# ABSTRACT

Title of dissertation: IMPROVING EFFICIENCY AND SCALABILITY IN VISUAL SURVEILLANCE APPLICATIONS

Radu Mihai Dondera
Doctor of Philosophy, 2013

Dissertation directed by: Professor Larry S. Davis
Department of Computer Science

We present four contributions to visual surveillance: (a) an action recognition method based on the characteristics of human motion in image space; (b) a study of the strengths of five regression techniques for monocular pose estimation that highlights the advantages of kernel PLS; (c) a learning-based method for detecting objects carried by humans requiring minimal annotation; (d) an interactive video segmentation system that reduces supervision by using occlusion and long term spatio-temporal structure information.

We propose a representation for human actions that is based solely on motion information and that leverages the characteristics of human movement in the image space. The representation is best suited to visual surveillance settings in which the actions of interest are highly constrained, but also works on more general problems if the actions are ballistic in nature. Our computationally efficient representation achieves good recognition performance on both a commonly used action recognition dataset and on a dataset we collected to simulate a checkout counter.

We study discriminative methods for 3D human pose estimation from single im-

ages, which build a map from image features to pose. The main difficulty with these methods is the insufficiency of training data due to the high dimensionality of the pose space. However, real datasets can be augmented with data from character animation software, so the scalability of existing approaches becomes important. We argue that Kernel Partial Least Squares approximates Gaussian Process regression robustly, enabling the use of larger datasets, and we show in experiments that kPLS outperforms two state-of-the-art methods based on GP.

The high variability in the appearance of carried objects suggests using their relation to the human silhouette to detect them. We adopt a generate-and-test approach that produces candidate regions from protrusion, color contrast and occlusion boundary cues and then filters them with a kernel SVM classifier on context features. Our method exceeds state of the art accuracy and has good generalization capability. We also propose a Multiple Instance Learning framework for the classifier that reduces annotation effort by two orders of magnitude while maintaining comparable accuracy.

Finally, we present an interactive video segmentation system that trades off a small amount of segmentation quality for significantly less supervision than necessary in systems in the literature. While applications like video editing could not directly use the output of our system, reasoning about the trajectories of objects in a scene or learning coarse appearance models is still possible. The unsupervised segmentation component at the base of our system effectively employs occlusion boundary cues and achieves competitive results on an unsupervised segmentation dataset. On videos used to evaluate interactive methods, our system requires less interaction time than others, does not rely on appearance information and can extract multiple objects at the same time.

# IMPROVING EFFICIENCY AND SCALABILITY IN VISUAL SURVEILLANCE APPLICATIONS

by

Radu Mihai Dondera

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:
Professor Larry S. Davis, Chair/Advisor
Professor Rama Chellappa
Dr. David Doermann
Professor Amitabh Varshney
Professor Radu Balan, Dean's Representative

# Dedication

To Susan.

# Acknowledgments

I thank my advisor, Professor Larry Davis, for continuously offering me his support throughout my doctoral studies. I am also grateful to other faculty at the University of Maryland. Doctor David Doermann has provided much useful advice during my first year of work as a researcher in Computer Vision. Professor David Jacobs taught very insightful Computer Vision courses and shared his great expertise. Many interesting research perspectives were described by Professor Yiannis Aloimonos in his courses. Professors Radu Balan, Rama Chellappa and Amitabh Varshney found time in their busy schedules to serve as dissertation committee members.

Among my Computer Vision peers, a special acknowledgment goes out to Ryan Farrell, Behjat Siddiquie, Leonardo Claudino, Jayant Kumar, Anne Jorstad, Murad Al Haj, William Schwartz and Aniruddha Kembhavi. You have made graduate school a better experience for me. I also thank Vlad Morariu for supporting the carried object detection and interactive video segmentation work.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1: Introduction

Visual surveillance is an application field of computer vision in which fundamental questions must be answered efficiently and unequivocally: what action is a human performing? What pose is the human in? What object(s) does the human transport? What is the spatio-temporal support of the entities in the scene? This thesis explores the four questions via efficient and scalable solutions that offer additional insight on the structure of the problems.

One of the main challenges in human action recognition is action representation. Action recognition is far from being a solved problem and there are many settings in which it is possible to incorporate application-specific knowledge to improve recognition in terms of accuracy or computational efficiency. One such setting is the visual surveillance task of understanding human behavior at the checkout counter of a department store: the actions of interest are highly constrained and redundant in terms of image motion. We propose an action representation that is based solely on motion information and that leverages the characteristics of human movement in the image space. To build this representation, one does not need to estimate human pose, track humans, extract human silhouettes, or even find space-time interest points, operations which can be problematic under human appearance variation, occlusion and background clutter. Our computation-

ally efficient motion descriptor achieves good recognition performance on both a commonly used action recognition dataset and on a dataset we collected that contains ballistic movements.

Inferring the 3D pose of a human from a single image is an ill-posed problem, but restricted to settings in which the camera has similar intrinsic parameters and it observes the human from similar viewpoints, it becomes well defined. If a body model is available, it is intuitive to estimate pose by adjusting pose parameters until the difference between the generated image and the actual image becomes zero – essentially adopting a generative approach. We study discriminative methods, which follow the opposite path of building a direct map from image features to pose, and for which the main issue is insufficient training data, owing to the high dimensionality of the pose space. As datasets can be augmented with realistic data from character animation software, it is important to know how the existing approaches scale with the size of the training set. Gaussian Process (GP) regression forms the basis of a few state-of-the-art methods for pose estimation and we argue that Kernel Partial Least Squares (kPLS) regression approximates GP regression robustly, reducing the computational complexity of training and enabling the use of larger datasets. In our experiments, kPLS regression outperforms two state-of-the-art methods based on GP regression on a dataset with four activities and close to 100,000 instances.

An increasingly stringent visual surveillance problem is detecting objects carried by people. The extreme variability in the appearance of the objects people can carry makes it doubtful that one can learn an appearance model for them, and suggests using their relation to the human silhouette instead. We adopt a generate-and-test approach in which candidate image regions are produced based on protrusion, color contrast and occlusion

boundary cues and are then filtered with a kernel SVM classifier. We train the classifier in a Multiple Instance Learning framework that reduces annotation effort by two orders of magnitude and incorporates cue-specific information in an extension of the classic miSVM framework. With full supervision, our method exceeds the accuracy of the state of the art and has good generalization capability; in the weakly supervised setting, our extension improves accuracy over miSVM at negligible additional computational cost.

While fully automated visual surveillance is the main goal, methods that involve a human operator are also important. Video segmentation can be a viable alternative to tracking when a target is not easy to distinguish visually from the background, and it can also be used to learn appearance, shape or trajectory models for objects. Since it is often the case that unsupervised video segmentation lacks information to focus on the object of interest, the goal becomes to minimize the number of constraints added in an interactive video segmentation system. We present a system that trades off some amount of segmentation quality for significantly less guidance than necessary in systems in the literature. The unsupervised segmentation component that our system is built on effectively employs occlusion boundary cues and achieves competitive results on an unsupervised segmentation dataset. On videos used to evaluate interactive methods, our system requires less interaction time than others, does not rely on appearance information and can extract multiple objects at the same time.

# Chapter 2: Action Recognition Based on the Characteristics of Human Movement in Image Space

## 2.1 Overview

We present a motion descriptor for human action recognition where appearance and shape information are unreliable. Unlike other motion-based approaches, we leverage image characteristics specific to human movement to achieve better robustness and lower computational cost. Drawing on recent work on motion recognition with ballistic dynamics, an action is modeled as a series of short correlated linear movements and represented with a probability density function over motion vector data. We are targeting common human actions composed of ballistic movements, and our descriptor can handle both short actions (e.g. reaching with the hand) and long actions with events at relatively stable time offsets (e.g. walking). The proposed descriptor is used for both classification and detection of action instances, in a nearest-neighbor framework. We evaluate the descriptor on the KTH action database and obtain a recognition rate of 90% in a relevant test setting, comparable to the state-of-the-art approaches that use other cues in addition to motion. We also acquired a database of actions with slight occlusion and a human actor manipulating objects of various shapes and appearances. This database makes the use of appearance

and shape information problematic, but we obtain a recognition rate of 95%. Our work demonstrates that human movement has distinctive patterns, and that these patterns can be used effectively for action recognition.

## 2.2 Introduction

Human action recognition is an active field of computer vision research with applications to visual surveillance, human computer interaction and video indexing. One of the main challenges in action recognition is action representation. Previous work has investigated the use of appearance, shape, motion and sequencing information and their combinations. Although state of the art methods [6] [8] [7] achieve near perfect results on standard databases [9], action recognition is still challenging for real-world data.

Understanding human behavior at a checkout counter of a department store is a challenging application and current statistics on revenue loss due to illicit cashier actions [27] make it an important problem. It is hard because the appearance and shape of people can vary considerably: customers can wear different clothes, have different heights and body types. Moreover, as customers push shopping carts, cashiers stand behind counters, and large items are manipulated, occlusions and cluttered background frequently occur. Action recognition approaches that track humans, estimate pose, extract silhouettes, find space-time interest points or employ any cues other than motion are therefore problematic. Since actions relevant to this context have distinctive movement patterns, a motion-based approach can effectively compensate for missing and noisy motion vectors through partial matching. Typical motion-based representations compute optical flow over a video

volume of interest and classify the action in the volume without assuming a particular model for the resulting set of motion vectors. But ignoring the characteristics of human movement in these and other motion-based approaches limits performance in terms of both robustness and computational cost.

Efros et al. [2] track human actors in a video where an action is defined by optical flow for each of the actor-centered bounding boxes in a time interval. The approach is time consuming because it computes similarity using the cross-correlation score and thus examines each pair of corresponding frames in training and testing video sequences. Fathi and Mori [4] also work on carved volumes obtained through tracking, but increase efficiency by using a two level boosting scheme with weak classifiers defined on small and medium sized cuboids. For both approaches, occlusions, background changes or lighting differences occurring for long periods of time can lead to misalignment of the bounding box and substantially reduce performance. Ke et al. [3] use a one level boosting scheme with weak classifiers on arbitrary sized cuboids (volumetric features) and do not require bounding boxes. However, they scan entire videos with a query video volume and during training they select relevant volumetric features from a very large number of candidates, even for a query volume of short duration (1 million for 40 frames, 25 fps).

We propose a motion-based action representation loosely based on psycho-kinesiological models of human movement that achieves high robustness to appearance and shape variation, handles both short and long actions and has very low computational requirements. The representation is suitable for common human actions composed of ballistic movements, which create linear image patterns that can be efficiently leveraged to detect and classify actions. While most other motion-based approaches require initial processing

involving appearance and/or shape, our action recognition approach relies on movement only, offering advantages in certain recognition scenarios and potential for extension by incorporating appearance and shape information.

### 2.2.1 Related Work

In early work, Bregler [19] formulated the action recognition problem as grouping pixels based on color, motion and time continuity and then classfiying actions based on sequences of atomic group motions. Bobick and Davis [10] considered the shape and motion of the human body during actions, assuming either stationary background or reliable background subtraction. Restricting the recognition context, Cutler and Davis [11] specifically targeted periodic actions. In a broader context, Parameswaran and Chellappa [22] focused on viewpoint invariant action recognition.

A class of approaches has been built on the concept of space-time interest points, which capture distinctive local events in videos [13] [12]. 3D neighborhood descriptors of interest points detected in training videos are clustered and actions represented in terms of resulting cluster representatives (visual words) [9] [14]. In recent work, Mikolajczyk and Uemura [7] detected space (image) interest points and matched them, based on appearance descriptors and velocity, to a learned motion-appearance vocabulary of interest points. The consistency of appearance and motion plays a major role in both action detection and classification, allowing them to use interest points from only a small number of frames. Relying on consistency is typical of interest points approaches, which become problematic in settings with appearance variation, background clutter and extraneous ob-

jects, especially for non-repetitive actions.

A complementary class of approaches was designed to work with dense video volume information, either optical flow [2] or intensity values directly [6]. Other approaches used boosting schemes. Ke et al. [3] and Fathi and Mori [4] computed optical flow and represented video volumes with arbitrary size cuboids and, respectively, small fixed size cuboids containing salient points for motion. Drawing on neurobiological models of perception, Jhuang et al. [15] proposed a hierarchy of motion features and were able to reduce the number of features at the top level to hundreds for video sequences a few seconds long.

In model-based motion recognition work, Fablet and Bouthemy [18] employed low level information derived from the optical flow equation, and captured co-occurrence statistics with temporal multiscale Gibbs models. High level information was used by Song et al. [16], who modeled the positions and velocities of vertices of a human body graph detected in a sequence of frames. More recently, Ali et al. [17] represented actions with chaos theory invariants extracted from trajectories of six semantically relevant points on the human body (e.g., "head"). The latter two approaches assume that points with clear semantics are detected and tracked throughout the entire video, which is difficult to achieve in practice. In the related field of motion segmentation, a number of approaches clustered motion vectors to obtain meaningful tracks [23] [24]. Jiang and Martin [25] detected actions by matching tracks of pixels on the human body contour. In the absence of high level information, clustering may produce inconsistent partitions (very different for slightly different action instances) and track matching is computationally difficult.

Specifically targeting the same physical setting as our work (checkout counters),

Fan et al. [27] proposed a framework for recognition of repetitive sequential actions. Action primitives (e.g. cashier picks up an item) were efficiently integrated into sequences of actions (e.g. cashier picks up, scans, puts down an item) through a Viterbi-like algorithm leveraging spatio-temporal and sequencing constraints. Action primitive detection is the low level of the framework and is implemented mainly through heuristics related to physical constraints. This is in line with other work in recognition of composite actions, which focused on the high level of the framework by attempting to support parallel execution, partial ordering of actions and robustness to detection errors [28] [29].

Recently, Vitaladevuni et al. [1] proposed an action recognition framework based on ballistic dynamics. Inspired by psychological studies showing the units of movement planning are ballistic in nature (e.g. reach), they segmented videos into time intervals with semantically relevant movements of the whole body in one direction (e.g. reach for object on floor). Motion History Images [10] were computed on segments of both training and test videos and then matched using dynamic time warping, resulting in better recognition performance than Motion History Images on entire videos. We draw on [1] for the idea of linear movements, central to our representation.

### 2.2.2 Approach

The underlying assumption of our representation is that a human action is composed of a set of short and small correlated linear movements ($< 1$ s, tens of pixels). Although actions with multiple ballistic movements can be seen as having multiple such sets (one per ballistic movement), we represent the entire set without temporally segmenting the

video, in contrast to [1].

Given a video volume, we compute sparse optical flow [5] between consecutive frames, transform the directions and positions of the resulting motion vectors into a position-augmented Hough space, and compute a probability density function over vector coordinates in this space. We use the Bhattacharyya coefficient of two pdf's as the similarity measure in a k-NN framework for action detection and classification. The proposed motion descriptor can function as execution rate dependent or independent; however, it assumes the action events occur at relatively stable time offsets. Our descriptor recognizes actions performed in the same image location, but it can also be made location independent through a simple scheme to estimate the displacement between two similar action instances. The descriptor is viewpoint dependent, but in the checkout counter scenario, the camera is fixed and the number of possible postures for each action of interest is small and manageable in a k-NN framework with a smoothly varying similarity measure.

## 2.3 Motion Descriptor

### 2.3.1 Motion Profiles

Our assumption that "human actions consist of small and short correlated linear movements" implies that (1) the motion vectors will lie on a small number of image lines, (2) the motion vectors that lie approximately on the same image line will form groups spanning small image regions and small time intervals and (3) groups will be correlated in terms of image regions and time intervals. We represent an action with a mixture of Gaussians, each modeling the image coordinates and times of motion vectors in a group,

and we call this representation a motion profile. We leverage the linearity hypothesis by transforming motion vectors into an augmented Hough space that enables efficient use of the mixture model. The following subsections present the steps involved in computing motion profiles.

#### 2.3.1.1 Sparse Optical Flow

In the context of our assumptions on human movement, accurate motion information is more important than dense information, so we track KLT (Kanade-Lucas-Tomasi) feature points [20] from frame to frame and process the resulting tracks to reduce noise. Static feature points are eliminated with a threshold on velocity magnitudes (e.g. 1 pixel/frame) and short movements (most often occurring as background noise confusing the KLT tracker) are eliminated with a threshold on track duration (e.g. 5 frames). We subsequently discard track information and what remains is an unordered set of motion vectors given by origin, velocity and time (video frame index relative to a base frame): $\{(u_1, v_1, du_1, dv_1, t_1), \ldots, (u_n, v_n, du_n, dv_n, t_n)\}$.

Modeling motion vectors with a mixture of Gaussians at this point requires some form of partitioning, but general purpose frameworks (e.g. Expectation Maximization, Mean Shift) do not produce partitions consistent across action instances. We solve this problem by a transformation step that allows us to solve the partitioning problem with a sliding window technique.

#### 2.3.1.2 Augmented Hough Space

The Hough space [21] can be used to represent the image lines along which movement occurs. To completely encode direction and position information for motion vectors, we augment the Hough space $(\rho, \theta)$ by attaching a 1D coordinate system $d$ to each line. The origin of that coordinate system is the projection of the image origin on the line and the positive direction is left of the perpendicular (see Figure 2.1a). For each motion vector, we drop magnitude information and express direction and position by $(\rho, \theta, d)$ according to the equations

$$\alpha = atan2(dv, du) \tag{2.1}$$

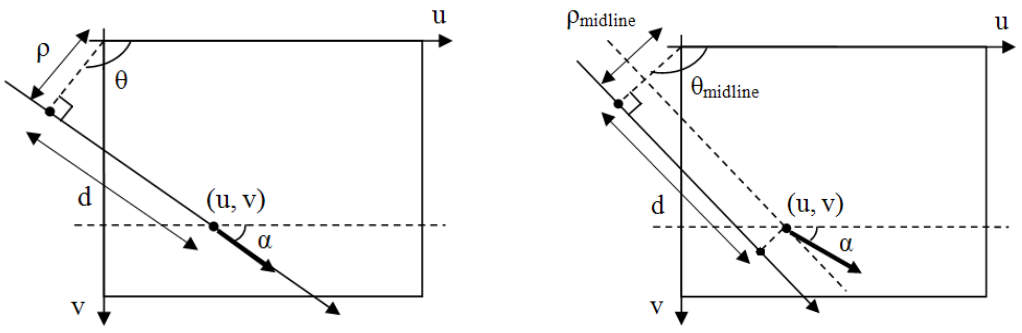
$$\rho = -usin\alpha + vcos\alpha \tag{2.2}$$

$$\theta = \alpha + \frac{\pi}{2} \tag{2.3}$$

$$d = ucos\alpha + vsin\alpha \tag{2.4}$$

$\theta$ is normalized to the range $[0, 2\pi]$ and $\rho$ can be positive or negative. The set of motion vectors becomes $\{(\rho_1, \theta_1, d_1, t_1), \ldots, (\rho_n, \theta_n, d_n, t_n)\}$.

We discretize the augmented Hough space (cell size e.g. 10 pixels and 5 degrees) and approximate the motion vectors in cell $(i, j)$ as belonging to the cell midline, $(\rho_{midline} = (i + 0.5) \times (\rho\text{-}cell\text{-}size), \theta_{midline} = (j + 0.5) \times (\theta\text{-}cell\text{-}size))$. $i$'s can be negative because $\rho$'s can be negative, but we offset them so all values are positive. The $d$ value is replaced by the $d$ of the projection of the motion vector origin on the cell midline (see Figure 2.1b). At this point, the set of motion vectors in a video volume is stored as




(a) (b)

Figure 2.1: (a) Coordinates of a motion vector in augmented Hough space. The coordinate system of a line is defined using the perpendicular from the image origin. (b) Coordinates in the discretized version of the augmented Hough space.

$\{(i_1, j_1, d_1, t_1), \ldots, (i_n, j_n, d_n, t_n)\}$.

#### 2.3.1.3 Motion Vector Groups

The visualization of $(d, t)$ values in a Hough space cell $(i, j)$ shows clear separation into groups (see Figure 2.2), confirming the existence of short linear movements. Since the groups typically span short time intervals, we use a sliding window to extract groups from a cell. We order vectors in a cell by time and scan the list with a small window (e.g. 0.5 s), counting the number of motion vectors inside, and detecting peaks. Peaks with too few motion vectors (e.g. $< 3$) are considered noise and eliminated, and groups are formed with the vectors in the windows of the remaining peaks. Groups are included in one cell only and cells may contain zero, one or multiple groups. The partitioning may be inadequate when two groups are too close (i.e. within 0.5 s of each other) or a group splits into two or more small groups in adjacent Hough space cells, but we observed the

procedure works well for the end goal of action recognition.

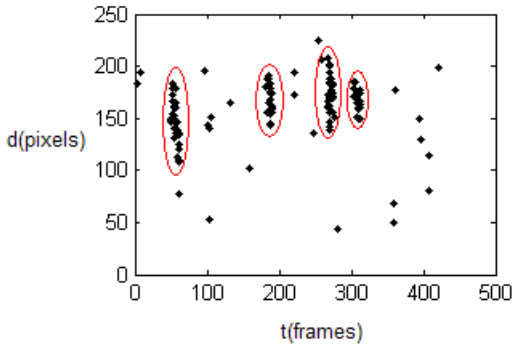



Figure 2.2: Typical $(d, t)$ values of motion vectors in a Hough space cell, for a 500 frame video volume (25 fps). Our partitioning procedure produces the four highlighted groups.

#### 2.3.1.4 Mixture Model

We represent an action's motion vectors with a mixture of bivariate Gaussians, each modeling the $(d, t)$ values in a group of motion vectors. Body parts may have different numbers of feature points on them (due to different body size, clothing, illumination etc) and we weigh Gaussians equally to reduce the influence of appearance and shape. The probability density function is

$$p(\rho, \theta, d, t) = \frac{1}{n} \sum_{(\rho,\theta) \in cell(i)} N_i(d, t) \tag{2.5}$$

where $n$ is the number of groups and $N_i$ is the bivariate Gaussian for group $i$. The summation is over the groups in the cell containing $(\rho, \theta)$.

If we stipulate that a Gaussian is zero outside the cell of its group, the probability

density function becomes

$$p(\rho, \theta, d, t) = \frac{1}{n} \sum_i N_i(d, t) \tag{2.6}$$

(summation over all groups). We observe that groups in the same cell are usually far away from each other in time, therefore there is no overlap between Gaussians in the same cell, so at any $(\rho, \theta, d, t)$ point all terms of the sum in Equation 2.6 are negligible or zero except for at most one.

Our representation maintains a Hough space cell index and $(d, t)$ averages and covariance matrices for each group. The averages and covariances are efficiently computed via an integral video structure [3] that stores for each cell the number of motion vectors and the sum of their $d$, $t$, $d^2$, $t^2$, $dt$ values. The values of these statistics for arbitrary time intervals are computed in $O(1)$ and so the averages and covariances are also computed in $O(1)$.

Although our approach is **largely inspired** by the use of the Hough transform in line detection, it is **significantly different**. In line detection schemes, edge pixels vote for cells and local maxima are the detection results, while our approach **does not vote** for cells, but instead maintains **all non-empty cells**. Also, note there are no assumptions about the numbers of groups in cells and the mixture model is built from the groups discovered in data, rather than by learning the parameters of a fixed cell structure. Finally, even though it might seem that the $d$ values in a group would depend linearly on $t$ values, this is not the case. The points on body parts slowly change direction and their trajectories are multiple connected small segments (best modeled by Gaussians), rather than one large segment (best modeled by a uniform distribution).

### 2.3.2 Motion Profile Similarity

To compute similarity between motion profiles, we use the Bhattacharyya coefficient, defined as

$$BC(p_1, p_2) = \int_X \sqrt{p_1(x)p_2(x)}dx \tag{2.7}$$

for two probability density functions $p_1$ and $p_2$ over X. We chose the Bhattacharyya coefficient as the similarity measure because of the simplifications it enables in the final expression. Also, the Bhattacharyya coefficient can handle "spiky" distributions like the ones used here, for which other similarity metrics are not suitable (e.g. the symmetric KL divergence would equal infinity).

For mixtures of equally weighted Gaussians,

$$BC(p_1, p_2) = \int_X \sqrt{\frac{1}{n_1}\sum_i N_{1,i}(x) \;\; \frac{1}{n_2}\sum_j N_{2,j}(x)}dx \tag{2.8}$$

where $n_1$ and $n_2$ are the numbers of groups in the profiles and $N_{1,i}$ and $N_{2,j}$ are the Gaussians for groups $i$ and $j$, respectively.

As observed in Section 2.3.1.4, in a motion profile there is no overlap between Gaussians in the same cell or between Gaussians in different cells. Consequently, the similarity computation for two motion profiles can be simplified to summing Bhattacharyya coefficients for pairs of Gaussians in the same cell.

$$BC(p_1, p_2) = \frac{1}{\sqrt{n_1 n_2}} \sum_{cell(i)=cell(j)} BC(N_{1,i}, N_{2,j}) \tag{2.9}$$

Gaussians in a cell of one motion profile may be better matched with Gaussians from neighboring cells in the other motion profile. This is not reflected in Equation 2.9, but we alleviate the problem with a multiresolution technique (described below).

The Bhattacharyya coefficient for two Gaussians reflects the closeness of means and the similarity of covariances in two distinct factors:

$$BC(N_1, N_2) = \exp\left(-\tfrac{1}{8}(\mu_1 - \mu_2)^t \Sigma_{12}^{-1}(\mu_1 - \mu_2)\right) \sqrt{\frac{\sqrt{\det \Sigma_1 \det \Sigma_2}}{\det \Sigma_{12}}} \tag{2.10}$$

where $N_1 \sim N(\mu_1; \Sigma_1)$, $N_2 \sim N(\mu_2; \Sigma_2)$ and $\Sigma_{12} = \frac{1}{2}(\Sigma_1 + \Sigma_2)$.

Since the number of groups in a motion profile is typically on the order of hundreds, naive evaluation of the sum in Equation 2.9 would be computationally expensive ($O(n_1 n_2)$). We use a reverse lookup table, mapping cells to groups belonging to them to reduce complexity to $O(n_1 + n_2)$ and we store precomputation results (e.g. covariance matrix determinants) for faster evaluation of the right hand side of Equation 2.10.

We use a multiresolution technique to ensure the similarity measure does not become unreliable due to inadequate cell size, and observe that recognition performance improves. We compute each motion profile at $r$ exponentially decreasing resolutions of Hough space and sum similarity measures with exponentially decreasing weights, as in the pyramid match kernel scheme [26]:

$$similarity(p_1, p_2) = \frac{\sum_{k=0}^{r} 2^{-k} BC_k(p_1, p_2)}{\sum_{k=0}^{r} 2^{-k}} \tag{2.11}$$

where $BC_k(p_1, p_2)$ is the Bhattacharyya coefficient of the two motion profiles at resolution $k$.

The similarity measure can be made invariant to execution rate by using time values normalized with respect to the video volume's time extent. It can also be made invariant

to image location through a scheme to estimate image displacement between motion profiles (see Section 2.3.3). We experiment with the four resulting variants of the similarity measure in a k-NN framework for action detection and classification.

### 2.3.3 Estimating the Displacement between Similar Action Instances

In certain surveillance scenarios, the image location where an action is performed carries meaning, but in general recognition tasks (e.g. is the human running or walking?) the location is irrelevant. We develop a scheme to estimate the displacement between two similar action instances that is robust to small changes in the motion vector sets (due to e.g. body occlusion or missed feature points).

We observe that the translation of a motion vector affects its $\rho$ and $d$ values but not its $\theta$ value. Given the $(\rho, \theta, d)$ values of a motion vector and $(\rho', \theta, d')$ values of its translated version, from Equations 2.2-2.4 the translation $(du, dv)$ is

$$du = (\rho' - \rho)cos\theta + (d' - d)sin\theta \tag{2.12}$$

$$dv = (\rho' - \rho)sin\theta - (d' - d)cos\theta \tag{2.13}$$

The groups in the motion profile of a set of motion vectors correspond to the groups in the profile of a displaced version of that set. For two matched groups, the displacement can be estimated via Equations 2.12 and 2.13, but we do not attempt to find correspondences between groups. Instead, we average the displacements estimated for pairs of groups in cells with the same discretized $\theta$ value; errors for wrong pairs compensate if the two action instances are similar. We iterate over all discretized $\theta$ values, weigh displacements proportional to closeness in time, and use a multiresolution technique as for profile

similarity (Section 2.3.2).

With an estimate of the displacement between two profiles, we can translate the groups of one profile and then compare it to the other. The $\rho$ and $d$ values of groups are updated by adding $du$, $dv$ to $u$, $v$ in Equations 2.2 and 2.4. The new $\rho$ values are discretized, possibly shifting groups to new cells. The estimated displacement is precise only for reasonably similar action instances, and the computed similarity value is high. Dissimilar instances tend to produce random displacements, which could still lead to high similarity values, but this is not a problem in practice.

## 2.4 Experimental Evaluation

### 2.4.1 KTH Database

We tested our system on the commonly used KTH action database [9], which has 6 action classes (boxing, clapping, waving, jogging, running, walking) performed by 25 persons under 4 scenarios (outdoors, outdoors with camera zoom, outdoors with different clothes, indoors). We followed the leave-one-person-out procedure [14] [6]: train on videos from 24 persons and test on videos from the remaining 1 person, iterating through all the 25 persons. The videos with camera zoom are discarded because our motion descriptor does not yet handle scale changes. We obtained an overall recognition rate of 0.90; the confusion matrix is shown in Figure 2.1a. We also tested on the entire database (including videos with camera zoom) and used a 16-9 persons random split [15] (20 trials) in order to compare our results with state of the art approaches (see Figure 2.1b). On an Intel Core2 3GHz and with a MATLAB implementation, our system is an order of magnitude

| | box | clap | wave | jog | run | walk |
|---|---|---|---|---|---|---|
| box | 0.97 | 0.03 | 0.00 | 0.00 | 0.00 | 0.00 |
| clap | 0.10 | 0.81 | 0.08 | 0.00 | 0.01 | 0.00 |
| wave | 0.00 | 0.04 | 0.96 | 0.00 | 0.00 | 0.00 |
| jog | 0.00 | 0.00 | 0.00 | 0.84 | 0.11 | 0.05 |
| run | 0.00 | 0.00 | 0.00 | 0.16 | 0.84 | 0.00 |
| walk | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.99 |

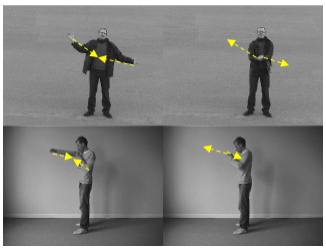
(a)

| Method | Cues | Rate for LOO | Rate for $\frac{2}{3} - \frac{1}{3}$ | Time per frame (s) |
|---|---|---|---|---|
| Ours | motion | 0.8712 | 0.8510 | 0.02 |
| Ke [3] | motion | – | 0.6296 | – |
| Fathi [4] | motion & tracking | – | 0.9050 | 0.75 |
| Niebles [14] | appearance & motion | 0.8150 | – | – |
| Jhuang [15] | appearance & motion | – | 0.9170 | 0.6 |
| Mikolajczyk [7] | appearance & motion | – | 0.9317 | 0.5 – 10 |
| Laptev [8] | appearance & motion | – | 0.9650 | – |

(b)

Table 2.1: Results on the KTH database. (a) Confusion matrix when videos with camera zoom are excluded. The overall recognition rate is 0.90. (b) Comparison with other results when videos with camera zoom are included.

faster than state of the art methods. Recognition performance is slightly lower than state of the art, but a large fraction of the misclassified instances are virtually impossible to discriminate using motion only. Instances of clapping vary a lot in their motion patterns and some patterns are similar to those of boxing and waving (see Figure 2.3); there is confusion between clapping and boxing and clapping and waving, but not viceversa as boxing and waving are more homogenous. The classic jogging-running confusion and the less common jogging-walking are more pronounced because our approach does not use appearance or shape, which can discriminate further when speeds are too similar.



(a)

(b)

Figure 2.3: Clapping examples (top row a,b) with motion tracks similar to boxing and waving (bottom row a,b). (a) arms move diagonally; in boxing, fists are thrown forward and elbows drawn back on a single diagonal direction and they occlude each other. (b) arms are dropped after hands come together and then raised for the next cycle.

### 2.4.2 Checkout Counter Database

We also tested the system on a database of videos simulating a checkout counter of a department store in a laboratory setting. One person performed 5 actions: pick up item from cart, put item down on belt (customer), pick up item from belt, scan item and put item down on table (cashier). There are 447 action instances and the average and maximum duration are 0.89 and 2.13 s respectively (shorter than the average KTH duration of $\approx 4$ s). The human actor is sometimes occluded and manipulates objects of various shapes and appearances (see Figure 2.4). We evaluated the system on two action recognition tasks: classification and detection. For the first task, we randomly chose a training set of $\frac{2}{3}$ of the instances and a validation set with the remaining $\frac{1}{3}$. Averaging over 20 trials, we obtain an overall recognition rate of 0.95 and the confusion matrix in Figure 2.2a. For the second task, we randomly chose a training set of $\frac{2}{3}$ of the videos and a validation set with the remaining $\frac{1}{3}$. An action detection is considered correct if its endpoints are within 0.5 s of the endpoints of an annotated instance of that action. Although this seems generous when compared to the average duration, instances of the same action occur more than one second away from each other (there is at least one other instance in between or, in the case of "scan item", a "reset" gesture). Averaging over all actions and over 20 trials, we trace the Precision-Recall curve in Figure 2.2b; the crossover point is 0.71.

We evaluated the displacement estimation procedure in a similar way, but randomly translated the motion vector sets of validation instances. The translation can be expressed with distance and angle and we computed overall recognition rates for increasing distances. For each distance value, we ran 20 trials with a random $\frac{2}{3}$-$\frac{1}{3}$ split and a random

(a)

(b)

(c)

(d)

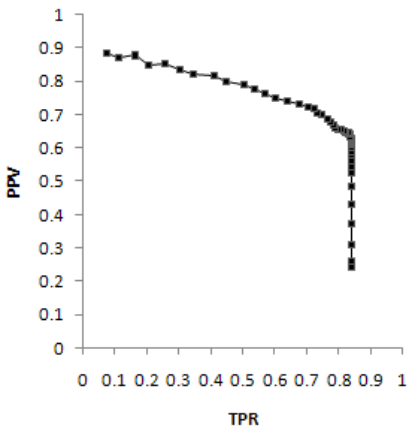
(e)

Figure 2.4: The actions in the checkout counter database: (a) pick up item from cart, (b) put item down on belt, (c) pick up item from belt, (d) scan item, (e) put item down on table.

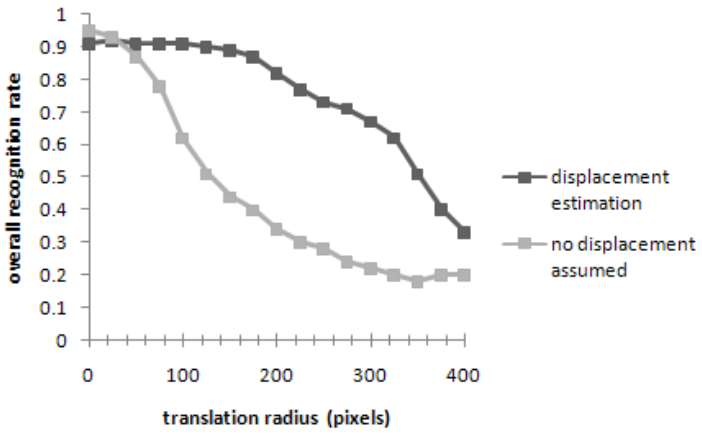
angle. With displacement estimation, the overall recognition rate is approximately 0.9 for displacement less than $150$ pixels, and decreases slowly as motion profiles are clipped by the image window. Without estimation, the rate degrades faster to random guessing (Figure 2.2c).

| | pick up from cart | put down on belt | pick up from belt | scan | put down on table |
|---|---|---|---|---|---|
| pick up from cart | 0.97 | 0.03 | 0.00 | 0.00 | 0.00 |
| put down on belt | 0.00 | 1.00 | 0.00 | 0.00 | 0.00 |
| pick up from belt | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 |
| scan | 0.00 | 0.00 | 0.00 | 0.96 | 0.04 |
| put down on table | 0.00 | 0.00 | 0.03 | 0.16 | 0.81 |

(a)




(b)




(c)

Table 2.2: Results on the checkout counter database. (a) Confusion matrix for the classification task. The overall recognition rate is 0.95. (b) PR curve for the detection task. (c) Overall recognition rate as function of displacement (similarity measure with and without displacement estimation).

# Chapter 3: Kernel PLS Regression for Robust Monocular Pose Estimation

## 3.1 Overview

We evaluate the robustness of five regression techniques for monocular 3D pose estimation. While most of the discriminative pose estimation methods focus on overcoming the fundamental problem of insufficient training data, we are interested in characterizing performance improvement for increasingly large training sets. Commercially available rendering software allows us to efficiently generate large numbers of realistic images of poses from diverse actions. Inspired by recent work in human detection [36], we apply PLS and kPLS regression to pose estimation. We observe that kPLS regression incrementally approximates GP regression using the strongest nonlinear correlations between image features and pose. This provides robustness, and our experiments show kPLS regression is more robust than two GP-based state-of-the-art methods for pose estimation.

## 3.2 Introduction

Human pose estimation is a well studied class of problems in computer vision that has applications to human computer interaction and relevance to motion capture and human

action recognition. Inferring 3D pose from images is difficult mainly because of the high dimensionality of the pose space and the multimodal character of the mapping from image space to pose space, while additional sources of problems include anthropometric differences, appearance variability, occlusion, and background clutter. For discriminative approaches to monocular 3D pose estimation, the high dimensionality of the pose space manifests itself in the insufficiency of training data. Many approaches are based on Gaussian Process (GP) regression, which offers good performance for scarce training data, and/or apply dimensionality reduction techniques that leverage the small dimension of activity-specific pose manifolds [34, 40, 42, 33]. Motion capture and synthetic rendering cannot produce exhaustive quantities of pose-image pairs, but Computer Graphics is currently capable of generating realistic images at low computational cost, so a natural question to ask is "how does the performance of discriminative pose estimation methods improve as the number of training samples increases?". An answer to this question helps to understand more about the relative strengths and weaknesses of these methods and eventually to make optimal choices for specific pose estimation tasks. We perform experiments in which we resample initial sets of training and test poses using Parzen window distributions with a relatively large window size, in order to increase the density of the training set and, respectively, to make the test set more challenging. We use character animation software to render images of all pose instances and compare the error rates of five methods to regress pose on image features, including two methods not previously investigated for pose estimation.

In the human detection community, Projection to Latent Structures (PLS) analysis has recently been used for linear dimensionality reduction in a setting with very high di-

mensional inputs and relatively small training sets [36]. An immediate extension to pose estimation would be to use rich image descriptors and PLS regression, but the highly nonlinear relation between image features and pose calls for "kernelization". It is interesting to compare kernel PLS and GP regression: the former incrementally builds an output-constrained approximation of the inverse of the input Gram matrix, while the latter uses its exact inverse, enforcing output constraints through kernel parameters. For kPLS, the computational complexity of training is $O(n^2p)$, where $n$ is the number of training samples and $p$ the number of latent vectors (typically $\leq 100$), while for GP it is $O(n^3)$, which means kPLS regression can train on larger datasets than GP regression in its original form. A host of sparsification techniques have been developed to make GP regression computationally feasible, but denser training data enables kPLS to find the strongest nonlinear correlations between image features and pose, and thus to build a more robust model. We show kPLS regression outperforms, in terms of robustness, the state-of-the-art methods sparse GP regression [34] and twin GP regression [40] on a dataset with 4 activities and 95,800 pose-image pairs. Our main contribution is to demonstrate the advantages kPLS regression offers when applied as a method for pose estimation.

## 3.3 Related Work

In a well-known early paper on pose tracking, Bregler and Malik represented pose sequences with an initial pose and a composition of twists (special parameterization of rigid body transformations), and related the optical flow constraint equation analytically to these twist parameters [37]. Other early pose tracking work employed a generative frame-

work and proposed "annealed" particle filtering to circumvent the need for very large numbers of particles when filtering in high dimensionsional spaces [44]. The discriminative method of Rosales and Sclaroff [45] regressed 2D pose on Hu moments computed from the silhouette and used pose space partitions and temporal continuity constraints for disambiguation. Shakhnarovich and Darrell optimized nearest neighbor computation for pose estimation by modifying a probabilistic hashing-based nearest neighbor scheme to use pose information in the hashing functions [43].

The pose literature is vast and we limit our review of recent papers to those most relevant to the problem of building a mapping between single-image features and 3D pose. Among these, Aggarwal and Triggs' work [32] used Relevance Vector Machines for non-linear regression of pose on histogram of shape contexts features, aiming for sparsity of regression coefficients. They disambiguated between very different poses with similar image observations in a tracking framework that combines a dynamical model with a local regressor that predicts a new pose from image features and the current pose. More recently, Urtasun and Darrell proposed a sparsification technique to approximate global GP regression with virtual local regressors centered on training instances and defined over small pose neighborhoods [34]. Bo and Sminchisescu leveraged the correlations between pose space dimensions by assuming an additional GP prior over pose points, and achieved significant improvement over basic GP regression [40]. Even though they targeted the tasks of head pose estimation and object tracking, Ranganathan and Yang introduced a computationally efficient method for GP regression with applicability to pose estimation [41]. Instead of directly computing the inverse of the (typically sparse) input Gram matrix, they updated it efficiently for each training instance and lowered the learning complexity

to square of the number of training instances. kPLS regression exploits both the correlations in the feature space and those in pose space, like Bo and Sminchisescu's method [40], but in contrast from it, kPLS explictly looks for the strongest correlations across both input and output dimensions. Although Urtasun and Darrell [34] and Ranganathan and Yang [41] make GP regression tractable, their potential for improvement when denser training data is available is limited, because the input Gram matrix depends only weakly on the outputs. On the other hand, denser data more clearly reveals the strongest feature-pose associations to kPLS regression.

An important line of research employs latent variable models for nonlinear dimensionality reduction of the pose or image feature space or of both. One of the first works based on Gaussian Process Latent Variable Models (GPLVMs) represented activity-specific joint density over the latent and pose space and required very few training instances [33]. The authors derived a prior over pose space using the joint density and integrated it in a tracking framework together with reprojection error-based likelihood and with a smooth motion model. Navaratnam et al. learned a single GPLVM-based latent space representation for both pose and image feature spaces and were thus able to augment their training set with instances for which only image features or only pose is available [48]. Another extension was made by Gupta et al [38] by adding a mapping from image features to the latent space, which, for the purpose of inferring pose by sampling in latent space, provided automatic initialization and guaranteed distance preservation in the image feature space. Lee and Elgammal separated pose into global and root-relative parameters, added anthropometric parameters, and constructed a generative mapping from embedding spaces of all three categories of parameters to image features, using Higher Order Singular Value

Decomposition [50]. More recently, Tian et al. used Locality Preserving Projections to independently learn latent spaces for both the image feature space and the pose space, and regressed with Gaussian Mixture regression (with a small number of components) between the two latent spaces [42]. Despite the significant reduction in the number of training instances needed in discriminative approaches and of particles/samples in generative approaches, it is not clear that low dimensional representations are effective beyond simple actions like walking or running, as observed by Gall et al. [52].

Other authors follow the format of partitioning the pose or the observation space and learning the relation between image features and pose in regions that are smaller than the original training set, and therefore easier to model. Bissacco et al. [31] selected features from a set of randomly generated Haar features and partitioned the image feature space hierarchically, in a similar fashion to Classification and Regression Trees. By finding optimal split points and constant approximation coefficients at each node of the tree, they progressively built a piecewise constant function from image features to pose. Okada and Soatto partitioned the pose space by k-means clustering and learned partition-specific classifiers and linear regressors to first select the applicable model and then use it to predict the pose of a test instance [39]. The method was designed specifically for images with background clutter and the main contributions were two image feature selection mechanisms. Sminchisescu et al. presented a Bayesian Mixture-of-Experts (BMoE) model that captures the multimodal character of the mapping from image features to pose [35]. Given image features, each expert proposes a unimodal distribution over pose and its output is weighted by a probabilistic gating function; this partitions the image feature space via the finite support regions of the gating functions, but unlike in [31], the re-

gions may overlap. A problem with partitioning-based methods is that the models behave poorly at region borders.

There are also papers that do not fit neatly in any of the categories in the paragraphs above. Kanaujia et al. experimented with four types of hierarchical image features, metric learning, and image manifold regularization in order to improve the robustness of their BMoE predictor to background clutter and appearance variability [30]. A semi-supervised learning framework allowed training with real-world images with unknown pose. As a way to address the problem of insufficient training data for discriminative methods, Salzmann and Urtasun proposed generative search initialized with a discriminative prediction and constrained by body segment lengths [49]. The constraints reflect the correlations of pose dimensions typically ignored in discriminative methods ([40] is a recent exception). Fossati and Fua use the observable space of 2D positions of hands/feet as the latent space for 3D pose tracking of skiing, skating and golf [51].

## 3.4 GP Regression

Given training inputs $\boldsymbol{X} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_n]^t$ and outputs $\boldsymbol{Y} = [\boldsymbol{y}_1 \dots \boldsymbol{y}_n]^t$, $\boldsymbol{x}_i \in \mathbb{R}^{n_x}, \boldsymbol{y}_i \in \mathbb{R}^{n_y}, i = 1 \dots n$, we want to model the relation $\boldsymbol{f}$ between them. A GP prior is imposed on functions $\boldsymbol{f}$, within a Bayesian framework [53]:

$$\mathbb{E}(\boldsymbol{f}(\boldsymbol{x})) = 0 \tag{3.1}$$

$$cov(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x'})) = k_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{x'}) I_{n_y} \tag{3.2}$$

We consider a kernel with a Gaussian term and a noise term:

$$k_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{x'}) = \alpha_2 \exp\left(-\alpha_1 ||\boldsymbol{x} - \boldsymbol{x'}||^2\right) + \alpha_3 \delta(\boldsymbol{x}, \boldsymbol{x'}) \tag{3.3}$$

Conditioning $\boldsymbol{f}$'s with the training data $\boldsymbol{X}$ and $\boldsymbol{Y}$, the prediction at a test input point $\boldsymbol{x}$ is

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{\mu}, \sigma^2 I_{n_y}) \tag{3.4}$$

where

$$\boldsymbol{\mu} = \boldsymbol{Y}^t \boldsymbol{K}_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{X}) \boldsymbol{k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{3.5}$$

$$\sigma^2 = k_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_{\boldsymbol{\alpha}}^t(\boldsymbol{x}) \boldsymbol{K}_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{X}) \boldsymbol{k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{3.6}$$

and

$$(\boldsymbol{K}_{\boldsymbol{\alpha}}(\boldsymbol{X}))_{ij} = k_{\boldsymbol{\alpha}}(\boldsymbol{x}_i, \boldsymbol{x}_j);\ \boldsymbol{k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \begin{bmatrix} k_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{x}_1) \\ \dots \\ k_{\boldsymbol{\alpha}}(\boldsymbol{x}, \boldsymbol{x}_n) \end{bmatrix} \tag{3.7}$$

In the learning phase, the kernel hyperparameters $\boldsymbol{\alpha}$ are estimated by maximizing the negative log-likelihood of the training data, as is typically done:

$$\begin{aligned} -\log p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\alpha}) = & \frac{n n_y}{2} \log 2\pi + \frac{n_y}{2} \log |\boldsymbol{K}_{\boldsymbol{\alpha}}(\boldsymbol{X})| \\ & + \frac{1}{2} Tr\left(\boldsymbol{K}_{\boldsymbol{\alpha}}^{-1}(\boldsymbol{X}) \boldsymbol{Y} \boldsymbol{Y}^t\right) \end{aligned} \tag{3.8}$$

We used the Scaled Conjugate Gradient algorithm, starting from a few different $\boldsymbol{\alpha}$s to ensure the search does not become trapped in a local optimum. The complexity of the learning phase is $O(n^3)$, with the constant behind the $O$ notation proportional to the number of search iterations.

## 3.5 PLS and kPLS Regression

We briefly describe PLS and kPLS regression based on the presentation in [54]. PLS regression assumes the input and output variables depend on a common set of latent vari-

ables and it approximates the input and output matrices in terms of their projections on $p$ latent vectors

$$\boldsymbol{X} \cong \boldsymbol{T}\boldsymbol{P}^t \tag{3.9}$$

$$\boldsymbol{Y} \cong \boldsymbol{U}\boldsymbol{Q}^t \tag{3.10}$$

where $\boldsymbol{T}$ and $\boldsymbol{U}$ have dimension $n \times p$ (their columns are latent vectors), $\boldsymbol{P}$ has dimension $n_x \times p$ and $\boldsymbol{Q}$ has dimension $n_y \times p$. A few methods exist for computing the decomposition in Equations 3.9 and 3.10 and we used Nonlinear Iterative Partial Least Squares (NIPALS). NIPALS performs $p$ iterations of the following three steps:

(1) find the linear combinations of columns of $\boldsymbol{X}$ and $\boldsymbol{Y}$ that are maximally correlated:

$$(\boldsymbol{r}, \boldsymbol{s}) = \underset{|\boldsymbol{w}|=|\boldsymbol{c}|=1}{\operatorname{argmax}} [cov(\boldsymbol{X}\boldsymbol{w}, \boldsymbol{Y}\boldsymbol{c})]^2 \tag{3.11}$$

(2) compute the latent vectors $\boldsymbol{t} = \boldsymbol{X}\boldsymbol{r}$ and $\boldsymbol{u} = \boldsymbol{Y}\boldsymbol{s}$

(3) deflate $\boldsymbol{X}$ and $\boldsymbol{Y}$ by their rank-1 approximations based on $\boldsymbol{t}$'s direction $\boldsymbol{d} = \boldsymbol{t}/||\boldsymbol{t}||$:

$$\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{d}\boldsymbol{d}^t\boldsymbol{X} \tag{3.12}$$

$$\boldsymbol{Y} = \boldsymbol{Y} - \boldsymbol{d}\boldsymbol{d}^t\boldsymbol{Y} \tag{3.13}$$

Concatenating all $p$ vectors $\boldsymbol{t}$ and $\boldsymbol{u}$ into matrices $\boldsymbol{T}$ and $\boldsymbol{U}$, respectively, the output prediction for test input $\boldsymbol{x}$ is

$$\boldsymbol{y} = \boldsymbol{Y}^t\boldsymbol{T}(\boldsymbol{U}^t\boldsymbol{X}\boldsymbol{X}^t\boldsymbol{T})^{-1}\boldsymbol{U}^t\boldsymbol{X}\boldsymbol{x} \tag{3.14}$$

The "kernel trick" requires solving a kernelized version of the eigenvalue problem

in Equation 3.11 and updating a Gram matrix instead of $\boldsymbol{X}$:

$$\boldsymbol{K} = \boldsymbol{K} - \boldsymbol{d}\boldsymbol{d}^t\boldsymbol{K} - \boldsymbol{K}\boldsymbol{d}\boldsymbol{d}^t + \boldsymbol{d}\boldsymbol{d}^t\boldsymbol{K}\boldsymbol{d}\boldsymbol{d}^t \tag{3.15}$$

The output prediction for test input $\boldsymbol{x}$ is

$$\boldsymbol{y} = \boldsymbol{Y}^t\boldsymbol{T}(\boldsymbol{U}^t\boldsymbol{K}_{\boldsymbol{\alpha}}(\boldsymbol{X})\boldsymbol{T})^{-1}\boldsymbol{U}^t\boldsymbol{k}_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{3.16}$$

Note the similarity between Equations 3.5 and 3.16: GP regression computes the exact inverse of the input Gram matrix, while kPLS approximates it through projection on latent vectors. The approximation is computed much faster: $O(n^2p)$ as opposed to $O(n^3)$, with $p$ comparable to the size of the output, $n_y$. (In typical pose estimation settings, $n_y \leq 100$ and $n$ is in the order of thousands or higher.) This allows kPLS regression to work on larger training sets than are computationally feasible for GP regression - the memory complexity of $O(n^2)$ is what effectively limits the size of the training set. However, a more important advantage of kPLS regression is that it incrementally discovers the strongest correlations between input and output variables, and so it provides a performance/generalizability tradeoff via the number of latent vectors $p$. GP regression also captures correlations between input and output variables through kernel hyperparameters $\boldsymbol{\alpha}$, but kPLS regression does so more directly and offers explicit control over the process. A limitation of both kPLS and GP regression is that they do not model pose ambiguity; however, this can be easily solved by learning models specific to unambiguous pose regions. We do not concern ourselves with this issue in the present study.

## 3.6 Experimental Evaluation

### 3.6.1 Pose Data

We selected from the CMU motion capture database [47] subsequences of 4 different activities - walks, jumping jacks, picking up and directing traffic - and created 4 corresponding datasets. Within each dataset we randomly assigned each subsequence to be used for training or testing and we obtained an approximately 50%-50% split of training and test instances (note that we avoid partitioning at instance level and consider whole subsequences instead). The 4 datasets had between 1300 and 2300 training instances and GP regression training took a reasonable amount of time (maximum was 22 minutes on an Intel Core2 Quad @ 3GHz).

Next, we created for each dataset an augmented training set and a new test set, starting from the original training and original test set, respectively. We approximated the training and test pose distributions with kernel density estimates based on the training and test instances, and we drew random independent samples from these two distributions. The KDE kernels were separable box kernels of width $30^\circ$ for training and $15^\circ$ for testing (in this task, pose was represented with three Euler angles per body joint). We generated 10 times more samples for the augmented training set than in the original training set and a fixed number of samples (1000) for the new test set; the total number of instances in our study was 95,800. Some of the poses in the augmented training set look contrived, but the poses in the new test set are close to those in the original test set, making for a harder but still realistic test set, see Figure 3.1. We evaluated the performance of regression methods

in four training-test settings: (S-1) original training - original test; (S-2) original training - new test; (S-3) augmented training - original test; (S-4) augmented training - new test. S-2 shows the generalizability of the originally learned model; S-3 and S-4 measure prediction improvement and generalizability improvement when training instances are added. A robust method should perform well in settings S-2 and S-4; in S-3, good performance or improvement over S-1 is not guaranteed, as robustness implies learning a more general model than that needed for good performance on the original test set.

### 3.6.2 Regression Tests

For regression, we used 66 dimensional pose vectors that encode the 3D positions of 22 human body joints, recentered so the position of the hip joint is $(0, 0, 0)$. We measured the error of the predicted 3D positions, but unlike [55], we used $L_2$ norm instead of $L_1$, in order to penalize large errors more heavily. The errors are reported in millimeters, assuming a collarbone length of 300mm. We rendered images in Poser 8 [46] using the two characters models "Ryan_Casual" and "Alyson_Casual"; since we only model unimodal mappings, we chose viewpoints so that pose can be recovered unambiguously, see Figure 3.2. The features extracted from images were multi-resolution foreground image moments (FIM) and Bags of Shape Contexts (BOSC) as defined in [32]. For computing FIM features, we centered the image at the center of mass of the foreground and used 6 grids of exponentially increasing resolution. For BOSC, we used shape contexts with a maximum radius proportional to the height of the foreground, 5 log-radius bins and 12 angular bins; as in [32], shape context descriptors were computed at points sampled along the

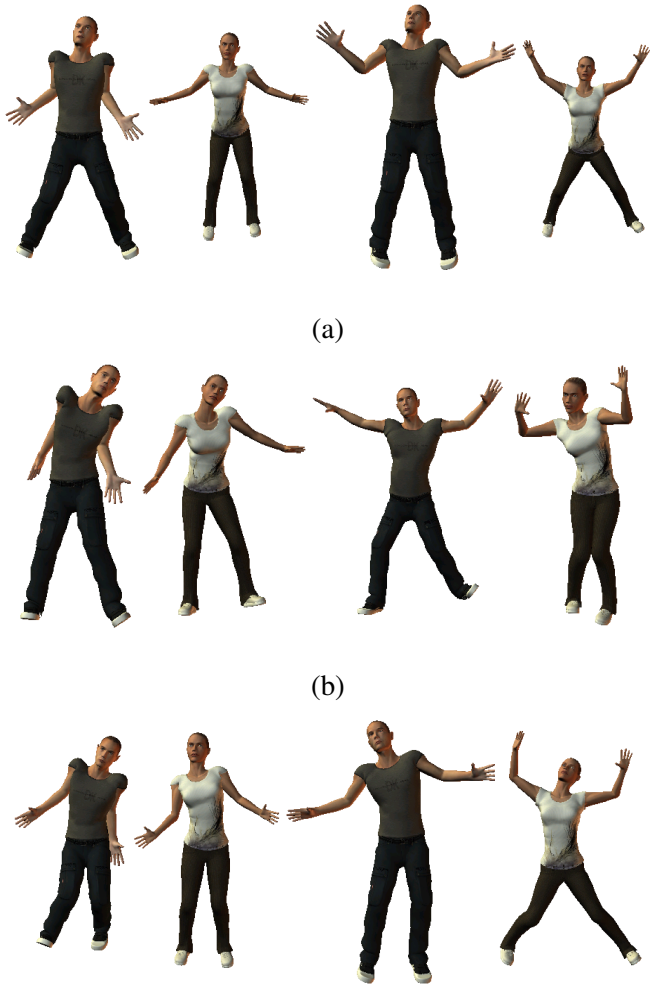


Figure 3.1: Example instances from the jumping jack dataset, rendered with Poser [46]. (a) Instances from the original training set. (b) Instances from the augmented training set (separable box kernel with $30^\circ$ width). Though contrived, they improve the generalizability of the model. (c) Instances from the new test set. In the Euler angle pose representation, each new test instance has angles at most $7.5^\circ$ different from the angles of an instance in the original test set. Note that the images depict random unrelated instances from three different phases of the motion cycle.

silhouette and quantized using a 100-word dictionary. The number of non-constant FIM features varied from 3,080 in the walking dataset to 8,493 in the traffic signals dataset.

(a)

(b)

Figure 3.2: Images of walking rendered with Poser [46] from a view at a diagonal with the walking direction, so that the left-foot-forward (3.2a) and right-foot-forward (3.2b) poses are easily distinguished, even when using silhouette-based features. This ensures meaningful comparison between unimodal prediction models.

We experimented with five regression techniques: GP, Sparse GP [34], Twin GP [40], PLS and kPLS regression. In addition to these techniques, we used weighted nearest neighbor (WNN) regression and "oracle" weighted nearest neighbor (OWNN) re-

gression as benchmarks. OWNN regression invokes an oracle to obtain the distances in pose space from a test instance to the training instances, and then linearly interpolates between the poses of the nearest training instances. OWNN regression simulates WNN regression with a perfect feature space metric and allows us to quantify the potential for improvement through better features or feature space metrics. OWNN and WNN regression were run with 50 neighbors for all 4 datasets. In GP regression training, the searches were initialized with $\alpha_2 = 1$, $\alpha_3 = 0.001$ and $\alpha_1$ from the set of values $\{0.1, 0.2, 0.5, 0.7, 1, 2, 4, 5, 7, 10, 12, 15, 17, 20, 25\}\times$ the inverse of the variance of pairwise distances between training feature vectors $\boldsymbol{x}$. To replicate the sparse GP regression configuration in [34] as close as possible, we used 10 experts for prediction, an expert neighborhood size of 50 instances, and a number of learned experts roughly equal to the number of training instances in a dataset divided by the expert neighborhood size (so as to cover the entire training set). For twin GP regression, we used the implementation publicly made available by the authors, with $\gamma$ parameters obtained through cross validation. We performed two fold cross validation like [40], but had to trim each validation set to 100 randomly selected instances in order to make this process computationally feasible. Twin GP regression employs online sparsification by interpolating among the training instances closest to the test instance. Smaller training instance neighborhoods increase execution speed but also potentially decrease accuracy; we decided to use 1000 neighbors, as in [40]. The values considered for the two $\gamma$s were $\{1, 2, 10, 15, 20, 25, 30\}\times$ the inverse of the variance of the pairwise distances between training feature vectors $\boldsymbol{x}$ and training pose vectors $\boldsymbol{y}$, respectively. For PLS and kPLS regression, we chose the number of latent vectors (and kernel hyperparameters for kPLS) using a cross validation scheme

in which we randomly selected pairs of small subsets of the training set for training and validation. The values considered for the number of latent vectors in both PLS and kPLS were $\{10, 20, 30, 45, 60, 80, 100\}$. The values considered for kernel hyperparameter $\alpha_1$ in kPLS were $\{1, 5, 10, 20\}\times$ the inverse of the variance of the pairwise distances between the training feature vectors $\boldsymbol{x}$. We used FIM features with both WNN and PLS regression and BOSC features with GP, SGP, TGP and kPLS regression.

| RMS error | OWNN | WNN | GP | SGP | TGP | PLS | kPLS |
|---|---|---|---|---|---|---|---|
| S-1 | 37 | 62 | 53 | 61 | **51** | 62 | 55 |
| S-2 | 78 | 118 | **108** | 121 | 110 | 120 | 110 |
| S-3 | 36 | 61 | x | 60 | **52** | 73 | 61 |
| S-4 | 63 | 94 | x | 95 | 96 | 99 | **80** |

(a) walking

| RMS error | OWNN | WNN | GP | SGP | TGP | PLS | kPLS |
|---|---|---|---|---|---|---|---|
| S-1 | 47 | 70 | 63 | 73 | 61 | **58** | 60 |
| S-2 | 91 | 113 | 111 | 114 | **102** | 105 | 105 |
| S-3 | 45 | 60 | x | 59 | **52** | 60 | 54 |
| S-4 | 67 | 87 | x | 87 | 83 | 84 | **70** |

(b) jumping jack

| RMS error | OWNN | WNN | GP | SGP† | TGP‡ | PLS | kPLS |
|---|---|---|---|---|---|---|---|
| S-1 | 111 | 171 | **143** | 180 | 168 | 186 | 153 |
| S-2 | 128 | 197 | **170** | 207 | 205 | 217 | 179 |
| S-3 | 77 | 138 | x | 129 | 194 | 151 | **110** |
| S-4 | 83 | 144 | x | 141 | 167 | 158 | **120** |

(c) pick up

| RMS error | OWNN | WNN | GP | SGP† | TGP‡ | PLS | kPLS |
|---|---|---|---|---|---|---|---|
| S-1 | 137 | 188 | **172** | 202 | 199 | 184 | 176 |
| S-2 | 151 | 217 | **196** | 230 | 224 | 206 | 201 |
| S-3 | 84 | 159 | x | 158 | 154 | 142 | **116** |
| S-4 | 90 | 168 | x | 167 | 178 | 154 | **127** |

(d) traffic signals

Table 3.1: The RMS error of the predicted 3D joint positions, normalized w.r.t. the root position. The errors are expressed in mm and scaled to reflect a collarbone length of 300mm. Table entries containing an "x" indicate infeasibly large training times.

We present our results in the tables in Figure 3.1. In test settings S-1 (original training - original test) and S-2 (original training - new test), the error rate of kPLS regression

is virtually the same as that of GP regression: the average difference is less than 3mm and the largest difference is 4 mm. In test setting S-3, kPLS regression can improve its prediction for additional training data. This is not the case for walking and jumping jack, but for traffic signals, kPLS improves by 34 mm (19%) from test setting S-1 and it outperforms sparse GP regression by 20 mm (12%). The traffic signals dataset is much more varied than walking (Figure 3.3) and sampling the pose space more densely helps improve accuracy. In test setting S-4 (augmented training - new test), kPLS regression has only 1mm higher error than the best performer (sparse GP) on two datasets, while outperforming all other methods by 3mm and 16mm respectively on the remaining two datasets.

Figure 3.3: Example images from the traffic signals dataset. The large variety of poses is not characteristic of the walking or jumping jack datasets.

Direct comparisons with the other methods highlight additional features of kPLS regression. Even though it has 9mm lower error in setting S-3 (augmented training - original test) on the jumping jack dataset, sparse GP regression is outperformed by kPLS by 6% overall. Twin GP regression is more accurate on the walking and jumping jack

datasets in 4 settings out of 8, but has much higher errors than kPLS on the pick up and traffic signals datasets, which we interpret as evidence that finding parameters for twin GP regression by cross validation is significantly harder for more complex actions. Compared to PLS, kPLS regression generally has lower errors and, more importantly, the parameters obtained by cross validation are more reliable for kPLS than for PLS.

The final observation we make based on our results is that the errors of GP, SGP, TGP, PLS and kPLS regression tend to fall in the range of the errors of OWNN and WNN regression (and sometimes exceed WNN errors). This shows room for improvement by using better features and/or better feature space metrics, and suggests using OWNN regression as a benchmark for pose estimation that can be used to evaluate both new features and new regression techniques.

# Chapter 4: Learning to Detect Carried Objects with Minimal Supervision

## 4.1 Overview

We propose a learning-based method for detecting carried objects that generates candidate image regions from protrusion, color contrast and occlusion boundary cues, and uses a classifier to filter out the regions unlikely to be carried objects. The method achieves higher accuracy than state of the art, which can only detect protrusions from the human shape, and the discriminative model it builds for the silhouette context-based region features generalizes well. To reduce annotation effort, we investigate training the model in a Multiple Instance Learning framework where the only available supervision is "walk" and "carry" labels associated with intervals of human tracks, i.e., the spatial extent of carried objects is not annotated. We present an extension to the miSVM algorithm that uses knowledge of the fraction of positive instances in positive bags and that scales to training sets of hundreds of thousands of instances.

## 4.2 Introduction

In the field of visual surveillance one of the important problems that has received increased attention in recent years is the detection of objects carried by people. The train

bombings carried out in Madrid and London in recent years are strong incentives for a computer vision solution, but there are also other applications, especially military, that require awareness of object presence. While significant progress has been made in detecting and tracking humans, the variability in the appearance of the objects people can carry makes carried object detection a very challenging problem. Capturing the relationships of the object with the human silhouette is also hard, as objects may or may not have color contrast with clothing; may occupy a small fraction of the human silhouette or can be comparable in height with the human; may be carried by hand, under the arm, with both arms, or on the back. Finally, objects may be swung or held still and they may be occluded in some of the frames in which the human is observable.

The most successful approaches so far to finding carried objects have extracted a foreground mask of the human and then matched and subtracted a generic body template (either 2D [78] or 3D [79]), returning the protrusions as objects. While this approach is intuitively appealing, it cannot detect objects in the frequent case when they are mostly inside the human silhouette, in the 2D setting, and it requires a stereo camera moving among people, in the 3D setting. Directly using other cues such as color and motion to find carried objects is bound to produce numerous false alarms corresponding to the head, feet, hands, or just noise, but for human vision it is easy to distinguish body parts from carried objects when displayed together with the human silhouette. We propose a method to detect carried objects that applies three types of low level detectors inside human bounding boxes (based on protrusions, color contrast and occlusion boundaries) and models the resulting image regions as carried objects with a kernel SVM on features related to the human silhouette context.

As the performance of the classifier is directly related to the size of the training set, and as the object annotation process is time consuming (roughly 40,000 precise bounding boxes are needed for one of the datasets in this work), we investigated using a multiple instance learning (MIL) framework. MIL, introduced by [80], departs from the classic supervised learning setting by making labels available for sets of instances (bags) rather than individual instances; in each positive bag there is at least one positive instance while all the instances in negative bags are negative. In our setting, instances are image regions produced by low level detectors and bags are sets of instances from intervals of human tracks annotated as "walk" (no carried object) or "carry" (at least one object), and we focus on instance level classification. Most MIL approaches are computationally intractable for our datasets (our problems range from approximately 12,000 to 192,000 instances), and the few that are tractable–miSVM [81] and sbMIL [82]–can have significantly lower test set accuracy than a fully supervised classifier. Observing that our low level detectors produce a roughly constant fraction of correct regions when the human is carrying an object, we extend miSVM to adjust the fraction of positive labels in positive bags accordingly at each iteration.

Our contribution is two fold: (1) we propose a novel learning-based method for carried object detection with accuracy exceeding state-of-the-art and with good generalization capability; (2) we extend the miSVM algorithm to account for an expected positive bag density, achieving improved accuracy for virtually the same computational cost.

## 4.3 Related Work

The majority of papers on carried object detection follow the pattern of estimating the pixel mask of the person and object, subtracting from it a human template (either abstract or learned from data) and returning the remaining regions. Haritaoglu et al. [76] used background subtraction, averaged human masks temporally, and relied on the symmetry of the walking human silhouette around a principal axis and on the periodic nature of limb motions. Lee and Elgammal [75] proposed a generative silhouette appearance model parameterized by viewpoint, body proportions and gait phase, and iteratively estimated these parameters together with holes in the foreground mask and outlier regions (carried objects). Noting the sensitivity of Haritaoglu et al.'s method to the principal axis estimate, Damen and Hogg [78] matched and subtracted synthetically rendered templates of unencumbered humans. To select the correct template, they require a ground plane homography and an estimate of the walking direction. The most recent work related to carried objects utilized a cylindrical 3D shape representation of humans both in a tracking-before-detection framework and for carried object detection [79]. 2D template subtraction approaches are limited to discovering objects that significantly protrude from the silhouette and their accuracy is dataset dependent – the results section shows Damen and Hogg's method [76] performing poorly when people wear robes. To improve both the recall and the precision of 2D carried object detection, we propose using multiple sources of candidate object regions and then pruning these candidates in the context of the human silhouette.

Interesting context modeling work by Zheng et al [83] effectively combines the ap-

pearance of an object with that of its neighborhood but shows limited performance for carried object detection. At a higher conceptual level, Albanese et al [84] developed a logical language that allows reasoning about object manipulation actions. Other efforts focus on deciding whether people carry something or not, without providing an actual location for the object [77] [74]. While knowing carrying status is valuable, precise object masks are directly usable in important higher level tasks like detecting abandoned objects, theft or object exchange. Unfortunately, much more annotation effort is involved in learning-based methods that explicitly localize objects, but we adopt a MIL framework and still require only weak supervision in the form of carry status.

The Multiple Instance Learning literature is extensive, covering aspects as varied as discovering a single concept shared by positive but not negative bags [71], finding the most appropriate exemplar embedding [73], explicitly factoring in the cost of false positives in the classification task [70], to give just a few examples. As Li et al. [72] noted, most approaches that can classify instances have prohibitive training cost. An exception is the miSVM framework of Andrews et al. [81], who cast MIL as a mixed integer program involving the labels of instances in positive bags and the parameters of the separating hyperplane, and solved it with an iterating heuristic with good performance in practice. Gehler and Chappelle [69] added to the SVM formulation of [81] a term correlated to label uncertainty that allows finding better local minima of the objective function. However, this leads to very high computational cost if the number of instances in positive bags is large, since the SVM solver sees these instances duplicated as both positive and negative. The approach most directly applicable to our setting is due to Bunescu and Mooney [82], who loosened a constraint in their SVM formulation so that as few as one instance per

positive bag can be labeled positive. The results of their approach are inferior to miSVM [81] in our problems, which we believe is because too few of the actual positive instances are labeled positive.

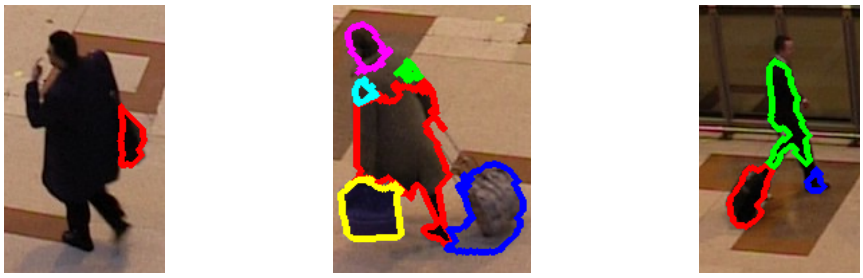
A few researchers used MIL to cope with noisy labels when learning from images retrieved with search engines [67] [72] [68]. Li et al. [72] leveraged the constraint that the fraction of positives in a positive bag is relatively large ($0.6$) and proposed an iterative scheme that trained on an increasingly larger number of bags. In [68], they reduced the high computational cost of the optimization run in each iteration and updated a separating hyperplane incrementally. It is very unlikely that these two methods would be applicable to our problem setting, as the positive bag density varies from $0$ to $0.5$ and the decision surface has to make multiple local distinctions between various objects and body parts.

Lastly, two papers bear superficial resemblance to our work. Fathi et al. [66] used egocentric video to learn to discriminate between object appearances with little supervision. While both works learn to classify image regions in a MIL framework, the problems considered are significantly different: [66] employs multi-class MIL for relatively small training sets, while we use two class MIL for large amounts of data. Ghanem and Davis [65] also adopted a learning approach in connection to carried objects, but could only predict object appearance/disappearance events holistically.

## 4.4 Low Level Detectors

Our method assumes that human tracks are available and runs background subtraction [56] and optical flow [64]. Next, three types of image region detectors are run: an op-

tical flow-based protrusion detector, a segmentation-based color contrast detector and an occlusion boundary-based moving blob detector, see Figure 4.1. The three detectors are simple but have high probability of finding carried objects if they exist; if none of them fires during an interval in which a person is carrying an object, then most likely the object does not protrude, has poor contrast, and is static with respect to the body – an extremely hard target to find. We ignore such cases and instead address the problem of disambiguating between image regions corresponding to body parts/noise versus those that are carried objects, using the context of the human silhouette. With respect to [78], we additionally require optical flow but we improve on their detector (section 4.4.1), use two additional detectors and employ a mechanism to select the correct regions.



(a) (b) (c)

Figure 4.1: Sample output of low level detectors: (a) optical flow-based protrusion (b) segmentation-based color contrast (c) occlusion boundary-based moving blob. Each of these is too noisy as a carried object detector, but human silhouette context can be used effectively to filter its output.

### 4.4.1 Optical Flow-based Protrusion Detector

The optical flow-based protrusion detector builds a probabilistic mask for each human bounding box that reflects how close the motion of a pixel is to the average translation in the box. We call this the carried probability mask (CP) and we define it by assuming that the projection of a pixel's velocity on the average translation is normally distributed:

$$CP(p) \propto \exp\left(-\frac{\left(\frac{\mathbf{w}(p)\cdot\bar{\mathbf{w}}}{|\bar{\mathbf{w}}|_2^2}-1\right)^2}{2\sigma^2}\right) \tag{4.1}$$

where $\mathbf{w}(p) = (u(p), v(p))$ is the optical flow vector at pixel $p = (x, y)$ and $\bar{\mathbf{w}}$ is the mean optical flow of the points in the human bounding box. (To compensate for camera motion, the average optical flow over the image is subtracted from all optical flow vectors.) We visualized CPs for a range of $\sigma$'s, observed that smaller values produce holes and larger values overestimate the human shapes, and chose $\sigma = 0.4$ for all videos we process. Limbs swinging opposite to the walking direction tend to be removed, which is advantageous over using background subtraction masks as in [78], since the temporal aggregation for noise reduction can be done effectively on a smaller time interval, e.g. 9 frames as opposed to 50 needed by [78]. We aggregate the CP masks by simply translating them opposite to the average optical flow vector and call the thresholded resulting mask average carrying shape (ACS). The ACS's of unencumbered pedestrians tend to be urn-shaped regardless of viewpoint, which allows matching against a single urn+head template with shape contexts [60] and then retrieve protrusions, see supplemental material. Compared to our protrusion detector, Damen and Hogg [78] incur the disadvantage of needing a ground plane homography and an estimate of the walking direction to select

the proper template.

### 4.4.2 Segmentation-based Color Contrast Detector

The color contrast detector runs mean shift clustering on the foreground mask obtained with background subtraction. Foreground pixels are represented with $[0, 1]$ normalized rgb and image positions (the positions are normalized with respect to the human bounding box). The clustering bandwidth is set to 0.2 for all videos in all datasets; other values do not lead to significantly different segmentations with respect to the carried objects. This detector is designed for situations when the object's color clearly stands out from the colors of the human silhouette, as in Figure 4.1b. As the figure shows, many false positives occur, but a large portion are meaningful parts of the silhouette e.g., body and head.

### 4.4.3 Occlusion Boundary-based Moving Blob Detector

If the person moves the carried object with respect to the body or changes viewpoint while walking, occlusion boundaries will likely appear around the object. To detect them we employ criteria from [63]: occlusion boundaries are pixels where the flow forward from a frame is inconsistent with the flow back into the frame or where the flow gradient has large magnitude. With respect to [63], we tighten the first condition and loosen the second, requiring more consistency but allowing for larger gradient magnitudes:

$$|\mathbf{w}(p) + \mathbf{w}'(p')|_2^2 > 0.01(|\mathbf{w}(p)|_2^2 + |\mathbf{w}'(p')|_2^2) + 0.01 \tag{4.2}$$

$$|\nabla u(p)|_2^2 + |\nabla v(p)|_2^2 > 0.01|\mathbf{w}(p)|_2^2 + 0.01 \tag{4.3}$$

where $p' = p + \mathbf{w}(p)$ and $\mathbf{w}'$ is the backward optical flow field. Superimposing the boundary mask on the foreground mask from background subtraction segments the latter into candidate regions. Empirically we observe this detector frequently finds people's heads and feet.

## 4.5 Learning a Model for Carried Object Regions

The candidate image regions retrieved by the low level detectors are filtered to remove noise: regions less than $10$ pixels in width or height, or greater than half the size of the human mask are eliminated. We also use a compactness filter requiring a region to occupy at least half its minimum area (not necessarily axis aligned) enclosing rectangle. The method might miss some types of objects (e.g. semi-automatic weapons), but since compactness is one of the features we compute for regions, the choice can be reverted by simply removing this filter. The cost is introducing more types of negatives and making learning harder.

### 4.5.1 Region Features

The inspiration for features comes from common sense knowledge about body parts, e.g. the head is near the top of the silhouette, shares contour points with it and is relatively small. We compute twelve features and use a Gaussian kernel SVM for classification. Three features characterize the shape of a region and nine capture its relation to the human silhouette. (To clarify, we use the term silhouette to denote all points inside a shape

as opposed to just its contour.) The silhouette produced by background subtraction is processed with a morphological "open" prior to feature computation to reduce the noise of the estimated silhouette height. The features are:

- compactness: ratio of the region size to the area of its enclosing rectangle
- orientation: the angle of the largest side of the enclosing rectangle with the vertical direction ($\in [0, \frac{\pi}{2}]$)
- aspect ratio: the ratio of the larger side of the enclosing rectangle to the smaller side
- relative size: the ratio of the region size to the silhouette size
- relative x: the absolute difference between the $x$ of the region centroid and the $x$ of the silhouette centroid, normalized by silhouette height (the width is too noisy)
- relative y 1: minimum $y$ of the region normalized with respect to vertical silhouette span
- relative y 2: maximum $y$ of the region normalized with respect to vertical silhouette span
- fraction of horizontal occupancy: the ratio of the region size to the silhouette area between the region's smallest and largest $y$
- fraction of vertical occupancy: the ratio of the region size to the silhouette area between the region's smallest and largest $x$
- fraction of contour points 1: the fraction of points on the region contour that are at most 5 pixels away from the silhouette contour

- fraction of contour points 2: the fraction of points on the silhouette contour that are at most 5 pixels away from the region contour
- local color contrast: $\chi^2$ distance between the color histogram of the region and the color histogram of the silhouette pixels in a bounding box four times larger than the region bounding box (like the CC cue from [61] but projected on the silhouette)

Note that due to different video resolutions, the 5 pixel threshold represents roughly the same quantity relative to the silhouette height.

## 4.6 A Multiple Instance Framework for Learning a Model for Carried Object Regions

One of the typical ways to apply MIL to computer vision is to treat images as bags and their segments as instances. In our framework, the instances are still image regions but the bags are sets of regions produced by the low level detectors in human track intervals annotated as "carry" or "walk". The label "carry" means that the walking human has at least one visible object in some frames of the interval and "walk" means no object visible. The annotations are independent of region detector output, so a slight complication arises that some bags labeled positive may not contain any positive instances at all due to low-level detectors failing to retrieve carried objects. However, a more important aspect is problem size: the smallest problem in this work has approximately $12,000$ instances, about twice more than the well known MIL dataset MUSK-2, and the largest is approximately $192,000$, two orders of magnitude larger. Another difficulty is that the union of positive bags has $51\%$ to $85\%$ of the training instances while the fraction of actual pos-

itives is between $7\%$ and $14\%$ of the training instances. (Note that the latter is different from the expected fraction of actual positives in each positive bag, $25\%$.) Learning an instance level classifier requires overriding bag labels for large numbers of instances in positive bags with the support of a limited number of known negatives.

Numerous MIL methods assume the existence of a few prototypical positive instances common to many positive bags and/or a meaningful Euclidian distance, assumptions which do not hold for our datasets. The most suitable approach is miSVM [81], which iterates two steps: (1) compute the separating hyperplane given all instance labels (initialized with bag labels) and (2) relabel the instances in positive bags according to the current separating hyperplane, correcting so that each positive bag has at least one positive instance. A characteristic of our problem is that the low level detectors produce a fraction of correct regions close to $\alpha_0 = 0.25$ when the person carries an object, so we adapt miSVM to reflect an expectation of the fraction of positives in positive bags, see Algorithm 1. The relabeling is now done so that the fraction of positive instances shifts towards $\alpha_0$ and we call this extension miSVM-Positive Fraction Shift (miSVM-PFS).

The algorithm minimizes the modified SVM objective

$$\begin{aligned} \mathscr{L}(w, b, y_{1..N_+}) &= \frac{1}{2}||w||_2^2 \\ &\quad + C_1 \sum_{i=1}^{N} max(0, 1 - y_i(wx_i + b)) \\ &\quad + C_2 \sum_{j=1}^{n_+} \left| \sum_{k \in B_j} \frac{y_k + 1}{2} - \alpha_0 n_j \right| \end{aligned} \tag{4.4}$$

where $y_{1..N_+}$ are the labels of the instances in positive bags, $N$ is the total number of instances ($N_+$ in positive bags, $N_-$ in negative bags), $n_+$ is the number of positive bags, $B_j$

**Algorithm 1** miSVM-Positive Fraction Shift

**input** : instances, bags, bag labels; $T$, $\alpha_0$, $\theta$

label all instances with their bag labels

**for** $i = 1 \rightarrow T$ **do**

compute separating hyperplane with SVM solver

compute decision values for instances in positive bags

**for** each positive bag **do**

$\alpha \leftarrow$ fraction of instances with decision value $\geq 0$

order the instances by decision values

relabel top $(1-\theta)\alpha + \theta\alpha_0$ instances as positive

relabel rest of bag instances as negative

**end for**

**end for**

**return** separating hyperplane computed with SVM solver for current labels

are the indexes of instances in the j-th (positive) bag and $n_j = |B_j|$. In each iteration, the SVM training minimizes the sum of the first two terms over $w$ and $b$, and the subsequent instance relabeling minimizes the sum of the second and third over $y_{1..N_+}$. To see why the latter is true, consider the change in the second loss term when label $y_k$ switches:

$$\Delta\mathscr{L}_{2k} = \begin{cases} y_k \cdot 2dv_k & |dv_k| < 1 \\ y_k \cdot (dv_k + sign(dv_k)) & |dv_k| \geq 1 \end{cases} \tag{4.5}$$

where $dv_k = wx_k + b$. For each positive bag, minimizing the second term is achieved by switching the labels of instances with decision values of opposite sign to the old labels

($\Delta\mathscr{L}_{2k} < 0$). Any set of label changes can be composed as a set that minimizes the second term of the objective and then some other set of changes. The other set will strictly increase the second term while potentially decreasing the third, so to minimize their sum, it must include only instances with $dv$ between 0 and a threshold depending on $\alpha_0$ and $\frac{C_2}{C_1}$ (smallest $|dv|$'s). This is because $\Delta\mathscr{L}_{2k}$ is monotonically increasing in $dv_k$ and the third loss term does not depend on which labels are switched but on how many. The algorithm implements the two sets of label changes together, by sorting instances by $dv$ and relabeling them in relation to a threshold between 0 and the $dv$ of the top $\alpha_0$-th instance. Parameter $\theta$ equivalently models the effect of $\frac{C_2}{C_1}$.

We observe that the algorithm changes very few labels after 20 iterations in all problem settings, so we fix $T$ to this value. We also set $\theta$ to $0.333$. Note that $\theta = 0$ does not make our algorithm equivalent to miSVM but makes it overfit (miSVM counters overfitting by switching a label when no positives are left in a positive bag). By relabeling instances in positive bags in a controlled manner, with $\theta > 0$ bias towards fraction $\alpha_0$, miSVM-PFS smoothes the trajectory in label space and so is less likely to find local minima. The ALP-SVM version of Gehler and Chapelle's deterministic annealing approach [69] has a similar smoothing effect and employs a similar objective function, but it incurs far higher computational cost in the SVM training step because it duplicates the instances in positive bags as both positive and negative. Running ALP-SVM on a subsampled version of one of the smallest MIL problems in our datasets (thousands of instances) took over one hour while miSVM-PFS finished in under one minute; for the other problems the disparity will be much greater due to SVM training time increasing roughly quadratically with the number of instances.

(a) Pets2006 (b) Cd2a (c) Towncenter

Figure 4.2: Snapshots from datasets used in this work.

## 4.7 Experimental Results

We ran experiments on three datasets: Pets2006, Cd2a and Towncenter, see Figure 4.2 for representative images. Pets2006 [59] is a well known visual surveillance benchmark that contains videos of people walking with luggage in a busy train station. For comparison with the method of Damen and Hogg [78], we ran our system on the 7 videos from the third camera. These videos range from 2,371 to 3,401 frames in length, with an average of approximately 25 people in the scene. Cd2a consists of 16 videos we selected from a corpus collected to highlight carry and exchange actions [57]. The Cd2a videos show people in varied viewpoints in two types of outdoor scenarios: a country road and a safe house. There are few object types (small packets, large boxes, duffel bags and backpacks), but people wear robes and head scarves, which complicate silhouettes. The videos are between 2,430 and 18,023 frames and show an average of approximately 14 people. The Towncenter dataset [58] consists of a single high resolution video of a busy pedestrian-only zone near store fronts. We evaluate our approach on the first 4500 of the 7500 video frames, for which [58] provide (noisy) ground truth human bounding boxes; we annotate the objects carried by the 230 people.

We manually annotate the human tracks which are input to our method and perform training and testing on image regions detected in the parts of the tracks where humans move. These are annotated as "walk" or "carry" according to object presence and also used by the MIL version of our approach. Note that these two settings reflect the scope of our method: given human tracks, the goal is to detect objects carried by walking people, as is done in prior work [78] [76]. Each region feature is normalized by subtracting its training set mean and dividing by standard deviation. For classification, we use libsvm [62] with a Gaussian kernel with $\sigma$ the mean of pairwise distances between instances in the training set.

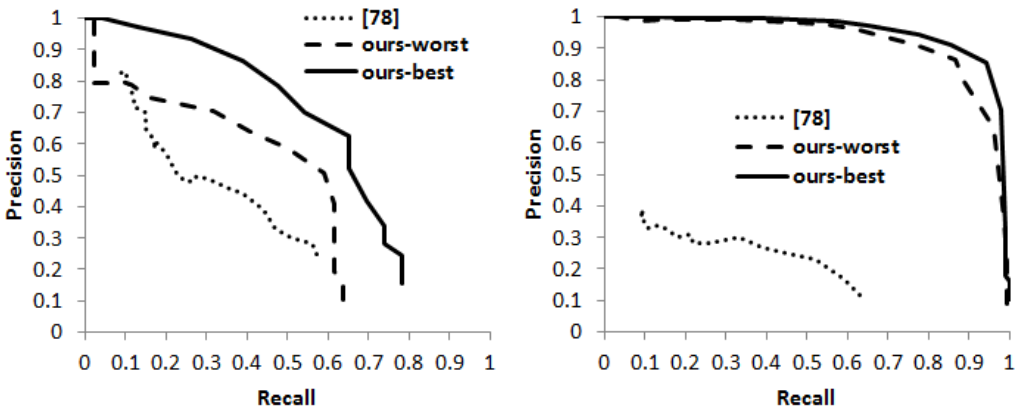
We compare our method against Damen and Hogg's [78]; note that a comparison with the more recent method due to Mitzel et al [79] is not meaningful because they use video and depth data. [78] evaluate carried object detections with a criterion requiring that the bounding box of a detected region overlap at least $15\%$ with a ground truth object bounding box. The threshold is much lower than typically used in human detection ($50\%$) in order to recognize correct matches when the protrusion is small, but this has a serious flaw – a method can return random large parts of the human silhouette and score high when most people carry objects. We remedy the criterion: a detected region is correct if it covers at least $20\%$ of a ground truth object bounding box **and** at least $66.6\%$ of its area is inside the box. We measure the performance of carried object detection methods in terms of region precision and of object track recall. Precision is defined as the fraction of regions (out of all regions eventually returned) that match ground truth and recall as the fraction of object tracks (out of all object tracks) for which there are correct detections in at least $10\%$ of the frames. We perform non-maximum suppression by removing any

region that has high pixel mask overlap with another region with higher detection score. The low recall threshold allows detections to be sparse in time (3/s), but since our method is very precise, a blob tracking extension of it can achieve both high frame level recall and high precision.

### 4.7.1 Fully Supervised Learning

Precisely annotated object bounding boxes determine labels for the image regions from the low level detectors: a region is positive if and only if it matches a box by the criteria in the preceding paragraph. In the fully supervised setting, the label of each training region is given. We perform cross validation experiments on all three datasets. We randomly divide the sets of videos 10 times into roughly half for training and half for testing; for Towncenter, the split was on persons. For each of the three datasets, Figure 4.3 shows two precision-recall curves: the curve with the smallest and the largest area among the 10 splits. The curves were obtained with $C = 100$ for the kernel SVM for all datasets; values 10 and 1000 virtually did not change any of the Pets2006 curves and two curves of Cd2a, showing no need for cross validation. The Towncenter video is especially hard because people walk in all directions, wear very diverse clothing, have vastly different body builds, and carry many types of baggage, all while the number of training regions is about twice that for Pets2006. Also, it is difficult to obtain accurate foreground masks as the scene is densely populated.

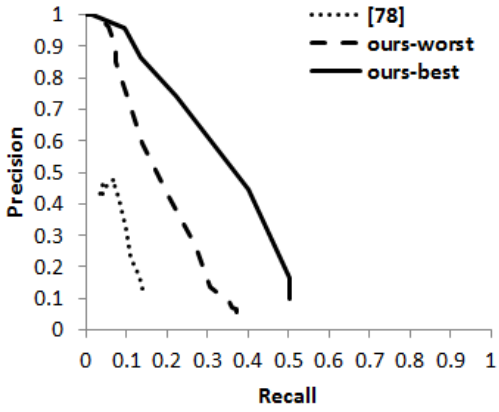
To compare against Damen and Hogg's [78] full method (using spatial prior and continuity) we modify the code made public by the authors to return correctly aligned




(a) Pets2006




(b) Cd2a




(c) Towncenter

Figure 4.3: Pairs of representative precision-recall curves for the fully supervised version of our method (solid and dashed) and the curve for [78] (dotted). The solid curve has largest area among the curves obtained on the 10 training-test splits, dashed smallest. The curves for [78] do not extend right more than shown; in particular, on Pets2006 [78] does not obtain more than 0.57 recall.

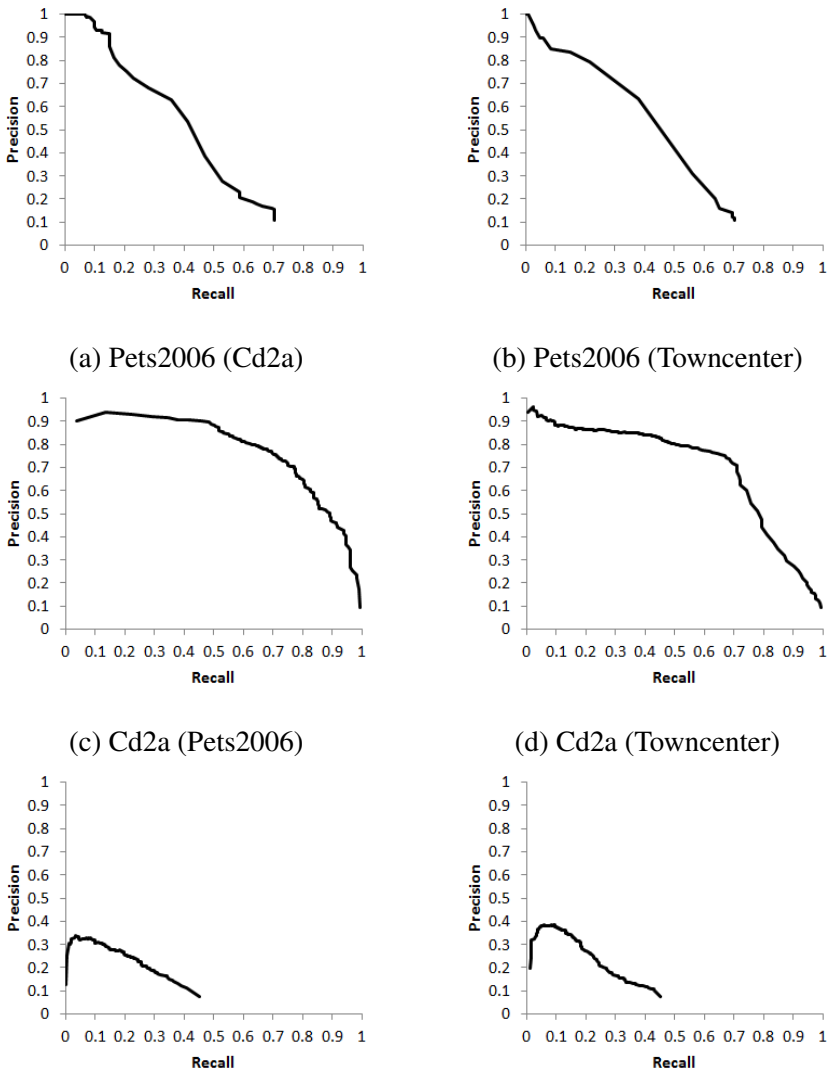
carried object pixel masks for all video frames. As was done in [78], we vary parameter $\lambda$ representing the pairwise cost in an MRF-based segmentation and trace the PR curves displayed in dotted black in Figure 4.3. Their method tends to return large parts of the human silhouette together with the carried object, which is significantly less precise than

our method. A qualitative analysis complementing numerical results can be found in the supplemental material, which we urge reviewers to consult.

Given large differences between the three datasets in the appearance of people and objects, it is legitimate to doubt that a model learned on one dataset would work well on the other two, but experiments in which we train on a complete dataset and test on the others highlight the generalization capability of our models, see Figure 4.4. The PR curves are below those obtained when training and testing on subsets of the same dataset (Figure 4.3), but good precision-recall values are achieved when we train on people wearing tight clothing and test on people wearing robes (Pets2006→Cd2a) or when we train with 4 object types and then test on more than 10 object types (Cd2a→Pets2006), for example. The models learned on Pets2006 and on Cd2a (Figure 4.4, e and f) perform poorly on Towncenter but there is strong reason to believe this is because Towncenter is more complex than Pets2006 and Cd2a: the model learned on the former tests well on both latter datasets (Figure 4.4, b and d).

### 4.7.2 Multiple Instance Learning

In this setting, the only supervision is labels "carry" and "walk" associated with intervals of human tracks, partitioning the image regions retrieved by low level detectors into positive and negative bags. We compare the peformance of miSVM [81], our extension miSVM-PFS ($\alpha_0 = 0.25$), sMIL and sbMIL [82] in Table 4.1, which includes the results of a fully supervised SVM (labels available for each image region) for reference. We use the same 10 training-test splits as in the fully supervised experiments and report the




(a) Pets2006 (Cd2a)

(b) Pets2006 (Towncenter)

(c) Cd2a (Pets2006)

(d) Cd2a (Towncenter)

(e) Towncenter (Pets2006)

(f) Towncenter (Cd2a)

Figure 4.4: Precision-recall curves when training and testing on different datasets. Format: test dataset (training dataset).

mean area under the PR curve. In many of the splits the total number $N_+$ of instances in positive bags is larger than the total number $N_-$ of instances in negative bags (sometimes drastically so), biasing classifiers towards false positives. In the SVM formulations of both miSVM and miSVM-PFS, we kept the weights of the instances in positive bags 1 and assigned weights $\frac{N_+}{N_-}$ to instances in negative bags. We set $C$ to 1 for both miSVM

| | Pets2006 | Cd2a | Towncenter |
|---|---|---|---|
| SVM (fully supervised) | 0.5238 | 0.9410 | 0.2488 |
| miSVM | 0.3526 | 0.8158 | 0.0881 |
| miSVM-PFS | 0.4098 | 0.8496 | 0.0971 |
| sMIL | 0.1413 | 0.3031 | 0.0507 |
| sbMIL | 0.3086 | 0.6878 | 0.1019 |

Table 4.1: Mean area under PR curve for different learning methods. The second row of the table shows results when object bounding boxes are available, while for the other rows only "carry" and "walk" information is given.

and miSVM-PFS; other values produce little change in the relative peformance of the two. Note that since some positively labeled bags may not actually contain any positive instances due to low level detectors failing to find any object regions, it is inappropriate to set $C$ by bag-based cross validation. For sMIL and sbMIL, we report the best mean area under PR curve over a number of parameter combinations. In particular, $\eta$ in sbMIL for the expected positive bag density ($\alpha_0$ in our work) varied in the set $\{0.1, 0.25, 0.5\}$. Table 4.1 shows the effectiveness of miSVM-PFS compared to other approaches on Pets2006 and Cd2a. All approaches perform poorly on Towncenter, confirming the difficulty of this dataset. sbMIL is slightly better on Towncenter; we attribute this to imbalance in positive bag densities (many more values close to 0 and 0.5 than to $\alpha_0 = 0.25$) due to errors in background subtraction.

The training times obtained on an Intel Core2 Quad at 3GHz for miSVM and

| dataset $\backslash \alpha_0$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 |
|---|---|---|---|---|---|
| Pets2006 | 0.2481 | 0.3791 | 0.4298 | 0.4159 | 0.3933 |
| Cd2a | 0.7065 | 0.8370 | 0.8508 | 0.8325 | 0.8221 |
| Towncenter | 0.0552 | 0.0832 | 0.0903 | 0.0959 | 0.0963 |

Table 4.2: Mean area under PR curve when the expected positive bag density in miSVM-PFS is varied.

miSVM-PFS are very similar. On Cd2a, the largest dataset, the training time averaged over the 10 splits was 20.7 minutes for miSVM and 16.3 minutes for miSVM-PFS; the average training set size over the 10 splits was 154,000 instances. The two algorithms both took about 1 minute on Pets2006 and about 10 minutes on Cd2a per training set respectively.

Because the positive bag density is only approximately known, we characterize the sensitivity of miSVM-PFS to parameter $\alpha_0$. Table 4.2 shows good mean area under the PR curve for the range $[0.1, 0.5]$; on Pets2006 and Cd2a we still outperform competing approaches, suggesting that an accurate estimate of $\alpha_0$ is not critical.

# Chapter 5: Interactive Video Segmentation Using Occlusion Boundaries and Temporally Coherent Superpixels

## 5.1 Introduction

Video segmentation is the problem of determining precise support regions for the objects in a video. A satisfactory solution would benefit important tasks like object detection from video and human action recognition, and this has motivated much research on unsupervised methods. However, the progress of these methods is fundamentally limited because video segmentation is an underconstrained problem. Motion cues typically reduce ambiguity compared to pure image segmentation, but objects can move too slowly or too quickly or can have too many articulated parts for an automated procedure to correctly separate them. It is clear that guidance from a human operator is needed, but it remains to be seen how to optimally specify it and combine it with techniques relevant to unsupervised video segmentation. Interactive video segmentation work in the Computer Graphics community [97] [98] focused on collecting user annotations to obtain perfect object boundaries in all frames. There is no guarantee that the tradeoff between segmentation quality and user effort is optimal and this suggests to explore solutions that incur a small decrease in quality for a large reduction in effort. While applications like video

editing could not use the output of such a system directly, weakly supervised learning of object appearance models from video is highly likely to improve if temporally consistent image segments are available. High level video analyses, e.g. generating qualitative descriptions of the appearance, size and trajectories of the objects in a scene, can also build on less-than-perfect segmentations.

A number of papers [98] [99] [100] cast the interactive segmentation problem as assisted contour tracking, in which the user traces the initial object contour and in subsequent frames the system propagates it and the user corrects it. This scheme does not exploit the global spatio-temporal structure of the video and as a result may require excessive supervision when resolving local ambiguities. Our system integrates spectral clustering in superpixel graphs for long time intervals with occlusion boundary detection and constraints from the user. It accepts input in the form of superpixel labels and propagates them through between-frame graph edges with high affinity that indicate reliable optical flow. When an object moves, it typically generates occlusion boundaries and the unconstrained clustering will likely produce segments that follow these boundaries; on the other hand, if an object is stationary, the unconstrained clustering can fail but the object mask will likely propagate well with optical flow. Our system collects supervision efficiently in both situations, as dynamic frames only need minor corrections and static frames, while potentially requiring more interaction, indirectly constrain the labels of numerous superpixels in adjacent frames.

Our contribution is to propose a system for interactive video segmentation that leverages long term motion information to reduce user annotation effort. On videos used to evaluate interactive segmentation systems, we obtain satisfactory results significantly

faster than existing systems [98] [99]. Other advantages of our system are that it does not assume discriminative appearance for the object(s) and that it allows extracting multiple objects at the same time, saving annotation effort if they share contours. A secondary finding of this work is the effectiveness of occlusion cues for video segmentation. On a standard dataset for unsupervised segmentation, superpixel clustering using occlusion boundaries has performance comparable with a state-of-the-art superpixel based method [101].

## 5.2 Related Work

Due to space considerations, we limit our review to the more recent papers in the video segmentation literature. Recent seminal work by Brox and Malik [85] performed unsupervised clustering of long term pixel trajectories, providing a basic algorithmic framework for authors to build on. Ochs and Brox [86] densified the trajectories by using them as seeds in a multi-level variational approach for image segmentation, and later [87] used a hypergraph to capture the similarities of trajectories on objects that rotate or scale. Fragkiadaki and Shi [90] filtered out background trajectories using motion saliency maps and examined the topology of foreground masks to define foreground trajectory dissimilarity. Later work in the same group [91] improved clustering by defining a spectral embedding space discontinuity and merging the clusters of an oversegmentation based on it; the authors also densified trajectories using Gabriel graph superpixels.

Other frequently referenced work is by Grundmann et al. [88], who adapted Felzenszwalb's agglomerative image segmentation method [89] to video and were able to output

hierarchical segments. Brendel and Todorovic [96] assumed that the image segmentations of consecutive frames vary only because of splits and merges, so they matched regions from one frame to the next and finally clustered them. A number of papers relied on generic object segmentation proposals such as [92] to produce image region seeds to be aggregated in video segmentation. Lee at al [93] created hypothesis sets from the seed regions and, using color and shape information, interpolated the object mask in the frames of the video not covered by a set. Ma and Latecki [94] grouped the seeds by solving a maximum weight cliques problem which allowed enforcing common sense constraints. Zhang et al. [95] augmented the seed set with regions from adjacent frames obtained by propagation with optical flow. They solved a longest path problem in the directed acyclic graph of proposals, in which edge costs characterize both motion and appearance similarity.

Many methods compute oversegmentations of each video frame and then cluster superpixels. Levinshtein et al. [102] generated temporally coherent superpixels, built a superpixel graph, and hypothesized multiple node groupings with parametric maxflow. We adopt Levinshtein et al.'s method to generate superpixels, but take occlusion boundaries into account in the affinities of edges between superpixels. Galasso et al. [101] evaluated combinations of superpixel affinity types to be used in a spectral clustering framework. While they obtained good performance on the Berkeley moseg dataset, it is not clear if in general the affinity between two superpixels in the same frame can carry more meaning than the presence or absence of an occlusion boundary between the superpixels. Vazquez-Reina et al. [103] computed multiple superpixel segmentations for each frame, generated plausible superpixel tracks and finally labeled individual superpix-

els as belonging to one of the tracks using a higher order CRF that enforces photometric consistency and exclusiveness constraints. Their method can lead to severe oversegmentation, as there is no mechanism to restrict splitting a single object into arbitrarily many superpixel track candidates.

At the border of the category of unsupervised and interactive methods is work that assumes the existence of a predefined set of pixel labels. Yuen et al. [107] interpolated polygons representing object contours with a simple 3D motion model. Wang and Collomosse [104] propagated labels from frame to frame and used probabilistic optical flow in order to construct more accurate labeling priors for pixels in the next frame. Badrinarayanan et al. [105] modeled the evolution of both the image data and the labels in a video frame with a common set of latent patches and, given the labels in the start and end frame of a video, they inferred the labels in between. Vijayanarasimhan and Grauman's work [106] is closer to interactive video segmentation: they estimated label propagation errors in order to select those video frames whose complete annotation minimizes total user effort (the remaining frames still require annotation, but for refinement). This procedure cannot be directly applied in our framework because the time it costs to label a frame (which they assume fixed) varies drastically according to how successful the initial unsupervised segmentation is.

In one of the earlier works in interactive video segmentation, Wang et al. [97] employed hierarchical segmentation to shrink the graph input to a min-cut optimizer, and they extended a 2D alpha matting scheme to work on video volumes. Recognizing the difficulty of building a global model to discriminate between foreground and background pixels, Bai et al. [98] developed a system that used multiple local color and shape classi-

fiers, each valid in a small support region on the object boundary. The object contour was automatically predicted by moving the regions according to optical flow and applying the classifiers; the user's correction served to update the support regions and classifiers. Later work by authors in the same group [100] modeled foreground and background appearance in a scale adaptive manner, by updating color space Gaussians for individual pixels using probabilistic optical flow. Price et al. [99] proposed additional types of local classifiers, such as color adjacency relations, but more importantly they dynamically weighted them to adapt to change in foreground and background appearance. Updating the object contour frame-by-frame with help from the user is intuitive, but significant supervision may be required to compensate for using such small temporal neighborhoods. Large motions and locally similar foreground and background appearances pose great difficulty to such systems.

## 5.3 System Description

Our system performs an unsupervised clustering in the superpixel graph corresponding to a video interval and then repeated supervised clusterings with more and more annotation from the user. First, we compute optical flow [64] forward and back and then temporally coherent superpixel segmentations for each video frame [102]. Next, a simple occlusion detector is run that finds occluded pixels as those with forward flow inconsistent with backward flow or with large flow gradient magnitude [63]. With respect to [63], we tighten the first condition and loosen the second, requiring more consistency but allowing

for larger gradient magnitudes:

$$||\mathbf{w}(\mathbf{p}) + \mathbf{w}'(\mathbf{p}')||_2^2 > 0.01(||\mathbf{w}(\mathbf{p})||_2^2 + ||\mathbf{w}'(\mathbf{p}')||_2^2) + 0.01 \tag{5.1}$$

$$||\nabla u(\mathbf{p})||_2^2 + ||\nabla v(\mathbf{p})||_2^2 > 0.01||\mathbf{w}(\mathbf{p})||_2^2 + 0.01 \tag{5.2}$$

where $\mathbf{p}$, $\mathbf{p}'$ are pixel coordinates, $\mathbf{w}$ and $\mathbf{w}'$ are the forward and backward optical flow with components $u$ and $v$ ($\mathbf{p}' = \mathbf{p} + \mathbf{w}(\mathbf{p})$).

### 5.3.1 Graph Construction

We divide the input video into non-overlapping time intervals of 100 frames (or less, in the last interval) and run the system independently on each of them. Each interval is represented with a graph of superpixels which has two types of edges: within-frame edges that link superpixels in the same frame and between-frame edges that link superpixels in temporally adjacent frames. There are within-frame edges only between superpixels that share pixel borders in the image. Each superpixel has at most one between-frame edge linking it to a superpixel in the next frame, depending on the output of the occlusion detector. If the superpixel has pixels that are not occluded, then its center is shifted with the mean optical flow and it is linked to the superpixel with the closest center in the next frame. If all its pixels are occluded, then it is not linked to any other superpixel in the next frame (note that it will be linked with superpixels in the current frame and possibly with a superpixel in the previous frame). We do not consider 2-layer spatial neighborhoods or temporal edges across multiple video frames, like [101]. Segmentation results with this type of graph are slightly more robust, but the computation is noticeably slower due to the affinity matrix used by spectral clustering becoming more dense. For the videos we

test our system on, the maximum number of superpixels in an interval is approximately $130,000$, so it is computationally infeasible to work with a complete graph/affinity matrix. However, the more important difficulty is that it is unclear how to define a meaningful distance measure for superpixels many pixels or many frames apart.

We define the distance between the two superpixels of a within-frame edge in a similar way to [102]:

$$d^w_{i,j} = min\left(1, \frac{||\overline{\mathbf{w}}_i - \overline{\mathbf{w}}_j||_2}{1 + max(||\overline{\mathbf{w}}_i||_2, ||\overline{\mathbf{w}}_j||_2)}\right) \tag{5.3}$$

where $\overline{\mathbf{w}}_i$ is the mean optical flow of the pixels that are not occluded in superpixel $i$. If all pixels in a superpixel are occluded, then the mean flow is arbitrarily set to 0, but the graph modifications described in the paragraph below remedy the situation. The difference from [102] is the 1 in the denominator, which makes the distance tend to 0 if both motions are small. We define the distance between the two superpixels of a between-frame edge as the average of an overlap distance and a color distance:

$$d^b_{i,j} = \frac{d^o_{i,j} + d^c_{i,j}}{2} \tag{5.4}$$

The overlap distance $d^o_{i,j}$ is the 1-complemented intersection-over-union ratio of the support of the earlier superpixel shifted with its mean flow and the support of the later superpixel:

$$d^o_{i,j} = 1 - \frac{|\{\mathbf{p} + round(\mathbf{w}_i) | \mathbf{p} \in \mathbf{s}_i\} \cap \mathbf{s}_j|}{|\{\mathbf{p} + round(\mathbf{w}_i) | \mathbf{p} \in \mathbf{s}_i\} \cup \mathbf{s}_j|} \tag{5.5}$$

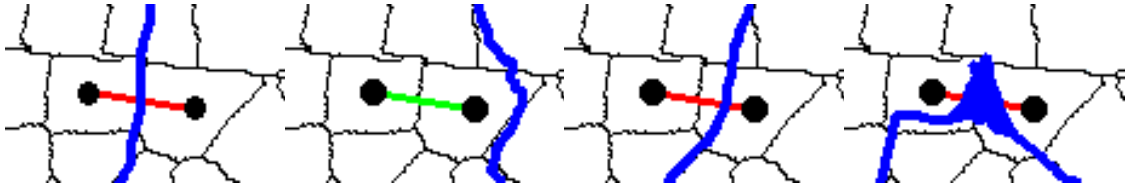
The color distance $d^c_{i,j}$ is the 1-complemented histogram intersection value of the normalized 8-bin rgb histograms of the superpixels. In most situations, the overlap distance is by itself adequate for the purpose of $d^b_{i,j}$, which is to characterize confidence in a superpixel's

most likely temporal successor. The color distance was included to prevent superpixels with erroneous mean optical flow from matching random similar shaped superpixels in the next frame, which can lead to superpixel tracks leaking across objects.

Both $d^w$ and $d^b$ are in the range $[0, 1]$ and are converted to affinities with a Gaussian kernel with standard deviation $3$. In practice, many affinities are very close 0, so we cap the affinities from below with $\epsilon = 0.01$ to avoid having noisy superpixel sets with normalized cut very close to 0. Since occlusion boundaries are correlated with different objects moving separately, we set the within-frame edges of superpixels that are cut by an occlusion boundary to affinity $\epsilon$ as well, regardless of the distance defined in Equation 5.3. We determine that a pair of superpixels is cut by an occlusion boundary or not based on the connected components that can be formed with pixels that are not occluded, see Figure 5.1. If the pixel component(s) inside a superpixel that reach the superpixel border are smaller than half the size of the superpixel, then the superpixel pair is cut, the reasoning being that occlusion effectively splits one of the superpixels. The criterion is not perfect (Figure 5.1d), but it ensures that even if an occlusion boundary does not run exactly along a superpixel border, it will still influence the the graph structure.

### 5.3.2 Clustering with Constraints

We use the spectral clustering algorithm described by Ng et al [108], which eigendecomposes the symmetrically normalized graph laplacian $L = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ (where $A$ is the affinity matrix), row-normalizes the eigenvectors and runs k-means in this space. User input was converted into must-link and cannot-link constraints on pairs of nodes



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

Figure 5.1: Four representative occlusion cases for superpixels linked by a within-frame edge. (a) Ideally, the occlusion boundary (blue) coincides with the superpixel border. (b) The occlusion boundary splits the superpixel on the right into a large component near the superpixel on the left and a small component away from it, so the superpixel pair is not cut. (c) The component near the border is small and the occlusion boundary resembles that in (a), so the pair is cut. (d) Errors of the occlusion boundary detector near the superpixel border lead to the pair of superpixels being cut.

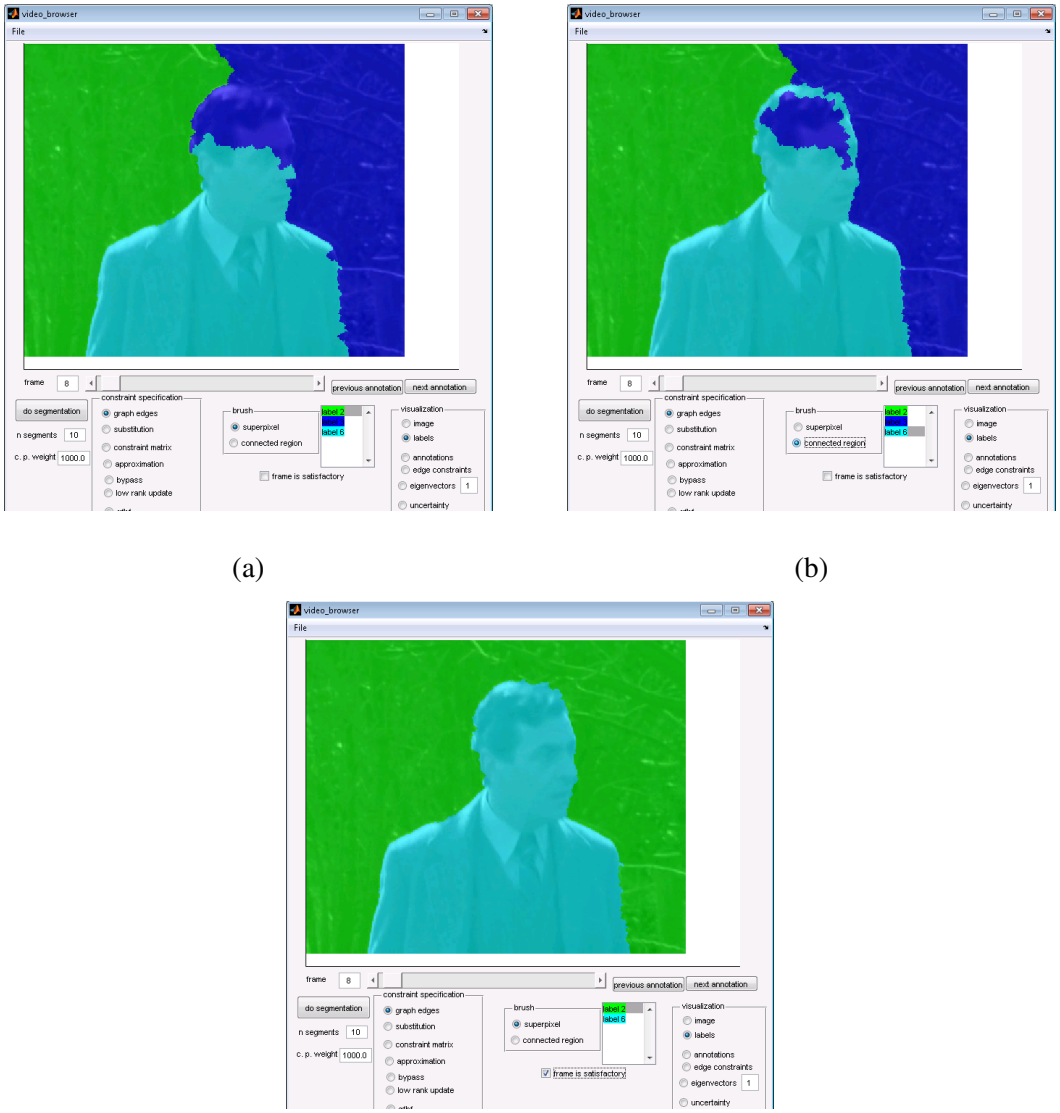
already linked by edges, essentially projecting all annotation information on the original graph structure. The constraints were integrated in a straightforward way into the clustering scheme by changing the affinities to a large value (1000) for must-link and to 0 for cannot-link. This makes spectral clustering strongly prefer to cut the cannot-link edges, since all the others in the original graph have non-zero cost $\epsilon$, and to avoid cutting must-link edges. Although many constrained spectral clustering methods exist in the literature [109, 111, 110, 113], they target settings in which the graph is relatively small (hundreds or thousands of nodes) and complete and there are few constraints; many work for 2 class tasks only. For our video segmentation problem, the graph is large (tens of thousands or hundreds of thousands nodes) and sparse, there can be multiple classes (background and more than one object) and the user indirectly specifies many constraints (e.g. $10\%$

of graph edges) at once. Rangapuram and Hein's method [112] works on large graphs, but they implement multiclass clustering by recursive partitioning of the original graph, which is not compatible with the structure of our problem.

We set the number of clusters to 10, a value larger than the number of actual objects in each of the videos of our dataset. Since the graphs we encounter in practice frequently have node subsets with normalized cut value lower than for actual objects, we do not focus on extracting the correct number of segments, and prompt the user to assign the over-segmented clusters to objects at the end. The amount of extra work is negligible, as there are only 10 possible cluster labels. It would have been possible to allow the user to change the number of clusters during the interaction, but this can be unintuitive for a non-computer vision/machine learning expert and the total interaction time would have been strongly user-dependent.

### 5.3.3 User Interface

The interface continuously displays a proposed segmentation of the video, a completely unsupervised one in the beginning or a supervised one, following user input. The user navigates through the video to decide which frame(s) to correct, makes the corrections and reruns the segmentation, iterating these steps until a satisfactory result is achieved for the entire video. Corrections are specified as new superpixel labels: starting with the previous clustering result, the user clicks to change the labels of single superpixels or of entire connected regions of superpixels. In the example in Figure 5.2, the user fixes two superpixels on the left arm of the person, traces the contour of the head, which was




(a) (b)

(c)

Figure 5.2: (a) The user browses through the video and finds a frame that is relatively easy to correct. The user selects a target label and changes superpixels to this label, in "superpixel" or "connected regions" mode. "superpixel" means only the clicked superpixel changes; "connected regions" means all superpixels in the connected component of the clicked superpixel change. (b) The head contour is completely traced and what is left is to change the mode to "connected regions" and fill holes. (c) A checkbox allows to inform the system that the current frame labeling is satisfactory.

wrongly segmented by the system, and finally fills in the two holes. Once the labeling becomes satisfactory, the user informs the system (Figure 5.2c, bottom center of the window); s/he can then request a segmentation given the labels entered so far, or can proceed to correct other frames. In our experiments, not more than 10 frames needed to be labeled in any given 100 frame interval.

The labels of the superpixels in the satisfactory frames are propagated temporally via the between-frame edges of the graph. If a labeled node A has distance less than $0.25$ to an unlabeled node B and it is the closest among B's temporal neighbors, then B receives A's label. (This is a stringent criterion, but in practice many labels are propagated even for videos with moderate amounts of motion.) From a frame marked as satisfactory the propagation proceeds frame by frame, first forward and then backward in time. There could be multiple satisfactory frames propagating labels to the same node, but we choose to bypass the issue of establishing correspondences between the labels in different satisfactory frames. Instead, we directly derive constraints on the original graph edges. There are three possible relations between the two labels propagated to the nodes of an edge: same (resulting in a must-link constraint), different (cannot-link) and unknown (at least one of the propagated labels is missing, no constraint). If two satisfactory frames contradict themselves in the relation of labels they propagate at an edge (same and different), then that edge does not generate a constraint. Otherwise, the constraint is set according to the common relation that is different from unknown (if the only common relation is unknown, then there is no constraint on the edge).

## 5.4 Experimental Results

### 5.4.1 Interactive Video Segmentation

For interactive video segmentation, we run our method on four of the videos used by [99]. Our method currently needs preprocessing time on the order of hours for a video, mostly because of optical flow computation. However, this could be easily reduced to minutes by running on a computer cluster or by using faster optical flow implementations. Furthermore, if segmentation is needed for a batch of videos, computationally heavy preprocessing can be run on a different machine in parallel with the interactive annotation by the user. Running spectral clustering on 100 frame intervals of the videos takes from 3 to 45 seconds on an Intel Xeon E5 @3GHz equipped with 8GB of RAM (it can take less time if more constraints are available). To minimize the user's wait, the initial (unconstrained) spectral clustering is done in the preprocessing stage and the user corrects 3-5 well spaced out in time frames before asking the system to re-segment. In practice we observe that good segmentation results can typically be achieved by labeling a total of 5-10 frames in each interval of 100 frames. Figure 5.3 shows sample segmentations for videos "lemurs", "elephant", "manincap" and "stairs". In Table 5.1 we compare running times (including segmentation wait time) with those of [99].

### 5.4.2 Unsupervised Video Segmentation Using Occlusion Boundaries

To verify that occlusion boundaries are effective for video segmentation purposes, we compare our approach to [101] on the Berkeley motion segmentation dataset [85]. The

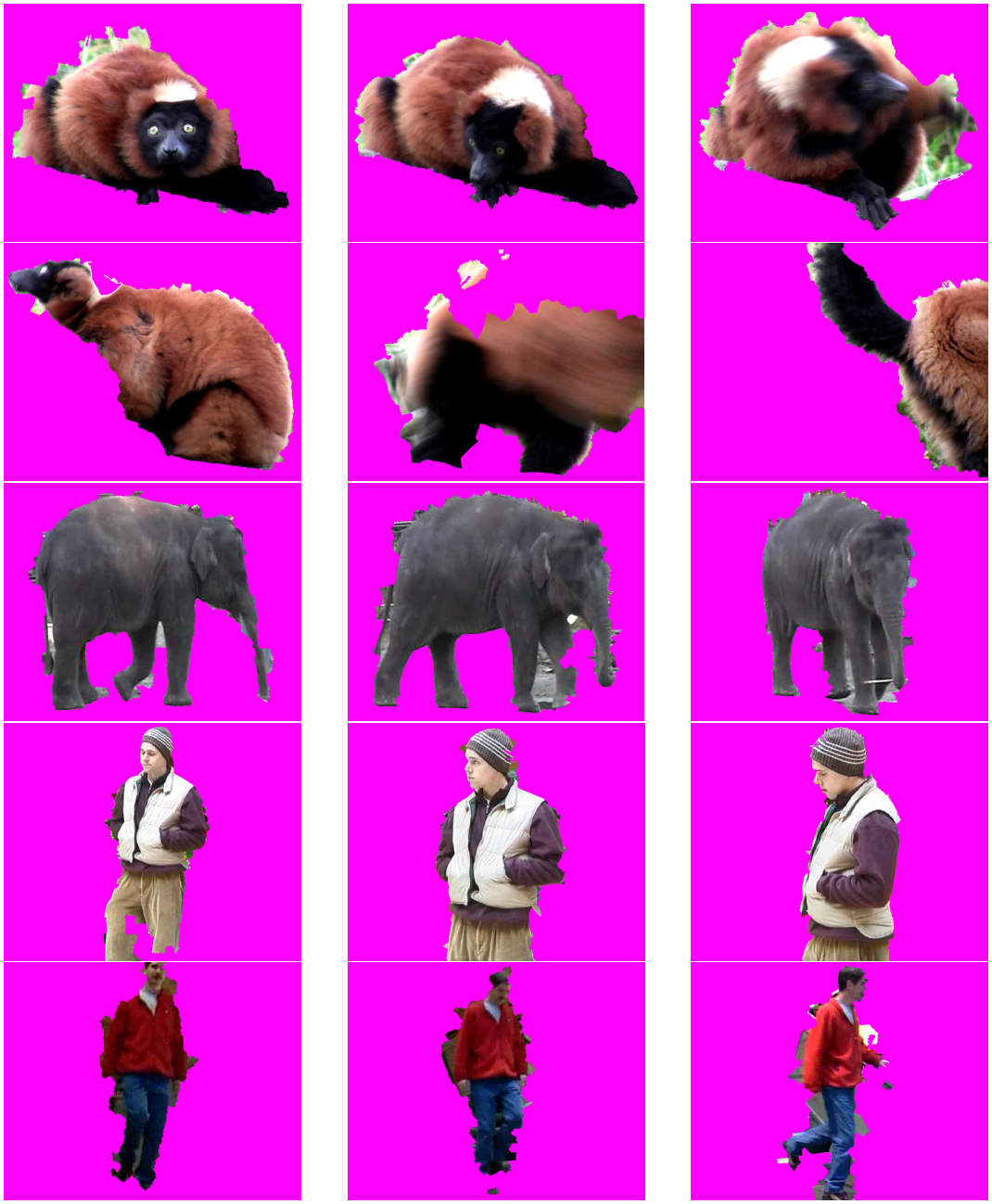


Figure 5.3: Interactive video segmentation results. The “lemurs” sequence used by [99] contains two foreground objects (first and second row). While the masks are not perfect, they are obtained faster than with state of the art systems. Other than superpixel borders not coinciding with object contours, reasons for errors include heavy motion blur (second row, second column: only the bottom part of the animal is retrieved) and contours with inlets (third row, second and third columns).

| Video | Frames | Livecut user time (mins) | Our user time (mins) |
|---|---|---|---|
| lemurs | 86 | 38 | 12 |
| elephant | 100 | 38 | 13 |
| manincap | 150 | 23 | 11 |
| stairs | 63 | 13 | 7 |

Table 5.1: Comparison of running times obtained by [99] and our system. Note that we extract both foreground objects in the "lemurs" sequence while [99] extracts just one.

dataset consists of 26 sequences showing relatively simple traffic scenes (10 videos), short clips from the movie "Ms. Marple" (13 videos), 2 pedestrian scenes, and a video of a tennis player. Challenges include drastic (though not extremely fast) camera motion, occlusions, objects sweeping across the screen, objects that are stationary for long periods of time and highly non-rigid motion. We report results obtained with the evaluation software accompanying the dataset. Given the proposed segmentation of a video, the software matches its segments with the ground truth segments available in a few hand annotated frames and then computes five metrics reflecting the quality of the proposal: density, oversegmentation, overall error, average error and number of extracted objects. The density is the fraction of voxels covered by segments (the proposed segmentation is allowed to be incomplete). The oversegmentation is the average number of proposed segments assigned to the same ground truth segment (ideally 1). The overall error is the fraction of voxels covered by the wrong segment, given the assignment of proposed segments to ground

truth segments. The average error is also related to voxels covered by the wrong segment, but it is an average fraction over ground truth segments instead of over the entire video. The number of extracted objects is the number of proposed segments matching ground truth segments (except the background) with less than $10\%$ error.

The evaluation software was run on the first 100 frames of each video, or the length of the video if it was shorter, to compare with [101]. Results are shown in Table 5.2. Our method's density is slightly less than $100\%$ because we eliminate small groups of superpixels that have small affinity with the rest of the superpixels in the graph (in theory, they should not affect spectral clustering, but in practice this step is necessary for better results). At virtually the same oversegmentation rate, our method has slightly higher overall and average error, but also slightly higher number of extracted objects, putting its performance on par with [101]. The last row of the table shows results when the affinities of superpixel pairs cut by occlusion boundaries are not modified. The almost 2-fold increase in both overall and average error demonstrates the importance of occlusion boundaries as cue for video segmentation.

| Method | Density | Overseg-mentation | Overall Error | Average Error | Extracted Objects |
|---|---|---|---|---|---|
| Galasso et al. [101] | 100% | 6.77 | 9.92% | 16.52% | 17 |
| Ours | 99.31% | 6.65 | 12.35% | 23.38% | 19 |
| Ours w/o occlusion boundaries | 99.71% | 6.54 | 22.36% | 44.17% | 9 |

Table 5.2: Results obtained with our Galasso et al.'s method versus ours. While [101] use complex combinations of superpixel distance measures, we employ simple distances but integrate input from an occlusion boundary detector in the superpixel graph construction. These distances perform poorly by themselves, as the last row shows.

# Chapter 6: Conclusions and Future Research Directions

## 6.1 Action recognition based on the characteristics of human movement in image space

We presented a novel motion descriptor for action recognition based on human movement characteristics. Our approach assumes short and small correlated linear movements during an action and models motion vectors with a probability distribution over the Hough space augmented with position information. We used a feature point tracker to obtain motion vectors, formed motion vector groups, aggregated group statistics into mixtures of Gaussians, and measured the similarity of two such pdf's with the Bhattacharyya coefficient.

We demonstrated good recognition performance on the standard KTH action database, as well as on a locally acquired database on which approaches that use appearance and shape are bound to fail. Our descriptor has very low computational requirements and can handle both short and long actions.

## 6.2 Kernel PLS regression for robust monocular pose estimation

We studied the performance of five regression methods for pose estimation on four datasets with synthetically generated images, and showed that kPLS regression, not previously considered by the pose community, outperforms other methods in terms of robustness. While most methods for discriminative pose estimation are designed to address the difficulty of insufficient training data, we focused on characterizing accuracy improvement as the training set becomes more dense, to further the understanding of the advantages and limitations of five pose estimation methods. kPLS regression can be viewed as a robust incremental approximation of GP regression, and we verified this observation through quantitative experiments. Our results also indicate that investigating new features and new feature space metrics holds significant potential to improve pose estimation accuracy.

## 6.3 Learning to detect carried objects with minimal supervision

We proposed a learning-based method for carried object detection that finds objects even when they do not protrude, achieves high accuracy, and has good generalization capabilities. Our method obtains candidate image regions from three cues (protrusions, color contrast and occlusion boundaries) and selects the plausible object regions with a kernel SVM classifier on features characterizing the region properties and the context of the human silhouette. To avoid annotating tens of thousands of carried object bounding boxes, we investigated training the classifier in a MIL framework which only required hundreds

of “walk” and “carry” labels for intervals of human tracks. We extended the miSVM algorithm [81] to effectively account for a known fraction of positive instances in positive bags and this extension consistently improved accuracy while keeping computational cost low.

## 6.4 Interactive video segmentation using occlusion boundaries and temporally coherent superpixels

We built a system for interactive video segmentation on the base of occlusion and spatio-temporal structure cues. With respect to other systems, ours provides a large reduction in the amount of user supervision for a small degradation of the segmentation results. Our system has the advantages that it does not rely on a discriminative object appearance model and allows extracting multiple foreground objects together (in other systems, extracting each object is a separate task). Additional experiments with unsupervised clustering based on occlusion boundaries highlight the importance of this cue for video segmentation.

## 6.5 Future Work

The motion descriptor for action recognition uses a single reference point at the top left corner of the image, but other image locations could be used. In the future, we will investigate selecting the location automatically and aggregating the descriptors from multiple locations. Also, by normalization with a quantity related to actor size, the descriptor can be extended to be scale invariant. Finally, since the descriptor relies only on movement,

there is potential to extend it by incorporating appearance and shape information.

We plan to apply kPLS regression to learn multiple separate models on pose regions with unambiguous image observations, for multimodal pose prediction. We speculate that in addition to handling ambiguity, this method will be more precise at region borders than other partitioning-based pose estimation methods, because of the robustness of kPLS regression. Another direction we will work in is combining synthetically generated training instances with pose-image pairs acquired in motion capture sessions, with the goal of improving generalizability when estimating pose from real images.

For future carried object detection work, we intend to evaluate the importance of the existing features and to investigate new features, potentially drawing inspiration from the image cosegmentation literature. Since there is a performance gap between the fully supervised method and the weakly supervised one, it would be interesting to extend the system by combining MIL with active learning. After the weakly supervised method runs, another round of training can query the labels of instances in positive bags that are the closest to the decision surface or that would lead to satisfying MIL constraints, or both.

In future video segmentation work, we will incorporate a more sophisticated occlusion boundary detector and refine object masks with graph cuts. Extensions are possible at higher levels as well, such as automatically selecting frames for the user to correct according to an estimate of the labeling time balanced against utility for segmentation. A faster scheme for constrained spectral clustering or for incremental constrained spectral

clustering would shorten the time taken to respond to re-segmentation requests and provide for a better interactive experience.

# Chapter A: List of Publications

R. Dondera, D. Doermann, L. Davis, Action Recognition based on Human Movement Characteristics. IEEE Workshop on Motion and Vision Computing, 2009.

R. Dondera, L. Davis, Kernel PLS Regression for Robust Monocular Pose Estimation. CVPR Workshop on Machine Learning for Vision-Based Motion Analysis, 2011.

R. Dondera, V. Morariu, L. Davis, Learning to Detect Carried Objects with Minimal Supervision. CVPR Workshop on Socially Intelligent Surveillance and Monitoring, 2013.

# Bibliography


[1] S.N.P. Vitaladevuni, V. Kellokumpu, L.S. Davis, Action recognition using ballistic dynamics. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[2] A.A. Efros, A. C. Berg, G. Mori, J. Malik, Recognizing Action at a Distance. IEEE International Conference on Computer Vision, 2003.

[3] Y. Ke, R. Sukthankar, M. Hebert, Efficient Visual Event Detection Using Volumetric Features. IEEE International Conference on Computer Vision, 2005.

[4] A. Fathi, G. Mori, Action recognition by learning mid-level motion features. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[5] J. Shi, C.Tomasi, Good features to track. IEEE Conference on Computer Vision and Pattern Recognition, 1994.

[6] T.-K. Kim, R. Cipolla, Canonical Correlation Analysis of Video Volume Tensors for Action Categorization and Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2008.

[7] K. Mikolajczyk, H. Uemura, Action recognition with motion-appearance vocabulary forest. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[8] I. Laptev, B. Caputo, C. Schuldt, T. Lindeberg, Local velocity-adapted motion events for spatio-temporal recognition. Computer Vision and Image Understanding, 2007.

[9] C. Schuldt, I. Laptev, B. Caputo, Recognizing human actions: A local SVM approach. International Conference on Pattern Recognition, 2004.

[10] A. F. Bobick, J. W. Davis, The Recognition of Human Movement Using Temporal Templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001.

[11] R. Cutler, L. Davis, Robust real-time periodic motion detection, analysis, and applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

[12] I. Laptev, T. Lindeberg, Space-time interest points. IEEE International Conference on Computer Vision, 2003.

[13] P. Dollar, V. Rabaud, G. Cottrell, S. Belongie, Behavior Recognition via Sparse Spatio-Temporal Features. VS-PETS, 2005.

[14] J.C. Niebles, H. Wang, L. Fei-Fei, Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. International Journal of Computer Vision, 2008.

[15] H. Jhuang, T. Serre, L. Wolf, T. Poggio, A Biologically Inspired System for Action Recognition. IEEE International Conference on Computer Vision, 2007.

[16] Y. Song, L. Goncalves, P. Perona, Unsupervised learning of human motion. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003.

[17] S. Ali, A. Basharat, M. Shah, Chaotic Invariants for Human Action Recognition. IEEE International Conference on Computer Vision, 2007.

[18] R. Fablet, P. Bouthemy, Motion recognition using nonparametric image motion models estimated from temporal and multiscale co-occurrence statistics. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003.

[19] C. Bregler, Learning and Recognizing Human Dynamics in Video Sequences. IEEE Conference on Computer Vision and Pattern Recognition, 1997.

[20] `http://www.ces.clemson.edu/~stb/klt/`

[21] P.V.C. Hough, Machine Analysis of Bubble Chamber Pictures. Proc. Int'l Conf. High Energy Accelerators and Instrumentation, 1959.

[22] V. Parameswaran, R. Chellappa, View Invariants for Human Action Recognition. IEEE Conference on Computer Vision and Pattern Recognition, 2003.

[23] M. Hu, S. Ali, M. Shah, Detecting global motion patterns in complex videos. International Conference on Pattern Recognition, 2008.

[24] J. Yan, M. Pollefeys, A General Framework for Motion Segmentation: Independent, Articulated, Rigid, Non-rigid, Degenerate and Non-degenerate. European Conference on Computer Vision, 2006.

[25] H. Jiang, D.R. Martin, Finding Actions Using Shape Flows. European Conference on Computer Vision, 2008.

[26] K. Grauman, T. Darrell, The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[27] Q. Fan, R. Bobbitt, Z. Yun, A. Yanagwa, S. Pankanti, A. Hampapur, Recognition of Repetitive Sequential Human Activity. IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[28] B. Laxton, J. Lim, D.J. Kriegman, Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[29] Z. Zhang, K. Huang, T. Tan, Multi-thread Parsing for Recognizing Complex Events in Videos. European Conference on Computer Vision, 2008.

[30] A. Kanaujia, C. Sminchisescu, D. Metaxas, Semi-Supervised Hierarchical Models for 3D Human Pose Reconstruction. IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[31] A. Bissacco, M.H. Yang, S. Soatto, Fast Human Pose Estimation using Appearance and Motion via Multi-Dimensional Boosting Regression. IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[32] A. Agarwal, B. Triggs, Recovering 3D Human Pose from Monocular Images. Pattern Analysis and Machine Intelligence, 2006.

[33] R. Urtasun, D.J. Fleet, A. Hertzmann, P. Fua, Priors for People Tracking from Small Training Sets. IEEE International Conference on Computer Vision, 2005.

[34] R. Urtasun, T.J. Darrell, Sparse probabilistic regression for activity-independent human pose inference. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[35] C. Sminchisescu, A. Kanaujia, Z. Li, D. Metaxas, Discriminative Density Propagation for 3D Human Motion Estimation. IEEE Conference on Computer Vision and Pattern Recognition, 2005.

[36] W.R. Schwartz, A. Kembhavi, D. Harwood, L.S. Davis, Human Detection using Partial Least Squares Analysis. IEEE International Conference on Computer Vision, 2009.

[37] C. Bregler and J. Malik, Tracking People with Twists and Exponential Maps. IEEE Conference on Computer Vision and Pattern Recognition, 1998.

[38] A. Gupta, T. Chen, F. Chen, D. Kimber, L.S. Davis, Context and observation driven latent variable model for human pose estimation. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[39] R. Okada, S. Soatto, Relevant Feature Selection for Human Pose Estimation and Localization in Cluttered Images. European Conference on Computer Vision, 2008.

[40] L. Bo, C. Sminchisescu, Twin Gaussian Processes for Structured Prediction. International Journal of Computer Vision, 2010.

[41] A. Ranganathan, M.-H. Yang, Online Sparse Matrix Gaussian Process Regression and Vision Applications. European Conference on Computer Vision, 2008.

[42] T. Yan, L. Sigal, B. Hernan, F. De la Torre, Y. Liu, Latent Gaussian Mixture Regression for Human Pose Estimation. Asian Conference on Computer Vision, 2010.

[43] G. Shakhnarovich, P. Viola, T. Darrell, Fast Pose Estimation with Parameter-Sensitive Hashing. IEEE International Conference on Computer Vision, 2003.

[44] J. Deutscher, A. Blake, I. Reid, Articulated Body Motion Capture by Annealed Particle Filtering. IEEE International Conference on Computer Vision, 2000.

[45] R. Rosales, S. Sclaroff, Inferring Body Pose without Tracking Body Parts. IEEE Conference on Computer Vision and Pattern Recognition, 2000.

[46] `http://smithmicro.com/Poser`

[47] `http://mocap.cs.cmu.edu`

[48] R. Navaratnam, A.W. Fitzgibbon, R. Cipolla, The Joint Manifold Model for Semi-supervised Multi-valued Regression. IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[49] M. Salzmann, R. Urtasun, Combining discriminative and generative methods for 3D deformable surface and articulated pose reconstruction. IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[50] C.-S. Lee, A. Elgammal, Coupled Visual and Kinematic Manifold Models for Tracking. International Journal of Computer Vision, 2010.

[51] A. Fossati, M. Salzmann, P. Fua, Observable subspaces for 3D human motion recovery. IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[52] J. Gall, A. Yao, L. Van Gool, 2D action recognition serves 3D human pose estimation. European Conference on Computer Vision, 2010.

[53] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). 2005.

[54] R. Rosipal, L. J. Trejo, Kernel partial least squares regression in reproducing kernel hilbert space. Journal of Machine Learning Research, 2002.

[55] L. Sigal, M.J. Black, HumanEva: Synchronized Video and Motion Capture Dataset for Evaluation of Articulated Human Motion. Technical Report CS-06-08 Brown University, 2006.

[56] K. Kim, T.H. Chalidabhongse, D. Harwood, L. S. Davis, Real-time foreground-background segmentation using codebook model. Real-Time Imaging, 2005.

[57] `www.darpa.mil/Our_Work/I2O/Programs/Minds_Eye.aspx`

[58] B. Benfold, I. Reid, Stable Multi-Target Tracking in Real-Time Surveillance Video. IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[59] J. Ferryman, WPETS, 2006.

[60] S. Belongie, J. Malik, J. Puzicha, Shape Matching and Object Recognition Using Shape Contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.

[61] B. Alexe, T. Deselaers, V. Ferrari, Measuring the Objectness of Image Windows. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.

[62] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines. ACM TIST. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[63] N. Sundaram, T. Brox, K. Keutzer, Dense point trajectories by GPU-accelerated large displacement optical flow. European Conference on Computer Vision, 2010.

[64] D. Sun, S. Roth, M.J. Black, Secrets of optical flow estimation and their principles. IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[65] N.M. Ghanem, L.S. Davis, Human Appearance Change Detection. ICIAP, 2007.

[66] A. Fathi, X. Ren, J.M. Rehg, Learning to recognize objects in egocentric activities. IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[67] S. Vijayanarasimhan, K. Grauman, Keywords to visual categories: Multiple-instance learning for weakly supervised object categorization. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[68] W. Li, L. Duan, I.W.-H. Tsang, D. Xu, Batch mode adaptive Multiple Instance Learning for computer vision tasks. IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[69] P.V. Gehler, O. Chapelle, Deterministic Annealing for Multiple-Instance Learning. Journal of Machine Learning Research, 2007.

[70] Y. Han, Q. Tao, J. Wang, Avoiding False Positive in Multi-Instance Learning. Neural Information Processing Systems, 2010.

[71] O. Maron, T. Lozano-Prez, A Framework for Multiple-Instance Learning. Neural Information Processing Systems, 1998.

[72] W. Li, L. Duan, D. Xu, I.W.-H. Tsang, Text-Based Image Retrieval using Progressive MIL. IEEE International Conference on Computer Vision, 2011.

[73] Y. Chen, J. Bi, J.Z. Wang, MILES: Multiple-Instance Learning via Embedded Instance Selection. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006.

[74] T. Senst, A. Kuhn, H. Theisel, T. Sikora, Detecting People Carrying Objects Utilizing Lagrangian Dynamics. IEEE International Conference on Advanced Video and Signal-Based Surveillance, 2012.

[75] C.-S. Lee, A. Elgammal, Carrying object detection using pose preserving dynamic shape models. AMDO, 2006.

[76] I. Haritaoglu, R. Cutler, D. Harwood, L.S. Davis, Backpack: Detection of People Carrying Objects Using Silhouettes. Computer Vision and Image Understanding, 2001.

[77] D. Tao, X. Li, X. Wu, S.J. Maybank, Human carrying status in visual surveillance. IEEE Conference on Computer Vision and Pattern Recognition, 2006.

[78] D. Damen, D. Hogg, Detecting Carried Objects from Sequences of Walking Pedestrians. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.

[79] D. Mitzel, B. Leibe, Taking Mobile Multi-object Tracking to the Next Level. European Conference on Computer Vision, 2012.

[80] T.G. Dietterich, R. H. Lathrop, T. Lozano-Perez, Solving the multiple instance problem with axis-parallel rectangles. AI, 1997.

[81] S. Andrews, I. Tsochantaridis, T. Hofmann, Support Vector Machines for Multiple-Instance Learning. Neural Information Processing Systems, 2002.

[82] R.C. Bunescu, R.J. Mooney, Multiple Instance Learning for Sparse Positive Bags. International Conference on Machine Learning, 2007.

[83] W.-S. Zheng, S. Gong, T. Xiang, Quantifying contextual information for object detection. IEEE International Conference on Computer Vision, 2009.

[84] M. Albanese, R. Chellappa, N. Cuntoor, V. Moscato, A. Picariello, V.S. Subrahmanian, O. Udrea, PADS: A Probabilistic Activity Detection Framework for Video Data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010.

[85] T. Brox, J. Malik, Object segmentation by long term analysis of point trajectories. European Conference on Computer Vision, 2010.

[86] P. Ochs, T. Brox, Object segmentation in video: a hierarchical variational approach for turning point trajectories into dense regions. IEEE International Conference on Computer Vision, 2011.

[87] P. Ochs, T. Brox, Higher Order Motion Models and Spectral Clustering. IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[88] M. Grundmann, V. Kwatra, M. Han, I. Essa, Efficient Hierarchical Graph Based Video Segmentation. IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[89] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient Graph-Based Image Segmentation. International Journal of Computer Vision, September 2004.

[90] K. Fragkiadaki, J. Shi, Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[91] K. Fragkiadaki, G. Zhang, Jianbo Shi, Video segmentation by tracing discontinuities in a trajectory embedding. IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[92] I. Endres, D. Hoiem, Category Independent Object Proposals. European Conference on Computer Vision, 2010.

[93] Y.J. Lee, J. Kim, K. Grauman, Key-segments for video object segmentation. IEEE International Conference on Computer Vision, 2011.

[94] T. Ma, L. Latecki, Maximum weight cliques with mutex constraints for video object segmentation. IEEE Conference on Computer Vision and Pattern Recognition, 2012.

[95] D. Zhang, O. Javed, M. Shah, Video Object Segmentation through Spatially Accurate and Temporally Dense Extraction of Primary Object Regions. IEEE Conference on Computer Vision and Pattern Recognition, 2013.

[96] W. Brendel, S. Todorovic, Video object segmentation by tracking regions. IEEE International Conference on Computer Vision, 2009.

[97] J. Wang, P. Bhat, R.A. Colburn, M. Agrawala, M.F. Cohen, Interactive video cutout. ACM Transactions on Graphics, 2005.

[98] X. Bai, J. Wang, D. Simons, G. Sapiro, Video Snapcut: Robust Video Object Cutout Using Localized Classifiers. ACM Transactions on Graphics, 2009.

[99] B.L. Price, B.S. Morse, S. Cohen, LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. IEEE International Conference on Computer Vision, 2009.

[100] X. Bai, J. Wang, G. Sapiro, Dynamic color flow: a motion-adaptive color model for object segmentation in video. European Conference on Computer Vision, 2010.

[101] F. Galasso, R. Cipolla, B. Schiele, Video Segmentation with Superpixels. Asian Conference on Computer Vision, 2012.

[102] A. Levinshtein, C. Sminchisescu, S. Dickinson, Spatiotemporal Closure. Asian Conference on Computer Vision, 2010.

[103] A. Vazquez-Reina, S. Avidan, H. Pfister, E. Miller, Multiple hypothesis video segmentation from superpixel flows. European Conference on Computer Vision, 2010.

[104] T. Wang, J. Collomosse, Probabilistic Motion Diffusion of Labeling Priors for Coherent Video Segmentation. IEEE Transactions on Multimedia, 2012.

[105] V. Badrinarayanan, F. Galasso, R. Cipolla, Label Propagation in Video Sequences. IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[106] S. Vijayanarasimhan, K. Grauman, Active Frame Selection for Label Propagation in Videos. European Conference on Computer Vision, 2012.

[107] J. Yuen, B. Russell, C. Liu, A. Torralba, LabelMe Video: Building a Video Database with Human Annotations. IEEE Conference on Computer Vision and Pattern Recognition, 2009.

[108] A.Y. Ng, M.I. Jordan, Y. Weiss, On Spectral Clustering: Analysis and an algorithm. Neural Information Processing Systems, 2001.

[109] X. Wang, I. Davidson, Flexible constrained spectral clustering. International conference on knowledge discovery and data mining, 2010.

[110] A. Sharma, E. von Lavante, R. Horaud, Learning Shape Segmentation Using Constrained Spectral Clustering and Probabilistic Label Transfer. European Conference on Computer Vision, 2010.

[111] Z. Lu, M. Carreira-Perpinan, Constrained spectral clustering through affinity propagation. IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[112] S.S. Rangapuram, Matthias Hein, Constrained 1-Spectral Clustering. Journal of Machine Learning Research, 2012.

[113] L. Xu, W. Li, D. Schuurmans, Fast normalized cut with linear constraints. IEEE Conference on Computer Vision and Pattern Recognition, 2009.