DYNAMIC VIEW SELECTION FOR HUMAN MOTION ANALYSIS IN CAMERA NETWORKS

by

Scott Spurlock

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2015

Approved by:

____

Dr. Richard Souvenir

____

Dr. Min Shin

____

Dr. Shaoting Zhang

____

Dr. Jing Xiao

____

Professor Eric Sauda

ABSTRACT

SCOTT SPURLOCK. Dynamic view selection for human motion analysis in camera networks. (Under the direction of DR. RICHARD SOUVENIR)

Automated human activity analysis for multi-camera networks requires algorithms that are both accurate and efficient for practical, real-time use. Current approaches face a trade-off between accuracy and speed, with the most accurate methods having high computational cost. This work is motivated by the observation that multi-camera networks provide redundant data across views. In many cases, algorithms could perform accurate analysis with a subset of the available information. We propose algorithms that dynamically select a subset of the network cameras for each stage of the automated analysis pipeline. The goal of this research is to develop algorithms for multi-camera networks with high computational efficiency without sacrificing accuracy. In particular, we focus on solving core computer vision problems related to the human motion analysis pipeline: detection, tracking, pose estimation, and action recognition. Experiments on benchmark datasets demonstrate the applicability of dynamic view selection to each of these areas.

ACKNOWLEDGMENTS

I would like to thank my research advisor, Richard Souvenir, for mentorship above and beyond the call of duty. I also appreciate the help and advice of my dissertation committee members, Min Shin, Shaoting Zhang, Jing Xiao, and Eric Sauda. Thanks also to the members of the VIA Lab for all the brainstorming, celebration, and commiseration; if we don't end up solving computer vision, it won't be for lack of trying. I am grateful for the financial assistance of the UNC Charlotte Graduate Assistant Support Plan, as well as the NSF Graduate Assistance in Areas of National Need Program. Finally, my deepest gratitude to my family for their unfailing support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Distributed camera networks (DCN) are becoming less expensive and more commonly available, enabling important applications across many domains. Surveillance of areas such as airports or other public locations can be enhanced with automation augmenting manual observation. In the sports domain, automated summarization and highlight generation of, for example, basketball or soccer matches could find immediate commercial application. For next-generation smart homes, in-home cameras can take advantage of automated algorithms to detect dangerous events such as falls, which cause more injury-related deaths than any other factor for people over the age of 78 [58].

Another application area for DCN is architectural analysis. Traditionally, an architect may conduct a Post Occupancy Evaluation (POE) of a space after construction to understand how people use the space and whether it is responsive to its users' needs. Data for POEs are commonly gathered by ethnographers through in-place observation. This approach limits the scale of the possible analysis to short time-horizons, typically measured in hours. Although long-term video (weeks, months, or even years) may be available, time and cost constraints make manual analysis of such large-scale data impractical.

For large-scale analysis of how people use spaces, as well as applications in surveillance, sports analysis, and smart homes, automated analysis of human behavior in

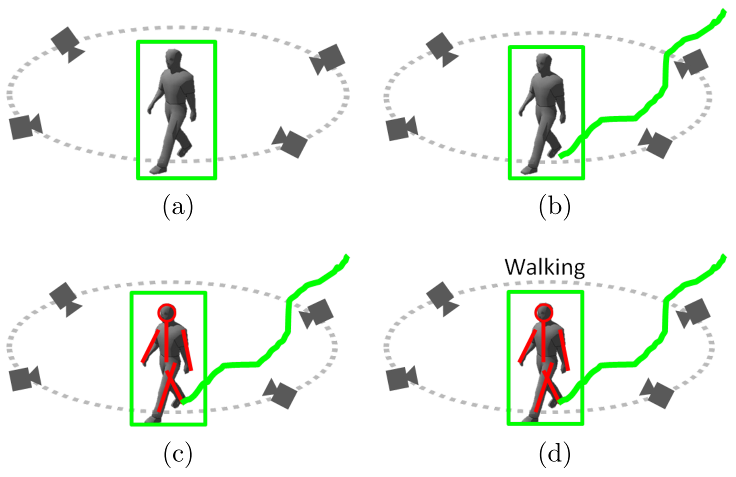


Figure 1: The human motion analysis pipeline in a distributed camera network consists of (a) detection, (b) tracking, (c) pose estimation, and (d) action recognition.

DCNs is needed. There are several key computer vision problems that make up the automated *human motion analysis pipeline* (Figure 1). For a network of cameras, it is necessary to detect the presence of a person, follow the person over time, reason about a person's body configuration, and classify an observed sequence of motions into an action category, distinguishing between such potentially similar actions as falling or sitting down. In the computer vision literature, these tasks are known as detection, tracking, pose estimation, and action recognition. Recently published approaches have shown promising results in each of these areas.

Multi-camera algorithms, which aggregate information from multiple sensors, tend to have higher accuracy than their single-camera counterparts. The availability of more than one perspective ameliorates issues such as occlusions due to multiple people in the scene and pose ambiguity due to viewpoint. However, the improved accuracy comes at the cost of increased resource consumption. In general, processing time increases at least linearly with the number of cameras due to feature computation, plus the additional overhead required for integration of information from multiple sensors. For distributed computing, network capacity may also play a role due to the heavy loads required for transmitting multiple video streams from cameras to servers.

This dissertation investigates when and how selecting a subset of the available cameras in a distributed camera network can reduce latency with minimal increase in error for human activity analysis. We propose dynamic selection of a variable number of the available cameras at each time step, which reduces computational burden while keeping error rates low, by using only the most discriminative cameras. For each area of the human motion analysis pipeline, we describe new algorithms that learn when and how multi-camera visual data should be sampled and processed to optimize both speed and accuracy.

In the next chapter, we review prior work prior work related to balancing computation and accuracy in computer vision. The following chapters will describe the details of our specific methods for detection, tracking, pose estimation, and action recognition. We conclude in Chapter 7.

CHAPTER 2: RELATED WORK

Typically, existing multi-camera algorithms face a compromise between accuracy and speed, with the most accurate methods being computationally expensive. However, for applications targeted at human motion analysis, it is often important to be able to accomplish these tasks both quickly and with high accuracy. One common approach to reducing computation time of algorithms is subsampling data, i.e., using a subset of the available information. In computer vision, previous work has explored subsampling in the spatial and temporal domains; however, no prior work has considered the idea of explicitly addressing the accuracy-speed trade-off in terms of the number of cameras evaluated at one time. In object detection, cascade frameworks (e.g., [77, 83]) are commonly used to limit the number of classifiers that must be evaluated for each candidate location. For action recognition, some work has investigated balancing accuracy and number of frames observed (e.g., [18, 27]) for the single-camera case. One multi-view tracking method dynamically varies the computational resources dedicated to each camera based on sensor reliability [34].

Other work has undertaken view selection in the context of selecting a single best view. Rudoy et al. [70] present a method to identify the best viewpoint for an action from a human observer's standpoint based on motion features. Other approaches select the best view based on the number of spatio-temporal features detected [92], estimated camera distances and person orientation [72], or silhouette properties [54].

However, unlike our approach, which seeks to reduce computation by reducing the number of views required by the algorithm, these methods still require features to be computed for every camera at each time step.

The idea of using information from one view to inform view selection can also be found in the area of active vision [26], usually in the context of a mobile agent. For example, Arbel and Ferrie perform object recognition using entropy maps, which model the predicted suitability of potential viewpoints to help an agent determine the object with minimal observations [2]. In robotics, this is commonly called the next best view (NBV) problem [25]. The active vision paradigm is not generally applicable to human motion analysis in a DCN because the space of potential views is continuous rather than limited to a discrete set of available cameras. Also, unlike in a DCN, the environment or objects of interest are usually static in active vision, so the time to compute or decide on the next view is not a factor.

In the following chapters, we will describe how dynamic view selection can be applied to detection, tracking, pose estimation and action recognition.

CHAPTER 3: DETECTION

3.1 Introduction

Detection of multiple people in a scene has many important applications, including automated surveillance, crowd modeling, and sports analysis. For applications involving video input, these person detections often serve as input to a subsequent tracking stage, to link detections across time. In multi-camera, multi-object (MCMO) detection, information from multiple cameras is aggregated to identify each person in the scene. A particular challenge for multiple-person detection is occlusion. As the number of people in the scene increases, it becomes more likely that a camera's view of a person may be blocked by other people. Figure 2 shows an example scene where two people have been detected, indicated by green and yellow bounding boxes. From the first camera's perspective, both people are clearly visible, while from the second camera, one person is occluded by the other. Compared with single-camera approaches (e.g., [61, 100]), by making use of multiple cameras with overlapping fields

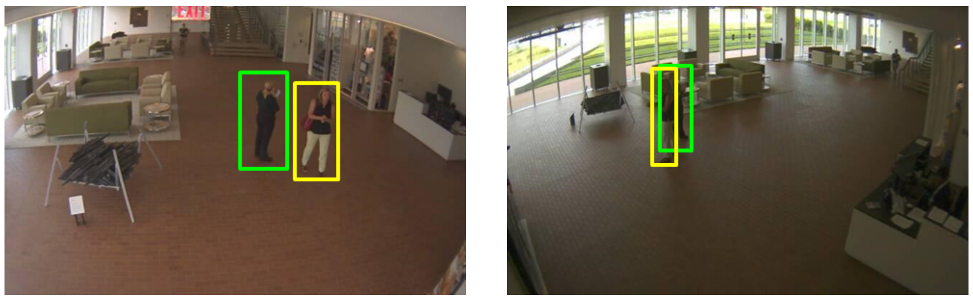


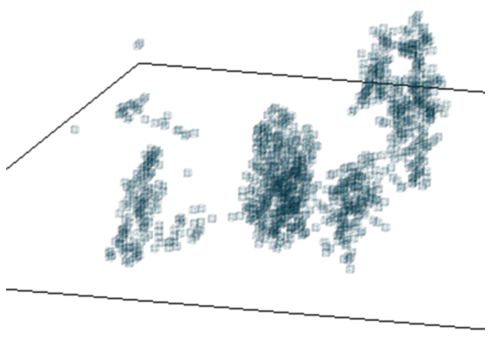
Figure 2: In distributed camera networks, occlusions may make person detection more difficult from some viewpoints.

of view, several recent methods [1, 68, 32, 65] have shown robustness to occlusion in crowded scenes.

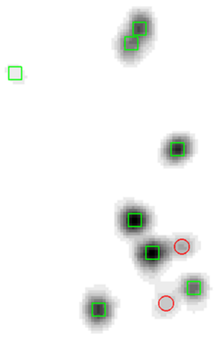
Of recently developed MCMO approaches, the most accurate involve significant computational expense. The fastest methods tend to be less accurate; in some cases (e.g., [32]), detection methods may provide only a probability map of possible person locations in the scene, delaying final localization to a subsequent tracking phase.

In parallel, recent methods for pedestrian detection have shown promising results for identifying individual people in (single-view) images. At low resolutions and in the presence of occlusion, however, even the best detectors perform poorly. Further, while detector speed has improved significantly in recent years, these methods are not designed to be used for multi-camera person detection in real-time at typical resolutions using the common approach of sliding windows at multiple scales and locations. Even the fastest pedestrian detectors operate at speeds of ≈ 2.7 fps for pedestrians of size 50 pixels in images of size 640×480 pixels for a single image [23]. For multi-camera networks, the computation required increases with the number of cameras. Speeds for these high-level detectors can be significantly improved, however, by limiting the search space in the spatial and scale domains, as well as in terms of which cameras are used for detection.

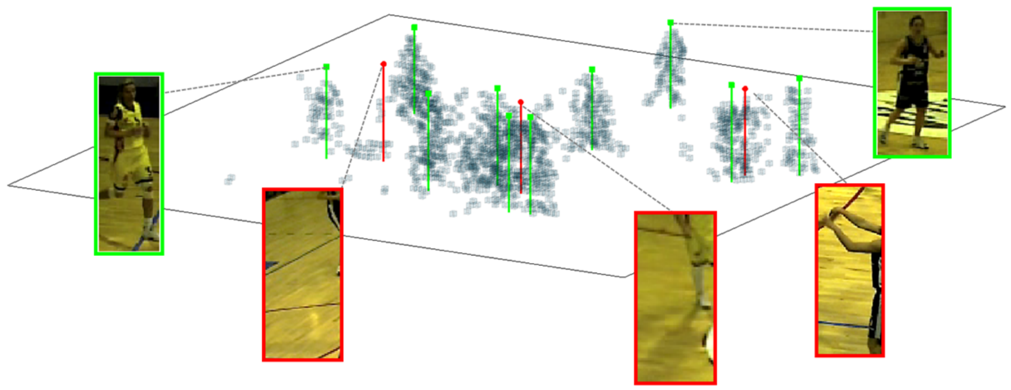
In this chapter, we propose a hybrid approach that uses fast low-level detection and targeted high-level verification, achieving high accuracy at real-time speed. Our framework is modular, consisting of low, medium, and high-level detection steps. The modularity of the design allows our framework to incorporate new or pre-existing detector implementations as needed. With each successive step more computationally



(a) Low-level



(b) Mid-level



(c) High-level

Figure 3: MCMO detectors based on (a) low- and (b) mid-level features often identify a small number of candidate locations, but are prone to "ghosts" (false positives) caused by shadows, occlusions, and projective effects among the true positive detections. Our method (c) incorporates high-level image-based features from the camera(s) with the best view to identify ghosts (red lines) and verify actual people (green lines).

expensive than the previous, the goal is to discard as many hypotheses as possible using computationally inexpensive methods, and only use high-level detectors to *verify* uncertain earlier hypotheses. Figure 3 illustrates the idea. A low-level occupancy detector identifies 3D foreground voxels, shown as gray cuboids. A mid-level aggregation step localizes objects, finding both true detections (green lines) as well as false positives (red lines), known as ghosts. For high-level pedestrian verification, image patches are extracted corresponding to locations to be verified. The goal is for a pedestrian detector to accurately evaluate the presence of a person in the image patch. However, in a multi-camera environment, certain viewpoints may be preferable to others, in terms of the expected accuracy of the detector.

Our main contribution is a multi-stage, coarse-to-fine framework for MCMO detection, which includes a probabilistic model for selecting a subset of cameras based on expected detection accuracy. The targeted use of high-level verification keeps computational cost low. We evaluate our method on a challenging benchmark dataset for MCMO detection and tracking. Our results show the efficacy of our real-time approach, outperforming recent methods in both detection accuracy and computational efficiency.

3.2 Related Work

Detecting people in images and video has been well-covered over many years [98]. Our focus is on multi-camera methods that incorporate low-level features for occupancy estimation.

3.2.1 Multi-Camera, Multi-Object Detection

Most MCMO methods start with background subtraction (e.g., [76]) and then fuse extracted foreground silhouettes to a common 3D coordinate system or ground plane. For example, Khan and Shah [48] use homographies to warp foreground probability maps to a common reference plane and detect feet locations, while Eshel and Moses [28] detect head tops by incorporating intensity correlation in a similar homography-based framework. Fleuret et al. [32] introduce a probabilistic framework to model occupancy over a ground plane grid. Several methods [68, 52, 9] employ a

3D reconstruction approach, where occupancy is calculated over a discrete 3D grid of voxels, instead of just the 2D ground plane. These methods may detect people in the 3D space [9] or project the volumetric reconstruction to the ground plane [68, 52]. Typically, exact localization is delayed to a later tracking phase based on, e.g., graph cuts [48] or dynamic programming [32] over temporal windows.

3.2.2 Reducing False Positive Detections

Some recent MCMO detection methods have explicitly incorporated schemes to address ghosts. Alahi et al. [1] model ground plane occupancy estimation as a sparse optimization problem. A sparsity constraint is intended to rule out false positives during the detection phase. While this method achieves high detection accuracy, the authors' implementation takes 10 seconds per frame, making it unsuitable for real-time applications. Peng et al. [65] incorporate a graphical model that explicitly encodes occlusion relationships among discretized ground-plane locations. An iterative algorithm finds the occupancy configuration that best explains the camera foreground images. The method reduces the occurrence of ghost detections due to the occlusion reasoning, but takes 3 seconds per frame in the authors' implementation. Other methods incorporate simple rules to reduce ghosts, such as fixing a priori the number of objects to be detected [19].

Our framework, which includes concepts common to MCMO methods, incorporates pedestrian verification directly into the detection stage rather than a subsequent tracking step or with ad hoc rules. The verification step relies on selecting the best viewpoints for image-based pedestrian detection. However, compared to the

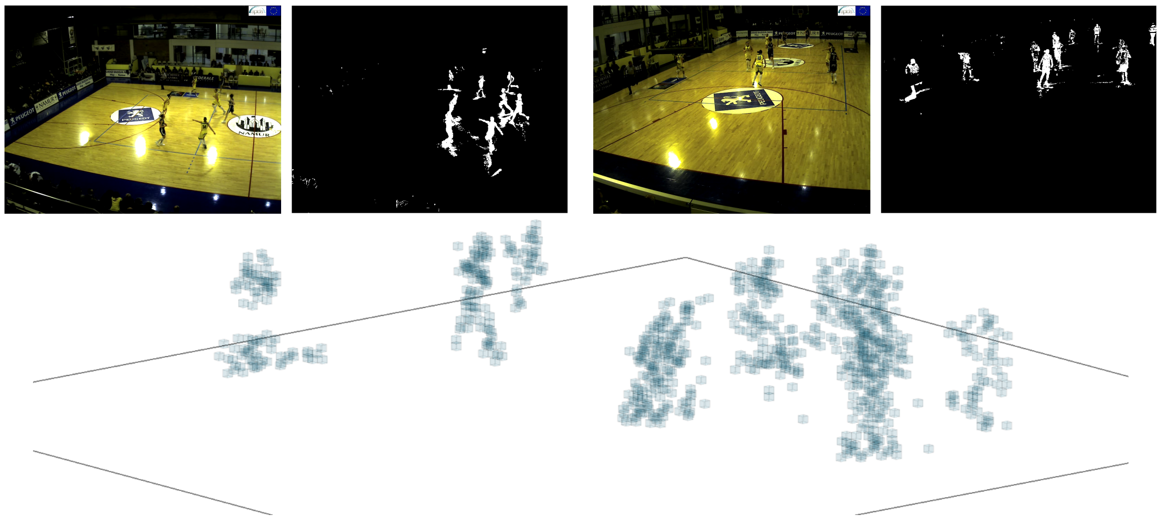


Figure 4: Example input frames and extracted foreground silhouettes used to perform a coarse 3D reconstruction for our low-level detector.

sliding window approach commonly employed for single image pedestrian detectors, our method drastically reduces the search space by only evaluating selected image patches. Viewed in this light, the low-level detection step provides geometric context similar to approaches (e.g., [37]) that use scene context to reduce false positives. By combining efficient low-level detection, mid-level aggregation, and targeted use of high-level verification, our framework is capable of real-time multi-person detection in multi-camera networks.

3.3 Base Detector

Our pedestrian verification approach could be used with any low- or mid-level MCMO detector. In this section, we describe our base detector implementation.

3.3.1 Low-level Detection

As shown in Figure 4, our low-level detector performs change detection on N_C cameras viewing the scene in order to create a coarse 3D reconstruction of the visual

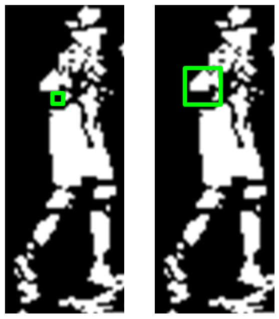


Figure 5: (a) Point sampling takes the single pixel in the image corresponding to the projected voxel center, while (b) area sampling considers the 2D bounding box of the projected voxel's 3D extent.

hulls of moving objects. The scene volume is discretized into a voxel grid, $\mathcal{V} = \{\nu_1, \nu_2, \ldots, \nu_n\}$, where each voxel is identified as either background or foreground by a straightforward voting scheme:

$$\nu_i = \begin{cases} 1 & \text{if } \sum_c g(\nu_i, c) \geq \tau_c \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $g(\nu_i, c)$ indicates whether voxel ν_i projects to foreground in camera c, and τ_c is the threshold for the number of cameras in the network that must agree for a positive voxel detection. To implement the voxel-image occupancy function, g, other MCMO detectors (e.g., [68]) employ *point sampling*, where the voxel center is projected to a single pixel in an image. For greater robustness to noisy foreground extraction, we employ *area sampling*, where the 3D extent of the voxel is projected to a bounding box in an image. Figure 5 shows a case where, with point sampling, a voxel incorrectly projects to background due to noise while, with area sampling, most of the projected bounding box consists of foreground pixels.

We define the voxel-image occupancy function as:

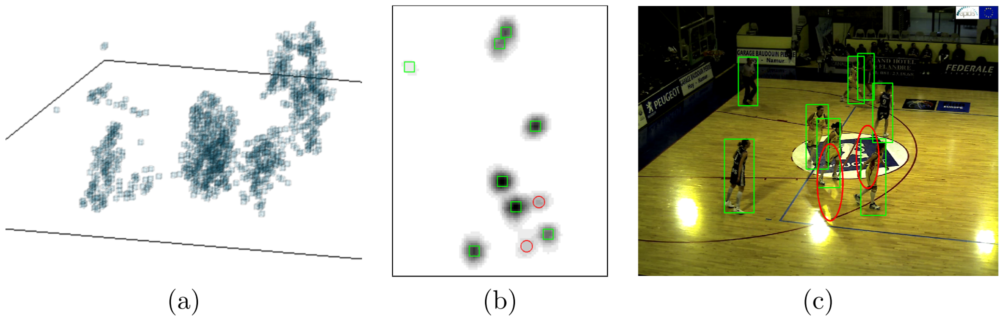


Figure 6: Given foreground voxels (a), mean shift clustering (b) localizes objects. For the identified cluster centers, green squares are true positives and red circles are false positives (ghosts). Note that the two ghosts are more pronounced than the correct detection of the person at the top-left. (c) An image from a camera in the network shows the projected detections.

$$g(\nu_i, c) = \begin{cases} 1 & \text{if } \lambda(\nu_i, c) \geq \tau_f \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where λ represents the proportion of pixels in the associated bounding box in image c corresponding to foreground and τ_f is a system-specific threshold that can be tuned based on the noise level of the foreground segmentation process. The voxel-image occupancy function with area sampling can be implemented efficiently using the integral image technique [82] with the foreground mask image.

3.3.2 Mid-level Aggregation

The next stage is aggregation of voxel detections to objects, illustrated in Figure 6. Our method relies on mean shift clustering (MSC) [14] for this step. MSC is a non-parametric clustering approach that can find non-uniform or narrow modes in a distribution, which, in our case, correspond to potential object locations in the scene. MSC is well-suited to the problem because no prior knowledge about the number or location of objects is needed.

Let $\{\mathbf{z}_i\}$ be the set of points in $\mathbb{R}^3$ corresponding to the centers of the identified foreground voxels. We define the kernel density estimator [14] for occupancy at a point $\mathbf{z}$ as

$$\hat{f}(\mathbf{z}) \propto \sum_i K_{\mathbf{H}}\left(\mathbf{z} - \mathbf{z}_i\right) \tag{3}$$

where $K_{\mathbf{H}}(\mathbf{z}) = |\mathbf{H}|^{-1/2} K\left(\mathbf{H}^{-1/2}\mathbf{z}\right)$. Here $\mathbf{H}$ is a d $\times$ d bandwidth matrix and K is the unit flat kernel [12]

$$K(z) = \begin{cases} 1 & \text{if } \|z\|_\infty \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\|\cdot\|_\infty$ is the infinity norm, which implies an axis-aligned, box-shaped kernel with dimensions controlled by bandwidth matrix, $\mathbf{H}$, a diagonal matrix, where each element along the diagonal is the squared bandwidth for a dimension of the box. For person detection, we choose $\mathbf{H}$ to approximate the dimensionality of an upright person, i.e., $h_1 = h_2 = h_3/4$. While MSC implementations typically incorporate the smoothly differentiable Epanchnikov or Gaussian kernels, our choice of an axis-aligned box kernel allows for faster computation and works well in practice.

Each cluster is scored based on the proportion of foreground voxels within the bandwidth to total bandwidth volume

$$\omega_p(\delta_m) = \frac{\hat{f}(\delta_m)}{\prod_{j=1}^{d} h_j} \tag{5}$$

where δ_m is the d-dimensional cluster mean (for our application, $d = 3$). The cluster score can be thresholded to discard low-scoring detections, which often correspond to ghosts. However, care must be taken to avoid rejecting valid detections. Figure 6 shows an example where a valid detection scores lower than two ghost detections. In the next section, we describe how pedestrian detection can help distinguish between correct and incorrect detections.

3.4 Pedestrian Verification

In some systems [1, 32, 65], the output from low- and/or mid-level stages are directly used as output detections. However, some of these may actually be "ghosts," or false positive detections due to shadows, reflections, or occlusions. These errors become increasingly common as crowd density increases, and, in complex scenes, significantly degrade overall system accuracy. Figure 6 shows two examples of ghost detections in red. Our high-level detection stage, pedestrian verification, is aimed at identifying and eliminating these false detections without filtering out correct detections.

3.4.1 Predicting Verification Accuracy

For a given cluster, represented by center location, δ_m, the 3D bounded region corresponds to an image patch in each camera. For each candidate patch, we compute the *Expected Detection Accuracy (EDA)*, $E[Q|\Theta]$, where Q is a continuous random variable representing accuracy of a pedestrian detector under the conditions encoded by the vector, Θ. Ideally, the model attributes would be features that are efficient to compute following the low-level detection phase. A recent survey [23] provides an evaluation of the performance of numerous detectors as a function of occlusion and scale. The best performing detectors work well for near-scale (at least 80 pixels high)

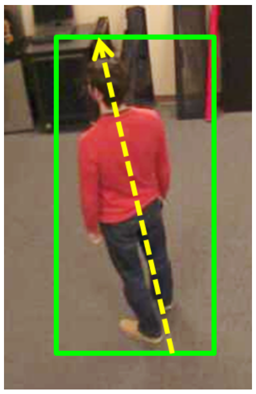


Figure 7: The projected bounding box for a candidate detection is shown in green. The up vector, U^c_m, is shown in yellow.

examples, with rapid performance decrease as pedestrian size decreases. Additionally, all of the detectors were sensitive to occlusion; even partial occlusion (<35%) led to a log-average miss rate of 73% for the best detector. To estimate the predictive power of a pedestrian detector from a given viewpoint, our model incorporates occlusion, scale, and also verticality, a measure of how upright a person appears from a particular viewpoint. For a candidate location and corresponding image patches, these three features can be computed using the projection of the 3D bounding boxes.

3.4.2 Model Attributes

We define the bounding box for detection m projected into camera c as the (rectangular) area of pixels, R^c_m. For the candidate detection, the up vector, U^c_m, is the projection of a 3D vector pointing up along the positive Z-axis from the candidate ground location, m, to the target's estimated height, as viewed in camera c (Figure 7).

3.4.3 Occlusion

In order to estimate an occlusion ratio for each detection based on the other (potential) detections in the scene, we adapt the painter's algorithm [36] from computer

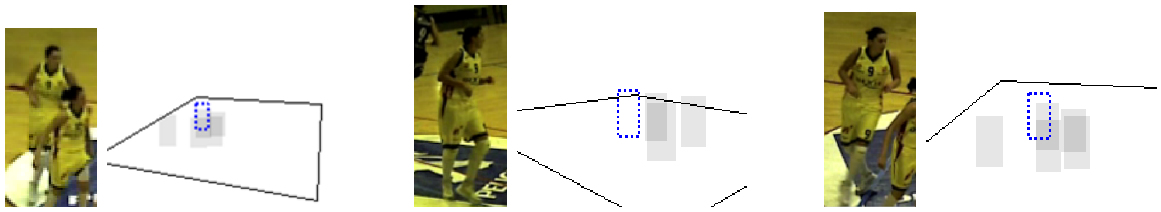


Figure 8: The occlusion value is estimated by calculating the overlap of the candidate bounding box (dashed blue rectangle) with other, closer detections (gray boxes). For the views shown, occlusion is 0.82, 0.00, and 0.56, respectively. (Lower is better.)

graphics. The idea is to order the detections by proximity to the camera center, and project a synthetic bounding box, R_j^c, into a 2D accumulator for each detection that is closer to the camera than R_m^c. The occlusion ratio measures the overlap of other (potential) detections with the candidate location:

$$\omega_o(m,c) = \frac{\left|\left(\bigcup_j R_j^c\right) \cap R_m^c\right|}{|R_m^c|} \tag{6}$$

where $|\cdot|$ is the number of pixels in the box. Figure 8 shows an example of how occlusion is calculated for three different views of one example detection from the scene depicted in Figure 6.

3.4.4 Verticality

Typically, pedestrian detectors are trained on examples containing mostly upright (vertical) people. So, rather than incur the cost of training many detectors or applying a warp to each image patch, we estimate how upright a person at a given 3D location will appear from a particular view. Verticality is computed as:

$$\omega_u(m,c) = \left\langle \frac{U_m^c}{\|U_m^c\|}, \frac{F_m^c}{\|F_m^c\|} \right\rangle \tag{7}$$

where F_m^c is a vector pointing in the up direction (along the positive Y axis) in the image and $\langle \cdot, \cdot \rangle$ indicates the inner product.

3.4.5 Height

One of the features most correlated with pedestrian detection accuracy is the pixel height of the pedestrian [23]. The height, in pixels, of a projected object is simply the magnitude of the projected up vector, $\omega_h(m, c) = \|U_m^c\|$.

3.4.6 Model

Given a set of training examples, we compute, for each attribute, the expected accuracy (true positive, true negative) using a binary logistic regression model. That is, we compute $E[Q \mid \omega_x]$ for each attribute. To model the joint expectation for a given image patch, we make the Naive Bayes assumption of conditional independence between the features. This gives:

$$E[Q \mid \Theta] \propto E[Q \mid \omega_o] \cdot E[Q \mid \omega_u] \cdot E[Q \mid \omega_h] \tag{8}$$

Figure 9 shows some examples of the expected detection accuracy evaluated for selected patches. In the next section, we show how this value can be used to compare multiple image patches of the same object detection to select the best view(s) for pedestrian verification in a real-time MCMO detection framework.

3.5 Results

We evaluated our method on the APIDIS dataset[1], which contains footage from a basketball game captured by 7 calibrated, pseudo-synchronized cameras (Figure 10).

[1] http://www.apidis.org/

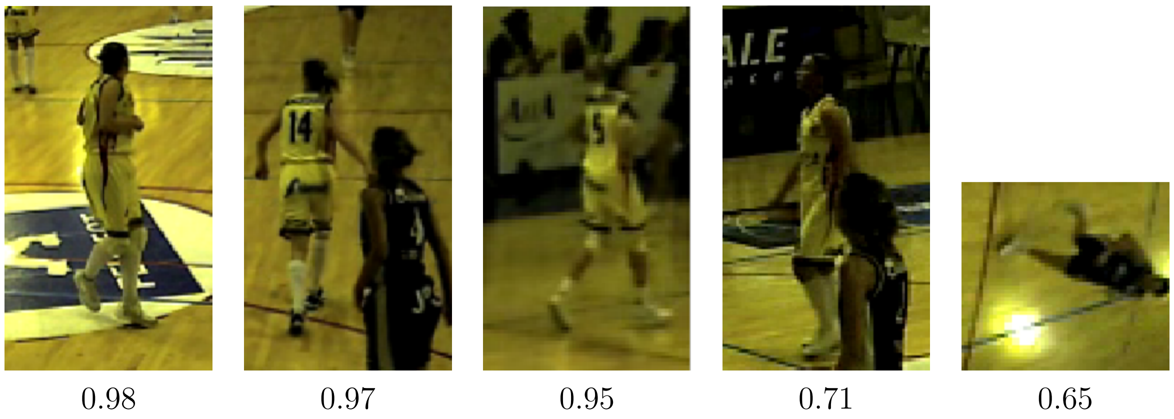


Figure 9: The estimated detection accuracy (Equation 8) for selected image patches. The first patch depicts an ideal case (unoccluded, upright, and near-field). The remaining patches show examples of slight occlusion, smaller height, moderate occlusion, and non-verticality, respectively.

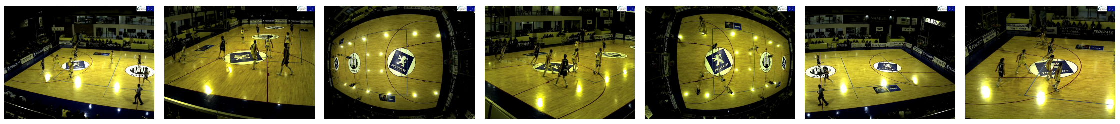


Figure 10: Frames from the 7 cameras in the APIDIS dataset.

The dataset contains people of similar appearance and heavy occlusions, as well as shadows and reflections on the court. In order to compare results with other recent work [1, 68, 32, 65], we followed the most common protocol of measuring performance within the bounds of the left side of the basketball court, which is covered by the most cameras. For quantitative evaluation, we used precision and recall, where a true positive is a detection whose estimated location projected onto the ground plane is within a person-width (50 cm) of the ground truth, a false positive is a detection unmatched to an actual person, and a false negative is a missed detection. Figure 11 shows an example scene with two people.

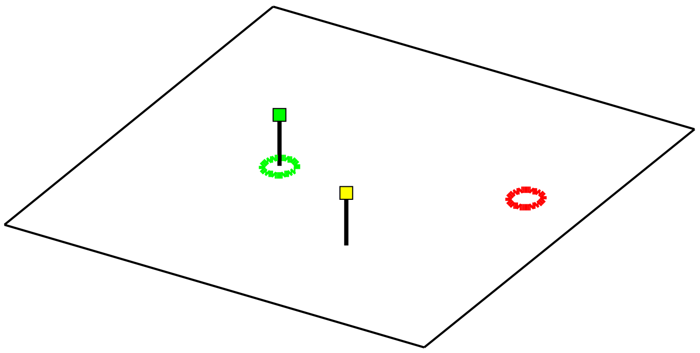


Figure 11: In this synthetic scene, two people are indicated with lines. A true positive detection (green circle) must be within 50 cm of the ground truth ground-plane location, indicated by the circle radius. A false positive detection is indicated by a red circle. A missed detection (false negative) is indicated by yellow.

3.5.1 Implementation Details

We set the minimum number of cameras for voxel occupancy voting, τ_c, to 3, and the foreground ratio threshold, τ_f, to 0.25. For mean shift clustering, the bandwidth was $45 \times 45 \times 180$ cm. In the 3D occupancy grid, each voxel covered 10 cm^3. For change detection, our method uses a GPU implementation of adaptive background subtraction [104].

3.5.2 Pedestrian Detector Evaluation

Our method supports most image-based pedestrian detectors. We evaluated four commonly-used, pre-trained detectors: HOG [17], VJ, based on the Viola-Jones cascade classifier [82], and the Dollár et al. [21] detector, trained with the INRIA dataset [17] (DOLLAR-INRIA) and the CalTech dataset [22] (DOLLAR-CALTECH). The Viola-Jones detector is a cascade classifier trained specifically on upper body examples [50], while the others are trained to identify full-body pedestrians. Each of these pedestrian detectors provides a detection score, and a threshold is commonly applied to obtain the final result. For a set of image patches containing both positive (people) and negative (background) examples, we computed the ROC curve across a range of thresholds for each detector and used the Area Under the Curve (AUC) measure as a basis for comparison (Figure 12. HOG, VJ, DOLLAR-INRIA, and DOLLAR-CALTECH achieved 0.65, 0.56, 0.60, and 0.67, respectively. Overall, DOLLAR-CALTECH performed the best, and, unless otherwise specified, is the implementation we employed for subsequent experiments. These values are much higher than would be expected from the typical approach of pedestrian detection of sliding windows across multiple image scales and locations. Beyond the efficiency concerns, this approach leads to many false positives and false negatives. However, with a fixed location and scale (i.e., an image patch corresponding to a particular 3D location), such detectors can be quite accurate. This phenomenon was noted in a recent survey [23], which found that classifier performance on image patches is only weakly correlated with detection performance on full images.

3.5.3 Pedestrian Verification

To evaluate the effect of pedestrian verification on MCMO detection, we implemented the base detector, and performed experiments applying pedestrian verification from multiple cameras. We tested two schemes: (1) using the top-c cameras, and (2) selecting a variable number of cameras based on predicted accuracy. To combine the results from multiple cameras, the c detector scores are averaged, weighted by the expectation (Equation 8), prior to thresholding. In the variable-camera scheme, all cameras with an EDA above .9 are included in the ensemble.

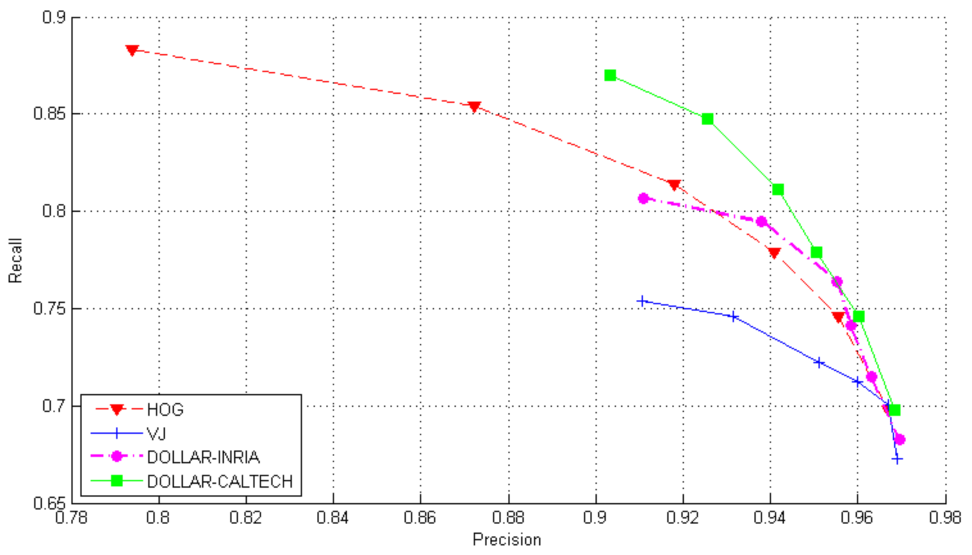


Figure 12: ROC curves for four pedestrian detectors over a range of thresholds. For both precision and recall, higher is better.

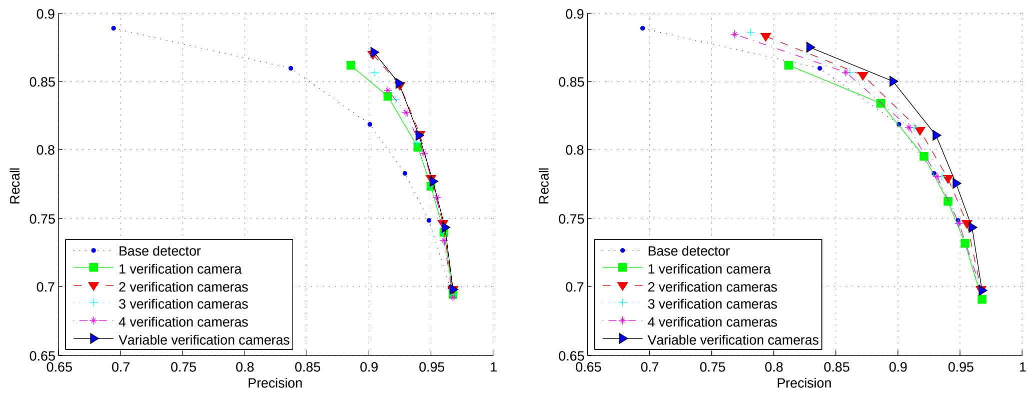


Figure 13: Precision-recall curves for base detection with pedestrian verification with DOLLAR-CALTECH (left) and HOG (right) using both fixed and variable number of camera schemes.

Figure 13 shows precision-recall curves for various verification schemes on the APIDIS dataset for two pedestrian detectors (HOG and DOLLAR-CALTECH). While increasing from $c = 0$ to $c = 2$ cameras improves the overall performance, adding a third or fourth camera does not. This result suggests that, for this particular dataset, there are many instances where two of the available cameras provide complementary suitable views of a particular location, but additional viewpoints are neither helpful and perhaps contradictory. Overall, using a variable number of cameras for each location performed best, although the effect is more pronounced with the HOG detector than with DOLLAR-CALTECH. On average, the variable scheme resulted in 2.56 image patches evaluated for each candidate location.

3.5.4 Comparison with Other MCMO Methods

Table 1 compares the results of our method with several recently published approaches on the APIDIS dataset. For each method, the precision, recall, F-score, and frames-per-second (FPS) are shown. Excluding our method, the speed-accuracy trade-off is evident across the related approaches. To the best of our knowledge, our method using $c = 2$ verification cameras (base detector + verification) outperforms all other reported detection results on the APIDIS dataset, while performing at real-time speeds. Note that our detection method outperfroms approaches that also incorporate tracking. Figure 14 shows some examples from this experiment.

Precision, recall, and framerate (FPS) numbers for the other methods are taken from results reported in the respective papers. Timing numbers, in particular, may not be directly comparable due to differences in hardware and other implementation

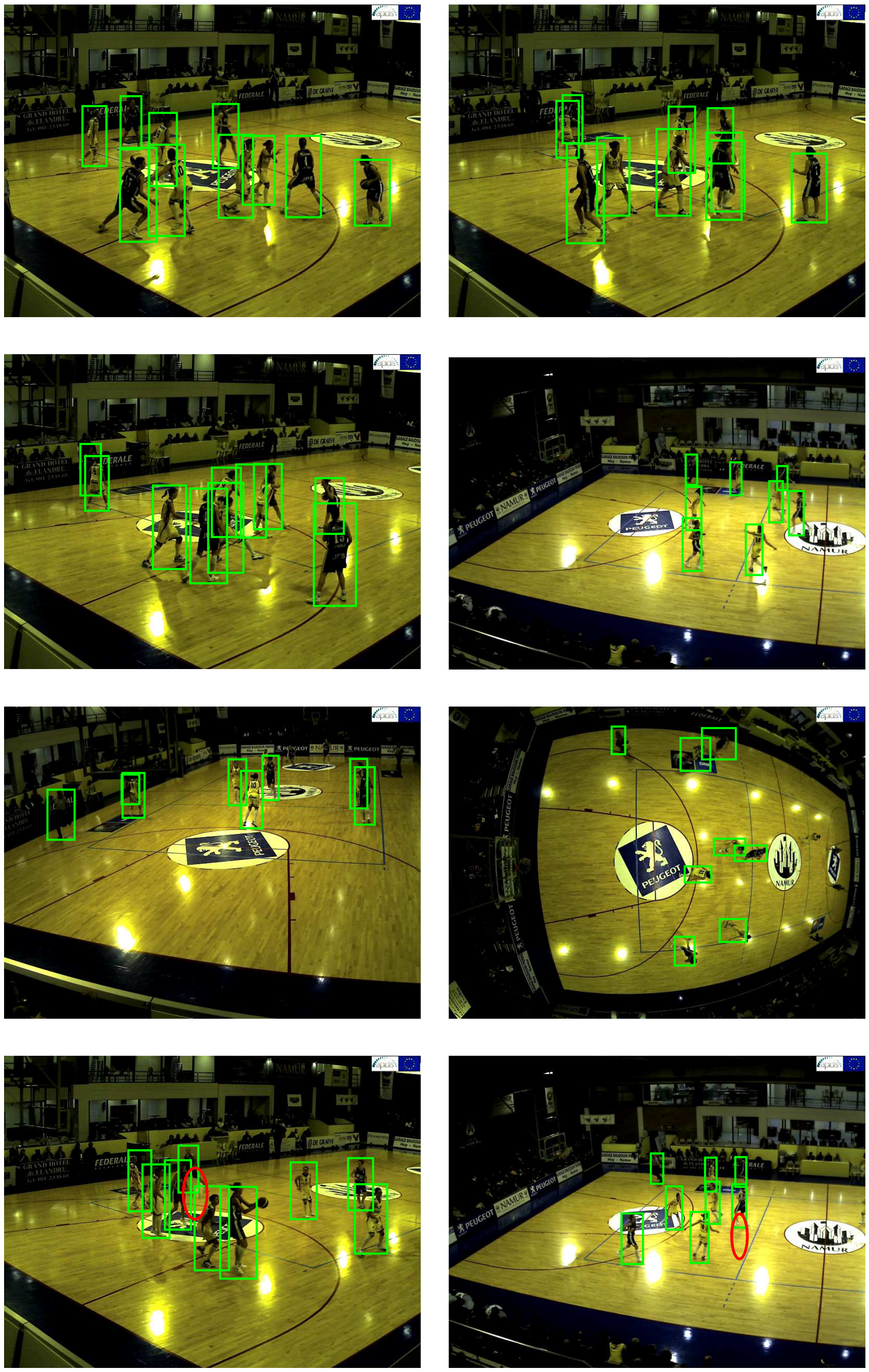


Figure 14: The top frames show examples of our method correctly identifying the presence of multiple people (green rectangles). The bottom two frames show challenging cases where the selected pedestrian detector failed for a given patch (red oval).

Method	Precision	Recall	F-Score	FPS
Base detector	0.84	**0.86**	0.85	13.13
Base detector + verification	**0.93**	0.85	**0.89**	10.40
Alahi [1]	0.92	0.82	0.87	0.1
Peng [65]	0.90	0.84	0.87	0.33
Posseger [68] (with tracking)	0.88	0.79	0.83	4.42
POM+KSP [32, 4] (with tracking)	0.80	0.73	0.76	0.03
POM [32]	0.51	0.63	0.56	**80.70**

Table 1: Comparison of our method and several recent approaches using precision and recall rate on the APIDIS dataset. POM+KSP results are taken from [68] and POM results are from [1].

details. Our method was implemented in C++ with OpenCV and deployed on a 2.5 GHz PC with 8 GB RAM and a Tesla C2075 GPU. For the base detector with verification, processing time is roughly 73%, 6%, and 21% for low-, medium-, and high-level detection, respectively.

3.6 Summary

We presented a framework for multi-camera, multi-object detection. Our multi-stage approach incorporates fast low-level detection and more accurate high-level pedestrian detection to verify uncertain hypotheses. The method is agnostic to any specific implementation of the base detector or verification method. This hybrid approach was shown to be effective in experiments on a challenging dataset, achieving state-of-the-art performance at real-time speeds. Note that while our focus in this chapter is on MCMO detection, the method can easily be incorporated into any end-to-end tracking system that incorporates linking detections over time. The tracking performance of classic approaches such as, e.g., Kalman filters [5, 76], as well as more recent data association-based tracking methods (e.g., [30, 40]), could directly benefit from our detection scheme. In the following chapter, we present an alternate approach to tracking, where detection is tightly integrated into the tracking process, and show how dynamic camera selection can be applied.

CHAPTER 4: TRACKING

4.1 Introduction

In this chapter, we apply dynamic camera selection to the problem of multi-camera object tracking. Recent multi-camera methods have helped to overcome some of the issues associated with object tracking, such as drift and occlusion, that arise in the single-camera case. However, the integration of multiple cameras introduces new challenges in terms of resource consumption (e.g., power, computing, and networking), and algorithm complexity. Moreover, while tracking accuracy tends to increase with the number of cameras, the potential also increases for poor measurements from individual cameras to negatively affect aggregate tracking estimates.

Our dynamic camera selection framework balances the computational efficiency of single-camera tracking with the power of a distributed camera network. The method allows for more efficient use of network and computing resources, and can scale to reduce error with the allocation of additional cameras depending on the desired trade-off between performance and accuracy.

4.2 Related Work

Recently developed multi-camera approaches have an improved ability to mitigate mistracking due to occlusion because of the increased likelihood of an available, un-occluded view of the target [98]. Multi-camera methods tend to be one of two types:

measurement correspondence approaches, where features are warped into a common view prior to state estimation, or *trajectory correspondence* approaches, where state estimates are computed independently in each view before being integrated [78].

Many measurement correspondence approaches make use of plane-to-plane transformations to warp detected objects from multiple camera views to an occupancy map in a reference plane, typically the ground plane [47]. Occlusions of the targets' feet and shadows mistakenly classified as foreground can create problems for methods that use the ground as a reference plane. More recent approaches [48, 19] address these issues by finding the ground plane occupancy using homographies to planes above and parallel to the ground plane. Most of these approaches focus on occupancy rather than using template-based object tracking. This can lead to difficulty in disambiguating nearby targets or discriminitive tracking in crowds. In [32], ground plane foreground occupancy masks are combined with color and motion models, which helps to address these issues.

Compared to measurement correspondence approaches, trajectory corresponce approaches tend to be less resource-intensive and more conducive to distributed processing. One of the early methods [7] used single-camera tracking until the system predicted the camera would no longer have a good view of the target and leveraged epipolar geometry to select a nearby camera to take over tracking. This method differs from ours in that our framework seeks to find the optimal subset of cameras to track the target *in every frame*, rather than simply switching to a different, single camera when necessary. More recent methods have attempted to incorporate the tracking information from all of the cameras in the network. In [62], particle filtering with color-based detection is used to track a target independently in each camera. Their system also uses epipolar geometry to reinitialize a poorly tracking camera from a higher-scoring neighbor. In [101], single-camera detection is used to estimate the 3D target location and these estimates are combined using the Extended Kalman Filter. These methods differ from our proposed framework in that they aggregate information from all available cameras rather than the best subset. In [46], utility functions are defined that model which pair of cameras provides the optimal view of each physical location in the environment. However, the focus is on coverage of physical locations rather than the best view of a particular target, which can lead to problems with multiple occluding targets.

Several hybrid methods have also been proposed that combine the measurement and trajectory correspondence approaches. For example, one method integrates particle filter results from each camera and the ground plane in both the detection and prediction stages by making use of “principal axes” to find reliable target intersection at the ground plane [24]. Han et al. assign a particle filter to each camera and vary the number of particles and relative contribution from each camera using a Gaussian mixture model weighted based on sensor reliability [34]. These hybrid approaches report excellent tracking results; however, like measurement corresponce methods, these come at the price of increased computational and algorithmic complexity.

Our hybrid approach, which is more similar to the trajectory correspondence methods, allows for dynamic camera switching at each timestep and tracking from an arbitrary number of cameras at once. Cases such as targets entering or leaving a camera view, target occlusion, and drift are handled automatically as the framework selects the best cameras to track actively at any given time based on a novel voting and ranking scheme.

4.3 Tracking Model

In our framework, each camera performs 2D object tracking independently. We chose to implement a particle filter-based tracker, but any number of other tracking methods could have been selected.

4.3.1 Object Model

Objects are represented as a weighted distribution, q, of color values of the pixels in image patch $J_{\mathbf{m},t} = \{\mathbf{m}' \in I \mid \mathbf{m}' \text{ is bounded by } \mathbf{m} \pm \langle w, h\rangle\}$, where h and w are the half-height and half-width of the bounding box centered at position $\mathbf{m}$, I is the image, and t is the frame number. The distribution, q, is over a quantized RGB color space. In our experiments, we use 16 bins per color channel, which results in a histogram of dimensionality $16 \times 16 \times 16 = 4096$. Each pixel location, $\mathbf{m}'$, is weighted based on:

- Kernel Function: Pixels closest to the image patch center, $\mathbf{m}$, are weighed more heavily than pixels at the edge using the Epanechnikov kernel [15]:

$$k(\mathbf{m}, \mathbf{m}') = \max\left[0, \frac{3}{4}\left(1 - \left|\left|\frac{\mathbf{m} - \mathbf{m}'}{\langle 2w, 2h\rangle}\right|\right|^2\right)\right] \tag{9}$$

- Foreground Likelihood: Pixels are additionally weighted on the likelihood of belonging to the foreground model. Let $B(\mathbf{m}')$ be the background model, represented as a pixel-wise Gaussian distribution with a mean equal to the mode at location $\mathbf{m}'$ in the video sequence, and standard deviation equal to the expected image noise. The foreground likelihood, ϕ, is:

$$\phi(\mathbf{m}') = 1 - P\left[B(\mathbf{m}') = I(\mathbf{m}')\right]$$

The resulting kernel is calculated as $\kappa(\mathbf{m}, \mathbf{m}') = k(\mathbf{m}, \mathbf{m}')\phi(\mathbf{m}')$. We define a function $u = b(\mathbf{m}')$ that maps each pixel from the target image patch into a discrete bin u based on the pixel's location in RGB color space. Taking these elements together, the probability of each feature $u = 1...n$ in the model for the target centered at location $\mathbf{m}$ within the image patch $J_{\mathbf{m},t}$ is given by:

$$q_u(\mathbf{m}) = C \sum_{\mathbf{m}' \in J_{\mathbf{m},t}} \kappa\left(\mathbf{m}, \mathbf{m}'\right) \delta[b(\mathbf{m}') - u] \tag{10}$$

where δ is the Kronecker delta function, and C is a normalization factor.

4.3.2 Reference Plane Transform

Each camera tracks objects using the coordinate system in a global reference plane. For a calibrated camera, projection matrix P converts a 3D point $\mathbf{Z} = (X, Y, Z, W)^T$ to a 2D image location $\mathbf{m} = (x, y, w)^T$ as $\mathbf{m} = P\mathbf{Z}$, where $\mathbf{m}$ and $\mathbf{Z}$ are expressed as homogeneous coordinates. Denoting the j^{th} column of P as $\mathbf{p}_j$, we can derive a homography, $\mathcal{H}_{ref} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \mathbf{p}_4 + Z_{ref}\mathbf{p}_3 \end{bmatrix}$, which maps points on the image plane to a plane Z_{ref} units above the ground plane along the vertical axis where Z_{ref} is expressed in world coordinates [16]. Ground-plane homographies ($Z_{ref} = 0$) are commonly used (e.g., [47, 32]); however, it has been shown that reference planes above the ground plane can improve tracking accuracy [19]. In our model, we choose Z_{ref} to be half the approximate height of a target. Figure 15(a) illustrates the reference

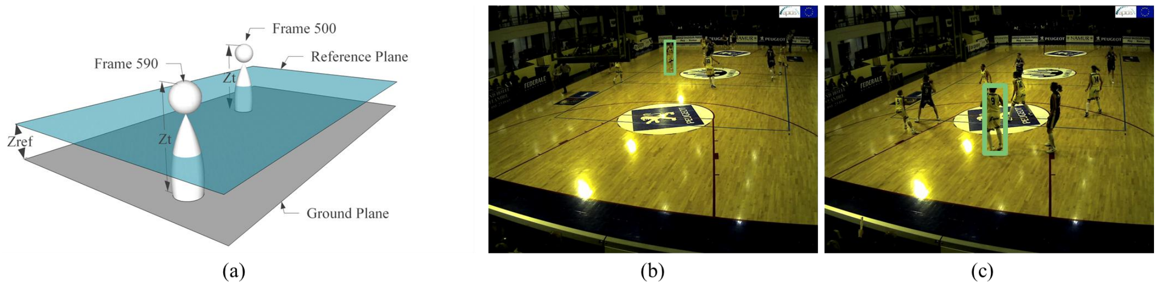


Figure 15: Targets are tracked in a reference plane parallel to and Z_{ref} units above the ground plane. To resize the bounding box dynamically for detection, the 3D points representing the top and bottom of the target are projected onto the image plane. The images in (b) and (c) show the automatically calculated bounding boxes for the same target at different times.

plane used for tracking. Given projection matrix P and the free parameter Z_{ref}, we define the transform, Γ, from reference coordinates, $\mathbf{z}$, to image coordinates $\mathbf{m}$:

$$\Gamma(\mathbf{z}, P) = (\mathcal{H}_{ref})^{-1}\mathbf{z}. \tag{11}$$

4.3.3 Dynamic Target Resizing

We additionally use the reference plane coordinates to resize the target bounding box dynamically during tracking. Figure 15 illustrates this process for two video frames of the same target. Given reference plane coordinates $\mathbf{z} = (x, y)$, we project the 3D points representing the bottom and top of the target, $(x, y, 0, 1)^T$ and $(x, y, Z_t, 1)^T$, respectively, to image coordinates using projection matrix P, where Z_t is the approximate height of the target. The difference gives us the height of the bounding box, in image pixels, and we assume the width to be $\frac{1}{3}$ of this value.

4.3.4 Particle Filter Tracking

A particle filter [3] estimates a nonparametric posterior distribution of the target object's state represented as a set of particles. We denote the j^{th} particle at time t as

$$o_t^{(j)} = \begin{bmatrix} \mathbf{z} & \dot{\mathbf{z}} \end{bmatrix} \tag{12}$$

where $\mathbf{z}$ and $\dot{\mathbf{z}}$ are the position and velocity, respectively, of the target. At time $t = 0$, the positions of the initial set of particles are manually initialized to the starting location of the object. In the case where initial velocity estimates are not provided, we randomly sample the velocity from a 2D, zero-mean, Gaussian distribution where the covariance is set to be the average velocity of object motion. After object template initialization, we iterate through prediction, detection, and resampling at each timestep for online tracking.

4.3.5 Prediction

For each particle $o_{t-1}^{(j)}$ we predict the new state at time t:

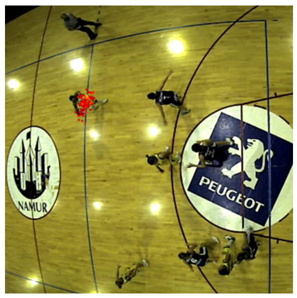
$$\mathbf{z_t^{(j)}} = \mathbf{z_{t-1}^{(j)}} + \dot{\mathbf{z}}\mathbf{_t^{(j)}}$$

$$\dot{\mathbf{z}}\mathbf{_t^{(j)}} = \dot{\mathbf{z}}\mathbf{_{t-1}^{(j)}} + \xi$$

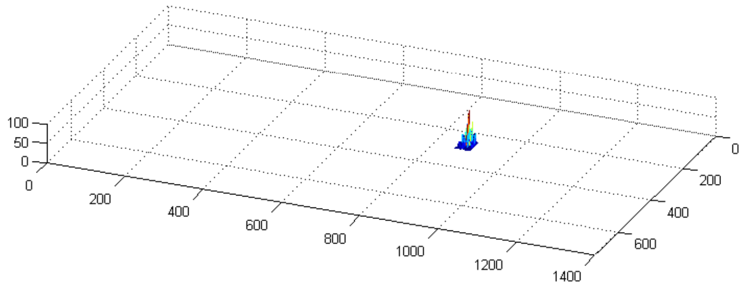
where ξ is a noise term.

4.3.6 Detection

To match two n-dimensional image patch histograms, q and p, for similarity, we employ the Bhattacharyya coefficient [45], a measure of similarity between two dis-



(a)



(b)

Figure 16: (a) The particles for a given target are shown projected to image coordinates. Each particle is scored based on the similarity of its associated image patch with the reference template. (b) The particles form a distribution over the target's state, shown with respect to the reference plane.

tributions:

$$\beta(q,p) = \sum_{u=1}^{n} \sqrt{q_u p_u} \tag{13}$$

Equation 13 is used to calculate the probability, $\rho(q, o, P)$, that the state represented by a given particle, $o_t^{(j)}$, coincides with the reference template, q_T. The position, in reference coordinates, $\mathbf{z_t^{(j)}}$, is converted to image coordinates using camera projection matrix, P, and Equation 11, and the candidate object distribution, q', is calculated using Equation 10. The *detection score*, ρ, is given as:

$$\rho(q_T, o, P) = \beta\left(q_T, q'\right) \tag{14}$$

For the set of particles, $\mathbf{o_t}$, the detection scores, ρ, are normalized to give a non-parametric distribution of the target's state. The state of the object, $\hat{o}_t$, at time t is the maximum likelihood estimate (MLE) of this posterior distribution. Figure 16 shows an example particle distribution. To continue tracking, the particles are resampled from this distribution to generate the set of particles for time $t + 1$. The prediction and detection steps iterate on the next frame.

4.4 Camera Coordination Model

In the standard prediction-detection paradigm for object tracking, computation is typically dominated by the detection step, when many image locations are compared to the target's template. In this section, we describe our coordination model for multi-camera object tracking, which overcomes some of the issues associated with single-camera tracking, such as drift and occlusion, and also reduces overall computation by minimizing the aggregate number of detection processes needed in the complete system.

Let $\mathcal{C} = \{1, 2, \ldots, N_C\}$ be the set of identifiers for N_C cameras in a multi-camera network. We partition $\mathcal{C}$ into three subsets: *active* ($\mathcal{C}_A$), *passive* ($\mathcal{C}_P$), and *inactive* ($\mathcal{C}_I$) cameras. Active cameras track, as described in Section 4.3. Passive cameras are available, but are not actively tracking, and inactive cameras either do not contain the target within the field of view or are otherwise disabled (e.g., power saving). Our meta-algorithm for camera coordination iterates through tracking and reassignment steps.

4.4.1 Tracking

At time step t, for each active camera $i \in \mathcal{C}_A$, we obtain the state estimate $o_t^{(i)} = \begin{bmatrix} \mathbf{z}^{(i)} & \dot{\mathbf{z}}^{(i)} \end{bmatrix}$ and detection score, $\hat{\rho}^{(i)}$, for the target. Each camera $j \in (\mathcal{C}_A \cup \mathcal{C}_P)$ evaluates the target state estimates, $\{\hat{o}_t^{(i)}\}$, identified by the active cameras and returns detection scores $\rho_j^{(i)}$. In the case of multiple active cameras ($|\mathcal{C}_A| > 1$), we determine an aggregate state estimate using a simple voting scheme. Each camera, j, selects the state estimate from active camera i with the highest value of $\rho_j^{(i)}$. The state estimate with the highest number of votes is selected as the network's estimate for time step t. This voting approach differs from previous multi-camera methods [62, 57] that directly compare detection scores across cameras.

4.4.2 Reassignment

After tracking in frame t, cameras are reassigned to the active, passive, or inactive sets for frame $t+1$. Cameras whose state estimate indicates the object is out of the field of view or are otherwise disabled are labeled as inactive. Using a preference function, we rank the remaining cameras. This function could incorporate multiple factors, such as power remaining, long-term camera reliability, or priors from an environment model. In our experiments, we evaluate two preference functions. The first uses the detection scores as direct measures of tracking confidence. The second approach relies on the relative change in detection score. Due to differing color calibration, the size of the target in the frame, or occlusions in the scene, raw detection scores may not be meaningful between cameras. As an alternative, we also evaluated the relative change in detection score, $(\rho_t - \rho_{t-1})/\rho_{t-1}$, as a preference function. Based on the preference function, the top T cameras become active, and the remaining are passive.

We iterate through these two steps, tracking and reassignment, for each frame of the video. In the tracking step, the computation savings arise for inactive and passive cameras. Rather than evaluating a large number of locations in the detection step (typically hundreds using particle filters), only a few locations (those returned by the active cameras) need to be evaluated. This savings would be realized not only

Scenario	Number of Frames	Target Acceleration	Occlusion Severity
1	420	Medium	Medium
2	200	Low	High
3	250	High	Low

Table 2: Description of the three test scenarios.

for particle filter tracking, but for any tracker using the prediction-detection model. In the next section, we show how this coordination method improves tracking on a challenging data set.

4.5 Results

To test our method, we used the APIDIS dataset (Section 3.5). We selected three tracking scenarios (summarized in Table 2), which contain a mix of sudden target acceleration, difficult occlusions across multiple cameras, and instances of the target exiting the view of one camera and entering another.

4.5.1 Single-Camera Tracking

As a baseline, we performed single-camera tracking (as described in Section 4.3) using 200 particles per camera for tracking. The error metric is the distance on the reference plane from the target's estimated position to the ground truth position. Figure 17 shows the mean tracking error for three different cameras. (Only cameras 2, 3, and 5 contain a view of the target for the duration of the tracking experiment.) Error is measured in feet on the ground plane. The high error values (i.e., mean error greater than 3 feet) usually indicate the tracker became "lost" and the target was completely mistracked. For scenarios 1 and 3, the best-performing individual cameras (3 and 5, respectively) were able to track the target successfully. So, even in

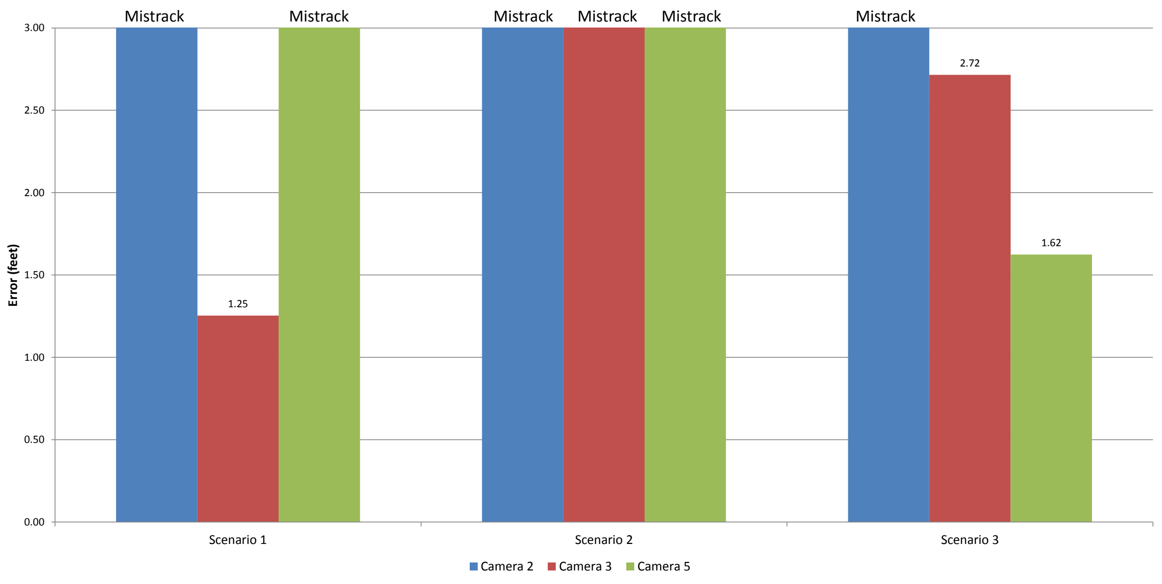


Figure 17: Mean single-camera error for each scenario.

cases of high target acceleration or moderate occlusion, the base tracker can perform well. However, for scenario 2, no individual camera was able to successfully track the target due to an occlusion with multiple players in the same area.

4.5.2 Coordinated Tracking

We tested our multi-camera coordinated model using up to four active cameras.[2] For reassignment, we evaluated the two strategies described in Section 4.4, which we call *Score* and *RelScore*. As with the single-camera experiments, each active camera used 200 particles for tracking. Figure 18 shows the mean error for the *Score* coordination scheme over the same three scenarios from the APIDIS data. Figure 19 shows the mean error for *RelScore*. Both multi-camera approaches outperformed the individual cameras.

The first method, *Score*, which uses the raw detection scores as the preference

[2]Even though the network contains seven cameras, a single object often does not appear in more than four camera views.

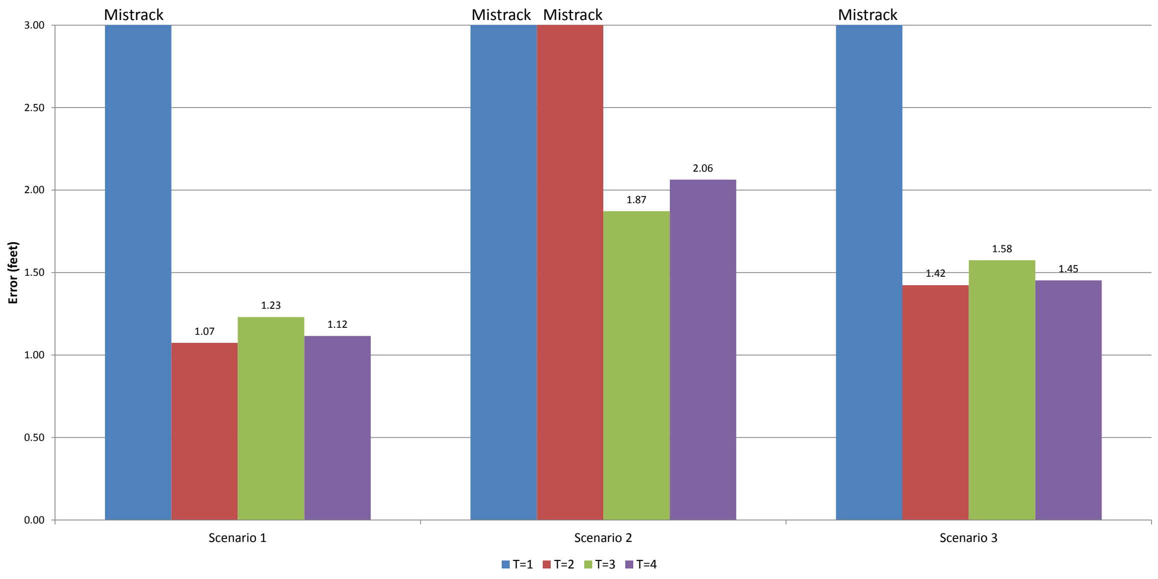


Figure 18: Mean multi-camera error for each scenario using the *Score* method for varying number of active cameras.

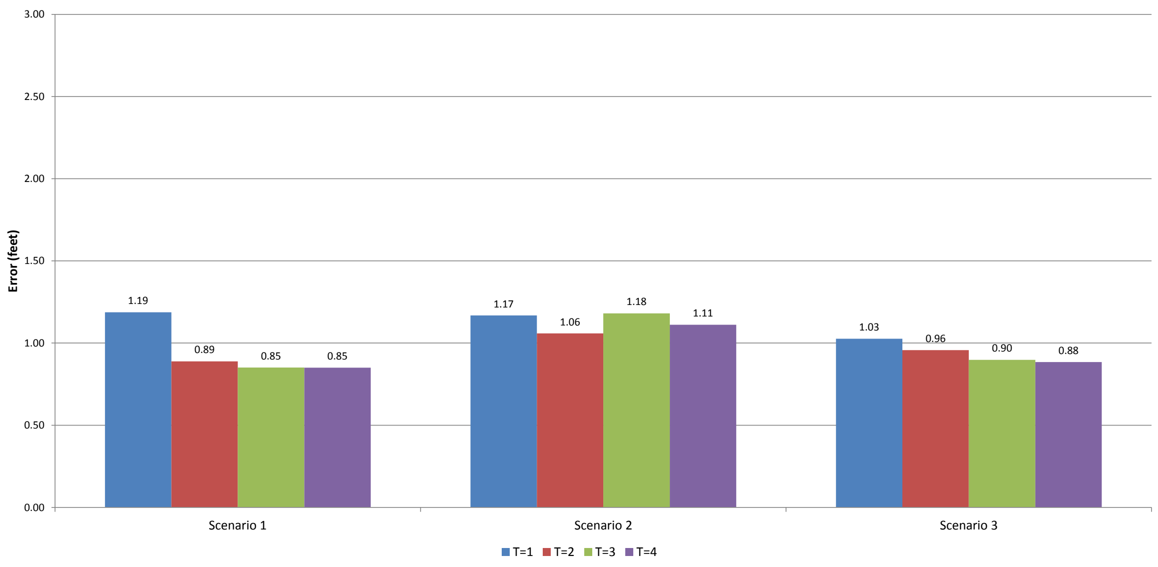


Figure 19: Mean multi-camera error for each scenario using the *RelScore* method for varying number of active cameras.

function performs poorly in the case of a single active camera ($T = 1$), where the target is mistracked in 2 of the scenarios. This method is particularly vulnerable to a single high-scoring camera dominating the rankings. This effect appears to be mitigated at $T = 2$, even though the target is still mistracked in Scenario 2. With $T \geq 3$, the target in each scenario is successfully tracked, however with higher error rates than the *RelScore* approach. The second multi-camera method, *RelScore*, performed the best out of the three tracking methods. Even with a single active camera ($T = 1$), the method is able successfully track the target in all three scenarios, and error generally decreases as successive cameras are added. Over all the experiments, the *RelScore* scheme achieved a mean error of 1.00, compared to 3.58 and 14.10 for the *Score* and single-camera methods, respectively. Figure 20 illustrates how the system switches among cameras with $T = 2$ active cameras over scenario 2. Figures 21 and 22 show our method handling situations of mistracking and occlusion, respectively.

4.5.3 Computational Load

To measure the computational load of our approach, we focus on what is typically the most computationally expensive step in tracking, evaluating a candidate location. In particle filter tracking, this step occurs once for each particle in each frame. In our approach, the number of evaluations is reduced, as only active cameras evaluate all of the particles, and non-active cameras perform just T evaluations, where T is the number of active cameras. For the multi-camera case with a single active camera ($T = 1$) and 200 particles for tracking, 206 detections are carried out per frame, a reduction of 85% of the computational effort compared to when all the cameras track

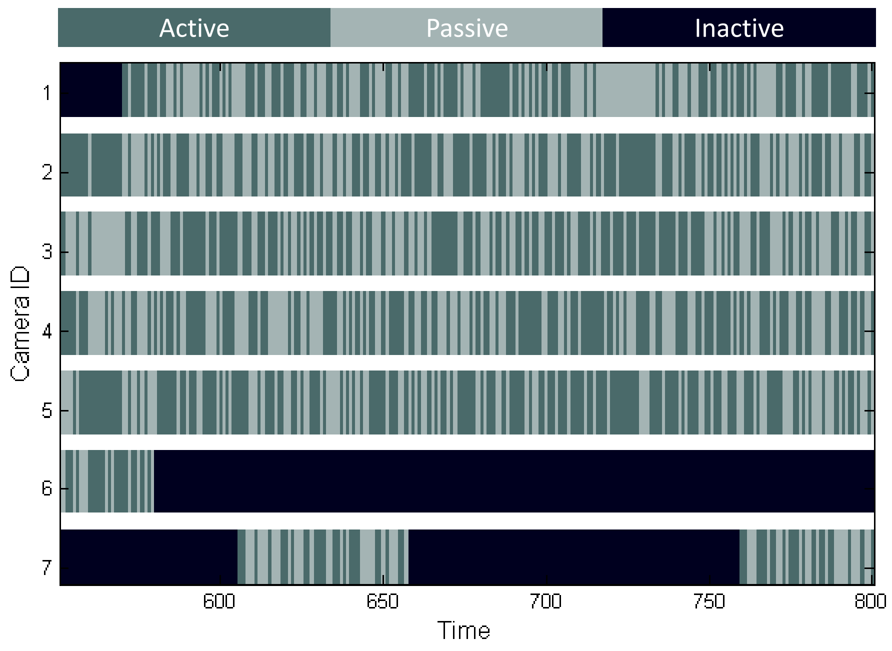


Figure 20: This timeline depicts the state of each camera for tracking an object using $T = 2$ active cameras.

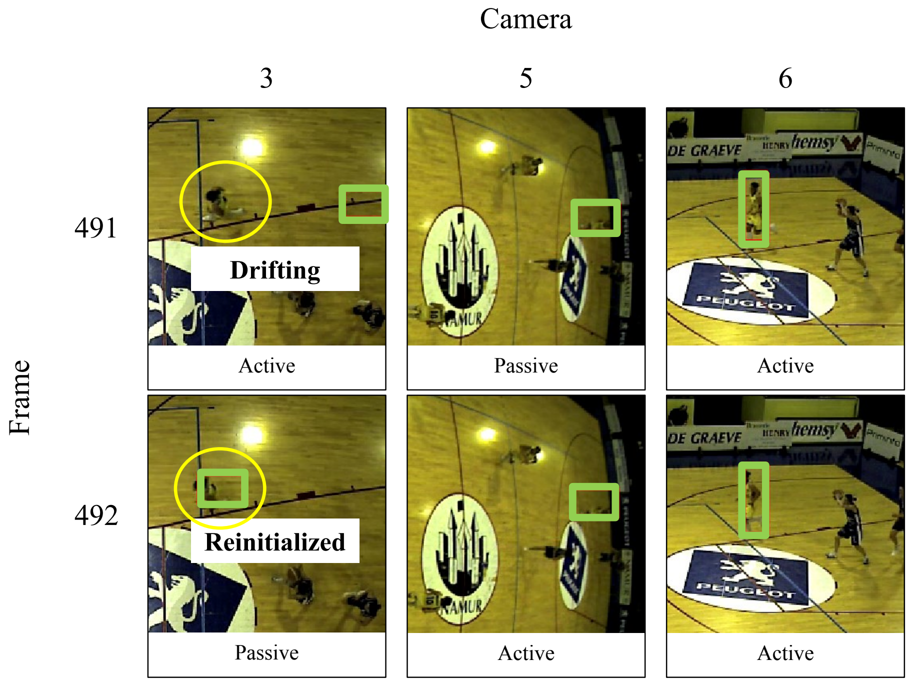


Figure 21: Cooperative camera switching for three cameras. In the top row (frame 491), camera 3 has drifted. In the bottom row (frame 492), camera 5 becomes active, and camera 3 reinitializes using the estimates from the best cameras.

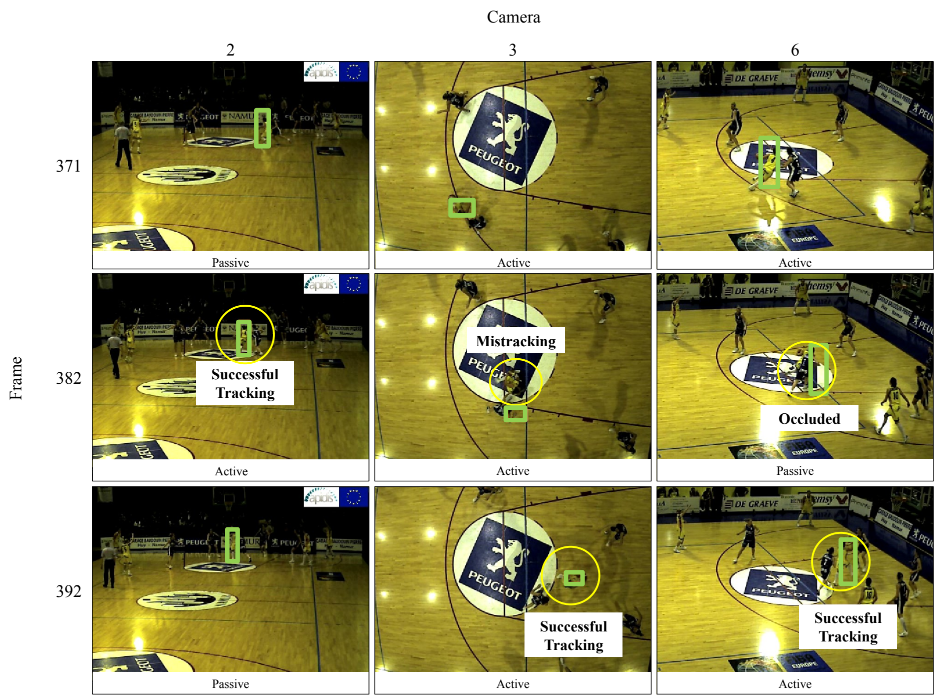


Figure 22: Pre-occlusion (frame 371), cameras 3 and 6 are active. During the occlusion (frame 382), camera 2 maintains a good view of the target and becomes active. Post-occlusion (frame 392), cameras 3 and 6 become active again.

independently, which is the approach taken with typical trajectory correspondence tracking methods. While adding active cameras increases the computational load, this number can be selected to balance the tradeoff between efficiency and tracking accuracy.

4.6 Summary

We have presented a coordination framework for object tracking in multi-camera networks. The main contribution lies in the coordination model, which allocates resources dynamically to the best cameras at each time step. We introduced a voting scheme to aggregate multiple tracking estimates and a flexible preference function for dynamically switching between cameras. In contrast to typical approaches to multi-camera tracking, which aggregate data from all available sensors, our method reduces resource requirements, making it suitable for real-time application.

CHAPTER 5: POSE ESTIMATION

5.1 Introduction

Human pose estimation is a key step in automated human behavior understanding from video. Specifically, in this chapter, we focus on head pose estimation. Head pose provides cues to a subject's attention and focus, which can be important for applications in surveillance, marketing, and HCI. Multi-camera networks are well-suited to support these applications; however, in the most common deployments, cameras observe a wide field-of-view, and a person occupies a small area of image, with heads sometimes as small as 20 pixels. In addition, the motion of people in the scene introduces challenges due to changes in scale and perspective.

In this chapter, we propose a computationally efficient method for head pose estimation in multi-camera networks that allows for an explicit trade-off between speed and accuracy. Our approach is based on an ensemble of exemplars, which can be used to build a strong predictor using relatively simple features. Finally, we introduce a dynamic camera selection scheme, which allows the system to use the prediction from fewer cameras in easy cases (e.g., large faces, visible facial features) and more cameras in cases of ambiguity. Our main contributions are (1) providing fine-grained predictions of head pose angles, (2) adapting exemplar classification to perform regression for the problem of head pose estimation, and (3) dynamic camera selection

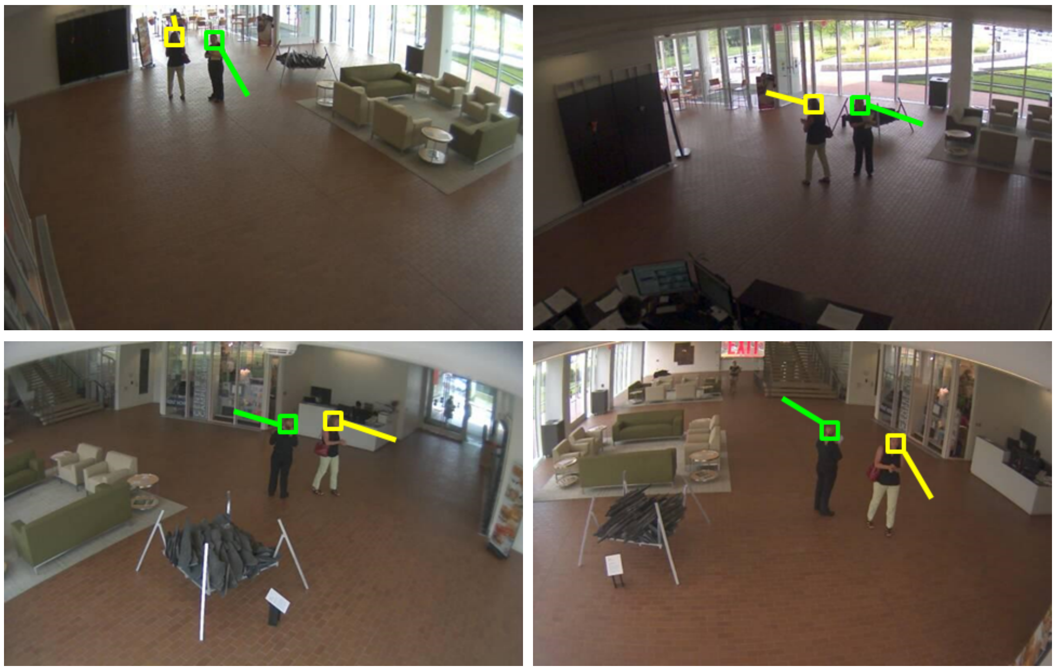


Figure 23: In multi-camera networks, head pose estimation can support applications in surveillance, marketing and HCI. Our method uses an ensemble of exemplars to provide finer-grained predictions than previous approaches.

for computational efficiency.

5.2 Related Work

Head pose estimation is often used as a proxy for gaze estimation [60]. In the cases where facial features are readily identifiable from images, gaze direction can be estimated directly via eye detection and pupil tracking [35]. At medium-scale resolutions, some approaches rely on locating salient features such as the eyes, ears, and nose [103]. Several recent efforts have sought to estimate head pose relative to a single camera from low-resolution images. One approach introduced a descriptor based on Kullback-Leibler distance applied to facial appearance [63]. Tosato et al. describe a new feature descriptor (ARCO) targeted at vision tasks in low-resolution images [79].

For the multi-camera setting, approaches tend to fall into one of several categories. Some work has investigated synthesizing 3D head shapes, e.g., ellipsoids [8] or spheres [94]. These methods tend to be computationally expensive and require many cameras. Other methods concatenate images from network cameras to learn a single discriminative function [44]. The most common approach applies existing monocular head pose techniques separately to individual views, computing relative pose (or a probability distribution of relative pose) for each camera and combining to compute the absolute pose estimate [84, 99, 59]. Our approach is most similar to these in style, but, unlike these methods, is applicable to the case of moving targets.

Some recent work has specifically addressed head pose estimation of moving people in wide-field-of-view, multi-camera settings. Yan et al. proposed a multi-task learning scheme to model how head appearance changes with respect to position within an environment [96, 97]. Other methods have incorporated transfer learning to leverage information from datasets with stationary people [69]. These approaches treat head pose estimation as a classification task and provide coarse predictions of head pose as one of a small number of pre-defined directions. Our method is designed for continuous pose estimation for moving people in low-resolution images. Further, it has low computational requirements, necessitating neither complex features nor expensive 3D model fitting.

5.3 Method

The focus of this chapter is to estimate the head pan (azimuth) and tilt (elevation) angles with respect to a global coordinate system in calibrated, multi-camera

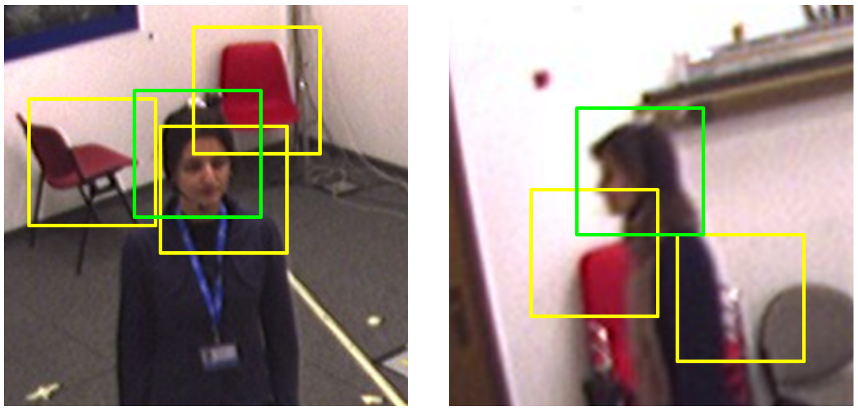


Figure 24: For joint localization and pose estimation, multiple patches are evaluated at each camera. The highest-scoring detection is shown in green.

networks. This is normally one step in a pipeline that includes detection and tracking, so we assume that the target has been localized (e.g., bounding box in each camera). This leaves the problems of head localization and pose estimation. Previous approaches consider these two issues separately and often employ computationally expensive methods for head localization [69]. In this section, we describe our computationally-efficient, joint approach to head localization and pose estimation from a single camera, and also the aggregation scheme for multi-camera networks.

5.3.1 Single-Camera Head Pose Estimation

Previous work has shown that most features used for head pose estimation are sensitive to localization, especially in the case where the targets move freely [97]. Given a tracked target in a multi-camera network, a rough localization of the head can be obtained using simple rules (e.g., top third of of the target's bounding box). We use a sliding window approach, as shown in Figure 24, for evaluating multiple locations and a prediction scheme that provides a confidence level associated with the prediction.

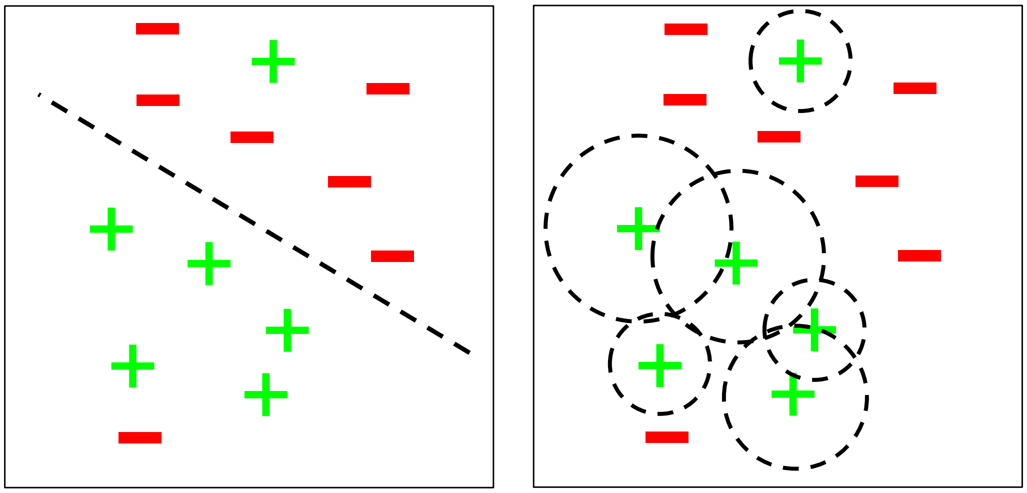


Figure 25: (Left) Most learning methods fit a global model to the training data. (Right) Exemplar SVMs learn local models centered on individual exemplars. The ensemble of the local learners represents a complex prediction model.

Exemplar SVM

For joint localization and estimation, we propose the use of an ensemble of exemplar SVMs (ESVM), which has been previously applied to object detection [55], tattoo recognition [91], and place recognition [20]. Figure 25 provides a visual overview of ESVM, where local detectors are trained using a single (positive) exemplar. Figure 26 shows positive and negative image patches for the problem of head pose estimation, where negative examples are gathered from images of the scene with no people present. Each local model can be considered as a binary classifier for the metadata (e.g., head pose angle) associated with the training exemplar.

It is necessary to calibrate the predictions of the local models in order to obtain output values that can be directly compared as confidence values of a query matching the local models. Calibration requires a separate training stage, using only labeled image patches containing heads. For each exemplar, positive examples are those

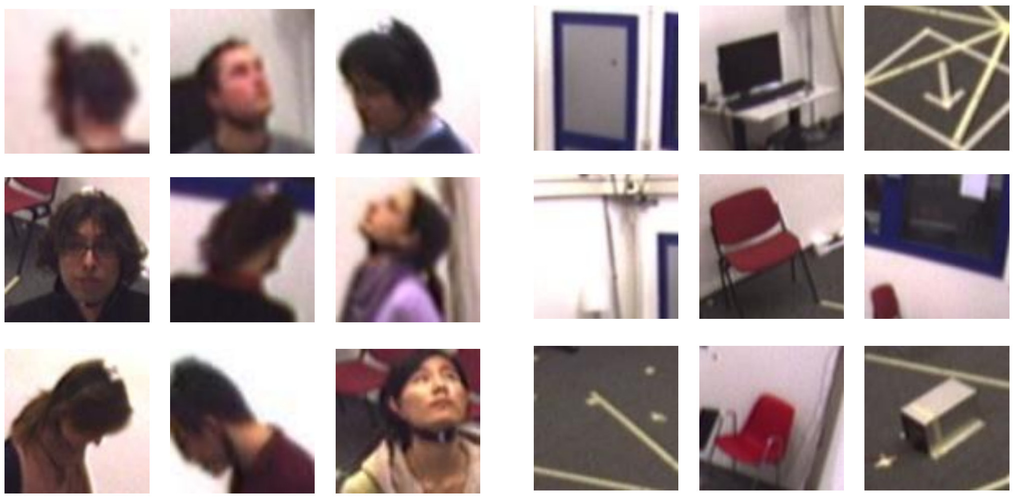
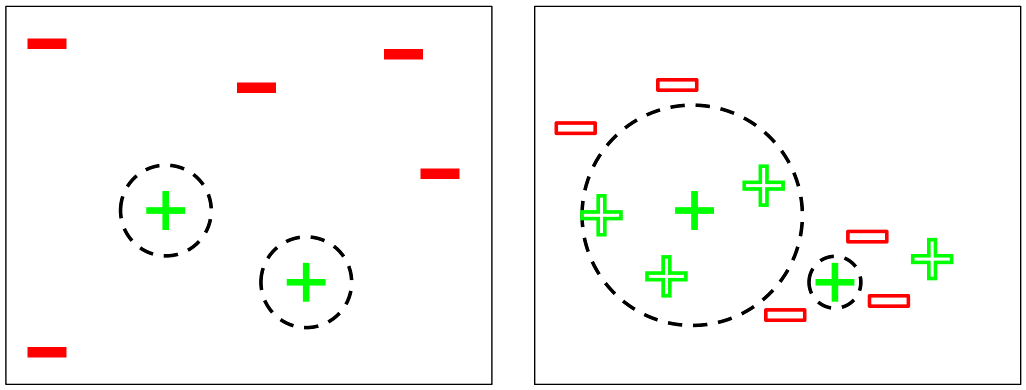


Figure 26: Exemplar SVM models are trained using a single positive training example (left) and many negative examples (right).

patches for which the labels "match" and the remaining are negative. For example, in the case of head pan angle estimation, positive examples would correspond to image patches with pan angles within a specified threshold of the exemplar. As shown in Figure 27, calibration has the effect of dampening the output of less reliable detectors, while amplifying those that generalize better. Platt scaling [66] is used to convert the raw SVM output of the post-calibration model to a probability value, which can be used as a detector confidence value. Figure 28 shows the top 5 exemplar detections for a query image before and after calibration.

Single-Camera Algorithm

Let $\mathcal{D}$ represent the set of N_E trained exemplar detectors, as described above. Each detector, D_i, is associated with the label, y_i, of the corresponding exemplar, and a scoring function, $\omega_i(\cdot)$, that returns the Platt-scaled probability for a query example, $\mathbf{x_q}$. A query example, $\mathbf{x_q}$, corresponds to the feature representation for an image



(a) Pre-calibration (b) Post-calibration

Figure 27: For an ensemble of exemplars, calibration has the effect of dampening the output of less reliable detectors, while amplifying those that generalize better.

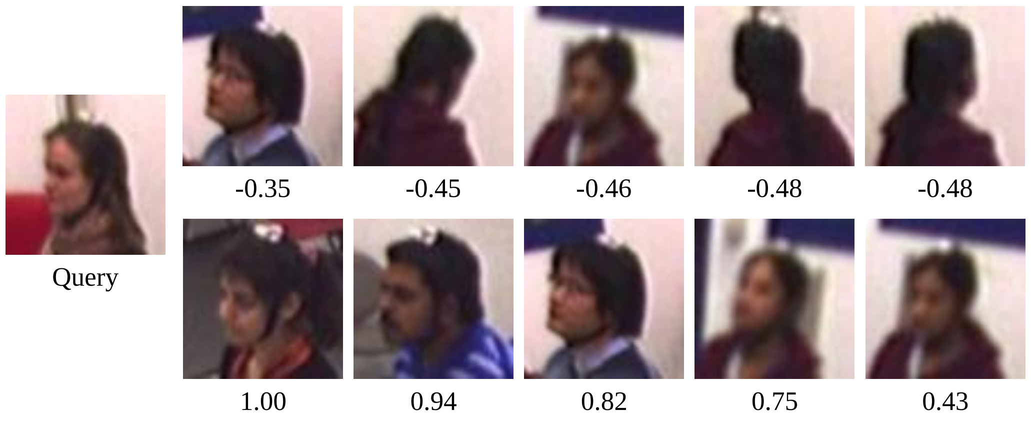


Figure 28: For a query example (left), the top-scoring exemplars are shown before (top) and after (bottom) calibration.

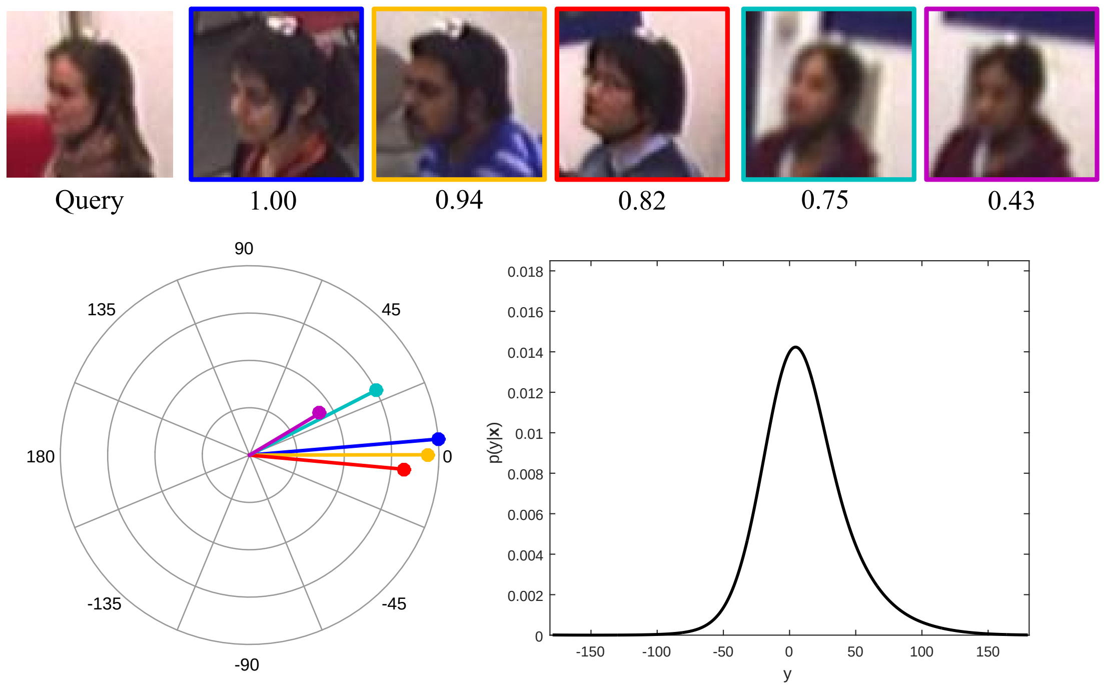


Figure 29: A query image and the top scoring exemplars are shown (top). The radial plot (bottom-left) shows the predicted pan angle and detector score for each of the corresponding exemplars. The individual pose angle estimates are combined into an ensemble estimate, represented as a PDF over the range of pan angles (bottom-right).

patch extracted from a roughly localized image window.

For each query in the search area, we obtain the top N_S scoring detectors. The query with the best matches to head pose exemplars is retained as the head location prediction. To predict the head orientation at this location, we consider each of the top matching detectors as a noisy predictor of the query label, y_q, and model the ensemble prediction using a Mixture of Gaussians model. The contribution of each detector, D_i, is represented by a Gaussian with mean $\mu = y_i$ and standard deviation $\sigma = \frac{1}{\alpha\omega_i(\mathbf{x_q})}$, where α is a scaling parameter. Figure 29 shows an example of predicting the head pan angle of a query image using this approach.

5.3.2 Multi-Camera Head Pose Estimation

In a multi-camera network, predictions from multiple cameras can be aggregated by multiplying the resulting probability density functions from each camera. For clarity, all references to direction-based predictions are assumed to be from a global coordinate frame. Figure 30 shows an example of multi-camera prediction of head pan angle. This example is representative of the typical case where cameras that observe the front of the target's face provide more confident predictions. For the multi-camera system, the final prediction can be taken as the mode of the combined PDF.

The observation that certain viewpoints in a multi-camera setting are preferable motivates our approach for dynamic camera selection. Rather than aggregating the predictions from all the cameras in the network at once, we sequentially incorporate single-camera predictions until sufficient confidence is achieved. To estimate the number of cameras to sequentially sample to make a prediction, we incorporate the multi-class sequential probability ratio test of Davis and Tyagi [18].

To apply the ratio test, we discretize the probability density function $p(y|\mathbf{x})$ of the per-camera prediction. For a discretized label, y, the ratio is defined as:

$$r(y|\mathbf{x}_{1:c}) = \frac{P(y|\mathbf{x}_{1:c})}{\sum_{y' \neq y} P(y'|\mathbf{x}_{1:c})} \tag{15}$$

where $\mathbf{x}_{1:c}$ denotes the input from a sequence of c cameras. The class conditional probabilities, $P(y|\mathbf{x}_{1:c})$, are estimated using the Naive Bayes and uniform priors as-

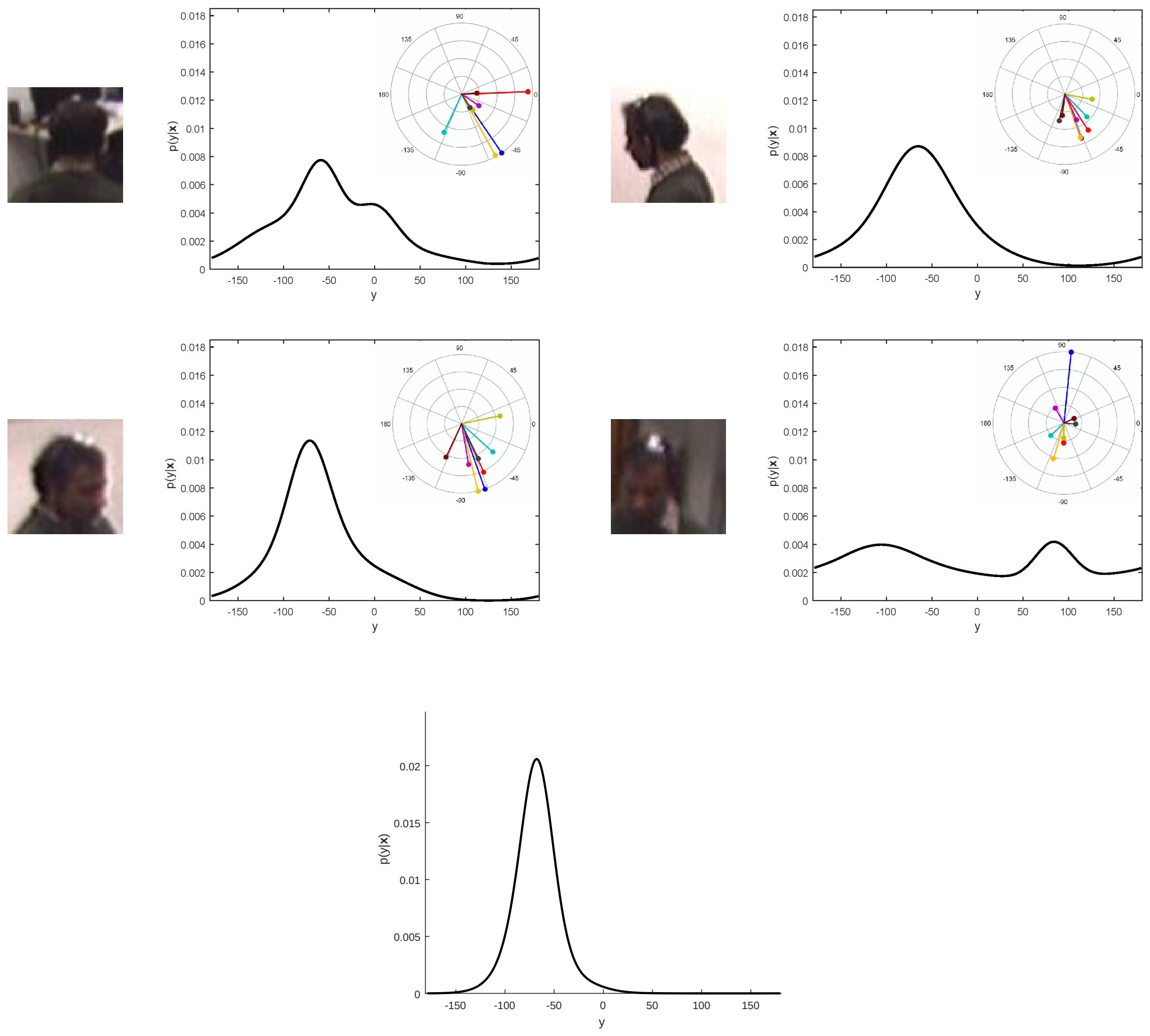


Figure 30: (Top) For the query image from each view, the top-scoring exemplar estimates are used to compute probabilistic predictions, which are combined to give the system prediction (bottom).

sumptions:

$$P(y|\mathbf{x}_{1:c}) = P(y|\mathbf{x}_{1:c-1})P(y|\mathbf{x}_c) \tag{16}$$

A prediction is made for a class when the ratio is greater than a user-specified threshold, τ_r. A ratio greater than 1 indicates that the probability for a particular class is greater than the sum of the other choices.

5.3.3 Method Summary

Our method generalizes an efficient single-camera algorithm for head pose estimation to the multi-camera setting. It is applicable to a wide variety of multi-camera configurations, and, with dynamic camera selection, the computational efficiency does not necessarily grow with the number of cameras in the network. In the next section, we evaluate our approach on a benchmark dataset.

5.4 Results

We evaluate our method on DPOSE [44], a publicly-available dataset for multi-camera head pose estimation, consisting of over 50,000 frames of 16 moving people captured by 4 calibrated, synchronized cameras. Figure 31 shows example frames from DPOSE, with a zoomed-in crop of the localized head region.

5.4.1 Exemplar Head Pose Learning

Targets are tracked using a multi-camera tracking algorithm that estimates a 3D bounding cube for each target [74]. The initial head search area and window size is based on the projected size of the target in a camera. From each rough localization image patch, square image patches are extracted and scaled to 70 x 70 pixels and represented using HOG features [17] with 7x7 cells and the 31-dimensional descriptor of Felzenszwalb et al. [31]. For ESVM learning, training and validation examples are randomly selected from DPOSE. For each training example, an exemplar model is trained. Negative examples are extracted from background images of the scene known not to contain people. Each exemplar model is calibrated using the validation

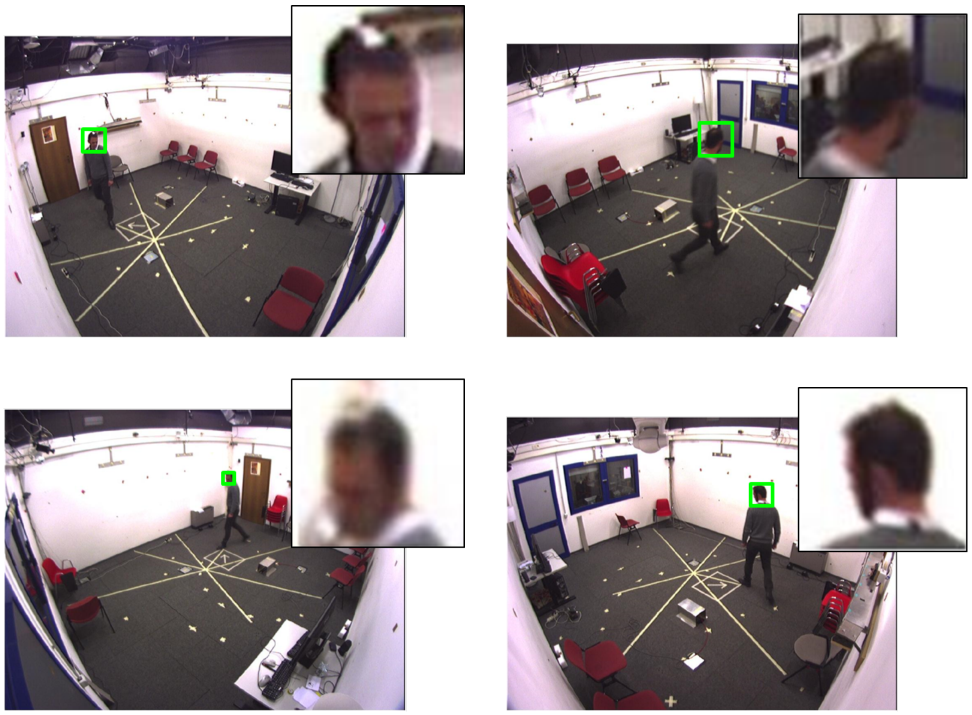


Figure 31: The DPOSE dataset consists of labeled images from multiple people observed by four cameras.

examples. For head pose angle estimation, examples where the angle difference between exemplar and validation example is less than 10 degrees are taken as positives, and those greater than 90 degrees are negatives. Figure 32 shows the top matching exemplars for sample query examples from DPOSE.

5.4.2 Head Pose Estimation

Our approach provides real-valued predictions for both pan and tilt angles. To the best of our knowledge, no previous work has reported real-valued predictions on the DPOSE dataset for the problem of head pose estimation in multi-camera networks. Here, we compare several variants of our method:

- *Single-camera (SC)* is the baseline approach, evaluated per-camera.

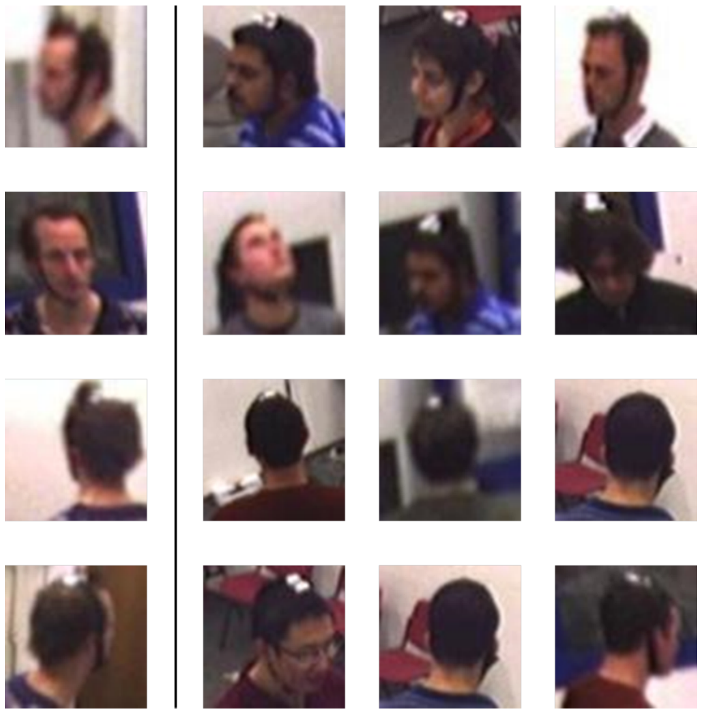


Figure 32: For the image patch from each camera (column 1), the top scoring exemplars are shown.

- *Best-camera (BC)* applies SC to each camera and returns the highest-confidence estimate.
- *All-cameras (AC)* applies SC to each camera and aggregates the predictions.
- *Variable-cameras (VC)* incorporates our dynamic camera selection scheme.

Each method used the same ensemble of exemplars. Two sets of head pose patches were used to train the models, 960 examples each for training and validation. The scaling constant, $\alpha = 0.05$ and the number of top-scoring exemplars, $N_S = 25$. In practice, our algorithm is robust to a range of values for these parameters. Figure 33 shows the results for head pose localization on DPOSE reported as the mean absolute error of the prediction compared to the provided ground truth over 1000 testing examples, averaged over 5 trials. The multi-camera methods (AC and VC) outper-

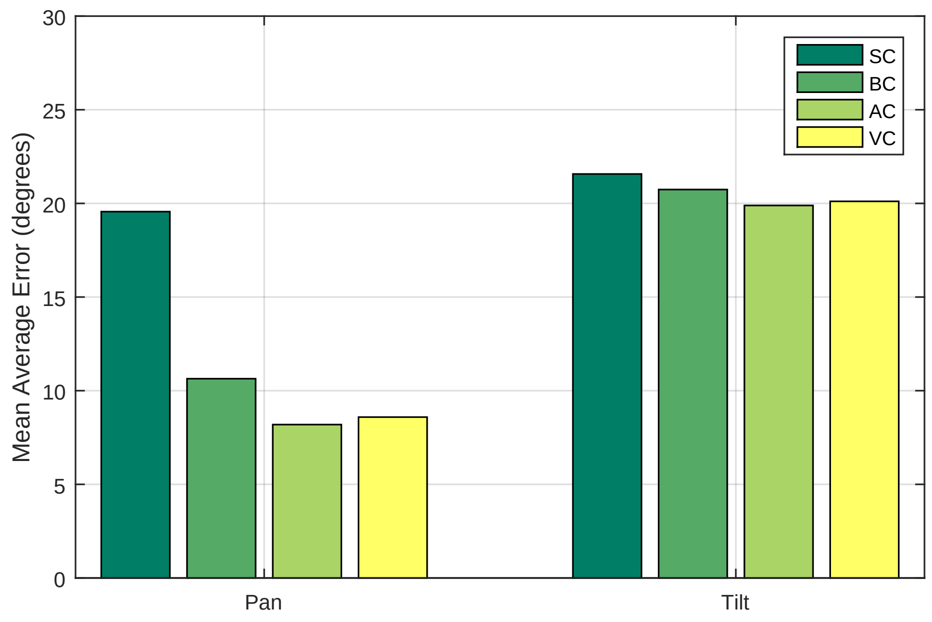


Figure 33: Mean absolute error for head pan and tilt angle estimation.

formed the single-camera methods (SC and BC), with the dynamic camera selection scheme (VC) performing comparably to the all-cameras (AC), with less computation. Figure 34 shows example predictions from our VC method.

For the variable-cameras (VC) method, the confidence threshold serves as a tunable parameter that changes the behavior from the single-camera to all-cameras paradigms. As such, pose estimation error decreases and the number of cameras sampled increases as the confidence threshold increases. Figure 35 shows these trends for head pan angle estimation with DPOSE. In our experiments, we set the variable camera threshold, $\tau_r = 1.0$.

5.4.3 Discrete Head Pose Classification

Previous methods that have used DPOSE have only provided predictions for head pan angle into one of eight 45° bins. To compare our results to recent related work,

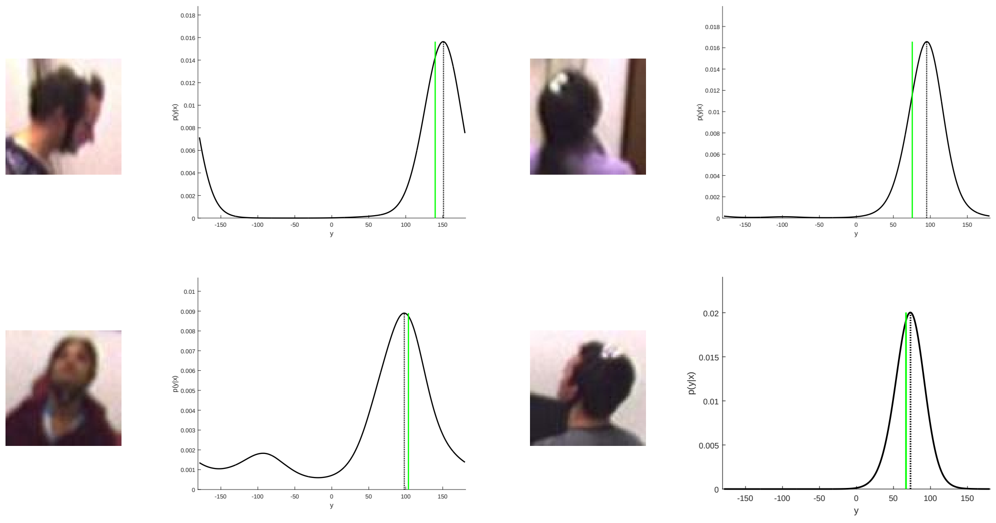


Figure 34: Head pan angle predictions from our VC method for selected DPOSE image patches. Ground truth is shown in green, VC estimate in black.

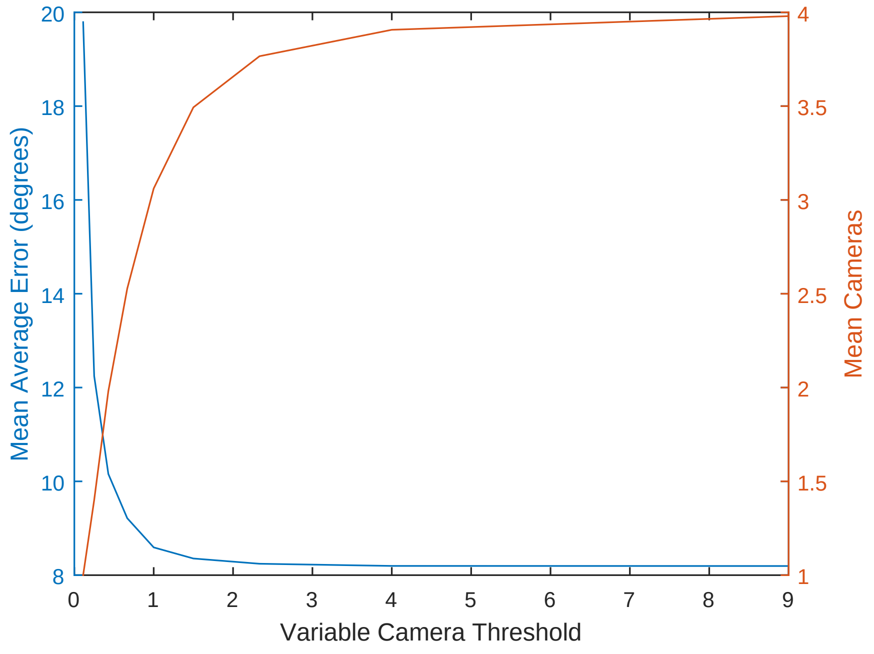


Figure 35: For variable-camera selection, pose estimation error decreases and the number of cameras sampled increases as the threshold increases.

Table 3: Discrete head pose classification accuracy.

Method	Accuracy
Ours	85.22%
Yan 2013	86.10%
Yan 2014 (HOG)	80.00%
Yan 2014 (HOG+KL)	87.00%

we follow the ensemble exemplar learning approach previously described with the modification that the label associated with each exemplar corresponds to one of 8 classes rather than the provided real-value ground truth. We follow the same experimental protocol as other recent work [97]. For training, the scene is divided into four quadrants and 30 training examples are randomly selected from each region for each of 8 quantized head poses. Results are averaged over 5 trials.

While our method was designed to provide precise real-valued predictions of head pose, it is competitive with the state of the art for the discrete classification task. Figure 36 shows example results from our method for this discrete prediction task. Closer inspection of the results shows that most of the errors in our variable-camera approach tend to lie within one discrete bin of the true pose. Since our method on this data achieves a mean absolute error of 8.59 degrees, it is likely that some portion of the misclassifications are due to quantization artifacts at the boundaries of the artificially-defined classes. Figure 37 shows the confusion matrix for the variable-camera pan classification experiment. Each row represents the true pose angle, and each column the angle predicted by our method. The diagonal banding illustrates the tendency of errors to fall in neighboring classes.

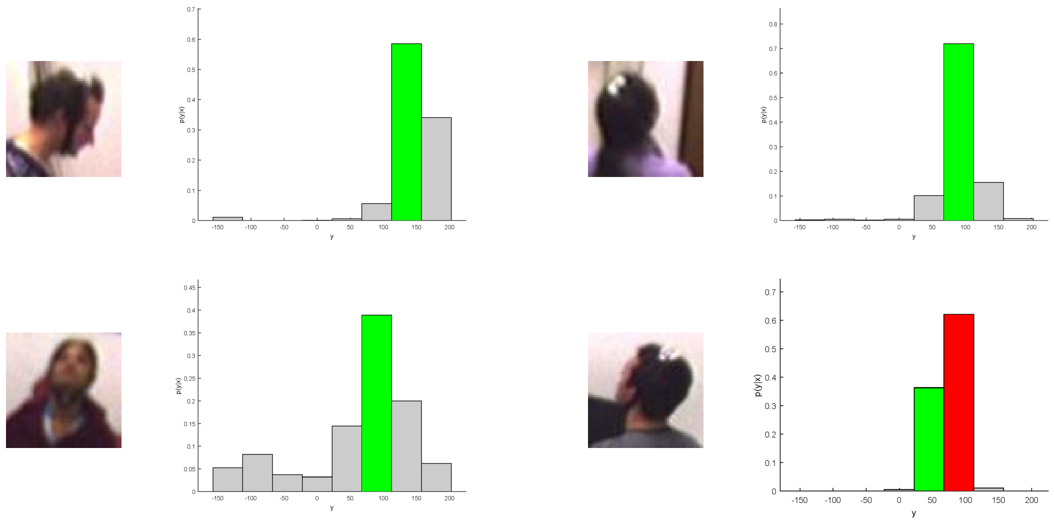


Figure 36: For each image patch, the ground truth class is shown in green. An incorrect classification is shown in red.

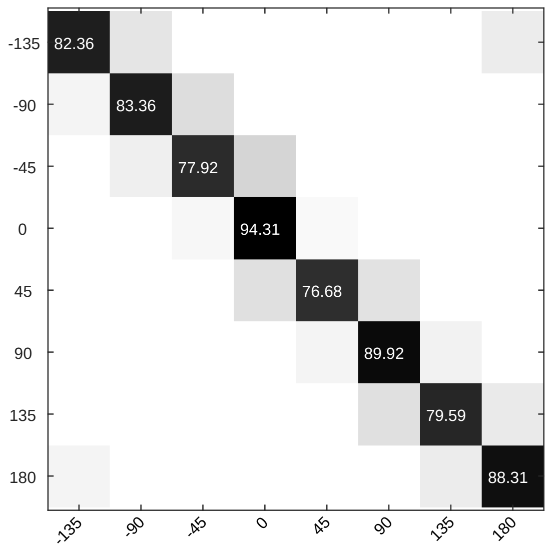


Figure 37: Confusion matrix for discrete pan angle classification on DPOSE.

5.5 Summary

In this chapter, we described a novel approach for head pose estimation designed for multi-camera networks. Our framework is robust to low-resolution images, poorly localized bounding boxes, and appearance changes induced by changing person location in the scene. The computational requirements are also modest due to the use of inexpensive features and fast linear classifiers. In addition, we described a variable-camera scheme to dynamically select a subset of the available cameras for pose estimation, allowing for explicit trade-off between efficiency and accuracy. Experiments on a benchmark dataset show that our approach provides discrete classification accuracy on par with the state-of-the-art.

CHAPTER 6: ACTION RECOGNITION

6.1 Introduction

In this chapter, we apply dynamic camera selection to multi-camera action recognition, with the goal of predicting the class of an observed action sequence within a distributed camera network. Our method is based on the observation that certain human poses, *keyposes*, are highly discriminative and can provide much of the evidence needed for a prediction by the system. This observation is not new; Schindler and Van Gool [71] framed the problem of action recognition as detecting poses that are highly predictive for particular action classes. We call such poses *class-discriminative*. Our work extends this observation to *shift-discriminative* poses. Shift-discriminative poses are not predictive of a particular class, but indicate potential ambiguity is imminent from the current viewpoint and a *view-shift* is necessary. Figure 38 shows an example containing both types of discriminative poses where a view-shift leads to a view containing a class-discriminative pose. The addition of shift-discriminative poses extends the utility of class-discriminative poses to the multi-camera setting.

Our approach to view-shift learning facilitates multi-camera action recognition that dynamically selects a single camera for use at each time step. The method casts view-shift learning as a Markov decision process and employs reinforcement learning to estimate the utility of the available viewpoints in a distributed camera network for

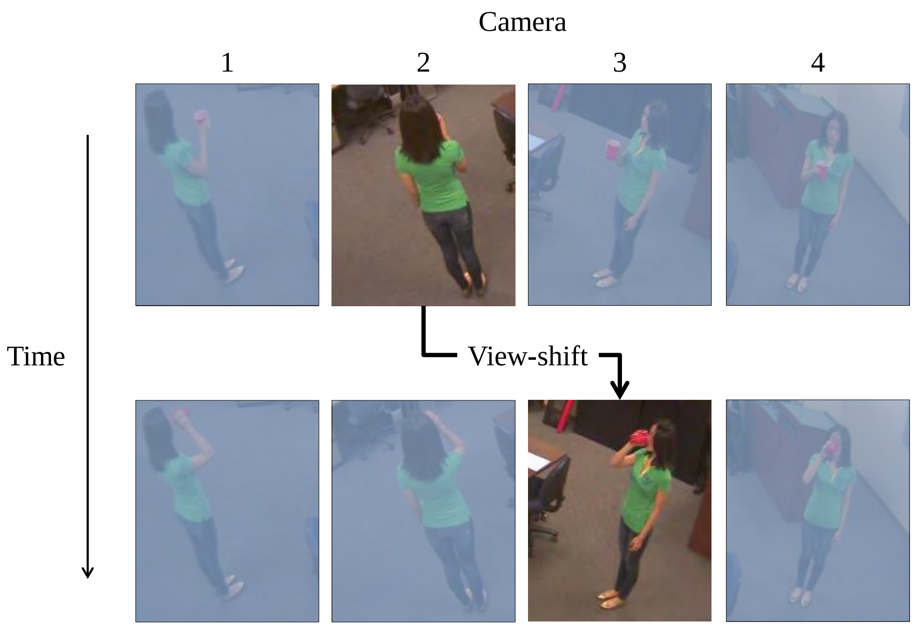


Figure 38: Our method learns if a pose represents potential future ambiguity (top) and which available viewpoints will facilitate disambiguation (bottom). For this "drinking" example, the active camera is changed via a learned *view-shift* to a new view with a more discriminative pose.

human action recognition. Our approach does not require the camera configuration to be the same in training and testing and is applicable to both batch and online recognition. We compare our view-shift method with recent multi-camera approaches and show that it is comparable to the state of the art on two standard benchmark datasets in terms of recognition accuracy, but with greatly reduced computational effort.

6.2 Related Work

There has been extensive work in human action recognition from video; see [67, 90] for recent surveys. Our focus in on multi-camera action recognition in indoor environments. Methods designed for these scenarios often must balance the gains in accuracy from computationally expensive inference schemes with the costs in efficiency. In this section, we organize the related approaches based upon the explicit use of 3D models and/or information from all of the cameras in the network.

6.2.1 4D Recognition

One approach in multi-camera networks is to make use of multi-view geometry to explicitly construct 3D models and solve 4D (3D plus time) recognition problems; several recent methods follow this general approach. Motion history volumes (MHV) encode a person's spatial and temporal motion using the circular Fast Fourier Transform [89]. Turaga et al. incorporate MHV in a manifold-learning-based scheme [81]. These methods are computationally expensive, requiring, in addition to models constructed during training, dynamic construction of models from multi-camera input during testing. Some methods construct 3D models during training, but not testing. In [87], 3D silhouette-based exemplars are learned and matched to 2D observations with model projections. Yan et al. [95] construct 4D models and perform recognition by back projecting 2D features. While these methods require less computational expense at test time than approaches that construct 3D models for both training and recognition, they still incur significant overhead due to the size of the search space to fit model parameters [88].

6.2.2 Multi-View Recognition without 3D models

Other approaches, rather than explicitly constructing 3D models, use a set of 2D image views with some aggregation scheme. Many methods simply extend single-view algorithms by, for example, concatenating features from multiple cameras (e.g., [93, 13, 75]) or using voting schemes among the cameras in the network [53, 64, 102].

Rather than extending single-view methods to camera networks, some approaches present new features designed for the multi-view setting. Souvenir and Babbs introduce a manifold-based model to estimate how observed 2D features change as a function of viewpoint [73]. Farhadi et al. define discriminative aspect to encode how actions look different from different perspectives [29]. In general, these approaches tend to be computationally more efficient than the 4D methods, but still require computation to occur for each camera in the network in addition to the cost of aggregation. Our approach neither fits 3D models nor requires computation at each camera per target at test time. Our approach learns to select the single best view for recognition dynamically.

Our method, which does not require expensive fitting of 3D model parameters, nor computation of features in multiple cameras at each time step, achieves similar accuracy to existing multi-view methods with greatly reduced computational expense by dynamically selecting the best view for the next frame. In the next section, we describe our approach to view selection based on predicted future ambiguity.

6.3 Method

Our action recognition method is designed for systems of calibrated cameras. In particular, we consider architectures with peer camera nodes, where for a given target, there is an *active* camera for detection and tracking, and the remaining cameras are passive (with respect to a particular target). The initial designation can be fixed, where a particular camera is active for targets in a specified region, or dynamic, e.g., the active camera is selected to track a target [85]. In either case, passive views are

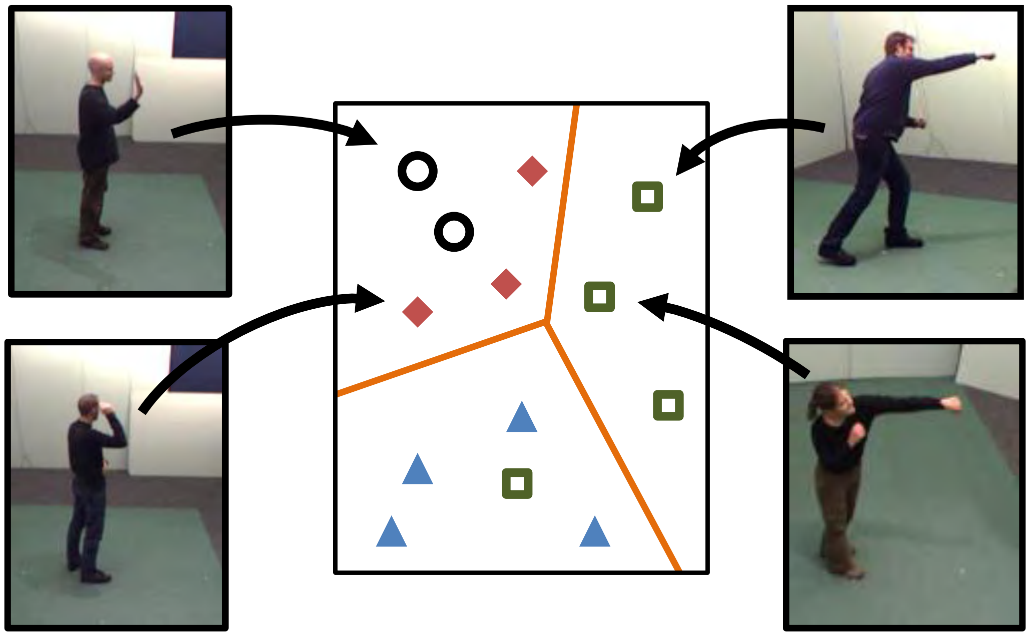


Figure 39: Keyposes are learned by clustering the feature descriptors for each frame in the training set. Some clusters (left) are heterogeneous and include poses from multiple classes. Homogeneous clusters (right) represent class-discriminative keyposes and contain primarily examples from the same class.

represented by their relative offset from the active camera.

Under this model, each camera performs single-view recognition independently. There is a large amount of work in the area of image and video feature representations for human activity (e.g., temporal templates [6], STIP [51], HOG3D [49], Motion Context [80]); our approach is not specific to a particular feature representation. We follow the common approach (e.g., [11, 53, 80, 100]) of feature quantization to learn a dictionary of keyposes. As illustrated in Figure 39, for most modern feature representations and reasonable quantization schemes, certain clusters will correspond to *class-discriminative* keyposes, namely when the component elements from the cluster are mainly from a single class. Certain neutral or ambiguous poses, however, may be shared among multiple actions. Our model seeks to learn if these keyposes are *shift-*

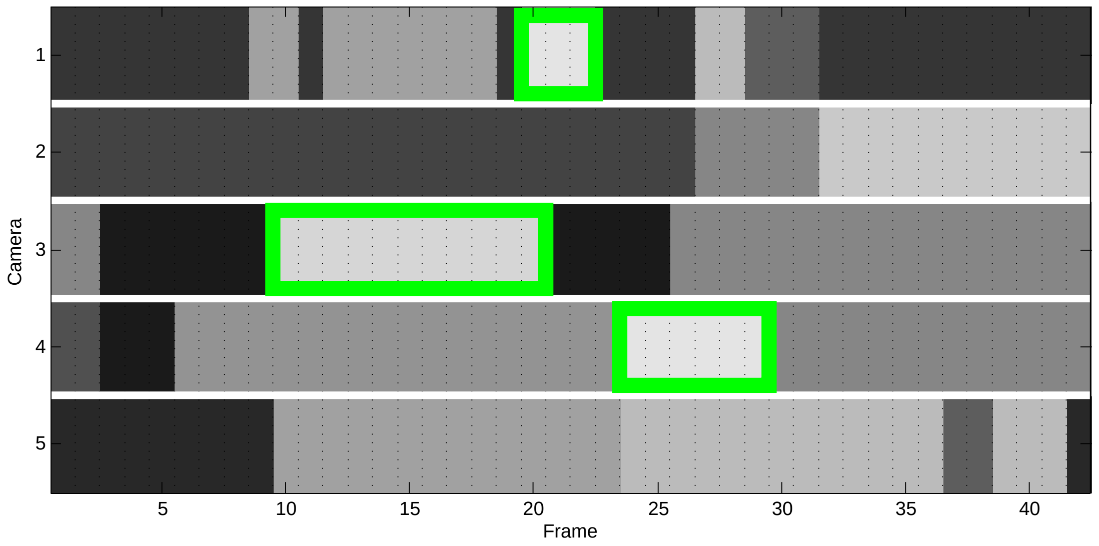


Figure 40: An action sequence captured by five cameras is shown with each frame's quantized keypose indicated by intensity. Class-discriminative keyposes are indicated with a green box.

discriminative for a *view-shift* to a new active camera in the network, with the goal of correctly predicting the action. In this section, we describe our keypose learning (§6.3.1), training (§6.3.2) and recognition (§6.3.3) steps.

6.3.1 Keypose Learning

The training input is a set of synchronized image streams, $\mathcal{I} = \{\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_{N_C}\}$ from N_C different cameras. Each stream, $\mathbf{I}_j = \{I_{j,1}, I_{j,2}, \ldots, I_{j,N_F}\}$, consists of N_F frames and could include example actions from a variety of actors and relative positions. Let $\{y_t\}$ represent the associated class labels for each time step, t.

The first step is to extract features for each frame and learn a keypose dictionary. Let $\mathcal{X} = \{x_1, \ldots, x_{N_K}\}$ represent the learned dictionary of keyposes where N_K is the user-specified dictionary size.

The class-discriminative power of each keypose is estimated by the distribution

$P(x_i \mid y)$, which is computed from the training data using Laplace smoothing:

$$P(x_i \mid y) = \frac{N_D + 1}{N_A + N_K} \tag{17}$$

where N_A is the total number of training frames with label y and N_D is the size of the subset of those frames assigned to keypose x_i. A keypose, x_i, is considered class-discriminative if the correct label, y^*, is most likely, i.e., $\underset{y'}{\operatorname{argmax}}\ P(x_i \mid y') = y^*$. Figure 40 shows a diagram representing typical input. Each frame of each camera stream is matched to a keypose, represented by the intensity in the figure. The class-discriminative keyposes, i.e., the ones most likely to lead to correct classification, are boxed in green. In this example, cameras 1, 3, and 4 each contain class-discriminative keyposes and would likely lead to a correct classification in a single-view setting. Conversely, cameras 2 and 5 see no such poses. Our goal is to learn when to view-shift from viewpoints with potential ambiguity (e.g., 2 and 5, in this example) to viewpoints with a higher likelihood of observing class-discriminative poses.

6.3.2 Learning View-Shifts

For a given target, the positions of the cameras in a network can be described by a half-sphere centered on the target. A view-shift is a change of viewpoint based on a relative offset around the viewsphere. The possible view-shifts are determined by the physical location of the cameras relative to the target. To facilitate learning, the half-sphere and, hence, view-shifts, are discretized into a fixed number of azimuth and elevation offsets. Let L_m and L_n be the discretized location of two cameras (possibly the same camera) in the network, represented in (cyclic) azimuth and elevation. The

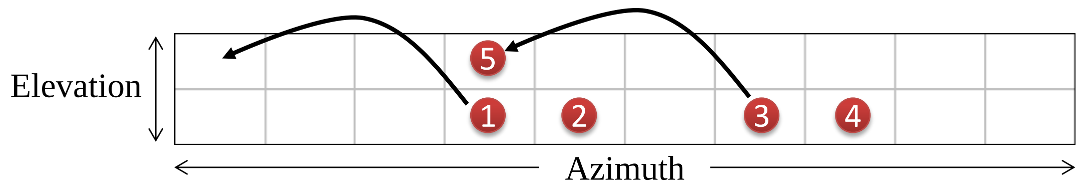


Figure 41: Each bin represents a discretized portion of the viewsphere for a particular target; in this case, there are 10 azimuth and 2 elevation bins. The red circles indicate cameras, and the arrows represent viewshifts. Both arrows indicate the same view-shift: $\langle +3, -1 \rangle$. Relative view-shifts allow our method to learn the utility of view-shifts to viewpoints unavailable during training.

view-shift between cameras is $\vec{v} = L_n - L_m$. The view-shift $\langle 0, 0 \rangle$ represents maintaining the current view, while $\langle +1, -1 \rangle$, represents shifting to a camera one offset around (azimuth) and one offset up (elevation) the view-sphere. Figure 41 shows an example configuration where the view-sphere has been divided into two elevation and ten azimuth bins with 5 cameras indicated as circles within the corresponding bin. For $N_C = 5$ cameras, there are at most 20 $(N_C \times (N_C - 1))$ possible view-shifts. The actual number of potential view-shifts is often lower as there are shared view-shifts. For example, in Figure 41, the view- shift $\langle +1, 0 \rangle$ describes both the shift from camera 2 to camera 1 and camera 4 to camera 3. Because the view-shifts are relative, and not tied to absolute camera locations, the approach is not specific to any particular camera network configuration.

View-shift learning can be framed as a Markov Decision Process (MDP) where the goal is to maximize the expected reward (correct classification) by taking actions (view-shifting) given an observed keypose. Decisions are drawn from the available view-shifts (including maintaining the current view).

Reinforcement Learning (RL) provides an efficient approach to solving MDP problems. We apply the Q-learning algorithm [86] to learn the state-action value function,

$Q(x, \vec{v})$, which encodes the value of applying view-shift, $\vec{v}$, after observing keypose x. The Q-learning algorithm iteratively approximates $Q(x, \vec{v})$ over a series of episodes. For an observed keypose in the sequence, Q can be updated:

$$Q(x, \vec{v}) \leftarrow (1 - \eta) Q(x, \vec{v}) + \eta \left[\Lambda(x', \vec{v}, y^*) + \gamma \max_{\vec{v'}} Q(x', \vec{v'}) \right] \tag{18}$$

where η is the learning rate, Λ is the immediate reward function, x' is the keypose observed after view-shift $\vec{v}$, and y^* is the correct label for the training sample. Reward allocation is based on whether the next keypose in the sequence is class-discriminative, with a small penalty for view-shifting. We allocate rewards as follows:

$$\Lambda(x, \vec{v}, y^*) = \Psi(\vec{v}) + \begin{cases} +1 & \text{if } \underset{y'}{\operatorname{argmax}}\ P(x \mid y') = y^* \\ -1 & \text{otherwise} \end{cases} \tag{19}$$

where the view-shift penalty, $\Psi(\vec{v}) = -0.1$ if $\vec{v} \neq \langle 0, 0 \rangle$. So, we provide a positive reward when the most likely class for the keypose, given by the distribution, $P(x \mid y')$, is the correct class, y^*, and a negative reward otherwise (i.e., the frame is misclassifed). Figure 42 shows examples of two frames captured from different views and the class probability distributions, $P(x \mid y)$, for each frame with the correct class highlighted in green. Based on the scoring scheme, in both cases, the pose on the right (boxed in green) represents a positive reward and that on the left (boxed in red) represents a negative reward.

The training process continues until the values for $Q(x, \vec{v})$ converge. Pseudocode for the training phase is provided in Algorithm 1.

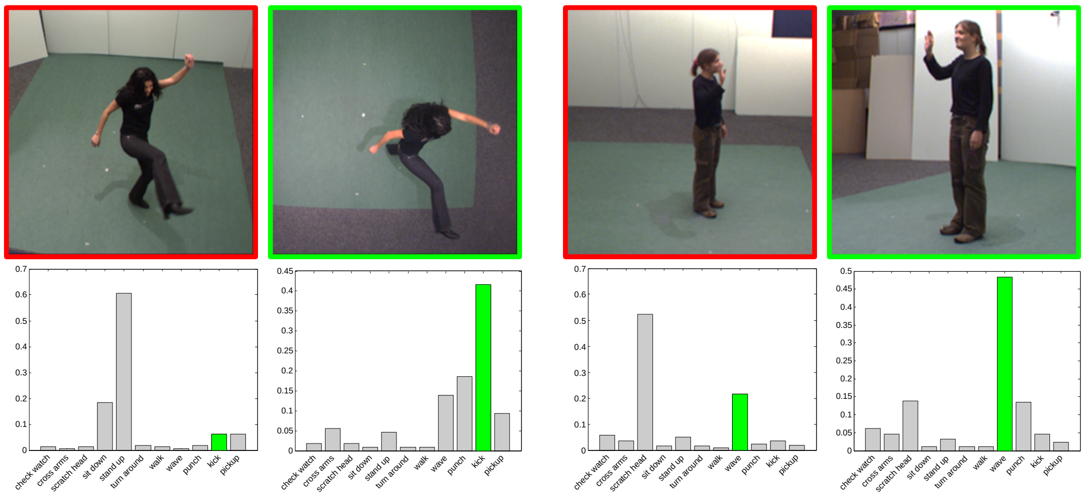


Figure 42: Examples of kicking (left) and waving (right) frames captured from two different views and the class probability distributions for each frame with the correct class highlighted in green. In our reinforcement learning method, the frames boxed in green would trigger a positive reward and the red-boxed frames would trigger a negative reward.

Algorithm 1: View-shift Value Learning

```
   Input: Synchronized image sequences, $\mathcal{I}$; frame labels, $\{y_i\}$; number of
          keyposes, $N_K$; discretized camera locations, $\{L_m\}$
   Output: State-value function, $Q(x, \vec{v})$; keypose dictionary, $\mathcal{X}$
 1 Learn keypose dictionary, $\mathcal{X} \leftarrow \{x_1, \ldots, x_{N_K}\}$
 2 For each $x_i$, calculate $P(x_i \mid y)$ (Equation 17)
 3 Initialize $Q(x, \vec{v})$ with random values drawn from $(0, 1)$
 4 while not converged do
 5     Randomly select time, $t$, and camera, $c$
 6     Get keypose, $x_i$, for frame, $I_{c,t}$
 7     foreach camera location, $L_m$ do
 8         Compute view-shift, $\vec{v} = L_c - L_m$
 9         Compute reward, $\Lambda(x_i, \vec{v}, y_t)$ (Equation 19)
10         Update $Q(x_i, \vec{v})$ (Equation 18)
11     end
12 end
```

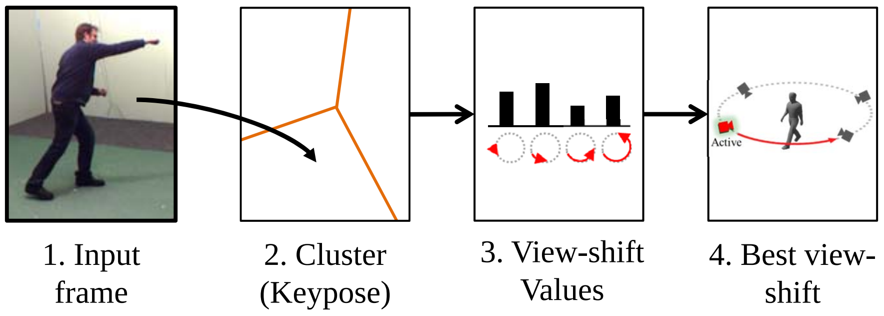


Figure 43: For recognition, a frame's assigned keypose is used to predict the value associated with each possible view-shift. The best view-shift leading to an available camera in the network is selected for viewing the next frame.

6.3.3 Recognition

For recognition, we assume that each target is currently associated with a primary camera, so the initial viewpoint might be any camera in the system. For each observed frame in a sequence, features are calculated and quantized to the nearest keypose, which is used to determine the camera that will capture the next frame (Figure 43). Similarly to [80], for a sequence of keyposes, $S = \{s_t\}$, we compute the posterior probabilities, $P(y \mid S)$, using a frame-wise approach with the Naive Bayes and uniform priors assumptions:

$$P(y \mid S) \propto \sum_t \log P(s_t \mid y). \tag{20}$$

For recognition, the final prediction for sequence, S, is the class label, $\hat{y}$, that maximizes the posterior probability:

$$\hat{y} = \underset{y'}{\operatorname{argmax}} \sum_t \log P(s_t \mid y') \tag{21}$$

It is worth noting that, in our framework, a "sequence" consists of a single keypose at a time; however, different views may be used as a result of view-shifting (Figure 44).

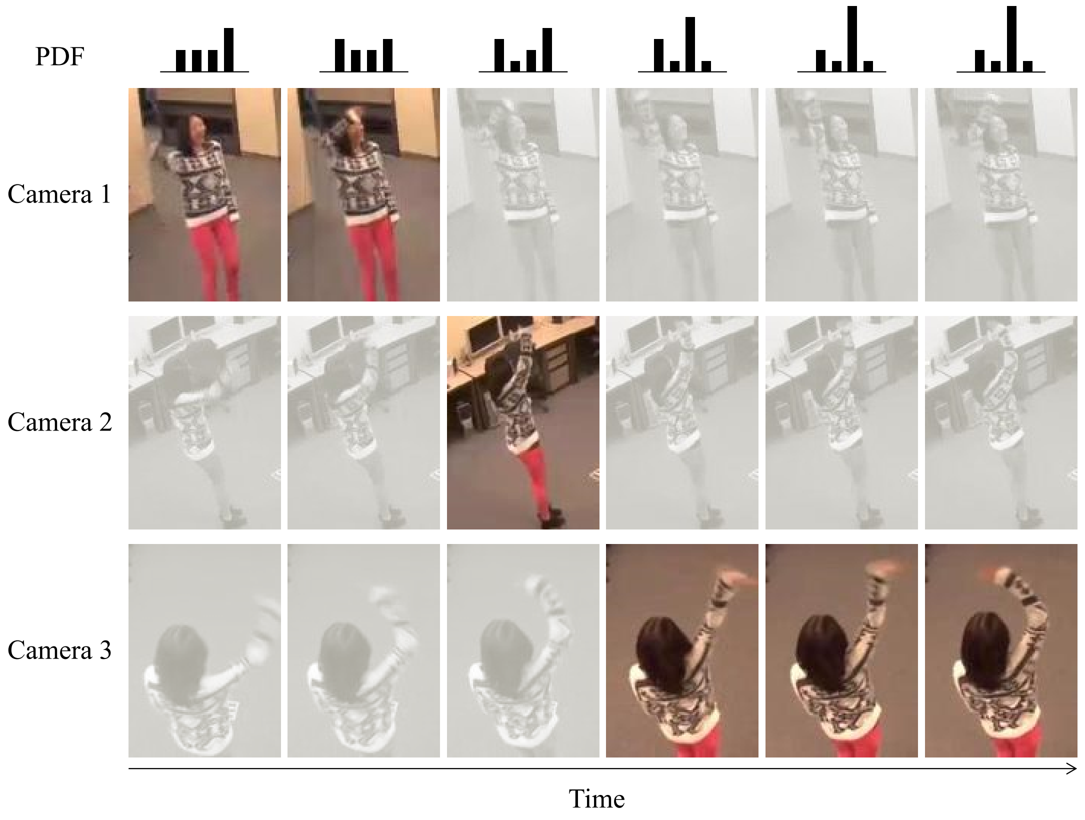


Figure 44: In this waving sequence, frames from the currently active camera are shown in color, while the other frames are not processed by the method.

So, during the recognition phase, the keypose both serves to provide evidence for the prediction and also drives dynamic selection of a potential view-shift using the learned Q-table values for the viewpoints available during recognition. The selected view-shift determines which camera will be used to view the next frame in the sequence. The details of the algorithm are given in Algorithm 2. The final prediction can be based on either time- or confidence-based thresholds.

6.4 Results

In this section, we describe several different action recognition experiments performed to evaluate our framework. All of the methods were implemented in Matlab on a standard PC. We evaluated our approach on two widely-used multi-view hu-

```
Algorithm 2: Dynamic View-shift Recognition
Input: Synchronized image sequences, $\mathcal{I}$; keyposes, $X$; state-value function,
       $Q(x, \vec{v})$; discretized camera locations, $\{L_m\}$
Output: Predicted label, $\hat{y}$
1 Let $a \equiv$ index of active camera
2 Let current time, $t = 1$
3 while not yet classified do
4     Get keypose, $x_i$, for frame, $I_{a,t}$
5     Update classification posterior (Equation 21)
6     Perform (potential) view-shift: $a \leftarrow \operatorname{argmax}_j Q(x_i, L_j - L_a)$
7     $t \leftarrow t + 1$
8 end
```

man action recognition datasets: i3DPost [33] and INRIA Xmas Motion Acquisition Sequences (IXMAS) [89]. i3DPost consists of 8 people each performing 10 actions: walk, run, jump forward, bend, hand wave, jump in place, sit-stand, run-fall, walk-sit, and run-jump-walk. IXMAS contains 10 actors performing 11 actions (check watch, cross arms, scratch head, sit down, stand up, turn around, walk, wave, punch, kick, and pickup) 3 times each. Example frames from the datasets are shown in Figure 45.

6.4.1 Implementation Details

Our approach can be applied to any frame-based feature descriptor; these experiments use the Motion Context descriptor [80], which represents the distribution of occupancy and $x-$ and $y-$ components of optic flow in a bounding box surrounding the object of interest combined with a low-dimensional projection of the feature vectors for neighboring frames.

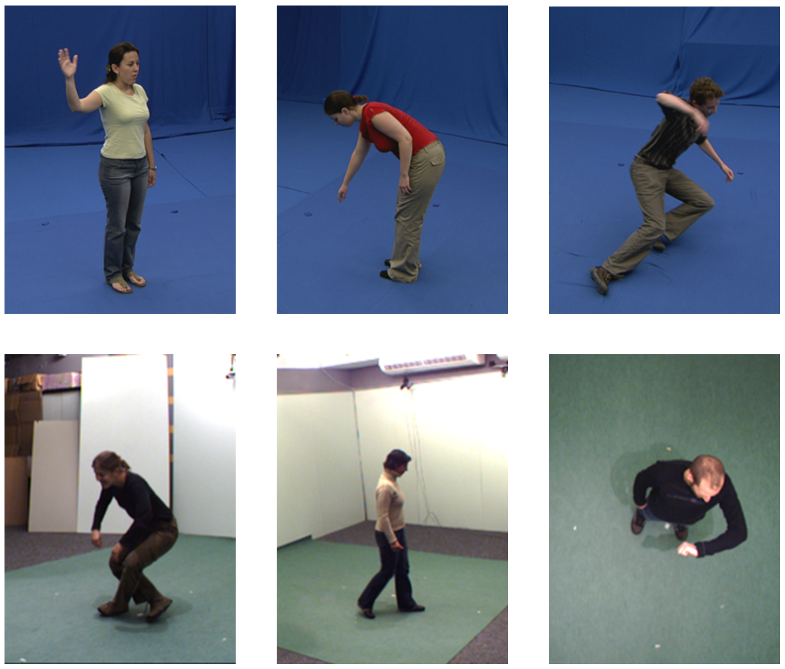


Figure 45: Example frames from the i3DPost (top) and IXMAS (bottom) data sets.

Camera Configuration

For view-shift learning, the view half-sphere is divided into discrete bins covering vertical (elevation) and horizontal (azimuth) shifts. For i3DPost, the 8 cameras are situated at the same elevation at 45-degree azimuth intervals. For this configuration, we discretize the view-sphere into 8 bins for azimuth. IXMAS consists of five cameras. Four of the cameras are at similar elevations, and one is nearly overhead. For this configuration, we discretize the view-sphere into two bins for elevation and 10 for azimuth, as shown in Figure 46. This results in 30 possible viewshifts: 10 for clockwise shifts in azimuth, and 3 for elevation $\{+1, 0, -1\}$.

Keypose Learning

In Section 6.3, we described how a keypose dictionary is used in both training and recognition. There are a number of approaches for dictionary learning in this context;

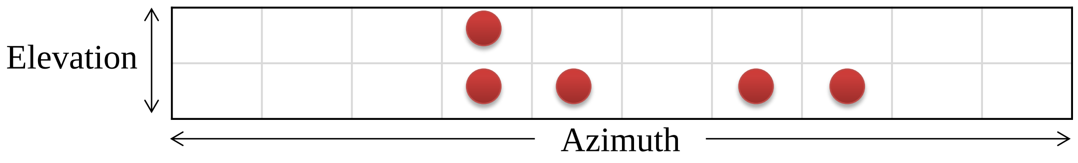
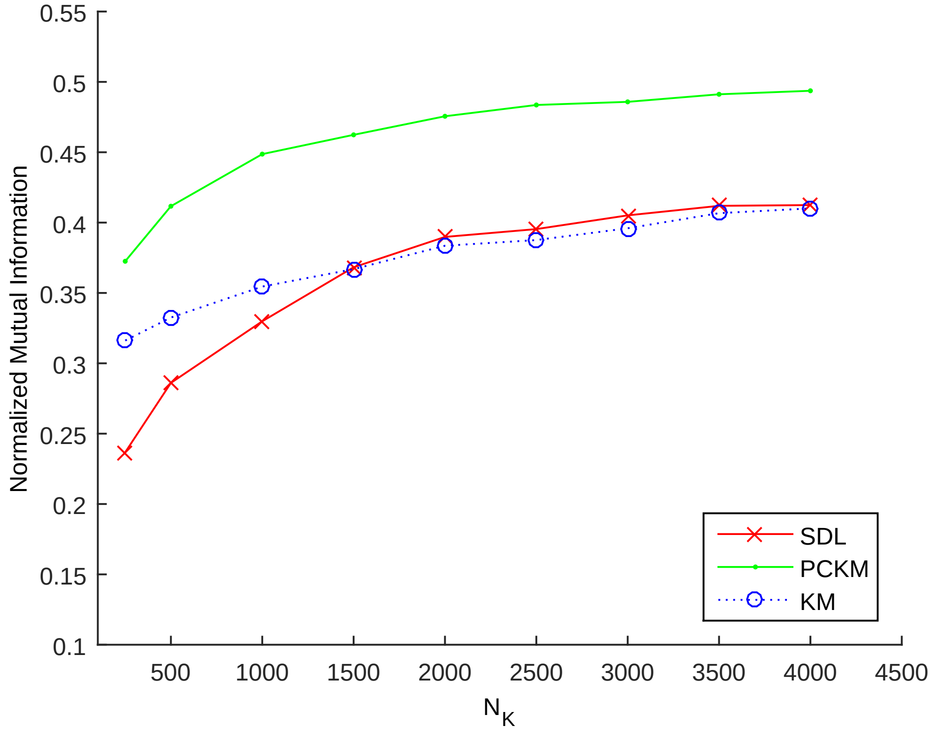


Figure 46: For the 5-camera IXMAS dataset, this diagram shows the location of each of the cameras (red circles) on the discretized view half-sphere.

we evaluated three: k-means (KM), Submodular Dictionary Learning (SDL) [43], and per-class k-means (PCKM). KM is the most basic approach to unsupervised dictionary learning (e.g., [29]). SDL is a recently-developed supervised dictionary learning method that optimizes cluster compactness, element similarity, and class discriminativeness. PCKM applies standard (unsupervised) k-means to each class separately and merges all the discovered cluster centers for the final learned dictionary. Finding key poses separately per class has been similarly considered in other recent work [11, 10].

For each method, N_K, the number of clusters (keyposes), is a free parameter. For PCKM, where clustering is performed separately for each class, N_K refers to the number of total clusters in the final, combined set for fair comparison with the other methods. In order to determine the effect of varying N_K on the clustering results, we used normalized mutual information (NMI), which is a measure of how homogeneous the clusters are with respect to the class label of the examples, balanced against the number of clusters [56]. Figure 47 shows the NMI for increasing values of N_K for the three dictionary learning methods for both datasets. For all values of N_K, PCKM results in more homogeneous clusters than KM or SDL. Based on the assumption that more homogoneous clusters lead to more class-discriminative



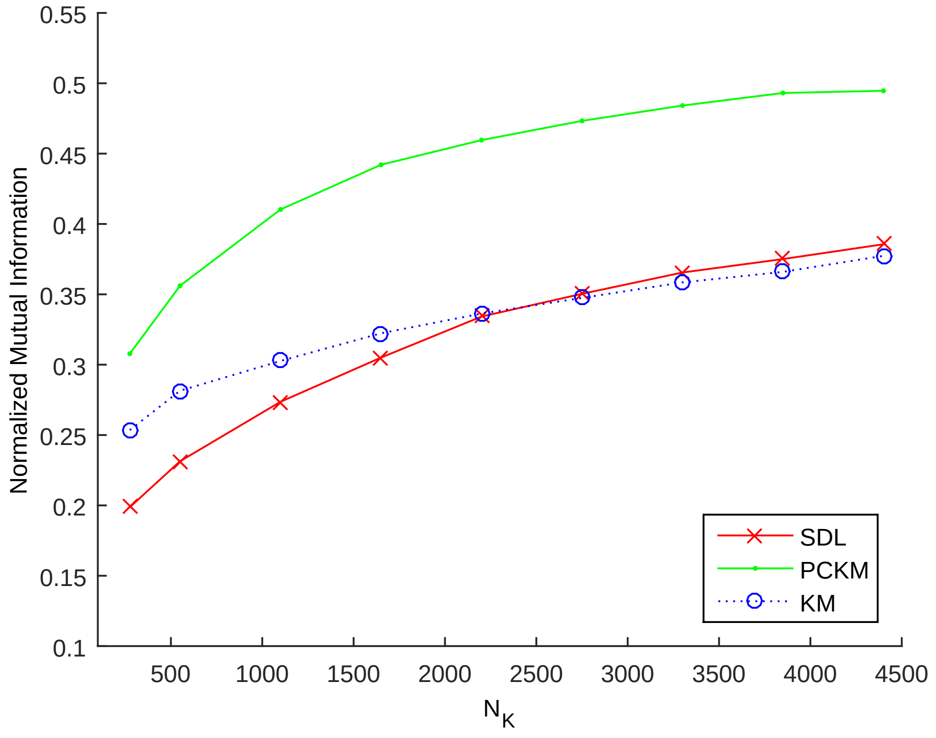


Figure 47: Normalized Mutual Information for clustering on i3DPost (top) and IX-MAS (bottom) with three different methods over a range of N_K values.

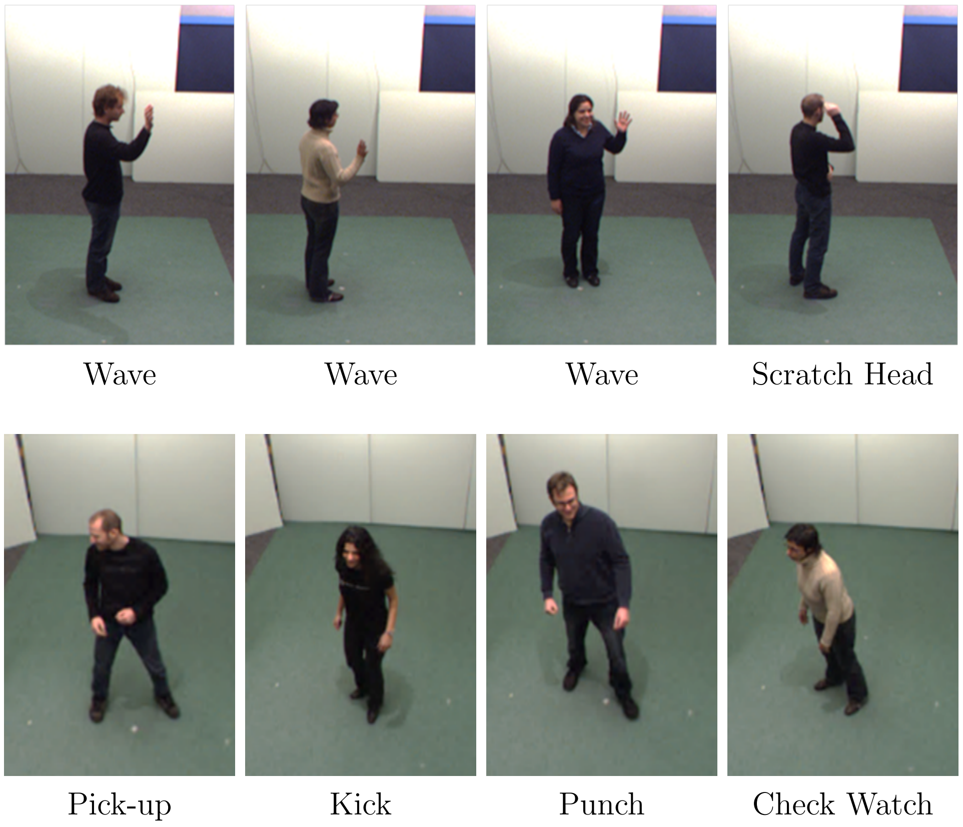


Figure 48: Each row shows frames co-clustered after unsupervised dictionary learning. (Top) The cluster is mostly homogeneous; most of the frames are examples of waving. (Bottom) The cluster is heterogeneous. From left to right, the actions are pick-up, kick, punch, and check watch.

keyposes, we select PCKM for dictionary learning for the remaining experiments. Using the elbow method, we choose $N_K = 1000$ for i3DPost and $N_K = 1650$ for IXMAS.

To build the dictionary of keyposes, the feature descriptors for every fourth frame in the training data were clustered via the PCKM algorithm. We initialized PCKM 5 times and selected the cluster assignment with minimum energy, as measured by average intra-class similarity. Figure 48 shows example frames from IXMAS from two clusters. The first row depicts a cluster that is mainly homogeneous; most of the frames are examples of waving. The second row shows the more common case of a pose shared among multiple classes.

Q-Learning Parameters

For Q-learning, the free parameters were determined empirically. The discount factor, γ, is 0.5, and the learning rate, η, is initially 1.0 with a decay rate of .997. In addition to the reward allocation scheme represented in Equation 19, we evaluated other schemes, including rewards proportional to the keypose classification margin or based on logistic regression; in general, the selected scheme performed at least as well as these alternatives. Training terminates when Q-table values have converged (i.e., successive entries are updated by less than 10^{-5}). With these values, the training phase in our experiments typically require between 5,000 - 10,000 iterations to converge, which takes about 30 seconds on a standard PC.

The view-shift penalty, Ψ, represents the cost for switching to a different camera in the network. Figure 49 shows the number of view-shifts per frame as a function of Ψ for a system trained and tested on the IXMAS data, averaged over 10 trials. (Other data showed a similar trend.) Increasing Ψ decreases the frequency of view-shifts. In a deployed system with a measurable physical or computation cost for view-shifting, this parameter can be tuned to the desired setting. For our experiments, we set the view-shift penalty, $\Psi = 0.1$, reflecting relatively little penalty to switching views.

6.4.2 Experiments

We apply our method to three different action recognition experiments, including complete sequences, early recognition, and different training and testing environment camera configurations. We refer to our view-shift action recognition method as `vs`

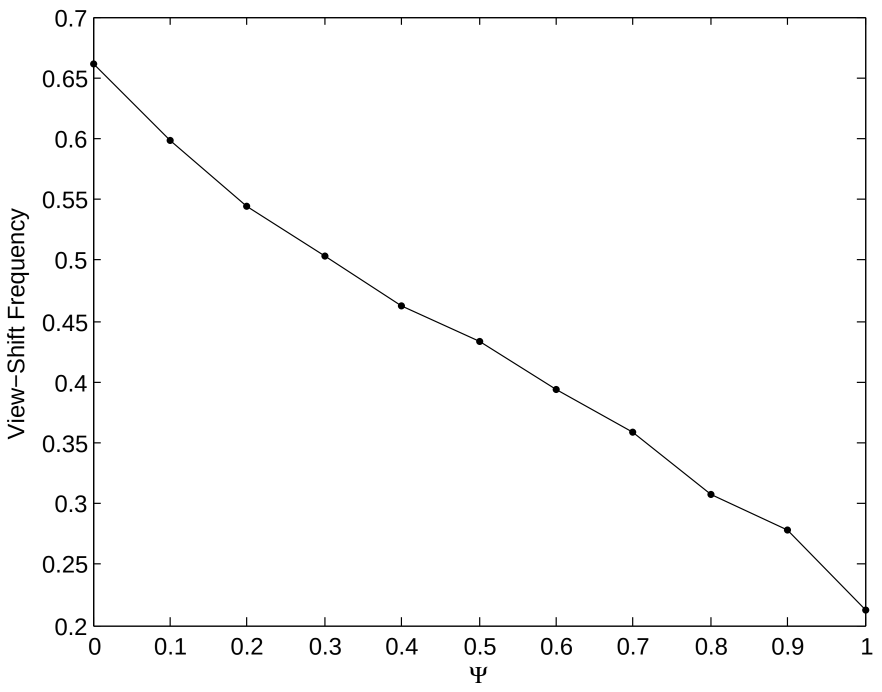


Figure 49: The frequency of view-shifts as a function of the view-shift penalty, Ψ.

and compare to alternative frame-based multi-view classification schemes that use the same features and aggregation methods:

- The single-camera (`sc`) method is our implementation of an algorithm described in [80], which uses Naive Bayes classification on single-view action sequences without view-shifting.
- The multi-camera voting (`vote`) method applies a common multi-view aggregation technique where the majority decision serves as the final classification.

For each experiment, we followed the leave-one-actor-out (LOAO) cross-validation experimental protocol, which is most commonly used in the literature for both i3DPost and IXMAS (e.g., [39, 38, 89, 103, 53]). For LOAO, all action sequences for a particular actor are used for testing, while the remaining sequences are used for training. Accuracy is averaged over all permutations.

Method	type	i3DPost	IXMAS
`vs`	single	**97.65%**	**94.24%**
`vote`	multi	96.25%	93.33%
`sc`	single	93.91%	86.06%
Iosifidis et al. [42]	4D	99.22% (8 actions)	-
Iosifidis et al. [41]	multi	95.50% (8 actions)	-
Gkalelis et al. [33]	multi	90.00% (5 actions)	-
Holte et al. [39]	multi	80.00% (10 actions)	-
Turaga et al. [81]	4D	-	98.8%
Liu et al. [53]	multi	-	93.7%
Wu et al. [93]	multi	-	88.2%
Zhu et al. [103]	multi	-	88.0%

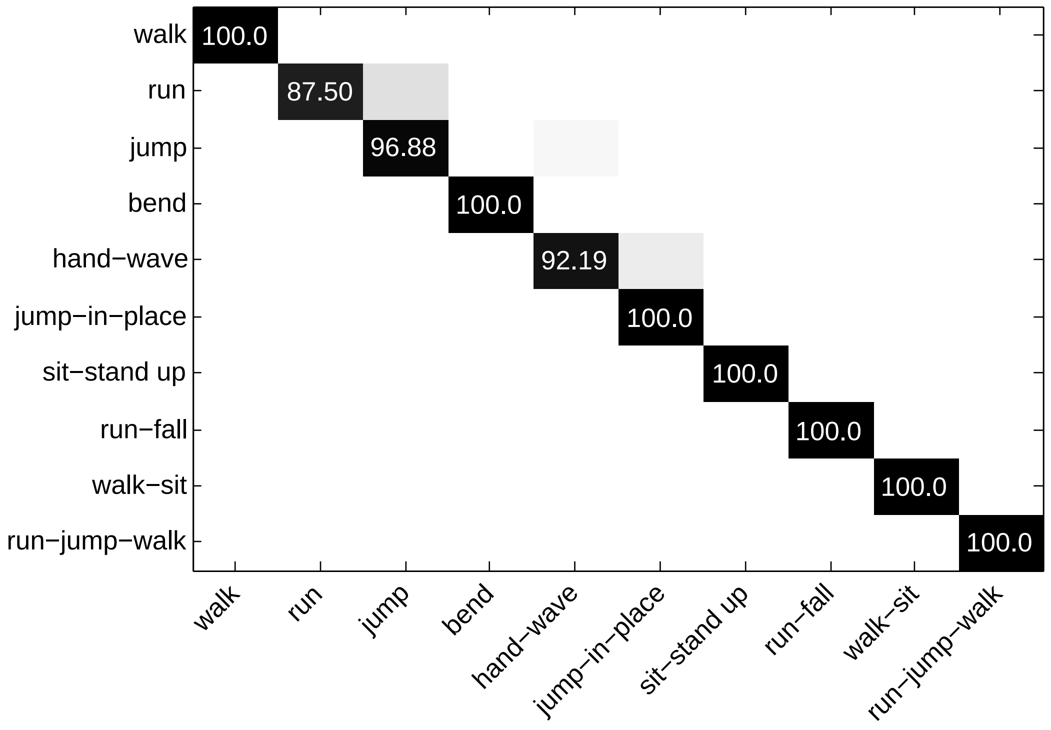
Table 4: Multi-view classification rates on the i3DPost and IXMAS data sets. The method type refers to the number of simultaneous views. (Top) Rates for our approach, `vs`, and variants. (Bottom) Representative multi-view recognition rates reported in the literature.

Trimmed Video Sequences

For classification on video sequences of prescribed length, we compared our method, `vs`, to `sc` and `vote`, as well as other recent multi-view recognition methods on both datasets. Table 4 shows the overall accuracy of each of the methods. In general, our method, `vs`, outperformed competing approaches. For each dataset, our results are better than previously reported results from related multi-view methods. To the best of our knowledge, the only methods that outperform our approach on these data sets (e.g., 99.22% for i3DPost [42], 98.78% for IXMAS [81]) are 4D approaches that either explicitly build a 3D model from the multiple views or rely on an expensive model parameter fitting step. Compared to both the multi-view and 4D approaches, ours is more efficient since, per target, processing occurs on only a single view per time step.

Figure 50 shows the confusion matrices for our method, `vs`, for this classification experiment on each data set. Each row represents the actual class and each column represents the predicted class. For many actions, accuracy is 100%. For i3DPost, the greatest confusion is between run and jump (forward), while for IXMAS the most challenging case involves confusion between waving and scratching head. These results are reasonable as the confusion in each case is between actions which share similar poses. Figure 51 shows representative examples of view-shifts for the IXMAS experiment. For each example, the opaque images correspond to active cameras and reflect the input processed by our method. The figure shows the expected value of a view-shift to each camera in the network (including not shifting) and the graphs in the rightmost column show the class probability distribution associated with the pose observed in the frame. The first two examples are representative of the general case, where the learned Q-value leads to a future class-discriminative pose. The third example shows a case where the view-shift leads to an incorrectly classified pose.

The overall accuracy on the i3DPost dataset is generally higher than on IXMAS, not only for our method, but for those reported in the literature. This may be explained by the difference in the complexity of the actions between the datasets. Compared with i3DPost, IXMAS contains far more self-occlusions and similar-looking actions with subtle variations, such as between scratching head and waving. Additionally, the position and orientation (relative to the cameras) of the IXMAS actors are not prescribed, so the relative pose of an actor is not a function of which camera is recording the action sequence. The same complexity difference is reflected in lower NMI scores for IXMAS than i3DPost (Figure 47). For the remaining experiments, we focus on the more challenging IXMAS dataset.



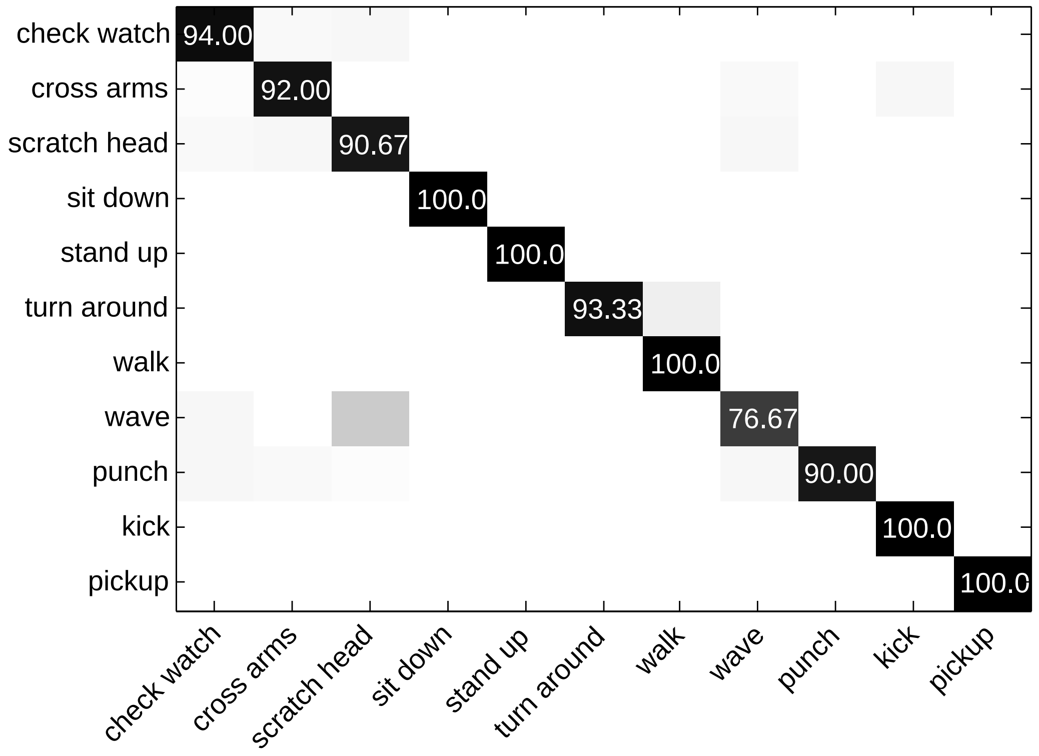


Figure 50: Confusion matrices for the proposed view-shift method on the i3DPost (top) and IXMAS (bottom) datasets.

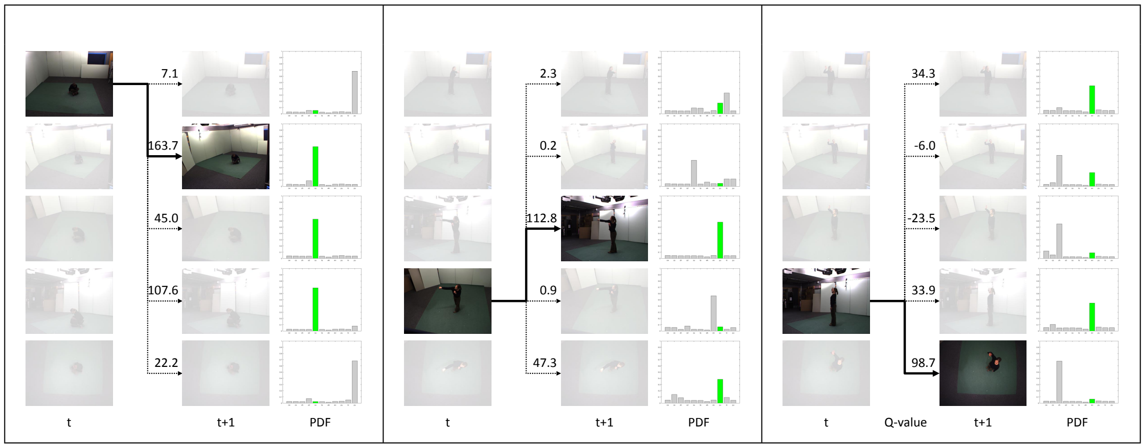


Figure 51: Example view-shifts for stand up, punch, and wave, respectively. For the active camera (opaque image), the assigned keypose is associated with Q-values for each available view-shift. The highest scoring view-shift determines the active camera for the next frame. The graphs in the rightmost column show the class probability distribution associated with the pose observed.

Early Recognition

Previous work (e.g., [71]) has included the observation that many action sequences can be identified with snippets of only a few frames. For this experiment, using the IXMAS dataset, we classified only the first portion of an action sequence for various lengths from 10% to 100%. Figure 52 shows the results for our method and the representative single- and multi-view methods. As expected, the classification accuracy of all methods increases as a greater portion of each sequence is observed. As with the experiment with the trimmed sequences, our method achieves comparable performance to the method that incorporates all views simultaneously. For sequences comprising just the first 40% of the available frames, the view -shift method accuracy is 85%, which is competitive with recent methods that observe the entire sequence and utilize all the cameras simultaneously (as reported in Table 4).

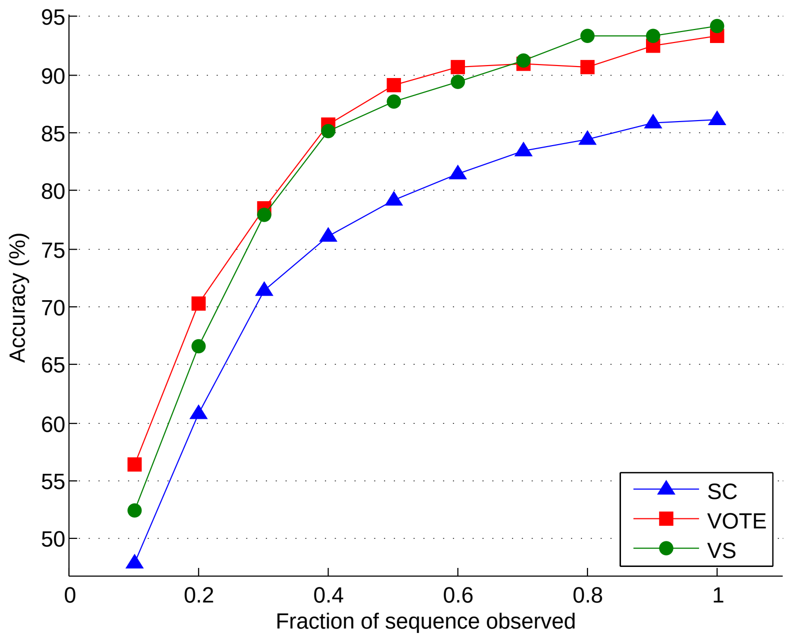


Figure 52: Accuracy of action recognition as a function of the observed fraction of each sequence using the IXMAS dataset. As the fraction increases, the accuracy of all methods improves. Accuracy of our view-shift method, `vs`, is similar to `vote`, despite using only a single camera per frame.

Camera Network Configuration

To evaluate the performance of our method when different camera configurations are used in training and testing, we perform experiments using various combinations of the network cameras. Table 5 shows results for an experiment where four of five cameras were used in training, and all five cameras were used for testing. That is, the test environment contains a camera view unseen during training. Single-camera recognition (`sc`) accuracy drops by an average of 12%, while, with the view-shift method, `vs`, recognition accuracy does not significantly drop. As before, our method achieves similar performance to the more computationally expensive approach, `vote`, which uses multiple views simultaneously. For IXMAS, the overhead viewpoint, camera 5, captures much different representations of the actions compared to the other

Training Views	`sc`	`vote`	`vs`
2 3 4 5	77.70%	91.52%	92.00%
1 3 4 5	74.79%	91.82%	92.12%
1 2 4 5	76.06%	90.61%	91.70%
1 2 3 5	75.52%	92.12%	91.94%
1 2 3 4	74.36%	90.61%	91.09%
Average	75.69%	91.34%	91.77%

Testing Views	`sc`	`vote`	`vs`
4 cameras	86.12%	91.52%	91.93%
3 cameras	86.06%	90.91%	90.66%
2 cameras	85.99%	84.30%	90.33%

Table 5: Classification accuracy on the IXMAS dataset with various camera combinations. (Top) The left column shows which cameras were used for training, while all views were used for testing. (Bottom) The left column shows how many cameras were used for testing while all views were used for training; these results are averaged over all the 4-, 3-, and 2-camera combinations, respectively.

four cameras. So, in the case when this view is unavailable for training, but present in testing, it is effectively ignored. Compared to the other combinations of training viewpoints, in this case where the overhead camera is excluded, we observe the lowest overall classification accuracy. However, the effect is limited since the overhead camera is, in general, the least useful for distinguishing among different actions.

Additionally, we perform an experiment using fewer cameras in testing compared to training, so that fewer view-shift options are available during testing. We train the method using all five cameras, and perform recognition using a subset of these views. Averaging the results over all the permutations of cameras, our method, `vs`, achieves 91.93%, 90.66%, 90.33% for four, three, and two cameras, respectively. As the number of cameras in the test environment decreases, the accuracy of our method decreases only slightly, while the accuracy of `vote` decreases sharply when only two cameras are available for testing. While these experiments may not replicate disparate camera networks, they provide evidence that relative view-shifts are not tied to specific camera configurations.

6.5 Summary

In this chapter, we presented a new approach to multi-camera action recognition, which makes use of a single active camera at a time, resulting in computational efficiency, while achieving results equal to or slightly better than methods that incorporate multiple views simultaneously. Our method is applicable to a wide variety of image features and distributed camera network architectures.

CHAPTER 7: CONCLUSIONS

This dissertation presents novel approaches for human motion analysis in multi-camera networks. We applied the concept of dynamic camera selection to each stage in the motion analysis pipeline. For detection, a variable number of cameras is selected for pedestrian verification. For tracking, a subset of the available cameras in a network is selected for active tracking. For pose estimation, exemplar detectors are applied sequentially to a variable number of views. For action recognition, a single camera is selected at each time frame both for recognition and to predict the utility of future views. In each case, our goal is to achieve accuracy similar to approaches that use all the cameras, but with greater speed by dynamically selecting a subset of the most discriminative cameras.

Common to our approaches for each area is the focus on solving 2D problems rather than 3D problems. Considering each camera in a network separately makes possible the use of a variable number of cameras since they are not all required *a priori*. Instead, multiple single-camera outputs can be fused. In addition, 2D approaches tend to be faster than 3D methods, which often require computationally complex model fitting.

Another advantage in focusing on 2D approaches is increased modularity of our framework. For the human motion analysis applications described in this dissertation, we have incorporated existing 2D methods into the view-selection framework. For example, for pedestrian verification, we took advantage of the current state-of-the-art 2D pedestrian detector, and our action recognition method is built on top of a recently introduced 2D feature descriptor. Because our camera-selection framework is modular, these 2D components can be replaced with new, faster and more accurate components as they are developed. In comparison, 3D approaches tend toward more monolithic architectures that are less amenable to such straightforward substitution into existing frameworks. Further, in the computer vision community, research on single-camera problems is far more common than on 3D problems, allowing for more frequent updates of the constituent methods.

7.1 Future Work

There are several potential extensions of this work. One direction is to apply view selection beyond human motion analysis to domains such as object recognition. For whole-scene analysis in, for example, a surveillance application, it would be useful to identify, e.g., tables, chairs, and baggage. While much of the focus in this dissertation is on motion analysis, non- or seldom-moving objects like chairs would require different features to be extracted from video, but the broader camera selection framework is very much relevant to multi-camera object detection and recognition.

A major tenant of this work is finding a balance between accuracy and speed. For truly real-time human motion analysis, where detection, tracking, pose estimation, and action recognition are all performed at rates of multiple frames per second, more work is needed. One of the largest contributors to computational latency is feature extraction. Many of the best performing feature detectors and descriptors are slow to calculate, requiring computationally expensive optimizations or dense window search across multiple scales. While dynamic camera selection is aimed at reducing the time impact of feature calculation, additional gains are possible by directly improving the speed of feature computation. Speed gains may come from new implementations that are more efficient or take advantage of hardware such as GPUs, as well as from newly developed algorithms. Future work in this area can directly improve the speed of our algorithms.

Another interesting extension is to uncalibrated networks. In this work, the scene geometry is assumed to be known, requiring camera networks to be fully calibrated as a preliminary step. Calibration allows for explicit reasoning about, for example, relative view-shifts between cameras for action recognition or coordinate transformations for switching between actively tracking cameras. However, the process of calibrating a large network of cameras can be time-consuming and requires special expertise. Avoiding this necessity by learning inter-camera relationships dynamically would make installation and configuration of camera networks for human motion analysis much simpler. Because our approach is based on integrating 2D information from individual cameras, the aggregation stage of combining single-camera inputs could potentially be extended to dynamic calibration.

REFERENCES

[1] A. Alahi, L. Jacques, Y. Boursier, and P. Vandergheynst. Sparsity driven people localization with a heterogeneous network of cameras. *Journal of Mathematical Imaging and Vision*, 41(1-2):39–58, 2011.

[2] T. Arbel and F. Ferrie. Viewpoint selection by navigation through entropy maps. In *Proc. International Conference on Computer Vision*, volume 1, pages 248–254, 1999.

[3] N. G. Arnaud Doucet, Nando de Freitas. *Sequential Monte Carlo Methods in Practice*, chapter An introduction to Sequential Monte Carlo Methods, pages 3–14. Springer-Verlag, 2001.

[4] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[5] D. Beymer and K. Konolige. Real-time tracking of multiple people using continuous detection. In *IEEE Frame Rate Workshop*, 1999.

[6] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):257–267, 2001.

[7] Q. Cai and J. Aggarwal. Tracking human motion in structured environments using a distributed-camera system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1241 –1247, Nov 1999.

[8] C. Canton-Ferrer, J. R. Casas, and M. Pardàs. Head pose detection based on fusion of multiple viewpoint information. In *Multimodal Technologies for Perception of Humans*, pages 305–310. Springer, 2007.

[9] C. Canton-Ferrer, J. R. Casas, M. Pardàs, and E. Monte. Multi-camera multi-object voxel-based monte carlo 3d tracking strategies. *EURASIP Journal on Advances in Signal Processing*, 2011(1):1–15, 2011.

[10] A. A. Chaaraoui, P. Climent-Pérez, and F. Flórez-Revuelta. Silhouette-based human action recognition using sequences of key poses. *Pattern Recognition Letters*, 34(15):1799–1807, 2013.

[11] S. Cheema, A. Eweiwi, C. Thurau, and C. Bauckhage. Action recognition by learning discriminative key poses. In *Proc. International Conference on Computer Vision Workshops*, pages 1302–1309, 2011.

[12] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995.

[13] R. Cilla, M. A. Patricio, A. Berlanga, and J. M. Molina. Fusion of single view soft k-nn classifiers for multicamera human action recognition. In *Hybrid Artificial Intelligence Systems*, pages 436–443. Springer, 2010.

[14] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[15] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–575, 2003.

[16] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *International Journal of Computer Vision*, 40:123–148, 2000.

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

[18] J. W. Davis and A. Tyagi. Minimal-latency human action recognition using reliable-inference. *Image and Vision Computing*, 24(5):455–472, 2006.

[19] D. Delannay, N. Danhier, and C. De Vleeschouwer. Detection and recognition of sports(wo)men from multiple views. In *Proc. IEEE International Conference on Distributed Smart Cameras*, pages 1 – 7, Sept 2009.

[20] C. Doersch, S. Singh, A. Gupta, J. Sivic, and A. Efros. What makes paris look like paris? *ACM Transactions on Graphics*, 31(4), 2012.

[21] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. In *Proc. European Conference on Computer Vision*, pages 645–659. Springer, 2012.

[22] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311. IEEE, 2009.

[23] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[24] W. Du and J. Piater. Multi-camera people tracking by collaborative particle filters and principal axis-based integration. In *Proc. Asian Conference on Computer Vision*, pages 365–374, 2007.

[25] E. Dunn and J. Frahm. Next best view planning for active model improvement. In *Proc. British Machine Vision Conference*, 2009.

[26] S. Dutta Roy, S. Chaudhury, and S. Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

[27] C. Ellis, S. Z. Masood, M. F. Tappen, J. J. LaViola Jr, and R. Sukthankar. Exploring the trade-off between accuracy and observational latency in action recognition. *International Journal of Computer Vision*, 101(3):420–436, 2013.

[28] R. Eshel and Y. Moses. Homography based multiple camera detection and tracking of people in a dense crowd. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[29] A. Farhadi, M. Tabrizi, I. Endres, and D. Forsyth. A latent model of discriminative aspect. In *Proc. International Conference on Computer Vision*, pages 948–955, 2009.

[30] T. Fasciano, H. Nguyen, A. Dornhaus, and M. C. Shin. Tracking multiple ants in a colony. In *IEEE Winter Conference on Applications of Computer Vision*, pages 534–540. IEEE, 2013.

[31] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[32] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:267–282, 2008.

[33] N. Gkalelis, H. Kim, A. Hilton, N. Nikolaidis, and I. Pitas. The i3dpost multiview and 3d human action/interaction database. In *Conference on Visual Media Production*, pages 159–168. IEEE, 2009.

[34] B. Han, S.-W. Joo, and L. Davis. Multi-camera tracking with adaptive resource allocation. *International Journal of Computer Vision*, 91:45–58, 2011.

[35] D. Hansen and Q. Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):478–500, 2010.

[36] D. Hearn, P. Baker, and W. Carithers. *Computer Graphics With OpenGL*. Prentice Hall, 2011.

[37] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.

[38] M. B. Holte, B. Chakraborty, J. Gonzalez, and T. B. Moeslund. A local 3-d motion descriptor for multi-view human action recognition from 4-d spatio-temporal interest points. *IEEE Journal of Selected Topics in Signal Processing*, 6(5):553–565, 2012.

[39] M. B. Holte, T. B. Moeslund, N. Nikolaidis, and I. Pitas. 3d human action recognition for multi-view camera systems. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 342–349. IEEE, 2011.

[40] C. Huang, B. Wu, and R. Nevatia. Robust object tracking by hierarchical association of detection responses. In *Proc. European Conference on Computer Vision*, pages 788–801. Springer, 2008.

[41] A. Iosifidis, A. Tefas, and I. Pitas. Multi-view human action recognition under occlusion based on fuzzy distances and neural networks. In *Proc. of the European Signal Processing Conference*, pages 1129–1133. IEEE, 2012.

[42] A. Iosifidis, A. Tefas, and I. Pitas. View-independent human action recognition based on multi-view action images and discriminant learning. In *IEEE Workshop on Image, Video, and Multidimensional Signal Processing*, pages 1–4, June 2013.

[43] Z. Jiang, G. Zhang, and L. S. Davis. Submodular dictionary learning for sparse coding. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3418–3425. IEEE, 2012.

[44] A. K. Rajagopal, R. Subramanian, R. Vieriu, E. Ricci, O. Lanz, K. Ramakrishnan, and N. Sebe. An adaptation framework for head-pose classification in dynamic multi-view scenarios. In *Proc. Asian Conference on Computer Vision*, 2013.

[45] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Trans. on Comm. Tech.*, 15(1):52–60, 1967.

[46] D. Karuppiah, R. Grupen, Z. Zhu, and A. Hanson. Automatic resource allocation in a distributed camera network. *Machine Vision and Applications*, 21:517–528, 2010.

[47] S. Khan and M. Shah. A multiview approach to tracking people in crowded scenes using a planar homography constraint. In *Proc. European Conference on Computer Vision*, volume 3954 of *Lecture Notes in Computer Science*, pages 133–146. Springer Berlin / Heidelberg, 2006.

[48] S. Khan and M. Shah. Tracking multiple occluding people by localizing on multiple scene planes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 505–519, 2008.

[49] A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *Proc. British Machine Vision Conference*, pages 995–1004, sep 2008.

[50] H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 157–164, 2003.

[51] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2-3):107–123, 2005.

[52] M. Liem and D. M. Gavrila. Multi-person tracking with overlapping cameras in complex, dynamic environments. In *Proc. British Machine Vision Conference*, pages 199–218. British Machine Vision Association, 2009.

[53] L. Liu, L. Shao, and P. Rockett. Boosted key-frame selection and correlated pyramidal motion-feature representation for human action recognition. *Pattern Recognition*, 46(7):1810 – 1818, 2013.

[54] T. Määttä, A. Härmä, and H. Aghajan. On efficient use of multi-view data for activity recognition. In *Proc. IEEE International Conference on Distributed Smart Cameras*, ICDSC '10, pages 158–165, New York, NY, USA, 2010. ACM.

[55] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *Proc. International Conference on Computer Vision*, pages 89–96. IEEE, 2011.

[56] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[57] A. Mittal and L. S. Davis. M2Tracker: a multi-view approach to segmenting and tracking people in a cluttered scene. *International Journal of Computer Vision*, 51:189–203, 2003.

[58] M. Mubashir, L. Shao, and L. Seed. A survey on fall detection: Principles and approaches. *Neurocomputing*, 100:144–152, 2013.

[59] R. Munoz-Salinas, E. Yeguas-Bolivar, A. Saffiotti, and R. Medina-Carnicer. Multi-camera head pose estimation. *Machine Vision and Applications*, 23(3):479–490, 2012.

[60] E. Murphy-Chutorian and M. Trivedi. Head pose estimation in computer vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):607–626, 2009.

[61] C. Nakajima, M. Pontil, B. Heisele, and T. Poggio. Full-body person recognition system. *Pattern recognition*, 36(9):1997–2006, 2003.

[62] K. Nummiaro, E. Koller-Meier, T. Svoboda, D. Roth, and L. V. Gool. Color-based object tracking in multi-camera environments. *Lecture Notes in Computer Science*, 2781:591–599, 2003.

[63] J. Orozco, S. Gong, and T. Xiang. Head pose classification in crowded scenes. In *Proc. British Machine Vision Conference*, volume 1, page 3, 2009.

[64] K. Parrigan and R. Souvenir. Aggregating low-level features for human action recognition. In *Advances in Visual Computing*, Lecture Notes in Computer Science, pages 143–152, 2010.

[65] P. Peng, Y. Tian, Y. Wang, and T. Huang. Multi-camera pedestrian detection with multi-view bayesian network model. In *Proc. British Machine Vision Conference*, pages 1–12, 2012.

[66] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[67] R. Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976 – 990, 2010.

[68] H. Possegger, S. Sternig, T. Mauthner, P. M. Roth, and H. Bischof. Robust real-time tracking of multiple objects by volumetric mass densities. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2395–2402. IEEE, 2013.

[69] A. K. Rajagopal, R. Subramanian, E. Ricci, R. L. Vieriu, O. Lanz, K. Ramakrishnan, and N. Sebe. Exploring transfer learning approaches for head pose classification from multi-view surveillance images. *International Journal of Computer Vision*, 109(1-2):146–167, 2014.

[70] D. Rudoy and L. Zelnik-Manor. Viewpoint selection for human actions. *International Journal of Computer Vision*, 97(3):243–254, 2012.

[71] K. Schindler and L. Van Gool. Action snippets: How many frames does human action recognition require? In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[72] C. Shen, C. Zhang, and S. Fels. A multi-camera surveillance system that estimates quality-of-view measurement. In *Proc. International Conference on Image Processing*, volume 3, pages III–193. IEEE, 2007.

[73] R. Souvenir and J. Babbs. Learning the viewpoint manifold for action recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, 2008.

[74] S. Spurlock and R. Souvenir. Pedestrian verification for multi-camera detection. In *Proc. Asian Conference on Computer Vision*, 2014.

[75] G. Srivastava, H. Iwaki, J. Park, and A. C. Kak. Distributed and lightweight multi-camera human activity classification. In *Proc. IEEE International Conference on Distributed Smart Cameras*, pages 1–8. IEEE, 2009.

[76] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, 1999.

[77] R. Sznitman, C. Becker, F. Fleuret, and P. Fua. Fast object detection with entropy-driven evaluation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3277. Ieee, 2013.

[78] M. Taj and A. Cavallaro. Distributed and decentralized multi-camera tracking: a survey. *IEEE Signal Processing Magazine*, 28(3), 2011.

[79] D. Tosato, M. Farenzena, M. Spera, V. Murino, and M. Cristani. Multi-class classification on riemannian manifolds for video surveillance. In *Proc. European Conference on Computer Vision*, pages 378–391. Springer, 2010.

[80] D. Tran and A. Sorokin. Human activity recognition with metric learning. In *Proc. European Conference on Computer Vision*, pages 548–561. Springer-Verlag, 2008.

[81] P. Turaga, A. Veeraraghavan, and R. Chellappa. Statistical analysis on stiefel and grassmann manifolds with applications in computer vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[82] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–511. IEEE, 2001.

[83] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[84] M. Voit, K. Nickel, and R. Stiefelhagen. Neural network-based head pose estimation and multi-view fusion. In *Multimodal technologies for perception of humans*, pages 291–298. Springer, 2007.

[85] X. Wang. Intelligent multi-camera video surveillance: A review. *Pattern Recognition Letters*, 2012.

[86] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[87] D. Weinland, E. Boyer, and R. Ronfard. Action recognition from arbitrary views using 3d exemplars. In *Proc. International Conference on Computer Vision*, pages 1–7, 2007.

[88] D. Weinland, M. Özuysal, and P. Fua. Making action recognition robust to occlusions and viewpoint changes. In *Proc. European Conference on Computer Vision*, pages 635–648. Springer, 2010.

[89] D. Weinland, R. Ronfard, and E. Boyer. Free viewpoint action recognition using motion history volumes. *Computer Vision and Image Understanding*, 104(2):249–257, 2006.

[90] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.

[91] M. J. Wilber, E. Rudd, B. Heflin, Y.-M. Lui, and T. E. Boult. Exemplar codes for facial attributes and tattoo recognition. In *IEEE Winter Conference on Applications of Computer Vision*, pages 205–212. IEEE, 2014.

[92] C. Wu, A. H. Khalili, and H. Aghajan. Multiview activity recognition in smart homes with spatio-temporal features. In *Proc. IEEE International Conference on Distributed Smart Cameras*, pages 142–149. ACM, 2010.

[93] X. Wu, D. Xu, L. Duan, and J. Luo. Action recognition using context and appearance distribution features. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 489–496, 2011.

[94] A. A. X. Zabulis, T. Sarmis. 3d head pose estimation from multiple distant views. *Proc. British Machine Vision Conference*, 2009.

[95] P. Yan, S. M. Khan, and M. Shah. Learning 4d action feature models for arbitrary view action recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2008.

[96] Y. Yan, E. Ricci, R. Subramanian, O. Lanz, and N. Sebe. No matter where you are: Flexible graph-guided multi-task learning for multi-view head pose classification under target motion. In *Proc. International Conference on Computer Vision*, pages 1177–1184. IEEE, 2013.

[97] Y. Yan, R. Subramanian, E. Ricci, O. Lanz, and N. Sebe. Evaluating multi-task learning for multi-view head-pose classification in interactive environments. In *Proc. International Conference on Pattern Recognition*, pages 4182–4187. IEEE, 2014.

[98] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38, December 2006.

[99] Z. Zhang, Y. Hu, M. Liu, and T. Huang. Head pose estimation in seminar room using multi view face detectors. In *Multimodal Technologies for Perception of Humans*, pages 299–304. Springer, 2007.

[100] Z. Zhao and A. M. Elgammal. Information theoretic key frame selection for action recognition. In *Proc. British Machine Vision Conference*, pages 1–10, 2008.

[101] Q. Zhou and J. Aggarwal. Object tracking in an outdoor environment using fusion of features and cameras. *Image and Vision Computing*, 24(11):1244 – 1255, 2006.

[102] F. Zhu, L. Shao, and M. Lin. Multi-view action recognition using local similarity random forests and sensor fusion. *Pattern Recognition Letters*, 2012.

[103] X. Zhu and D. Ramanan. Face detection, pose estimation and landmark localization in the wild. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012.

[104] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.