CAVIAR
Context Aware Vision using Image-based
Active Recognition

D26
Network Activity
HAREM’2005

CAVIAR : D 26 Network Activity.
Date: September 2005
Author(s): José Santos-Victor (IST)
Work package WP 7 Dissemination
Document status: Version 0.1
Usage: Internal
Keywords: HAREM2005, Human Activity Recognition and Modeling Workshop
Abstract
This report describes the First International Workshop on Human Activity Recognition and Modeling (HAREM’2005) held in Oxford, September 9th, 2005. It reviews the objectives for the workshop, presents the program, lists the registered participants, and provides the papers published in the proceedings.

DOCUMENT HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason of change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Call for papers</td>
</tr>
<tr>
<td></td>
<td>1 June 2005</td>
<td>Paper submission deadline</td>
</tr>
<tr>
<td></td>
<td>1 July 2005</td>
<td>Returned paper reviews</td>
</tr>
<tr>
<td></td>
<td>20th July 2005</td>
<td>Final version due.</td>
</tr>
<tr>
<td></td>
<td>9 September 2005</td>
<td>Workshop held in Oxford, Cambridge, UK.</td>
</tr>
<tr>
<td>1.0</td>
<td>30 September 2005</td>
<td>Deliverable prepared</td>
</tr>
</tbody>
</table>
Table of contents

1. Introduction ............................................................................................................... 1
2. The HAREM2005 programme ............................................................................... 2
3. HAREM’2005 participants ...................................................................................... 4
6. HAREM’2005 Proceedings ...................................................................................... 5
1. Introduction

The CAVIAR consortium has organized an international workshop on Human Activity Recognition and Modeling, HAREN’2005, on September 9th, 2005 in Oxford in conjunction with the British Machine Vision Conference, organized at the Oxford Brookes University, Oxford, UK. The workshop webpage is http://www.isr.ist.utl.pt/~jasv/harem2005/.

Motivation

Recent advances in computer vision and learning methodologies, together with the massive increase of computational power of standard computers are enabling the deployment of a new generation of computer vision systems that go beyond traditional approaches. While most systems were designed in the past for modelling the scene geometry from video data, the challenge is now to develop systems able to detect and signal interesting events and understand, interpret and describe the observed video sequence.

One potential application of such methodologies lies in the context of video surveillance. Due to the increasing number of cameras installed in public spaces, it is no longer possible (or efficient) to have human operators continuously monitoring a multitude of video channels. Instead we need to provide computer vision systems able to process these video streams, use and learn contextual information, characterize the behaviour in a given situation and trigger an alert, only when an interesting “event” is detected. Needless to say, these systems must be designed to function and learn in an open-ended way and must have built-in, self-regulatory and reconfigurable capabilities.

The goal of HAREN’2005 is to bring together researchers in cognitive computer vision and, particularly, in the domains of surveillance and human activity recognition, with an emphasis on the topics such as low-level feature extraction and selection, modeling of attention and control; human activity recognition; cognitive and self-adaptive architectures and video interpretation.

The call for papers elicited the submission of 25 high quality manuscripts from all over the world, exceeding our expectations for a one day workshop. The Program Committee carried out the thorough task of assessing the papers technical quality and their suitability for presentation at the workshop. A total of 13 papers were accepted for presentation, covering multiple domains related to Human Activity Recognition and Modeling and providing a unified vision of the state of the art and current challenges.

The actual workshop was quite successful due to the variety and quality of the papers presented as well as the length of the discussion. Also there was clear interest for the topic of the workshop and there was an expression of interest to organize a following edition in 2006.
2. The HAREM2005 programme

The HAREM’2005 was organized in sessions addressing various levels involved in human activity analysis and recognition systems. In addition to the initially registered participants, several BMVC attendees participated in the workshop. At the end of the day a panel discussion was held, particularly focusing in the aspects of academic/industrial partnerships and viewpoints as to the deployment of video surveillance systems able to characterize behavior and activity.

Session 1: DETECTION/DYNAMIC MODELLING

A Vision System for Automated Customer Tracking for Marketing Analysis: Low Level Feature Extraction, Alex Leykin and Mihran Tuceryan, Indiana university, USA.

Modeling Behavior Trends and Detecting Abnormal Events using Seasonal Kalman Filters, James W. Davis and Mark A. Keck, Ohio State university, USA.

Classification and Prediction of Motion Trajectories using Spatiotemporal Approximations, Andrew Naftel and Shehzad Khalid, University of Manchester, UK.

Session 2: MOTION ANALYSIS AND RECOGNITION

Human Motion Recognition based on Dynamic Shape Analysis, Ning Jin and Farzin Mokhtarian, University of Surrey, UK.

Continuous Time-Varying Gesture Segmentation by Dynamic TimeWarping of Compound Gesture Models, Hong Li and Michael Greenspan, University of Kingston, Canada.

What are you looking at? Gaze estimation in medium-scale images, Neil Robertson, Ian Reid and Michael Brady, QinetiQ and Oxford University, UK.

Sessions 3 and 4: ACTIVITY RECOGNITION

Human Activity Recognition from Video: modeling, feature selection and classification architecture. Pedro Canotilho Ribeiro and José Santos-Victor, Instituto Superior Técnico, ISR, Portugal.

Activity Recognition via Autoregressive Prediction of Velocity Distribution, Miha Peternel and Ales Leonardis, University of Ljubljana, Slovenia.


Human Activity Learning and Segmentation using Partially Hidden Discriminative Models, Tran The Truyen, Hung H. Bui and Svetha Venkatesh, Curtin Univ, Australia; SRI Intl, USA.

Exploring Techniques for Behaviour Recognition via the CAVIAR Modular Vision Framework, David Tweed, Wanli Feng, Robert Fisher, José Bins and Thor List, Univ. of Edimburgh, UK.

Recognition of Action, Activity and Behaviour in the ActIPret Project Kingsley Sage, Jonathan Howell and Hilary Buxton, University of Sussex, UK.
3. HAREM’2005 participants

A total of 44 registered participants attended the workshop and about 6 non-registered participants.

<table>
<thead>
<tr>
<th>Nº</th>
<th>Name</th>
<th>Affiliation</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Buxton, Hilary</td>
<td>University of Sussex</td>
<td>UK</td>
</tr>
<tr>
<td>2</td>
<td>Caulfield, Darren</td>
<td>Trinity College Dublin</td>
<td>Ireland</td>
</tr>
<tr>
<td>3</td>
<td>Chen, Xilin</td>
<td>Chinese Academy of Sciences</td>
<td>China</td>
</tr>
<tr>
<td>4</td>
<td>Chong, Clarence</td>
<td>University of York</td>
<td>UK</td>
</tr>
<tr>
<td>5</td>
<td>Clocksin, William</td>
<td>University of Oxford</td>
<td>UK</td>
</tr>
<tr>
<td>6</td>
<td>Dente, Enrica</td>
<td>Imperial College</td>
<td>UK</td>
</tr>
<tr>
<td>7</td>
<td>Ferryman, James</td>
<td>University of Reading</td>
<td>UK</td>
</tr>
<tr>
<td>8</td>
<td>Fisher, Bob</td>
<td>University of Edinburgh</td>
<td>UK</td>
</tr>
<tr>
<td>9</td>
<td>Gabriel, Othmezouri</td>
<td>Toyota</td>
<td>Belgium</td>
</tr>
<tr>
<td>10</td>
<td>Gilbert, Andrew</td>
<td>University of Surrey</td>
<td>UK</td>
</tr>
<tr>
<td>11</td>
<td>Greenspan, Michael</td>
<td>Queen's University</td>
<td>Canada</td>
</tr>
<tr>
<td>12</td>
<td>Grossmann, Etienne</td>
<td>Univ. Kentucky</td>
<td>USA</td>
</tr>
<tr>
<td>13</td>
<td>Hashim, Rathiah</td>
<td>University of Wales Swansea</td>
<td>UK</td>
</tr>
<tr>
<td>14</td>
<td>Hongeng, Somboon</td>
<td>University of Birmingham</td>
<td>UK</td>
</tr>
<tr>
<td>15</td>
<td>Jin, Ning</td>
<td>University of Surrey</td>
<td>UK</td>
</tr>
<tr>
<td>16</td>
<td>Kagehiro, Tatsuhiko</td>
<td>University of Surrey</td>
<td>UK</td>
</tr>
<tr>
<td>17</td>
<td>Kaimakis, Paris</td>
<td>Cambridge University</td>
<td>UK</td>
</tr>
<tr>
<td>18</td>
<td>Keck, Mark</td>
<td>Ohio State University</td>
<td>USA</td>
</tr>
<tr>
<td>19</td>
<td>Keith, Bromley</td>
<td>Office of Naval Research Global</td>
<td>UK</td>
</tr>
<tr>
<td>20</td>
<td>Khalid, Shehzad</td>
<td>University of Manchester</td>
<td>UK</td>
</tr>
<tr>
<td>21</td>
<td>Lam, K. P.</td>
<td>University of Keele</td>
<td>UK</td>
</tr>
<tr>
<td>22</td>
<td>Leykin, Alex</td>
<td>Indiana University</td>
<td>USA</td>
</tr>
<tr>
<td>23</td>
<td>Lucie, Alison</td>
<td>SELEX</td>
<td>UK</td>
</tr>
<tr>
<td>24</td>
<td>Maisonnasse, Jerome</td>
<td>INRIA Rhône-Alpes</td>
<td>France</td>
</tr>
<tr>
<td>25</td>
<td>Maybank, Stephen</td>
<td>Birkbeck College</td>
<td>UK</td>
</tr>
<tr>
<td>26</td>
<td>Meng, Renee</td>
<td>University of York</td>
<td>UK</td>
</tr>
<tr>
<td>27</td>
<td>Mokhtarian, Farzin</td>
<td>University of Surrey</td>
<td>UK</td>
</tr>
<tr>
<td>28</td>
<td>Naftel, Andrew</td>
<td>University of Manchester</td>
<td>UK</td>
</tr>
<tr>
<td>29</td>
<td>Nascimento, Jacinto</td>
<td>Instituto Superior Técnico</td>
<td>Portugal</td>
</tr>
<tr>
<td>30</td>
<td>Ng, Kia</td>
<td>University of Leeds</td>
<td>UK</td>
</tr>
<tr>
<td>31</td>
<td>Oliver, Brdriczka</td>
<td>INRIA Rhône-Alpes</td>
<td>France</td>
</tr>
<tr>
<td>32</td>
<td>Peternel, Miha</td>
<td>University of Ljubljana</td>
<td>Slovenia</td>
</tr>
<tr>
<td>33</td>
<td>Rajpoot, Nasir</td>
<td>University of Warwick</td>
<td>UK</td>
</tr>
<tr>
<td>34</td>
<td>Reid, Ian</td>
<td>University of Oxford</td>
<td>UK</td>
</tr>
<tr>
<td>35</td>
<td>Robertson, Neil</td>
<td>University of Oxford</td>
<td>UK</td>
</tr>
<tr>
<td>36</td>
<td>Sako, Hiroshi</td>
<td>Hitachi</td>
<td>Japan</td>
</tr>
<tr>
<td>37</td>
<td>Santos-Victor, José</td>
<td>Instituto Superior Técnico</td>
<td>Portugal</td>
</tr>
<tr>
<td>38</td>
<td>Senanayake, Chris</td>
<td>UCL</td>
<td>UK</td>
</tr>
<tr>
<td>39</td>
<td>Sugrue, Mark</td>
<td>University of London</td>
<td>UK</td>
</tr>
<tr>
<td>40</td>
<td>Tran, Truyen</td>
<td>Curtin University of Technology</td>
<td>Australia</td>
</tr>
<tr>
<td>41</td>
<td>Tweed, David</td>
<td>Univ. of Edinburgh</td>
<td>UK</td>
</tr>
<tr>
<td>42</td>
<td>Vacek, Stefan</td>
<td>University of Karlsruhe</td>
<td>Germany</td>
</tr>
<tr>
<td>43</td>
<td>Watson, Norman</td>
<td>SELEX</td>
<td>UK</td>
</tr>
<tr>
<td>44</td>
<td>Zou, Xuan</td>
<td>University of Surrey</td>
<td>UK</td>
</tr>
</tbody>
</table>
6. HAREM’2005 Proceedings

The HAREM’2005 proceedings are included as an annex to this deliverable.
International Workshop
On
Human Activity Recognition and Modelling

HAREM’2005

Oxford, UK,
September 9th, 2005

Sponsored by EU Project IST 2001 37540
CAVIAR

Local organisation by
Instituto Superior Técnico/ Institute for Systems and Robotics
Lisbon, Portugal
HAREM’2005 ORGANIZATION

ORGANISING COMMITTEE

José Santos-Victor        Instituto Superior Técnico, Instituto de Sistemas e Robótica, Portugal.
Bob Fisher               University of Edinburgh, UK.
James Crowley           INRIA- Rhone Alpes, France.

PROGRAMME COMMITTEE

Alexandre Bernardino  Instituto Superior Técnico / ISR, (Portugal)
Aaron Bobick          Giorgia Tech, (USA )
James Ferryman        University of Reading, (UK)
Daniela Hall          INRIA Rhône-Alpes, (France)
David Hogg            University of Leeds, (UK)
Graham Jones          Kingston University, (UK)
Seong-Whan Lee        Korea University, (Korea)
Jorge Marques         Instituto Superior Técnico/ ISR, (Portugal)
Filiberto Pla         University Jaume I, (Spain)
Hiroshi Sako          Hitachi, (Japan)
Monique Thonnat       INRIA Sophia Antipolis, (France)

SPONSORS

CAVIAR Project:
Context Aware Vision using Image-based Active Recognition
http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

Instituto Superior Técnico
Instituto de Sistemas e Robótica
Lisboa, Portugal
http://www.isr.ist.utl.pt
We wish you a warm welcome to the HAREM’2005, the International Workshop on Human Activity Recognition and Modelling.

Recent advances in computer vision and learning methodologies, together with the massive increase of computational power of standard computers are enabling the deployment of a new generation of computer vision systems that go beyond traditional approaches. While most systems were designed in the past for modelling the scene geometry from video data, the challenge is now to develop systems able to detect and signal interesting events and understand, interpret and describe the observed video sequence.

One potential application of such methodologies lies in the context of video surveillance. Due to the increasing number of cameras installed in public spaces, it is no longer possible (or efficient) to have human operators continuously monitoring a multitude of video channels. Instead we need to provide computer vision systems able to process these video streams, use and learn contextual information, characterize the behaviour in a given situation and trigger an alert, only when an interesting “event” is detected. Needless to say, these systems must be designed to function and learn in an open-ended way and must have built-in, self-regulatory and reconfigurable capabilities.

The goal of HAREM’2005 is to bring together researchers in cognitive computer vision and, particularly, in the domains of surveillance and human activity recognition, with an emphasis on the topics such as low-level feature extraction and selection, modelling of attention and control; human activity recognition; cognitive and self-adaptive architectures and video interpretation.

The call for papers elicited the submission of 25 high quality manuscripts from all over the world, exceeding our expectations for a one day workshop. The Program Committee carried out the thorough task of assessing the papers technical quality and their suitability for presentation at the workshop. A total of 13 papers were accepted for presentation, covering multiple domains related to Human Activity Recognition and Modelling and providing a unified vision of the state of the art and current challenges.

We would like to thank all the authors who submitted to HAREM’2005 to tell us about their work and latest results. We are grateful to the Programme Committee for doing a remarkable job in a short time frame. Finally, we are grateful to the British Machine Vision organizers, Professor William Clocksin and Dr. Andrew Fitzgibbon, for accepting hosting the HAREM workshop immediately after BMVC and for providing all the assistance, regarding local arrangements.

The HAREM workshop was sponsored by the EU Project CAVIAR (IST 2001 37540) and the Instituto Superior Técnico/ Institute for Systems and Robotics, in Lisbon, Portugal.

We hope HAREM will provide an opportunity for fruitful scientific exchange.

José Santos-Victor
Bob Fisher
James Crowley
# CONTENTS

## DETECTION/DYNAMIC MODELLING

A Vision System for Automated Customer Tracking for Marketing Analysis: Low Level Feature Extraction, ................................................................. 1

*Alex Leykin and Mihran Tuceryan, Indiana university, USA.*

Modeling Behavior Trends and Detecting Abnormal Events using Seasonal Kalman Filters, ........ 9

*James W. Davis and Mark A. Keck, Ohio State university, USA.*

Classification and Prediction of Motion Trajectories using Spatiotemporal Approximations, .... 17

*Andrew Naftel and Shehzad Khalid, University of Manchester, UK.*

## MOTION ANALYSIS AND RECOGNITION

Human Motion Recognition based on Dynamic Shape Analysis .................................................. 27

*Ning Jin and Farzin Mokhtarian, University of surrey, UK.*

Continuous Time-Varying Gesture Segmentation by Dynamic TimeWarping of Compound Gesture Models, ................................................................. 35

*Hong Li and Michael Greenspan, University of Kingston, Canada.*

What are you looking at? Gaze estimation in medium-scale images ........................................... 43

*Neil Robertson, Ian Reid and Michael Brady, QinetiQ and Oxford University, UK.*

Human Gait Recognition with 3DWavelets and Kernel based Subspace Projections, ................. 53


## ACTIVITY RECOGNITION

Human Activity Recognition from Video: modeling, feature selection and classification architecture, ................................................................. 61

*Pedro Canotilho Ribeiro and José Santos-Victor, Instituto Superior Técnico, ISR, Portugal.*

Activity Recognition via Autoregressive Prediction of Velocity Distribution, .......................... 71

*Miha Peternel and Ales Leonardis, University of Ljubljana, Slovenia.*

Segmentation and Classification of Human Activities................................................................. 79


Human Activity Learning and Segmentation using Partially Hidden Discriminative Models, ........ 87

*Tran The Truyen, Hung H. Bui and Svetla Venkatesh, Curtin Univ, Australia; SRI Intl, USA.*

Exploring Techniques for Behaviour Recognition via the CAVIAR Modular Vision Framework, .... 97

*David Tweed, Wanli Feng, Robert Fisher, José Bins and Thor List, Univ. of Edimburgh, UK.*

Recognition of Action, Activity and Behaviour in the ActIPret Project .................................. 105

*Kingsley Sage, Jonathan Howell and Hilary Buxton, University of Sussex, UK.*
A Vision System for Automated Customer Tracking for Marketing Analysis: Low Level Feature Extraction

Alex Leykin and Mihran Tuceryan
Computer Science Department
Indiana University
Bloomington, USA
oleykin@cs.indiana.edu
tuceryan@iupui.edu

Abstract

We present the first stages of a system that tracks customers in a store with the goal of activity analysis. The ultimate goal is to provide a tool for making various marketing decisions. In this paper, we focus on the low level processing methods for determining the position of the customers in the store. We present a method to extract the low-level head coordinates to be further used for tracking customers in the crowded situations. The algorithm relies on the knowledge of image vanishing points that are used to compute a “vanishing point projection histogram” as well as to extract camera calibration parameters. Vanishing points and scale factor can be computed with the help of a simple interactive interface that we also present in this paper.

Keywords: customer tracking, background subtraction, camera calibration, vanishing point

1 Introduction

Modern businesses are relying more and more on automated analyses of their customers’ behaviors for making optimal marketing decisions. One recent trend is to give customers discount cards that can be used to track the detailed purchase histories of customers so that the restocking decisions can be optimized locally. Another example is product placement on the shelves. This is typically influenced by a variety of factors one of which is customer shopping habits. Customer interest or, in other words, how much attention customers pay to a particular product on the shelves could be important. This can be inferred from how much time they spend at a particular location looking at the shelves, possibly picking up products and examining them, etc. The goal of our system would be to analyze the activity of the customers in the stores from video clips obtained from fixed cameras mounted on ceilings. The advantage of this particular task compared to the security applications is that the analysis need not be real-time. This means that we can deploy more complex algorithms, off-line, and yet gather information about customer activity and behavior that will help with the business decisions of a store or company.

With this motivation in mind, we have implemented some techniques as the first stages of a system that will do customer activity analysis in video sequences. We have implemented the low level extraction of foreground figures (customers). We have also created a method that utilizes camera calibration information for localizing the customers’ head in the image as well as the customers’ locations in the store. The full scale tracking system and analysis of customer activity is left for future work.

In the videos taken with a stationary camera, background subtraction is a primary technique used to extract the foreground pixels. Statistical background modeling based on color distortion has been presented in [8], but a single mean for each pixel is unlikely to account for the noisiness of the background in the changing environment of the store. Our attention has also concentrated on the methods that use a mixture of gaussians to model each pixel [12]. These methods are superior to the single-modality approaches, yet they operate on the fixed number of modalities which fails to comprehensively accommodate the noise and artifacts created by video compression, such as the DivX algorithm [9].

To create the initial estimates for any tracking algorithm, some form of head position estimation has been used in related works. In [7, 16, 15] the vertical projection histogram was computed to reliably establish the location of head-candidates. The vertical projection is simply a count of the foreground pixels along the vertical column. Operating under the assumption that the humans present in the scene are in the upright walking or standing position the authors extract the local maxima in what appears to be the top of the blobs contour. Although the aforementioned approach shows promising results with the horizontally looking camera, we make an argument that it will be prone to significant distortion in the case of ceiling mounted camera if the camera extrinsic parameters are not accounted for.

That was the reason we have decided first to consider the camera information, which would help us determine where the top and bottom of the body are in each point in the image.

Substantial work exists in the field to try and use the scene constraints such as perpendicular planes or parallellopedal structures to determine the camera model [14, 13]. Parallel structures are easy to come by in the man-made
environment of the store. We have implemented, based on the work of Criminisi et. al [4], the method to extract the vanishing points. The method relies on marking in the image the parallel lines in 3D, and requires the least expertise from the user.

2 Outline of the Method

Our method consists of four major steps (figure 1): (i) background modeling and subtraction, (ii) camera modeling, (iii) head candidates detection, and (iv) human height measurement. The output from the background subtraction is the binary foreground map as well as an array of foreground blobs, each represented as a 2D contour. Camera calibration provides next stage of the system with the location of vanishing points $V_X$, $V_Y$ and $V_Z$ as well as the scale factor.

![Figure 1: Major components of the algorithm (dotted boxes refer to future work)](image)

3 Component Description

In this section we will describe the major parts of our method in more detail.

3.1 Camera Model

While building realistic human body models during the higher-level tracking stages of the system, it is important to work in 3D scene space. To accomplish this, intrinsic and extrinsic camera parameters must be obtained in order to go from image space to scene space. Many man-made environments contain rectilinear structures in the scene. We have used algorithms that extract vanishing points from the images of parallel lines in such rectilinear scene structures. All parallel lines in the image converge in the so-called vanishing point [5]. We are interested in finding the vertical vanishing point $V_Z$ as the center of intersection of the lines which point in the vertical direction. Two lines are sufficient to find $V_Z$, but in a noisy environment it is beneficial to consider more lines to achieve higher accuracy in the location of the vertical vanishing point $V_Z$. This is computed as the centroid of the intersection points of the images of all the 3D vertical lines. In our application environment there is an abundance of man-made rectilinear structures with vertical lines that can be used for that purpose (isles, boxes, markings on the floor, doors and windows).

In the calibration phase, a number of lines, parallel in space are designated manually with a help of a simple point and click interface (figure 2). Each line is represented as two endpoints $e_1 = [x_1, y_1]$ and $e_2 = [x_2, y_2]$. Prior to computing the vanishing point all line endpoints are converted into the homogeneous coordinates with the origin in the center of the image $[\frac{w}{2}, \frac{h}{2}]$, where $w$ and $h$ are the width and height of the image in pixels, respectively. The scaling factor is set to the average of image half-width and half-height $(w + h)/4$ for better numerical conditioning.
\[
e_i' = \left[ x_1 \times \frac{w}{2}, y_1 \times \frac{w}{2}, (w + h)/2 \right]
\]
\[
e_2' = \left[ x_2 \times \frac{w}{2}, y_2 \times \frac{w}{2}, (w + h)/2 \right]
\]

Then in homogeneous coordinates each line can be computed as a cross-product of its endpoints \( l = e_1' \times e_2' \).

The \( 3 \times 3 \) “second moment” matrix \( M \) is built from an array of lines \( l \) and \( V_Z \) is computed from the solution of \( M \) by singular value decomposition as the eigenvector that corresponds to the smallest eigenvalue \( [2] \).

Figure 2: VZ can be found by manually marking two or more vertical straight lines

Figure 3: Marking the objects of known height to determine the scale

### 3.2 Background Modeling and Subtraction

Video sequences from the in-store surveillance cameras are frequently compressed with MPEG-like algorithms, which normally create a periodic noise on the level of a single pixel. We have incorporated a multi-modal statistical background model based on the codebook approach implemented in [10] with a number of improvements.

Each pixel in the image is modeled as a dynamically growing vector of codewords, a so-called codebook. A codeword is represented by: the average pixel \( RGB \) value and by the luminance range \( I_{low} \) and \( I_{hi} \) allowed for this particular codeword. If an incoming pixel is within the luminance range and within some proximity of \( RGB \) of the codeword it is considered to belong to the background. During the model acquisition stage the values are added to the background model at each new frame if there is no match found in the already existing vector. Otherwise the matching codeword is updated to account for the information from the new pixel. Empirically, we have established that there is seldom an overlap between the codewords. However if this is the case, i.e more than one match has been established for the new pixel, we merge the overlapping codewords. We assume that the background noise due to compression is of periodical nature. Therefore, at the end of training we clean up the values (“stale” codewords) that have not appeared for periods of time greater than some predefined percentage frames of in the learning stage as not belonging to the background. For this as outlined in [10], we keep in each codeword a so-called “maximum negative run-length (MNRL)” which is the longest interval during the period that the codeword has not occurred. One additional benefit of this modeling approach is that, given a significant learning period, it is not essential that the frames be free of moving foreground object. The background model can be learned on the fly, which is important in the in-store setting.

As a further enhancement we eliminated the background learning stage as such to enable our system to operate dynamically. This was done by adding the \( age \) parameter to each codeword as the count of all the frames in which the codeword has appeared. Now, we can start background subtraction as soon as the majority of the codewords contain “old-enough” modalities. Typically, around 100 frames in our test sequences presented in section 4 were enough for reliable detection of the foreground objects. This improvement also allows us to perform the removal of “stale” codewords periodically and not as a one-time event. Now, to determine the “staleness” of a codeword we consider the ratio between its \( MNRL \) and it overall \( age \). We have found that when employing “stale” pixel cleanup for the heavily compressed sequences the length of the codebook required to encapsulate the background complexity within one pixel is usually under 20 codewords.

Additionally, we store the number of the last frame number \( f_{last} \) in which the codeword was activated (i.e. it matched a pixel). To make our model dynamic, we discard the codewords that have not appeared for long periods of \( time \), which can be computed as the difference between the current frame and \( f_{last} \) for any given codeword. Such instances are indicating that the interior has change, due to possibly a stationary object placed or removed from the scene, thus causing our model to restructure dynamically.

The binary mask after background subtraction is filtered with morphological operators to remove standalone noise pixels and to bridge the small gaps that may exist in otherwise connected blobs. This results in an array of blobs.
created where each blob \( b \) is represented as an array of vertices \( b_i, i = 1, \ldots, n \) in two-dimensional image space. The vertices describe the contour of \( b \) in which each adjacent pair of vertices \( b_j \) and \( b_i \) is connected by a straight line.

### 3.3 Finding Head Candidates

The surveillance cameras are typically mounted on the ceiling, more than ten feet above the ground. This can be advantageous in discriminating separate humans within a crowd. The head of a human will have the lowest chance to be occluded, therefore we pursued the goal of finding head candidates - points that represent the tops of the heads in the blob. In this section, we describe our approach in more detail.

To generate human hypotheses within a blob detected in the scene we have used a principle similar to that of the vertical projection histogram of the blob. Our novel method utilizes information about the vanishing point location we obtain from the camera during the calibration stage. The projection of the blob is done along rays going through the vanishing point instead of the parallel lines projecting onto the horizontal axis of the image.

![Figure 4: Vanishing point projection histogram](image4.png)

![Figure 5: Vanishing point projection histogram](image5.png)

In our implementation each foreground blob is represented as an array of contour vertices \( T_i \) (see figure 5), converted to homogeneous coordinates as described in section 3.1. For each \( i \) our method starts at \( T_i \) and counts the number of pixels \( h_i \) along the line \( r_i = T_i \times V_Z \) coming through the vanishing point, obtained earlier as part of camera calibration process.

Then \( r_i \) is rasterized by Bresenham’s algorithm. Notice that \( V_Z \) is an ideal point which can sometimes fall out of the image boundary or even be situated at an infinity (in the case that the 3D parallel lines are also parallel to the image plane). Therefore we needed to modify the rasterization algorithm to stop as soon as it reaches the image boundary or \( V_Z \), whichever comes first. Note that there is no risk of the process spreading to adjacent blobs, because the foreground mask is rendered for each blob from its contour independently.

The process continues even after the end of the foreground region is reached, which can be defined as the first non-foreground pixel, to allow for important contour concavities, such as arms as well as gaps that are due to camera noise (e.g. see the line originating from \( P_1 \) in 5). The last foreground pixel reached in such a manner is considered a bottom candidate \( B_i \) and the count of foreground pixels between \( T_i \) and \( B_i \) is recorded into the histogram bin \( i \). The rays where \( T_i = B_i \) are discarded as coming from the “underside” of the contour.

Resulting from this is our vanishing point projection histogram \( H = [h_i] \). We attempt to isolate local maxima in the histogram in two steps. First, the value \( h_i \) is considered a local maximum within a window if it is greater or equal of \( M \) of its neighbors on either side (figure 4 shows as an example the window of size \( M = 5 \)).

\[
h_i \geq h_j, \forall j = i \pm \frac{M-1}{2}
\]

Because this may result in a number of neighboring vertices of with equal values of \( h \) selected as local maxima, we merge all such peaks within their window \( M \) and use their average as a candidate. Notice that to represent the cyclic nature of the contour for the leftmost and rightmost bins the neighbors are wrapped around from the end or the beginning of the histogram correspondingly. Typically the window size can be determined as the total number of bins in the histogram divided by the maximum amount of candidates allowed with one blob. This number is set normally from 3 to 10 depending on the average complexity or “crowdedness” of the scene. After this stage all the local peaks \( h_i < \max(h_i)/2 \) are further removed to ensure that we are only considering the vertices from that correspond to the upper parts of the body.
3.4 Human height detection

Utilizing the same interactive approach used to obtain $V_Z$ (figure 5) we also have found $V_X$ and $V_Y$ (see section 3.1 for more details). Note that for a stationary camera this calibration procedure has to be performed only once for the entire video sequence, assuming the environment does not change. In the same manner (figure 3), the user can designate a number of vertical linear segments of known height (e.g. isles, shelves or boxes). Using the heights of the reference objects to compute the projection scale and knowing the positions in the image of head candidates with their corresponding floor locations we have employed the approach from [4, 3] to find human heights in centimeters.

4 Results and Discussion

We have tested our approach on a number video sequences from two different cameras (figure 7 (a)-(f)) mounted in a retail store chain and on the publicly available CAVIAR dataset [1] (figure 7 (g)-(l)). Some sample frames and results of the head candidates detection as well as height estimation from the test video sequences are presented in figure 7.

One of the most frequent cases of detecting false positives was occurring when there was not enough frames allotted for the background acquisition and therefore some people standing were interpreted as part of the background. When these people later moved, not only the moving person but the pixels where she used to stand are detected as a foreground objects. The background subtraction approach has given good results even under extreme lighting conditions (see (i) and (j) in figure 7).

Analyzing falsely detected head locations, we see that these are primarily due to the video compression artifacts influencing the background subtraction process. Nevertheless, the algorithm has shown robust performance with the significant levels of illumination noise, under the low-quality, real-life capturing conditions.

The false negative head candidates were primarily due to two reasons. First, parts of the foreground region become separated from the body or sometimes a part of the shadow is considered as a separate body, and this causes a false candidate to be detected (see (k) in figure 7). We believe that human shape modeling may help solve this problem. A second factor that badly influences the detection is when the heads are not pronounced enough to create a local maximum in the histogram (see (l) in figure 7). This problem can be attended in the future by color and texture analysis within the blob.

![Figure 6: Algorithm performance evaluation. This graph shows the number of frames when 1, 2, or 3 heads were detected. The true number of heads is 2.](image)

To partially evaluate the quality of the results we have analyzed a number of detected head candidates in the sequences with two people, that were detected as a single blob (Figure 6). The evaluation shows that the outputs from our methods can be used at the initialization stage of tracking algorithm. To further evaluate the quality of our method candidate hit/miss and average error analysis based on their coordinates may be required.

5 Future Work

We intend to further extend the method in a number of ways. One shortcoming of the proposed head candidate detection scheme is that it will not discriminate between the projections of the bodies superimposed on top of one another if the heads have a significant overlap. To make the method fully applicable for complex scenes where sometimes a large group of people may be represented after background subtraction as a single blob, we plan to combine this approach with the foreground color and texture analysis. Imagine a scenario where one person’s silhouette is included as a part
of another human contour. By detecting the facial color or head outline we can improve the probability of correct head detection.

To build upon the results presented here we plan to implement the future stages of tracking and activity recognition shown in figure 1. Having obtained the image coordinates of the head and the foot points $T_i$ and $B_i$, and knowing the three vanishing points and well as scaling coefficients, the camera fundamental matrix can be obtained thus rendering tracking in 3D space possible. Specifically we are interested in localizing the position of each customer on the floor plane. This way the position and orientation of body ellipsoid can be combined with multiple-view color representation for more reliable color tracking [11]. This kind of tracking will provide a wealth of information of marketing oriented customer activity analysis, such as customer flow in designated isles or areas or location of shopping groups as in [6]. Another outcome is detecting events that show “lingering,” when the customer is stopping for a prolonged amount of time in front of certain items or sections. Combined with the customer counting camera installed at the entrance this method will allow to compute the percentage customer distribution in the different areas of the store as well as provide important clues into the “conversion rate” analysis (the ratio of the amount of purchases to the total number of customers).

References


Figure 7: (a) - (l) Head candidates from test frames. Left image is the original frame. On the right image red represents foreground mask, small black dots indicate the locations of $T_i$ and $B_i$; blue ellipses are fitted with $T_iB_i$ as the major axis; (m) and (n) Height detection: brown plates contain height mean and variance for each ellipse.
Modeling Behavior Trends and Detecting Abnormal Events using Seasonal Kalman Filters

James W. Davis  Mark A. Keck
Dept. of Computer Science and Engineering
Ohio State University
Columbus, OH 43210 USA
{jwdavis,keck}@cse.ohio-state.edu

Abstract
We present a seasonal state-space model using Kalman recursions to learn and predict structured behavior patterns. The model is employed to detect events using the learned expectations of typical scene activity. Abnormal events are detected when a new observation exceeds the confidence range for the predicted behavior. We demonstrate the approach for modeling (over multiple days) the number of pedestrians in a scene, door access card-reader activity, and the departure rate of vehicles from a parking garage. We then use the model to detect abnormal events in each of these domains. The proposed framework provides a single long-term model by exploiting the natural trends in daily human activity.

1 Introduction
An important aspect of any intelligent surveillance and monitoring system, beyond person detection and activity recognition, is the ability to automatically learn/discover patterns or trends of scene behavior over time and to identify abnormal/anomalous events. As most video cameras are positioned to monitor a fixed area over long periods of time (even several years), it is not unreasonable to insist that the system should learn the activity trends by observation and adapt itself through experience (modifying its model parameters). Statistical modeling of activity trends over long periods can be employed to automatically derive expectations of scene behaviors. Then, any observed behavior that violates the scene expectation can be considered to be an abnormal event. The main advantage of a statistical approach is that individual detectors for all possible abnormal events do not need to be hand-crafted.

Due to our highly scheduled lifestyles, human activity tends to exhibit the pervasive phenomenon of “seasonality”, referring to the tendency to repeat patterns of behavior across time (see [2]). For example, on a college campus many people will be seen between classes and few during class times (except for the occasional latecomers). In Fig. 1(a), we show a plot of the number of people present in a particular scene on a campus from 9–10am (sampled every 15 seconds) on a “typical” Monday. In Fig. 1(b) we show the mean ±3 SD for each time step computed from multiple Mondays. There is a fairly large variation at any particular time, however there does exist a structured pattern across the hour (with a peak at the time between classes, 9:20–9:30am). This pattern is especially visible in the median-filtered version of Fig. 1(a), as shown in Fig. 1(c). To represent and predict such behavior patterns for event detection, we will present a new framework and show that both the natural seasonal trend and variation at each time step must be considered.

Our approach is to employ a seasonal state-space model for learning, predicting, and validating expectations of typical scene activity over long periods of time. The method is based on a dual seasonal time series model to accommodate the underlying behavior pattern and the variation at each time step. We detect an abnormal event when a new observation exceeds the confidence range for the predicted (expected) behavior. We demonstrate the applicability of the approach on multiple data sets and compare the results to alternative methods that use a separate model for each time step.

The remainder of this paper is described as follows. We begin with a review of related work (Sect. 2). Next we describe the background and formulation for the proposed seasonal framework (Sect. 3). We then present experimental results (Sect. 4). Lastly, we conclude with a summary of the research and describe future work (Sect. 5).

2 Related Work
To model activity patterns and detect abnormal events, several graph-based and trajectory-based approaches have been proposed. A Dynamic Bayesian Network was employed in [10] to model temporally-correlated events for detecting typical and atypical behaviors. In [8], the trajectories of people were used to build a semantic scene model (entry zones, paths, junctions) from which activity expectations could be extracted. In [7], a statistical model of pedestrian trajectories was used to learn typical paths and to identify incidents of unexpected behavior. In [3], abnormal patterns
were recognized using chronicle models (time constrained events labeled by situations). A polygonal shape configura-
tion and its deformation for path behaviors of people were examined for abnormal changes in [9]. Anomalous office
activities were detected in [1] by identifying inputs having a low likelihood to an entropically-estimated HMM.

Many of the above methods can be used to learn interesting behavior patterns, but unlike our method they generally
ignore how the occurrence of these behaviors change with respect to time (e.g., over hours or days). For example, a
particular behavior can have event/non-event status depending on the time of day (e.g., person present at 3am vs.
3pm). The novelty of our proposed approach is the exploitation of the natural seasonality of behaviors to learn proper
time-based expectations for detecting abnormal events. Furthermore, we show that examination of both the behavior
pattern and variation are necessary to properly model and evaluate count-based detection data.

3 Seasonal Forecasting

The classical decomposition model for a seasonal time series is an explicit representation composed of the underlying
trend, seasonal variation, and irregular (random) noise components [2]. An advantage to the classical decomposition
model is that we have a representation for an underlying process model of the time series.

The classical decomposition model for a univariate observation $y$ at time $t$ is given by

$$y_t = m_t + s_t + n_t$$  \hspace{1cm} (1)

where $m_t$ is the trend component, $s_t$ is the seasonal component, and $n_t$ is noise. The seasonal component $s_t$ has a period
of $d$ with the properties $s_{t+d} = s_t$ and $\sum_{i=0}^{d-1} s_{t+i} = 0$ (i.e., zero mean). The choice of the period length $d$ is an open
parameter that depends on the application (frequency analysis may offer suggestions) and controls the granularity and
speed of the system. However, the onset and duration of the period is not critical to the framework. The selected time
window for the period need only be consistent (e.g., from day to day).

In Fig. 1(a), the trend $m_t$ is a constant value ($\sim 4$) with the seasonal values having a unimodal rise-and-fall pattern
over the period (with the addition of noise). Once the three components have been estimated (from training data), the
properties of the time series can be used to predict a future observation $y_{t+n}$ (n-step prediction) by running the model
forward in time. This will be useful for detecting events.

3.1 Seasonal Kalman Filter (SKF)

A state-space representation of the classical decomposition model can be used to avoid the deterministic strictness
of the components by allowing the trend and seasonal components to evolve randomly in a recursive manner [2]. A
state-space model for a time series $Y_t$ (potentially multivariate) consists of two fundamental equations:

$$Y_t = HX_t + W_t$$  \hspace{1cm} (2)

$$X_{t+1} = GX_t + V_t$$  \hspace{1cm} (3)

The observation equation (Eqn. 2) gives $Y_t$ as a linear function of the state variable $X_t$ plus measurement/observation
noise $W_t$, and the state equation (Eqn. 3) determines the next state $X_{t+1}$ from a linear function of the current state
$X_t$ plus process noise $V_t$. Typically, the noise is treated as independent of time (dropping the subscript $t$), and the
covariance noise matrices are defined as $E[WW^T] = R$ and $E[VV^T] = Q$.

The univariate decomposition model (Eqn. 1) can be formulated in a recursive state-space model (here with
$Y_t = y_t$) as follows. First, the $d$-dimensional state $X_t$ is set to

$$X_t = \begin{bmatrix} m_t & s_t & s_{t-1} & \cdots & s_{t-d+2} \end{bmatrix}^T$$  \hspace{1cm} (4)
consisting of the trend \((m_t)\) and the \((d-1)\) most-recent seasonality values \((s)\). The corresponding state and observation matrices are given by

\[
G = \begin{bmatrix}
1 & 0 & 0 & \cdots & 0 & 0 \\
0 & -1 & -1 & \cdots & -1 & -1 \\
0 & 1 & 0 & \cdots & 0 & 0 \\
0 & 0 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & 1 & 0 \\
\end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \end{bmatrix}
\]

In this system, \(G\) is a state transition matrix, which retains the trend and updates the seasonality values using the seasonality constraints (with the last \(d-1\) seasonal values). The matrix \(H\) then sums the trend and most current seasonality value to get the observation (as in Eqn. 1).

With initial values for the state and noise covariances, we can use Kalman recursions to optimally estimate the state sequence \(\{X_t\}\) for a given observation sequence. The Kalman recursions slowly adapt the model to changes in the trend and seasonality over time by incorporating new observations into the model.

### 3.2 Event Detection

The Kalman recursions for the state-space model also propagate an error covariance \(P_t\) for each state \(X_t\). When we encounter a new observation, we can use the estimated state error covariance as a means to set the confidence level/bound (e.g., 90% confidence) for detecting outliers (abnormal events) from the predicted observation.

To transform the state error covariance \(P_t\) into a covariance measurement in the observation space, we use

\[
C_t = HP_tH^\top + R
\]

We then use the Mahalanobis distance to determine the statistical match of the actual observation \(Y_t^{obs}\) to the model prediction \(Y_t^{pred}\). The general (multivariate) Mahalanobis distance is given by

\[
D = \sqrt{(Y_t^{obs} - Y_t^{pred})^\top C_t^{-1} (Y_t^{obs} - Y_t^{pred})}
\]

The distance \(D\) is given in standard deviations (SD), therefore we can statistically threshold the distance (e.g., >3 SD is a statistical outlier).

The distance calculation could be applied to each new observation \(Y_t^{obs}\) (at each time step) to detect abnormal events, followed by an immediate update of the model (with the new observation). However, since we are employing seasonality constraints in the model, we instead employ a seasonal forecast of the entire period and use the state error covariance \(P_t\) computed for the first time step as the covariance for the entire period. We only update the model and recompute \(P\) after the entire period has been evaluated.

This method is used for two reasons. First, the model update with Kalman recursions (including matrix inversions) can be computationally taxing when the length of the period is large. We can however update the model after the period on a secondary processor to ensure real-time performance (while the new period is being evaluated on the primary processor with a different model). Second, if we blindly incorporate severe outliers into the model as they are encountered, the remaining predictions for the period can be adversely affected. We therefore incorporate the observations into the model only after the entire period has been examined, and also limit the influence of any outliers to \(\pm 3\) SD of the expected value (to allow only slow changes over time). The model prediction step for the next occurrence of the target period can be performed in real-time. For the experiments in this paper, the total computation time for predicting the following period and updating the model was typically only a few seconds.

### 3.3 Extension for SKF Modeling of Activity Counts

The above univariate SKF formulation is well-suited to seasonal patterns, however there is a problem if it is applied directly to the particular class of seasonal count data shown in Fig. 1 (and in other domains). In that data, there is a fairly large variation of counts at each time step (across multiple days). Thus, after initialization and training, a new sequence of observations need only be individually within the wide tolerance at each time step to be considered “typical” (see Fig. 1(b)), and thus no strict consideration of the overall “pattern” for the period is enforced. This would certainly cause a problem if we instead employed a brute-force approach of using an individual mean-variance model (or Kalman filter) for each time-step.

Consider the case for a new test period of observations with a low activity count between 0-3 at each time step. In relation to Fig. 1(b), at best only a few outliers near the bump in the time period will be detected as atypical. Individually, most observations may be within the wide bounds of the model, but collectively there is no rise-and-fall seasonality across the period as expected. One may be tempted to first filter the counts over the period (to reduce the variation) and then model the smoother pattern. But unfortunately this would smooth out and ignore any brief, yet
meaningful, spikes or vacancies in the count data. Such rapid events do in fact occur and could be especially prevalent in time-lapse monitoring.

Our approach is to simply employ two parallel SKF models to account for both the smoothed and raw counts, allowing us to better model the seasonality with the smoothed counts yet still detect brief spikes/vacancies using the raw counts. Due to the brief and prominent nature of spikes/vacancies, we found that the use of a median filter is more appropriate than a linear filter (average or Gaussian) for count data, as the surrounding filtered data is not corrupted.

The key insight into modeling and detecting events from count data is that both the raw counts and the smoothed counts must be examined. In our method, an event is detected if an outlier is found in either of the raw/smoothed SKF models (using Eqn. 7). Simple OR-ing is not the only form of a dual (raw, median) SKF model one can employ; it is only necessary that both types of data are evaluated.

### 3.4 Model Initialization

For initialization of the SKF model (either raw or median), one needs estimates of four parameters: \( \mathbf{X}_0 \), \( \mathbf{P}_0 \), \( \mathbf{Q} \), and \( \mathbf{R} \) (defined in Sect. 3.1). For the initial state, \( \mathbf{X}_0 \), we calculate the mean for the first period of the training data and assign it to \( m_t \), and then use the most recent \((d - 1)\) mean-subtracted data values as the seasonality estimates for \( s_t - d + 2 \) through \( s_t \). For the initial state error covariance \( \mathbf{P}_0 \), we use a diffuse prior of \( \mathbf{P}_0 = 10^5 \mathbf{I} \) (begins large and reduces during training).

To estimate the noise parameters, we first compute the measurement noise covariance \( \mathbf{R} \) from training data using the differences between the automatic system detection counts and ground-truth manual counts when available. When not available, we assume the noise is mostly process noise and set the measurement noise to a small value (e.g., \( \mathbf{R} = \mathbf{I} \)). Similarly, the process noise covariance \( \mathbf{Q} \) is estimated from training data (manual counts) for each model (raw, median) using the period-long mean trend differences and the mean-subtracted seasonal differences at each time step (across multiple periods). We average the variances across the time steps to give a single seasonal noise variance. When ground truth is not available, we perform the same operations using the detected counts. We note that an alternate maximum likelihood noise estimation process could also be employed for these estimates [2].

### 4 Experiments

To evaluate the proposed SKF framework, we tested the modeling and event detection approaches with synthetic and real data sets. We also present results comparing the approach to alternate methods.

#### 4.1 Experiments with Synthetic Data

We created a set of synthetic data similar to the person-count data shown in Fig. 1 to initially test the approach. We first employed a Gaussian rise-and-fall pattern stretched over a period of \( d = 240 \) (\( \sigma = 24 \)). Next the pattern was shifted up by a constant trend value (\( \sim 4 \)). To incorporate the noise, we began by substituting Eqn. 3 into Eqn. 2 (assuming \( \mathbf{W} \) and \( \mathbf{V} \) are uncorrelated) to yield

\[
\mathbf{Y}_t = \mathbf{H} \mathbf{G} \mathbf{X}_{t-1} + \mathbf{\omega}
\]

where \( \mathbf{\omega} = \mathbf{H} \mathbf{V} + \mathbf{W} \) and \( \mathbf{\Omega} = E[\mathbf{\omega} \mathbf{\omega}^\top] = \mathbf{H} \mathbf{Q} \mathbf{H}^\top + \mathbf{R} \). To initialize the model, we used the \( \mathbf{Q} \) and \( \mathbf{R} \) covariances estimated (using the method provided in Sect. 3.4) from three actual count periods. The noise \( \mathbf{\omega} \) was added to the raw synthetic data (the data could also be clipped at zero) and the corresponding median data was formed using a 12-tap causal median filter.

We trained the SKF models (raw, median) using 5 synthetic periods and then analyzed the model on three periods of new synthetic test data. First we tested a new period generated using the same parameters employed to create the synthetic training data. As expected the model closely matched the new test data. The largest deviation found (using Eqn. 7) in the test period was 2.7 SD. Next we applied 4 event spikes (using a constant \( \pm 10 \) SD of the noise \( \mathbf{\omega} \)) to a new test period, as shown in Fig. 2(a). Using a threshold of 3 SD in Eqn. 7, the 4 event spikes (and only those 4) were detected as events. The median-filtered values were unaffected (as desired) by the events (see bottom plot of Fig. 2(a)), and thus the events were detected solely in the raw count model.

Next, we tested a uniform count of 0 (inactivity) over the 240 samples. The results are shown in Fig. 2(b), in which 61 samples were identified as outliers (using the same 3 SD threshold as before). In this example, most of the outliers were detected with the median model, but only a few (8) of the events were found in the raw model, thus illustrating the need for the median-filtered data. We also added several normal periods after the inactivity period to determine how long the models would take to recover from the outliers after using the constrained update process (as explained at the end of Sect. 3.2). The model required 2 periods of normal activity (after the inactivity period) to stabilize and report absolutely no events (required 3 periods with unconstrained updates).
4.2 Application Domains

We next present results for three real security application domains: person presence, door access, and vehicle exits. For each of the domains, we present results of our SKF framework and compare to alternative methods. The first competing model is a simple mean-variance (MV) model applied to both the raw and median counts at each time step (across multiple periods). The second model is also a mean-variance model, but in this case a single variance is employed for all time steps (the average variance over all time steps). We refer to this model as mean-single-variance (MSV). Our SKF model makes a similar variance assumption for the period (as stated in Sect. 3.2).

To provide quantitative results, we collected a perceptual labeling of events in the data for each domain. A scoring system in which different humans marked the events in the testing data was used. The scorers were presented the raw/median count sequences of the training data, and were asked to mark each point of the raw/median counts in the testing data as “normal” or “abnormal”. We then took the majority vote at each time step across the scorers and computed the accuracy (ratio of the number of correct results to the total number tested) of the different algorithm results.

4.2.1 Pedestrian Activity

In this experiment, we show the capabilities of our technique to model the number of people in an outdoor scene during a one-hour period for three different days of the week.

We employed a database of several thermal surveillance video sequences of pedestrian traffic recorded on a university campus from 9-10am (time-lapse recorded at 1 frame every 15 sec.) on several Mondays, Wednesdays, and Fridays at a particular location (15 days, 3600 frames total). The number of pedestrians in each image were manually counted for estimating the initial parameters of the models (see Sect. 3.4). The manual counts for one Monday are shown in Fig. 1(a).

To automatically detect the pedestrians in each image, we employed the two-stage template-based method of [4]. The approach initially performs a fast screening procedure using a generalized thermal contour template [5, 6]. Next an AdaBoosted ensemble classifier using automatically tuned filters is employed to test the hypothesized pedestrian locations. This method was trained using a sample of 114 frames from the video collection, and then the detector was used to count the number of pedestrians in all of the images. An example thermal image showing the detection results is shown in Fig. 3(a).

We assigned a separate raw/median SKF model to each day of the week (e.g., only for Mondays) with $d = 240$. Though the overall pattern in each day of the week was somewhat similar, we found that Fridays during that time period were different enough from Mondays and Wednesdays to warrant a separate model (fewer people were present in the scene on Fridays). We trained the models (for each day of the week) using 3 typical periods (with manually-counted pedestrians). The noise covariances $Q$ and $R$ were estimated from the training data, and a 12-tap causal median filter was used for smoothing. To set the detection threshold for the raw (or median) model, we tested each training day in parallel and selected an outlier threshold such that no events occurred on any of the training data. The event thresholds (raw/median) were 2.0/1.5, 4.5/1.0, and 3.0/1.0 SD for the Monday, Wednesday, and Friday models, respectively.

We tested each of the SKF models with two new typical test periods. We additionally tested the Monday model with data collected on a holiday, as well as with a period in which we placed a synthetic spike of counts simulating a
tour group passing through the scene (since we had no true spikes in this recorded data set). We also trained and tested the alternative MV and MSV approaches with the exact same training data and thresholds.

We present the accuracy for the methods (including raw-only and dual raw-median versions) in the top row of Table 1. In this experiment, the SKF models outperform the competing models. Although in this case the SKF only gains slightly in accuracy when including the median counts, we can see clearly that the SKF outperforms the MV models by at least 8%. The SKF models are comparable with the MSVraw model, but are slightly better. It is also interesting to note that in this case the MV and MSV raw models have a decrease in accuracy when the median-filtered data is included, whereas the SKF results slightly increase. This is because the variance envelope around the median-filtered data is so tight that many false positives occur in the MV/MSV models with respect to the perceptually-labeled events.

4.2.2 Card-reader Activity

We next modeled the card-reader door activity in a Computer Science building over four weeks in day-long periods (from 12:00am–11:59pm). Each lab door in the building has a card-reader sensor (see Fig. 3(b)) that logs access information in a database whenever the door is opened. We employed only the timestamp information (of all doors) in the database for our experiments, and binned the timestamps into ten minute intervals ($d = 144$). An example day-long period is shown in Fig. 4(a).

Our task again is to model the seasonal behavior and detect abnormal events. Interestingly, due to this experiment, we found that the sensor logging mechanism would sometimes delay in reporting door accesses, which caused certain large spikes at times (a backlog of entries would all receive the same timestamp). An example of this anomaly is shown in Fig. 5(a). This provided an excellent testbed for the approach. We used three weeks (fifteen days) of “typical” data (i.e. no holidays, weekends, atypical spikes, or inactivity) to train the model. We then tested the SKF models on five new periods. We set the thresholds to 3.0/1.0 SD for the raw/median detections respectively. We similarly trained the MV/MSV models and used the same thresholds.

The comparative results are shown in the middle row of Table 1. From these results, we can see that the accuracy of the MV model is poor (in the 66% range) regardless of whether or not the median-filtered data is considered. The MSV model performs slightly better, with over 70% accuracy, but again gains no significant improvement with the median-filtered data. The median counts here had larger variances than those in the first experiment, and therefore did not introduce as many false positives as in the previous experiment. However, both of these methods do incur more false positives when the median data is included (from 18 to 211 for MV and 6 to 99 for MSV), but this did not decrease the accuracy of the MV and MSV models because the number of false negatives was decreased. A single SKF using only raw counts performs comparably to the MSV model. But when the median-filtered data is included, the dual SKF model outperforms the nearest competing model by over 13%.

<table>
<thead>
<tr>
<th></th>
<th>MVraw</th>
<th>MV_dual</th>
<th>MSVraw</th>
<th>MSV_dual</th>
<th>SKFraw</th>
<th>SKF_dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian</td>
<td>.8094</td>
<td>.5958</td>
<td>.8828</td>
<td>.6766</td>
<td>.8854</td>
<td>.8859</td>
</tr>
<tr>
<td>Door</td>
<td>.6569</td>
<td>.6669</td>
<td>.7264</td>
<td>.7292</td>
<td>.7236</td>
<td>.8569</td>
</tr>
<tr>
<td>Vehicle</td>
<td>.5938</td>
<td>.7500</td>
<td>.6250</td>
<td>.9375</td>
<td>.5104</td>
<td>.7813</td>
</tr>
</tbody>
</table>

Table 1: Comparison of accuracy for SKF, MV, and MSV frameworks on the different data sets.
4.2.3 Vehicle Activity

The final application consisted of monitoring a parking garage on a university campus and modeling the number of vehicles exiting the garage over a two-hour period. A color video camera was placed high above the exit gates to detect the vehicles. An example image is shown in Fig. 3(c). We collected data over a two-hour period between 3:30–5:30pm, sampled at 1 FPS. We then did a simple region-based background subtraction from a median background model within two small exit lane locations (marked in Fig. 3(c)). We detected a vehicle in a lane region when more than 40% of its pixels were detected as foreground. To keep from counting the same vehicle in consecutive frames, we used the heuristic that after a lane region has become active, it must become inactive before another vehicle exits. We then binned the data into 2.5 minute intervals ($d = 48$). An example training period is shown in Fig. 4(b). We trained our model using three days and tested it using two new periods (one typical, one holiday).

Results for all three models are shown in the bottom row of Table 1. In this case the dual MSV model achieves the maximum performance. This occurs because the holiday period was perceptually marked as entirely abnormal, and the past problem with the dual MSV model (usually producing many false positives) becomes an advantage. The MSV model detected every perceptually marked outlier in the holiday sequence, whereas our model did not detect any events in the first 15-20 samples of the period (see Fig. 5(b)). This was mainly due to a single large variance ($C$) computed and employed for the entire period by our model. The MSV approach also uses a single variance, but it was much smaller in this experiment.
5 Summary and Conclusions

We presented a seasonal state-based model for learning, predicting, and validating longer-term behavioral patterns from count data. The approach is based on a recursive state-based model containing the trend, seasonal variation, and noise over a time period. Predictions for the following period and their confidence are used to detect deviations (abnormal events) from the expected behavior.

We first showed the performance of the model using synthetic data, and then tested the method with pedestrian detection counts, entry/exit logs of door activity in a building, and the number of vehicles exiting a parking garage. Using manually-labeled test data, we compared the framework with alternative mean-variance approaches. In most of the experiments the SKF framework achieved higher accuracy of detection over the competing models (lower performance in the last experiment was mainly due to a large variance employed for the period). Overall, the SKF framework was able to effectively model the seasonality in both the raw and median counts, both of which are necessary for a consistent model of behavior.

In future work, we plan to extend the approach to include action recognition labels (e.g. running, walking) in a multivariate prediction of seasonal presence and action. As seasonal behavior patterns are quite common in our society, we expect this model to be useful for further study in relation to automatic surveillance and monitoring systems.

Acknowledgements

This research was supported in part by the National Science Foundation under grant No. 0236653.

References


Classification and Prediction of Motion Trajectories using Spatiotemporal Approximations

Andrew Naftel & Shehzad Khalid

School of Informatics
University of Manchester
Manchester, M60 1QD
a.naftel@manchester.ac.uk; s.khalid-2@postgrad.manchester.ac.uk

Abstract

A new technique is proposed for classification and similarity retrieval of video motion clips based on spatio-temporal object trajectories. The trajectories are treated as motion time series and modelled using orthogonal basis polynomial approximations. Trajectory clustering is then carried out to discover patterns of similar object motion behaviour. The coefficients of the basis functions are used as input feature vectors to a Self-Organising Map which can learn similarities between object trajectories in an unsupervised manner. Clustering in the basis coefficient space leads to efficiency gains over existing approaches that encode trajectories as discrete point-based flow vectors. Experiments on pedestrian motion data demonstrate the effectiveness of our approach leading to accuracy improvements in trajectory prediction. Applications to motion data mining in video surveillance databases are envisaged.

1 Introduction

An increasing number of systems are now able to capture and store data about object motion such as those of humans and vehicles. This has acted as a spur to the development of sophisticated content-based visual data management techniques. General purpose tools are now urgently required for motion data search and retrieval, discovery and grouping of similar motion patterns, detection of anomalous behaviour, motion understanding and prediction.

Much of the recent research focus has been on representation schemes for motion indexing and retrieval [1]-[11]. This work presupposes the existence of some low-level tracking scheme for reliably extracting object-based motion trajectories. A description of relevant tracking algorithms is not within the scope of this paper but recent surveys can be found in [12, 13]. The literature on trajectory-based motion understanding and pattern discovery is less mature but advances using learning Vector Quantization (LVQ) [14], Self-Organising Feature Maps (SOMs) [15, 16], hidden Markov Models (HMMs) [17], and fuzzy neural networks [18] have been reported. Most of these techniques attempt to learn high-level motion behaviour patterns from sample trajectories using discrete point-based flow vectors as input to the learning phase. For realistic motion sequences, convergence of these techniques is slow and the learning phase is usually carried out offline. This is due to the high dimensionality of the input feature space. In this paper, we introduce a dimensionality reduction technique for learning trajectory patterns based on a spatio-temporal modelling scheme previously used for time series indexing. The resulting efficiency gains make this approach feasible to implement online.

Related work within the temporal database community on approximation schemes for indexing time series data is highly relevant to the parameterisation of object trajectories. However, computer vision researchers have been slow to appreciate the relevance of this work. For example, spatio-temporal trajectories have been successfully modelled using discrete Fourier transforms (DFT) [19], discrete wavelet transforms (DWT) [20], adaptive piecewise constant approximations (APCA) [21], and Chebyshev polynomials [22], to name but a few.

In this paper, we aim to apply time series indexing of spatiotemporal trajectories to the problem of trajectory classification and show how to learn motion patterns by using the indexing scheme as an input feature vector to a neural network learning algorithm. The remainder of the paper is organized as follow. We review some relevant background material in section 2. In section 3, we present our trajectory modelling approach. The algorithm for learning trajectories is then presented in section 4 within the framework of a self-organising map. This is applied in the context of clustering motion trajectories and experimental results for a pedestrian object
tracking database are reported in section 5. The paper concludes with a discussion of the advantages of our proposed technique over competing approaches and outlines further work.

2 Review of Previous Work

Motion trajectory descriptors are known to be useful candidates for video indexing and retrieval schemes. Previous work has sought to represent moving object trajectories through piecewise linear or quadratic interpolation functions [1, 2], motion histograms [4] or discretised direction-based schemes [3, 8, 9]. Spatiotemporal representations using piecewise-defined polynomials were proposed by Hsu [6], although consistency in applying a trajectory-splitting scheme across query and searched trajectories can be problematic. Affine and more general spatiotemporally invariant schemes for trajectory retrieval have also been presented [5, 7, 10]. The importance of selecting the most appropriate trajectory model and similarity search metric has received relatively scant attention [11].

In addition to polynomial models, a wide variety of basis functions have been used to approximate object trajectories [19, 20, 22]. Efficient indexing schemes can then be constructed in the coefficient space of the basis functions. These schemes have been compared with respect to search pruning power, CPU and I/O efficiency costs [21, 22]. It is surprising to find that many of these candidate time series indexing schemes have not yet been applied to the problem of motion data mining and trajectory clustering. Recent work has either used probabilistic models such as HMMs [17] or discrete point-based trajectory flow vectors [14, 16, 18] as a means of learning patterns of motion activity. Point-based flow vectors generally consist of spatial coordinates augmented by instantaneous object velocities and accelerations. These can be normalised to account for variations in trajectory length.

The contribution of this paper is to show that a trajectory-encoding scheme based on input feature vectors consisting of basis function approximation coefficients can be used to learn motion patterns more effectively. Hence, clustering and classification processes can be carried out more efficiently in the basis coefficient space.

3 Trajectory Representation

3.1 Least Squares Polynomials

The output of a motion tracking algorithm is usually a set of noisy 2-D tracker points \((x_i, y_i)\) representing the object’s motion path over a sequence of \(n\) frames, where \(i = 1, \ldots, n\). Often the representative point is taken to be the centroid or edge midpoint of the object’s minimum bounding rectangle. The motion trajectory can be considered as two separate 1-dimensional time series, \(<t_i, x_i>\) and \(<t_i, y_i>\), the horizontal and vertical displacement against time where \(t_1 < \ldots < t_n\). We consider three alternative trajectory models: Least Squares (LS), Chebyshev polynomial approximations (CS) and discrete Fourier transform (FS). LS polynomials are suitable for modelling simple motion trails in the spatial domain, e.g. vehicles moving uniformly along highways, or for smoothing \(x-y\) projections of more complex spatio-temporal trajectories. Chebyshev approximations are more appropriate for modelling highly complex spatiotemporal trajectories such as pedestrian motion exhibiting stop-start and looping motions, whilst Fourier series approximation are suitable for mixed types of trajectory. We compare the performance of the 3 different modelling approaches in section 5.

The trajectory can be approximated by a polynomial \(P_m(t)\) of degree \(m < n\) as

\[
[x \mid y] \approx P_m(t) = a_0 + a_1 t + \ldots + a_m t^m
\]  

(1)

The projections in \((x, t)\) and \((y, t)\) planes are modelled as independent polynomials \(P_m^x, P_m^y\) in \(t\). Note that separate 1-D trajectories are created for each spatial coordinate \([x \mid y]\). The unknown \(2(m+1)\) coefficients \([a_{0i}, a_{ji}], i = 0, \ldots, m\) can be determined using LS by minimising the function \(E\) with respect to \(a_0, a_1, \ldots\)

\[
E(a_0, a_1, \ldots, a_m) = \sum_{i=1}^{n} \left( [x_i \mid y_i] - (a_0 + a_1 t + \ldots + a_m t^m) \right)^2
\]

(2)

The motion trajectories are thus indexed by vector of coefficients \((a_{0i}, \ldots, a_{mi}), (a_{0j}, \ldots, a_{mj})\).
3.2 Chebyshev Polynomials

Alternatively, a spatiotemporal trajectory can be approximated by a function \( f(t) \) expressed as a weighted sum of Chebyshev polynomials \( C_k(t) \) up to degree \( m \), defined as

\[
[x \mid y] = f(t) \approx \sum_{k=0}^{m} b_k C_k(t)
\]

where \( C_k(t) = \cos(k \cos^{-1}(t)) \) and

\[
b_0 = \frac{1}{m} \sum_{k=1}^{m} f(t_k), \quad b_i = \frac{2}{m} \sum_{k=1}^{m} f(t_k) C_i(t_k)
\]

for \( t \in [-1,1] \) and \( i = 1, \ldots, m \). The \( k \) roots of \( C_k(t) \) are given by \( t_j \) for \( 1 \leq j \leq k \). Implementation details can be found in [23]. Occasionally, it may be possible to approximate the motion trail (spatial trajectory shape only) in the \( x-y \) plane. In this case, we would replace \( t \) by \( x \) or \( y \) in one of the above equations depending on the choice of principal axis [6]. This would only be worthwhile if all trajectories could be aligned with the same principal axis. An example would be the modelling of vehicle trajectories in highway traffic surveillance.

3.3 Discrete Fourier Transform (DFT)

The DFT coefficients of the spatiotemporal trajectory can be calculated using the well known Fast Fourier Transform (FFT). The formulas for evaluating the Fourier coefficients can be found in [19, 23].

3.4 Similarity Search Metric

A Euclidean distance is used as the basis for comparing the similarity of motion trajectories. Each approximation produces a vector of coefficients which can be used to index a 2-dimensional spatiotemporal trajectory. Given two trajectories \( Q \) and \( S \), we can index these by a vector of \( 2(m+1) \) coefficients \( Q = \{ q_0, \ldots, q_m \} \) and \( S = \{ s_0, \ldots, s_m \} \), where \( q_i, s_i \) are \( q_i = [q_{xi}, q_{yi}]^T \) and \( s_i = [s_{xi}, s_{yi}]^T \) (\( i = 0, \ldots, m \)).

A Euclidean distance function (ED) on the coefficient space can be expressed as

\[
ED(Q, S) = \sqrt{\sum_{i=0}^{m} (q_i - s_i)^2} = \sqrt{\sum_{i=0}^{m} (q_{xi} - s_{xi})^2 + (q_{yi} - s_{yi})^2}
\]

4 Learning Trajectory Patterns Using Self-Organizing Maps

Self-organised maps (SOMs) have been previously used for motion trajectory clustering and classification [15, 16] with trajectories encoded as discrete point-based flow vectors. This step can be replaced by the proposed coefficient indexing scheme. A SOM can discover the underlying structure of motion trajectory in an appropriate feature space through unsupervised learning. As we shall see the feature space can be the original high dimensional trajectory space or the reduced coefficient subspace.

4.1 Network Model

The architecture chosen for the SOM is very simple with a layer of input neurons connected directly to a single 1-D output layer. Each input neuron is connected to every output neuron with the connection represented by a weight vector. The network topology is shown in Fig. 1. A similar architecture was used in [16] for learning vehicle trajectories as a means for accident prediction.
In a SOM network, physically adjacent output nodes encode the patterns in the trajectory data that are similar and, hence, it is known as a topology-preserving map. Consequently, similar object trajectories are mapped to the same output neuron. The number of input neurons is determined by the size of the feature vector which in this case relates to the selected number of coefficients for the basis functions. The degree of the polynomial can be chosen by setting a threshold on the maximum deviation of the function approximation from the data or mean-squared error. The number of output neurons represents the number of clusters in the trajectory data and this is selected manually.

4.2 Learning Algorithm

The algorithm used to cluster the trajectories differs slightly from the original SOM proposed by Kohonen [24]. The number of output neurons is initially set to a higher value than the desired number of clusters which we wish to produce. After training the network, clusters representing the most similar patterns are merged using an agglomerative method until the cluster count is reduced to the required number. The weights are initialised to linearly spaced values lying within the range of input values. Neighbourhood size is initially set to cover over half the diameter of the output neurons.

Let $B$ be the input feature vector representing the set of trajectory basis function coefficients, and $W$ the weight vector associated to each output neuron. The learning algorithm comprises the following steps:

1. Determine the winning output node $k$ (indexed by $c$) such that the Euclidean distance between the current input vector $B$ and the weight vector $W_k$ is a minimum amongst all output neurons, given by the condition
   \[
   \|B - W_c(t)\| \leq \|B - W_k(t)\| \quad \forall k
   \] (6)

2. Train the network by updating the weights. A subset of the weights constituting a neighbourhood centred around node $c$ are updated using
   \[
   W_k(t+1) = W_k(t) + \alpha(t) \eta(k,c)(B - W_k(t))
   \] (7)
   where $\eta(k,c) = \exp(-|r_k-r_c|^2 / 2 \sigma^2)$ is a neighbourhood function that has value 1 when $k=c$ and falls off with distance $|r_k-r_c|$ between nodes $k$ and $c$ in the output layer, $\sigma$ is a width parameter that is gradually decreased over time and $t$ is the training cycle index.

3. Decrease the learning rate $\alpha(t)$ linearly over time.

4. After a pre-determined number of training cycles, decrease the neighbourhood size.

5. At the end of the training phase, merge the most similar cluster pairs until the desired number of groupings is achieved. Clusters are merged by calculating the weighted mean of the weights associated with each neuron taking into account the number of input samples allocated to the cluster. Assuming $W_a$ and $W_b$ are the weight vectors associated with output neurons representing the most similar clusters, and $m, n$ are the number of sample trajectories mapped to these neurons respectively, a new weight value $W_{ab}$ for the merged cluster can be calculated as
   \[
   W_{ab} = \frac{mW_a + nW_b}{m+n}
   \] (8)
5 Experiments

We now present some results to indicate the effectiveness of the proposed trajectory clustering technique. We have evaluated the performance of the trajectory clustering algorithm using the CAVIAR visual tracking database [26]. The database consisted of hand annotated video sequences of moving and stationary people and are intended to provide a testbed for benchmarking vision understanding algorithms. Semantic descriptions of the target object behaviours and motion had been previously generated and stored in XML files. These files have been parsed to extract ground truth-labelled object trajectories. The dataset contains 222 trajectories as shown in Fig. 2.

![Background scene containing ground truth labelled object trajectories extracted from the CAVIAR dataset [26].](image)

The performance of the 3 different trajectory representation schemes have been compared. The purpose of the experiment is to test the retrieval accuracy for each approximation scheme and to investigate the effect of varying the number of coefficients used for model fitting. We also wish to examine the effect on retrieval performance by introducing additive noise. A corrupted dataset $S_C$ is produced by adding the term $\eta U[0,1] \times rangeValues$ to each coordinate in the original set $S$, where $\eta$ is a scaling factor such that $0 \leq \eta \leq 1$, $U[0,1]$ is uniform random noise on the interval $[0,1]$, and $rangeValues$ is the range on $X$ or $Y$ coordinates.

Each corrupted trajectory in $S_C$ then serves as an example query $Q_C$ and we search for its closest match in the original dataset $S$ by searching for minimum $ED(Q_C,S)$. A set of rankings over all $Q$ is produced. In the absence of noise and when no data is excluded, the mean ranked value should be 1. For ease of comparison we record the proportion of times (as a percentage) the query trajectory is ranked correctly as 1. This is repeated for different number of coefficients in LS, Chebyshev and Fourier approximations and for various values of $\eta$. The results are summarised in Fig. 3.

![Effect of scaled uniform noise on trajectory retrieval accuracy.](image)

Fig. 3. Effect of scaled uniform noise on trajectory retrieval accuracy. $w$ is the scaling factor, LS = least squares, CS = Chebyshev polynomials, and FS = Fourier series approximation.
For small amounts of noise, the choice of approximation scheme or number of coefficients does not appear to be too critical, although there is a slight fall off in performance of LS as the coefficient number increases. For higher noise levels, it is apparent that Fourier series approximations outperform LS and Chebyshev polynomials.

This may be explained by the fact that Euclidean distance defined over Fourier coefficients is more noise resistant in the frequency domain. In previous work, it has been shown that a LS-RANSAC approach would be beneficial if it is known that the tracking algorithm produces very noisy estimates with a significant number of outliers [11].

The retrieval experiment was then repeated but this time under simulated object occlusion by removing at random a subsequence of points. The proportion of points removed (p) varied between 10, 20 and 30% of the trajectories’ length. Each of the results obtained were averaged over 10 random occluding subsequences. In this instance there was no added noise. The percentage retrieval accuracy over all query trajectories was determined as before for each choice of approximation scheme and number of coefficients. The results are shown in Fig. 4.

In this case LS approximations perform best under conditions of simulated occlusion.

The $ED$-metric defined over the coefficient subspace is now used to perform the trajectory clustering step.

We ran the SOM clustering algorithm using the 3 different spatiotemporal approximations. From empirical observations it was noticed that if the number of coefficients is too low (typically $m < 3$), poor clustering results are obtained. As a sanity check, we repeated the clustering using a standard $K$-Means algorithm [25] and the same result was observed. Although satisfactory results are obtained in retrieval experiments with a small number of coefficients, there is insufficient discriminatory power in a very low dimensional coefficient subspace to achieve a meaningful clustering outcome.

In practice we have found no discernible differences in SOM clustering results between spatiotemporal trajectory models generated by $CS=4..8$ and $FS=4..6$ for about $K=10$ clusters. The SOM algorithm always produces visually better cluster separations than $K$-means. This is to be expected given that SOM better preserves the topology of the original trajectory space. We do not attempt to normalise to achieve scale or translational invariance since we wish to preserve these differences in the clustering step.

![Fig. 4](image-url)

**Fig. 4.** Effect of occluding subsequences on trajectory retrieval accuracy. $p$ is the percentage of subsequence length removed from the original trajectory. Results are averaged over 10 random removals. LS = least squares, CS = Chebyshev polynomials, and FS = Fourier series approximation.
We initially train a SOM network with 50 output neurons and then reduce these to 9 using the agglomerative clustering method described in section 4.2. The resulting trajectory groups are shown in Fig. 5 for Chebyshev polynomial of degree 8, although similar results are obtained using Fourier approximations of degrees 4 to 6. Visual inspection shows that qualitatively similar motion trajectories have been grouped together quite successfully. Motions across the corridor from left-to-right and right-to-left are grouped into separate clusters as expected.

In order to visualise the effects of trajectory clustering in the coefficient subspace, we have performed Principal Component Analysis (PCA) on the vector of Fourier coefficients ($m = 4$) and calculated the first 3 PCs which explain 94% of the total variation. The results are shown in Fig. 6. Each point represents an instance trajectory and these are colour/marker coded to highlight the separate cluster groups each trajectory is allocated to. These plots show a good degree of cluster separation in the PCA reduced coefficient subspace.

To investigate the effectiveness of clustering in the dimensionally reduced coefficient subspace compared to clustering in the original space using trajectories encoded as discrete point-based flow (PBF) vectors, we performed some classification tests. The class labels of the motion trajectory patterns were learned using the SOM and K-means unsupervised techniques on the CAVIAR training data. The dataset $\mathcal{X}$ was then randomly partitioned into training and test sets of equal sizes for cross-validation purposes. We used a $k$-NN classifier
(with \(k = 1\)) to classify instance trajectories from the test set and generated the overall classification accuracy. To avoid bias, we repeated the random partitioning 500 times and averaged the classification errors over the test set. The results summarised in Table 1 demonstrate the superiority of learning trajectory patterns in the coefficient subspace. The classification accuracy obtained using coefficient subspace learning is higher than that of discrete PBF vector encoding for both SOM and \(K\)-means algorithms.

<table>
<thead>
<tr>
<th>Method type</th>
<th>Accuracy %</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM: coefficient subspace</td>
<td>91.9</td>
</tr>
<tr>
<td>SOM: PBF vectors</td>
<td>79.9</td>
</tr>
<tr>
<td>K-Means: coefficient subspace</td>
<td>88.8</td>
</tr>
<tr>
<td>K-Means: PBF vectors</td>
<td>85.6</td>
</tr>
</tbody>
</table>

**Table 1.** Comparison of mean overall classification accuracy for 2 different clustering techniques (SOM and \(K\)-means) and 2 different trajectory encodings (coefficient subspace and discrete PBF vectors). #classes : #trajectories = 9 : 111.

Significant speed-ups are thus possible using this approach and it now becomes feasible to learn motion activity patterns and perform trajectory classification online. This is not currently possible when trajectories are encoded as discrete PBF vectors.

In the next experiment we compare the performance of all 4 methods in trajectory prediction and classification. From the original set \(S\), we define a set of partial trajectories \(S_P\) by sampling a subset of points from the original trajectories from 10% of the original length up to 100% in steps of 10. A partial trajectory is said to be misclassified if it is not assigned to the original cluster group based on the full trajectory set \(S\). The classification is performed based on the input vectors consisting of coefficients or PBF vectors. This is repeated for the SOM and \(K\)-Means defined set of codebook vectors. Assuming the size of the input vector to be \(m\), the Euclidean distance is then calculated between the input vector \(B_i\) representing the partial trajectory and weight vector \(W_{ik}\) associated with \(k\)th output neuron as

\[
D_k = \sqrt{\sum_{i=1}^{m} (B_i - W_{ik})^2} \quad \forall k
\]

The partial trajectory is then allocated to the class associated with the minimum value of \(D_k\). If the class is different from the one to which the full trajectory was allocated then it is recorded as a misclassification. In the case of flow vector encoded trajectories, the point-based Euclidean distance measure is used. The percentage of misclassified trajectories taken over \(S_P\) is calculated for each method and for various sampling rates. The use of eq. (9) rather than \(k\)-NN classifier saves time as only the \(m\) weight vectors need be examined rather than the full test set. As evidenced from Fig. 7, the classifier derived from SOM in the coefficient subspace again outperforms \(K\)-Means. Furthermore, parameterized models prove more effective than point-based flow vectors in the trajectory prediction and classification task. These results give further impetus to the development of alternative dimensionality reduction techniques for learning and prediction of motion activity patterns.

![Fig. 7](image-url)
Finally, we repeated the above experiment on trajectory prediction but compared the classification accuracy performance of the 3 approximations schemes for coefficient subspace learning using SOM and $K$-means. The use of SOM outperforms $K$-means in all cases of unsupervised learning and Fourier approximations records the highest classification accuracy over the partial sampling range 40-90%. The potential for predicting the correct spatiotemporal trajectory pattern from a partial trajectory has therefore been demonstrated.

![Diagram showing classification accuracy](image)

6 Discussion and Conclusion

This paper presents a neural network learning algorithm for classifying spatiotemporal trajectories. Global features of motion trajectories are found to be represented well by polynomial and Fourier series approximations and this is apparent in the cluster visualizations. Using coefficients of basis functions as input feature vectors to a neural network learning algorithm offers an efficient alternative to the use of flow vectors for trajectory classification and prediction.

A current disadvantage is the representation of highly complex trajectories (e.g. hand trajectories in sign language) which are inherently unsuited to a global polynomial-based or Fourier representation. One possibility is to use a trajectory segmentation or multiscale approach and augment the feature vector with additional entries relating to object shape or colour. In future work we would like to extend this work to the autonomous detection of anomalous trajectories and prediction of unusual motion behaviour. A more comprehensive performance evaluation of other dimensionality reduction techniques in trajectory classification is also needed, e.g. ICA, HMMs and distance metric learning.

References


Human Motion Recognition based on Dynamic Shape Analysis

Ning Jin and Farzin Mokhtarian
Centre for Vision, Speech, and Signal Processing
University of Surrey
Guildford, Surrey GU2 7XH
Email: \{ning.jin, f.mokhtarian\}@surrey.ac.uk

Abstract

Dynamic shape is a time series of the outlines of a moving object, which records the temporal variation of the shape of the object during its movement. We believe that the dynamic shape provides clues about the motion performed by the object. In this paper, we borrow tools from system identification to capture the “essence” of the dynamic shape, so that we convert the problems of modelling, learning, and recognizing object motions to the modelling, learning, and comparing of dynamical systems where each motion is represented. Concretely, the object boundary extracted from each frame is represented by its Curvature Scale Space (CSS) image. The CSS method has been selected for MPEG-7 standardization. Then we utilize the CSS representations as the training data to learn the dynamical systems, i.e. estimate system parameters; finally supervised pattern classification techniques based on various types of distance measure are adopted for recognition.

1 Introduction

Vision-based human motion recognition is currently one of the most active research areas in the domain of computer vision. It is motivated by a wide spectrum of applications, such as visual surveillance, virtual reality, and human-computer interface. The state of the arts in this area may be learnt from some comprehensive surveys [6] [5] [1].

Motion recognition may simply be thought as the classification of time varying feature data, i.e. matching an unknown test sequence with a group of learned reference motions. It is then obvious that there are four fundamental problems related to vision-based motion recognition, that is,

1. what abstract features are extracted from original image data and to be employed for recognition;
2. how to characterize these features;
3. how to learn reference motions from the characterized features;
4. how to match similarities between reference motions and unknown test sequences.

In the past few years, much work has been carried out in this area. In [2], Bobick et al. present a method for recognition of temporal templates. A temporal template is a static vector-image where the vector value at each point is a function of the motion properties at the corresponding spatial location in an image sequence. Then actions are represented by the cumulative motion images called Motion Energy Image (MEI) and Motion History Image (MHI). The MEI represents where the motion has occurred in the image plane, and the MHI represents the recency of motion using different gray scale level. For recognition, the Hu moments obtained from the templates are known to yield reasonable shape discrimination in a translation and scale invariant manner. Extracted Hu moments are matched using a nearest neighbor approach against the exemplars of already learned motions. In [7], human body postures are estimated frame by frame, which starts from using the horizontal and vertical histograms to represent the binary shapes associated to humans, and then feeding them into an unsupervised clustering algorithm. The Manhattan distance is used for both clusters building and run-time classification. Statistical modelling of the detected postures is performed by discrete Hidden Markov Models (HMMs). Liang Wang et al., in [15], present an automatic gait recognition algorithm based on statistical shape analysis, in which temporal changes of the detected silhouettes are represented as an associated sequence of complex vector configurations in a common coordinate frame, and are further analyzed using the Procrustes shape analysis method [4] to obtain the mean shape as gait signature. Supervised pattern classification techniques based on the full Procrustes distance measure are adopted for recognition. In [16], Liang Wang et al. propose another gait recognition algorithm also based on silhouette analysis. The 2D binary silhouette image is extracted from each frame, and then is mapped into a 1D normalized distance signal by contour unwrapping with respect to the silhouette centroid. Then, eigenspace transformation based on Principal Component Analysis (PCA) is applied to the time-varying distance signals to reduce the dimensionality of the input feature space. Finally, supervised pattern classification techniques are performed in the lower-dimensional eigenspace for recognition. Recently, Liu and Ahuja [8] proposed an approach to modelling gradual changes in the 2D shape of an object. They represented 2D
region shape in terms of the spatial frequency content of the region contour using Fourier coefficients. The temporal changes in these coefficients are applied as the temporal signatures of the shape variations. Then, the coefficient time series is modelled as an autoregressive (AR) process.

Simple but elementary human motions, such as walking, running, sitting down, squatting, and so forth, account for a large proportion of human daily life. So, the ability to recognize this type of human motions plays a crucial role in many promising applications, such as smart visual surveillance, content-based video retrieval, and video coding. In [2], Bobick et al. have made an excellent effort.

In this paper, we propose an efficient view-dependent human motion recognition algorithm based on statistical shape analysis. We believe that the temporal variation of the shape of an object provides clues which will help us identify objects and even recognize activities performed by the object. In our approach, therefore, human body contours are extracted from original image data frame by frame and to be used as the abstract features for human motion recognition. To characterize human body contours, curvature scale space (CSS) image [10] is applied. In order to capture the intrinsic motion information hidden behind the temporal deformation of human body contour, we borrow tools from system identification [9] to model the temporal variation of human body contours in the space of dynamical systems where each motion is represented. Finally, various types of distance measure between dynamical systems are calculated for recognition.

The remainder of this paper is organized as follows: section 2 gives a presentation of the theory of the proposed method, which is further divided into two subsections respectively addressing CSS techniques and system identification tools used in our proposed algorithm; section 3 shows experimental results; and this paper is ended with section 4 on conclusions and future work.

2 Motion Recognition System

In our motion recognition system, we seek answers to the following three questions:

1. how to characterize the abstract features of a moving object;
2. how to learn dynamical systems from the characterized features;
3. how to recognize unknown motions of an object in the space of dynamical systems.

The first question is addressed in the subsection 2.1; and the latter two questions are addressed in the subsection 2.2.

2.1 Curvature Scale Space Representation

What abstract features are extracted from original image data and to be employed for recognition? Our motion recognition algorithm is based on shape, i.e. human body contours. As far as how to characterize these features is concerned, in this section, we will offer a solution to this problem, i.e. curvature scale space. The curvature scale space (CSS) technique is a powerful and popular shape analysis tool. In [10], a comprehensive presentation on the theory and applications of the CSS technique is given.

Consider a parametric vector equation for a curve:

\[ \Gamma(u) = (x(u), y(u)), \]  

where \( u \) is an arbitrary parameter. The formula for computing the curvature function can be expressed as

\[ \kappa(u) = \frac{\ddot{x}(u)\dot{y}(u) - \dot{x}(u)\ddot{y}(u)}{(\dot{x}^2(u) + \dot{y}^2(u))^{3/2}} \]  

where \( \cdot \) represents the derivative. Eq. (2) is for continuous curves and there are several approaches in calculating the curvature of a digital curve [14]. We make use of the idea of curve evolution which basically studies shape properties while deforming contour shape over time.

If \( g(u, \sigma) \) is a one-dimensional Gaussian kernel with width \( \sigma \), then \( X(u, \sigma) \) and \( Y(u, \sigma) \) represent the components of evolved curve \( \Gamma_{\sigma}(u) \).

\[ X(u, \sigma) = x(u) \ast g(u, \sigma) \]
\[ Y(u, \sigma) = y(u) \ast g(u, \sigma) \]

where \( \ast \) is the convolution operator. Due to the properties of convolution, the derivatives of each component can be calculated as

\[ X_u(u, \sigma) = x_u(u) \ast g(u, \sigma) = x(u) \ast g_u(u, \sigma) \]
\[ X_{uu}(u, \sigma) = x_{uu}(u) \ast g(u, \sigma) = x(u) \ast g_{uu}(u, \sigma) \]
\[ Y_u(u, \sigma) = y_u(u) \ast g(u, \sigma) = y(u) \ast g_u(u, \sigma) \]
\[ Y_{uu}(u, \sigma) = y_{uu}(u) \ast g(u, \sigma) = y(u) \ast g_{uu}(u, \sigma) \]

and similarity for \( Y_u(u, \sigma) \) and \( Y_{uu}(u, \sigma) \). Since the exact forms of \( g_u(u, \sigma) \) and \( g_{uu}(u, \sigma) \) are known, the curvature of an evolved digital curve is given by

\[ \kappa(u, \sigma) = \frac{X_u(u, \sigma)Y_{uu}(u, \sigma) - X_{uu}(u, \sigma)Y_u(u, \sigma)}{(X_u(u, \sigma)^2 + Y_u(u, \sigma)^2)^{3/2}} \]
As $\sigma$ increases, the shape of $\Gamma_\sigma$ changes. The process of generating ordered sequences of curves as $\sigma$ varies from a small to a large value is referred to as the evolution of $\Gamma$.

If we locate the curvature zero crossings of $\Gamma_\sigma$ during evolution, we can display the resulting points in $(u, \sigma)$ plane. For every $\sigma$ we have a certain curve $\Gamma_\sigma$ which in turn has some curvature zero crossing points. As $\sigma$ increases, $\Gamma_\sigma$ becomes smoother and smoother, and the number of zero crossing points decreases. When $\sigma$ is sufficiently high, $\Gamma_\sigma$ will be a convex curve with no curvature zero-crossings, and the process of evolution in terminated. The result of this process can be represented as a binary image called CSS image of the curve. From Fig. (1), we can see the course of the evolution of the curve and the final CSS image.

As seen in Fig. (1), the CSS image consists of several arch-shaped contours, each related to a segment of the shape. This shape is finally represented by the locations of the maxima of its CSS contours, i.e. the $\sigma$-maxima in the CSS image, e.g. the locations marked by the symbol X in the Fig. (1). Small contours of the CSS image are related to noise or small ripples of the curve. For the sake of simplicity, small maxima are not included in the representation. As a result, the CSS image collapses into a small feature vector consisting of the first $k$ maxima of CSS contours which are above a certain size.

Figure 1: The evolution of the curve.
2.2 System Identification

The temporal deformation of the shape of an object provides informative clues about the object identity and even activities performed by the object. We therefore consider that, given a shape sequence that corresponds to a motion, if we can find a system that is able to model this temporal shape deformation, we may take this system as the template of that motion; and then in recognition, given an unknown motion sequence, we just need to match its similarities with learned templates. Based on the above consideration, we should solve the following three problems.

1. What systems can be used to model the temporal shape deformation?
2. How to learn the systems from feature data?
3. How to match similarities between systems?

2.2.1 The System Model

What systems can be used to model the temporal shape deformation? We start from the assumption that a sequence of CSS representations

\[ y(t) = \{(u_{t1}, \sigma_{t1}), \ldots, (u_{tk}, \sigma_{tk})\}_{t=1}^{\tau} \]  

is a realization from a second-order stationary stochastic process, where for each frame we use the locations of the first \(k\) maxima of the CSS image to represent the shape, i.e. \(\{(u_{t1}, \sigma_{t1}), \ldots, (u_{tk}, \sigma_{tk})\}\) given the time \(t\).

It is well known that a second-order stationary process with arbitrary covariance can be modelled as the output of a linear dynamical system driven by white, zero mean Gaussian noise [9]. In our case, therefore, we assume there exist a positive integer \(n\), a process \(\{x(t)\}_{t=1, \tau} \in \mathbb{R}^n\) (the “state”), with symmetric positive-definite matrices \(Q \in \mathbb{R}^{n \times n}\) and \(R \in \mathbb{R}^{m \times m}\), such that \(\{y(t)\}\) is the output of the following Gauss-Markov “ARMA” model

\[
\begin{align*}
  x(t+1) &= Ax(t) + v(t) \\
  y(t) &= Cx(t) + \omega(t)
\end{align*}
\]

(7)

for some matrices \(A \in \mathbb{R}^{n \times n}\) and \(C \in \mathbb{R}^{m \times n}\). \(m = k \times 2; n\) being also called system order, is an empirical value, and in our experiment it is chosen to be 4. Matlab system identification toolbox N4SID can help estimate the order of the system.

Our models are discrete-time, continuous-state dynamical systems, and motions are coded in such dynamical systems which is represented by the so-called system parameters, i.e. \(\{A, C, Q, R\}\).

2.2.2 Learning the System Model

In our algorithm, learning the dynamical systems must be done in a canonical way to guarantee that a particular data set just corresponds to one and only one model. From the observation on the model (7), however, the choice of system parameters \(A, C, Q, R\) is not unique. Fortunately, this can be solved through using well established algorithms from system identification [9], i.e. transforming the model (7) into a particular form - “innovation representation” [11] - that is unique.

The problem of learning the dynamical system from data is then formulated as follows: given measurements: \(y(1), \ldots, y(\tau)\), i.e. a sequence of CSS representations Eq. (6), estimate the system parameters \(A, C, Q, R\). Ideally, we would want the maximum likelihood solution from the finite samples

\[
\hat{A}, \hat{C}, \hat{Q}, \hat{R} = \arg \max_{A, C, Q, R} p(y(1), \ldots, y(\tau) | A, C, Q, R)
\]

(8)

Asymptotically optimal solutions of this problem, in the maximum-likelihood sense, have been presented in the system identification theory [9]. In particular, the subspace identification algorithm N4SID proposed by Van Overschee and De Moor in [12] has been available as a Matlab toolbox. Moreover, in [13], Soatto et al. propose a closed-form sub-optimal solution of the problem in the sense of Frobenius. Given \(y(1)y(2)\ldots y(\tau)\) being a sequence of CSS representations, let \(y(1)y(2)\ldots y(\tau) = UZV^T\) be the singular value decomposition of the sequence. Then

\[
\begin{align*}
  \hat{C}(\tau) &= U \\
  \hat{A}(\tau) &= \Sigma V^T D_1 V (V^T D_2 V)^{-1} \Sigma^{-1}
\end{align*}
\]

(9)

where \(D_1 = \begin{bmatrix} 0 & 0 \\ I_{\tau-1} & 0 \end{bmatrix}\) and \(D_2 = \begin{bmatrix} I_{\tau-1} & 0 \\ 0 & 0 \end{bmatrix}\). The state transition matrix \(A\) and the output transition matrix \(C\) are the intrinsic characteristics of the model, nevertheless the input and output noise covariances \(Q\) and \(R\) are not significant for the purpose of recognition. Therefore, for the sake of simplicity, we will concentrate out attention only on the \(A\) and \(C\).
2.2.3 Distance Between Models

Through learning over characterized feature data, we construct a space of dynamical systems where each motion is coded by a corresponding system parameters. Performing recognition in such a space, therefore, entails computing distances between models. Since the parameters of the dynamical systems lie in a non-Euclidean space (even if the model itself is linear), the distance calculation between models is non-trivial. Here, we apply three distance metrics that has been widely adopted in the system identification for comparing ARMA models. The distance metric is based on the so-called subspace angles \([11]\). Subspace angles between two ARMA models are defined as the principal angles \((\theta_i, i = 1, 2, ..., n)\) between the column spaces generated by the observability matrices of the two models extended with the observability matrices of the inverse models \([3]\). The subspace angles between two ARMA models may be computed using the method proposed in \([3]\). Then three distances can be calculated based on these subspace angles \(\theta_i, i = 1, 2, ..., n\), i.e. Martin distance \((d_M)\), gap distance \((d_g)\), and Frobenius distance \((d_F)\).

\[
d_M^2 = \ln \prod_{i=1}^{n} \frac{1}{\cos^2(\theta_i)} \tag{10}
\]

\[
g = \sin \theta_{\text{max}} \tag{11}
\]

\[
F^2 = 2 \sum_{i=1}^{n} \sin^2 \theta_i \tag{12}
\]

The various distance metrics do not change results significantly.

3 Experimental Results

To demonstrate the ability of our proposed algorithm to perform motion recognition, we have performed a number of experiments.

3.1 Data Acquisition

In our database, so far, we have captured some human motions, e.g. walking, sitting down, standing up, bending down, straightening up, squatting down, and lying down. During the video capturing, a digital camera fixed on a tripod is used to capture motion sequences at a rate of 25 fps. Here, for the sake of simplicity, we assume that there is only one person in every sequence who naturally performs the above motions in the field of camera view without any occlusion. Furthermore, all the motion is performed in a straight-line path and in three different view angles with respect to the camera direction, i.e. frontally (90°), laterally (0°), and obliquely (45°). Up to now, the resulting database includes 5 different people and one sequence per view per person per motion, which means our database thus includes a total of 105 \((5 \times 3 \times 7)\) sequences. The length of each sequence varies with the time of a person performing that motion, but the average of the whole database is about 60 frames per sequence. The resolution of all images is 288 \(\times\) 352. In Figure 2, we show several sampled frames from our database.

Figure 2: Sampled frames from the database: walking, sitting, squatting.
3.2 Processing

The work flow of our proposed algorithm is presented as follows. First, for each sequence, a motion segmentation algorithm based on the single Gaussian model is applied to extract the foreground object from each frame. After the post-processing such as noise reduction and shadow elimination, the contour of each human is extracted using a contour tracing algorithm. A fixed number (250 in our experiments) of points is further re-sampled from each contour. Then, we calculate the CSS image of each contour, and select the locations \((u, \sigma)\) space of the first \(k\) maxima of the CSS image as the contour representations, i.e. \(\{(u_{t1}, \sigma_{t1}), \ldots, (u_{tk}, \sigma_{tk})\}\). (see Fig. 3). Up to now, we have prepared the output of the state-space model (7) well, i.e. the training data (see Fig. 4) for learning dynamical systems. The subsequent processes are to learn the state-space models, compute the subspace angles, and calculate the distances between models. In our experiments, given a sequence of CSS representations which describes a motion, we apply the Equation (9) to obtain the system parameters \(\{A, C\}\) which are considered as the signatures of that motion. The subspace angles \(\theta_i\) between two models \(\{A_1, C_1\}\) and \(\{A_2, C_2\}\) are then computed following the method proposed in [3]. Finally, three distances (10)(11)(12) are calculated for recognition.

![Figure 3: (a)The original image; (b)The motion segmentation; (c) The binary image; (d)The CSS image](image)

3.3 Results

Let \(T\) be the dynamical system of an unknown sequence, and \(R_i\) be the dynamical system of the \(i^{th}\) reference motion, we can classify this unknown sequence into the class \(c\) based on

\[
c = \arg \min_i d(T, R_i)
\]

(13)

where \(d\) may be anyone of the three distance metrics (10)(11)(12) mentioned above. Table 1 shows the Martin distances of a set of test motions matching the reference motions. It can be seen that all the test motions have the minimum distances with their corresponding reference motions. In fact, based on the pattern classification method mentioned above, we got very encouraging experimental results. The overall recognition rates on our database are 100%.

4 Conclusions and Future Work

Although the results of our experiments are encouraging, to provide a more general algorithm for human motion recognition in public places, much work remains to be done.

1. Our database is still too small and no steps are undertaken to ensure the randomicity of our data. The robustness of our proposed algorithm, therefore, still needs to be further verified.
2. Clothing will considerably influence the shapes of moving people, e.g. clothes in different seasons. Because our method is based on the statistical shape analysis, however, it will be inevitably affected. Furthermore, there so far exists no perfect motion segmentation algorithm. So in many cases, especially in the outdoor environments, the extracted contour will be very noisy.

3. The main drawback of the current method is that it is view-dependent. To provide more general approach to human motion recognition, we should explore the method which is able to capture view-invariant characteristics from original image data.

Nowadays, there are increasing demands for automatic human motion recognition in some promising applications, such as visual surveillance, content-based video retrieval, and video coding. The temporal deformation of the shape of an object provides informative clues about the object identity and even activities performed by the object. So in this paper, we introduce two well established theories (curvature scale space theory and system identification theory) into our research. From experiments, we have seen that this combination may give us encouraging results.

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MM1</td>
<td>0.016</td>
<td>4.68</td>
<td>5.71</td>
<td>8.99</td>
<td>4.78</td>
<td>0.94</td>
<td>0.73</td>
</tr>
<tr>
<td>MM2</td>
<td>6.52</td>
<td>0.06</td>
<td>0.22</td>
<td>0.51</td>
<td>0.45</td>
<td>3.66</td>
<td>3.37</td>
</tr>
<tr>
<td>MM3</td>
<td>5.59</td>
<td>0.41</td>
<td>0.088</td>
<td>0.43</td>
<td>0.27</td>
<td>3.18</td>
<td>3.26</td>
</tr>
<tr>
<td>MM4</td>
<td>13.9</td>
<td>1.82</td>
<td>1.07</td>
<td>0.021</td>
<td>0.79</td>
<td>5.41</td>
<td>5.01</td>
</tr>
<tr>
<td>MM5</td>
<td>5.83</td>
<td>0.44</td>
<td>0.34</td>
<td>0.45</td>
<td>0.054</td>
<td>2.52</td>
<td>3.05</td>
</tr>
<tr>
<td>MM6</td>
<td>0.97</td>
<td>2.37</td>
<td>1.87</td>
<td>2.49</td>
<td>1.4</td>
<td>0.027</td>
<td>0.26</td>
</tr>
<tr>
<td>MM7</td>
<td>0.98</td>
<td>1.98</td>
<td>1.86</td>
<td>2.83</td>
<td>1.5</td>
<td>0.154</td>
<td>0.024</td>
</tr>
</tbody>
</table>

Table 1: Martin distances between a set of test motions and the reference motions. M1~M7 in turn denote the test motions: walking, sitting down, standing up (sit), squatting down, standing up (squat), bending down, and straightening up. MM1~MM7 in turn denote the corresponding reference motions with respect to M1~M7.
References


Continuous Time-Varying Gesture Segmentation by Dynamic Time Warping of Compound Gesture Models

Hong Li¹ Michael Greenspan¹,²

¹ Dept. of Electrical & Computer Engineering, Queen’s University, Kingston, Canada
²School of Computing, Queen’s University, Kingston, Canada
email: 1hl5@qlink.queensu.ca, michael.greenspan@queensu.ca

Abstract

A novel method is introduced to simultaneously recognize and segment time-varying human gestures from continuous video streams. Motion is represented by a 3D spatio-temporal surface based upon the evolution of a contour over time. The temporal endpoints of a gesture are estimated by Dynamically Time Warping the input sequence against a set of Compound Gesture Models, which are composed of the concatenation of two gestures. The system has been implemented and tested on 8 different gestures performed by 5 subjects at a variety of time scales. The results demonstrate that the proposed method is very effective, achieving recognition rates of 88.1% for multiple scale and 93.3% for single scale tests.

1 Introduction

The modeling and recognition of human gestures from video streams is driven by a wide range of promising applications, e.g., smart surveillance, human-machine interaction, biometrics, etc [1, 9]. Unlike isolated gesture recognition which requires segmentation prior to recognition [2, 8, 11, 14, 19], for continuous gesture recognition from streamed video, recognition and segmentation become aspects of the same problem, since recognition requires determining the start and end times of a gesture [4, 5, 20, 22]. That human gestures are dynamic signals with both spatial and temporal variations makes segmentation difficult. Indeed, if we allow that gestures are time-varying (i.e., that a given gesture may occur at a variety of speeds) then even the temporal differences between the start and end times are initially unknown.

Yacoob and Black [22] modeled the temporal shifting and scaling as a parameterized transformation and a search was executed over the parameter space to find the best match. Darrell et al. [4] performed an exhaustive search at each time instance using Dynamic Time Warping (DTW) [10]. On the other hand, work based on Hidden Markov Models (HMM) [13], which models the temporal evolution of poses of a gesture as transition probabilities between states, has been reported. Liang et al. [7] used the time-varying parameter to detect the endpoint in continuous Taiwan Sign Language, and HMM is employed for recognition. Lee et al. [5] constructed a threshold model based on HMM to spot gestures in sequences. Starner et al. [15] employed HMM to recognize structured American Sign Language sentences without explicit segmentation at the word level.

In previous work [6], we proposed a method capable of recognizing and estimating the temporal scale of human gestures simultaneously. In this paper, we extend that work to segment time-varying gestures from continuous video sequences. In particular, motion is represented by a 3D spatio-temporal surface (a Motion Signature) based on the evolution of a contour over time, as in [11, 19]. Rather than exhaustively searching through all possibilities, we instead estimate the endpoints of a gesture by dynamically warping the input sequences with Compound Gesture Models composed of the concatenation of two gestures. Gestures are then recognized by finding the best match only over the estimated endpoints. Since only a small number of gestures will pass the DTW test, the search is bounded within a small range and runtime efficiency improves significantly compared with searching exhaustively through all Gesture Models at all possible start times and scales.

2 Motion Signature

Separating the subject of interest from the background is in general a nontrivial problem and is a research topic in and of itself [16]. As our work focuses on gesture recognition, we make the simplifying assumption that there is only one moving subject in a static background, and a relatively simple background subtraction method based on image subtraction is used to segment the foreground subject [21]. Starting from the lower left corner of the foreground region, the border following method proposed by Suzuki et al. [17] is then used to extract the contour of the subject.

It is desirable to parameterize each contour as a shape descriptor that uniquely characterizes the contour. A shape descriptor is a feature vector that preserves the important characteristics of a shape, and is ideally invariant...
to translation, (size) scale, and rotation [3]. We chose the 1D distance-to-centroid as the shape descriptor, similar to that which has been used in [11, 19]. An example of this shape descriptor is illustrated in Fig.1 (which is similar to Fig.3 in [19]). The length of the 1D descriptor varies with the size of the human, the pose at a given frame, the distance of the subject to the camera, and the intrinsic camera parameters, such as focal length. Even when all these conditions are identical, due to image noise, it is not possible to guarantee exactly the same number of contour points for distinct image frames. We therefore normalize the contour size by sub-sampling the contour signal at equal intervals to a fixed length $L$, and then divide the 1D signal by the largest distance among the points. The resulting shape descriptor $D_t$ at image frame $t$ is given in Eq. (1):

$$D_t = [\hat{d}_1, \hat{d}_2, \hat{d}_3, \ldots, \hat{d}_L] \quad 0 \leq \hat{d}_i \leq 1$$  \hspace{1cm} (1)

where $\hat{d}_i$ is the normalized Euclidean distance between contour point $p_i$ and the contour centroid. Fig. 1(a) illustrates a contour, where $d_A$ and $d_B$ are the distances between the centroid and the initial points $A$ and $B$, respectively. Fig. 1(b) plots the resulting shape descriptor $D_t$, where $\hat{d}_A$ and $\hat{d}_B$ correspond to the normalized $d_A$ and $d_B$ in (a), respectively.

To represent a sequence of $t$ contours, a matrix $M$ of size $t \times L$ is formulated from all poses during the motion period:

$$M = [D_t, D_{t-1}, \ldots, D_1]^T_{t \times L}$$  \hspace{1cm} (2)

The shape descriptors are ordered temporally such that the first frame lies at the bottom row of the matrix.

We called $M$ a Motion Signature, which describes a motion as a 3D spatio-temporal surface. The 3D surface can also be visualized as a 2D gray level image with the $x$ axis denoting the contour points and the $y$ axis as time (i.e., frame number). The pixel intensity value is the normalized distance $\hat{d}_i$ which is further scaled linearly to fall within a range of $[0, 255]$. Fig. 2 shows both the 3D and 2D representations of $M$ for three gestures, i.e. “wave right hand”, “wave left hand” and “wave two hands”. It can be observed that the Motion Signature captures the dynamic properties of each motion and exhibits different patterns for each distinct gesture.
3 Temporal Endpoint Estimation

Unlike isolated gesture recognition, where the start and end times of a gesture are known a priori, for continuous gesture recognition it is critical to extract this information automatically. We assume that an actor could perform a gesture at any time and transition to another gesture with or without a punctuating pause. Therefore, the localization of the temporal endpoints of a gesture cannot rely on spotting the presence of a single special signal, such as a “standstill” condition.

Continuous gesture recognition is further complicated by the fact that although different motions will have significantly different Motion Signatures, distinct repetitions of a single gesture will also exhibit some spatial and temporal variations, among which motion speed (i.e. scale) is the most influential. Motion Signatures will appear to be either dilated when the motion is slow, or compressed when the motion is fast.

Because of the endpoint constraint, which requires the start and end points of two signals have to be aligned, DTW cannot be directly applied to continuous gesture recognition. In this work, we extend DTW to estimate the endpoints of a gesture in sequences by introducing the Compound Gesture Model as a reference signal.

3.1 Compound Gesture Model

To account for variations of the motion, particularly scale, a Gesture Model consisting of a set of Motion Signatures at various scales is first constructed. A number of image sequences are required for each gesture at each scale, and it would be labor intensive to collect Motion Signatures for all possible scales during training. As a practical alternative, only a single moderate scale is collected directly for each gesture, while the others are interpolated from this data. We found that if the actor does not change speed abruptly during one motion period (thereby combining multiple scales in one gesture), then this strategy produces highly accurate results. To build a Gesture Model, the actor is required to perform a gesture repeatedly at a single moderate speed in the training process.

We assume that the scale $s_{\text{max}}$ of the training gesture is the slowest scale that will be encountered. Given that each training motion lasts $t$ frames and that the increment between two successive scales is set to $\varepsilon$ frames, the total number of scales will be $t/\varepsilon$, which is equal to $s_{\text{max}}$. The Motion Signature of the training data at scale $s$ is produced using bilinear interpolation with the function $\phi$:

$$M_s = \phi(M_{s_{\text{max}}}, s, \varepsilon) \quad 1 \leq s \leq s_{\text{max}}$$

During training, for all actors, $H$ occurrences of each gesture are acquired at scale $s_{\text{max}}$. With this training set, the mean image and variance image are obtained by calculating the mean and standard deviation of each point on the Motion Signature using Eq. (4) and (5), respectively:

$$\mu_s(x, y) = \frac{1}{H} \sum_{k=1}^{H} M_{ks}(x, y)$$

$$\sigma_s(x, y) = \sqrt{\frac{1}{H-1} \sum_{k=1}^{H} (M_{ks}(x, y) - \mu_s(x, y))^2}$$
In Eq.(6), $R_i$ where along the temporal axis. The distance measurement $r_i$ constraints are imposed upon the path traversing process.

In the case of continuous gesture recognition, dynamically warping the test sequence to the Gesture Models will result in time warp will minimize the accumulated distance along the path through the grid from $(i,0)$ to $(t,t)$:

$$R_{i,t'} = r_{i,t'} + \min(R_{i,t'-1} + R_{i-1,t'} + R_{i-1,t'-1})$$

where

$$R_{1,1} = r_{1,1}$$
$$R_{i,1} = r_{i,1} + R_{i-1,1}$$
$$R_{i,t'} = r_{i,t'} + R_{i,t'-1}$$

In Eq.(6), $R_{i,t'}$ is the partial sum cost, and $r_{i,t'}$ measures the distance of gesture signals at two temporal instances. The test pattern is classified as the reference pattern where the accumulated distance is minimum.

The above classification criterion based on minimum distance value is only applicable for single pattern matching.

Although both the test sequence and Compound Gesture Model are 2D images, the warping is only performed along the temporal axis. The distance measurement $r_{i,t'}$ in Eq.(6) is defined as:

$$r_{i,t'} = \sum_{j=1}^{L} \frac{|Z_t(i,j) - \mu_r(i',j)|}{\sigma_r(i',j)}$$

3.2 Dynamic Time Warping and Temporal Endpoint Detection

Dynamic Time Warping makes use of dynamic programming to align two similar signals. A global optimum path is found by recursively accumulating the locally optimal paths. Given a test pattern $Z_t$ and reference pattern $Z_{t'}$, the best time warp will minimize the accumulated distance along the path through the grid from $(0,0)$ to $(t,t')$:

$$R_{i,t'} = r_{i,t'} + \min(R_{i,t'-1} + R_{i-1,t'} + R_{i-1,t'-1})$$

where

$$R_{1,1} = r_{1,1}$$
$$R_{i,1} = r_{i,1} + R_{i-1,1}$$
$$R_{i,t'} = r_{i,t'} + R_{i,t'-1}$$

In Eq.(6), $R_{i,t'}$ is the partial sum cost, and $r_{i,t'}$ measures the distance of gesture signals at two temporal instances. The test pattern is classified as the reference pattern where the accumulated distance is minimum.

The above classification criterion based on minimum distance value is only applicable for single pattern matching.

In the case of continuous gesture recognition, dynamically warping the test sequence to the Gesture Models will generally not give ideal results because the test sequence contains multiple gestures. Instead, we match the test sequence to all Compound Gesture Models to find the endpoints of the gesture. Given the endpoints of the head gesture, the endpoints of the motion in the test sequence can be obtained by finding the corresponding points in the warped path. Fig. 5 illustrates this idea: $t_0'$ and $t_1'$ are the start and end times of the head gesture of the Compound Gesture Model and $i_0$ and $i_1$ are the estimated endpoints of the test gesture. Notice that $t_0'$ is not the first frame of the Compound Gesture Model, but rather is the second frame. This is because the first frame of the Compound Gesture Model is always aligned to the first frame of the test sequence due to the endpoint constraint of DTW. By setting $i_0'$ to the second frame, we bypass the problem that the start time of the test gesture is fixed to be the first frame. No global constraints are imposed upon the path traversing process.

Although the test sequence and Compound Gesture Model are 2D images, the warping is only performed along the temporal axis. The distance measurement $r_{i,t'}$ in Eq.(6) is defined as:

$$r_{i,t'} = \sum_{j=1}^{L} \frac{|Z_t(i,j) - \mu_r(i',j)|}{\sigma_r(i',j)}$$

3.2 Dynamic Time Warping and Temporal Endpoint Detection

Dynamic Time Warping makes use of dynamic programming to align two similar signals. A global optimum path is found by recursively accumulating the locally optimal paths. Given a test pattern $Z_t$ and reference pattern $Z_{t'}$, the best time warp will minimize the accumulated distance along the path through the grid from $(0,0)$ to $(t,t')$:

$$R_{i,t'} = r_{i,t'} + \min(R_{i,t'-1} + R_{i-1,t'} + R_{i-1,t'-1})$$

where

$$R_{1,1} = r_{1,1}$$
$$R_{i,1} = r_{i,1} + R_{i-1,1}$$
$$R_{i,t'} = r_{i,t'} + R_{i,t'-1}$$

In Eq.(6), $R_{i,t'}$ is the partial sum cost, and $r_{i,t'}$ measures the distance of gesture signals at two temporal instances. The test pattern is classified as the reference pattern where the accumulated distance is minimum.

The above classification criterion based on minimum distance value is only applicable for single pattern matching.

Although both the test sequence and Compound Gesture Model are 2D images, the warping is only performed along the temporal axis. The distance measurement $r_{i,t'}$ in Eq.(6) is defined as:

$$r_{i,t'} = \sum_{j=1}^{L} \frac{|Z_t(i,j) - \mu_r(i',j)|}{\sigma_r(i',j)}$$

3.2 Dynamic Time Warping and Temporal Endpoint Detection

Dynamic Time Warping makes use of dynamic programming to align two similar signals. A global optimum path is found by recursively accumulating the locally optimal paths. Given a test pattern $Z_t$ and reference pattern $Z_{t'}$, the best time warp will minimize the accumulated distance along the path through the grid from $(0,0)$ to $(t,t')$:

$$R_{i,t'} = r_{i,t'} + \min(R_{i,t'-1} + R_{i-1,t'} + R_{i-1,t'-1})$$

where

$$R_{1,1} = r_{1,1}$$
$$R_{i,1} = r_{i,1} + R_{i-1,1}$$
$$R_{i,t'} = r_{i,t'} + R_{i,t'-1}$$

In Eq.(6), $R_{i,t'}$ is the partial sum cost, and $r_{i,t'}$ measures the distance of gesture signals at two temporal instances. The test pattern is classified as the reference pattern where the accumulated distance is minimum.

The above classification criterion based on minimum distance value is only applicable for single pattern matching.

Although both the test sequence and Compound Gesture Model are 2D images, the warping is only performed along the temporal axis. The distance measurement $r_{i,t'}$ in Eq.(6) is defined as:

$$r_{i,t'} = \sum_{j=1}^{L} \frac{|Z_t(i,j) - \mu_r(i',j)|}{\sigma_r(i',j)}$$
Figure 5: Temporal Endpoint Estimation Using DTW

where, $Z_t(i, j)$ is the input gesture at time $i$, and $\mu_t(i', j)$ and $\sigma_t(i', j)$ are the mean and variance images of the Compound Gesture Model at time $i'$. $L$ is the normalized length of the contour as defined in Eq.(1).

4 Bounded Search

Since a motion cannot be infinitely slow, it is reasonable to set an observation point at time $t_p$ in the test sequence such that at least one gesture is included backwards from this time. Once a gesture is recognized, the end time of that gesture becomes the start time of the next sequence and another observation point is set such that the length of the segment remains the same. In this way, we continually recognize gestures in a video stream.

Given a motion segment, a possible solution for classifying a gesture is to execute an exhaustive search through all Gesture Models at all possible start times and scales. Obviously, this exhaustive approach would be computationally prohibitive. More efficiently, we propose a two-phase process, where the temporal endpoint estimation method described in the previous section serves as an initial step to segment the gesture from the rest of the sequence. Warping two very different signals will result in a large accumulated distance value, so those Compound Gesture Models with the smallest DTW distances are considered to be the most similar to the test motion segment. In the second phase, only those Compound Gesture Models with the smallest DTW distances are further segmented around the endpoints and tested against the Gesture Models.

Suppose that $i_0$ and $i_1$ are the estimated endpoints of an input gesture. The scale $s$ of the gesture can then be easily obtained as: $s = (i_0 - i_1)/\varepsilon$. To accurately localize the gesture, we search through the neighbors of the estimated endpoints to find the best matched segment.

More precisely, a total of fifteen segments, whose contexts vary slightly according to the start times (i.e., $i_0 \pm \varepsilon$, $i_0$) and scales (i.e., $s \pm 1$, $s \pm 2$, and $s$), are tested for each Gesture Model. Whenever the motion segment is shorter than the maximum scale $s_{max}$ of a given Gesture Model, a similarity measure is evaluated between these motions at the corresponding scale; otherwise, the motion is compressed to the maximum scale and the comparison is conducted at the maximum scale. This process is detailed in the following equations:

$$M*_{J_{i_0}s} = \arg \max_{J, s} \left\{ \begin{array}{ll}
\frac{f(M_{i_0s}, G_{js})}{f(M'_{i_0s}, G_{js_{max}})} & 1 \leq s \leq s_{max} \\
1 & s > s_{max}
\end{array} \right. \quad (9)$$

where

$$M'_{i_0s} = \phi(M_{i_0s}, s_{max}, \varepsilon) \quad (10)$$

In the above, $P$ is the total number of motion patterns remaining after the initial DTW. $M_{i_0s}$ is a test Motion Signature from start time $i_0$ at scale $s$, and $G_{js}$ is the Gesture Model $j$ at scale $s$. $M'_{i_0s}$ is interpolated from $M_{i_0s}$ using the function $\phi$ defined in Eq.(3). The gesture and correct start time and scale is assigned to the Gesture Model with the maximum value found in Eq. (9). The function $f$ is used to measure the similarity between two motions, using the Mutual Information criterion [12, 18]. Mutual Information compares the statistical dependence of two patterns based upon a measure of their joint entropy, and is more robust than correlation which tends to be sensitive to variations in individual pixel intensity values [6].
5 Experimental Results

Experiments have been conducted to evaluate the performance of the proposed method. A total of eight gestures performed by five subjects were collected with an image size of 320 x 240 and a frame rate of 15 frames/sec. Fig. 6 shows several frames of four gestures for four different actors. Gestures No. 1 to 3, (namely “wave right hand”, “wave left hand” and “wave two hands”) are periodic gestures which contain repeated waving motions, while the remaining gestures No. 4 to 8 (“raise right hand”, “raise left hand”, “raise two hands”, “left down right up” and “left up right down”) are non-periodic. Since periodic gestures have a longer duration than the others, their maximum scale values $s_{max}$ are larger. Given that the scale interval $\varepsilon$ is set to 5, $s_{max}$ of the training data for the first three gestures is set to 11 (55 frames). For gestures No. 4 to 6, $s_{max}$ is set to 6 (30 frames), and No. 7 to 8, $s_{max}$ is 7 (35 frames). The values of $s_{max}$ in this experiment are generally smaller than that in [6], due to the fact that “stand still” is now separated from other gestures and regarded as an individual gesture, numbered as 0. The normalized length $L$ of the shape descriptor is also set larger than that in [6], which is 150.

Each gesture was learned using 30 instances performed at similar rates by three different actors. Another 60 instances per gesture were collected from two different subjects for testing. Since the goal was to test the capability of our approach for continuous gesture recognition, we manually constructed sequences composed of multiple gestures based on the acquired data. For each of the 8 gesture classes, every instance was tailed by the other 9 gestures (including gesture No. 0), which constituted 540 different testing sequences per gesture. All of the sequences were set to 120 frames in length: when the combined length of the head and tail gestures were less than 120, we padded the sequences with gesture No. 0; otherwise, gestures outside the range would be truncated. Moreover, the testing sequences were divided into two data sets. In data set 1, each gesture included 360 sequences whose constituent gestures had similar (although not exactly the same) temporal scales to that of the training data. Data set 2 was composed of another 180 sequences per gesture whose temporal scales were arbitrary, i.e. different from the training
data scale. The purpose of the first single temporal scale test was to provide a general idea of the performance of the approach under ideal conditions, while the second multiple scale test evaluated the robustness of the approach under arbitrary conditions.

The results for the two tests are listed in Table 1. The recognition rate in the table refers to the percentage of correctly recognized head gestures. For the single scale test, the average recognition rate reaches as high as 93.3%, which demonstrates that the proposed approach was very effective at recognizing gestures in continuous sequences.

The purpose of the multiple scale test was to demonstrate the robustness of our approach for more general time-varying gesture sequences. The last two columns of Table 1 summaries the scale variations for data set 2, from which it can be seen that the gestures were performed over a relatively large scale range. For example, for gesture No. 5, the slowest scale contained 131 frames, and was over 8 times slower than the fastest scale containing 16 frames. The smallest difference occurred for gesture No. 1, which had a factor of 2.2 between the slowest and fastest scales. The average recognition rate for the multiple scale test was 88.1%, which is quite high. Given the varied conditions, it’s not surprising that the average recognition rates have reduced somewhat compared to the single scale tests.

We also did some preliminary tests on longer sequences containing multiple gestures, which were acquired by having an actor move continuously through a specified sequences of gestures. Fig. 7 shows two of these sequences, where each gesture is labeled by two sets of numbers, which indicate the real (left) and the recognized (right) gesture numbers. The results are very promising, and both sequences only have one gesture misclassified. For sequence (b), between gesture No. 3 and 8, a small pause has been recognized as gesture No. 0, which is correct since “stand still” is regarded as one of the gesture types. From Fig. 7, we also found that the failure to identify a gesture won’t necessarily affect the identification of subsequent gestures. For example, for both sequences in Fig. 7, gesture No. 2 in the middle is misclassified, but all three remaining gestures are recognized correctly.

6 Conclusions and Future Work

We have presented a novel method for the recognition and segmentation of time-varying continuous human gestures, and experimental results have demonstrated the method to be effective. Using Compound Gesture Models, the temporal endpoints of a gesture were estimated by DTW, and a bounded search was performed to recognize the gesture. The proposed method is both computationally efficient and robust: in experiments containing 9 different gestures and 5 different subjects, the resulting average recognition rates were 93.3% for single scale and 88.1% for multiple scale continuous gestures.

In future work, we plan to build a larger database with more gestures and more subjects to test this approach. We are also implementing a real-time gesture recognition system based on the described method.

Acknowledgements

The authors would like to thank NSERC and Precarn for their financial support, the MONNET project team for providing the context for this work, and the anonymous reviewers for their helpful comments.

References


What are you looking at? Gaze estimation in medium-scale images

Neil Robertson †‡ Ian Reid† & Michael Brady†
†‡QinetiQ †Oxford University
E 206 Dept Engineering Science
St Andrews Road Parks Road
Malvern, WR14 3PS, UK Oxford, OX1 3PJ, UK

Abstract
In this paper we describe a new method for estimating where a person is looking in images where the head of a person is typically 20 pixels high. We use a feature vector based on skin detection to estimate the orientation of the head, which is discretised into 8 different orientations, relative to the camera. A fast sampling method returns a distribution over head pose relative to a camera centred frame. The overall body pose relative to the camera frame is approximated using the velocity of the body, obtained via colour-based tracking in the image sequence. We show that, by combining direction and head pose information gaze is determined more robustly than using each feature alone. We demonstrate this technique on surveillance and sports footage.

1 Introduction
In applications where human activity is under observation, be that CCTV surveillance or sports footage, knowledge about where a person is looking (i.e. their gaze) provides observers with important clues which enable accurate explanation of the scene activity. It is possible, for example, for a human readily to distinguish between two people walking side-by-side but who are not “together” and those who are acting as a pair. Such a distinction is possible when there is regular eye-contact or head-turning in the direction of the other person. In soccer head position is a guide to where the ball will be passed next i.e. an indicator of intention, which is essential for causal reasoning. In this paper we present progress towards automatically inferring gaze direction in images where any one person represents only a small proportion (the head is around 20 pixels high) of the frame.

The first component of our system is a descriptor based on skin colour. This descriptor is extracted for each head in a large training database and labelled with one of 8 distinct head poses. This labelled database can be queried to find either a nearest-neighbour match for a previously unseen descriptor or (as we discuss later) is non-parametrically sampled to provide an approximation to a distribution over possible head poses.

Recognising that general body direction plays an important role in determining where a person can look, we combine direction and head pose using Bayes’ rule to obtain the joint distribution over head pose and direction, resulting in 64 possible gazes (since head pose and direction are discretised into 8 sectors each, shown in figure 1).

The paper is organised as follows. Firstly we highlight relevant work in this, and associated, area(s). We then describe how head-pose is estimated in section 2. In section 3 we provide motivation for a Bayesian fusion method by showing intermediate results where the best head-pose match is chosen and, by contrast, where direction alone is used. Section 3 also discusses how we fuse the relevant information we have at our disposal robustly to compute a distribution over possible gazes, rejecting non-physical gazes and reliably detecting potentially significant interactions. Throughout the paper we test and evaluate on a number of datasets and additionally summarise comprehensive results in section 4. We conclude in section 5 and discuss potential future work in section 6.

1.1 Previous work
For human action recognition at a distance Efros [6] showed how to distinguish between human activities such as walking, running etc. by comparing gross properties of motion using a descriptor derived from frame-to-frame optic-flow and performing an exhaustive search over extensive exemplar data. Head pose is not discussed in [6] but the use of a simple descriptor invariant to lighting and clothing is of direct relevance to head pose estimation. Dee and Hogg [5] developed a system for detecting unusual activity which involves inferring which regions of the scene are visible to an agent within the scene. A Markov Chain with penalties associated with state transitions is used to return a score for observed trajectories which essentially encodes how directly a person made his/her way towards predefined goals, typically scene exits. In this work, clearly gaze inference is vital, but this is inferred from trajectory information alone which can lead to significant interactions being overlooked. In fact, many systems have been created to aid urban surveillance. The AI Lab at MIT has developed an entirely automated system for visual surveillance and monitoring.
of an urban site [9] but it appears that only trajectories are utilised. The same is true in the work of Buxton (who has been prominent in the use of Bayesian networks for visual surveillance) [2], Morellas et al [18] and Makris [15]. Johnson and Hogg’s work [12] is another example where trajectory information is only considered.

Gee and Cipolla’s [8] gaze determination method based on the 3D geometric relationship between facial features was applied to paintings to determine where the subject is looking. In medium-scale images locating significant features such as the eyes and corners of the mouth as used in [8] is an impossible task. Related work has tackled expression recognition using information measures. Shinohara and Otsu demonstrated that Fisher Weights can be used to recognise “smiling” in images. Unsurprisingly, the main application focus of gaze recognition work has been Human-Computer Interfaces and the technical aspects have focused on detecting the eyeball primarily. Matsumoto [16] computes 3-D head pose from 2-D features and stereo tracking. Perez et al. [21] focus exclusively on the tracking of the eyeball and determination of its observed radius and orientation for gaze recognition. Kaminski et al. [13] have achieved a very similar goal but using a single image while retaining a face and eye model. While this is most useful in HCI where the head dominates the image and the eye orientation is the only cue to intention, this approach is too fine-grained for surveillance video where it must be assumed the eye is aligned with head-pose.

Skin detection has received much attention in the Computer Vision community [3] [10] [11], but it is clear that determining gaze in surveillance images is a challenging problem that has received little or no attention by the vision community. We recognise that skin detection alone will be too error-prone when the skin region is very small as a proportion of the image. However, additional cues such as direction can help to disambiguate gaze using even a very coarse head-pose estimation. By combining this information in a principled (i.e. probabilistic, Bayesian) fashion, gaze estimation at a distance becomes a distinct possibility as we demonstrate in this paper.

2 Head pose detection

2.1 Head pose feature vector

Though people differ in colour and length of hair and some people may be wearing hats it is reasonable to assume that the amount of skin that can be seen and the position of the skin pixels within the frame is a relatively invariant cue for a person’s coarse gaze in a static image. To obtain this descriptor there is a small degree of manual intervention required. First, a mean-shift tracker [4] is hand-initialised on the head. While we anticipate this could be done automatically in the future by modelling the person as distinct “blocks” e.g. head and torso, in this work we concentrate on gaze estimation and assume we have a coarse estimate of which part of a moving “blob” is the head. (In practice we find the meanshift algorithm, when hand-initialised is stable in its fixation on the region of interest.) Second, because there is no specific region of colour-space which represents skin in all sequences it is necessary to define a skin histogram for each scenario. We hand-select a region of one frame in the current sequence to compute a (normalised) skin-colour
Figure 2: The original target frame (bottom-left) is used to compute a descriptor representing skin pixel likelihood as shown in figure 1. A set of Principal Components is computed from this descriptor and the exemplar database (which has been formulated as a binary tree split on the sign of the Principal Components derived from the training dataset) is sampled in a probabilistic fashion to obtain a distribution over potential matches. The distribution resulting from 10 samples of the database for this input frame is shown in the graph above. The leaf nodes of the database contain indices into matching frames and the matching frame images and assigned probabilities of a match are shown below the graph.

In RGB-space. We then compute the weights for every pixel in the head images which the tracker produces to indicate how likely it is that it was drawn from this predefined skin histogram. Every pixel in each tracked head image is drawn from a specific RGB bin and so is assigned the relevant weight which can be interpreted as a probability that the pixel is drawn from the skin model histograms. So for every bin \( i \) in the predefined, hand-selected skin-colour histogram \( q \) the histogram of the tracked image \( p \) is a weight is computed

\[
    w_i = \sqrt{\frac{q_i}{p_i}}
\]  

(1)

every pixel in the tracked frame where falls into one of the bins according to its RGB value and the normalised weight associated with that pixel is assigned to compute the overall weight image, as shown in figure 1. The weight image therefore defines our feature vector for head orientation per frame.

### 2.2 Training data

We assume that we can distinguish head pose to a resolution of 45 degrees. There is no obvious benefit to detecting head orientations at a higher degree of accuracy and it is unlikely that the coarse target images would be amenable in any case. This means discretising the 360 degrees orientation-space into 8 distinct views as shown in figure 1. (In this first attempt we have not made provision for scale changes.) The training data we select is from a surveillance-style camera position and around 100 examples of each view are selected. The head was tracked and the example labelled accordingly. The weight image for each frame is then computed and this feature vector stored in our exemplar set. **The same example set is used in all the experiments reported** (e.g. there are no footballers in the training dataset used to compute the gaze estimates presented in figure 8).
Figure 3: Detecting head pose in different scenes using the same exemplar set. The main image shows the frame with the estimated gaze angle superimposed, the pair of images directly beside each frame shows the input image that the head-pose detector uses (top) and the best (ML) match in the database with corresponding label (bottom).

2.3 Matching head poses

The descriptors for each head pose are \((20 \times 20 =) 400\) element vectors. With 8 possible orientations and 100 examples of each orientation searching this dataset rapidly becomes an issue. Although linear-time nearest-neighbour search is not intractable unless near real-time performance is desired we elect to structure the database using a binary-tree in which each node in the tree divides the set of exemplars below the node into roughly equal halves. Such a structure can be searched in roughly \(\log n\) time to give an approximate nearest-neighbour result. We do this for two reasons: first, even for a modest database of 800 examples such as ours it is faster by a factor of 10; second, we wish to frame the problem of gaze detection in a probabilistic way and Sidenbladh [23] showed how to formulate a binary tree (based on the sign of the Principal Components of the data) search in a pseudo-probabilistic manner. This technique was later applied to probabilistic analysis of human activity in [22]. We achieve recognition rates of 80\% using this pseudo-probabilistic method based on Principal Components with 10 samples. An example of such a distribution in this context is shown in figure 2. Results of sampling from this database for a number of different scenes are shown in figure 3.

3 Gaze estimation

3.1 Bayesian fusion of head-pose and direction

The naive assumption that direction of motion information is a good guide as to what a person can see has been used in figure 6. However, it is clear the crucial interaction between the two people is missed. To address this issue we compute the joint posterior distribution over direction of motion and head pose. The priors on these are uniform for direction of motion, reflecting the fact that for these purposes there is no preference for any particular direction in the scene, and for head pose a centred, weighted function that models a strong preference for looking forwards rather than sideways. The prior on gaze is defined using a table which lists expected (i.e. physically possible, \(p = 0.8\)) gazes and unexpected (i.e. non-physical, \(p = 0.2\)) gazes.

We define \(g\) as the measurement of head-pose, \(d\) is the measurement of body motion direction, \(G\) is the true gaze direction in camera frame and \(B\) is the true body direction in the camera frame. We compute the joint probability of true body pose and true gaze:

\[
P(B, G|d, g) \propto P(d, g|B, G)P(B, G)
\]

Now given that the measurement of direction \(d\) is independent of true gaze \(G\) once true body \(B\) pose is known:

\[
P(d|B, G) = P(d|B)
\]

1This will be recognised as a similar approximation to the Battacharyya coefficient as implemented in the meanshift algorithm [4].
Figure 4: Fusing head-pose and direction estimates improves gaze estimation. In this example, the ML match for head pose is incorrectly chosen as “back” as shown at the top right. The direction is correctly identified as “S” (top left). It is not possible to turn the head through 180 degrees (relative to the body) therefore this gaze has a low (predefined) prior and is rejected as the most likely at the fusion stage. The MAP gaze is identified as “Face” as shown in the bottom-left image which is a very good approximation to the true gaze as observed in the video sequence. The angle of this estimated gaze is superimposed on the frame (bottom-right).
Estimating gaze using only head-pose

NON-PHYSICAL

Estimating gaze from direction-of-motion only

INTERACTION MISSED  INCORRECT  INCORRECT

Figure 5: In this example the tracked person turns to look at the other person on his right twice. This is a significant visual cue strongly suggesting the people are together. Estimating gaze using the head-pose estimate alone (first row), results in a non-physical gaze (second frame) since the head must turn through more than 90 degrees. However there is no mechanism for rejecting such a gaze, even if the likelihood is objectively very low, when the direction of the person is not incorporated into the gaze estimation calculation. Computing gaze using only the direction-of-motion estimated from the trajectory results in critical interactions being missed (second row). The turning of the head is not reliably detected (see frame 2) and, moreover, it is estimated that the second person is in view in images 3 and 4, which is incorrect. (Note the frames shown are not for direct comparison between the rows but are selected to highlight the weaknesses of each method.) Our method for improving gaze estimation is to fuse the estimates from direction-of-motion and head-pose. This is described in the text (see also figure 4) and the results of applying the technique to this sequence are shown in figure 6.

and, similarly, that the measurement of gaze $g$ is independent of true body pose $B$ given true gaze $G$:

$$P(g|B, G) = p(g|G)$$

(4)

Therefore,

$$P(B, G|d, g) \propto P(g|G)P(d|B)P(G|B)P(B)$$

(5)

Now $P(g|G)$ is the likelihood of measurement of gaze which is a Gaussian centred on the current measurement of gaze,

$$P(g|G) = \mathcal{N}(g, \sigma^2)$$

(6)

Similarly, $P(d|B)$ is the likelihood of measurement of direction,

$$P(d|B) = \mathcal{N}(d, \sigma^2)$$

(7)

$P(G|B)$ is specified in advance in a look-up table which allows us to reject non-physical gazes. $P(B)$ is the prior on direction which, in the general case, is uniform but need not be e.g. in tennis one player is expected to be predominantly facing the camera.

Now we compute the joint distribution for all 64 possible gazes resulting from possible combinations of 8 head poses and 8 directions. This posterior distribution allows us to maintain probabilistic estimates without committing to a defined gaze which will be advantageous for further reasoning about overall scene behaviour. Immediately though we can see that gazes which we consider very unlikely given our prior knowledge of human biomechanics (since the head cannot turn beyond 90 degrees relative to the torso [20]) can be rejected in addition to the obvious benefit that the quality of lower-level match can be incorporated in a mathematically sound way. An example is shown in figure 4.
Gaze estimation using head-pose and direction-of-motion

Figure 6: By fusing the direction-of-motion information and head-pose estimates the MAP gaze is much improved and the crucial interaction is captured. Contrast the gaze estimation in this figure with that in figure 5. The error between the MAP estimate and the hand-labelled ground truth is shown in the graphs. The mean of the absolute value errors here is 22.27 degrees, the median zero degrees. This error corresponds to one half of our discretisation of head angle (which is 45 degrees), and is a clear improvement on using the ML head-pose or direction estimate alone. Moreover the errors are isolated and Markov smoothing (either Kalman Filter or HMM) of the head-pose could improve the results even further. The frames for which the head is turned are clearly evident since the angle relative to zero degrees vertical (which represents the view of the back of the head) in the image plane increases significantly.

Figure 7: Second surveillance sequence. The same training data set as used to obtain the results above is used to infer head pose in this video. The ground truth is estimated by hand from the images. The mean error is 5.64 degrees, the median 0.5 degrees.
Figure 8: This final example demonstrates the method in soccer footage. The skin histogram is defined only at the start of this sequence to compensate for lighting changes, but the exemplar database remains the same as that constructed initially and used on all the sequences i.e. it contains no examples from this sequence.
4 Results

We have tested this method on various datasets (see figures 5, 6, 7 and 8). The first dataset provided us with the exemplar data for use on all the test videos shown in this paper. In the first example in figure 6 we show significant improvement over using head-pose or direction alone to compute gaze (c.f. figure 6, top-left). The crucial interaction which conveys the information that the people in the scene are together is the frequent turning of the head to look at each other. We reliably detect this interaction as can be seen from the images and the estimated head angle relative to vertical. The second example is similar but in completely different scene. The skin histogram is recomputed for this video but the training data remains the same. Once more the interaction implied by the head turning to look at his companions is determined. Finally we demonstrate the method on sports video in figure 8.

5 Conclusions

In this paper we have demonstrated that a simple descriptor, readily computed from medium-scale video, can be used to estimate head pose robustly. In order to speed up non-parametric matching into an exemplar database and to maintain probabilistic estimates throughout we employed a fast pseudo-probabilistic binary search based on Principal Components. To resolve ambiguity, improve matching and reject known implausible gaze estimates we used a simple application of Bayes’ Rule to fuse priors on direction-of-motion and head-pose, evidence from our exemplar-matching algorithm and priors on gaze (which we specified in advance). We demonstrated on a number of different datasets that this gives acceptable gaze estimation for people being tracked at a distance.

6 Future work

One source of error is the video tracker which can produce inconsistency in the positions of the skin pixels in the target frame. Matches are to some degree dependent on the location of the skin pixels in the centre of the frame and tracking inconsistency can cause discrepancies to arise. This needs to be investigated. Additionally a uniform skin-colour histogram would improve our method by preventing the re-initialisation of skin colour for different lighting conditions.

It seems to us the work reported here would be most useful in a causal reasoning context where knowledge of where a person is looking can help solve interesting questions such as, “Is person A following person B?” or determine that person C looked right because a moving object entered his field-of-view. We are currently combining this advance with our reported work on human behaviour recognition [22] to aid automatic reasoning in video.

References

[2] H. Buxton. Learning and Understanding Dynamic Scene Activity. ECCV Generative Model Based Vision Workshop, Copenhagen, Denmark, 2002


Human Gait Recognition with 3D Wavelets and Kernel based Subspace Projections

Nasir Rajpoot1 Khalid Masood2

1 Department of Computer Science, University of Warwick, UK
2 COMSATS Institute of Information Technology, Pakistan

Abstract

Gait recognition can be regarded as a problem of uniquely representing spatiotemporal surfaces associated with a person’s walking pattern in an efficient manner. In this paper, we describe the approach of using projections of such surfaces onto subspace spanned by appropriate axes using a single framework. Two new algorithms for gait recognition are presented which use projection on subspace of kernel induced higher dimensional spaces using PCA and Fisher’s LDA. Wavelet transform in 3D is used to reduce the complexity of the problem. The proposed methods have been applied to datasets containing subjects walking at three different viewing angles in outdoor environment. The results show high accuracy results for recognizing subjects from their gait, even with a thumbnail size (16 × 16 × 12) of the gait patterns.

1 Introduction

Gait is the walking pattern of an individual. It has been shown [8, 2] that the human gait can be used as a biometric in passive surveillance applications. One of the reasons for it to be used as a biometric is that the human gait consists of synchronized integrated movements of hundreds of muscles and joints in an almost unique way for an individual [2]. Gait recognition is the task of identifying an individual from analysing their gait video. First, we present a brief review of the relevant work and explain our approach.

1.1 Relevant Work

A variety of methods for gait recognition can be found in the literature. Niyogi and Adelson [11] observed that the spatiotemporal behaviour of the human gait can be approximated by a smooth periodic surface. Indeed, a sequence containing a walking person can be processed (often simply by removing the background) to extract such periodic surfaces in 3D, assuming the background is static. In [11], such periodic surfaces are expressed as a combination of a standard parameterized surface, the so-called canonical walk, and a deviation surface which is more directly related to an individual’s walk. Ben Abdelkader et al. [3] compute self-similarity plots (SSPs) from the spatiotemporal volume of a walking person by alignment and scaling of blobs containing silhouettes in each frame. Normalized SSPs are computed and standard classification algorithms are used in order to identify a person. Lee and Grimson [8] divide silhouettes of a walking person into seven regions and fit one ellipse to each region. Four parameters for each of the ellipses and one additional parameter for height of the silhouette make up their feature vector. Spatiotemporal analysis based on the Fourier transform is proposed by Ohara et al. [12], where Fourier transform is applied to the extracted gait volume and characteristic frequency of a walking person is determined. Three-dimensional (3D) kinematic models [4] have also been used to achieve good performance for individual recognition by their gait.

Perhaps of a more direct relevance to our work in this paper are subspace projection methods proposed in the literature. For instance, Mursae and Sakai [10] used PCA on the silhouette sequences to represent a person’s gait by coordinates of projections on the subspace spanned by a few principal components. Huang et al. [6] represented gait in canonical space on temporal templates, recorded from optical flow changes between two consecutive spatial templates. Matching of the templates projected from high dimensional image space to low dimensional canonical space was shown to produce excellent results, although only one viewing angle is considered for training and testing. In a slightly different approach, Wang et al. [14] proposed to use normalized distance signals, made up of distance values between the silhouette centroid and the boundary pixels, to approximate temporal patterns of gait. Principal component analysis (PCA) is applied to reduce the dimensionality of the distance signal sequences and a gait signature consisting of projections on the lower dimensional eigenspace is generated.

1.2 Our Approach

We perceive the problem of gait recognition as that of the representation of approximately periodic spatiotemporal surfaces in an efficient manner, preferably using only a few parameters. It is to be noted that the spatiotemporal surfaces associated with a person’s gait normally consist of intertwined surfaces due to the motion of legs, torso, and
the arms. The representation of such complex surfaces using measures such as gait frequency and phase is not sufficient to characterise a person’s gait. We represent these surfaces as points in some high dimensional space $\mathbb{R}^n$, where $n$ is the size of the spatiotemporal volume containing the gait pattern. Based on the fact that the surfaces associated with human gait belong to a particular class of surfaces, our hypothesis is that these surfaces can be efficiently represented with a few parameters using projections onto appropriate axes spanning a subspace of $\mathbb{R}^n$.

If a gait recognition system employing such a representation is to work, it will be required that gait surfaces of the same individual are mapped to points close by in the subspace, regardless of the viewpoint, illumination variations, person’s mood, and speed of walking. Our work in this paper is motivated by the success of kernel based subspace projection methods for face recognition [15, 9]. Kernel based methods have also been successful in other applications, see [13] for a detailed list. The basic idea behind support vector machines (SVMs) and other kernel based methods is that mapping the data from input space to a higher dimensional space often enables linear separation of data points belonging to different classes, which may otherwise not be linearly separable. We show that canonical space representation of the spatiotemporal gait patterns, when mapped onto a kernel induced higher dimensional space $\mathbb{R}^f$ where $f > n$, is view-invariant and produces high recognition accuracy. Wavelet transform in 3D is used to obtain smaller resolution approximation of the gait patterns in order to reduce both storage and computational complexity.

In the next section, the training and recognition framework used in our work is presented. Building blocks of the framework are described in some detail. Experimental results are presented and discussed in Section 3, and conclusions and future directions are summarized in the final section.

## 2 The Training/Recognition Framework

![Block diagram of the proposed framework](image)

A block diagram of the training/recognition framework employed in this work is shown in Figure 1. Let $C$ and $N_c$ respectively denote total number of subjects and number of training sequences for a particular subject $c$. Further, let $N$ and $K$ respectively denote the total number of sequences used for training and the number of representative gait patterns (i.e., the axes of subspace used for projection, principal vectors in case of PCA and generalized eigenvectors in case of LDA). At the training stage, representative gait patterns $R_i(x,y,t)$ for $i = 1, \ldots, K$ are computed using one of the subspace analysis methods (Section 2.3). Coordinates of the gait surfaces $G_i(x,y,t)$ for $i = 1, \ldots, N$, with respect to $R_i(x,y,t)$ are calculated and recorded as gait signatures. The recognition system takes as input a test sequence, computes the projection of its associated gait pattern onto the representative axes, and finds the closest match in terms of the normalized Euclidean distance.

### 2.1 Pre-Processing

The main purpose of the pre-processing stage is to extract spatiotemporal surfaces associated with gait pattern from the given sequence of a walking person. This is achieved by applying the following three modules:
i) **Background modelling:** A background image containing the scene information and without the walking figure can be reliably generated by taking the median of the image sequence. Assuming that the camera is stationary and only the object is moving, the background can be estimated as follows,

\[ B(x,y) = \text{median}\{I(x,y,t)\} \]  

where \(1 \leq t \leq 60\) for each of the sequences.

ii) **Background subtraction:** In our experiments, we found that simple subtraction of the background through differencing between the background and the original frame, reversing the order, and adding the two subtracting images gives an equally good estimate of the foreground as compared to a relatively complex extraction function proposed in [7]. We chose to use the differencing approach due to its relatively low complexity.

iii) **Image enhancement:** Morphological erosion is used to filter spurious pixels and dilation is then used to fill any remaining holes. Histogram equalization and some contrast adjustment are performed before tracking each foreground region from frame to frame and placing a bounding box over the silhouette, with the size of the bounding box being the maximum of all the silhouette sizes. It is worth noting that unlike most other approaches to gait recognition, our silhouettes are not binary.

![Figure 2: Results of pre-processing](image)

Results of pre-processing for different views (the red rectangle is the bounding box): (a)–(c) fronto-parallel, (d)–(f) subject walking at 45° to the camera, (g)–(i) subject walking at 90° to the camera. Two-dimensional views of spatiotemporal surfaces \(G_i(x,y,t)\) can be seen in (c), (f), and (i) for all three views.

### 2.2 3D Wavelet Decomposition

The pre-processed gait sequence \(G(x,y,t)\) is decomposed into frequency subbands using a 3D wavelet transform. The lowest frequency subband gives a coarse approximation of the gait sequence, while the remaining subbands contain its contents corresponding to medium-to-high frequency phenomena in all three directions \(x, y,\) and \(t\). Our experiments with subspace analysis on a number of combinations of different subbands of the 3D wavelet decomposition of \(G(x,y,t)\) revealed that it was sufficient to perform subspace analysis on the lowest frequency subband. These findings are in agreement with an earlier study [5] which concluded that changes in angles or directions affect only the low-frequency spectrum and that only a change in subject will affect all frequency components. Hence we consider
only a coarse approximation \( g(x, y, t) \) of the gait sequence using the lowest frequency subband. Doing so not only substantially reduces the computational complexity of subsequent operations, it also allows us to accommodate the intermediate calculations (such as scatter matrices, see Section 2.3) in the memory.

### 2.3 Subspace Analysis/Projection

Subspace analysis can be regarded as the most important of all the components of training part of the framework in Figure 2. We employed four different methods for finding the representative gait patterns: principal component analysis (PCA), Fisher’s linear discriminant analysis (LDA) [1], kernel PCA [13], and kernel LDA [9]. For the sake of completeness, here we provide an overview of all the methods in the context of the representation of spatiotemporal surfaces associated with an individual’s gait.

#### 2.3.1 PCA

PCA is a classical statistical method commonly used for dimensionality reduction. Given \( N \) gait sequences rearranged as \( n \)-dimensional points, PCA allows us to project the data points onto first \( K \) directions (\( K < N \)) while capturing the maximum variance of the data. The first \( K \) directions, which can be thought of as the first \( K \) major axes of an \( N \)-dimensional ellipsoid, are the eigenvectors corresponding to the largest \( K \) eigenvalues of the following covariance matrix:

\[
C = \frac{1}{N} \sum_{i=1}^{N} (g_i - \bar{g})(g_i - \bar{g})^T
\]

where \( \bar{g} \) denotes the average of all \( N \) gait vectors.

#### 2.3.2 LDA [1]

One of the problems with subspace projections using PCA is that it does not explicitly take into consideration any variation there may be between a class of gait patterns of the same subject. The LDA method, as described by Belhumeur et al. [1], uses both PCA and LDA to produce a subspace projection matrix, minimizing within-class variation and maximizing between-class variation. Two scatter matrices, within-class scatter \( S_w \) and between-class scatter \( S_b \), are computed for the training data set as follows.

\[
S_b = \sum_{c=1}^{C} N_c (\bar{g}_c - \bar{g})(\bar{g}_c - \bar{g})^T
\]

\[
S_w = \sum_{c=1}^{C} \sum_{i=1}^{N_c} (g_i - \bar{g}_c)(g_i - \bar{g}_c)^T
\]

where \( \bar{g} \) is as defined previously, and \( \bar{g}_c = \frac{1}{N_c} \sum_{i=1}^{N_c} g_i \) is the average of each individual’s gait patterns. If \( S_w \) is non-singular, the optimal matrix containing eigenvectors \( W_{opt} \) is chosen as the matrix with orthonormal columns which maximizes the following ratio,

\[
W_{opt} = \arg\max_W \frac{|W^T S_b W|}{|W^T S_w W|}
\]

where \( W_{opt} = [w_1, w_2, \cdots, w_{C-1}] \) is a matrix whose column vectors are generalized eigenvectors (forming the representative gaits for subspace projection during the recognition stage in Figure 2) of \( S_b \) and \( S_w \) corresponding to the \( K \) largest generalized eigenvalues \( \{\lambda_i \mid i = 1, 2, \cdots, K\} \) satisfying the equation \( S_b w_i = \lambda_i S_w w_i \).

#### 2.3.3 Kernel PCA [13]

The basic assumption behind standard PCA is that the probability distribution in the \( n \)-dimensional gait space is a multidimensional Gaussian, making it possible to reduce the dimensionality of the problem by projecting the data onto first few principal axes capturing a large amount of the variation among the data. In case of non-Gaussian distributions, one way to deal with the problem is to make the data in the gait space more Gaussian-like (and hence more easily separable) by non-linearly mapping it to a higher dimensional space \( \mathcal{R}^f \) using an appropriate map \( \Phi(\cdot) : \mathcal{R}^n \to \mathcal{R}^f \), with \( f > n \).

The so-called kernel trick allows us to reduce the complexity of the mapping by using a kernel which when applied to two data points is equivalent to the dot product of the data points mapped to \( \mathcal{R}^f \). PCA on \( \mathcal{R}^f \) can be applied as follows. Let \( \mathbf{C}^{\Phi} \) denote the covariance matrix of data points from the training set mapped onto \( \mathcal{R}^f \). In order to compute the eigenvectors in \( \mathcal{R}^f \), the following eigenvalue problem needs to be solved

\[
\lambda w^{\Phi} = \mathbf{C}^{\Phi} w^{\Phi}
\]
where \( \mathbf{w}^\Phi \) denotes eigenvector in \( \mathbb{R}^f \) and for all non-zero eigenvalues, it lies in the span of \( \Phi(\mathbf{g}_1), \Phi(\mathbf{g}_2), \ldots, \Phi(\mathbf{g}_N) \). In other words,

\[
\mathbf{w}^\Phi = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{g}_i)
\]

(7)

If we denote by \( \mathbf{K} \) the kernel matrix whose elements are given by

\[
K_{ij} = k(\mathbf{g}_i, \mathbf{g}_j) = \Phi(\mathbf{g}_i) \cdot \Phi(\mathbf{g}_j),
\]

(8)

it can be shown that the eigenvalue problem of (6) is equivalent to the following eigenvalue problem [13],

\[
N \lambda \alpha = \mathbf{K} \alpha.
\]

(9)

Solving the above equation yields eigenvalues \( \lambda \) and corresponding eigenvectors \( \alpha \) consisting of \( N \) elements. Importantly, it does not require explicit computation of the mapping \( \Phi \). A knowledge of the kernel matrix \( \mathbf{K} \) suffices.

Given a test gait pattern \( \mathbf{g} \), its projection onto an eigenvector \( \mathbf{w}^\Phi \) of \( \mathbb{R}^f \) can be calculated using the following equation:

\[
\mathbf{w}^\Phi \cdot \Phi(\mathbf{g}) = \sum_{i=1}^{N} \alpha_i (\Phi(\mathbf{g}_i) \cdot \Phi(\mathbf{g})) = \sum_{i=1}^{N} \alpha_i k(\mathbf{g}_i, \mathbf{g})
\]

(10)

Extracting the first \( K \) eigenvectors \( \mathbf{w}^\Phi \) gives nonlinear principal components using the kernel function without explicitly mapping onto \( \mathbb{R}^f \). In the gait space, first \( K \) eigenvectors of (9) give representative gaits which can be used for projections as above.

### 2.3.4 Kernel LDA [9]

Similar to kernel PCA, kernel LDA in its standard form is the application of LDA to the high-dimensional space \( \mathbb{R}^f \).

In other words, it requires the solution to the following generalized eigenvalue problem,

\[
\lambda \mathbf{S}^\Phi \mathbf{w}^\Phi = \mathbf{S}^\Phi \mathbf{w}^\Phi
\]

(11)

which can be obtained by

\[
\mathbf{w}^\Phi_{opt} = \arg\max_{\mathbf{w}^\Phi} \frac{\mathbf{w}^\Phi^T \mathbf{S}^\Phi \mathbf{w}^\Phi}{\mathbf{w}^\Phi^T \mathbf{S}^\Phi \mathbf{w}^\Phi} = [\mathbf{w}^\Phi_1, \mathbf{w}^\Phi_2, \ldots, \mathbf{w}^\Phi_m].
\]

(12)

Once again, any solution \( \mathbf{w}^\Phi \in \mathbb{R}^f \) should lie in the span of all training samples in \( \mathbb{R}^f \). Mathematically,

\[
\mathbf{w}^\Phi = \sum_{c=1}^{C} \sum_{i=1}^{N_c} \alpha_{c,i} \Phi(\mathbf{g}_i)
\]

(13)

It can be shown that the solution to above equation can be found by solving the following problem [15]:

\[
\lambda \mathbf{K} \mathbf{K} \alpha = \mathbf{K} \mathbf{Z} \mathbf{K} \alpha
\]

(14)

where \( \mathbf{Z} = (\mathbf{z}_c)_{c=1}^{C} \) and \( \mathbf{z}_c \) is a \( N_c \times N_c \) matrix with all its terms equal to \( 1/N_c \). Projections onto eigenvectors \( \mathbf{w}^\Phi \) can be computed in the same way as in kernel PCA.

### 3 Experimental Results

In this section, we compare the performance of all the four subspace projection techniques on pre-processed training and test sequences containing a walking person. We selected NLPR dataset because of its complexity in viewing directions and use of outdoor environment. Our experiments were conducted on 15 subjects walking at three different angles relative to the camera, as shown in Figure 2.1, recorded at two different times on two different days. The data contains subjects walking at different speeds in two opposite directions for each of the angles. Half of the dataset, that is six sequences per subject (two per angle), is taken as the training set and the other half is used for the test purposes. The pre-processing resulted in gait sequences \( \mathbf{G}_i(x, y, t) \), each of dimensions \( 64 \times 64 \times 48 \). Coarse approximation for each of the gait sequences was obtained by extracting the lowest frequency subband after performing a two-level wavelet transform using Daubechies-4 filters, giving \( 16 \times 16 \times 12 \) sequence \( g_i(x, y, t) \) to be used for subspace analysis (projection) during training (testing). Kernel PCA uses a polynomial kernel of degree 2, which gave better accuracy as compared to the Gaussian kernel in our experiments. Kernel LDA is performed with tuning of polynomial kernel of degree 3, which produced the best results.

In order to have a quantitative evaluation of the performance, normalized coordinates of each test subject sequence is compared with the sequences in the training set using the normalized Euclidean distance. The recognition accuracy for each technique is defined as the ratio of the number of sequences correctly identified to the total number of
sequences in the tests. A graph of recognition accuracy against the number of subspace components $K$ is shown in Figure 3. The value of $K$ is varied from 4 to 30 for PCA and kernel PCA, whereas the maximum value of $K$ is one less than the number of subjects in the training set for both LDA and kernel LDA. The best results for all four algorithms are shown in Table 1.

Based on these results, we make the following observations. Kernel LDA is the overall winner in terms of accuracy, although it is computationally intensive, has high memory requirements, and requires careful tuning. PCA is relatively faster in computation but least accurate. Kernel PCA performance is closely matched by PCA, similar to the results in [15], perhaps due to the inclusion of difficult viewing angles. Nevertheless, the fact that both the training and test data contained three different viewing angles shows that the results are promising. Extensive experimentation using different datasets is the next logical step of our research.

Based on these results, we make the following observations. Kernel LDA is the overall winner in terms of accuracy, although it is computationally intensive, has high memory requirements, and requires careful tuning. PCA is relatively faster in computation but least accurate. Kernel PCA performance is closely matched by PCA, similar to the results in [15], perhaps due to the inclusion of difficult viewing angles. Nevertheless, the fact that both the training and test data contained three different viewing angles shows that the results are promising. Extensive experimentation using different datasets is the next logical step of our research.

![Figure 3: Comparative Results of Four Subspace Projection Methods](image)

<table>
<thead>
<tr>
<th>Method</th>
<th>$K$</th>
<th>Kernel</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>20</td>
<td>-</td>
<td>75.5</td>
</tr>
<tr>
<td>Kernel PCA</td>
<td>20</td>
<td>polynomial (order 2)</td>
<td>76.6</td>
</tr>
<tr>
<td>LDA</td>
<td>14</td>
<td>-</td>
<td>83.3</td>
</tr>
<tr>
<td>Kernel LDA</td>
<td>14</td>
<td>polynomial (order 3)</td>
<td>86.6</td>
</tr>
</tbody>
</table>

Table 1: Best recognition results using the four methods

4 Conclusions

In this paper, we presented a comparison of four subspace projection methods for gait recognition. After pre-processing, the gait sequence was approximated at a coarse resolution using 3D wavelet transform thus reducing the storage and computational complexity of the subsequent subspace analysis. Two of the methods were based on the subspace analysis of the kernel induced higher dimensional space, which have not previously been reported in the literature to the best of our knowledge. Kernel LDA produces the best results using a polynomial kernel, while LDA offers a relatively less expensive alternative with competitive performance. Our results show that view-invariant representation of the spatiotemporal surfaces associated with a person’s gait may be achievable using kernel based analysis methods. It is our belief that subspace projection can also be used to recognize certain human activities. More research, however, is needed to improve robustness of these methods.
Acknowledgements

The authors would like to thank the NLPR administrators for providing access to their gait database used in this work.

References


Human Activity Recognition from Video: modeling, feature selection and classification architecture

Pedro Canotilho Ribeiro & José Santos-Victor∗
Instituto Superior Técnico
Instituto de Sistemas e Robótica
Lisboa, Portugal
{pribeiro, jasv}@isr.ist.utl.pt

Abstract

In this paper, we address the problem of recognizing human activities, such as \{Active, Inactive, Walking, Running, Fighting\} from video sequences, with a particular emphasis on the problems of feature selection, data modeling and classifier structure.

The need for such systems is increasing everyday, with the number of (hundreds or thousands) of surveillance cameras deployed in public spaces. This massive number of cameras calls for systems able to detect, categorize and recognize human activity, requesting human attention only when necessary.

Our work is focused on three fundamental issues: (i) the design of a classifier and data modeling for activity recognition; (ii) how to perform feature selection and (iii) how to define the structure of a classifier.

We use a Bayesian classifier, and model the likelihood functions as Gaussian mixtures, adequate to cope with complex data distributions, that are learned automatically. As for feature selection, we propose several (suboptimal) methods to evaluate the recognition rate achieved with different feature combinations, with the Bayesian classifier. Finally, we investigate the use of hierarchical classifiers (including the possibility of automatic generation).

Our results were based on nearly 16,000 images of five activities and we achieved an error rate as low as 1.5%. These experiments clearly demonstrate the importance of powerful methodologies for data modeling and how intertwined feature selection, classifier design and the structure of the classifier are.

1 Introduction

In this paper, we address the problem of recognizing human activities from video sequences, with a particular emphasis on feature selection, data modeling and classifier structure.

Recently, human motion analysis has become one of the most active research areas in computer vision. This is due to promising applications in areas such as visual surveillance, human performance analysis, computer-human interfaces (robotic interaction with humans), content-based image retrieval/storage and virtual reality.

Human movements can be considered at different levels of (temporal) detail: the analysis of the movement of body parts, single person activities and, over increasing temporal windows, large-scale interactions. As a consequence of the application, systems must deal with different activities and temporal integration times.

Most systems that perform human motion analysis address general common tasks, such as: person detection & tracking, activity classification, behavior interpretation and also person identification. Obviously, although some of these tasks can be considered independently, they must be solved in a common framework, where information can be communicated and exchanged between the different system modules.

As the detection and tracking systems have progressed significantly in the past few years, [4, 12, 5], human motion and behavior interpretation have naturally become the following step. In a surveillance scenario, tracking is the very first step and behavior recognition the final goal. The task of activity recognition can be viewed as a bridge between the pixel measurements, given by the tracker, and a more abstract behavior description.

In this paper, we focus on this intermediate level, that is essential to achieve the desired final large-scale interpretation. The need for such systems is increasing everyday with the number of surveillance cameras deployed in public spaces. Needless to say, the “traditional” job of the security operator, monitoring several video streams for extended periods of time, becomes impossible, as the number of cameras grows exponentially. Instead, we need systems able to detect, categorize and recognize human activity, calling for human attention only when necessary.

Generally, human action interpretation can be divided in three [1] major approaches. Generic model recovery tries to fit a 3D model to the person pose [12], and is usually strongly dependent on a accurate 3D feature extraction.

∗This work has been partially supported by EU project CAVIAR(IST-2001-37540) and by the portuguese Fundação para a Ciência e Tecnologia, Programa Operacional Sociedade de Informação (POSI), in the framework of the III Quadro Comunitário de Apoio.
Appearance-based models, are based on the extraction of a 2D shape model, directly from the images, to be classified (or matched) against a trained one [4, 2, 10, 1]. Motion-based models do not rely on static models of the person, but on people motion characteristics [3, 1, 10].

In [5] human actions are represented by stochastic finite automaton of event states, that recognize human interactions like conversing or taking objects away, from trajectory and shape characteristics. Shape analysis and tracking can also be used to construct models of people appearance, ordering the body parts on the silhouette boundary, in order to monitoring their activity [4]. In this case the work is constrained to lateral view of the people.

Bobick and davis work [1] used Motion Energy and Motion History Images (MEI and MHI) to classify “aerobic”-type exercises. In [2], these same images are used determine the shortest video exposure needed for low-latency recognition, using a reliable-inference framework to differentiate between walking, running and standing activities. The work of Rosales [10] also uses the invariant 7-Hu moments of MEI and MHI to estimate Gaussian Mixtures models of Walking, Running, Roller Blading and Biking activities.

Efros et al [3] compute optical flow measurements in a spatio-temporal volume to recognize human activities in a nearest-neighbor framework. In this work, Ballet, Tennis and Football sequences are used to evaluate the proposed approach. In [8] several activities, like Walking, Running, Marching and skipping were recognized, although in a constrained scene, using motion filtered images and space reduction techniques.

The CAVIAR [13] sequences are used in [7] to recognize a the set of activities, scenarios and roles. The approach generates a list of features and automatically chooses the smallest set, that accurately identifies the desired class. However, the results reported use the same sequences (different images) for testing and training. Therefore, it is not clear how the recognition rates would generalize with different sequences.

Although there are many works that can identify interesting activities, many of the applications are scene dependent, i.e. frequently tailored to one specific training scenario. This is the case of [7], because of the use of features like the target position and orientation. Even though this process can model scene depth variations (needing a large set of training examples), it needs to be repeated for each new scenario.

In our work, we try to overcome some of the difficulties discussed before and address three fundamental issues for an activity recognition system: (i) the design of a classifier; (ii) how to perform feature selection and (iii) how to define the structure of a classifier.

We assume that a tracking system computes the subject spatial and temporal grouping (i.e. the person detected pixels over time). We further assume that the tracker measurements are corrected using the image-to-ground plane projective transformation, thus achieving scene (viewport) invariance [9].

As for the design of the classifier, we propose the use of a Bayesian classifier, where the likelihood functions are modeled as Gaussian mixtures, estimated from data. This modeling approach proves to be adequate for the complexity of data distribution we face in this domain.

Regarding the crucial aspect of feature selection, we start with a rich set of 29 features, that describe the subjects global motion, as well as local motion within the region of interest. We propose several (suboptimal) methods for feature selection, to evaluate the recognition rate with the Bayesian classifier. All results are obtained with real images (ground truth available), with different training and test sequences (not just different images, but sequences).

To improve the results, we investigated the use of hierarchical classifiers by grouping the activities to recognize. We show that a hierarchical classifier, with automatic feature selection, achieves an error rate as low as 1.5%. Finally, we briefly address the problem of automatically generating the structure of the hierarchical classifier.

Our experiments clearly show the importance of data modeling methodologies and how intertwined feature selection, classifier design and the structure of the classifier are. The experiments conducted and the encouraging results achieved have been obtained with nearly 16,000 images, with ground truth available.

Section 2 describes the activities to recognize within this work, as well as the large set of features we will analyze and evaluate. Section 3 is devoted to the description of the Bayesian Classifier as well as the feature selection methods. Results with a single layer classifier are presented. Section 4 addresses the problem of hierarchical classification and proposes several classifiers, with an average error rate as low as 1.5%. Finally, in Section 5 we draw some conclusions and establish future directions of work.

2 Low level activities and features

We have concentrated on low-level, short term human activities, as those studied in the context of the CAVIAR project [13]. We will explore the availability of a large data set of video sequences and manually classified activities, in a total of about 16,000 images with ground truth data. The activities (classes) considered can be detected from a relatively short video sequence (a few seconds) and are described in Table 1.

From this basic set of activities, more complex activities can be formed, over larger intervals of time. In those cases, the use of contextual information can help distinguishing activities that would otherwise be ambiguous. Figure 1 shows images acquired in one test scenario corresponding to the referred activities.

We now have to define a number of low-level features, that can be extracted from short video sequences, for activity recognition. Of course, one can only assess the quality of a feature once a classifier (or a metric) is designed for a specific problem. In other words, feature definition and classification are intertwined problems.
<table>
<thead>
<tr>
<th>id</th>
<th># Frames</th>
<th>Activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3,211</td>
<td>Inactive</td>
<td>a static person/object</td>
</tr>
<tr>
<td>2</td>
<td>1,974</td>
<td>Active</td>
<td>person making movements but without translating in the image.</td>
</tr>
<tr>
<td>3</td>
<td>9,831</td>
<td>Walking</td>
<td>there are movements and overall image translation</td>
</tr>
<tr>
<td>4</td>
<td>297</td>
<td>Running</td>
<td>as in walking but with larger translation.</td>
</tr>
<tr>
<td>5</td>
<td>594</td>
<td>Fighting</td>
<td>large quantities of movement with few translation.</td>
</tr>
</tbody>
</table>

Table 1: Low-level activities and data distribution.

Figure 1: Example images and observed activities in one of the CAVIAR test scenarios.

We start by defining a comprehensive set of features that, intuitively, may distinguish the different activities. Then, we assess the quality of those features by analyzing the error rate of our classifiers, against the CAVIAR data-sets.

We assume the existence of a tracking system, able to identify moving blobs in the scene (e.g. though background subtraction, with adaptive background models [4]) and provide information regarding the position of the target over time. When tracking a person across the camera field of view, size, appearance and speed are influenced by perspective. This makes the classification process more difficult, since the observed measurements depend both on the ongoing activity and on the viewing geometry. To achieve viewpoint invariance, all image measurements are back-projected onto the ground plane, using the projective planar transformation between the image and scene ground planes. Figure 2 illustrates the observed (perspectively distorted) image and an orthographic view of the ground plane obtained with the estimated homography.

Figure 2: Original (left) and resulting transformed (right) images of the INRIA - Caviar scenario. Perspective distortion is removed by mapping the ground plane using the estimated homography.

From this point onwards we will assume that the tracker uses the ground-image planes homography, to express the position estimates of the bounding box, directly in the ground floor coordinates. The first sub-set of features have the goal of representing the target global position over time. The second sub-set of features describes the internal motion of the blob.

### 2.1 Features

We consider two reasonably large sets of features, each organized in several subgroups. The first subset of features code the instantaneous position and velocity of the tracked subject. The target velocity, \( \vec{v}(t) \), and speed, \( v(t) \), are obtained through differentiation of the instantaneous position estimate (first sub-group). Since the target inter-frame displacements can be very small, the temporal derivative can be quite noisy. Hence, we introduce different ways of averaging the velocity and speed estimates over an interval of \( T \) frames (second sub-group). The average velocity can be estimated in 4 different processes: (i) simple averaging; (ii) combining velocity estimates over different time intervals; (iii) least squares fit to a constant velocity (linear) model or (iv) least squares estimate of a constant acceleration (quadratic) model.

One interesting potential feature is the ratio between the average speed and the norm of the average velocity. It describes how irregular the motion actually is, approaching 1 if the target always moves in the same direction along a straight line and tending to 0 if it moves irregularly in various directions, or returns to the initial position.

The third sub-group of features aim at coding the temporal energy or activity. It basically consists in taking second order moments of the speed or velocity, either centered or non centered. For the matrices corresponding to the
second order moments of the target velocity we further compute the trace and the ratio of the two eigenvalues. These parameters describe the total energy and the dominant directions with respect to the velocity energy distribution. Table 2 shows the features described. Only those features with a corresponding index in the first column of the table will be evaluated as features in the system. The others correspond to auxiliary computations.

The second subset of features are based on estimates of the optic flow or instantaneous pixel motion inside the bounding box, as described in Table 3. We consider that the motion inside each tracked blob can be as important for activity recognition, if not more, as the overall trajectory of the bounding box.

Again, these features are divided in different sub-groups. After computing the optic flow in the target region, we compute the mean flow. As an option, the mean flow can be centered (subtracted from) with the average blob velocity. In this way, the resulting flow corresponds to the relative motion inside the blob. We also compute the norm of the (spatially) averaged flow.

The next subgroup of features takes second order (spatial) moments of the flow or its norm (centered or non-centered). Also, the ratio of the eigenvalues of the second-order matrix is taken as indicator of the directionality of the overall energy. The following subgroup of features introduces temporal averaging over $T$ frames, namely of the mean flow and the motion energy. Finally, the last subgroup consists in (temporal) second-order moments of the flow and the ratio of the eigenvalues, taken as an indicator of the directionality of the motion. Again, only those rows in the table that have an index in the first column correspond to features to be evaluated. All the others correspond to auxiliary computations.

We have seen that some of the presented feature correspond to the average of instantaneous measurements, taken over $T$ frames. Figure 3 shows the autocorrelation of the instantaneous speed for the {Running} and {Walking} activities. It shows the “periodic” nature of the processed signal and can be used to determine a meaningful value for $T$. Based on this type of curves we have used $T = 25$ frames in most of our experiments.

In the next section we will address the problem of finding out the most promising features - amongst the 29 features described - for the purpose of recognizing the human activities of Active, Inactive, Walking, Running or Fighting.

<table>
<thead>
<tr>
<th>#</th>
<th>NAME</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>position $\bar{v}(t)$</td>
<td>$(X(t),Y(t))$</td>
</tr>
<tr>
<td>2</td>
<td>mean velocity $\bar{v}(t)$</td>
<td>$\bar{v}(t) = \frac{1}{T} \sum_{i=1}^{T} v(i)$</td>
</tr>
<tr>
<td>3</td>
<td>mean flow $\bar{v}(t)$</td>
<td>$\bar{v}(t) = \frac{1}{T} \sum_{i=1}^{T} v(i)$</td>
</tr>
<tr>
<td>4</td>
<td>mean vector $\bar{v}(t)$</td>
<td>$\bar{v}(t) = \frac{1}{T} \sum_{i=1}^{T} v(i)$</td>
</tr>
<tr>
<td>5</td>
<td>trace $\bar{v}(t)$</td>
<td>$\bar{v}(t) = \frac{1}{T} \sum_{i=1}^{T} (\bar{v}(i) - \bar{v}(t))^2$</td>
</tr>
</tbody>
</table>

Table 2: Features derived from the target global position provided by a tracker. The features are organized in 3 groups: (i) instantaneous measurements, (ii) average speed/velocity based features and (iii) second order moments/energy-related indicators.

The second subset of features are based on estimates of the optic flow or instantaneous pixel motion inside the bounding box, as described in Table 3. We consider that the motion inside each tracked blob can be as important for activity recognition, if not more, as the overall trajectory of the bounding box.

Again, these features are divided in different sub-groups. After computing the optic flow in the target region, we compute the mean flow. As an option, the mean flow can be centered (subtracted from) with the average blob velocity. In this way, the resulting flow corresponds to the relative motion inside the blob. We also compute the norm of the (spatially) averaged flow.

The next subgroup of features takes second order (spatial) moments of the flow or its norm (centered or non-centered). Also, the ratio of the eigenvalues of the second-order matrix is taken as indicator of the directionality of the overall energy. The following subgroup of features introduces temporal averaging over $T$ frames, namely of the mean flow and the motion energy. Finally, the last subgroup consists in (temporal) second-order moments of the flow and the ratio of the eigenvalues, taken as an indicator of the directionality of the motion. Again, only those rows in the table that have an index in the first column correspond to features to be evaluated. All the others correspond to auxiliary computations.

We have seen that some of the presented feature correspond to the average of instantaneous measurements, taken over $T$ frames. Figure 3 shows the autocorrelation of the instantaneous speed for the {Running} and {Walking} activities. It shows the “periodic” nature of the processed signal and can be used to determine a meaningful value for $T$. Based on this type of curves we have used $T = 25$ frames in most of our experiments.

In the next section we will address the problem of finding out the most promising features - amongst the 29 features described - for the purpose of recognizing the human activities of Active, Inactive, Walking, Running or Fighting.
To assess the quality of the features, we have designed a Bayesian classifier that we describe hereafter. Let $F(t)$ denote the feature vector observed at time $t$. The size of $F(t)$ depends on the number of the feature components selected from the global set described in Section 2. Given a set of activities $\mathcal{A}_j$, $j = 1..n$, the posterior probability of a certain activity taking place can be computed using Bayes rule as:

$$P(\mathcal{A}_j|F(t)) = \frac{p(F(t)|\mathcal{A}_j)P(\mathcal{A}_j)}{p(F(t))}$$
where \( p(F(t)|\omega_j) \) is the likelihood of activity \( \omega_j \), \( P(\omega_j) \) is the prior probability of the same activity and \( p(F(t)) \) is the probability of observing \( F(t) \), irrespective of the underlying activity.

To build the Bayesian classifier, we must estimate the likelihood function of the features, given each class. The choice of the method to adopt depends on the complexity of the data distribution and dimension of the feature space. As a consequence of the complex feature distribution, we have modeled the data as Gaussian mixtures. This model can approximate arbitrarily complex distributions, when the number of Gaussians increases. The likelihood function is thus approximated by:

\[
p(F(t)|\omega_k) \approx \sum_{j=1}^{N} \pi_j \mathcal{N}(\mu_j, \sigma_j),
\]

where \( \mathcal{N}(\mu_j, \sigma_j) \) denotes a Normal distribution and \( \pi_j \) represents the weight of that Gaussian in the mixture, for each listed activity, \( \omega_k \). To guarantee that \( p(F(t)|\omega_k) \) is a proper probability distribution, we need to have \( \sum_{j=1}^{N} \pi_j = 1 \) and all \( \pi_j \geq 0 \).

The unknown parameters of the mixture, \( (\mu_j, \sigma_j, \pi_j) \) were estimated with the Expectation-Maximization algorithm [11]. We allow the number of Gaussians to vary by eliminating terms when they are too “similar”. In [11], a measure designated by kurtosis is used to measure how close a distribution is to a Gaussian. It is used to split the distribution if it is too different from a Gaussian. Similarly, it can be used as a closeness metric to merge distributions, if this distance becomes too small.

With this procedure, we can estimate the likelihood function without imposing any (global parametric) structure for the underlying distribution. In addition, the number of Gaussians required to model the data is an indication of how complex the data are.

For simplicity, we consider independence between the observations in different time instants. Thus, the inclusion of features extracted in a temporal window, \( T_c \), is modeled by the product of the probabilities. The (naive) Bayes rule becomes:

\[
P(\omega_j|F(t), F(t-1), ..., F(t-T_c)) = \prod_{i=0}^{T_c-1} p(F(t-i)|\omega_j) p(\omega_j) \frac{P(F(t))}{p(F(t))}.
\]

Having estimated the likelihood functions, and integrated the measurements over time, the Maximum a Posteriori (MAP) estimate of the observed activity is given by:

\[
\hat{\omega}_{MAP}(t) = \arg \max_{\omega_j} P(\omega_j|F(t-T_c, ..., t))
\]

Figure 4 shows the phase-space of two features for one given activity, illustrating the complexity of data distribution and the need for powerful modeling techniques. As desired, the mixture of Gaussians seems to be able to model the likelihood function.

![Figure 4: Phase plot of features 9 and 21. The ellipses represent the Gaussian modes of the mixture, estimated using EM. Blue dots correspond to the training set and red dots to the test set.](image)

### 3.2 Selecting promising features

We have proposed a Bayesian classifier that takes a feature vector as the input (actually, a collection of feature vectors over time) and generates a decision, regarding the observed activity.
We now have to decide upon which features to use, from the set described in Section 2. The reason for selecting a subset of those features, instead of using them all, has to do with the ability to model the likelihood function. The higher the dimension of the feature space, the more difficult it becomes to model the data distribution. In addition, learning feature distribution in a high dimensional space requires large amounts of data, which are not always available.

We evaluate the performance of the feature set using the proposed classifier. To be exhaustive, we would need to consider all possible combinations of 1 feature, 2-features, 3-features, etc. For M features (29, in our case), this would require running the classifier (and learning the likelihood functions) a large number of times:

$$\# \text{trials} = C_M^1 + C_M^2 + C_M^3 + \ldots + C_M^M$$

where $C_M^n$ denotes the number of combinations of $i$ features chosen amongst the possible total number of $M$ features. Needless to say, this number becomes extremely large, even for a few tens of features. Here, we follow several suboptimal strategies, to select a given number of features from the original set.

The first Brute-Search approach is the most straightforward and computationally expensive. The best $N_f$ features are obtained by trying all possible combinations of $N_f$ features, requiring the evaluation of $C_{N_f}^M$ feature-tuples.

The second method, that we call Lite-Search, coincides with the previous method, if we just look for the best feature. If two features are needed, we then use the best individual feature and search for the feature, amongst all others, to form the best possible pair with the first one. It would then require evaluating $M$ features individually, plus $M - 1$ additional runs to find the "best" pair. For three features, we use the best pair and search amongst the remaining $M - 2$ features for the best triplet, with a cost of $M + (M - 1) + (M - 2)$ runs.

The third method, called the Lite-lite Search, aims to reduce the computational cost even further. We first rank the features individually. If we want to use two features, we rely on the best two (isolated) features and run the modeling step once more to learn the likelihood functions. Table 4 summarizes the cost of these different methods, providing numeric examples for our case of $M = 29$.

<table>
<thead>
<tr>
<th>$N_f$</th>
<th>Brute Search</th>
<th>Lite Search</th>
<th>Lite-lite Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\frac{C^M}{N_f}$</td>
<td>$M + (M - 1) + \ldots + (N_f \text{ terms})$</td>
<td>$M + 1$</td>
</tr>
<tr>
<td>2</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>3</td>
<td>3,654</td>
<td>57</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 4: Comparison of the search complexity, in terms of the number of runs of the classifier and modeling the likelihood function, of the used feature search methods, for our case with $M = 29$ features.

The final method the well know Relief algorithm [6] that creates a weight vector over all features to quantify their "quality". This vector is updated for each of the data points presented, according to:

$$w_i = w_i + (x_i - \text{nearmiss}(x_i))^2 - (x_i - \text{nearhit}(x_i))^2,$$

where $w_i$ represents the weight vector, $x_i$ the $i^{th}$ feature for data point $x$, nearmiss($x$) and nearhit($x$) denote the nearest point to $x$ from the same and different class, respectively. The Relief algorithm gives larger weights to features that result in compact class clusters, while keeping the largest possible distance to the nearest class. Possible problems may arise when the data distribution does not lead to the formation of compact clusters. When all classes are "close" to each other, evaluating only the distance to the nearest one does not necessarily guarantee quality of the feature.

In spite of this limitation, the Relief algorithm is very efficient. On a Pentium IV machine, 2.8MHz, 512Mb Ram, Relief takes about one minute of computation time, in a non-optimized MatLab implementation. Instead, the Brute-Search method takes about 24 hours to find the best triplet of features. Table 5 show the results obtained using these different feature search criteria and for 1, 2 or 3 features.

<table>
<thead>
<tr>
<th># Features</th>
<th>Brute Search</th>
<th>Lite Search</th>
<th>Lite-lite Search</th>
<th>RELIEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>83.9%</td>
<td>7</td>
<td>83.9%</td>
</tr>
<tr>
<td>2</td>
<td>9.18</td>
<td>93.5%</td>
<td>7.25</td>
<td>89.8%</td>
</tr>
<tr>
<td>3</td>
<td>3.920</td>
<td>94%</td>
<td>7.19</td>
<td>92.1%</td>
</tr>
</tbody>
</table>

Table 5: Comparison of the recognition rate with different methods for feature selection.

From this table, we can see that the best results are obtained with the "Brute-Search" approach, as expected. With one feature only, we achieve a classification error of 16.1%, while using three features causes the error to drop down to 6%. The performance of the "Lite-Search" method is slightly inferior at a much lower search cost. Finally, the "Lite-lite-Search" does not lead to competitive results. Similarly, the performance of the RELIEF algorithm is clearly
Table 6: Confusion matrix for the five activities and using the “best” feature triplet: $F_1 = 9$, $F_2 = 15$, $F_3 = 22$. The rightmost column indicates the total number of processed bounding boxes (targets).

behind some of other methods, probably due to the poor modeling of complex data distributions. Finally, Table 6 shows the confusion matrix corresponding to the best classifier using the best triplet of features.

We have proposed a Bayesian classifier for human activity recognition where the likelihood function was modeled as a mixture of Gaussians and estimated using the EM algorithm. Then, we have used this classifier to evaluate different feature sub-sets, using four distinct methods with different computational cost. The use of three features allowed us to recognize the observed activities with an error as low as 6%. In the following section we will see how to further reduce the error rate by using alternative architectures for the classifier.

4 Classifier Structure

We have seen the impact of feature selection on the recognition rates and on the complexity of data modeling. To improve the classification rate even further, we will change the structure of the classifier. One of the limitations of the previous classifier was that all classes had to be considered simultaneously. If a certain feature is good for two classes and another feature proves successful for two other classes, both of them will be included in the selection set. As a consequence, we will have to model the joint likelihood of those features, as opposed to modeling them separately.

Here, we will follow a different approach. We will group activities in subsets and perform classification in a hierarchical manner. Each classifier will have to deal with a smaller number of classes. In addition, we can use different features for different classifiers/classes, instead of having the same feature set for all classes.

Figure 5 shows one binary hierarchical classifier. The problem of recognizing the 5 different human activities is broken down into 4 distinct classification problems, each producing a binary decision.

Figure 5: Binary tree classification with error percentage and best features selected for each classifier. The classification error is as low as 1.5%!

The first classifier distinguishes between two meta-classes: \{Active, Inactive\} versus \{Walking, Running, Fighting\}, using a set of features selected with the procedures described in the previous section. Then, other classifiers will try to distinguish \{Active\} versus \{Inactive\}, and so forth. We now have the flexibility to specialize the features for each subset of classes and to use different number of features for different layers of the hierarchical classifier.

Table 7 shows the recognition rate obtained with this hierarchical classifier using one, two or three features selected as described previously. The 6% error rate of the single layer classifier was reduced to 1.5%, using three features in all classifiers except for classifier 2 (see Figure 5).
Table 7: Comparison of recognition rates obtained for the hierarchical classifier using different methods for feature selection. Error rates are presented for each individual classifier and the overall score for the hierarchical classifier.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>18</td>
<td>97.6%</td>
<td>18</td>
<td>97.6%</td>
<td>18</td>
<td>97.6%</td>
<td>18</td>
<td>97.6%</td>
<td>18</td>
<td>97.6%</td>
<td>14</td>
<td>92.1%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>17</td>
<td>95.5%</td>
<td>17</td>
<td>95.5%</td>
<td>17</td>
<td>95.5%</td>
<td>17</td>
<td>95.5%</td>
<td>17</td>
<td>95.5%</td>
<td>23</td>
<td>86.7%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
<td>94%</td>
<td>5</td>
<td>94%</td>
<td>5</td>
<td>94%</td>
<td>5</td>
<td>94%</td>
<td>7</td>
<td>98.1%</td>
<td>11</td>
<td>85.5%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>7</td>
<td>98.1%</td>
<td>7</td>
<td>98.1%</td>
<td>7</td>
<td>98.1%</td>
<td>7</td>
<td>98.1%</td>
<td>11</td>
<td>85.5%</td>
<td>23</td>
<td>86.7%</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>17</td>
<td>97.3%</td>
<td>17</td>
<td>97.3%</td>
<td>17</td>
<td>97.3%</td>
<td>17</td>
<td>97.3%</td>
<td>17</td>
<td>97.3%</td>
<td>14</td>
<td>92.1%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>17</td>
<td>99.3%</td>
<td>17</td>
<td>99.3%</td>
<td>17</td>
<td>99.3%</td>
<td>17</td>
<td>99.3%</td>
<td>17</td>
<td>99.3%</td>
<td>14</td>
<td>92.1%</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>61</td>
<td>98.8%</td>
<td>61</td>
<td>98.8%</td>
<td>61</td>
<td>98.8%</td>
<td>61</td>
<td>98.8%</td>
<td>61</td>
<td>98.8%</td>
<td>14</td>
<td>92.1%</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>72</td>
<td>99.6%</td>
<td>72</td>
<td>99.6%</td>
<td>72</td>
<td>99.6%</td>
<td>72</td>
<td>99.6%</td>
<td>72</td>
<td>99.6%</td>
<td>14</td>
<td>92.1%</td>
</tr>
</tbody>
</table>

4.1 Discussion

The hierarchical classification strategy led to significant improvements over our previous results. However, the structure of the classifier, i.e. how the different classes were arranged in meta-classes, was done manually, based on our intuition as to how they should be combined together. Even if this type of domain knowledge may always prove useful when designing classification systems, it would be interesting to learn the structure of the classifier automatically, similarly to what we have done with the feature selection.

The straightforward solution would be considering all possible arrangements of classes and running the feature selection process, for each instance, but the computational cost would be prohibitive. Let us assume that we have 5 classes and consider all possible graphs similar to that in Figure 5. For the top level, we can arrange the activities in 1 + 4 or 2 + 3 subsets. This corresponds to \( C_5^1 + C_5^2 \) solutions. For the first case, the subgroup of 4 activities would need to further re-grouped in 1 + 3 or 2 + 2 subsets. In the 1 + 3 case, we would still need to consider all possible combinations of \( C_3^1 \), and so on. In all, the number of possible combinations can be shown to be:

\[
total = C_5^1 \times (C_4^1 \times C_1^1 + C_2^2) + C_5^2 \times C_1^1 = 120
\]

For each of these 120 possible hierarchical classifiers we would need to run the feature selection process. For 3 features and using the most performing (Brute-Search) method, this would imply 120 \( \times \) 3,654 = 438,480 runs.

Figure 6 shows an automatically generated binary tree, using a sub-optimal search method. The first level contains the class that is best separable from all others activities, yielding the lowest recognition error with the two best features. After selecting the “first” class, we continue the process to find which activity is now easier to separate and so forth. The process is repeated until the lower level (no more classes to un-group) is reached.

![Automatically generated hierarchical classification tree and features, with 2.6% of average recognition error.](image)

The results obtained with automatic generation of the classification tree and feature selection are quite encouraging. In particular, the recognition rate is better than the one obtained with the single-level classifier.
5 Conclusions

We addressed the problem of recognizing short-term human activities (Active, Inactive, Walking, Running, Fighting) from video sequences, with a emphasis on feature selection, data modeling and classifier structure.

The ability to recognize human activities responds to the need for systems able to process the ever increasing number of surveillance cameras deployed in public spaces. Such systems should be able to detect, categorize and recognize human activities, drawing the human attention only when necessary.

We focused on three fundamental issues: (i) the design of a classifier for activity recognition; (ii) how to perform feature selection and (iii) how to define the structure of a classifier.

We propose the use of a Bayesian classifier where the likelihood functions are modeled as Gaussian mixtures and learned automatically. We have shown how this approach succeeds in modeling complex data distributions. Regarding the problem of feature selection, we proposed a large number of features describing both the tracked subject global and internal motion. Then, we resort to several (suboptimal) methods to evaluate different combinations of features. The evaluation is based on the the recognition rate achieved with the classifier. Finally, we investigate the use of hierarchical classifiers (including the possibility of automatic generation). Our results were based on the use of nearly 16,000 images of five activities and we achieved an error rate as low as 1.5%.

Our experiments clearly demonstrate the importance of using powerful methodologies for data modeling. It also shows how intertwined feature selection, classifier design and the structure of the classifier are and how determinant this can be for the overall performance.

Future work will address the use of this low-level (short-term) activity recognition system for higher level actions extending over large periods of time and possibly replying on contextual information, as well as analyzing the possibility of dynamically enlarging the activity set.

References


Activity Recognition via Autoregressive Prediction of Velocity Distribution

Miha Peternel and Aleš Leonardis
University of Ljubljana
Faculty of Computer and Information Science
Tržaška 25, SI-1001 Ljubljana, Slovenia
{miha.peternel, ales.leonardis}@fri.uni-lj.si

Abstract

We present a novel approach for view-based learning and recognition of motion patterns of articulated objects. We formulate the intervals of motion as a predictive model of local spatio-temporal receptive field activation. We compute local velocity distribution using a Bayesian approach, and then approximate the local velocity distribution in space and time using a set of Gaussian receptive fields. The activation sequence of receptive fields over time is modeled in a PCA subspace using linear auto-regression to arrive at a model of the motion pattern. Recognition is performed using the MDL principle. We test the approach on a number of human motion patterns to demonstrate the applicability of the proposed approach to simple action recognition and identification.

1 Introduction

The ability to learn and later recognize articulated activities with few assumptions about the object geometry, appearance, and the nature of the activity would have a number of applications in monitoring, video interpretation, video indexing, smart environments, and human-robot interactions.

Several models have been developed to enable recognition of motion patterns, trajectories, and consequently activities [7]. Standard methods either use a predefined geometrical model and try to estimate its parameters [8, 1, 2, 11], or attempt to model the observed motion directly [3, 16, 4, 5]. Recently, attempts have been made to summarize motion using local spatio-temporal features [14, 12]. It has been shown [12] that distribution density of local trajectories can be used for learning and recognition of cyclic human motion, however such representation is scale dependent and applies to cyclic motion only. Fablet et al. [6] have used causal probabilistic models to represent video dynamics. Jebara and Pentland [11] have used Gaussian probabilistic models to predict reaction from an interval of action. Agarwal and Triggs [1] have demonstrated that a mixture of regression models can be used as a predictor with a geometrical model. Bissacco et al. [2] have used subspace angles between autoregressive models of skeletal angles to recognize gait.

There have been few attempts of motion-based learning of activities without assuming a specific geometric model. Most of these approaches are based on modeling of optical flow fields [3, 16, 5], motion history [4] or on the modeling of manifolds of local features [14, 12]. We propose that instead of modeling the motion directly, prediction of local motion features can be exploited as a cue for activity recognition.

In this paper we present a novel view-based model of articulated motion based on the premise that we can recognize a motion pattern if we can predict it. On top of a robust local velocity estimate we build an auto-regressive predictor of the change in the future velocity distribution. We demonstrate that the proposed model can be used to learn and recognize individual patterns of locomotion as well as short non-cyclic actions.

The key novelties of our approach compared to prior works are: we model motion patterns based on the changes in their velocity distribution instead of using estimated optical flow; we apply a predictive model of a large number of local receptive fields based on co-activation in their spatio-temporal neighborhood to enable approximate modeling of arbitrary geometry in motion; we predict local co-activation patterns in a PCA subspace to decrease the dimensionality of the predictive function and thus enable learning from short action sequences; and finally, we recognize the motion patterns based on the utility of the predictor as measured by the MDL criterion. The approach is summarized in Figure 1.

The rest of the paper is organized as follows. In the next section we introduce Bayesian estimate of local velocity distribution. In Section 3 we describe how the velocity distribution can be approximated by Gaussian receptive fields. In Section 4 we outline autoregressive learning of receptive field activation patterns. In Section 5 we detail the proposed activity recognition method. In Section 6 we present experimental results. In Section 7 we conclude with a summary and outline work in progress.
2 Bayesian estimate of local velocity distribution

A complex motion pattern will undoubtedly appear as local velocity in a point of space-time. We compute a Bayesian estimate of local velocity distribution in a point of the view space based on a model proposed by Weiss and Fleet [15]:

\[
P(v|I(x,t)) = \alpha P(v) * P(I(x,t)|v),
\]

where \(v\) is velocity, \(x\) is view space position, \(t\) is time, \(I\) is image patch, and \(\alpha\) is an undetermined constant. The prior velocity likelihood \(P(v)\) is assumed to be Gaussian. Such a model results in a number of properties that are similar with how people perceive local motion [15], moreover we treat the aperture problem by estimating the ambiguity. A useful estimate of image likelihood \(P(I(x,t)|v)\) can be computed by expressing the probability as a function of the energy between two image patches \(I(x,t)\) and \(I(x - v, t - 1)\) given a velocity estimate \(v\). We will compute the energy as sum of square differences between the luminance values:

\[
\text{SSD}(v) = \sum_{x \in W} (I(x,t) - I(x - v, t - 1))^2,
\]

where \(x\) runs over the image window \(W\) for which we estimate the probability. Assuming Gaussian noise with variance \(\sigma^2\), we can then estimate the local image likelihood as:

\[
P(I(x,t)|v) = \alpha \exp\left(-\frac{\text{SSD}(v)}{4\sigma^2}\right).
\]

We quantize and constrain the velocity space to be in the expected range, and normalize the distribution by choosing \(\sigma\), so that \(\sum_{v \in \mathcal{V}} P(v|I(x,t)) = 1\). The observed distribution space \(\mathcal{D}\) is a compositum of the view space \(\mathcal{X}\) and the local velocity space \(\mathcal{V}\) over time, where each point \((x, t)\) in space-time contributes an equal share of probability \(P(x, v | t)\) dispersed over the local velocity parameter \(v\). See Figure 2(a) for an illustration. We have to compute (1) for each \((x, v) \in \mathcal{D}\).

3 Approximating probability distribution using local receptive fields

Let \(\Phi_t\) represent the velocity probability distribution over \(\mathcal{D}\) at time \(t\). Inferring \(\Phi_t\) from \(\Phi_{t-1}, \Phi_{t-2}, \ldots\) is statistically an ill-posed problem in general due to a large number of parameters that have to be learned from short intervals. We propose \(\Phi_t\) be approximated using a large number of Gaussian bases \(\mathcal{N}_t\), such that \(\sum_{i=1}^{K} k_i \mathcal{N}_i \approx \Phi_t\). The evolution of coefficients \(k_i\) of the bases can then be modeled statistically using multivariate sequence prediction methods. In order to generalize the approximation we apply a four-dimensional grid of bases

\[
\mathcal{N}_t(x,v; \mu_x, \mu_v) = \frac{1}{4\pi^2 \sigma_x^2 \sigma_v^2} \exp\left(-\frac{1}{2} \left(\frac{(x - \mu_x)^2}{\sigma_x^2} + \frac{(v - \mu_v)^2}{\sigma_v^2}\right)\right)
\]
placed equidistantly in $X$ and $Y$ (see Figure 2(b) for an illustration). $\Phi_t$ is approximated by a vector $\phi_t$, where

$$\Phi_t \approx \beta \sum_{i=1}^{K} \phi_{t,i} \mathcal{N}_i, \quad \phi_{t,i} = \Phi_t * \mathcal{N}_i.$$  

(5)

Kernels $\mathcal{N}_i$ thus correspond to local receptive fields in the velocity distribution, each tuned for a specific velocity and a specific view space position. To observe changes in the distribution, an exact decomposition is not strictly necessary, however, the kernels are spaced apart and dimensioned for a small overlap in order to make them more orthogonal which approximately preserves (5) to a constant factor $\beta$.

4 Learning spatio-temporal receptive field activation patterns

An articulated motion sequence can be represented by approximating the distributions through all the frames: $\Phi_t \rightarrow \phi_t$. Thus we can model a video sequence by a multivariate sequence of vectors $\phi_1, \phi_2, ... \phi_T$, each vector representing the activation of Gaussian receptive fields in each frame. Alternatively we can represent an activity by approximating the change in the distribution by defining $\delta_t = \phi_t - \phi_{t-1}$. Such a sequence can be predicted by assuming a linear autoregressive model of order $\tau$:

$$\hat{\delta}_t = \sum_{i=1}^{\tau} A_i \delta_{t-i} + b.$$  

(6)

Parameters $A_i$ and $b$ are chosen to minimize the error between $\delta_t$ and $\hat{\delta}_t$. We use ARfit [13] algorithm to compute the parameters. Figure 3 illustrates autoregressive prediction of 30 active receptive fields in a locomotion sequence.

Figure 3: Measured receptive field activation sequence, autoregressive prediction and prediction error of 30 active receptive fields in a locomotion sequence.

If the number of receptive fields is large, it may be beneficial to compute a PCA subspace of $\delta_t$ to constrain the dimensionality of the predictive model as well as to constrain the prediction to the active subset of the receptive fields and exploit their co-activation statistics. Given a matrix of principal eigenbases $U$, we predict the subspace sequence $U \delta_t$ and compute the prediction:

$$\hat{\delta}_t = \sum_{i=1}^{\tau} U^T A_i U \delta_{t-i} + U^T b.$$  

(7)

PCA bases define complex receptive fields (i.e. a linear combination of Gaussian receptive fields) which together predict local receptive field activation.

5 Recognition as prediction vs. measurement

Given an input video sequence (or an interval from the sequence) and a set of models $\mathcal{M}_c = \{A_1^c, A_2^c, b^c, U^c\}$ we apply the Minimum Description Length [10] principle to choose the model in the following way. First, we compute the local velocity probability distribution in each frame. Secondly, the probability distribution in each timepoint is decomposed using a set of kernels corresponding to $\mathcal{N}_i$ to produce a sequence of activation measurements $\delta_1, ... \delta_T$. We define the prediction error sequence as follows:

$$\varepsilon_t^c = \delta_t - \hat{\delta}_t = \delta_t - U^T b^c - \sum_{i=1}^{\tau} U^T A_i^c U^c \delta_{t-i}.$$  

(8)
We can formulate maximum a posteriori model selection probabilistically:

\[ \arg \max_c P(M_c | \delta) = \frac{P(\delta | M_c)P(M_c)}{P(\delta)} . \]  \hspace{1cm} (9)

Assuming \( P(\delta) \) is the same for all models, and noting that \( P(\delta | M_c) = P(\epsilon^c) \), (9) simplifies to:

\[ \arg \max_c P(\epsilon^c)P(M_c) = \arg \max_c \log P(\epsilon^c) + \log P(M_c) . \] \hspace{1cm} (10)

Let \( \mathcal{L}(\delta) \) represent the description length of the measurement and \( \mathcal{L}(\epsilon | M_c) \) the description length of the residual \( \epsilon^c \) given a model \( M_c \) of length \( \mathcal{L}(M_c) \). We relate \( \mathcal{L} \) to probability using the Shannon information measure: \( \mathcal{L}(d) = - \log_2 P(d) \). Note that optimizing (10) equals optimizing:

\[ \arg \min_c \mathcal{L}(\epsilon | M_c) + \mathcal{L}(M_c) . \] \hspace{1cm} (11)

We can choose the most efficient model to explain the data by minimizing MDL based objective function:

\[ \arg \min_c \frac{\mathcal{L}(\epsilon | M_c) + \mathcal{L}(M_c)}{\mathcal{L}(\delta)} . \] \hspace{1cm} (12)

Note that \( \mathcal{L}(\delta) \) is independent of \( c \), we reintroduce it in (12) only to be able to interpret the results as relative bit savings. By assuming that the sequence vectors are conditionally independent, we can compute:

\[ \mathcal{L}(\epsilon | M_c) = \sum_{t=\tau+1}^{T} - \log_2 P(\epsilon^c_t) , \quad \mathcal{L}(\delta) = \sum_{t=\tau+1}^{T} - \log_2 P(\delta_t) . \] \hspace{1cm} (13)

Under the assumption that both the measurement and prediction error sequences are zero mean and normally distributed, and all model descriptions are equal in length Eq. (12) further simplifies to:

\[ \arg \min_c \frac{\sum_{t=\tau+1}^{T} (\epsilon^c_t)^2}{\sum_{t=\tau+1}^{T} (\delta_t)^2} . \] \hspace{1cm} (14)

Due to high dimensionality of both \( \delta_t \) and \( \epsilon^c_t \) it is not feasible to model their distribution directly. Instead, we observe and model the distribution of their magnitude \( |\delta_t| \) and \( |\epsilon^c_t| \) by computing their histograms from the test sequences. The resulting probabilities are used as priors for \( P(\delta_t) \) and \( P(\epsilon^c_t) \).

Figure 4: Distributions of activation vector magnitude and error vector magnitudes compared to Gaussian and Chi-square approximation.

Figure 4 presents actual distributions (as measured on 300 short action sequences) of activation vector magnitude \( |\delta_t| \), error vector magnitudes \( |\epsilon^c_t| \) using different numbers of PCA bases, a Gaussian approximation, and a Chi-square approximation. Note that the measured distributions are remarkably similar when the number of PCA bases is small. As the number of the bases increases the distribution flattens and the tail extends further right. The first (largest) mode is slightly offset from zero due to measurement noise, and the modes further right are likely the contributions of errors from the parts of sequences that are not accurately predicted by linear autoregression. Considering the nature of the valleys around the largest mode, a unimodal Gaussian approximation seems reasonable.
6 Experiments

We perform all the tests on MJPEG sequences downscaled 1:2 and the velocity estimated at every 4×4 pixels using an 8×8 patch and prior velocity likelihood distribution assuming standard deviation of 1.5 pixels. We use $K = 1024$ receptive fields in an 8×8 configuration with a distance of 28 pixels and standard deviation 8 in the view space $\mathcal{D}$ and a 4×4 configuration with a distance of 4 pixels/frame and standard deviation 1 in the velocity space $\mathcal{V}$. We use AR models of order $\tau = 3$.

6.1 Locomotion based identification

We perform two experiments on the CMU MoBo database [9]. We use 25 sequences of people walking on the treadmill. We divide each sequence in a learning half and a testing half. We learn 25 models of locomotion. In the first experiment, we classify the testing sequences using the proposed approach. The results are in Figure 5. As the number of PCA bases increases, the recognition performance initially increases towards 100%. The average of objective function Eq. (12) reaches its minimum at 14 PCA bases, where the average description length of the residuals is minimal. Note that the recognition performance around this point is 100%. With more bases, the models start to overfit, and the recognition performance finally deteriorates at more than 31 PCA bases. The identification performance is comparable to prior work by Peternel and Leonardis [12], however our model requires neither prolonged local tracking nor extraction of walking cycles.

In the second experiment, we vary the position and scale of the receptive field structure, to test the local monotonicity of the recognition function. We notice a clear extreme in $\mathcal{D}$ for all cases, however changes in scale introduce slight fluctuations.

![Figure 5: Locomotion pattern identification results on 25 fast-walk sequences.](image)

6.2 Stand-sit disambiguation

For the third experiment we recorded a sequence of 5 people standing up and sitting down on the same chair 4-5 times. We learn models of all the instances of both actions. We perform leave-one-out classification to test the categorization and identification with two simple actions. The results are in Figure 6. The method correctly categorizes all the sequences when the number of PCA bases used is lower than 12.

![Figure 6: Categorization and identification on 48 stand-up & sit-down sequences of 5 people (recognition curves overlap).](image)
Figure 7: Actions performed by manipulating a box of CDs.

<table>
<thead>
<tr>
<th>Method description</th>
<th>Error Likelihood Prior</th>
<th>Action Recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Leave-one-instance-out</td>
<td>Gaussian</td>
<td>285 95.0%</td>
</tr>
<tr>
<td>Leave-one-person-out</td>
<td>Gaussian</td>
<td>271 90.3%</td>
</tr>
<tr>
<td>Leave-one-instance-out</td>
<td>Chi-square</td>
<td>280 93.3%</td>
</tr>
<tr>
<td>Leave-one-person-out</td>
<td>Chi-square</td>
<td>264 88.0%</td>
</tr>
<tr>
<td>Leave-one-instance-out</td>
<td>average histogram</td>
<td>263 87.7%</td>
</tr>
<tr>
<td>Leave-one-person-out</td>
<td>average histogram</td>
<td>246 82.0%</td>
</tr>
</tbody>
</table>

Table 1: Summary of action categorization on 300 short action sequences of 6 people.

<table>
<thead>
<tr>
<th>ID</th>
<th>Action</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Recog. Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pick from right</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.0%</td>
</tr>
<tr>
<td>2</td>
<td>put from right</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93.3%</td>
</tr>
<tr>
<td>3</td>
<td>pick from left</td>
<td>1</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96.7%</td>
</tr>
<tr>
<td>4</td>
<td>put from left</td>
<td>0</td>
<td>1</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96.7%</td>
</tr>
<tr>
<td>5</td>
<td>flip right</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>93.3%</td>
</tr>
<tr>
<td>6</td>
<td>flip left</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>96.7%</td>
</tr>
<tr>
<td>7</td>
<td>move forward</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96.7%</td>
</tr>
<tr>
<td>8</td>
<td>move backwards</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>93.3%</td>
</tr>
<tr>
<td>9</td>
<td>move right</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>93.3%</td>
</tr>
<tr>
<td>10</td>
<td>move left</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>90.0%</td>
</tr>
</tbody>
</table>

Table 2: Confusion matrix for categorization of 10 actions from 300 sequences of 6 people (Leave-one-instance-out).

<table>
<thead>
<tr>
<th>ID</th>
<th>Action</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Recog. Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pick from right</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100.0%</td>
</tr>
<tr>
<td>2</td>
<td>put from right</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>90.0%</td>
</tr>
<tr>
<td>3</td>
<td>pick from left</td>
<td>2</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>93.3%</td>
</tr>
<tr>
<td>4</td>
<td>put from left</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>86.7%</td>
</tr>
<tr>
<td>5</td>
<td>flip right</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>93.3%</td>
</tr>
<tr>
<td>6</td>
<td>flip left</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>96.7%</td>
</tr>
<tr>
<td>7</td>
<td>move forward</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>93.3%</td>
</tr>
<tr>
<td>8</td>
<td>move backwards</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>24</td>
<td>5</td>
<td>0</td>
<td>80.0%</td>
</tr>
<tr>
<td>9</td>
<td>move right</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>90.0%</td>
</tr>
<tr>
<td>10</td>
<td>move left</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>23</td>
<td>76.7%</td>
</tr>
</tbody>
</table>

Table 3: Confusion matrix for categorization of 10 actions from 300 sequences of 6 people (Leave-one-person-out).

6.3 Manipulative action categorization

For the last set of experiments we recorded over 300 sequences of 10 simple actions performed by 6 people with a box of CDs (see Figure 7 for an illustration). The volunteers were instructed to perform each action at least 5 times in a natural way, and they did not see how the others performed the same actions. The approaches varied as did the exact locus of each action. Moreover, the volunteers were instructed to ignore the placement markers rather than to change their natural moves. The field of view contained a flat table area where the actions were performed, a box object, a pad and mostly one or two arms of the person performing the action. The background was only partially textured to introduce different levels of ambiguity in the velocity estimation.

We evaluate the performance of the proposed approach for the problem of learning and categorization of 10 short actions from 300 short video sequences. We trimmed each sequence to keep five activity-free frames before and after each action. We learn autoregressive models of order 3 in a PCA subspace dimensioned from 2–8. We use objective function Eq. (12) to select among models with different number of PCA bases.

We evaluate the action categorization in two ways, both with three error likelihood priors: Gaussian, Chi-square
and data-driven histogram based. We summarize the results in Table 1.

In the first experiment we categorize an instance of an action by ignoring its model while preserving the other instances of actions from the same person. The results for the Gaussian prior are in Table 2. To summarize the results, exactly 285 out of 300 video sequences are categorized correctly. In the second experiment we categorize an instance of an action by ignoring all the instances of all actions from the same person. The results for the Gaussian prior are in Table 3. Still, 271 of 300 sequences are categorized correctly, showing that the method is able to generalize actions from the exemplars of other people.

We repeat the experiments with the Chi-square prior and with the histogram based prior computed by accumulating distribution histograms over all available sequences. We already presented the distributions in Figure 4. Despite a more accurate modeling of error distribution by a histogram, recognition results are not improved, meaning that the average prior is not particularly useful for recognition of different activities. If the number of instances of each action were larger it might be possible to provide enough statistics for activity specific modeling of distribution.

7 Summary and Conclusions

We proposed a model for view-based learning and recognition of articulated motion sequences without specific assumptions about the geometry or cyclicity, and no requirement for local tracking.

The main contributions of this paper are a novel representation of articulated motion and the associated methods for learning and recognition of activity patterns from short video sequences, and a new database of short manipulative actions.

We have applied the proposed approach to learning and recognition of several types of articulated motion patterns: cyclic and non-cyclic motion of humans, as well as short actions performed by manipulating a box.

We demonstrated that even a rough linear prediction of the change in velocity distribution can be used with a MDL criterion to identify individual locomotion patterns and categorize simple actions. Experiments show that different levels of prediction accuracy are required for the problems of identification and categorization. Sufficiently accurate modeling of co-activation subspace is required for individual cyclic locomotion pattern identification, however weaker predictors attained with smaller dimensional subspaces are most useful for action categorization, whereas higher dimensional subspaces only help at disambiguation of some slightly more complex actions.

Current research is directed towards extending the method to activity localization in space and time, and other remaining problems in articulated motion recognition.

References


Segmentation and Classification of Human Activities∗

J.C. Nascimento1  M. A. T. Figueiredo2  J. S. Marques3
jan@isr.ist.utl.pt  mtf@lx.it.pt  jsm@isr.ist.utl.pt
1,3Instituto de Sistemas e Robótica  2Instituto de Telecomunicações
Instituto Superior Técnico
1049-001 Lisboa
PORTUGAL

Abstract

This paper describes an algorithm for segmenting and classifying human activities from video sequences of a shopping center. These activities comprise entering or exiting a shop, passing, or browsing in front of shop windows. The proposed approach recognizes these activities by using a priori knowledge of the layout of the shopping view. Human actions are represented by a bank of switch dynamical models, each tailored to describe a specific motion regime. Experimental tests illustrate the effectiveness of the proposed approach with synthetic and real data.

Keywords: Surveillance, Segmentation, Classification, Human Activities, Minimum Description Length.

1 Introduction

The analysis of human activities is an important computer vision research topic with applications in surveillance, e.g. in developing automated security applications. In this paper, we focus on recognizing human activities in a shopping center.

In commercial spaces, it is common to have many surveillance cameras. The monitor room is usually equipped with a large set of monitors which are used by a human operator to watch over the areas observed by the cameras. This requires a considerable effort of the human operator, who has to somehow multiplex his/her attention. In recent years a considerable effort was devoted to develop automatic surveillance systems providing information about which activities take place in a given space. With such a system, it would be possible to monitor the actions of individuals, determining its nature and discerning common activities from inappropriate behavior (for example, standing for a large period of time at the entrance of a shop, fighting).

In this paper, we aim at labelling common activities taking place in the shopping space. Activities are recognized from motion patterns associated to each person tracked by the system. Motion is described by a sequence of displacements of the 2D centroid (mean position) of each person’s blob. The trajectory is modelled by using multiple dynamical models with a switching mechanism. Since the trajectory is described by its appearance, we compute the statistics for the identification of the dynamical models involved in a trajectory.

The rest of the paper is organized as follows. Section 2 deals with related work. Section 3, describes the statistical activity model. Section 4 derives the segmentation algorithm. Section 5 reports experimental results with synthetic data and real video sequences. Section 6 concludes the paper.

2 Related Work

The analysis of human activities has been extensively addressed in several ways using different types of features and inference methods. Typically, a set of motion features is extracted from the video signal and an inference model is used to classify it into one of \( c \) possible classes.

For example in [16] the human body is approximated by a set of segments and atomic activities are then defined as vectors of temporal measurements which capture the evolution of the five body parts. In other works the human body is simply represented by the mass center of its active region (blob) in the image plane [12] or the body blob as in [4]. The activity is then represented by the trajectory obtained from the blob center, or from the correspondence of body blob regions respectively.

Other works try to characterize the human activity directly from the video signal without segmenting the active regions. In [2] human activities are characterized by temporal templates. These templates try to convey information about “where” and “how” motion is performed. Two templates are created: a binary motion-energy-image which represents where the motion has occurred in the whole sequence, and a scalar motion-history-image which represents

∗This work was partially supported by FCT under project CAVIAR(IST-2001-37540)

†This work is integrated in project CAVIAR, which has the general goal of representing and recognizing contexts and situations. An introduction and the main goals of the project can be found in http://homepages.inf.ed.ac.uk/rbf/CAVIAR/caviar.htm
how motion occurs for each activity. Motion patterns have also been used in [9] based on the concept of “recency”. This work integrates several frames into a single image, assigning higher weights to the most recent frames. In [10], the human motion is characterized by the optical flow.

Several inference techniques have been used for the recognition of human activities using static and dynamic techniques. In [12] a single-person or person-to-person interactions are modelled by Hidden Markov Models (HMMs) and Coupled Hidden Markov Models (CHMMs). Both techniques are used to characterize the evolution of the person mass center along the video sequence. In [4] a Bayesian networks are used to for making inference about the events. In [11] activities are modelled using banks of switched dynamic models each of which tailored to a specific motion regime.

Geometric constraints have also been used e.g., using the layout of the surveillance region [13, 3]. In [1, 3] Finite State Machines (FSM) are used for gesture and activity recognition. The later uses prior knowledge about the scene, where regions of interest are defined (e.g., entrances and exits).

When the human motion is characterized by global features static pattern recognition methods can be used to classify the human activities. In [15] neural networks are used for this purpose.

The previous methods have been used to deal with single pedestrians or a very limited number of pedestrians [12]. To deal with the interaction among multiple pedestrians Bayesian networks have been proposed [8] since they are able to represent the dependencies among several random variables.

3 Statistical Model

We represent the human activity by the trajectory of its centroid. The time evolution of this feature is modelled by a dynamical model. Since a single model may not suffice to describe an entire trajectory, we use multiple dynamical models and a switching mechanism.

In this paper, a trajectory will be represented by a sequence of 2D locations, \( x = (x_1, \ldots, x_n) \), with \( x_t \in \mathbb{R}^2 \). We assume that the trajectory is the output of a bank of switched dynamical systems of the form

\[
 x_t - x_{t-1} = \Delta x_t = \mu_{k_t} + w_t, \tag{1}
\]

where \( k_t \in \{1, \ldots, c\} \) is the label of the active model at time instant \( t \), \( \mu_{k_t} \) is a (model-dependent) displacement vector, and the \( w_t \sim \mathcal{N}(0, Q_{k_t}) \) are independent Gaussian random variable, with covariances \( Q_{k_t} \).

Since the observations are \( \{\Delta x_t; t \in \mathbb{N}\}, \Delta x_t \in \mathbb{R}^d \) (\( d \) is the dimension of the observation vector), instead of \( x_t \), equation (1) describes an independent increment process, given \( k_t \), as shown in Fig. 1.

![Figure 1: Architecture of the proposed approach.](image)

Finally we assume that the sequence of model labels is composed of \( T \) constant segments: \( \{k_1, \ldots, k_1, k_2, \ldots, k_2, \ldots, k_T, \ldots, k_T\} \).

4 Segmentation and classification Algorithm

In order to segment and classify the different activities, we first observed that all trajectories concerning a common activity follow a typical route. Fig. 2 shows trajectories corresponding to a person entering a shop (left), leaving a shop (middle) or just passing in front of a shop (right).

This work demonstrates that elementary actions such as: “moving upwards”, “stopped”, “moving downwards”, “moving left” and “moving right” (i.e., \( \mathcal{M} = 5 \)), are representatives of the trajectories. The underlying idea is: given a test trajectory \( x_t = (x_1, \ldots, x_n) \), segment it into its elementary actions and classify the activity. The number of segments will depend on the activity being considered, as described later.

4.1 Model Parameter Estimation

To segment and classify a given trajectory we have to previously obtain the parameters of each dynamic model. To accomplish this, we collect tens of trajectory samples from each model.
This maximization can be performed in a nested way, “segment” the sequence (i.e., estimate $x$ way to attain this goal is to compute the likelihood of $m$ where $\hat{x}_j$ is known to have been generated by the $i$th model. Defining $\Delta \mathbf{X}' = \{\Delta x_1', \Delta x_2', \ldots, \Delta x_n'\}$ as the vector containing all the displacements in $i$th model of the training set, we have, for the $i$th model:

$$\hat{\mu}_i = \frac{1}{\|\Delta \mathbf{X}'\|} \sum \Delta x'_i, \quad \hat{Q}_i = \frac{1}{\|\Delta \mathbf{X}'\|} \sum (\Delta x' - \hat{\mu}_i)(\Delta x' - \hat{\mu}_i)^T,$$

where $\hat{\mu}_i$ and $\hat{Q}_i$ are standard estimates of the mean and the covariance matrix respectively.

4.2 Segmentation and Classification

Having defined the set of models and the corresponding parameters, one can now classify a test trajectory $x_t$. One way to attain this goal is to compute the likelihood of $x_t$ into the model space. In this paper, the activity depends on the number of the model switchings. In Fig. 2, we see that “passing” can be described by using only one model. The activities “entering” and “exiting” can be described by using two dynamical models. The fourth activity considered “browsing”, requires three models to be described; we define “browsing” when the person is walking, stop to see the shop-window and restarts walking. This behavior was observed in all the other samples of the activities which come about in this context. This means that we have to estimate the time instants in which the model switching happens.

Assuming that the sequence $x_t$ has $n$ samples and is described by $T$ segments (and $T$ is known) the log-likelihood is

$$L(m_1, \ldots, m_T, t_1, \ldots, t_{T-1}) = \log p(\Delta x_1, \ldots, \Delta x_n \mid m_1, m_2, \ldots, m_T, t_1, t_2, \ldots, t_{T-1})$$

where $m_1, \ldots, m_T$ is the sequence of model labels describing the trajectory and $t_i$ for $i = 1, \ldots, T - 1$ is the time instant when switching from model $m_i$ to $m_{i+1}$ occurs. If $T = 1$, there is no switching.

Due to the conditional independence assumption underlying (1), the log-likelihood can be written as

$$L(\Delta x_1, \ldots, \Delta x_n \mid m_1, \ldots, m_T, t_1, \ldots, t_{T-1}) = \sum_{j=1}^{T} \sum_{i=t_{j-1}}^{t_j} \log p(\Delta x_i \mid m_j) = \sum_{j=1}^{T} \sum_{i=t_{j-1}}^{t_j} \log N(\Delta x_i \mid \mu_{m_j}, Q_{m_j})$$

where we define $t_0 = 1$, $T$ is the number of segments and $t_j$ the switch time. Assuming that $T$ is known, we can “segment” the sequence (i.e., estimate $m_1, \ldots, m_T$ and $t_1, \ldots, t_{T-1}$) using the maximum-likelihood approach:

$$\hat{m}_1, \ldots, \hat{m}_T, \hat{t}_1, \ldots, \hat{t}_{T-1} = \arg \max_{m_1, \ldots, m_T} L(\Delta x_1, \ldots, \Delta x_n \mid m_1, \ldots, m_T, t_1, \ldots, t_{T-1})$$

This maximization can be performed in a nested way,

$$\hat{t}_1, \ldots, \hat{t}_{T-1} = \arg \max_{t_1, \ldots, t_{T-1}} \left\{ \max_{m_1, \ldots, m_T} L(\Delta x_1, \ldots, \Delta x_n \mid m_1, \ldots, m_T, t_1, \ldots, t_{T-1}) \right\}$$

In fact, the inner maximization can be decoupled as

$$\max_{m_1, \ldots, m_T} L(\Delta x_1, \ldots, \Delta x_n \mid m_1, \ldots, m_T, t_1, \ldots, t_{T-1}) = \sum_{j=1}^{T} \max_{m_j} \sum_{i=t_{j-1}}^{t_j} \log p(\Delta x_i \mid m_j)$$

where the maximization with respect to each of $m_j$ is a simple maximum likelihood classifier of sub-set of samples $\langle \Delta x_{t_j-1}, \ldots, \Delta x_{t_j} \rangle$ into one of a set of Gaussian classes. Finally, the maximization with respect to $t_1, \ldots, t_{T-1}$ is done by exhaustive search (this is never too expensive, since we consider a maximum of three segments).
4.3 Estimating the number of models of the activity

4.3.1 MDL Criterion

In the previous section, we derived the segmentation criterion assuming that the number of segments $T$ is known. As is well known, the same criterion cannot be used to select $T$, as this would always return the largest possible number of segments. We are thus in the presence of a model selection problem, which we address by using the minimum description length (MDL) criterion [14]. The MDL criterion for selecting $T$ is

$$
\hat{T} = \arg\min_T \left\{ -\log p(\Delta x_1, \ldots, \Delta x_n | \hat{m}_1, \ldots, \hat{m}_T, \hat{t}_1, \ldots, \hat{t}_{T-1}) \\
+ M(\hat{m}_1, \ldots, \hat{m}_T, \hat{t}_1, \ldots, \hat{t}_{T-1}) \right\}
$$

where $M(\hat{m}_1, \ldots, \hat{m}_T, \hat{t}_1, \ldots, \hat{t}_{T-1})$ is the number of bits required to encode the selected model indices and the estimated switching times. Notice that we do not have the usual $\frac{1}{2} \log n$ term because the real-valued model parameters (means and covariances) are assumed fixed (previously estimated). Finally, it is easy to conclude that

$$
M(\hat{m}_1, \ldots, \hat{m}_T, \hat{t}_1, \ldots, \hat{t}_{T-1}) \approx T \log c + (T - 1) \log n
$$

where $T \log c$ is the code length for the model indices $m_1, \ldots, m_T$, since each belongs to $\{1, \ldots, c\}$, and $(T - 1) \log n$ is the code length for $\hat{t}_1, \ldots, \hat{t}_{T-1}$, because each belongs to $\{1, \ldots, n\}$; we have ignored the fact that two switchings cannot occur at the same time, because $T \ll n$.

5 Experimental results

This section presents results with synthetic and real data. In the synthetic case, we have performed Monte Carlo tests. We have considered five models ($c = 5$) shown in Fig. 3. The synthetic models shown in Fig. 3(a) were obtained by simulating four activities of a person, using the generation model in (1). Fig. 4 shows examples of activities (the trajectory shape of “Leaving” is the same as “Entering”, however with opposite direction). Here, the thin (green) rectangles correspond to areas where the trajectory begins. The first sample of $x_t$ in these areas is random, because the agent may appear at random places in the scene. The wide (yellow) rectangle is the area in which occurs a model switching. In this figure the trajectories are generated with two segments (“Entering”, “Leaving”, “Passing”) and with three segments (“Browsing”).

For each activity we generate 100 test samples using (1) and classify each of them in one of the four classes. Fig. 5 shows the displacements $\Delta x_t$ (black dots) of the test sequences (“Entering” and “Passing”) overlapped with the five models. We can see that the displacements lie on right-up clusters (“Entering”) and right cluster (“Passing”). In this experiment, all the test sequences were correctly classified (100% accuracy).

![Figure 3](image_url)

Figure 3: Five models are considered to describe trajectory. Each color corresponds to a different model. Synthetic case (a), real case (b).

We also generated different test trajectories, this is because the exiting and entering may occur in different direction from the ones in Fig. 4. These examples are illustrated in Fig. 6. In this new experiment, the same 100% accuracy was also obtained.
The proposed algorithm was also tested with real data. The video sequences were acquired in the context of the EC funded project CAVIAR. All the video sequences comprise human activities in indoor plaza and shopping center observations of individuals and small groups of people. Ground truth was hand-labelled for all sequences\(^2\). Fig. 7 shows the bounding boxes as well as the centroid, which is the information used for the segmentation.

As in the synthetic case, we also generate the statistics of the considered models. The procedure is the same as in the previous case using training sequences. Fig. 3(b) shows the clusters of the models.

Fig. 8 shows several activities performed at the shopping center with the time instants of the model switching marked with small red circle. From this experiment, it can be seen that the proposed approach correctly determines the switching times between models.

We have tested the proposed approach in more than 40 trajectories from 25 movies of about 5 minutes each. We just present the results of some of those activities in Tables 1 and 2. These Tables show the penalized log-likelihood values (8) of each test sequence. The first table refers to all activities performed in the left-right direction, whilst the second table reports all activities performed in the opposite direction. In the first table the classes referring to entering, exiting, passing and browsing are right-upwards, downwards-right, right, right-stop-right respectively, whereas in the second table the classes are left-upwards, downwards-left, left and left-stop-left. It can be observed that the output classifier correctly assigns the activities into the corresponding classes, exhibiting good results as in the previous synthetic examples.

6 Conclusions

In this paper we have proposed and tested an algorithm for modelling, segmentation, and classification of human activities in a constrained environment. The proposed approach uses a switched dynamical models to represent the human trajectories. It was illustrated that the time instants are effectively well determined, despite of the significant random perturbations that the trajectory may contain. It is demonstrated that the proposed approach provides good

\(^2\)The ground truth labelled video sequences is provided at http://homepages.inf.ed.ac.uk/rbf/CAVIAR/.
results with synthetic and real data obtained in a shopping center. The proposed method is able to effectively recognize instances of the learned activities. The activities studied herein can be interpreted as atomic, in the sense that they are simple events. Compound actions or complex events can be represented as concatenations of the activities studied in this paper. This is one of the issues to be addressed in the future.

Acknowledgement: We would like to thank Prof. José Santos Victor of ISR and the members of CAVIAR project, for providing video data of human activities with the ground truth information.
Figure 8: Samples of different activities. The large circles are the computed times instants where the model switches: Entering (first column); exiting (second column); browsing (third column).

<table>
<thead>
<tr>
<th>Classes</th>
<th>Test trajectories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_1$</td>
</tr>
<tr>
<td>Entering</td>
<td>187.2</td>
</tr>
<tr>
<td>Exiting</td>
<td>401.0</td>
</tr>
<tr>
<td>Passing</td>
<td>359.7</td>
</tr>
<tr>
<td>Browsing</td>
<td>299.1</td>
</tr>
</tbody>
</table>

Table 1: Penalized Log-likelihood of several real activities performed in left-right direction: $E$- entering, $Ex$- exiting, $P$- passing, $B$- browsing.

<table>
<thead>
<tr>
<th>Classes</th>
<th>Test trajectories</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$E_1$</td>
</tr>
<tr>
<td>Entering</td>
<td><strong>116.2</strong></td>
</tr>
<tr>
<td>Exiting</td>
<td>277.6</td>
</tr>
<tr>
<td>Passing</td>
<td>210.0</td>
</tr>
<tr>
<td>Browsing</td>
<td>207.4</td>
</tr>
</tbody>
</table>

Table 2: Penalized Log-likelihood of several real activities performed in right-left direction: $E$- entering, $Ex$- exiting, $P$- passing, $B$- browsing.
References


Human Activity Learning and Segmentation using Partially Hidden Discriminative Models

Tran The Truyen †, Hung H. Bui ‡ and Svetha Venkatesh †

† Department of Computing, Curtin University of Technology, GPO Box U 1987, Perth, Western Australia. {trantt2, svetha}@cs.curtin.edu.au

‡ Artificial Intelligence Center, SRI International 333 Ravenswood Avenue, Menlo Park, CA 94025-3493, USA. bui@ai.sri.com

Abstract

Learning and understanding the typical patterns in the daily activities and routines of people from low-level sensory data is an important problem in many application domains such as building smart environments, or providing intelligent assistance. Traditional approaches to this problem typically rely on supervised learning and generative models such as the hidden Markov models and its extensions. While activity data can be readily acquired from pervasive sensors, e.g. in smart environments, providing manual labels to support supervised training is often extremely expensive. In this paper, we propose a new approach based on semi-supervised training of partially hidden discriminative models such as the conditional random field (CRF) and the maximum entropy Markov model (MEMM). We show that these models allow us to incorporate both labeled and unlabeled data for learning, and at the same time, provide us with the flexibility and accuracy of the discriminative framework. Our experimental results in the video surveillance domain illustrate that these models can perform better than their generative counterpart, the partially hidden Markov model, even when a substantial amount of labels are unavailable.

1 Introduction

An important task in human activity recognition from low-level sensory data is segmenting the data streams and labeling them with meaningful sub-activities. The labels can then be used to facilitate data indexing and organisation, to recognise higher levels of semantics, and to provide useful context for intelligent assistive agents. The segmentation modules are often built on top of low-level sensor components which produce primitive and often noisy streams of events (e.g. see [7]). To handle the uncertainty inherent in the data, current approaches to activity recognition typically employ probabilistic models such as the hidden Markov models (HMMs) [14] and more expressive models, such as stochastic context-free grammars (SCFGs) [7], hierarchical HMMs (HHMMs) [6], abstract HMMs (AHMMs) [2], and dynamic Bayesian networks (DBNs).

All of these models are essentially generative, i.e. they model the relation between the activity sequence $y$ and the observable data stream $x$ via the joint distribution $p(y|x)$. Maximum likelihood learning with these models is then performed by finding a parameter that optimises the joint probability $p(y|x)$. This modeling approach has two drawbacks in general. Firstly, it is often difficult to capture complex dependencies in the observation sequence $x$, as typically, simplifying assumptions need to be made so that the conditional distribution $p(x|y)$ is tractable. This limits the choice of features that one can use to encode multiple data streams. Secondly, it is often advantageous to optimise the conditional distribution $p(y|x)$ as we do not have to learn the data generative process. Thirdly, as we are only interested in finding the most probable activity sequence $y^* = \arg\max_y p(y|x)$, it is more natural to model $p(y|x)$ directly.

Thus the discriminative model $p(y|x)$ is more suitable to specify how an activity $y$ would evolve given that we already observe a sequence of observations $x$. In other words, the activity nodes, rather than being the parents, become the children of the observation nodes. With appropriate use of contextual information, the discriminative models can represent arbitrary, dynamic long-range interdependencies which are highly desirable for segmentation tasks. Moreover, whilst capturing unlabeled sensor data for training is cheap, obtaining labels in a supervised setting often requires expert knowledge and is time consuming. In many cases we are certain about some particular labels, for example, in surveillance data, when a person enters a room or steps on a pressure mat. Other labels (e.g. other activities that occur inside the room) are left unknown. Therefore, it is more desirable to employ the semi-supervised approach. Specifically, we consider two recent discriminative models, namely, the undirected Conditional Random Fields (CRFs) [9], (Figure 1(b)) and the directed Maximum Entropy Markov Models (MEMMs) [11] (Figure 1(a)). As the original models are fully observed, we provide a treatment of incomplete data for the CRFs and the MEMMs. The EM algorithm [5] is presented for both the models although it is not strictly required for the CRFs.
We provide experimental results in the video surveillance domain where we compare the performance of the proposed models and the equivalent generative HMMs [15] (Figure 1(c)) in learning and segmenting human indoor movement patterns. Out of three data sets studied, a common behaviour is that the HMM is outperformed by the discriminative counterparts even when a large portion of labels are missing. Providing contextual features for the models increases the performance significantly.

The novelty of this paper lies in the first work on modeling human activity using partially hidden discriminative models. Although semi-supervised learning has been investigated for a while, much work has concentrated on unstructured data and classification. There has been little work on structured data and segmentation and how much labeling effort are needed.

The remainder of the paper is organised as follows. Section 2 reviews related work in human activity segmentation and background in CRFs and MEMMs and in semi-supervision. Section 3 describes the partially hidden discriminative models. The paper then describes implementation and experiments and presents results in Section 4. The final section summarises major findings and further work.

2 Related work

Hidden Markov models (HMMs) have been used to model simple human activities and human motion patterns [18, 3, 1]. More recent approaches have used more sophisticated generative models to capture the hierarchical structure of complex activities. The abstract hidden Markov model (AHMM) [2] is used in [10] to model human transportation patterns from outdoor GPS sensors, and in [12] to model human indoor motion patterns from sensors placed in mobile robots. Using the AHMM, multiple levels of semantics can be built on top of the HMMs allowing flexibility in modeling the evolution of activities across multiple levels of abstraction. To learn the parameters, the expectation maximisation (EM) algorithm can be used. However, these models are generative, and are not suitable to work with arbitrary or overlapping features in the data streams.

Discriminative models specify the conditional probability \( p(y|x) \) without modeling the data \( x \). Let \( y = \{ y_{i,n} \} \) and assume that the probability \( p(y|x) \) is specified with respect to a graph \( G = (\mathcal{E}, \mathcal{Y}) \), where each vertex \( i \in \mathcal{Y} \) represents a random variable \( y_i \) and the edges \( e \in \mathcal{E} \) encode the correlation between variables. The graph \( G \) can be undirected, as in the Conditional Random Fields (CRFs) [9] (Figure 1(b)) or directed as in the Maximum Entropy Markov Models (MEMMs) [11] (Figure 1(a)). The CRFs define the model as follows

\[
p(y|x;\lambda) = \frac{1}{Z(x;\lambda)} \prod_c \Psi_c(y_c, x; \lambda)
\]

where \( c \) is the clique defined by the structure of \( G \), \( \Psi_c(y_c, x; \lambda) \) is the potential function defined over the clique \( c \), \( \lambda \) are model parameters, and \( Z(x;\lambda) = \sum_{y} \prod_c \Psi_c(y_c, x; \lambda) \) is the normalisation factor.

We consider the chain structure CRFs for our labeling tasks (Figure 1(b)), that is \( y = \{ y_{1,T} \} \). The potential function becomes \( \Psi_c(y_{t-1}, y_t, x; \lambda) \), which is then typically parameterised using the log-linear model \( \Psi_c(y_{t-1}, y_t, x; \lambda) = \exp(\sum \lambda_{jk} f_k(y_{t-1}, y_t, x)) \). The functions \( \{ f_k(y_{t-1}, y_t, x) \} \) are the features that capture the statistics of the data and the semantics at time \( t \). The parameters \( \lambda \) are the weight associated with the features and are estimated through training.

The MEMM is a directed, local version of the CRFs (Figure 1(a)), in which each source state \( j \) has a conditional distribution

\[
p_j(y_t|x; \lambda) = p(y_t|y_{t-1} = j, x; \lambda) = \frac{1}{Z(x, j)} \exp(\sum \lambda_{jk} f_k(x, y_t, j))
\]

where \( \lambda_{jk} \) are parameters of the source state \( y_{t-1} = j \). The MEMMs can also be considered as conditionally trained HMMs (e.g. see the difference between Figures 1(a,c)). Although CRFs solve the label bias problem associated with the local normalised MEMMs [9], we believe that the MEMMs are useful in learning and understanding activity patterns because they directly encode the temporal state evolution through the transition model \( p(y_t|y_{t-1} = j, x; \lambda) \).

Supervised learning in the CRFs and MEMMs typically maximises the conditional log-likelihood \[^1\] \( \mathcal{L}(\lambda) = \log p(y|x;\lambda) \). Gradient-based methods [16] are considered the fastest up to now.

Partially hidden models have received significant attention recently. The partially hidden Markov model (PHMM) proposed in [15] (Figure 1(c)) addresses the similar partial labeling problem as ours and we will use this model to compare with our discriminative models. In [13], CRFs with a hidden layer are introduced but labels are never given for this layer, thus they are not concerned with how robust the model is with respect to amount of missing data. The idea of constrained inference is introduced in [8] but they do not address the learning problem as we do. The more recent work in [4] extends the work of [8] to learning and addresses the interactive labeling effort by users. The results, however, are difficult to generalise to non-interactive applications in a non-active learning fashion.

[^1]: For multiple iid data instances, we should write \( \mathcal{L}(\lambda) = \sum p(x) \log p(y|x;\lambda) \) where \( p(x) \) is the empirical distribution of training data, but we drop this notation for clarity.
3 Partially hidden discriminative models

3.1 The models

In our partially hidden discriminative models, the label sequence \( y \) consists of a visible component \( v \) (e.g., labels that are provided manually, or are acquired automatically by reliable sensors) and a hidden part \( h \) (labels that are left unspecified or those we are unsure). The joint distribution of all visible variables \( v \) is therefore given as

\[
p(v|x; \lambda) = \sum_h p(v,h|x; \lambda) = \sum_h p(v|x; \lambda)
\]

CRFs. For the log-linear CRFs, we have

\[
p(y|x; \lambda) = \frac{1}{Z(x)} \prod_t \exp(\sum_k \lambda_k f_k(y_{t-1},y_t,x))
\]

where \( Z(x) = \sum_y \prod_t \exp(\sum_k \lambda_k f_k(y_{t-1},y_t,x)) \). In this case, the complexity of computing \( p(v|x; \lambda) \) is the same as that of computing the partition function \( Z(x) \) up to a constant. Note that \( Z(x) \) has the sum-product form, which can be computed efficiently using a single forward pass.

MEMMs. As stated in Section 2, directed models like the MEMMs are important in activity modeling because they naturally encode the state transitions given the observations. Here we offer a slightly more general view of the MEMMs in that we define a single model for all source states rather than separate models for each source state as in (2). In addition, as the model is discriminative, we do not have to model the observation sequence \( x \). Thus we are free to encode arbitrary information exacted from the whole sequence \( x \) to the local distribution. In our implementation, this is realised by using a sliding window of size \( s \) centred at the current time \( t \) to capture the local context of the observation. The local distribution reads

\[
p(y_t|\Omega_t,y_{t-1}; \lambda) = \frac{1}{Z(\Omega_t,y_{t-1})} \exp(\sum_k \lambda_k f_k(\Omega_t,y_{t-1},y_t))
\]

where \( \Omega_t = \{x_{(t-s_1)\ldots(t-s_2)}\} \) is the context of size \( s = s_1 + s_2 + 1 \), and the parameter set \( \{\lambda_k\} \) is now shared across the states. This view of MEMMs reduces to the original model if the feature set \( \{f_k(\Omega_t,y_{t-1},y_t)\} \) consists of only indicator functions of states. The new view thus enjoys the same probabilistic inference properties but the learning is slightly different from the MEMM as it incorporates the structural constraint via the shared parameters while the MEMMs learns each local classifiers independently. The use of contextual features reflects the fact that the the current activity \( y_t \) is generally correlated with the past and the future of sensor data.

As the graphical model of the MEMMs forms a Markov chain conditioned on the observation \( x \), the joint incomplete distribution is therefore

\[
p(v|x; \lambda) = \sum_h \prod_t p(y_t|\Omega_t,y_{t-1}; \lambda)
\]

Again, this is a sum-product case, which can be computed by a single forward pass.

3.2 Parameters learning

To learn the model parameters that are best explained by the data, we maximise the penalised log-likelihood

\[
\Lambda(\lambda) = \mathcal{L}(\lambda) - \frac{1}{2\sigma^2}||\lambda||^2
\]

where \( \mathcal{L}(\lambda) = \log p(v|x; \lambda) \). The regularisation term is needed to avoid over-fitting when only limited data is available for training. For simplicity, the parameter \( \sigma \) is shared among all dimensions and is selected experimentally.

As with incomplete data, an alternative to maximise the log-likelihood is using the EM algorithm [5] whose Expectation (E-step) is to calculate the quantity

\[
Q(\lambda^j, \lambda) = \sum_h p(h|v,x; \lambda^j) \log p(h,v|x)
\]
and the Maximisation (M-step) maximises the concave lower bound of the log-likelihood \( Q(\lambda^t, \lambda) - \frac{1}{2} \sum_{k}\|\lambda_k\|^2 \) with respect to \( \lambda \). Unlike Bayesian networks, the log-linear models do not yield closed form solutions in the the M-step. However, as the function \( Q(\lambda^t, \lambda) \) is concave, it is still advantageous to optimise with efficient Newton-like algorithms.

CRFs. For the partially hidden CRFs, the gradient of incomplete likelihood reads

\[
\frac{\partial \mathcal{L}(\lambda)}{\partial \lambda_k} = \sum_t \sum_{v} p(h_{t-1}, h_{t}|v; \lambda) f_k(h_{t-1}, h_{t}, v) - \sum_{t} \sum_{y_{t-1}y_{t}} p(y_{t-1}, y_{t}|v; \lambda) f_k(y_{t-1}, y_{t}, x)
\]

(8)

Zeroing the gradient does not yield an analytical solution, so typically iterative numerical methods such as conjugate gradient and Newton methods are needed. The gradient of the lower bound in the EM framework of (7) is similar to (8), except that the pairwise marginals \( p(h_{t-1}, h_{t}|v; \lambda) \) are now replaced by the marginals of the previous EM iteration \( p(h_{t-1}, h_{t}|v; \lambda) \). The pairwise marginals \( p(y_{t-1}, y_{t}|x) \) can be computed easily using a forward pass and a backward pass in the standard message passing scheme on the chain. Details are omitted for space constraint.

MEMMs. In learning of MEMMs, the E-step is to calculate

\[
Q(\lambda^t, \lambda) = \sum_t \sum_{h_{t-1}} p(h_{t-1}|v; \Omega; \lambda^t) \sum_{h_{t}} p(h_{t}|h_{t-1}, \Omega; \lambda^t) \log p(h_{t}|h_{t-1}, \Omega; \lambda)
\]

and the M-step is to solve the zeroing gradient equation

\[
\frac{\partial Q(\lambda^t, \lambda)}{\partial \lambda_k} = \sum_t \sum_{h_{t-1}} p(h_{t-1}|v; \Omega; \lambda^t) \left\{ \sum_{h_{t}} p(h_{t}|h_{t-1}, \Omega; \lambda^t) f_k(h_{t-1}, h_{t}, \Omega) - \sum_{y_{t}} p(y_{t}|h_{t-1}, \Omega; \lambda) f_k(h_{t-1}, y_{t}, \Omega) \right\}
\]

Computation of the EM reduces to that of marginals and state transition probabilities, which can be carried out efficiently in the Markov chain framework using dynamic programming.

3.3 Segmentation

For segmentation, we use the MAP assignment \( y^* = \arg \max_y \ p(y|x, \lambda) \) to infer the most probable label sequence \( y^* \) for a given data sequence \( x \). For both the CRFs and MEMMs, the Viterbi algorithm [14] can be naturally adapted. If some labels are provided (e.g. by some reliable sensors, or by users in interactive applications) we have the so-called constrained inference [8], but this is a trivial adaptation of the Viterbi decoding [14].

3.4 Comparison with the PHMMs

The main difference between the models described in this section (Figure 1(a,b)) and the PHMMs [15] (Figure 1(c)) is the conditional distribution \( p(y|x) \) in discriminative models compared to the joint distribution \( p(y,x) \) in the PHMMs. The data distribution of \( p(x) \) and how \( x \) is generated are not of concern in the discriminative models. In the PHMMs, on the contrary, the observation point \( x_t \) is presumably generated by the parent label node \( y_t \), so care must be taken to ensure proper conditional independence among \( \{x_t\}_{t=1}^T \). This difference has an implication that, while the discriminative models may be good to encode the output labels directly with arbitrary information extracted from the whole observation sequence \( x \), the PHMMs better represent \( x \) when little information is associated with \( y \). For example, when \( y \) is totally missing, \( p(x) = \sum_y p(y,x) \) is still modeled in the PHMMs and provides useful information. Our experiments in the next section show this difference more clearly.

Moreover, whilst we employ the log-linear models with unconstrained parameters, the PHMMs use the constrained transition and emission probabilities as parameters. In terms of modeling label ‘visibility’, the PHMMs are more general as they allow a subset of labels to be associated with certain nodes, and not only a full set as in hidden nodes or a single label as in visible nodes. However, it is quite straightforward to extend our partially hidden discriminative models to incorporate the same representation.

4 Experiments and results

Our task is to infer the activity patterns of a person (the actor) in a video surveillance scene. The observation data is provided by static cameras while the labels, which are activities such as ‘go-from-A-to-B’ during the time interval \([t_a, t_b] \) (see Table 1), are recognised by the trained models.

4.1 Setup and data

The surveillance environment is a 4 x 6m² dining room and kitchen (Figure 2). Two static cameras are installed to capture the video of the actor making some meals. There are six landmarks which the person can visit during the meals: door, TV chair, fridge, stove, cupboard, and dining chair. Figure 2 shows the room and the special landmarks viewed from the two cameras.
Figure 2: The environment and scene viewed from the two cameras.

Table 1: The primitive activities (the labels).

<table>
<thead>
<tr>
<th>Activity</th>
<th>Landmarks</th>
<th>Activity</th>
<th>Landmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Door→Cupboard</td>
<td>7</td>
<td>Fridge→TV chair</td>
</tr>
<tr>
<td>2</td>
<td>Cupboard→Fridge</td>
<td>8</td>
<td>TV chair→Door</td>
</tr>
<tr>
<td>3</td>
<td>Fridge→Dining chair</td>
<td>9</td>
<td>Fridge→Stove</td>
</tr>
<tr>
<td>4</td>
<td>Dining chair→Door</td>
<td>10</td>
<td>Stove→Dining chair</td>
</tr>
<tr>
<td>5</td>
<td>Door→TV chair</td>
<td>11</td>
<td>Fridge→Door</td>
</tr>
<tr>
<td>6</td>
<td>TV chair→Cupboard</td>
<td>12</td>
<td>Dining chair→Fridge</td>
</tr>
</tbody>
</table>

We study three scenarios corresponding to the person making a short meal (denoted by SHORT_MEAL), having a snack (HAVE_SNACK), and making a normal meal (NORMAL_MEAL). Each scenario comprises of a number of primitive activities as listed in Table 1. Figure 3 shows the association between scenarios and their primitive activities. The SHORT_MEAL data set has 12 training and 22 testing video sequences; and each of the HAVE_SNACK and NORMAL_MEAL data sets consists of 15 training and 11 testing video sequences. For each raw video sequence captured, we use a background subtraction algorithm to extract a corresponding discrete sequence of coordinates of the person based on the person’s bounding box. The training sequences are partially labeled, indicated by the portion of missing labels \( \rho \). The testing sequences provide the ground-truth for the algorithms. The sequence length ranges from \( T = 20 - 60 \) and the number of labels per sequence is allowed to vary as \( T \ast (1 - \rho) \) where \( \rho \in [0, 100\%] \).

Figure 3: Associated primitive activities.

We apply standard evaluation metrics such as precision \( P \), recall \( R \), and the \( F \)1 score given as \( F1 = 2 \ast P \ast R / (P + R) \) on a per-token basis.

4.2 Feature design and contextual extraction

Features are crucial components of the model as they tie raw observation data with semantic outputs (i.e. the labels). The features need to be discriminative enough to be useful, and at the same time, they should be as simple and intuitive as possible to reduce manual labour. The current raw data extracted from the video contains only \((X,Y)\) coordinates. From each coordinate sequences, at each time slice \( t \), we extract a vector of five elements from the observation
sequence \( g(x,t) = (X_t, Y_t, uX_t, uY_t, s_t) = \sqrt{uX_t^2 + uY_t^2} \), which correspond to the \((X, Y)\) coordinates, the \(X\) & \(Y\) velocities, and the speed, respectively. Since the extracted coordinates are fairly noisy, we use the average velocity measurement within a time interval of small width \(w\), i.e. \(uX_t = (X_{t+w/2} - X_{t-w/2})/w\). Typically, these observation-based features are real numbers and are normalised so that they have a similar scale.

We decompose the feature set \( \{f_k(y_{t-1}, y_t, x_t)\} \) into two subsets: the state-observation features

\[
    f_{l,m,e}(x, y_t) := I[y_t = l]h_m(x, t, \epsilon) \tag{10}
\]

and the state-transition features

\[
    f_{l_1,l_2}(y_{t-1}, y_t) := I[y_{t-1} = l_1]I[y_t = l_2] \tag{11}
\]

where \(m = 1..5\) and \(h_m(x, t, \epsilon) = g_m(x, t + \epsilon)\) with \(\epsilon = -s_1, ... , 0, ... , s_2\) for some positive integers \(s_1, s_2\). The state-observation features in (10) thus incorporate neighbouring observation points within a sliding window of width \(s = s_1 + s_2 + 1\). This is intended to capture the correlation of the current activity with past and future observations, and is a realisation of the temporal context \(\Omega_t\) of the observations in (5). Thus the feature set has \(K = 5s|\mathcal{Y}| + |\mathcal{Y}|^2\) features, where \(|\mathcal{Y}|\) is the number of distinct label symbols.

To have a rough idea of how the observation context influences the performance of the models, we try different window sizes \(s\) (see Equation (2)). The experiments show that incorporating the context of observation sequences does help to improve the performance significantly (see Figure 4). We did not try exhaustive searches for the best context size, nor did we implement any feature selection mechanisms. As the number of features scales linearly with the context size as \(K = 5s|\mathcal{Y}| + |\mathcal{Y}|^2\), where \(s\) can be any integer between 1 and \(T\), where \(T\) is the sequence length, clearly a feature selection algorithm is needed when we want to capture long range correlation. For the practical purposes of this paper, we choose \(s = 5\) for both CRFs and MEMMs. Thus in our experiments, CRFs and MEMMs share the same feature set, making the comparison between the two models consistent.

4.3 Performance of models

To evaluate the performance of discriminative models against the equivalent generative counterparts, we implement the PHMMs (Figure 1(c)). The features extracted from the sensor data for the PHMMs include the discretised position \(\mathbf{X}_t\), velocities, \(\mathbf{Y}_t\), and the speed, respectively. Since the extracted coordinates are fairly noisy, we use the average velocity measurement within a time interval of small width \(w\), i.e. \(uX_t = (X_{t+w/2} - X_{t-w/2})/w\). Typically, these observation-based features are real numbers and are normalised so that they have a similar scale.

We decompose the feature set \( \{f_k(y_{t-1}, y_t, x_t)\} \) into two subsets: the state-observation features

\[
    f_{l,m,e}(x, y_t) := I[y_t = l]h_m(x, t, \epsilon) \tag{10}
\]

and the state-transition features

\[
    f_{l_1,l_2}(y_{t-1}, y_t) := I[y_{t-1} = l_1]I[y_t = l_2] \tag{11}
\]

where \(m = 1..5\) and \(h_m(x, t, \epsilon) = g_m(x, t + \epsilon)\) with \(\epsilon = -s_1, ... , 0, ... , s_2\) for some positive integers \(s_1, s_2\). The state-observation features in (10) thus incorporate neighbouring observation points within a sliding window of width \(s = s_1 + s_2 + 1\). This is intended to capture the correlation of the current activity with past and future observations, and is a realisation of the temporal context \(\Omega_t\) of the observations in (5). Thus the feature set has \(K = 5s|\mathcal{Y}| + |\mathcal{Y}|^2\) features, where \(|\mathcal{Y}|\) is the number of distinct label symbols.

To train discriminative models, we implement the non-linear conjugate gradient (CG) of Polak-Ribière and the limited memory quasi-Newton L-BFGS. After several pilot runs, we select the L-BFGS to optimise the objective function in (7) directly. In the case of MEMMs, the regularised EM algorithm is chosen together with the CG. The algorithms stop when the rate of convergence is less than \(10^{-5}\). The regularisation constants are empirically selected as \(\sigma = 5\) in the case of CRFs, and \(\sigma = 20\) in the case of MEMMs.

For the PHMMs, it is observed that the initial parameter initialisation is critical to learn the correct model. Random initialisations often result in very poor performance. This is unlike the discriminative counterparts in which all initial parameters can be trivially set to zeros (equally important).

Table 2 and Figure 5 show performance metrics (precision, recall and \(F1\)-score) of all models considered in this paper averaged over 10 repetitions. The three models have equivalent graphical structures. The CRFs and MEMMs share the same feature set but different from that of PHMMs. The generative PHMMs are outperformed by the discriminative counterparts in all cases given sufficient labels. This clearly matches the theoretical differences between these models in that when there are enough labels, richer information can be extracted in the discriminative framework, i.e. modeling \(p(y|x)\) is more suitable. On the other hand, when only a few labels are available, the unlabeled data is
Figure 5: Average performance of models (a: SHORT_MEAL, b: HAVE_SNACK, c: NORMAL_MEAL). x-axis: portion of missing labels (%) and y-axis: the averaged F-score (%) over all states and 10 repetitions.

Table 2: The averaged precision ($P$) and recall ($R$) over all labels and over 10 repetitions. Top row contains missing portion $\rho$. The three scenarios: SM=SHORT_MEAL, HS=HAVE_SNACK, NM=NORMAL_MEAL.

<table>
<thead>
<tr>
<th>Data</th>
<th>Model</th>
<th>Metric</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM</td>
<td>CRF</td>
<td>$P$</td>
<td>86.6</td>
<td>86.3</td>
<td>88.1</td>
<td>86.9</td>
<td>87.0</td>
<td>89.9</td>
<td>88.4</td>
<td>83.8</td>
<td>83.8</td>
<td>72.5</td>
</tr>
<tr>
<td>SM</td>
<td>CRF</td>
<td>$R$</td>
<td>87.4</td>
<td>87.1</td>
<td>88.1</td>
<td>87.7</td>
<td>87.4</td>
<td>91.3</td>
<td>90.1</td>
<td>81.6</td>
<td>82.5</td>
<td>68.5</td>
</tr>
<tr>
<td>SM</td>
<td>MEMM</td>
<td>$P$</td>
<td>81.7</td>
<td>87.8</td>
<td>87.0</td>
<td>84.2</td>
<td>85.2</td>
<td>83.1</td>
<td>81.2</td>
<td>80.5</td>
<td>73.2</td>
<td>57.0</td>
</tr>
<tr>
<td>SM</td>
<td>MEMM</td>
<td>$R$</td>
<td>83.4</td>
<td>88.4</td>
<td>87.5</td>
<td>84.2</td>
<td>86.1</td>
<td>82.7</td>
<td>81.5</td>
<td>75.8</td>
<td>67.8</td>
<td>55.2</td>
</tr>
<tr>
<td>SM</td>
<td>HMM</td>
<td>$P$</td>
<td>82.3</td>
<td>82.3</td>
<td>82.3</td>
<td>81.1</td>
<td>81.2</td>
<td>80.8</td>
<td>81.2</td>
<td>79.9</td>
<td>73.4</td>
<td>66.9</td>
</tr>
<tr>
<td>SM</td>
<td>HMM</td>
<td>$R$</td>
<td>83.2</td>
<td>83.2</td>
<td>83.2</td>
<td>83.7</td>
<td>84.1</td>
<td>83.3</td>
<td>84.1</td>
<td>83.1</td>
<td>75.7</td>
<td>70.9</td>
</tr>
<tr>
<td>HS</td>
<td>CRF</td>
<td>$P$</td>
<td>91.4</td>
<td>90.4</td>
<td>90.6</td>
<td>91.5</td>
<td>92.1</td>
<td>89.7</td>
<td>91.3</td>
<td>91.5</td>
<td>90.7</td>
<td>89.5</td>
</tr>
<tr>
<td>HS</td>
<td>CRF</td>
<td>$R$</td>
<td>92.4</td>
<td>91.5</td>
<td>90.1</td>
<td>90.6</td>
<td>91.7</td>
<td>90.0</td>
<td>91.9</td>
<td>91.5</td>
<td>91.1</td>
<td>88.8</td>
</tr>
<tr>
<td>HS</td>
<td>MEMM</td>
<td>$P$</td>
<td>89.9</td>
<td>88.9</td>
<td>90.8</td>
<td>89.2</td>
<td>91.5</td>
<td>88.7</td>
<td>89.6</td>
<td>89.4</td>
<td>85.1</td>
<td>80.0</td>
</tr>
<tr>
<td>HS</td>
<td>MEMM</td>
<td>$R$</td>
<td>91.2</td>
<td>90.3</td>
<td>91.4</td>
<td>89.5</td>
<td>93.7</td>
<td>90.4</td>
<td>91.3</td>
<td>91.0</td>
<td>87.7</td>
<td>81.4</td>
</tr>
<tr>
<td>HS</td>
<td>HMM</td>
<td>$P$</td>
<td>84.7</td>
<td>84.7</td>
<td>84.4</td>
<td>85.0</td>
<td>85.4</td>
<td>85.3</td>
<td>85.3</td>
<td>85.3</td>
<td>84.0</td>
<td>79.4</td>
</tr>
<tr>
<td>HS</td>
<td>HMM</td>
<td>$R$</td>
<td>88.5</td>
<td>88.5</td>
<td>88.1</td>
<td>87.2</td>
<td>87.6</td>
<td>87.3</td>
<td>87.3</td>
<td>87.3</td>
<td>87.4</td>
<td>83.4</td>
</tr>
<tr>
<td>NM</td>
<td>CRF</td>
<td>$P$</td>
<td>87.1</td>
<td>88.9</td>
<td>85.5</td>
<td>83.7</td>
<td>87.4</td>
<td>85.4</td>
<td>85.0</td>
<td>86.8</td>
<td>85.8</td>
<td>74.0</td>
</tr>
<tr>
<td>NM</td>
<td>CRF</td>
<td>$R$</td>
<td>83.5</td>
<td>88.5</td>
<td>81.8</td>
<td>80.7</td>
<td>86.6</td>
<td>85.7</td>
<td>81.5</td>
<td>86.3</td>
<td>84.9</td>
<td>72.8</td>
</tr>
<tr>
<td>NM</td>
<td>MEMM</td>
<td>$P$</td>
<td>85.4</td>
<td>85.0</td>
<td>84.6</td>
<td>83.5</td>
<td>84.8</td>
<td>81.9</td>
<td>77.9</td>
<td>78.3</td>
<td>73.0</td>
<td>62.0</td>
</tr>
<tr>
<td>NM</td>
<td>MEMM</td>
<td>$R$</td>
<td>81.7</td>
<td>82.1</td>
<td>81.3</td>
<td>81.0</td>
<td>84.9</td>
<td>81.4</td>
<td>78.4</td>
<td>79.7</td>
<td>76.9</td>
<td>62.6</td>
</tr>
<tr>
<td>NM</td>
<td>HMM</td>
<td>$P$</td>
<td>79.1</td>
<td>79.1</td>
<td>79.1</td>
<td>79.1</td>
<td>79.8</td>
<td>79.8</td>
<td>80.0</td>
<td>77.1</td>
<td>74.7</td>
<td>58.3</td>
</tr>
<tr>
<td>NM</td>
<td>HMM</td>
<td>$R$</td>
<td>80.4</td>
<td>80.4</td>
<td>80.4</td>
<td>80.4</td>
<td>81.3</td>
<td>81.3</td>
<td>81.6</td>
<td>79.5</td>
<td>78.0</td>
<td>63.8</td>
</tr>
</tbody>
</table>

important so it makes sense to model and optimise $p(x, y)$ as in the generative framework. On all data sets, the CRFs outperform the other models. These behaviours are consistent with the results reported in [9] in the fully observed setting. MEMMs are known to suffer from the label-bias problem [9], thus their performance does not match that of CRFs, although MEMMs are better than HMMs given enough training labels. In the HAVE_SNACK data set, the performance of MEMMs is surprisingly good.

A striking fact about the globally normalised CRFs is that the performance persists until most labels are missing. This is clearly a big time and effort saving for the labeling task.

5 Conclusions and further work

In this work, we have presented a semi-supervised framework for activity recognition on low-level noisy data from sensors using discriminative models. We illustrated the appropriateness of the discriminative models for segmentation of surveillance video into sub-activities. As more flexible information can be encoded using feature functions, the discriminative models can perform significantly better than the equivalent generative HMMs even when a large portion of the labels are missing. CRFs appear to be a promising model as the experiments show that they consistently outperform other models in all three data sets. Although less expressive than CRFs, MEMMs are still an important class of models as they enjoy the flexibility of the discriminative framework and enable online recognition as in directed graphical models.

Our study shows that primitive and intuitive features work well in the area of video surveillance. Semantically-rich and more discriminative contextual features can be realised through the technique of a sliding window. The
wide context is especially suitable for the current problem because human activities are clearly correlated in time and space. However, to obtain the optimal context and to make use of the all information embedded in the whole observation sequence, a feature selection mechanism remains to be designed in conjunction with the models and training algorithms presented in this paper.

Although flat CRFs and MEMMs can represent arbitrarily high-level of activities, in many situations it may be more appropriate to structure the activity semantics into multiple layers or into a hierarchy. Future work will include models such as Dynamic Conditional Random Fields (DCRFs) [17], conditionally trained Dynamic Bayesian Networks and hierarchical model structures. A drawback of the log-linear models considered here is the slow learning curve compared to the traditional EM algorithm in Bayesian networks. It is therefore important to investigate more efficient training algorithms.

Acknowledgments

Hung Bui is supported by the Defense Advanced Research Projects Agency (DARPA), through the Department of Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010.

The authors would like to thank reviewers for suggestions to improve the paper’s presentation. The Matlab code of the L-BFGS algorithm and of the conjugate gradient algorithm of Polak-Ribière is adapted from S. Ulbrich and C. E. Rasmussen, respectively. The implementation of PHMMs is based on the HMMs code by Sam Roweis.

References


Exploring techniques for behaviour recognition via the CAVIAR modular vision framework

David Tweed, Wanli Feng, Robert Fisher, José Bins & Thor List
Institute for Perception, Action and Behaviour, Dept of Informatics, Edinburgh University, {tweed,rbf,jbfilho,tlist}@inf.ed.ac.uk

Abstract

Visual analysis tasks are often very difficult and computer systems for tackling them are most effectively implemented as a network of routines which solve constituent subproblems with well-defined specifications. Most of these tasks do not have a clear-cut best algorithm, particularly dealing with real-world data. The problem arises how to characterise the performance on these sub-tasks in order to choose the best implementation for a given application, particularly when several techniques are used within a dataflow graph, with the output of later algorithms dependent on the output of earlier routines. In this work we look at competing algorithms for a simple behaviour recognition task within the framework of the CAVIAR modular vision platform.

1 Introduction

Creating software for challenging real-world visual analysis tasks is made much more tractable by adopting a modular approach, decomposing the problem into well-defined subtasks. Selecting the best algorithms the various sub-tasks is predicated upon characterising the performance of these routines.

Although the subtasks may correspond to operations a human is aware of performing – eg, detecting sudden movement – often they are motivated by creating intermediate data representations for use as a stepping-stone between low- and high-level representations, eg, consider image segmentation. This can be defined as dividing an image into contiguous regions maximising some measure of ‘intra-region homogeneity and cross-region difference’. Many applications segment an image into ‘potential objects’ before processing this reduced data to, eg, detect moving objects or decide if certain objects are present. Segmentation is clearly a useful and meaningful computer vision sub-task. However, virtually no real world task has subjects aware of segmenting their visual fields.

For a natural task, performance can be characterised by obtaining a ground truth dataset for an input dataset from human test subjects and then comparing this with the system’s output via established methodologies such as Receiver Operating Characteristic (ROC) curves [3]. This assumes ‘success’ is defined by human performance on the task. The extension to less natural sub-tasks by explaining the task to test subjects and getting them to produce ground truth data raises two issues: Firstly, the task definition has often been guided by algorithmic foreknowledge, so there can be an influence on the supposedly independent ground truth by the algorithm. More significantly, success is really defined not by human performance on this subtask, but rather by how the overall system performance on the end application (measured wrt a ground truth) is affected by using a given sub-algorithm. As an example, imagine a system for detecting cars starting with segmented images where the detection performance is governed just by the how accurately the distinctively shaped wing-mirrors are segmented, making comparison against a general segmentation ground truth misleading. If one knew wing-mirrors were crucial a tailored ground truth could be created, but in general one does not know these details; indeed, black box modules make non-experimental determination of the important aspects of the subtask output impossible. (We use segmentation in this and our later examples to illustrate the framework since segmentation is a more widely known basic subtask than the specialised behaviour analysis tasks which we evaluate later.)

This paper looks at how the final performance of a pipeline of analysis modules, which build up estimates of the behaviour of tracked individuals in a video sequence, varies with the performance of the earlier modules, an issue arising within the development of the surveillance system CAVIAR. It is important because the potential interactions/correlations between subtasks make attempting to tune the module in isolation misleading. This problem will worsen as the vision system becomes more modular.

Previous work. There has been much impressive work in the vision community on building systems for complex visual analysis, especially recently (see eg, the survey article [14]). Since in this work we are focusing on performance evaluation within the context of a larger task, we discuss in detail previous work in the setting of understanding and characterising algorithm performance.

Most machine learning focuses on two errors: false positives $p$ and false negatives $n$, which a given algorithm can trade-off by varying its parameters. Algorithms thus have ROC curves [3] in $p–n$ space characterising achievable
results. To build a system we choose a p–n ratio and select the associated parameters. Work such as [5] extends this to the general situation where we have N measures – eg, runtime, storage, etc – where it’s difficult to specify the trade-off a priori and there are multiple algorithms to compare. It should also be noted there is extensive literature (eg, [7, 9]) on statistically valid ways of estimating the variance of performance outputs with respect to variability in the input data, particularly given limited test data. Estimates of variance are necessary to decide whether differences between algorithms are significant.

In this paper we consider pipelined processes where the output of earlier modules becomes the input of later ones. In this vein, [16] considers the task of retrieving images with between l% and h% of the image being in some class (eg, sky, foliage, etc). Their detectors process sub-blocks of the image independently but can produce both false positives and negatives. A preprocessing module is developed which converts the user query to a synthetic query (l’, h’) designed to ‘cancel out’ expected detector errors and produce output matching to the original query. [11] investigates how the performance of various face recognition algorithms depends upon the accuracy of the eye localisation algorithm used for preprocessing. Experimental results suggest (counterintuitively) that absolute eye position errors have less effect on the recognition rates than merely eye separation and orientation (essentially because these features are used for normalisation). Finally, [10] analyses the performance of region classification algorithms given different segmentation algorithms in terms of extended ROC curves highlighting performance (with variance) for multi-class assignment. They demonstrate that a sensitivity to the choice of segmentation algorithm that is (again non-obviously) class-dependent.

2 The CAVIAR modular vision architecture

The name ‘CAVIAR system’ [2] refers to both a general software framework for implementing modular vision systems and a test system using example modules for visual surveillance/behaviour analysis. We give an overview, concentrating on issues of relevant to performance characterisation and omitting other interesting aspects of the architecture.

One way to look at the CAVIAR system is to start with the notion of dataset descriptions. These are ‘meta-data’ which essentially specify names and associated data structures. For example, there is a dataset named TrackedObjects which defines a syntactic representation for bounding boxes of individuals as they move through a video sequence. The lowest-level software element is the module, which is a block of code combined with a record of the datasets it takes as input and produces as output. The module shares the common understanding about what datasets mean, analogous to programming language interfaces (eg, ‘TrackedObjects describes individuals from a low-level tracker’).

The highest level software is the controller, a standard routine which is responsible for taking a collection of modules and figuring out firstly how they connect to form a complete system and then running this system over some input data. Connections are determined by building a directed-graph-like structure with two kinds of nodes – modules and datasets – and edges connecting a module with its inputs and outputs. In the case where each dataset is listed as an output for exactly one module this generates a simple dataflow analogous to a flowchart. However, the real utility arises when there is more than one module producing output x. In the simplest case (illustrated in Fig 1) we have two modules SEGMENT1 and SEGMENT2 which both produce dataset Regions as output. The controller automatically deduces that it can obtain Regions from either SEGMENT1 or SEGMENT2. This knowledge of functional equivalence can be used by the system controller to prompt the user to make a fixed choice of modules before running the system, to switch based upon static rules learned from offline performance characterisation, or to use a sophisticated auto-critical framework utilising intra-module feedback to swap modules (cf [4]). The first two approaches can use the analyses we develop here.

This highlights that the system is ‘discovered from the bottom up’ rather than specified top down: the controller does not have any built-in expectation about what inputs the Regions dataset needs or whether a single or multiple modules (using intermediate datasets) generate it, instead it splicing together modules to generate all outputs.

3 Behaviour analysis & interpretation

We used 80 image sequences of 500–2700 frames obtained from 3 separate camera-setups France, PortugalAcross, PortugalFront (taken from the CAVIAR datasets available from [1]). The ground truth describes the behaviour of both the individuals in the sequence and of groups of people, although here we only use individual behaviour. Since we will
be assessing performance against a pre-existing ground truth, the human interpretation incorporated into the ground truth must be structured into a form algorithms can output. The ground truth describes the behaviour of people within the sequences at a various levels of sophistication via four attributes per frame (expanded upon in [4]).

The two low level attributes are move and role. move divides the gross individual motion into {inactive, active, walking, running}. The role, which specifies more intensional short-term behaviour such as browsing, walking and shop-entering/shop-exiting, provides an immediate indication of the type of activity. (The role extends to describing relationships between individuals during group interactions, eg, context fighting involves participants in with roles of fighter and victim.)

Whole-sequence behaviour is specified via a set of transition graph models, with some typical examples shown in Fig 2c. The type of node called the situation and identifies individual frame activity within the context of the overall behaviour and is typically probabilistically related to the move and role determinations. The arrows show allowable transitions between the situation nodes for adjacent frames (including the explicitly shown self-transitions). Finally an x indicates that the given situation node must occur in order for the behaviour to occur. Without this we could, eg, match the Idleness graph in Fig 2a:iii against a sequence of move nodes, a result which is both intuitively wrong and creates ambiguity with the walk behaviour.

By providing these whole-sequence constraints the system can identify a whole sequence as, eg, browsing, even though the frames containing ‘looking into shop-windows’ might be a small fraction of the frames where the individual is walking between windows. The ‘true’ behaviour of an individual over the video sequence is (by fiat) described by the graph in Fig 2a:iii against a sequence of move nodes, a result which is both intuitively wrong and creates ambiguity with the walk behaviour.

By providing these whole-sequence constraints the system can identify a whole sequence as, eg, browsing, even though the frames containing ‘looking into shop-windows’ might be a small fraction of the frames where the individual is walking between windows. The ‘true’ behaviour of an individual over the video sequence is (by fiat) described by the graph in Fig 2a:iii against a sequence of move nodes, a result which is both intuitively wrong and creates ambiguity with the walk behaviour.

The graphs were finalised before sequences were annotated with ground truth. A typical set of images from one video sequence and the common data representation used for both the ground truth and analysis output is shown in Fig 2b. Consistency with the context graphs was enforced during markup by the labelling tool. However, the context graphs still leave many features that might be used for modelling unspecified, eg, module ProbContext estimates transition probabilities to enable HMM-based decoding.

This division into low- and high-level attributes is not completely clear-cut. The role can be relatively high level, e.g., browsing, but they are fundamentally discernable using only short-term analysis. In contrast, the situation describes a stage within a specific behaviour. Although we use this progression from move & role to situation and finally context in our analysis framework, it may not be necessary to get lower level attributes totally correct to get higher level ones correct: in particular within the correct context graph the precise frames where transitions between situation nodes occur may be highly ambiguous; however the overall match with the correct context may be clear.

Modular behaviour analysis. We want to break down the behaviour inference into three stages: estimating the move, estimating the role and estimating the situation & context. We also want to experiment with hard assignment and probabilistic assignment based estimation algorithms. This leads to the overall structure in Fig 3. The two inputs to each ‘pass-through switch’ node 7 are estimates of the same quantity (eg, the move values for the top right 7). In the full CAVIAR system the controller dynamically adjusts these switches based upon various performance feedback and other information. If instead we hardwire each switch node to pass through a given one of its inputs, we obtain eight distinct ‘systems’ which can produce the final output containing the estimates of all four attributes. (We
Figure 3: How various modules and implementations can be combined to generate a complete analysis in 8 distinct ways, where switches (?) pass through a selected input

assume move and role estimation are independent.) Clearly we would like to know how accuracy varies between these 'implementations' of the system.

There are two further sources of variability. Firstly, different kinds of deployment can have visual scenes, different camera details, and varieties of behaviour attributes, eg, surveillance of a shopping centre atrium vs web-cam based workplace analysis. Secondly, each module has settable module parameters which affect its output. Ideally, the parameters would be 'tuned' to compensate for the first type of variability. However, good system tuning (particularly with limited ground truth) is non-trivial, so we want to understand how performance varies with the degree of tuning of component modules, especially since low-level module performance may not relate linearly to high-level performance.

If each move module has 1 parameter quantised into 16 values, we have $16 \times 2$ ways of generating the move output. With the same assumptions on the role, a context module has $(16 \times 2)^2$ unique inputs. Thus there are $2 \times (16 \times 2)^2 = 2048$ systems to consider for optimal performance over our test data. We could use, eg, hill-climbing to attempt to find the maximum of this 'black-box function' without exhaustively mapping the space. However, because we want to understand the spectrum of behaviours rather just obtain a single number we choose here to exhaustively evaluate the possibilities.

Our fundamental representation is to associate with each individual within each frame a set of hypotheses $(\pi, move, role, situation, context)$, ie, a probability and values for the various attributes (some of which may be unset); this is allowed for by the XML representation in Fig 2 and is useful because hard assignment can be incorporated by represented by a single probability 1 entry, whilst input to a hard decision module takes the highest weighted hypothesis and resets its weight to 1. In practice we can take a few highest weighted entries without much error, reducing storage requirements. We now give very brief sketches of the estimation algorithms with more complete descriptions available in [13], using $T_i$ to denote module parameters which are going to be systematically varied.

Fitting moves and roles. We define some common notation: the images contain multiple numbered individuals with the bounding box record for individual $i$ in frame $t$ denoted $b_i[t]$. The vector $b_i[t] \cdot p$ denotes the estimated ground plane position for box $i$ at $t$ and the number of the currently considered frame is always $t$. Finally, we take the current velocity (estimated over a fixed temporal window $w$) as $v = b_i[t] \cdot p - b_i[t-w] \cdot p$.

- **LOGICAL MOVE.** This module only distinguishes between $inactive/active$ and $walking/running$ since simple rules were unable to distinguish further.
  1. move→ if $|v| < T_1$ then $\{(\frac{1}{2}, inactive), (\frac{1}{2}, active)\}$ else $\{(\frac{1}{2}, walking), (\frac{1}{2}, running)\}$

- **LOGICAL ROLE.** In rules 2 & 3 $signed_{vel} = \text{signum}(v \cdot e)/|v|$, where $e$ is a chosen vector along the direction out of the shop doorway.
  1. if $b_i[t'] \in browse_{areas}$ for all $t' \in [t-T_2, t]$ then role→$(1, browse)$
  2. if $b_i[t] \cdot p \in shop_{doorway}$ and $signed_{vel} > 0$ then role→$(1, shop-enter)$
  3. if $b_i[t] \cdot p \in shop_{doorway}$ and $signed_{vel} < 0$ then role→$(1, shop-exit)$
  4. if no other rule has triggered, then role→$(1, walker)$

These rules were found by guessing a functional form from training data; the motivation was to obtain a simple, fast module to compare with PROBROLE.

- **PROBMOVE & PROBROLE.** We want to approximate the generating distribution of a class from a training set. The *Kernel Density Estimator (KDE)* [12] estimates the density of observation $f$ for class $c$ from set of training points $\{f^c_i\}_{i=1}^N$ using

\[
p(f|c) = \frac{1}{N} \sum_{i=1}^N K(f, f^c_i; T_k)
\]

\[
K(x, y; T_k) = \max(1 - ||x-y||^2/T_k^2, 0)/Z
\]
(where $T_k$ is a parameter affecting the width of the local kernels and $Z$ is a normalisation factor). Providing kernel $K$ is roughly bell-shaped the exact form doesn’t particularly affect the accuracy; we use the Epanechnikov kernel [12] which has optimal asymptotic performance, a simple form allowing efficient implementation and converges to the true distribution as $N \to \infty, T_k \to 0$.

We use KDE for both the move and role. To obtain the feature vectors we apply a simple image-to-ground-plane transformation to the bounding boxes and then select a subset of effective values into an observation vector. In common with many other density estimators, KDE methods have two disadvantages: as they perform less implicit smoothing than other methods, eg GMMs, the estimate closely approximates the training set, even if this is non-representative. Secondly, the higher the dimension of the feature input space the more samples are needed for adequate estimates (‘the curse of dimensionality’). We found that for testing we could divide the ground truth into training and testing sets containing equal numbers of sequences and use 4 parameters to form the feature vectors in Eq 1. We use the position of the centre of the box projected onto the ground plane, $|v|$ and $\sqrt{\text{change in box area}}$ (both over a 32 frame window), applying empirical scale factors to each component to attain the same degree of variability over all axes.

Fitting against context graphs. As the ground truth satisfies the context models in Fig 2, we could find the correct context directly if we could assume perfect lower-level attributes. As this is unrealistic, we need matching techniques which are tolerant of errors. The first task is to convert lists of (move,role) pairs into situations (with probabilities). From the ground truth data we can tabulate (move,role,situation) triples to obtain function $f:\text{move} \times \text{role} \times \text{situation} \to R$ giving the probability of a situation given the move and role. We can use this to extend the set of hypotheses $\{(\pi_i, m_i, r_i)\}$ to a new set $\{(\pi_i \times f(m_i, r_i, s_i), m_i, r_i, s_i)\}$ which now incorporates the situations. (In the case of logical inputs we discard all but the highest entry.) We now ignore all but the probability and situation when matching the sequence against the context graphs.

- **LogicalContext.** Each context $c$ is considered in turn against the situation list:
  1. Run length compress the situations list wrt the situation node labels.
  2. Unwind the graph representation into a tree and apply interpretation tree matching (with wild-cards allowed) technique to find the least cost match [6].
  3. Take ‘probability’ of context $c$, with situations $s_{0:N}$, as $\propto \exp - \lambda \# \text{wild card frames}$.

- **ProbContext.** We generate probabilities for each context $c$ using observation probabilities $\pi_{\tau,c}$ and situation transition probabilities $p(s_{j}\mid s_{j-1})$ using a HMM algorithm:
  1. For each final situation $s_{\tau}=s$: find most likely situation assignment $s_{0:s_{\tau}}$ using the Viterbi recursion $p(s_{0:s_{\tau}}) = \pi_{\tau,c} \max_{s_{\tau-1}} p(s_{0:s_{\tau-1}}) p(s_{\tau}\mid s_{\tau-1})$ (see [15]).
  2. Choose most likely solution $s_{0:s_{\tau}}$ also satisfying the constraints on must-visit nodes.
  3. Take ‘probability’ of context $c$, with situations $s_{0:N}$, as $\propto \prod_{\tau=0}^{T} \pi_{\tau,c} p(s_{\tau}\mid s_{\tau-1})$.

4 Experiments

We used the CAVIAR system to run the behaviour interpretation algorithms over the 80 test sequences (using the module combinations shown in Fig 3) for a range of parameters for the move and role modules. Even with an optimised implementation, this took 37 hours. For space reasons we present a limited number of preliminary, untuned examples; more extensive data and analysis can now be found in [13]. (We abbreviate module names by initials, eg, PR is ProbRole.)

- **movement/role performance on France dataset.** Fig 4(a,b,d,e) show performance of the low-level modules as functions of parameters $T_i$ for the entire France dataset. (The ‘thresholding’ nature of logical rules explains the ‘steepness’ in a & b.) For (d) & (e) the solid line is proportional to the probability weight on the correct attributes and the dashed line is the proportion of times the highest probability attribute was correct. The graphs show that, except for low parameter values for the PM module the two measures are essentially the same.

- **situation/context performance wrt move/role.** Fig 4(c,f) show the situation output as function of the inputs to LM & LR and PM & PR and shows that, whilst smooth, the surface has many local maxima.

- **move/role performance on all datasets.** The top two rows of Fig 5 show the performance of LM, PM, LR & PR over the three datasets. Note all modules have different scores for the same parameter value in different datasets, and module PR clearly does not have a ‘dataset-independent’ optimal parameter value.

- **situation/context performance wrt move/role module parameters.** The bottom two rows of Fig 5 shows the correctness of the context (third row) and situation output of the PC module with respect to the parameters of the LR
Figure 4: For the France dataset: (a) LM (with parameter $T_1$); (b) LR (parameter $T_2$); (c) situation correctness from LC as a function of $T_1$ & $T_2$;(d) PM (parameter $T_3$); (e) PR (parameter $T_4$); (f) situation correctness from PC module as a function of $T_3$ & $T_4$ & PR modules, giving an indication of how sensitive the overall system performance at the given role parameter value is to the setting of the move module parameter.

To explain the format, the first graph on the third row has a horizontal axis corresponds to the LR module’s parameter value. For a complete network of modules from Fig 3 we also need a move module, so the solid line plots the correctness achieved on context when LR module is used in conjunction with the LM module, averaged over all possible parameter values for LM. The error bars show the standard deviation of this score wrt the LM parameters. Likewise, the dashed line and error bars shows performance when the PM module is used, again with the average and standard deviation wrt PMs parameter values. The plot shows that whilst on average the combination with LM gives better results than the PM, the standard deviation (uncertainty) for PM is so large that using a single parameter values for the move modules could lead to an erroneous plot suggesting PM is the better companion module. Indeed, there are large error bars on all four plots.

In general the performance is relatively poor, indicating the need for some fine-tuning of details (eg, better features for KDE); however these observations highlight the need to explore inter-module interactions when attempting to evaluate module performance.

5 Conclusions & further work

We developed a system to run empirical tests to gain understanding of the performance characteristics of a modular system for behaviour recognition, in particular relationships between correctness and the parameters of multiple modules; attempting to optimise module performance in isolation is can be confounded by interactions with other modules.

In future we hope to develop an intelligent sampling approach to make tractable analysis of much more complicated module ensembles. This will enable us to use the CAVIAR framework to characterise a wider set of vision algorithms from the vision community.

References

Figure 5: Top two rows: different relationships between parameters & performance of LM, PM, LR & PR for the datasets; bottom two rows: performance of context (3rd row) & situation (4th row) output from PC module as function of parameter for LR & PR modules.


Recognition of Action, Activity and Behaviour in the ActIPret Project

Kingsley Sage, Jonathan Howell and Hilary Buxton
Department of Informatics, University of Sussex, UK

Abstract

In this paper we focus on the task-based control of the selection of models for the interpretation of manipulation tasks and behaviour. These models are implemented as temporally structured Bayesian Belief Networks (BBNs) that can recognise constituent behaviour primitives. We present examples in the context of a scenario of learning to put a CD in a CD player. We then discuss how these ideas may be extended to more complex task scenarios and suggest future directions for this work.

1 Introduction

There is much current research interest in the development of frameworks for cognitive computer vision systems, for a review see [8]. Many of these frameworks draw inspiration from work on active (purposive) vision [1, 4, 5, 30, 31]. Task-based visual control is one useful element of the cognitive vision paradigm and is concerned with the application of a task relevance control structure to guide low level vision processes. The benefit of this approach is computational efficiency both in terms of the low level processes and subsequent scene interpretation. Processing is limited to only that data that is considered task relevant. Task-based visual control can therefore be thought of as data driven (bottom-up) processing limited in scope by task-based (top-down) control.

To achieve robust interpretation of activities within such frameworks, the interaction of visual attention, active camera behaviour, recognition, understanding, and knowledge from models, tasks, and context is required. The interactivity of these modules is an essential mechanism for disambiguating inherently uncertain information of visual processing. The ultimate framework architecture required should combine purposive (task/behaviour/context-driven) attention and reactive (data-driven, self evaluating) collection of evidence.

It is important here to distinguish between knowledge based/driver extensions to classical (non-cognitive) approaches to vision and task-based control. In the former, scene specific knowledge of object parametric properties (speed, size etc.) can be used to simplify scene interpretation or increase the effectiveness of tracking, but the vision task is still under constrained with a corresponding high computational cost in solving correspondence issues. Relevance is uniform across all parts of the scene purview. In contrast, task-based control seeks to extend computational efficiency by limiting the processing to only those objects/parts of the scene that have task-based relevance. Task relevance will evolve over time according to the requirements of different stages in the overall task.

For example, for a visually guided task such as ‘making a cup of tea’, initial task relevance might be identifying a kettle. So appropriate lower level vision operations are selected to find a kettle. Once a kettle has been found, attentional control might switch to finding a tap to fill the kettle with a corresponding new set of selected lower level operators. At each stage, lower level processing is limited to those functions that are task relevant. Relevance may encapsulate classification (what objects are we looking for), categorisation (what general sort of objects are we looking for), localisation (where might we find these objects) and other parameters.

An action is generally associated with second order changes in state (acceleration, contact) in contrast to an activity which refers to first order changes (long term movement, walking, handling, manipulation) [18]. An event can be seen as a separator between activities and behaviours, for instance, an object has been put on the table. We give a more specific meaning so that an action is something related to non-visual belief, for instance, ‘is the hand empty or holding something?’ and an action has a specific start and end point, for instance, ‘pick up an object’. Within this paper we refer to an action when meaning both actions and activities except when explicitly used separately. A behaviour can be seen as an aggregate of actions and activities and is closely related to a task.

Early approaches to building activity interpretation systems started with the pioneering work of Nagel [23], which has concentrated on refining analyses of traffic situations in a series of more and more sophisticated offline systems with explicit knowledge [13, 15]. Other early systems were developed in VITRA (Visual Translator) [16] and Esprit VIEWS projects [9]. These are all characterised by bottom-up techniques: vision provides object trajectories of cars (constrained to a ground-plane) and a static model representation, which after a number of pipeline stages, form conceptual descriptions. Later work in the Esprit PASSWORDS project and more recent surveillance work overcomes some of these limitations by providing top-down contextual expectations for tracking and behaviour evaluation using explicit logic formalisms [27, 33] and Bayesian belief networks [17, 18] to recognise the scenario in an on-line manner.

It is known that robust tracking of non-rigid objects such as human hands and bodies involved in a real-world activity is difficult due to rapid motion, occlusion and ambiguities in segmentation and model selection. This was partially addressed by the move to active vision and dynamic models for robust tracking using sophisticated Kalman filters, as exemplified by Blake and others [7]. These trackers have been specialised to allow the learning of complex
3 strands of research and development:

replayed at any time and location to many users using Augmented Reality equipment. Thus the project encompasses
machines and maintaining plant. The expert activities are interpreted and stored using natural language expressions
providing this functionality ActIPret will enable observation of experts executing intricate tasks such as repairing
into constituent behaviour elements, and on extracting the essential activities and their functional dependence. By
of people handling tools. The focus is on active observation and interpretation of activities, on parsing the sequences
Teaching and Education) [32] is to develop a Cognitive Vision methodology that interprets and records the activities
as in [2], aims at developing a virtual commentator which is able to translate visual information from video sequences
into a textual description, such as sign language interpretation and automated video annotation. In video annotation the
challenge is to recognize humans, object categories, materials and actions. In video annotation the
classification, even where no specific spatial structural analysis has actually taken place. This has proved important in future integrated cognitive vision systems such as ActIPret where we have both pre-attentive and attentive interplay between the objects of interaction and the activities themselves.

Recent research has seen a variety of cognitive vision projects funded under EU Framework 5: CogVis, for
example[24], uses an embodied mobile agent to interpret human actions and carry out a simple task such as fetching
objects. The Vampire project, for example[6], uses an active memory and retrieval system with head-mounted
camera to provide an augmented reality output for content-based retrieval and annotation of objects and events in the
user environment. CAVIAR, for example [12], develops a framework for context aware description of human activity
(originally aimed at public area surveillance and marketing data collection applications) using situation models to dy-
namically configure a set of visual processes for observing the entities and relations that define a situation. CogViSys,
as in [2], aims at developing a virtual commentator which is able to translate visual information from video sequences
into a textual description, such as sign language interpretation and automated video annotation. In video annotation the
tohas become possible due to advances in real-time tracking. For example, work on fire alarm systems [10] uses
motion of objects and their trajectories [10]. The GR is capable of detecting 4 different types of gesture: gross
motion over a certain Euclidean threshold (i.e. that hand objects are moving), purposeful motion of a hand
away from a nominal torso position (“reaching out”), purposeful motion of a hand towards a nominal torso position (“reaching in”) and purposeful motion of a hand at a fixed Euclidean radial distance from a nominal
torso position (“lateral motion”).

2 The ActIPret project

The objective of the EU IST ActIPret Project (Interpreting and Understanding Activities of Expert Operators for
Teaching and Education) [32] is to develop a Cognitive Vision methodology that interprets and records the activities
of people handling tools. The focus is on active observation and interpretation of activities, on parsing the sequences
into constituent behaviour elements, and on extracting the essential activities and their functional dependence. By
providing this functionality ActIPret will enable observation of experts executing intricate tasks such as repairing
machines and maintaining plant. The expert activities are interpreted and stored using natural language expressions
in an Activity Plan. The Activity Plan is an indexed manual in the form of 3-D reconstructed scenes, which can be
replayed at any time and location to many users using Augmented Reality equipment. Thus the project encompasses
3 strands of research and development:

- a generic framework for a distributed processing system that is intended to be reusable for future CV projects,
- the development of a set of computational components specifically for an ActIPret demonstrator, and
- the integration, testing and demonstration of this set of computational components within the CV framework on
  a distributed system to achieve the goals of the project for a set of demonstration scenarios.

The key components that are relevant to the following sections on activity and behaviour recognition are:

- Activity Reasoning Engine (ARE): The ARE is responsible for defining the visual task for the whole ActIPret
  system. It uses a high-level Control Policy to determine which lower-level components are required at any
time. It uses the data returned by the lower-level components to form a high-level natural language orientated
description of the scenario. This description forms the content of the Activity Plan.

- Gesture Recogniser (GR) / Hand Tracker (HT) combination: These two components provide information on
  hand objects and their trajectories [10]. The GR is capable of detecting 4 different types of gesture: gross
  motion over a certain Euclidean threshold (i.e. that hand objects are moving), purposeful motion of a hand
  away from a nominal torso position (“reaching out”), purposeful motion of a hand towards a nominal torso
  position (“reaching in”) and purposeful motion of a hand at a fixed Euclidean radial distance from a nominal
torso position (“lateral motion”).
2.1 Activity Plan

A system diagram for ActIPret is shown at Fig 1. ActIPret is a distributed system split into a number of components. Each of these components works in either a view dependent or view-independent system of co-ordinates. The lower-level vision components use 3-D world co-ordinates derived from calibrated stereo camera heads. These components are concerned with object detection and tracking. The higher-level components use symbolic representations to reason about objects as previously described. Higher-level components make service requests of the lower-level components to provide information about candidate objects. So, for example, the ARE might ask the ORG to establish whether objects are NEAR some reference target object. Lower-level components return data to the high-level components. The lower-level vision components only carry out processing as directed by the higher-level components, making the overall distributed system task-based in its focus. The ARE stores information about relevant objects in task-based scope at any time using data derived from the lower-level components. When these objects go out of scope, that information is deleted from within the ARE.

The following definitions are pertinent to the rest of our discussion on the Activity Plan:

- **Activity**: A continuous (or extended period) operation characterised by an “action verb” e.g. “holding”, “grasping”.
- **Action**: An operation with a discrete start and end point e.g. “grasp”, “hold”.
- **Event**: A second order operation characterised by a change in state with a causal link to actions and activities e.g. “dropped the CD” which impacts on the activity “holding”.

Activities are characterised as a set of internal beliefs such as “Hand-0 is holding CD-1”. The ARE maintains these internal beliefs and modifies them according to the detected actions. Actions are characterised by operator behaviours over a time period and are recognised using Dynamic Bayesian Belief Networks. It is these networks that are used to create the behaviour content for the our Activity Plan. In some ways, our Activity Plan might have been better named an Action Plan, since the start and end points are implicitly encoded (e.g. by knowing the time when objects were lost in defining the ‘pickup’ action). For ‘putting a CD in a CD player’, a typical single exemplar Activity Plan might be:
1. Hand-0 pressed Eject-button-1
2. Hand-0 picked-up CD-1 from an Undefined Location
3. Hand-0 put down CD-1 on CD-tray-1
4. Hand-0 pressed Play-button-1

Picking up and putting down have a location attached to the activity. ‘Undefined Location’ is used to specify anywhere not recognised as scenario-relevant, for example, on the desk. In this example, it is implied that pressing the Play Button also closes the CD-player. An equally valid Activity Plan might show the Eject Button being pressed a second time (closing the CD-tray) before the Play Button is pressed.

An Activity Plan is therefore a concise account of a scenario specifying the relevant objects and how they are acted upon. The motivation for building it (an expert demonstrating the scenario) is to allow comparison with the actions of a novice user (using the error between the stored expert plan and the plan primitives generated by the novice user to provide feedback to the novice user) and for reconstruction (we can use the plan to drive, for example, VR reconstruction).

3 Control Policy

3.1 Main control loop

As the ActIPret framework is a task driven system, no visual processing at all takes place until a high-level task is established by the Activity Reasoning Engine (ARE). In the expert mode illustrated here, the highest level task is to establish an Activity Plan. The highest level of reasoning within the ARE is called the Control Policy. This represents the highest and most abstract level of attentional control. The Control Policy is concerned with identifying relevant initial objects in the scene and with determining what type of behaviours it might be appropriate to look for. The top level control loop for the ARE can be summarised by the following pseudo-code:

Start a service HAND_INFO to look for moving hand objects anywhere in the scene
WHILE (HAND_INFO is running)
   HandCandidateSet = hand objects identified by HAND_INFO
   Maintain set of Visual Indexes for HandCandidateSet
   FOR ALL members y in HandCandidatesSet
      Request a service for GESTURE_RECOGNITION for object (y)
      Request a service for HAND_POSTURE_RECOGNITION for object (y)
      Request a service to determine objects for tuple (y, NEAR)
      Determine Visual Index internal variables for hand (y)
   END_FOR
END_WHILE

The service HAND_INFO is provided by the Gesture Recogniser (GR)/Hand Tracker (HT) pairing. The GR requests a service from the HT to provide data on all hand candidate objects based on a skin colour detection algorithm [3]. The HT provides this data to the GR which then uses a simple frame by frame measure of Euclidean distance between centroids to determine if any of these hand candidates are moving. Object labels for those that are (the ARE has no ability to process absolute positional data) is then returned to the ARE. The ARE then selects those candidates that are “interesting” and makes a request to the ORG to observe whether the selected hand objects are in relationships with other scene objects. The ORG uses information about the trajectories and posture of the hand objects to determine what 3-D space models need to be investigated (task based selection of 3-D ‘Spaces of Interest’). The service to generate hand posture data is generated by a Hand Posture Recogniser (HPR) and the services that generate data on spatial relationships are provided by the Object Relationship Generator (ORG).

For each object warranted worthy of investigation, the ARE creates a Visual Index (VI). A VI is simply a collection of data items and variables that relate to a specific hand object. These variables are used to encode internal beliefs about the state of a hand object that are relevant for task based control. Some of these beliefs are determined directly using information supplied by lower-level components (e.g. whether the hand is currently moving), some are inferred using internal knowledge acting on data returned from lower-level components (e.g. whether the hand is near an object whose categorisation suggests that it may be picked up) and some is inferred according to previous actions (e.g. whether the hand is full as a result of previously picking something up and the identity of the objects picked up). The VI includes:

- Hand reference ID: an ID tag common to other components in the distributed system
- Services list: Lists of handles for services requested from lower-level components but not yet satisfied. This is to ensure that data from lower-level components affects the correct VI when the data is returned to the ARE
- Boolean hand states: ‘hand-empty’ which is set when the hand is not believed to be holding anything, and a vector returned by the GR to show whether the hand is currently performing any of the functional gestures
• List of ‘holding’ objects: A list of reference IDs for objects that are believed to be currently in this hand (i.e. picked up)

• List of objects in specific relationships to this hand: The ARE can ask the ORG to watch for objects in specific spatio-temporal relationships (e.g. NEAR) with this hand object. These lists of objects are maintained on a per-hand basis in the VI

• List of current action BBNs: The BBNs currently being processed for this hand object (see later discussion)

The ARE uses the VI state data and a decision tree (shown in Fig 2) to generate action hypotheses (also stored in the VI as they are hand specific) for each selected hand object, which, when confirmed, are collected and recorded in the Activity Plan. Whilst the hand object remains in scope, these hypotheses are generated and updated regularly. When the hand object goes out of scope (i.e. disappears from view), the VI is deleted. The hypotheses are derived from instances of Dynamic Bayesian Belief Networks which define what spatial and temporal evidence is relevant and to be combined over time in a way that characterises the actions. An example of such a network is shown in the next section.

Figure 2: Predictive decision tree, based on Control Policy rules, for the selection of BBNs.

4 Bayesian Belief Networks

A Bayesian network [25] is a compact representation of the joint distribution over a set of variables. Each variable is represented by a single node and arcs between these nodes encode dependencies between these variables. The set of nodes and arcs is termed a graph or net. Bayesian Belief Networks are a widely studied method of inference and have been used in a diverse range of applications e.g. [18].

A commonly reported type of Bayesian network is the Directed Acyclic Graph (DAG). In a DAG the arcs operate only in one direction between any two nodes. This implies a causal type of dependence between connected nodes. Nodes in the graph are said to be conditionally independent of one another if they are not directly connected by an arc. In an acyclic graph, there is, at most, one path between any two nodes (i.e. there are no loops). For each node we can define a Conditional Probability Table (CPT) that defines the probability associated with that node depending on the value of its parent node. Each node may be either a parent node (its input is used in determining the probability of another child node) or a child node. Root nodes are defined as those that have no parents (i.e. their value is some quantity that is conditionally independent of all nodes in the graph and is due to some external influence. Leaf nodes are those that have no children. We can further classify nodes as evidence nodes and query nodes (those that we would like to infer probability values for). There are two separate aspects to the networks:
Continuous priors belief within Activity p(Lateral gesture) Continuous node p(Reach-out gesture) Continuous priors generated by the Gesture Recogniser

Continuous node p(X,reach-in) CPD specified by a 2-D Gaussian

Logit function can be approximated as:

<table>
<thead>
<tr>
<th>p(X is-empty)</th>
<th>p(Reach-in)</th>
<th>p(Mid-Point-State)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.75</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.75</td>
<td>1</td>
</tr>
</tbody>
</table>

Discrete node derived from internal belief within Activity Reasoning Engine

Continuous priors generated by the Gesture Recogniser

Continuous node p(X=hand) Continuous prior generated by the Gesture Recogniser

Figure 3: Template BBN for the pickup action.

- Calculating probabilities for query nodes at a static point in time (using the fixed structure of the graph and a evidence to query propagation, or inference, procedure.
- Propagating nodes values from one time-step to the next (“rolling out” the model over time). This involves developing a set of temporal update rules for each node.

In ActIPret, there are 3 template BBNs (for ‘pickup’, ‘putdown’ and ‘press a button’). These template BBNs are generic in the sense that they do not distinguish whether we are interacting with CDs or any other object, provided that the lower-level components can report the data necessary to drive the decision processes within the ARE. At present, the template BBNs are hand coded and the learning task is concerned with the acquisition of the Activity Plan, rather than the acquisition of BBN structure and/or parameters. There is an implicit assumption that the range of action BBNs are sufficient to document the scenario.

4.1 The ‘Pick-up’ Activity BBN

The template BBN for the pickup action is shown in Figure 3. In the pickup network, there are 10 root (evidence nodes) and the query node is \( p(X, \text{picked-up objects } Y) \) where \( X \) is a candidate hand object that has been previously determined to be moving. The network is constructed such that it can cope with continuous and discrete evidence values. The root nodes obtain their values either from data generated by lower-level components or from VI embedded logic acting on data generated by lower-level components. \( p(X = \text{hand}) \) is a continuous prior generated by the HT representing the probability that the candidate object is indeed a hand (the HT can use probabilistic techniques to determine whether the skin coloured object is a hand or some other skin-coloured object such as a head). \( p(\text{Reachout gesture}) \) is a continuous prior generated by the GR representing the instantaneous probability that the hand candidate is performing a purposeful gesture away from the nominal torso position. \( p(X, \text{Reachout}) \) is a continuous node with 2 continuous parents. Its Conditional Probability Table (CPT) is represented by a 2D Gaussian function \( N(\mu_X, \mu_{\text{reachout}}, \mu_{\text{lateral}}; \Sigma_X, \Sigma_{\text{reachout}}, \Sigma_{\text{lateral}}) \) where \( \mu_X = \mu_{\text{reachout}} = \mu_{\text{lateral}} = 1 \) and the function is only defined for values of \( X \) and \( \text{Reachout} \) in the interval \([0, 1]\). The output probability is calculated using the standard formula for multinomial Gaussians:

\[
p(X) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(X-\mu)^T \Sigma^{-1} (X-\mu)}
\]

where \( X \) is the vector of inputs, \( \mu \) is a the vector of corresponding mean values and \( \Sigma \) is the covariance matrix. In this case, \( N = 2 \).
\( p(\text{MidPointState}) \) is a continuous node with 1 discrete and 1 continuous parents. This node is a multinomial logit (softmax) function where the parameter set for the function is determined by the state of the discrete input node. The logit function acts as a soft thresholding operator and is defined as:

\[
p(Q = i | X = x) = \frac{\exp(w_i^T x + b_i)}{\sum_{j=1}^{|Q|} \exp(w_j^T x + b_j)}
\]

where \( Q \) is the set of all possible discrete inputs, \( w \) and \( b \) are the logit’s parameters and \( X \) is the vector of continuous valued inputs. \( w \) (\(|X| \times |Q|\) matrix) and \( b \) (length \(|Q|\) vector) determine the decision boundaries between classes \( Q \). In the special case that \( Q \) is binary, the logit function reduces to the logistic (sigmoid) function. The discrete node input is then used to select between parameters sets and vary the threshold setting. In this case, the CPT for the node can be approximated by the table shown in Figure 3. The Mid-Point-State node is also important as it represents a coupling between the upper part of the model and the lower part in temporal terms. This coupling is key to allowing a single topology model to model an action with discrete functional elements as a function of time.

The lower \( p(X = \text{Hand}) \) is identical to the previous one in value terms except that it occurs at a later point in time. Thus it can be regarded as conditionally independent from the upper instance of the same variable. \( p(\text{Reach In Gesture}) \) is similar to \( p(\text{ReachOut Gesture}) \) and \( p(X, \text{Reach In}) \) is a 3D Gaussian function since that we are only interested in it once the value of the Mid-Point-State node has achieved some significant value. The \( p(\text{Objects Y Lost Near X}) \) node is a discrete node derived from an internal belief value from the appropriate Visual Index. This internal belief is derived from data received from the ORG. If we receive any non-zero length list of such objects at a given point in time, then we have candidate objects that we might have picked up. Our inference is that the reason that these objects are “lost” (i.e. not where they were to be found previously) is that the hand has picked them up. The query node \( p(X, \text{Picked Up Objects Y}) \) is another multinomial logit function such that the threshold for “switching on” the output of the node is modulated by the discrete node \( p(\text{Objects Y Lost Near X}) \).

Supporting evidence for some root nodes does not need to be generated until that node is able to have an influence on the query node. For example, we do not need to know whether objects have been lost near the hand object until \( p(X, \text{Reach In}) \) has exceeded some threshold value. Thus we can use the output probability values from some nodes to trigger appropriate services or queries to establish data values for other nodes. This mode of generating services/data requests is referred to as attentive querying.

As the inference procedure is strictly from evidence to a single query node in a DAG the we can use a simple forward chaining procedure rather than needing to resort to the junction tree type inference procedure required of the more general case.

To demonstrate ActIPret working on on real data, we ran the framework on an image sequence of a scenario of ‘putting a CD in a CD-player’: a subject opening a CD-player picking up a CD, putting it on the open CD-player, then closing it. Six images taken from the image sequence are shown in Figure 4. Figure 5 shows a trace of values for the nodes in an example pickup Bayesian Belief Network over a 5 second period (10Hz data rate across all nodes). The MidPoint state first switches on just after 1.5s. The MidPoint state is used to temporally couple sub-components of the action. Actions may be performed at different speeds by different subjects. The MidPoint state uses a temporal decay to represent that temporally close sub-components are most indicative of the action, and more temporally remote sub-components are increasingly less probable. When its parent node values start to fall away at 2.5s, the node value starts to decay. The concept is “confirmed” at the moment the MakeConceptState logit node turns on around 4.75s when the ObjectsLost node flips “on”.

Figure 4: Six images from the sequence P0020 for ‘picking up the CD scenario’
5 Conclusions and future work

We have shown how temporally structured BBNs can be used to model actions and derive an Activity Plan by visual inference in our distributed task driven ActIPret system. Such Activity Plans can be learned from an expert user in an ‘expert’ mode and then used to diagnose the performance of a new user in a ‘tutor’ mode. Our BBN models are used to combine data from disparate sources over a period of time and thus provide compact and structurally uniform representations of the constituent actions and have been used to build a real world real time system.

However, at present, our approach requires that the prototypical BBN models for each class of action are hand crafted. In the future, we would like to be able to automatically learn appropriate temporally structured BBN structures and a number of approaches are possible. There are two separate problems in learning a relevant BBN model for ActIPret. First, we need to learn the graphical structure of connected nodes, both root nodes (those derived from available information sources within the distributed system) and abstract internal nodes that combine such data. Data sources will be a mixture of discrete and continuous types. Second, we have to learn a set of temporal update rules to enable us to roll out the model structure over time. Then we have a further complication in that the distributed system that generates data that we can use for learning is task driven. Data is only generated by components within the distributed system when specifically requested. But how can we know what data is required to learn a BBN for a specific class of action?

Learning a prototypical BBN for a specific class of action would need to be performed in a separate learning phase. To avoid the further added complexity of learning with incomplete data, it would be necessary for the distributed system to switch on all of the system components (a control policy of “global exploration”) and allow them to generate data in response to the visual input. Learning would then take the form of multiple examples of individual actions.

Acknowledgements

This work is supported by the EU-Project ActIPret under grant IST-2001-32184.

References


