]$. Using the Bayes law

$$\begin{aligned} P(x_i/y_0^t, r_0^t) &= P(x_i/y_0^{kT}, y_{kT}^t, r_0^{kT}, r_{kT}^t) \\ &= \alpha P(y_{kT}^t, r_{kT}^t/x_i) P(x_i/y_0^{kT}, r_0^{kT}) \end{aligned} \quad (12)$$

where $P(x_i/y_0^{kT}, y_0^{kT})$ is a prior, computed before in the inference step at time kT and $P(y_{kT}^t, r_{kT}^t/x_i)$ represents new

information. The choice of the best label x_i is performed by selecting the highest *a posteriori* probability $P(x_i/y_0^t, r_0^t)$. When x_i is a new variable which was created in the interval $[kT, t]$, then we assume that the prior $P(x_i/y_0^{kT}, y_0^{kT})$ is uniform: no label is preferred based on past information.

The previous strategy converts the batch algorithm into an on-line algorithm i.e., it solves the first problem. However, the network size increases as before. To overcome this difficulty, a simplification is needed. The main idea used in this work is to bound the memory of the system.

Old (hidden and visible) nodes influence the labeling assignment of current nodes. However this influence decreases and tends to zero as time goes by: recent variables are more important than old ones. So, we need to use techniques to forget the past. In this paper, we allow a maximum of N nodes and freeze all the other nodes by assigning them the most probable label obtained in previous inferences. In this way, the complexity of the network remains bounded and can be adapted to the computational resources available for tracking. Several strategies can be used to select the nodes to be frozen (dead nodes). A simple approach is used in this paper: we eliminate the oldest nodes and keep the N most recent. A comparison of this strategy with other using synthetic and real data will be presented elsewhere.

6. Experimental Results

Experimental tests were performed with video surveillance sequences using the implemented on-line tracker described in this paper. The tests were performed with PETS sequences (PETS2001 dataset1 training [5] and PETS2004 "Meet Split 3rdGuy" [7]) used as benchmarks in video surveillance, as well as other video sequences obtained in an university campus.

Figure 4 shows the performance of the tracker in the PETS2004 "Meet Split 3rdGuy" sequence at 25 fps. This is a difficult example, useful to illustrate the performance of the tracker in the presence of occlusions, group merging and splitting. Fig. 4a shows the evolution of all active regions detected in the video stream. This figure displays one of the coordinates of the mass center (column) as a function of time. Every time there is an occlusion or when two or more objects overlap it is no longer possible to associate the new active regions with the ones detected in the previous frame. The trajectories are interrupted in such cases. Fig. 4b shows the labeling results obtained with the on-line algorithm described in the paper. The BN tracker manages to disambiguate most of the occlusions well (only the yellow stroke is misclassified).

Figure 5 shows examples of the tracker performance in group merging and splitting for PETS 2004 sequence. This sequence has three moving objects (3,4,6) and three static objects. The tracker manages to correctly track the three

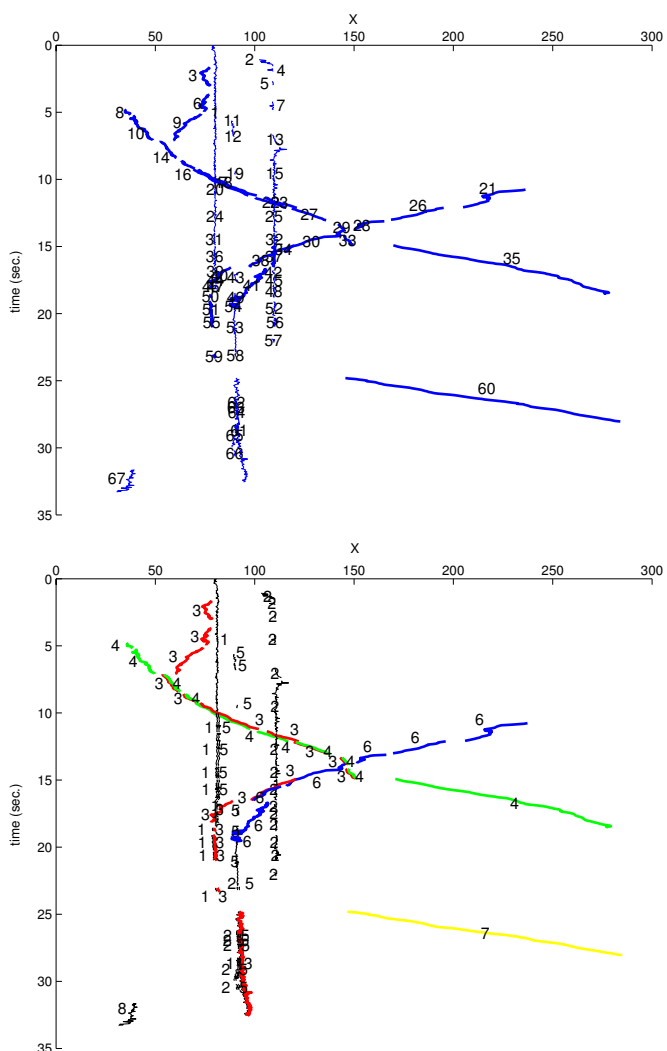


Figure 4. Example (PETS2004 test sequence) :
a) detected strokes; b) most probable labeling
obtained with the on-line algorithm.

moving objects most of the time as shown in Fig. 5. Three persons walk in separately (Fig. 5a), they merge in groups of two (Figs. 5b,c,e) and they split after a while (Figs. 5d,f). All these events are correctly interpreted by the tracker. Namely, the correct label is assigned after the two splits of Figs. 5d,f.

The tracker has some difficulty to deal with the static objects (labels 1,2,5) since they are not correctly detected by the low level algorithms (background subtraction). These objects remain in the same place during the whole sequence. They are therefore considered as background. However, there are small movements which are detected and appear in Figs. 4, 5.

The Bayesian network is automatically built during the

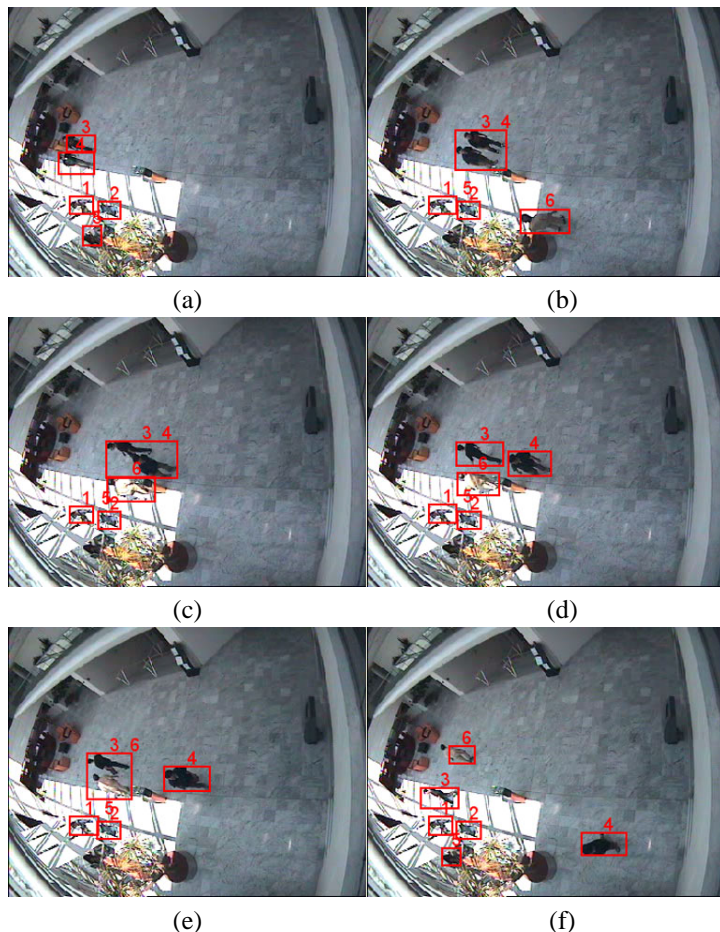


Figure 5. Labeling examples (PETS2004 sequence) after group formation (b,e) and splitting (d,f).

tracking operation. Figure 6 shows the Bayesian network architecture at the instant $t = 12$ sec. Although the number of nodes grows quickly with time, only the most recent ones are updated by the inference algorithm, therefore keeping the computational burden under control. The gray nodes were classified as frozen by the pruning algorithm and their labels and are not allowed to change.

The BN tracker was also applied to other video sequences as well. Figures 7 and 8 show two examples which illustrate the performance of the tracker in group merging and splitting in other video sequences (PETS2001 and campus sequences). Both occlusions are correctly solved e.e., a correct labeling is produced by the tracker once the persons appear isolated again.

Table I shows statistics which characterize the complexity of the three video sequences and the performance of the tracker. It displays the number of objects in the video se-

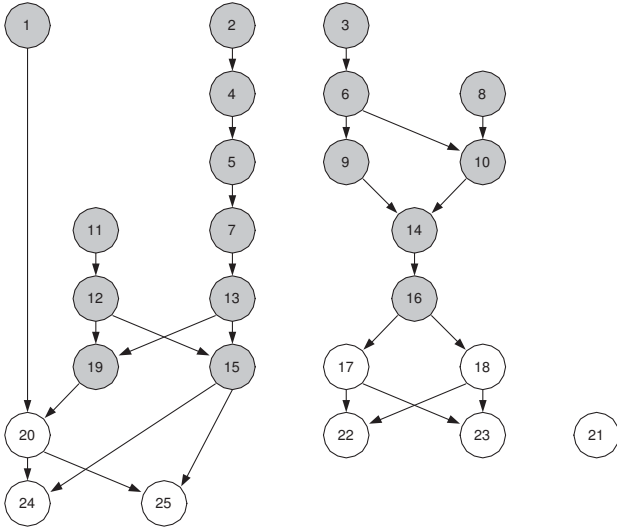


Figure 6. Bayesian network at time instant $t = 12$ sec. (gray nodes are frozen, white nodes are active).

Seq.	NO	NG	NT	LE	D	CT
CAMPUS	7	3	20	0	22.9	2.1
PETS2001	8	5	34	3	120	12.8
PETS2004	7	4	67	5	36	26.6

Table 3. Performance of the BN tracker: Seq. - sequence name; NO - number of objects; NG - number of groups; NT - number of tracks; LE - labeling errors; D - duration (sec.); CT - computational time (sec.).

sequence (NO), the number of groups (NG), the number of tracks detected by the low level processing (NT), the number of labeling errors (LE), the duration of the sequence (D) in sec and the computational time (CT). It is concluded from this table that most of the occlusions are well disambiguated by the proposed algorithm ($LE \ll NT$) and the computational time is smaller than the duration of the sequences¹.

7. Conclusions

This paper presents a system for long term tracking of multiple objects in the presence of occlusions and group merging and splitting. The system tries to follow all moving objects present in the scene by performing a low level detection of trajectories followed by a labeling procedure

¹these tests were performed with Murphy toolbox for Matlab [12], running on a P4 at 2.8 GHz

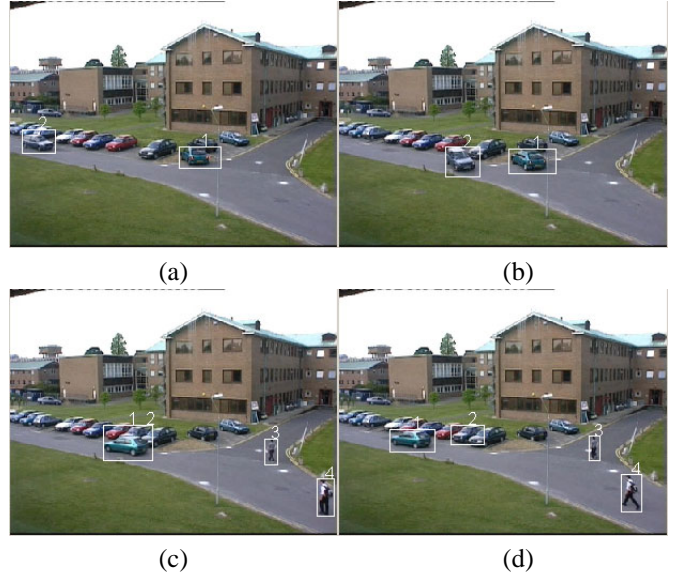


Figure 7. Labeling examples (PETS2001 sequence) after c) group formation and d) splitting.

which attempts to assign consistent labels to all the trajectories associated to the same object. The interaction among the objects is modeled using a Bayesian network which is automatically built during the surveillance task. This allows to formulate the labeling problem as an inference task which integrates all the available information extracted from the video stream and updates the interpretation of the detected tracks every time new information is available. This is a useful feature to solve ambiguous situations such as group splitting and occlusions in which long term memory is needed.

To allow an on line operation of the tracker, inference is periodically performed and pruning techniques are used to avoid a combinatorial explosion of the Bayesian network complexity.

Experimental tests with video sequences were carried out to assess the performance of the system. It is shown that the proposed tracker is able to disambiguate many difficult situations in which there is a strong overlap among different objects.

References

- [1] A. Abrantes, J. Marques, and J. Lemos. Long term tracking using bayesian networks. *IEEE ICIP*, III:609–612, September 2002.
- [2] Y. Bar-Shalom and T. Fortmann. *Tracking and Data Association*. Academic Press, 1998.

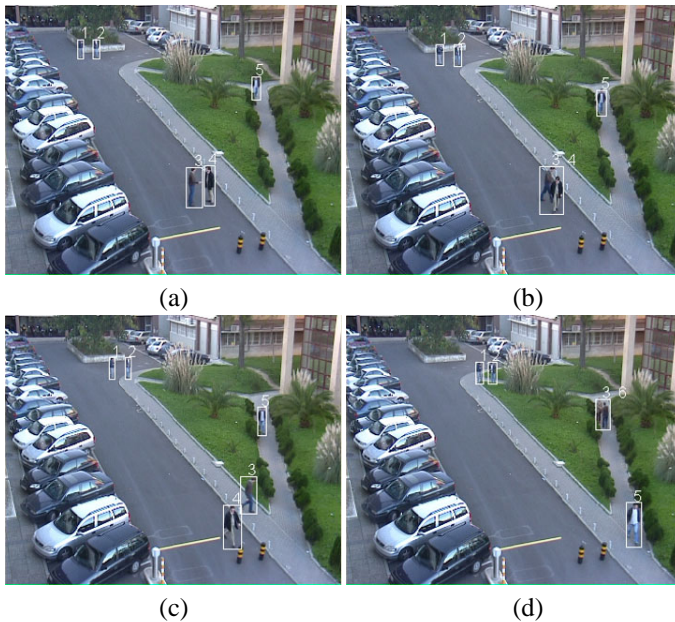


Figure 8. Labeling examples (CAMPUS sequence): after b) group formation and c) splitting.

- [3] I. Cox and S. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the propose of visaul traking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(2):138–150, Feb. 1996.
- [4] R. Deriche and O. Faugeras. Tracking line segments. *Image and Vision Computing*, 8(4):261–270, 1990.
- [5] <ftp://pets2001.cs.rdg.ac.uk>.
- [6] I. Haritaoglu, D. Harwood, and L. Davis. W4: Real-time surveillance of people and their activities. *IEEE Trans. on PAMI*, (22):809–830, 2000.
- [7] <http://www.dai.ed.ac.uk/homes/rbf/CAVIAR/> CAVIAR project/IST 2001 37540.
- [8] M. Isard and A. Blak. Condensation - conditional density propagation for visual tracking. *IEEE Inter. Journal of Computer Vision*, 29(1):5–28, 1998.
- [9] F. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.
- [10] J. Marques, P. Jorge, A. Abrantes, and J. Lemos. Tracking groups of pedestrians in video sequences. *IEEE WoMOT*, June 2003.
- [11] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Journal of CVIU*, (80):42–56, 2000.
- [12] K. Murphy. The bayes net toolbox for matlab. *Computing Science and Statistics*, 33, 2001.
- [13] C. Regazzoni and P. Varshney. Multi-sensor surveillance systems. *IEEE ICIP*, pages 497–500, 2002.
- [14] C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. on PAMI*, 8(22):747–757, 2000.

- [15] S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Trans. on PAMI*, (13):992–1006, 1991.
- [16] C. Wren, A. Azabayejani, T. Darrel, and A. Pentland. Pfunder: Real time tracking of the human body. *IEEE Trans. on PAMI*, (19):780–785, 1997.

A probabilistic model for an EM-like object tracking algorithm using color-histograms

Zoran Zivkovic and Ben Kröse

Intelligent Autonomous Systems Group, University of Amsterdam,
Kruislaan 403, 1098SJ Amsterdam, The Netherlands
{zivkovic,krose}@science.uva.nl,

WWW home page: <http://www.science.uva.nl/research/ias/>

Abstract

In this paper we present a generative probabilistic model of the appearance of a non-rigid object and an iterative procedure for searching for the maximum likelihood (ML) estimate of the position and shape of the tracked object in a new image. The shape of the object in an image is approximated by an ellipse that is described by a full covariance matrix. The appearance of the object is described by a color-histogram. The algorithm is used for tracking persons in image sequences.

1 Introduction

Object tracking requires a model of the tracked object at a certain level of abstraction. Using color-histogram as the model of the tracked object is a very robust representation of the object appearance [9]. Since color histogram abstracts away the information about spatial distribution of the image colors it is particularly interesting for modelling complex non-rigid objects. An efficient method for color-histogram-based object tracking is presented in [1]. The shape of the tracked non-rigid object is approximated by an ellipse. The similarity function between the color histogram of the object and the color histogram of the candidate ellipsoidal regions from a new image from an image sequence is regarded as a probability density function and the ellipse that approximates the shape of the tracked object is regarded as kernel smoothing factor. Then the 'mean-shift' procedure [5] is used to find the closest region in the new image that is the most similar to the object. The shape of the object can change in the image. The object can get close or far away from the camera. The ellipse that approximates the shape of the object should be adapted when the shape and the size of the tracked object change. This problem remained unsolved. In [1] after each tracking step the ellipse is adapted by checking a +10% larger and a -10% smaller ellipse and choosing the best one. Some local shape descriptors were used in [4]. In [7] an extensive search is performed within a range of scales of the ellipse.

The algorithm we present in this paper will address the mentioned unsolved problem of estimating the approximate shape of the tracked object during color-histogram-based object tracking. The contribution of the paper is threefold. First, we present a generative probabilistic model of the appearance of a non-rigid object. The shape of

the object is approximated by an ellipse. The appearance of the object is described by a set of features. The features should be invariant to complex non-rigid object deformations. We use here each bin of a color-histogram as a separate feature with an associated probability density function. Finding an object in a new image becomes then a maximum likelihood (ML) estimation of the object position. Approach [1] is using the Bhattacharyya distance measure between histograms as a similarity measure. It is possible to get the similarity measure from [1] for a particular choice of probability density function that is used to model features in our probabilistic model. Therefore [1] becomes just a particular case in our more general ML framework. Second, we present an iterative procedure to search for ML estimate of the position and also the covariance matrix that describes the ellipse that approximates the shape of the object. We recently proposed this procedure in [10]. However, this paper presents a probabilistic view on the problem. Finally, the paper is using PETS2004 data to present evaluation of the new method and comparison of the new method to [1].

The paper is organized as follows. In section 2 we present a probabilistic framework that can be used to model a non-rigid object. In section 3 we describe color-histogram features. We present in section 4 an EM-like algorithm to search for ML estimate of the position and shape of the non-rigid object in a new image. In section 5 we present the whole practical algorithm and in section 6 we present some experimental results.

2 Probabilistic model

We present here a general generative probabilistic model of the appearance of an object. The model that we presented here is related to the model for color-histogram features that was used for example for texture segmentation in [8]. The position of the tracked object in the image is θ and the approximate ellipsoidal shape of the object is described by a covariance matrix V . We describe the appearance of the tracked object using a set of M scalar features $f_1(\theta, V), \dots, f_M(\theta, V)$ that are extracted from the local area of an image defined by θ and V . Each feature f_m can be regarded as a stochastic variable which has a probability density function $p(f_m)$. We assume that the features are independent. The log-likelihood of a certain observed set of features for a certain θ and V is then given by:

$$\log \mathcal{L}(f_1(\theta, V), \dots, f_M(\theta, V)) = \sum_{m=0}^M \log p(f_m(\theta, V)) \quad (1)$$

For each feature f_m we need its probability density functions $p(f_m)$. This can be learned for example from some test images of the object. Finding the object in a new image is performed by finding the θ and V that maximize the log-likelihood function (1).

3 Color-histogram features

The shape of a non-rigid object is approximated by an elliptical region in an image. We will constrain ourselves here to a class of features that describe color statistics of the

elliptical region. The color-content can be represented by a color-histogram. This representation is very popular and considered to be quite robust to non-rigid transformations of the object. We regard each bin of the color histogram as a feature. Let the histogram have M bins and let the function $b(\mathbf{x}_i) : \mathbb{R}^2 \rightarrow 1, \dots, M$ be the function that assigns a color value of the pixel at location \mathbf{x}_i to its bin. We define a function h_m for the m -th bin of the histogram as:

$$h_m(\mathbf{x}_i) = \delta [b(\mathbf{x}_i) - m], \quad (2)$$

where δ is the Kronecker delta function. The m -th bin of the histogram can then be calculated using:

$$f_m(\boldsymbol{\theta}, V) = \sum_{i=1}^N h_m(\mathbf{x}_i) K(\mathbf{x}_i; \boldsymbol{\theta}, V) \quad (3)$$

The kernel K defines the elliptical region. We use Gaussian kernel:

$$K(\mathbf{x}; \boldsymbol{\theta}, V) = \frac{1}{(2\pi)^{1/2} |V|^{1/2}} e^{-0.5(\mathbf{x}-\boldsymbol{\theta})^T V^{-1}(\mathbf{x}-\boldsymbol{\theta})}. \quad (4)$$

to rely more on the pixels in the middle of the object and to assign smaller weights to the less reliable pixels at the borders of the objects. For computational reasons we usually use only the N pixels from a finite neighborhood of the center of the kernel. This and other practical issues are discussed in more detail later in section 5.

4 An EM-like algorithm

Given a new image we would like to find the position and the approximate shape of the object. We use the model described in section 2 and we search for the parameters $\boldsymbol{\theta}$ and V that maximize the log-likelihood function (1). We propose here an iterative procedure. The procedure is using EM-like iterations [2]. Before the E and the M step the log-likelihood function (1) is approximated locally. These three steps, approximation of the log-likelihood, E and M are repeated iteratively. In this section we describe the local approximation and the E and M steps.

Let us denote current estimated values of the parameters by $\boldsymbol{\theta}^{(k)}$ and $V^{(k)}$. Under the assumption that feature density functions $p(f_m)$ are smooth, we can approximate the log-likelihood function (1) locally using first order Taylor expansion approximation:

$$\log \mathcal{L}(\boldsymbol{\theta}, V) \approx \dots + \sum_{m=1}^M \frac{p'(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))}{p(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))} f_m(\boldsymbol{\theta}, V) \quad (5)$$

where we omitted the constant terms. We denote the variable term from above by $f(\boldsymbol{\theta}, V)$. We replace (3) into this term and get:

$$\begin{aligned}
f(\boldsymbol{\theta}, V) &= \sum_{m=1}^M \frac{p'(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))}{p(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))} f_m(\boldsymbol{\theta}, V) \\
&= \sum_{m=1}^M \frac{p'(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))}{p(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))} \sum_{i=1}^N h_m(\mathbf{x}_i) K(\mathbf{x}_i; \boldsymbol{\theta}, V) \\
&= \sum_{i=1}^N w_i K(\mathbf{x}_i; \boldsymbol{\theta}, V), \tag{6}
\end{aligned}$$

where

$$w_i = w_i(\boldsymbol{\theta}^{(k)}, V^{(k)}) = \sum_{m=1}^M \frac{p'(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))}{p(f_m(\boldsymbol{\theta}^{(k)}, V^{(k)}))} h_m(\mathbf{x}_i). \tag{7}$$

Maximum of the approximated log-likelihood function (5) is achieved for the maximum of (6). Finding the $\boldsymbol{\theta}$ and V that maximize (6) can be done using EM-like iterations [6]. From the Jensen's inequality we get:

$$\log f(\boldsymbol{\theta}, V) \geq G(\boldsymbol{\theta}, V, q_1, \dots, q_N) = \sum_{i=1}^N \log \left(\frac{\omega_i K(\mathbf{x}_i; \boldsymbol{\theta}, V)}{q_i} \right)^{q_i}, \tag{8}$$

where q_i -s are arbitrary constants that meet the following requirements:

$$\sum_{i=1}^N q_i = 1 \text{ and } q_i \geq 0. \tag{9}$$

The E and M steps described below are repeated then until convergence:

1. E step: find q_i -s to maximize G while keeping $\boldsymbol{\theta}^{(k)}$ and $V^{(k)}$ fixed. It is easy to show that the maximum (equality sign in (8)) is achieved for:

$$q_i = \frac{\omega_i K(\mathbf{x}_i; \boldsymbol{\theta}^{(k)}, V^{(k)})}{\sum_{i=1}^N \omega_i K(\mathbf{x}_i; \boldsymbol{\theta}^{(k)}, V^{(k)})}. \tag{10}$$

2. M step: maximize G with respect to $\boldsymbol{\theta}$ and V while keeping q_i -s constant. The q_i -s are now fixed we need to minimize only a part of G that depends on the parameters:

$$g(\boldsymbol{\theta}, V) = \sum_{i=1}^N q_i \log K(\mathbf{x}_i; \boldsymbol{\theta}, V). \tag{11}$$

For the Gaussian kernel and $\frac{\partial}{\partial \boldsymbol{\theta}} g(\boldsymbol{\theta}, V) = 0$ we get:

$$\boldsymbol{\theta}^{(k+1)} = \sum_{i=1}^N q_i \mathbf{x}_i = \frac{\sum_{i=1}^N \mathbf{x}_i \omega_i K(\mathbf{x}_i; \boldsymbol{\theta}^{(k)}, V^{(k)})}{\sum_{i=1}^N \omega_i K(\mathbf{x}_i; \boldsymbol{\theta}^{(k)}, V^{(k)})}. \tag{12}$$

Note that this update equation for the position estimate is equivalent to the 'mean shift' for functions in form of (6) and for the Gaussian kernels. For other kernel types this

might be different. This new EM-like view of the problem gives us the update equations for the smoothing factor in straightforward manner. For the Gaussian kernel and from $\frac{\partial}{\partial V}g(\boldsymbol{\theta}, V) = 0$ we get:

$$\mathbf{V}^{k+1} = \sum_{i=1}^N q_i (\mathbf{x}_i - \boldsymbol{\theta}^{(k)})(\mathbf{x}_i - \boldsymbol{\theta}^{(k)})^T \quad (13)$$

Further discussion about the estimation of the smoothing factor is given in the next section.

The iterative procedure repeats the described three steps. First, for the local approximation we need to compute weights (7). After that, E-like step is given by (10) and M-like step updates the parameter estimates using (12) and (13). The EM algorithm converges always to a local maximum. However, the described procedure does not have guaranteed convergence because of the additional local approximation (5). The EM algorithm makes steps in gradient direction. Using the 'chain-rule' for calculating derivatives we can easily show that the total result from the described three step procedure is also in the gradient direction. It is then easy to make a procedure with guaranteed convergence by adding a line search step at the end [3]. However, in practice it turns out that this is not needed. This was also noticed for the simpler but related procedure from [1].

5 Implementation

In this section we discuss some practical issues.

5.1 Normalization

The feature value defined by (3) should not depend on the size of the object. Total mass under the Gaussian function (4) is equal to one. However in practice for computational reasons we usually use only a finite neighborhood of the center of the kernel. Therefore the feature values should be re-normalized. This can be done by multiplying the feature values by $|V|^{\gamma/2}$:

$$f_{m,\gamma} = |V|^{\gamma/2} f_m \quad (14)$$

The EM-like iterative algorithm from the previous section can be applied to the γ -normalized features. The only difference is in the M-step. Instead of (11) we have now:

$$g(\boldsymbol{\theta}, V) = \sum_{i=1}^N q_i \log |V|^{\gamma/2} K(\mathbf{x}; \boldsymbol{\theta}, V). \quad (15)$$

The position update equation (12) stays the same. From $\frac{\partial}{\partial V}g(\boldsymbol{\theta}, V) = 0$ it is easy to show that the update equation (13) for the covariance matrix V is now given by:

$$\mathbf{V}^{k+1} = \beta \sum_{i=1}^N q_i (\mathbf{x}_i - \boldsymbol{\theta}^{(k)})(\mathbf{x}_i - \boldsymbol{\theta}^{(k)})^T, \quad (16)$$

where $\beta = 1/(1 - \gamma)$. In our implementation we disregard the samples further than 2.5-sigma from the center of the Gaussian kernel. It is easy to show that we should use then $\beta \approx 1.1$. Small errors in choice of β leads to slightly biased solution but since the ellipse is just an approximation of the shape this is acceptable.

One more practical issue might rise from the fact that we use a discrete version of the Gaussian kernel. We use a discrete approximation defined by the image grid and the discretization effects can be also large for small objects.

5.2 Feature probability densities

We use Gaussian distribution to model the color features $p(f_m) = \mathcal{N}(f_m; \mu_m, \sigma_m)$. For current position and shape estimates $\theta^{(k)}$ and $V^{(k)}$ we get the following weights:

$$w_i = w_i(\theta^{(k)}, V^{(k)}) = \sum_{m=1}^M \frac{\mu_m - f_m(\theta^{(k)}, V^{(k)})}{\sigma_m^2} h_m(\mathbf{x}_i). \quad (17)$$

The parameters μ_m, σ_m can be learned from some training data. In the simplest case we have only one training image - the object is selected in the first image of the sequence. Then we can simply set μ_m -s to the measured feature values and set all σ_m -s to be equal.

5.3 Practical algorithm

For the sake of clarity we present here the whole algorithm:

Input: the object model defined by the density functions $p(f_m)$ (μ_m -s and σ_m -s in the Gaussian case), the object initial ($k = 0$) location $\theta^{(k)}$ and shape defined by $V^{(k)}$.

1. Compute the features of the current region defined by $\theta^{(k)}$ and $V^{(k)}$ from the current frame using (3).
2. Calculate weights using (17).
3. Calculate q_i -s using (10).
4. Calculate new position estimate $\theta^{(k+1)}$ using (12).
5. Calculate new variance estimate $V^{(k+1)}$ using the normalized equation (16).
6. If no new pixels are included using the new elliptical region defined by the new estimates $\theta^{(k+1)}$ and $V^{(k+1)}$ stop, otherwise set $k \leftarrow k + 1$ and go to 1.

The procedure is repeated for each frame. In the simplest version the position and shape of the ellipsoidal region from the previous frame are used as the initial values for the new frame.

6 Evaluation and comparison

A number of experimental results is presented here. If the object is tracked for T frames we will get a set of position estimates $\theta(1), \dots, \theta(T)$ and a set of shape estimates $V(1), \dots, V(T)$. To describe the quality of a trajectory we will use the following measures.

Position Errors Measure should describe how close is an estimated track of an object to the ground truth track. The ground truth position is given by $\theta_{gt}(1), \dots, \theta_{gt}(T)$. If at time t we have ground truth size of the object in x and y direction $R_{x,gt}(t)$ and $R_{y,gt}(t)$, then the positional accuracy in x and y direction can be defined as:

$$\begin{aligned} e_x(t) &= (\theta_x(t) - \theta_{x,gt}(t))/R_{x,gt}(t) \\ e_y(t) &= (\theta_y(t) - \theta_{y,gt}(t))/R_{y,gt}(t) \end{aligned} \quad (18)$$

We also report the average value and variance per track.

Shape Errors Measure gives an indication of the level of agreement between the tracked object shape and the estimated shape. The ground truth about the shape in the PETS2004 data set is given by a set of rectangular regions $R_{gt}(1), \dots, R_{gt}(T)$. Therefore we convert our shape estimates $V(1), \dots, V(T)$ to $R(1), \dots, R(T)$ where $R(t)$ is bounding box of an 2-sigma ellipse defined by $V(t)$. Note that our approximate shape estimate by an ellipse with a full covariance matrix is closer to the real shape than the rough bounding box. Unfortunately, only the bounding box ground truth data was available in the PETS2004 data. The error measure we are going to use is given by:

$$e_{shape}(t) = \frac{\text{nonoverlapping area between } R(t) \text{ and } R_{gt}(t)}{\text{area of } R_{gt}(t)} \quad (19)$$

The average and variance of this measure are also reported.

6.1 Computation time



Fig. 1. Example estimated shape for the frame 350 of the Walk1 sequence algorithm.

First in figure 1 we illustrate the performance of the algorithm on the Walk1 sequence from the PETS2004 data. The estimated position and shape of the tracked objects is represented by the dashed ellipse. We observe that both the new shape and the position are accurately estimated by the new algorithm. The mean-shift algorithm provides just an rough approximation. In [7] it was shown that when the size of the object

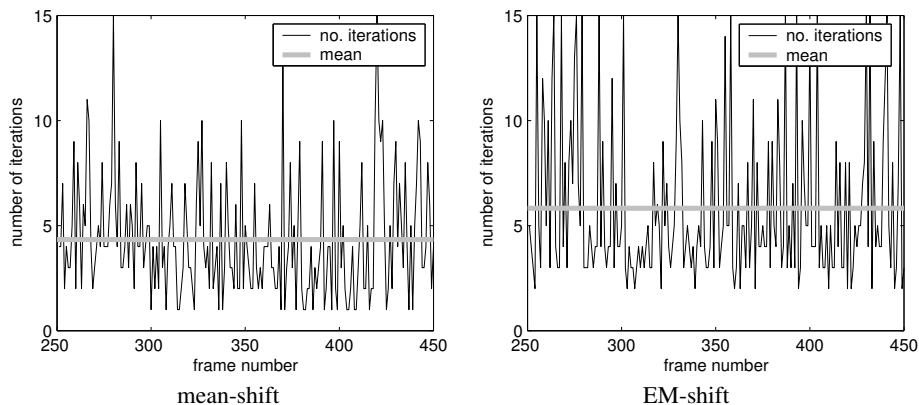


Fig. 2. Number of iterations per frame for the mean-shift and the new EM-like algorithm (for the Walk1 sequence).

changes considerably the mean-shift algorithm fails to adapt the ellipse and often loses its track. Unfortunately there were no such sequences in the PETS data. We tested our algorithm on some of our sequences similar to the sequences shown in [7]. The new algorithm has no problems with adapting and it is much faster than the extensive search method proposed in [7]. In figure 2b) we present the number of iterations of the mean-shift algorithm and the new algorithm for a part of the Walk1 sequence. The average number of iterations per frame was approximately 6. This is slightly more than 4 for the mean-shift based iterations (the same number of iterations was reported in [1]). The computational complexity of one iteration of the new algorithm is slightly higher than the computational complexity of the previous algorithm from [1]. On average our 5-degrees of freedom algorithm is only around 2 times slower than the 2-degrees of freedom mean-shift algorithm but still fast enough for real-time performance. In our current implementation the algorithm works comfortably in real-time on a 1GHz PC.

6.2 Orientation estimation

Our new algorithm estimates the full covariance matrix that describes the ellipse that is used to approximate the shape of the tracked non-rigid object. The covariance matrix V contains also the information about the orientation of the ellipse (see figure 1). In figure 3 we show the estimated orientation angle and the ground truth that was available for the Walk1 sequence. We notice large errors for the both algorithms around the frame number 300. Around this frame the person walks into an area in the scene with very different light conditions, the color histogram of the object changes drastically and the color-histogram model from the first image is not valid anymore. Some adaptation procedure that would adapt the model on-line might lead to some improvements. Another possibility is to combine the results with some other method that is invariant to such changes. However, this is beyond the scope of this article. For the rest of the track the estimates are close to the ground truth. In figure 1 an example was given that illustrates the quality of the shape estimates using the full covariance ellipse.

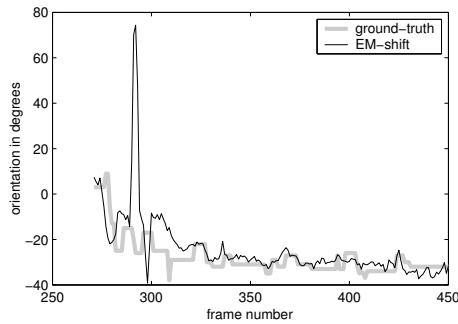


Fig. 3. Orientation estimated by the new EM-like algorithm on the Walk1 sequence and the ground-truth.

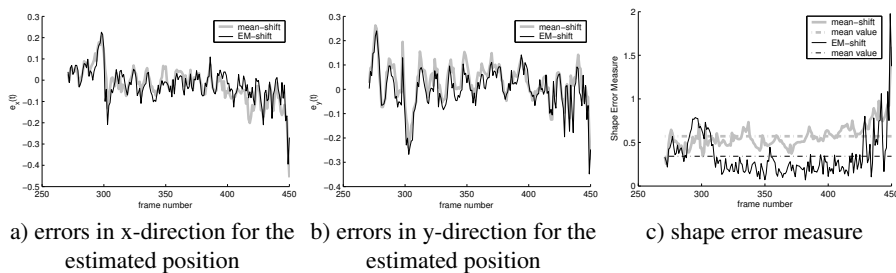


Fig. 4. Performance measures of the mean-shift and the new EM-like algorithm on the Walk1 sequence.

6.3 Quality of the track comparison

We use the proposed measures to describe the quality of the track and to compare the new algorithm with the mean-shift algorithm. The measures are shown for the Walk1 sequence for the track of the women that walks across the corridor (see 1). In figure 4a) and 4b) we present the position error measure in x and y direction for the mean-shift and the new algorithm. Both algorithms perform very similar. We observe again the problems and big errors around the frame number 300. In figure 4c) we show the Shape Error Measure we defined at the beginning of this section. Since our method can efficiently adapt the ellipsoidal approximation to the shape of the object we observe here a big improvement in performance. the average error for the new method is 0.34 while for the mean-shift has an average of 0.56. We should also mention again here that this comparison is not fair because there was only a crude ground truth given by the bounding box of the tracked object. If a better shape description was given, we expect that the difference would be even larger. Furthermore, in order to compare the results with the ground-truth we used the conversion from an ellipse to a bounding box as described and this could lead to some bias. However both for the mean-shift and the new algorithm we used the same conversion so the comparison can be considered to be fair in this sense.

7 Conclusions

We presented a new 5-DOF color-histogram-based non-rigid object tracking. We demonstrated that the new algorithm can robustly track the objects in different situations. The algorithm is evaluated on the PETS2004 data. The data is also used to compare the new algorithm with the 'mean-shift' algorithm. The new algorithm can adapt to changes in shape and scale of the object. The algorithm works in real-time and the computational cost is only slightly higher than for the previously proposed algorithms that had problems with shape and scale changes. The color-histogram-based object tracking procedure can be seen as more general version of the mean-shift algorithm. Presented probabilistic framework could be used to combine the color-histogram features with other features in a consistent way. This is a point of our further research. Furthermore, the presented ideas might be useful for other computer vision problems.

References

1. D.Comaniciu, V.Ramesh, and P.Meer. Real-time tracking of non-rigid objects using mean shift. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2:142–149, 2000.
2. A.P. Dempster, N. Laird, and D.B.Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 1(39):1–38, 1977.
3. R. Fletcher. *Practical Methods of Optimization*. J. Wiley, 1987.
4. G.R.Bradski. Computer vision face tracking as a component of a perceptual user interface. *Proc. IEEE Workshop on Applications of Computer vision*, pages 214–219, 1998.
5. K.Fukunaga and L.D.Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans. Information Theory*, 21:32–40, 1975.
6. R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In *M. I. Jordan editor, Learning in Graphical Models*, pages 355–368, 1998.
7. R.Collins. Mean-shift blob tracking through scale space. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2003.
8. N. Sebe and M.S. Lew. A maximum likelihood investigation into color indexing. In *Proceedings Visual Interface'00*, pages 101–106, 2000.
9. M. Swain and D. Ballard. Color indexing. *Intl. J. of Computer Vision*, 7(1):11–32, 1991.
10. Z. Zivkovic, and B. Krose. An EM-like algorithm for color-histogram-based object tracking. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

Tracking Pedestrians in a Multiple Cameras System with Trajectory Prediction and Occlusion Modelling

Jorge P. Batista
ISR - Institute of Systems and Robotics
DEEC/FCT - University of Coimbra
Coimbra - PORTUGAL
batista@isr.uc.pt

Abstract

This paper presents a integrated solution to track multiple non-rigid objects (pedestrians) in a multiple cameras system with ground-plane trajectory prediction and occlusion modelling. The resulting system is able to maintain the tracking of moving objects before, during and after occlusion. Moving targets are detected and segmented using a dynamic background model combined with motion detection and brightness and color distortion analysis. Two levels of tracking have been implemented: the image level tracking and the ground-plane level tracking. Several target cues are used to disambiguate between possible candidates of correspondence in the tracking process: spacial and temporal estimation, color and object height. A simple and robust solution for image occlusion monitoring and grouping management is described. Experiments in tracking multiple pedestrians in a dual camera setup with common field of view are presented.

1. Introduction

Tracking non-rigid objects (humans) and classifying their motion is a challenging problem. Effective solutions to these problems would lead to breakthroughs in areas such as video surveillance, motion analysis and recognition.

Tracking people in relatively unconstrained, cluttered environments as they form groups, and part from one another requires robust methods that cope with the varied motions of the humans, occlusions, and changes in illumination. When occlusion is minimal, a single camera may be sufficient to reliably detect and track objects, although, in most cases, robust tracking of multiple people through occlusions requires human models to disambiguate occlusions [11]. However, when the density of objects is high, the resulting occlusion and lack of visibility requires the use of multiple cameras and cooperation between them so that the objects are detected using information available from all the cameras covering a surveillance area [9, 13].

The system described on this paper explore the combination of multiple cameras to solve the problem of autonomously detect and track multiple people in a surveillance area. Since no *a priori* model of people is available, the paper presents a

tracking method based on appearance: tracking the perception of people's movements instead of tracking their real structure. An improved image tracking mechanism that combines *image segmentation* and *recursive trajectory estimation* is proposed. The recursive approach is used to feedback into the image tracking level the ground-plane predicted target information. The integration of this information in the image tracking level enables robust tracking of multiple pedestrians, helping to disambiguate problems of temporary occlusion (people crossing, forming and leaving groups) as well permanent occlusion situations (people standing behind closets). The ground-plane pedestrians trajectory prediction is obtained fusing the information supplied by the multiple cameras, managing people's grouping. Several target cues are used to disambiguate between possible candidates of correspondence in the tracking processes: spacial and temporal estimation, color and object height.

Experiments in tracking multiple objects in a dual camera setup with common field of view are presented. An accurate ground-plane trajectory prediction is obtained under several types of occlusion.

2. Related work

The problem of multiple tracking in multiple cameras has been widely studied. In [10], only relative calibration between cameras is used, and the correspondence is established using a set of feature points in a Bayesian probability framework. Geometric features of the targets such as the height of a person are used. The system is able to predict when a person is about to exit the current camera's field of view and selects the next optimal view for tracking. Occlusion problems are not robustly solved. The continuous tracking of moving objects in each view using a Tensor Voting based approach was presented in [19]. The objects trajectory is obtained by performing a perceptual grouping in 2D+t using Tensor Voting. The same authors presented a solution for continuous tracking of moving objects observed by multiple, heterogeneous cameras [18]. They addressed the tracking problem by separately modelling motion and appearance of the moving objects using two probabilistic models. Multiple color distribution components were used for the appearance model and the motion model was obtained using a Kalman Filter process. The track-

ing was performed by the maximization of a joint probability model. A system for tracking people in multiple uncalibrated cameras was presented in [17]. The system is able to discover spacial relationships between the camera fields of view and use this information to correspond between different perspective views of the same person. They proposed a solution to find the limits of the field of view (FOV) of a camera as visible in the other cameras, and the FOV constraint was used to disambiguate between possible candidates of correspondence. In [16] a multi-camera system based on Bayesian modality fusion was used to track people in a indoor environment. The system integrates multiple modalities based on motion continuity and the apparent color, using a kalman filter to estimate motion and modelling color as Gaussian mixture models in hue and saturation space. A solution with non-calibrated cameras was proposed in [20]. The camera calibration information is recovered by observing motion trajectories in the scene. Motion trajectories in different views are randomly matched against one another and plane homographies obtained for each match. The correct homography is the one that is statistically more frequent. A multiple camera solution was also presented by [13] to deal with severe occlusions situations. This approach assumes the use of calibrated cameras and that people are moving on a calibrated ground plane. Different characteristics of people, like color models at different heights of the person, are modelled in order to facilitate the segmentation of people in a crowded scene. The system uses an iterative solution for segmentation and ground plane position estimation, using segmentation results to find people's ground plane positions and the positions thus obtained to obtain segmentations. A multi camera image tracking was developed by [9]. Moving objects are detected by using background subtraction and viewpoint correspondence between the detected objects is established by using the ground plane homography constraint. The objects are tracked in 3D using a linear kalman filter. The problem of auto-calibration of a set of cameras was addressed by [15]. They employ a simple learning calibration procedure by merely watching objects entering, passing through and leaving the monitored scene. A linear model of the projected height of objects in the scene was used in conjunction with world knowledge about the average person height to recover the image to ground plane transformation of each camera. Each camera defines its own ground plane coordinate system. A Hough transform technique was used to recover the transformations between these local ground planes. A solution for counting people in crowds with a real time network of image sensors was presented in [14]. In this system, groups of image sensor segment foreground objects from the background, aggregate the resulting silhouettes over a network, and compute a planar projection of the scene's visual hull. This projection was to bound the number of possible location of people.

3. Multiple Target Detection

In this system, only moving objects are considered as targets (pedestrians). As the camera (sensor node) is fixed, target detection is based on a combination of motion detection and

brightness and chromaticity distortion analysis. This approach allows a robust segmentation of shading background from the ordinary background or moving foreground objects. The background image model is regularly updated to compensate for illumination change and to include or remove in the background model the objects that stopped or started their movement in the field of view of the camera. The detection model has to face two kinds of problems: it has to deal with image processing problems, solving problems like noise, shadows, reflections and highlights, and also more common segmentation/tracking problems like occlusions, split and merge of targets and the appearance or lost of targets.

Due to the non-rigid nature of the human motion, and since de present system don't use any kind of information concerning human shape, a segmented foreground region can either correspond to the perception of noise (e.g shadows, ghosts), a real moving target (e.g human body), just part of that target (e.g. head or hand of a person) or a group of targets (e.g. a crowd).

To solve the noisy perception of the target, is special when the detection module segments shadows and highlights has being part of the target, we used the brightness and color distortion approach proposed by [4] for static image background models, adjusting it to the case of dynamic image background models. Problems concerning incomplete single target perception due to occlusion, incorrect segmentation and targets split/merge was solved at the single-view tracking module combining the output of the image processing module and the information feedback from the master node.

3.1 Moving target segmentation

Each pixel in a new image is classified as one of background (B), object (O), shadow (S) or highlight (H) and ghost (G). The clustering of foreground pixels is based on the following validation rules

Object \rightarrow (foreground pixel)&[\sim (shadow/highlight)]&(in motion)

Shadow/Highlight \rightarrow (foreground pixel)&(shadow/highlight)

Ghost \rightarrow (foreground pixel)&[\sim (shadow/highlight)]&[\sim (in motion)]

The distinction between objects, shadows/highlights among pixels not classified as background is made using the brightness and chromaticity distortion [4]. Representing $E_{RGB}^t(x) = \mu_{RGB}^t(x)$ the expected background pixel's RGB value, the brightness distortion α_x is a scalar value that brings the observed color close to the expected chromaticity line. It is obtained by minimizing

$$\phi(\alpha_x) = (I_{RGB}^t(x) - \alpha_x E_{RGB}^t(x))^2 \quad (1)$$

representing α_x the pixel's strength of brightness with respect to the expected value. α_x is 1 if the brightness of the given pixel in the current image is the same as in the reference image. α_x is less than 1 if it is darker, and greater than 1 if it becomes brighter than the expected brightness. Color distortion is defined as the orthogonal distance between the observed color and the expected chromaticity line. The color

distortion of a pixel x is given by

$$CD(x) = \|I_{RGB}^t(x) - \alpha_x E_{RGB}^t(x)\| \quad (2)$$

The brightness distortion and the chromaticity distortion equations become

$$\alpha_x = \frac{\sum_{i=R,G,B} \frac{I_i^t(x)\mu_i^t(x)}{\sigma_i^t(x)}}{\sum_{i=R,G,B} \left(\frac{\mu_i^t(x)}{\sigma_i^t(x)}\right)^2} \quad (3)$$

$$CD(x) = \sqrt{\sum_{i=R,G,B} \left(\frac{I_i^t(x) - \alpha_x \mu_i^t(x)}{\sigma_i^t(x)}\right)^2} \quad (4)$$

Applying the suitable thresholds to the brightness distortion α_x and to the chromaticity distortion $CD(x)$ of a pixel x yields an object mask $M(x)$ according to

$$M(x) = \begin{cases} O, & : \widehat{CD}(x) > \delta_{CD} \text{ or } \widehat{\alpha}_x < \delta_{\alpha_{low}}, \text{ else} \\ B, & : \widehat{\alpha}_x < \delta_{\alpha_1} \text{ and } \widehat{\alpha}_x > \delta_{\alpha_2}, \text{ else} \\ S, & : \widehat{\alpha}_x < 0, \text{ else} \\ H, & : \text{otherwise} \end{cases} \quad (5)$$

where $\widehat{\alpha}_x$ and $\widehat{CD}(x)$ represent the normalized brightness distortion and normalized chromaticity distortion respectively. The normalization of α_x and $CD(x)$ is obtained by

$$\widehat{\alpha}_x = \frac{\alpha_x - 1}{a_x} \quad \widehat{CD}(x) = \frac{CD(x)}{b_x} \quad (6)$$

where a_x and b_x represents, respectively, the variation of the brightness distortion and chromaticity distortion of pixel x , defined as $a_x = RMS(\alpha_x)$ and $b_x = RMS(CD(x))$. In [4] the RMS values are obtained during the modelling of the static background over a stack of N frames. In a dynamic background model, just like the one adopted in this system, the RMS values need to be dynamically updated. In order to overcome this constraint, the RMS values are constantly updated considering only the pixels detected as background.

The three-frame difference rule suggests that a pixel x in frame t is moving if its intensity $I_{RGB}^t(x)$ has changed significantly between both the current image t and the last image $t-1$, and the current image t and the next-to-last frame $t-2$. Based on this rule, an image mask for moving pixels is create using

$$\begin{cases} 1, & \text{if} \begin{cases} |I_{RGB}^t(x) - I_{RGB}^{t-1}(x)| > \delta_{RGB}(x) \\ \text{and} \\ |I_{RGB}^t(x) - I_{RGB}^{t-2}(x)| > \delta_{RGB}(x) \end{cases} \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

where $\delta_{RGB}(x)$ are thresholds for R, G, B image components for pixel position x . Moving pixels are clustered into connected regions defining a bounding box per region.

Each pixel of the background image is modelled by a multidimensional Gaussian distribution in RGB space (mean and standard deviation). These parameters are updated with each new frame using the following linear filter

$$\mu_{RGB}^t(x) = (1 - \alpha) \mu_{RGB}^{t-1}(x) + \alpha I_{RGB}^t(x) \quad (8)$$

$$\sigma_{RGB}^t(x)^2 = (1 - \alpha) \sigma_{RGB}^{t-1}(x)^2 + \alpha (I_{RGB}^t(x) - \mu_{RGB}^t(x))^2 \quad (9)$$

being α used to control the rate of adaptation ($0 \leq \alpha \leq 1$). A critical situation occurs whenever objects stop their movement for a period or when objects modelled as being part of the background start moving. To deal with this situation, each pixel has a state transition map defining a dynamic pixel rate of adaptation. Pixels that have changed from a background state to a ghost state will have a high rate of adaptation whereas those changing from moving object state to ghost state will have a low rate of adaptation. The state transition map will encode in all the moving object pixels the elapsed time since the beginning of the object movement. Different rates of adaptation are used according to

$$\alpha = \begin{cases} 1.0 & : \text{if} [(Ghost) \& (elapsed\ time < \delta_t)] \\ 0.0 & : \text{if} [Object] \\ K \cdot e^{-\nabla t} & : \text{if} [(Ghost) \& (elapsed\ time \geq \delta_t)] \\ 0.05 & : \text{Otherwise} \end{cases} \quad (10)$$

where ∇t is the elapsed time since the target stopped its movement and $\delta_t = \frac{\sqrt{w_b^2 + h_b^2}}{f_r \cdot \sqrt{v_x^2 + v_y^2}}$ being (w_b, h_b) the width and height of the bounding box respectively, f_r the frame rate and (v_x, v_y) the image velocity components of the bounding box center of mass. This solution enable the detection of moving objects that stopped their motion for a period of time avoiding also the detection of the ghost cast created when background objects start moving.

The output of the target segmentation module is basically a collection of blobs that represents the segmented shape of the moving targets. Figure 1 shows the result of the target detection process in one of the static camera nodes.

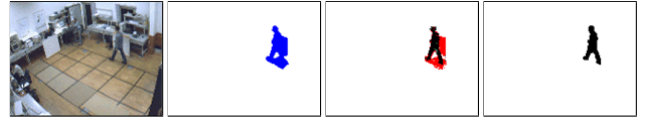


Figure 1. Target segmentation with shadow detection

3.2 Image target model

The target model adopted is composed of three primitives: the image coordinates of the point of contact of the pedestrian with the ground plane, $p_f = [x_f, y_f, 1]^T$, the image coordinates of the head of the pedestrian, $p_h = [x_h, y_h, 1]^T$, and the width of the bounding box (w_b) measured at the center of mass. Assuming an upright walking posture for pedestrians, the coordinate pairs p_f and p_h are defined as the intersection of the line passing through the bounding box center of mass and the image vanishing point of the vertical posture with the top and bottom lines of the bounding box (figure 2). This approach enables a very precise detection of the feet and head of the pedestrians. Two additional target cues have been used: the color information associated to the blob and the estimated 3D height of the target. A target n is defined by $T_n = \{p_f^n, p_h^n, w_b^n, H_n, \#n\}$, where H_n represents the color histogram of the target's blob and $\#n$ is the number of points of the target segmented blob.

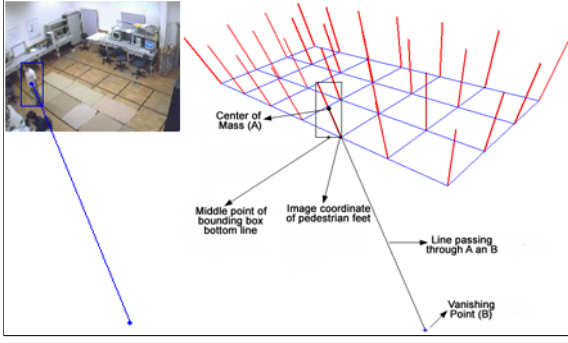


Figure 2. Image Target Model

3.3 Target color model

In order to track people consistently, in special when they merge and split, each person's appearance must be modelled. This information is particularly important during the tracking despite the ambiguities that may arise with partial occlusion and grouping. Color is a very important cue and play an important role in this process. A color model is built and adapted to each tracked target, being updated with each new frame.

Color distributions have been effectively modelled for tracking using both color histograms and gaussian mixture models [8]. Although both approaches perform well, on the present solution color histograms have been used. To avoid problems due to changing light intensity, a simple color constancy algorithm was used that simple normalizes the R,G,B color components through

$$r' = R/(R+G+B) \quad g' = G/(R+G+B) \quad b' = B/(R+G+B) \quad (11)$$

An histogram $H_n(i)$ simply counts the number of occurrences of $i = (r', g')$ within the detected blob for person n . A discrete probability distribution is obtained from the histogram

$$P(i|n) = \frac{H_n(i)}{A_n}, \quad (12)$$

where A_n is the area of the target blob in pixels. Histogram models are adaptively updated by storing the histograms as probability distributions and updating them as

$$P_t(i|n) = \beta P_{t-1}(i|n) + (1 - \beta) P_t^{tar}(i|n), \quad (13)$$

where $P_t^{tar}(i|n)$ is the probability distribution obtained from current image, and $0 \leq \beta \leq 1$.

Each normalized color component (r', g') is quantized into 64 values (6 bits). This give a total of $4096 = 64^2$ histogram bins. Histogram color models are matched using the histogram intersection method proposed in [7]. Given a pair of histograms, H_n and H_m respectively for target n and model m , each containing $k = 4096$ bins, the intersection of the histograms is defined as

$$\bigcap(H_n, H_m) = \sum_{j=1}^k \min(H_n(j), H_m(j)) \quad (14)$$

The result of the intersection of a model histogram with a target histogram is the number of pixels from the model that have

corresponding pixels of the same color in the image. The normalized histogram intersection can be obtained intersecting the discrete probability distribution histogram of the model P_m and the target P_n ,

$$M(H_n, H_m) = \sum_{j=1}^k \min(P_n(j), P_m(j)) \quad (15)$$

or considering

$$M(H_n, H_m) = \frac{\sum_{j=1}^k \min(H_n(j), H_m(j))}{\sum_{j=1}^k H_m(j)}. \quad (16)$$

Equation 15 is used in a one-to-one target matching, defining the existence of a correct match when $M(H_n, H_m) > 0.8$. Equation 16 is used to detect the presence of a certain target as a member of a group, considering the histogram of the target as the model histogram.

For the tracking purpose, each segmented and tracked target has a target descriptor that stores the updated normalized histogram $P_t(i|n)$.

Color information is also used in the master node to disambiguate between possible candidates in the matching process. Since each sensor node has a normalized color histogram for each target, the master combines this information to obtain a reliable color information of the target. A simple color histogram union is used to combine the color information of target n obtained from x sensor nodes

$$P_n^G = \bigcup (P(j|n_y))_{y=1..x} = \max(P(j|n_y))_{y=1..x} \quad (17)$$

where P_n^G is the normalized color histogram of target n in the master node, and $P(j|n_y)$ is the normalized color histogram of the target obtained by the sensor node y . P_n^G is adaptively updated using the same approach proposed for the sensor nodes.

3.4 Target height model

Target height is modelled as $t_h^n = \mathfrak{S}(p_f^n, p_h^n, H_i)$ being H_i the image to ground plane homography transformation of each sensor node ($H_i = [\bar{h}_1, \bar{h}_2, \bar{h}_3]^T$) (section 5). Representing the ground-plane location of sensor node i by $C_i = [c_x, c_y, c_z]$ (fig. 3), the height of target T_n is defined as

$$t_h^n = \frac{c_z \cdot \sqrt{(p_{g_h}^n(x) - c_x)^2 + (p_{g_h}^n(y) - c_y)^2}}{\sqrt{(p_{g_h}^n(x) - p_{g_f}^n(x))^2 + (p_{g_h}^n(y) - p_{g_f}^n(y))^2}} \quad (18)$$

where $p_{g_j|j=h,f}^n(x) = (\bar{h}_1 \cdot p_{j|j=h,f}^n) / (\bar{h}_3 \cdot p_{j|j=h,f}^n)$ and $p_{g_j|j=h,f}^n(y) = (\bar{h}_2 \cdot p_{j|j=h,f}^n) / (\bar{h}_3 \cdot p_{j|j=h,f}^n)$.

Knowing the height and ground-plane location of a pedestrian P_n , the image projection of his head and feet is modelled as $\tilde{p}_{j|j=f,h}^n = \varphi(t_h^n, p_{g_j}^n, H_i^{-1})$. The coordinates of the head and feet image projection are obtained by

$$\tilde{p}_f^n = H_i^{-1} \cdot \begin{bmatrix} p_{g_f}^n(x) \\ p_{g_f}^n(y) \\ 1 \end{bmatrix} \quad (19)$$

$$\tilde{p}_h^n = H^{-1} \cdot \begin{bmatrix} p_{g_f}^n(x) + \frac{t_h^n \cdot (p_{g_f}^n(y) - c_y)}{c_z - t_h^n} \\ p_{g_f}^n(y) + \frac{t_h^n \cdot (p_{g_f}^n(x) - c_x)}{c_z - t_h^n} \\ 1 \end{bmatrix} \quad (20)$$

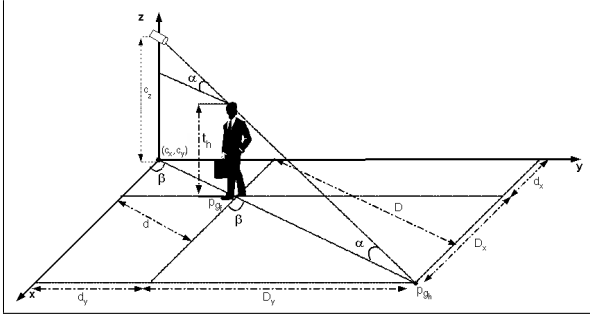


Figure 3. Target Height Model

4. Single-View Tracking

The single-view tracking aim to track at the image level all moving targets detected and segmented by the image processing level. The main advantage of using this tracking stage is basically to filter out noisy segmented target primitives and to model image occlusions and grouping, increasing the robustness of the global ground-plane pedestrians tracking system.

The target state vector is $X = [p_f \ p_h \ w_b \ \dot{p}_x \ \dot{p}_y \ \dot{w}_b]^T$ where $p_i = (x_i, y_i)|_{i=f,h}$ and $\dot{p}_i = (\dot{x}_i, \dot{y}_i)|_{i=f,h}$ are the position and velocity of the model target feature points and w_b is the width of the bounding box.

The system model used is the following discrete model:

$$X_k = \mathbf{f}(X_{k-1}, k-1) + \mathbf{W}_k \quad (21)$$

$$Z_k = \mathbf{h}(X_k, k) + \mathbf{V}_k \quad (22)$$

where \mathbf{W}_k is a discrete-time white noise process with mean zero and covariance matrix Q , \mathbf{V}_k is a discrete-time white noise process with mean zero and covariance matrix R , and \mathbf{W}_j , \mathbf{V}_k , and X_0 are uncorrelated for all j and k . We considered the assumption that trajectories are locally linear in 2D and the width of the bounding box changes linearly, resulting for the system model the following linear difference equation $X_k = A \cdot X_{k-1} + W_k$ where the system evolution matrix, A_k , is based on first order Newtonian dynamics and assumed time invariant.

The measurement vector is $Z_k = [p_f, p_h, w_b]^T$ and is related to the state vector via the measurement equation $Z_k = C \cdot X_k + V_k$.

4.1. Image occlusion and grouping management

At this stage it is important to define the concept of an *object*. An *object* represents an image tracked target and can be of a single or compound nature. It is represented by the descriptor $O_n = [T_n, \zeta_n, j, \{\mathcal{L}[i]|_{i=1..j}\}]$, where T_n represents the object descriptor, ζ_n the tracker parameters and j the number of targets associated to the object n . $\mathcal{L}[i]$ is a list of pointers to the j object descriptors that form the compound object ($j > 1$).

To disambiguate between possible candidates of correspondence in the tracking process two image cues were used : spatial and temporal estimation and color. The Histogram color matching

$$CM(O_n, T_n) = \bigcap (O_n, T_n) \quad (23)$$

and the bounding boxes overlapping ratio

$$OR(\hat{O}_n, T_n) = \max \left(\frac{\bigcap(\hat{O}_n, T_n)}{\nabla T_n}, \frac{\bigcap(\hat{O}_n, T_n)}{\nabla \hat{O}_n} \right) \quad (24)$$

were used to build correspondence matrices (*CMat*) between a *posterior* estimated image position *objects* (\hat{O}_n) and targets (T_n) for time frame t . ∇ represent the area of the bounding box and \bigcap the bounding boxes overlapping area.

$$CMat: \begin{array}{c|cccc} & T_1 & T_2 & \dots & T_n \\ \hline \hat{O}_1 & 1 & 0 & \dots & 1 & 2 \\ \hline \hat{O}_2 & 0 & 1 & \dots & 0 & 1 \\ \hline \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ \hline \hat{O}_n & 0 & 1 & \dots & 0 & 1 \\ \hline & 1 & 2 & \dots & 1 & \end{array} \quad (25)$$

A unitary value at the bottom row represents a 1 \longleftrightarrow 1 correspondence between the object and the target, values greater than one indicates the existence of object merges and null values indicates the existence of new detected targets. The last column of the matrix indicates the existence of an object split for values greater than one, a 1 \longleftrightarrow 1 correspondence for unitary values and the lost of an object for null values.

Based on the correspondence matrices, four managers, running in cascade, were used to handle the image *objects*: *split* manager, *merge* manager, *new/lost* manager and *update* manager.

The Split manager When a split situation is detected, two possible situations can happen: a compound object split (the most regular case) or a single object split. This last case can happen when a group enter the surveillance area and split.

To handle the compound object split, the manager creates a new correspondence matrix between the objects that form the compound object and the image targets (T_n) that were detected as split candidates. This time the correspondence is based on color histograms and target height, associating a segmented target to each object of the compound object. The descriptors of the objects are recovered from the compound object and added to the tracked object list, associating to each object the segmented target primitives of the target they matched. The compound object descriptor is removed from the tracked objects list and discarded.

For the case of a single split, a new object is created and added to the new born object list. This new object is definitely moved to the tracked object list after being tracked for 5 consecutive frames.

The Merge manager When a merge situation is detected, a compound object descriptor is created and added to the list of object trackers, moving the object descriptors of

the merged objects from the tracked object list to a dying object list, decreasing its life vitality over a period of 10 frames, being definitely discarded after this period. The new object descriptor includes the color histograms and the 3D target height of the objects merged (compound objects descriptors) and also the number of targets merged. If a split situation is detected before the death of the objects (ex: objects crossing), the objects descriptors are recovered from the dying objects list to the tracked list.

The New/Lost manager When a null value is detected on the last row of the OR matrix this means that a new object was detected. A single object descriptor is created and included on a list of new born object increasing its life vitality over a stack of 5 frames. After this period, the descriptor is moved to the tracked object list.

If a null value is detected on the last column of the OR matrix a lost object is considered to happen. Its descriptor is moved from the tracked object list to a dying object list decreasing its life vitality for a period of 10 frames over which it is definitely discarded.

The Update manager At this stage, the tracked object list has a complete *object* \rightarrow *target* matching, updating the object trackers with the segmented targets (T_n) information.

The feedback process supplies to each sensor node information about where and how many target should be detected at time k , taking $k-1$ observations from n cameras that are fused at the ground-plane tracking level. This information is useful to cross-check the existence of groups and also to validate the cardinal of those groups by counting the number of projected targets that fall inside the bounding area of a detected group. This approach enables a more robust image split/merge and targets grouping.

5. Target Ground-Plane Mapping

Each one of the elements of the tracked object list has a state vector X_k and an associated error covariance matrix P_k obtained from the tracker.

Each sensor node has an associated homography $H_i|_{i=1,2} = [\bar{h}_1, \bar{h}_2, \bar{h}_3]^T$ that maps image points into the ground plane surveillance area, mapping the tracked target's primitives, p_f and p_h , into the ground plane through the homography transformation $p_{g_j}|_{j=f,h} = H_i \cdot p_j|_{j=f,h}$.

Considering the existence of a certain uncertainty for the coordinates of $p_j|_{j=f,h}$ and an uncertainty for the homography estimation, which are considered uncorrelated, the mapping of $p_j|_{j=f,h}$ into the ground-plane will have an associated uncertainty that is given by $P_{p_g} = J_{H_i} P_{H_i} J_{H_i}^T + J_{p_g} P_k J_{p_g}^T$ where J represents the Jacobian matrices, P_k represents the error covariance matrix obtained from the object Kalman filter tracker and P_{H_i} the error covariance matrix obtained using the solution proposed by [3].

Since the 3D target height is modelled as $t_h^n = \mathfrak{S}(p_f^n, p_h^n, H_i)$, that results on the equation 18, the uncer-

tainty associated to the 3D target height is given by $P_{t_h^n} = J_{t_h} P_{p_g} J_{t_h}^T$, where J_{t_h} is the Jacobian matrix of equation 18.

6. Ground-Plane Tracking

The ground-plane tracking level has two major purposes: merge the information mapped on the ground-plane by the sensor nodes and perform the ground-plane tracking of the pedestrians detected by the sensor nodes, managing the group/ungroup occurrences.

Pedestrians are tracked on the ground-plane using a Kalman filter tracker. The state vector is $X = [p_g \ \dot{p}_g \ \ddot{p}_g \ t_h]^T$ where p_g is the pedestrian ground-plane position, \dot{p}_g is the pedestrian velocity and \ddot{p}_g is the pedestrian acceleration. t_h is the 3D target height. A constant acceleration model was adopted to the pedestrian movement and the height of the pedestrian was modelled as constant. The dimension of the measurement vector is dependent on the number of sensor nodes that are able to detect and track the pedestrian. Assuming this number to be m , the measurement vector is $Z_k = [p_g^1, t_h^1, p_g^2, t_h^2, \dots, p_g^m, t_h^m]^T$, being related to the state vector via the measurement equation $Z_k = C \cdot X_k + V_k$. The dimension of the matrix C is $3m * 7$. The measurement error covariance matrix V_k is defined using the uncertainty ground-plane mapping propagation described on previous section.

The ground-plane tracked objects are referenced as *Pedestrians* and they are represented by the descriptor $P_n = [K_n, \zeta_n, j, \{\mathfrak{L}[i]|_{i=1..j}\}]$, where K_n represents the pedestrian descriptor (ground-plane position, height and histogram color), ζ_n the tracker parameters and j the number of pedestrians in case of a group. $\mathfrak{L}[i]$ is a list of pointers to the j pedestrian descriptors that form the group ($j > 1$).

6.1. Tracking and group management

At the ground-plane level the major problem to overcome is the group formation. A group is defined when a pedestrian is not visible as a single target in any of the sensor nodes. This definition allow the existence of single and compound groups. A single group is defined when a pedestrian creates a compound object with different pedestrians in each one of the sensor nodes. A compound group is defined when more than one pedestrian shares a common compound object in different sensor nodes. In both cases, the system is unable to obtain the ground-plane position of the pedestrian directly from the sensor nodes.

Correspondence matching between the pedestrian trackers and the mapped measurements from the sensor nodes is obtained using correspondence matrices. The Mahalanobis distance between the *a posterior* estimated pedestrian position and the ground-plane mapped position is used as a matching measurement. This correspondence is cross-checked by matching the image tracked objects with the projection of the *a posterior* estimated position of the head and feet of the pedestrian into the image sensor nodes (recursive projection). An example of a correspondence matrix for the binocular case is shown below.

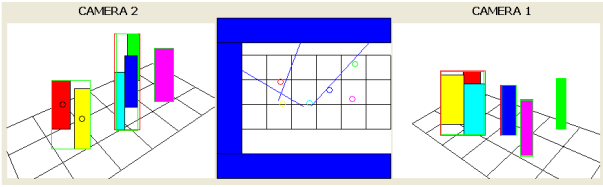


Figure 4. Ground-plane location of compound pedestrians

C_1	T_1	T_2	T_3	T_4	
\bar{P}_1	1	0	0	0	1
\bar{P}_2	1	0	0	0	1
\bar{P}_3	1	0	0	0	1
\bar{P}_4	0	1	0	0	1
\bar{P}_5	0	0	1	0	1
\bar{P}_6	0	0	0	1	1
	3	1	1	1	

C_2	T_1	T_2	T_3	
	1	0	0	1
	1	0	0	1
	0	1	0	1
	0	1	0	1
	0	1	0	1
	0	0	1	1
	2	3	1	

(26)

Four managers handled the correspondence and group formation: *split*, *grouping*, *new/lost* and *update*.

The major difference between these managers and the ones used on the single view tracking lies on the *grouping* occurrence. A split occurrence is detected when a pedestrian match more than one target. The pedestrian descriptors information stored on the compound pedestrian descriptor are recovered and new trackers are created. Color information is used to match the new targets with the pedestrian descriptors. Solving the split occurrences, it is time to handle the grouping occurrences. Analyzing the information stored in the last row of the correspondence matrix several groups can be created, representing the pedestrians grouping. The correspondence matrix for camera 1 establish four groups $G_1^1 = \{P_1, P_2, P_3\}$, $G_2^1 = \{P_4\}$, $G_3^1 = \{P_5\}$ and $G_4^1 = \{P_6\}$ while camera 2 establish three groups $G_1^2 = \{P_1, P_2\}$, $G_2^2 = \{P_3, P_4, P_5\}$, $G_3^2 = \{P_6\}$. The groups of cardinal one, like G_2^1 , G_3^1 , G_4^1 and G_3^2 , allows the recovery of the ground-plane position of the pedestrians directly from the sensor nodes, which means that the trackers of the pedestrians P_4 , P_5 , P_6 can be updated with the measurement supplied by the sensor nodes. The remaining trackers are unable to be updated directly from the sensor nodes. For these cases, a novel solution was implemented to estimate the ground-plane position of the *in group* pedestrians. Each compound *object* map on the ground-plane the target primitives p_f and p_h , defining a straight line on the ground-plane. Different groups define different lines and the estimated position of the pedestrian belonging to these groups is defined as the point that minimize the Euclidean distance to the lines. Figure 4 shows the outcome of this approach on a simulated situation considering the binocular case. Analyzing in detail what happens in this situation, the pedestrians P_1 and P_2 has grouped on the ground-plane defining a compound group whose location is obtained by the intersection of the lines defined by G_1^1 and G_1^2 , the position of P_3 is obtained by the intersection of the lines defined by G_1^1 and G_2^2 , the positions of P_4 and P_5 are obtained directly from sensor node 1



Figure 5. Pedestrians detection using the PETS04 dataset

and the position of P_6 is obtained directly from sensor node 1 and sensor node 2.

7. Performance Evaluation and Results

Several experiments in tracking multiple pedestrians has been undertaken. The performance of *part* of the image tracking process (single view tracking) was evaluated using the PETS2004 benchmark data. The *Meet walk split* MPEG video sequence was used and the results were compared with the ground-truth values available on the PETS04 database. The results are shown in figures 5-6. The image trajectories of the objects bounding boxes center of mass is superimposed on the images and the performance of the solution adopted is shown on figure 6. The image orientation of the pedestrian is the orientation of the major axes of the ellipse that was fitted to the detected pedestrian's blob.

The integration of multiple cameras to track multiple targets was also analyzed and some of the results are presented on figure 7. The green boxes represents the tracked *objects* while the blue ones represents the image projection (recursive trajectory) of the ground-plane tracked pedestrians. The red dots on the top and bottom of the blue boxes corresponds to the projection of the feet and head of the pedestrian. Figure 8

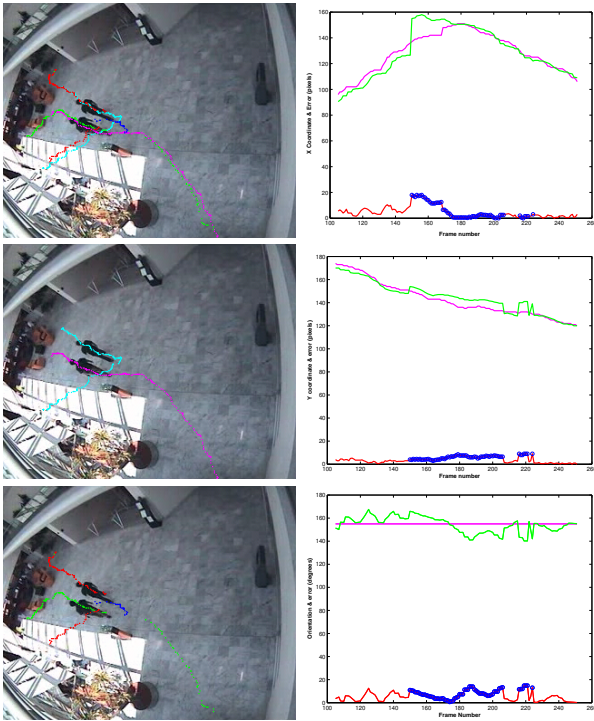


Figure 6. Image tracking of multiple pedestrians and ground-truth error evaluation (PETS04 dataset). (The image trajectories of the bounding boxes center of mass is superimposed on the images. The blue circles represents data obtained during the pedestrians grouping. *left top*: ground-truth and detected trajectories; *left center*: ground-truth trajectories; *left right*: detected trajectories. *right up¢er*:coordinates of the bounding box center of mass; *right bottom*:orientation.)

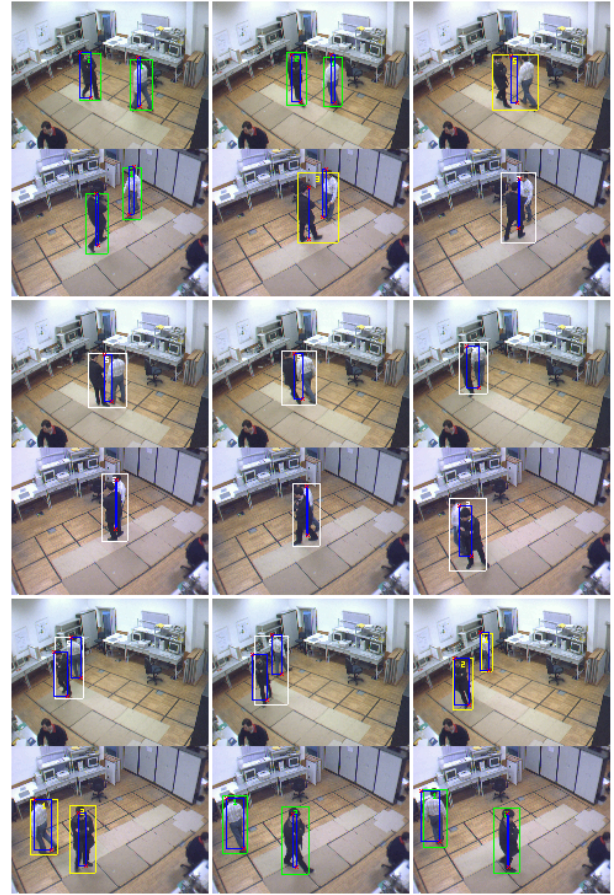


Figure 7. Tracking two pedestrians with long-term grouping

shows the ground-plane trajectory recovery for the situation presented on figure 7. The lines shown at the bottom represents the occurrence of merge situations at the image level.

Due to the nature of the proposed system, the performance of the system was also tested with the dataset 3 available from PETS2001 (figure 9). The system was tested on a small portion of the dataset, to be exact from frame 2625 to frame 2800. The homography transformation of each camera was obtained from the data supplied by PETS2001. Since the location of the cameras was not available the system is unable to estimate the real height of the pedestrians. In order to maintain the robustness of our ground-plane tracking and group management, a virtual height was assumed for all the pedestrians (175cm). The ground-plane trajectories of the pedestrians obtained by the presented system is shown in figure 9. The blue dots corresponds to the ground-plane trajectories obtained from camera 1 while the red dot corresponds to the ground-plane trajectories obtained from camera 2. The cyan dots represents the trajectory of a compound pedestrian group.

Although the good performance obtained, the system was unable to manage the situation that occurred when an image compound object was detected only in one of the images. That happened when the group of two boys was detected by camera 1 but was occluded in camera 2 by the tree. The error intro-

duced by the homography transformation for points located far from the calibration area also degraded the performance of the system.

8. Conclusions

The integration of several visual sensors for a common task of surveillance was presented. A simple and robust solution to handle image occlusion and grouping was proposed. The ground-plane pedestrian grouping and tracking was solved using a very simple solution, obtaining the ground-plane location of the pedestrian or group of pedestrians even in simultaneous camera grouping situations. Experimental results were presented with excellent results on tracking multiple pedestrians.

References

- [1] R. Collins, et al., A System for Video Surveillance and Monitoring. *CMU-RI-TR-00-12*, Carnegie Mellon University, 2000.
- [2] Bar-Shalom, Y., Fortmann, T., Tracking and Data Association. *Academic Press, Inc*, New-York 1988.
- [3] Hartley, R., Zisserman, A., Multiple View Geometry in Computer Vision, *Cambridge University Press*, 2000.

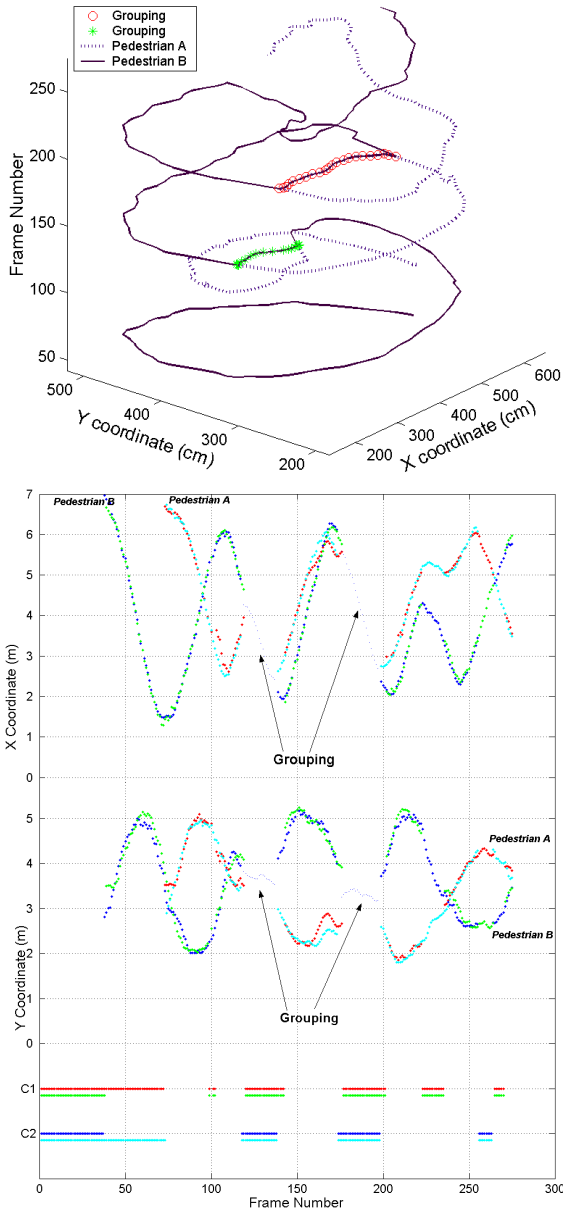


Figure 8. Ground-plane trajectory estimation

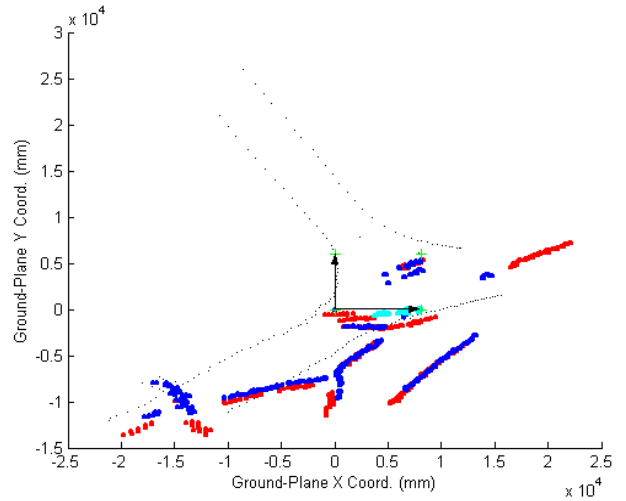
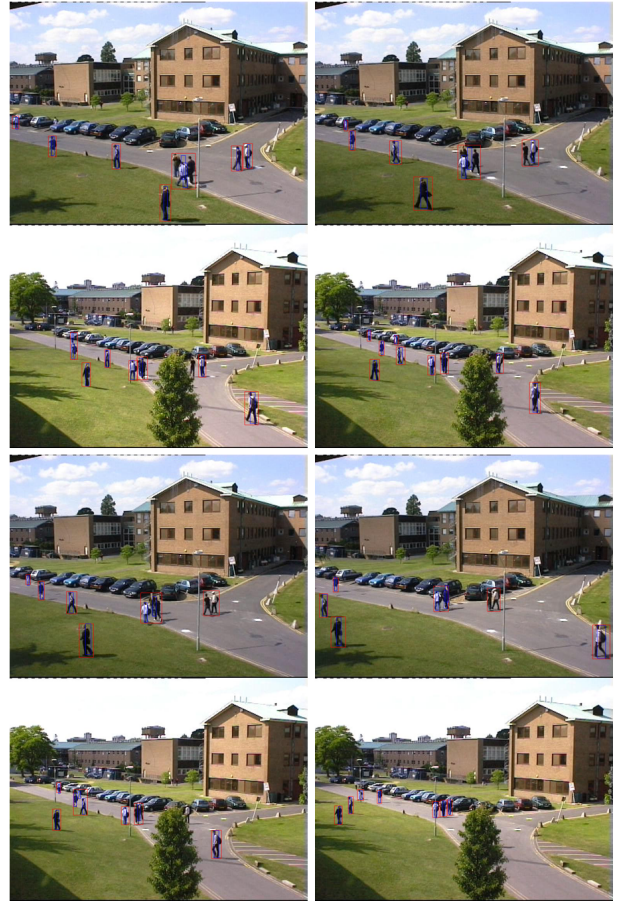


Figure 9. Tracking multiple pedestrians in complex situations(PETS2001-Dataset 3). (Bottom: Ground plane pedestrians trajectories from frame 2625 to frame 2800.)

- [4] Horprasert,T., Harwood,D., Davis,L., A statistical approach for real-time robust background subtraction and shadow detection, *ICCV'99 Frame Rate Workshop*, 1999.
- [5] Haritaoglu,I., Harwood,D., Davis,L., Hydra-Multiple people detection and tracking using silhouettes, *IEEE Workshop on Visual Surveillance*, 1996.
- [6] Pieter,J., Crowley, J., Multi-Modal Tracking of Interacting targets using Gaussian Approximations, *PETS2000*, 2000.
- [7] Swain,J., Ballard,D., Color Indexing, *IJCV*, 7:1,11-32, 1991.
- [8] McKenna,S., Raja,Y., Gong,S., Tracking Colour Objects using Adaptive Mixture Models, *Image and Vision Computing*, 17,225-231, 1999.
- [9] Black,J., Ellis,T., Multi Camera Image Tracking, *IEEE PETS2001*, 2001.

- [10] Qai,Q., Aggarwal,k., Automatic Tracking of Human Motion in Indoor Scenes Across Multiple Synchronized Video Streams, *ICCC98*, Bombay, 1998.
- [11] Zhao,T., Nevatia,R., Lv,F., Segmentation and Tracking of Multiple Humans in Complex Situations, *IEEE CVPR*, Hawaii, 2001.
- [12] McKenna,S., Jabri,S., Duric,Z., Rosenfeld,A., Tracking Groups of People, *CVIU*, 80,42-56, 2000.
- [13] Mittal,A., Video Analysis Under Severe Occusions, *PhD Thesis*, University of Maryland, 2002.
- [14] Yang,D., González-Baños,H., Guibas,L., Counting People in Crowds with a Real-Time Network of Simple Image Sensors, *IEEE ICCV03*, 2003.
- [15] Remagnino,P., Jones,G.A., Automated Registration of Surveillance Data for Multi-Camera Fusion, *ISIF*, 1190-1197, 2002.
- [16] Chang,T., Gong,S., Bayesian Modality Fusion for Tracking Multiple People with a Multi-Camera System, *In Proc. European Workshop on Advanced Video-based Surveillance Systems*, UK, 2001.
- [17] Khan,S., Javed,O., Rasheed,Z., Shah,M., Human Tracking in Multiple Cameras, *IEEE ICCV01*, 331-336, 2001.
- [18] Kang,J., Cohen,I., Medioni, G., Continuous Tracking Within and Across Camera Streams, *IEEE CVPR03*, 267-272, 2003.
- [19] Kang,J., Cohen,I., Medioni, G., Continuous Multi-Views Tracking using Tensor Voting, *Workshop on Motion and Video Computing (MOTION'02)*, 181-186, 2002.
- [20] Stein,G., Tracking from Multiple View Points: Self-calibration of Space and Time, *Image Understanding Workshop*, Nov. 1998.
- [21] Brémond,F., Thonnat,M., Tracking Multiple Non-Rigid Objects in Video Sequences, *IEEE Transaction on Circuits and Systems for Video Technology Journal*, vol.8,no.5, 1998.