Probabilistic Shape Parsing and Action Recognition Through Binary Spatio-Temporal Feature Description

by

Christopher James Whiten

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the MCS degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

In this thesis, contributions are presented in the areas of shape parsing for view-based object recognition and spatio-temporal feature description for action recognition. A probabilistic model for parsing shapes into several distinguishable parts for accurate shape recognition is presented. This approach is based on robust geometric features that permit high recognition accuracy.

As the second contribution in this thesis, a binary spatio-temporal feature descriptor is presented. Recent work shows that binary spatial feature descriptors are effective for increasing the efficiency of object recognition, while retaining comparable performance to state of the art descriptors. An extension of these approaches to action recognition is presented, facilitating huge gains in efficiency due to the computational advantage of computing a bag-of-words representation with the Hamming distance. A scene's motion and appearance is encoded with a short binary string. Exploiting the binary makeup of this descriptor greatly increases the efficiency while retaining competitive recognition performance.

Acknowledgements

I thank Robert Laganière, my supervisor, for his continued support and guidance throughout these past 16 months. Working alongside Robert has been both enlightening and rewarding, allowing me to work on exciting research projects. I would also like to thank Guillaume-Alexandre Bilodeau for his help and valuable discussions throughout my thesis, specifically pertaining to work on action recognition. I am grateful to Diego Macrini for his assistance throughout work pertaining to shape parsing.

Finally, I would like to thank my family, friends, and colleagues who have had the pleasure of witnessing my ups and downs throughout the course of this thesis. Thanks for your patience and understanding.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, two separate contributions are presented. While these contributions have very little overlap, they both fall under the domain of *recognition*, which is the underlying theme throughout the course of this thesis.

First, a novel approach for decomposing shapes into parts is introduced. This process is based on robust geometric features (the medial axis, concave corners) of a contour to define "shape cuts", which encode the latent boundaries between the semantic parts of a query shape. Although modelling shapes is an inherently uncertain process, the proposed approach is lenient, in that the desired parse of a shape only needs to be within its k most "probable" parses.

The second contribution of this thesis is a binary spatio-temporal feature descriptor. Current state of the art approaches to action recognition rely on large hierarchies of detectors or the extraction of costly, floating-point value descriptors. These descriptors are slow to compute and slow to compare. Stemming from the infeasibility of applying these techniques to massive data sets, we have devised an entirely binary descriptor that encodes both spatial and temporal data. These descriptors are fast to compute (with simple pixel-wise intensity comparisons) and fast to match (with the Hamming distance). With this newfound computational efficiency, we aim to accurately and efficiently detect and recognize events in an assortment of scenarios, from surveillance footage to Hollywood movie data.

1.1 Motivation

1.1.1 Shape Parsing

Whether or not shapes should be represented and matched as a single holistic shape or a set of shape parts remains an open problem. On one hand, it is difficult to formulate mathematical reasoning for decomposing a shape into parts. Standard shape-decomposition algorithms are sensitive to a variety of phenomena, such as scale, rotation, shape perturbations, and viewpoint changes. On the other hand, holistic shape representations may not contain enough information to robustly perform object recognition. Holistic approaches suffer from many of the same phenomena as part-based approaches, but also suffer from significant intra-class variations. Beyond these pitfalls of holistic approaches, matching algorithms for whole contour representations tend to be much simpler and more efficient than part-based algorithms, since there is no need to enforce spatial constraints. The decision on which shape representation paradigm to take is largely application-dependent.

A shape parsing framework that captures the benefits of both part-based and whole contour shape representations is presented in this contribution. In the context of this work, a part-based methodology is taken, where multiple parses of a query shape are computed and the whole contour is a possible “parse” of the shape. The result of this is that object recognition algorithms consider both a part-based decomposition of a shape, as well as the entire contour, when computing the similarity between a query shape and shapes with known labels.

A further goal of the work is to present a framework that permits accurate shape recognition, while remaining lenient to the initial parameterizations selected. That is, we aim to provide a shape parsing framework that allows for accurate shape recognition without the necessity to spend a great deal of time tuning parameters. The robustness to parameter tuning is achieved by keeping multiple parses of each shape in a precomputed dataset, so parameterizations in which a poor parsing of the shape is most likely are given a “second chance”, where better parameterizations are possible in later parses.

This work was done in collaboration with Diego Macrini, as an extension of his PhD work. As part of this previous work, a great deal of implementation was provided, including the corner detection, shape context implementation, and graph matching implementation. This thesis’ contribution to this work consists of further implementation of the presented ideas, as well as the conception of optimizations (as an example, the shape part training presented), running of experiments, and writing in the associated paper. While the elementary concept was proposed by Diego, the approach evolved through our collaborative discussions into the work presented in this thesis.

1.1.2 Spatio-Temporal Feature Descriptor for Action Recognition

Recognizing actions in video sequences has garnered a great deal of interest in recent years. With recent advances in natural user interface design, algorithms for gesture recognition are in very high demand. Furthermore, robust and efficient classification of surveillance video footage is desirable for efficient traversal of large volumes of data. Although these are among the top applications for action recognition, recent works have strayed from the path of applicability, favouring algorithms with high recognition accuracy but large and sometimes unrealistic running times.

Our main contribution is the presentation of a novel spatio-temporal feature descriptor that achieves significant running time improvements over state-of-the-art methods, while remaining highly accurate. We extend recent work [1] in feature description for object recognition, where spatial neighbourhoods are described by compact binary strings. We introduce an implicit temporal component into the descriptor, while removing components that are of less value for recognizing actions. Furthermore, we adapt a recent approach [21] to action recognition for increased efficiency, where the geometric structure of motion is encoded by dense sampling of self-similarities. Our extension of this work improves efficiency by avoiding dense sampling through keypoint detection. Throughout the construction of this descriptor, stress is placed on ensuring the entire descriptor remains binary, gifting us with highly optimized processing and feature matching. This yields significant computational gains in approaches such as standard bag-of-words models, where thousands of matches must often be made at each frame.

1.2 Problem Description

1.2.1 Shape Parsing for View-Based Object Recognition

The goal of the proposed shape parsing algorithm is to increase recognition accuracy. Here, the shape recognition problem is briefly described, within the context of this work.

We aim to devise an algorithm such that given a query shape $\mathcal{Q}$ without a class label, the true label of $\mathcal{Q}$ is returned. This is achieved by means of comparisons against a precomputed training database, $\mathcal{D}$. Within $\mathcal{D}$, several $(shape, label)$ pairs exist, where a series of precomputations has been performed on each shape to increase the efficiency and accuracy of the recognition process.

For the query shape $\mathcal{Q}$, the similarity between each shape $\mathcal{S}$ and $\mathcal{Q}$ is computed with some predefined distance metric. One commonly used metric between shapes, on which a variant is taken in this work, is the shape context distance [4]. This method of classification is often referred to as naïve nearest-neighbour classification, as the pairwise distance between $\mathcal{Q}$ and each $\mathcal{S} \in \mathcal{D}$ is computed. The label corresponding to the $\mathcal{S}$ which returns the highest similarity (or shortest distance) is deemed to be the true label of $\mathcal{Q}$.

1.2.2 Spatio-Temporal Feature Descriptor for Action Recognition

With the construction of our spatio-temporal feature descriptor, the goal is to provide a descriptor that is conducive to the process of accurate action recognition, while remaining as computationally efficient as possible. These priorities were handled by the exploitation of a binary feature descriptor which has been show to perform well in the object recognition domain, placed within a tried-and-true action recognition pipeline.

Given a training dataset $\mathcal{T} = (\mathcal{X}, \mathcal{Y})$, where $\mathcal{X}$ is the set of video sequences and $\mathcal{Y}$ is the set of labels, one label per video sequence, we aim to build a model on $\mathcal{T}$ that accurately captures the mapping $x \rightarrow y$, the mapping from an arbitrary video sequence x to its true label y. The constructed model should maximize the accuracy of the $x \rightarrow y$ mapping for future, unforeseen query videos.

The described model will be designed within the following common framework for action recognition. At each frame within a query video sequence x, a set of spatio-temporal descriptors, presented in this thesis, will be extracted. The entire set of features extracted from the video sequence will be provided as input to a bag-of-words model [11], which outputs a single feature for classification. This single feature is then provided as input to a support vector machine [10], which returns a single classification decision y. If y corresponds to the true label of x, then the recognition process is deemed a success.

1.3 Thesis Organization

This thesis is conceptually separated into two portions, one for each contribution of the thesis. The first chapters correspond to work on the contribution to shape parsing for view-based object recognition, while the latter chapters correspond to work on descriptor construction for action recognition in videos.

In Chapter 2, selected work from the literature on shape decomposition and shape matching is presented, as it relates to this thesis's contribution on shape parsing. An approach to whole-contour shape matching is presented, as it is used to match individual shape parts in this contribution. Related shape decomposition approaches are also presented.

Chapter 3 introduces and explains the presented work on shape decomposition and shape matching. After explaining the approach, experimental results are presented.

The chapters relevant to action recognition then begin. Chapter 4 provides an overview of the relevant literature, describing the different paradigms considered for action recognition. Previous work in this area is divided into three paradigms: high-level representation, trajectory analysis, and local spatio-temporal features. The work in this thesis falls into the latter category.

In Chapter 5, an in-depth explanation of the spatio-temporal descriptor construction is described. The selection and construction of both the appearance and motion components of the descriptor is described, as is the approach to represent and classify videos with the presented descriptor. Furthermore, an extension from action recognition to event detection is presented.

Chapter 6 describes the experiments performed against several benchmark datasets. A descriptor's performance is evaluated against a wide variety of data, from artificial scenes to Hollywood movie data to Youtube videos to airport surveillance data. Our experiments show that this descriptor competes with similar state of the art methods in the domain of action recognition by local features. We also showcase the significant computational advantage gained through the use of this descriptor, in comparison with other action recognition algorithms. We have found that our algorithm performs over 200 times faster than the current state of the art.

Finally, we conclude in Chapter 7. Along with the conclusion, possible avenues of further research are summarized.

1.4 Publications

There are three related publications for the work presented in this thesis. Two of these have already been presented at conferences, while one is to be presented at the Canadian Conference on Computer and Robot Vision (CRV) in 2013. Two of the papers pertain to the content from this thesis on spatio-temporal feature descriptors, while one of the papers pertains to the content on shape parsing for object recognition.

At TRECVID 2012, this work on spatio-temporal feature descriptors was presented as an oral presentation, demonstrating its capabilities for surveillance event detection. Further work on the descriptor was performed, and appaers in the paper for CRV 2013. This paper focuses on the actual construction of the descriptor, along with its performance against several benchmark datasets for action recognition.

A conference paper has been presented at the *21st International Conference on Pattern Recognition* (ICPR) on work related to shape parsing for object recognition. This work has been completed in conjunction with Diego Macrini, whose PhD work was on shape parsing. Diego's contribution to this work was substantial, going so far as to the initial proposal of shape cuts and prototype implementation. My contributions varied from extensions and elaborations on the initial algorithm, to implementation, writing and experiments.

- Diego Macrini, Chris Whiten, Robert Laganière, Michael Greenspan **Probabilistic Shape Parsing for View-Based Object Recognition**. In *21st International Conference on Pattern Recognition (ICPR 2012).* [32]

- Chris Whiten, Robert Laganière, Ehsan Fazl-Ersi, Feng Shi, Guillaume-Alexandre Bilodeau, Dmitry Gorodnichy, Jean-Philippe Bergeron, Ehren Choy **VIVA-uOttawa / CBSA at TRECVID 2012: Interactive Surveillance Event Detection**. In *Proceedings of TRECVID 2012*. [50]

- Chris Whiten, Robert Laganière, Guillaume-Alexandre Bilodeau. **Efficient Action Recognition with MoFREAK**. In *Tenth Conference on Computer and Robot Vision (CRV 2013).* [49]

Chapter 2

Shape Parsing - Related Work

2.1 Related Work

A fundamental problem in the representation and matching of shapes is to determine whether or not a shape should be divided into parts, and if so, what those parts should be. Part-based approaches are attractive because they provide a mechanism for comparing both shape contours and part structures, but require expensive matching algorithms to account for similar shapes having markedly different decompositions. Alternatively, whole contour approaches require simpler matching algorithms, but are only effective when objects have little articulation and within-class variation.

We briefly review shape matching approaches considering both part-based and whole contour shape representation. Our approach leverages the use of shape contexts (a whole contour representation) as an individual part descriptor, while using a part-based approach to encode the shape's spatial hierarchy. The reviewed part-based approaches all provide a unique part decomposition for each shape, impose hard constraints on which regions of a shape can become parts, and depend on fixed decomposition rules. In addition, having only one representation per shape demands a matching approach that can deal with incorrect choices made during the shape decomposition process, which can lead to large representational differences among similar shapes. Our approach aims to solve these problems.

2.1.1 Whole Contour-Based Approaches

Shape Context (Belongie *et al.*)

Belongie *et al.* [3] introduce shape context, a contour descriptor designed with the intention of measuring the similarity between two compared shapes. A set of n points is sampled along the contour, and at each sampled point, p_i, the shape context encodes the distribution of the locations of the remaining sampled points along the contour, relative to the point that shape context is anchored at. This is achieved by encoding the vectors anchored at p_i and connecting to each of the remaining $n - 1$ sampled points. A log-polar coordinate system is used to compute a coarse histogram of these vectors, encoding the approximate distribution of the remainder of the contour, relative to p_i. Shape context is a popular descriptor for shape matching, due to its invariance to translation, scale, rotation, and common deformations. Within the context of this work, shape context is used as a descriptor to match parts of the shape (including the scenario where the entire contour is considered a part).

Using the Inner-Distance for Classification of Articulated Shapes (Ling and Jacobs)

Ling and Jacobs [28] take an interesting approach, performing whole-contour shape matching under the assumption that there exists a decomposition into shape parts. An extension of Shape Context [3] is proposed, using the inner distance rather than Euclidean distance during descriptor construction. The inner distance is defined as the shortest path between two points in a shape, without the path leaving the shape at any time. This reduces to the Euclidean distance for convex shapes, but encodes information about the shape's structure when shapes are non-convex. A favourable quality is the inner distance's invariance to shape articulations. This property is proven under the assumption that there exists a valid decomposition of a shape into shape parts and junctions, although no such attempt to decompose shapes is made in the method. Experimental results show a moderate improvement in retrieval and recognition accuracy over shape context.

2.1.2 Part-Based Approaches

Separating Parts from 2D Shapes using Relatability (Mi and Decarlo)

Mi and Decarlo [34] follow a part-based approach, suggesting the decomposition of a shape into parts iteratively. Each iteration within the decomposition computes the "relatability" between two possible parts of a given shape parsing. The relatability is based on smooth local symmetries, which allows for cuts that appear clean. The relatability of a line segment joining two contour points (u, v) is a function of the turning angle between the unit tangents at u and v. Sweeping a line within a shape's contour, derivatives of relatability as a function of the spatial location of the sweep line are computed. The assumption is that large derivatives indicate moving through a transition region (sometimes called a junction area) to a natural part. A query contour is cut at locations with strong derivatives to create a part-wise decomposition of the shape. This work and the work in this thesis are based on the same principles by Singh *et al.* [44] in the psychophysics literature, where the concept of "shape cuts" has been defined.

Multi-Layer Eigenvector Shape Descriptor (Kim and Kim)

Kim and Kim [20] propose representing a shape by 21 distinct subregions, defined by the eigenvectors of different parts of the shape. The approach, which represents a shape in a quadtree, is invariant to scale, rotation and translation. With pixel i defined as $\ell_i = [x_i y_i]^T$, the mean vector of the shape is computed as

$$m_L = \frac{1}{N} \sum_{i=1}^{N} \ell_i \tag{2.1}$$

.

Building on the mean vector, the covariance matrix is computed by

$$C_L = \frac{1}{N} \sum_{i=1} N \ell_i \ell_i^T - m_L m_L^T \tag{2.2}$$

Finally, the eigenvectors e_j and eigenvalues $\lambda_j, j = 1, 2$ are computed by solving

$$C_L e_j = \lambda_j e_j \tag{2.3}$$

From these computations, the shape is initially divided into four regions, by cutting the shape along the axes defined by the eigenvectors at the center of mass of the shape. Then, the entire division process is repeated on each of those subregions. With the initial entire contour, plus all four of subregions, plus the four subregions from each of the initial subregions, $1 + 4 + (4)(4) = 21$ shape regions are constructed.

Within the quadtree representation, each node corresponds to a divided subregion. Stored in each node is a descriptor, encoding the ratio of its minor to major eigenvalues, the angle between the major eigenvector of the sub-region and the major eigenvector of the root shape, the distance between the subregion's center and the root shape's center, and the relative area of the subregion with respect to the whole shape.

The computed measures are concatenated into a feature vector. Then, two shape subregions can be compared for shape matching via a distance metric, where the distance is the Manhattan distance between the concatenated feature vectors of the shape sub-regions.

Bone Graphs (Macrini *et al.*)

Macrini *et al.* [31] present Bone Graphs, a shape decomposition algorithm based on ligature analysis. A query shape is decomposed based on the "bones" of the medial axis, where the bones correspond to medial branches. Having a shape part for every medial branch, however, results in oversegmentation of the shape due to perturbations in the medial axis. A ligature detection algorithm is used to locate and remove these perturbations, resulting in a simpler medial axis and a shape decomposition based on its medial representation. This is then represented in a graphical model, which allows for shapes to be matched through graph matching. This approach is similar to the work in this thesis, in that a medial representation of the shape is converted into a graphical representation, allowing for shapes to be matched through graph matching.

Parsing Silhouettes without Boundary Curvature (Juengling and Prasad)

Juengling and Prasad [18] propose a decomposition technique where subshapes are defined by simply connected regions whose skeleton does not bifurcate (branch out), leaving a "spine" as the only portion of the skeleton within the subshape in question. Subshapes are determined through an iterative process, where at each iteration a curve saliency algorithm is run to detect the most salient curve C_i on the contour. Then, the subshape corresponding to C_i is extracted as the union of triangles from the constrained Delaunay triangulation of the shape that correspond to vertices on C_i. Finally, this subshape has been extracted, so the saliency values for each of the points on C_i is set to 0 so the same subshape is not re-detected. From these subshapes, the shape cuts are computed as the lines of mutual intersection between subshapes. Each of these subshapes contributes up to two cuts of the shape, depending on criteria such as whether the endpoints of the spine reach the shape boundary or if it is entirely within the shape. This paradigm of shape parsing through boundary analysis is in line with the work presented in this thesis.

Parsing Silhouettes: The Short-Cut Rule (Singh *et al.*)

The inspiration for the presented parsing process comes from the "short-cut rule" for parsing silhouettes, proposed by Singh *et al.* [44]. The short-cut rule argues that, given several possible cuts of a shape and all other things being equal, the shortest possible cuts are selected in human vision for decomposing silhouettes. While this rule is concise and simple, it fails to specify how to generate the set of possible cuts to select from. For that, the “minima rule” is proposed. The minima rule states that silhouettes should be parsed across concave corners and negative minima of curvature along a shape’s boundary. Extensive experiments with human subjects demonstrates that the combination of the minima rule for cut generation and the short-cut rule for cut selection leads to shape parses that correspond well with what humans perceive as shape parts.

Chapter 3

Shape Parse Graphs for View-Based Object Recognition

3.1 The Shape Parsing Framework

A probabilistic model for shape parsing is presented, based on the assumption that the inter-part boundaries are a subset of the *shape cuts* proposed by Singh *et al*. [44]. We begin by presenting a novel definition of shape cuts, from which a graphical representation of a shape is built. Then, standard graph matching techniques for part-based representations are used to match shapes. Example output of this approach is visualized in Figure 3.1. The figure begins with two sample silhouettes, stacked vertically, and then provides sample output of this algorithm for these two silhouettes with two different parameterizations.

3.1.1 Shape Cut Identification

The first step in the parsing process involves identifying the set of shape cuts for a given query shape. Shape cuts are line segments that connect two boundary points on a silhouette. A subset of the shape cuts may be selected as "part boundaries", the shared edge between two adjacent shape parts. The proposed cut detection algorithm is not sensitive to rotation, scale, discretization artifacts or the precise location of corners.

Before defining shape cuts, we briefly introduce the two geometric features that define them: concave corners and the medial axis transform. In simple terms, for a silhouette boundary B, a concave corner $C \subset B$ is a maximal interval of connected boundary points whose negative curvatures are below some threshold. The curvature of a sequence of connected boundary

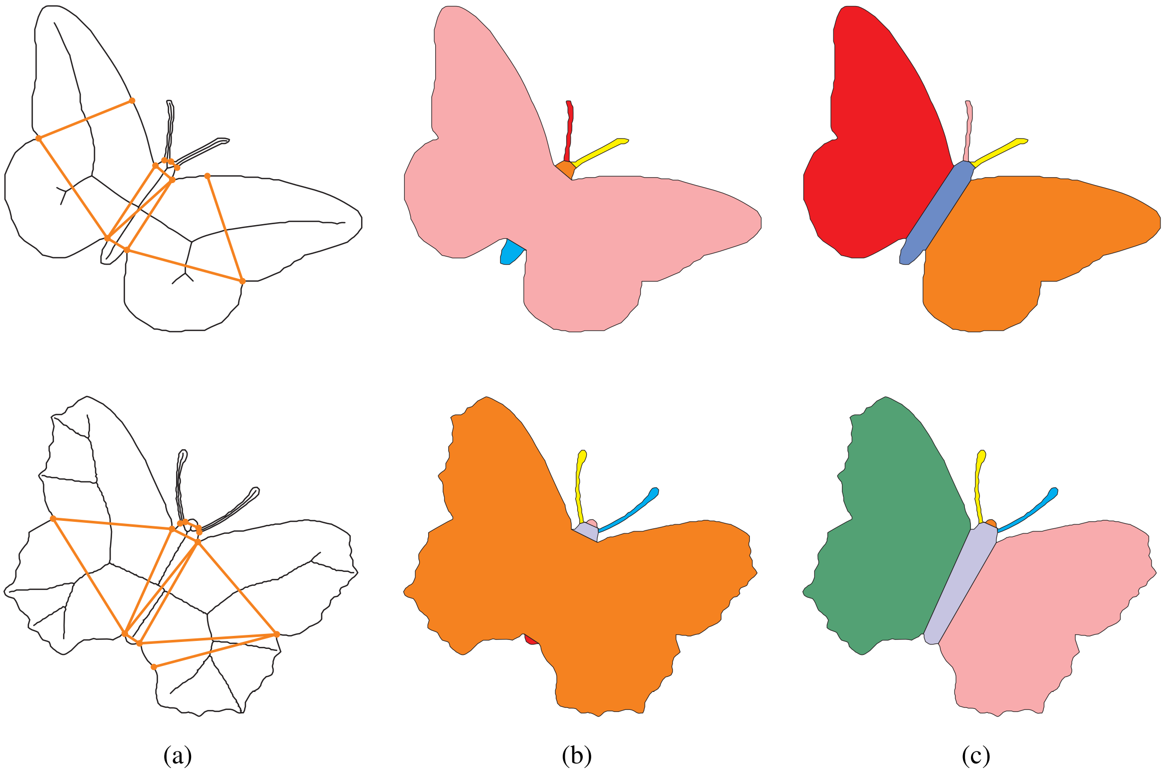

Figure 3.1: Example output of the algorithm: (a) Each shape cut (orange segments) joins a concave corner to another concave corner, or to its closest boundary point across selected axes of symmetry. (b) and (c) are the two most probable parses of the shapes in (a), according to a parameterization that encodes preferences for both short and parallel inter-part boundaries.

points is computed with the algorithm presented by Chetveriko and Szabo [8]. In this approach to corner detection, points are sampled along the boundary and a triangle is formed. The sequence of points is classified as a corner if the triangle has a sufficient opening angle. We often consider just the single *peak* boundary point in a concave corner, $\lambda(C)$.

The medial axis transform [13] is the union of maximally inscribed discs in a shape. Building on this, a *skeletal point* is the center of a single maximally inscribed disc, such that the disc is bitangent to at least 2 points, and a *regular* skeletal point s is the center of such a disc that touches exactly 2 points, p_i and p_j. The curves formed by connected regular skeletal points are called *skeletal branches*. These skeletal branches are used as axes of symmetry, such that the symmetry axis of points p_i and p_j is the skeletal branch containing the center of the maximally inscribed disc bitangent to p_i and p_j (Figure 3.2).

We are interested in keeping only one shape cut per corner-symmetry axis pairing, to avoid overpopulating the set of shape cuts that encode roughly the same information. Let the tuple (p_i, p_j, b) denote that the boundary point p_i is symmetric to the boundary point p_j with respect to skeletal branch b. Shape cuts are formed by pairing the peak of each concave corner with selected boundary points across different axes of symmetry. Let $W = \{(p_i, p_j, b)\}$ be the set of all possible tuples encoding pairs of symmetric points, such that i belongs to some concave corner C. Also, let $Z = \{(p_i, p_j)\}$ be the set of all shape cuts, such that for each $(p_i, p_j) \in Z$, there is a concave corner C with $\lambda(C) = p_i$, and a symmetry axis b for which one of the following two conditions is satisfied:

(a) for some $(p'_i, p'_j, b) \in W$ and some concave corner D, $p'_i \in C, p'_j \in D, \lambda(D) = p_j$, $p_i < p_j$;

(b) p_j is not a concave corner point, and the distance between points p_i and p_j is shorter than the distance between p'_i and p'_j, $\forall$ $(p'_i, p'_j, b) \in W$ and $p'_i \in C$.

That is,

(a) Shape cut $(i, j) \in Z$ connects either the peaks of two concave corners, $\lambda(C)$ and $\lambda(D)$, which contain symmetric boundary points, *or*

(b) Shape cut $(i, j) \in Z$ connects the peak of corner C and its closest point among all points symmetric to C across axis b. Figure 3.2 illustrates the set of cuts for a sample silhouette.

Note that $(i, j) \in Z$ does not necessarily imply that points i and j are symmetric, because of condition (a) above where we "snap" to the corner peaks if two concave corners have symmetric boundary points. This is advantageous because it makes the identification of shape cuts

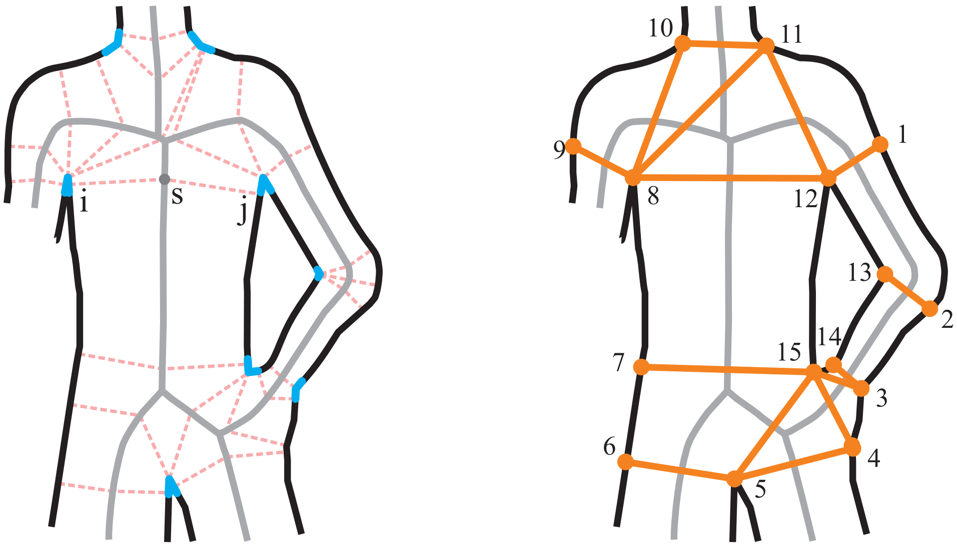

Figure 3.2: LEFT: Concave corner points (colored cyan). Example symmetric points $\mathbf{p}_i$ and $\mathbf{p}_j$ wrt skeletal point s. Subset of skeletal spokes (pink line segments). RIGHT: Shape cuts (orange segments). The shape cut endpoints correspond to peaks of concave corners and/or to nearest symmetric points to a corner. (8, 12) is a shape cut connecting 2 concave corners, while (7, 15) is a shape cut connecting a concave corner (12) to the symmetric point (7) across a medial branch.

less dependent on exact point symmetries, increasing robustness to noise. For example, the cut $(3, 14)$ in Figure 3.2 is formed by two corner peaks that are not strictly symmetric.

3.1.2 Shape Parse Graphs

It is assumed that the "hidden", latent boundaries separating a shape into parts correspond to a subset of the shape cuts in Z. The uncertainty about which parts are *inter-part boundaries* is represented by a joint probability distribution over a collection of binary random variables $(X_{ij})_{(i,j)\in Z}$, called *cut variables*, such that $X_{ij} = 1$ if shape cut (i, j) is an inter-part boundary, and $X_{ij} = 0$ otherwise. Then, each possible shape parse can be expressed by a joint instantiation of each X_{ij}.

With a given instantiation of each cut variable implies a set of inter-part boundaries, which implicitly yield a set of parts representing a given shape. These boundaries and shape parts give rise to a *shape parse graph (SPG)*, an undirected graph $G_0(V_0, E_0)$ with $V_0 = \{1, \ldots, n\}$, representing each maximal shape region containing no interior inter-part boundaries, and $E_0 =$

$\{(p_i, p_j)\}$, representing the inter-part boundaries $(p_i, p_j) \in Z$ connecting two adjacent shape parts in V_0.

Each shape maps to several Shape Parse Graphs, each of which correspond to a different joint instantiation of the cut variables. This graph representation permits enforcing spatial constraints between parts when matching SPGs, and casts the recognition problem as graph matching.

3.1.3 Matching Shape Parse Graphs

To match two Shape Parse Graphs, an inexact graph matching algorithm based on minimum-weight bipartite matching is employed [31]. Each SPG is represented by a directed acyclic graph (DAG), where the nodes are attributed by their shape context descriptor [4] and the edges between nodes encode the spatial hierarchy of the shape parts. From that, the graph matching algorithm measures the similarity between the two target DAGs by computing the bipartite node pairings that maximize the summation of each pair's similarity, while respecting the spatial hierarchy of the shape parts, which is imposed by the edges in the SPG.

To compute the distance between two SPGs, there are two factors at work: node similarity cost and hierarchical similarity cost. Given a possible bipartite matching between two SPGs, $\mathcal{G}_1$ and $\mathcal{G}_2$, computing the node similarity costs is a very simple process. For each matched node pairing $\mathcal{P} = (p_n, p_{n'})$, $p_n \in \mathcal{G}_1$, $p_{n'} \in \mathcal{G}_2$, the distance between these two nodes is computed by the shape context distance, $\delta = \mathcal{S}(p_n, p_{n'})$ [4]. For shape parts $(p_n, p_{n'})$, the shape context distance is computed as the sum of the average transformation costs for each sampled point from p_n to $p_{n'}$ and from $p_{n'}$ to p_n. With $C(x, y)$ denoting the transformation cost from point $x \in p_n$ to point $y \in p_{n'}$, the equation becomes:

$$\mathcal{S}(p_n, p_{n'}) = \frac{1}{|p_n|} \sum_{x \in p_n} \operatorname*{argmin}_{y \in p_{n'}} C(x, y) + \frac{1}{|p_{n'}|} \sum_{y \in p_{n'}} \operatorname*{argmin}_{x \in p_n} C(y, x) \tag{3.1}$$

Then, the node similarity $\mathcal{N}$ between two SPGs is simply the sum of the pairwise node similarity costs for each pair $\mathcal{P}$:

$$\mathcal{N} = \sum_{p \in \mathcal{P}} \mathcal{S}(p_n, p_{n'}) \tag{3.2}$$

Beyond node similarities, the cost for a minimum-weight bipartite matching solution also takes into account the hierarchical similarities between the two graphs. The hierarchies of interest include ancestor, descendent, and sibling correspondences. For each assigned node pairing $\mathcal{P} = (p_n, p_{n'})$, we assign a separate penalty to this assignment if one of the spatial hierarchies of interest are not respected. In Figure 3.3, we see a possible node correspondence

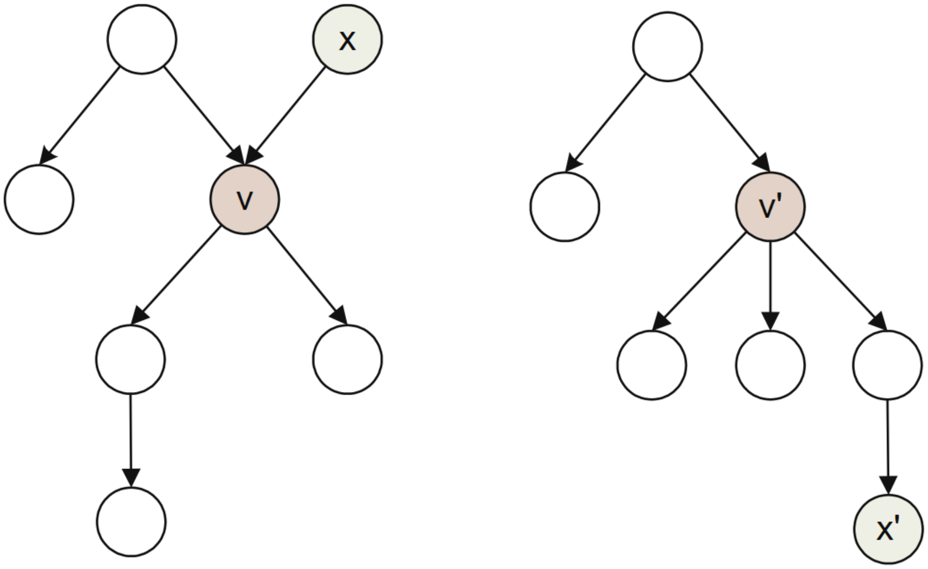

Figure 3.3: An illustration of the ancestor hierarchical constraint. [31]

between two SPGs. Assuming that $\mathcal{P}_1 = (x, x')$ is an assigned correspondence, the assignment of $\mathcal{P}_2 = (v, v')$ violates the ancestor hierarchy, because x is an ancestor of v, but x' is not an ancestor of v'. Therefore, a penalty is invoked on such a correspondence between the two graphs, making this correspondence assignment less likely for matching the graphs. A similar penalty is assigned for a violation of the descendant constraint, when x is a descendant of v, but x' is not a descendent of v' and the correspondences $\mathcal{P}_1 = (x, x')$ and $\mathcal{P}_2 = (x', v')$ are assumed. A final penalty is defined for the sibling hierarchical relationship, where x is a sibling of v (that is, x and v share a common parent node), but x' is not a sibling of v', and the same correspondences are assumed. The total hierarchical similarity cost $\mathcal{H}$ is computed as the sum of penalties assigned to each correspondence $\mathcal{P}$ between $\mathcal{G}_1$ and $\mathcal{G}_2$.

The minimum-weight bipartite matching algorithm computes the correspondences with the lowest joint-weight:

$$\mathcal{W} = \mathcal{N} + \mathcal{H} \tag{3.3}$$

While $\mathcal{W}$ returns the solution for the node correspondences between $\mathcal{G}_1$ and $\mathcal{G}_2$ with minimal cost, it also provides the cost of matching the two SPGs. Performing these computations between a query SPG $\mathcal{Q}$ and all SPGs in the database, the label assigned to $\mathcal{Q}$ is the labeling of the SPG with the lowest $\mathcal{W}$ among all SPGs in the database.

3.1.4 Shape Matching Against the k-Most Probable Parses

Given all cut variables for a target shape, it is necessary to decide which to activate to maximize matching accuracy. That is, which shape cuts should be used to break a silhouette into shape parts? We propose using a probablistic measure to decide which parses of a shape are more probable than others. This probablistic measure can be described by natural potential functions along connected shape cuts. Although we do not advocate a specific parameterization, an example of this could be "from each cycle of shape cuts, select the most parallel cuts" and "for each maximal cycle formed by the shape cuts, select the shortest cut in that cycle". This is similar to the parameterization used in Figure 3.1. The cycles we describe are purely in the sense of cycles on graphs, where two shape cuts are connected if they share a node p_i.

Once such a parameterization is established, all possible parses of a shape can be ranked by their probabilities. Rather than selecting the single most probable parameterization and always matching against it, we propose to keep the k most probable parses of each shape, where k is a small integer. Although this increases the size of the search space from $O(n)$ to $O(kn)$, the improvement in matching performance makes this beneficial. Since determining the class of a query object involves linearly matching each element in the database, this also means that the time required for recognition grows linearly by a factor of k. By keeping more than the single parsing of a shape, the matching algorithm becomes more resistant to choices of parameterizations where unexpected perturbations in the shape cause an incorrect parse to be ranked as more probable than an expected parse. This outlier is unlikely to match well against other query shapes, but the $k-1$ other parsings may yield the desired matching.

3.2 Shape Part Training

The effects of outlier parses can be further minimized by exploiting how common a single part is among all objects of its class. This is achieved by punishing graph matches that exclude shape parts that are "characteristic" of a given class. Each node within each SPG is weighted based on how well it matches other SPGs within the same model class. For each part p, all SPGs from all shapes in the same class as p are gathered. We denote this set as S. For example, if p corresponds to a shape of a dog, S is the set of all Shape Parse Graphs representing dogs. The SPG of p is then matched to each SPG $s \in S$, using the method described in Section 3.1.4. Based on these matches, a graph $G_1(V_1, E_1)$ is built where V_1 is the set containing p and all SPG nodes that p matched with, and E_1 is the set of edges connecting p to each $v \in V_1$. Then, every $v \in V_1$ is matched against the SPG of every other node in V_1. If v is matched to a node already in V_1, these two nodes are connected by an edge in G_1.

Finally, let the weight of shape part p be the cardinality of the maximum clique of the final graph G_1, built based on the nodes that p matches with. Larger cliques indicate good, representative parts for matching.The byproduct of this weighting scheme is that spurious outlier parses of a shape tend to generate smaller cliques, and will get poor weights. This encourages the matching algorithm to select one of the other $k-1$ parses of the shape for matching.

Although finding the maximum clique on a graph is NP-complete, we are more concerned with which graphs have larger max-cliques than others. This can be approximated by representing the nodes in a graph as a bit string and randomly permuting the bits to generate subgraphs. It is trivial to test if these subgraphs form a clique, making this significantly more tractable than finding the true maximal clique. Running this for the same number of iterations on every generated graph gives a good approximation for the weights, and generates favourable results in practice.

3.3 Shape Matching Experiments

We use an object recognition database that consists of 1065 different shapes, spanning 43 different object classes [42]. Intra-class object views have high variation, including different scales, rotations, views, and occlusions. We test recognition performance by taking a single query view out of the database and computing its SPGs. Then, the k most probable SPGs are retrieved and a graph matching algorithm [31] is used to compute the dissimilarity between the query SPGs and all SPGs in the database. We evaluate the impact of the selection of k in Figure 3.5. The descriptor used to match shape parts is shape contexts [4]. The result of the graph matching is a distance δ between the two graphs. The final value given to an SPG matching can be denoted $v = f(\delta, \omega)$, a function of the graph distance and the weights defined by computing the maximum cliques on our database shapes, ω. After v is computed between the query and all other SPGs, scores are ranked in increasing order. If the class of the top ranked SPG matches the query, then recognition is deemed successful. [1]

In Figure 3.4, the impact of considering not only the graph distance δ, but also the maximum clique weights ω is outlined. When matching against the 4-most probable SPGs for each model in the database, we achieved just over 90% recognition accuracy. However, this result is boosted to 95% when the maximum-clique weights ω are accounted for. We have observed similar improvements at all tested values of k.

[1] Source code available at `http://www.github.com/ChrisWhiten/VideoParser`

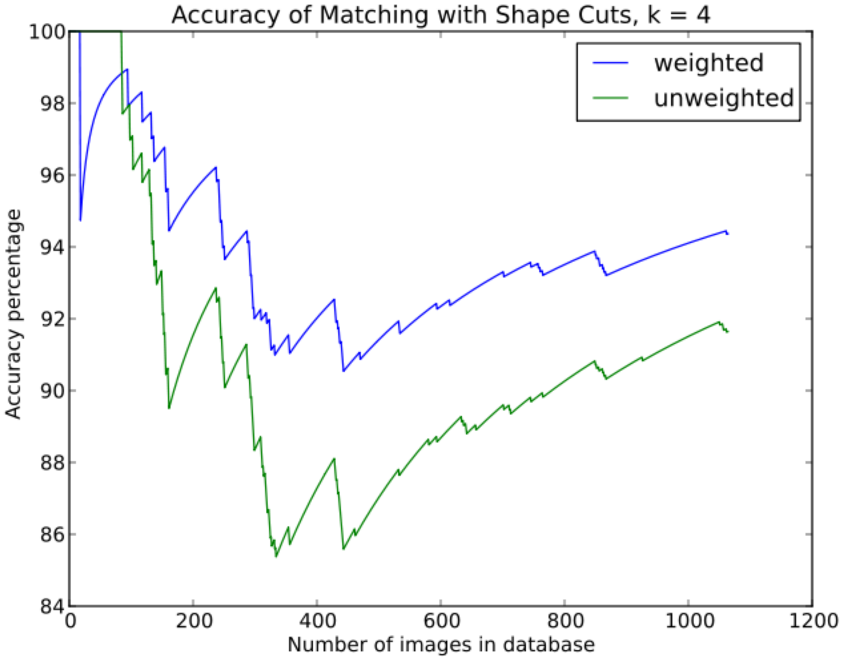

Figure 3.4: Object recognition results showing the accuracy of matching SPGs on a 1065 view, 43 class database. Considering how common a specific shape part is within its class (blue) achieves measurably better matching accuracy. On average, we see a 5% accuracy improvement over unweighted matching (green).

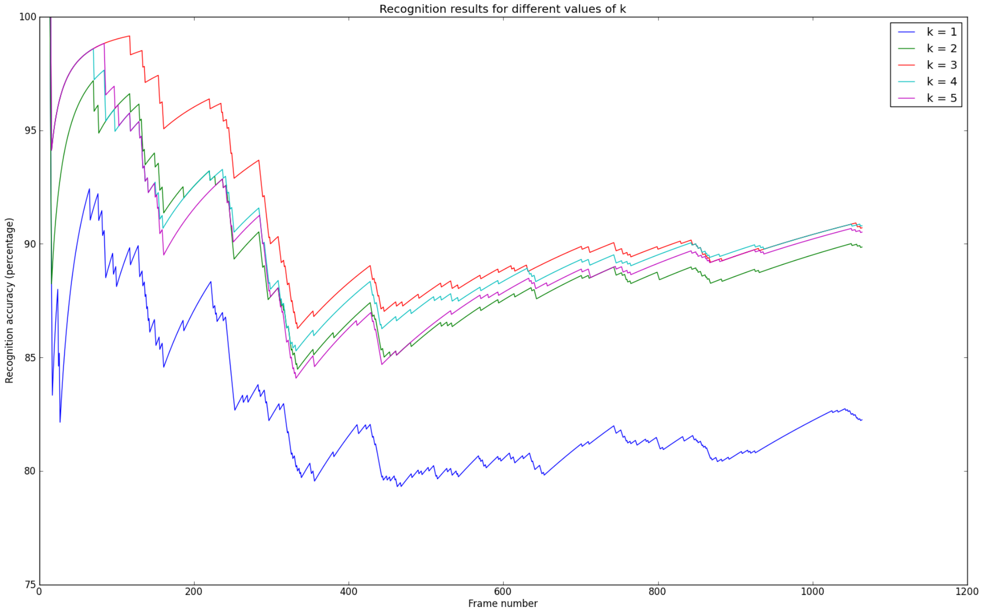

Figure 3.5: Otherwise identical recognition experiments performed with different values of k, showing that having a k larger than 1 can greatly impact recognition accuracy.

Finally, we evaluate the unweighted recognition performance on our data set for different values of k, which give the following results. This showcases the power of considering more than the single most probable parse.

k = 1	k = 2	k = 3	k = 4	k = 5
82.3%	89.2%	90.7%	90.9%	91.9 %

Chapter 4

Action Recognition - Related Work

4.1 Related Work

The avenues of research for action recognition generally fall into three categories: higher level representations, trajectory analysis, and local spatio-temporal features with a bag-of-words representation. Furthermore, within the categories of trajectory analysis and local spatio-temporal features, approaches can be separated into dense and sparse feature extraction. State of the art approaches in high-level representations [39] and dense trajectory analysis [48] currently demonstrate the greatest accuracy across a wide range of datasets, but methods in these categories tend to be prohibitively slow, making them infeasible for use in large datasets or real-time applications. On the other hand, there exist several more efficient approaches based on sparse features, but these approaches achieve their computational gains at the expense of recognition accuracy. In this section, relevant related work is summarized, with the intent of relating the contributions of this thesis to the present-day action recognition literature.

4.1.1 High-Level Representations

Many techniques build complex representations in an attempt to encode semantically meaningful descriptions. These representations are designed through a combination of low-level approaches, constructing a hierarchy of action representations. Higher level models are attractive due to their high recognition performance and intuitive construction. However, these systems suffer from an excessive amount of required computation time, making them unsuitable for most applications. We briefly outline interesting and successful algorithms that follow this paradigm.

Action Bank (Sadanand and Corso)

Sadanand and Corso [39] adopt a methodology of recognition by detection, which has recently been successful in object recognition [27]. The basic idea to such an approach is to test a large set of action detectors against each space-time cube, pooling the results of each detector to generate a single feature. Each detector returns a separate SVM response, returning various degrees of the likelihood of the given space-time cube containing a given action. Each of these responses is unified in a volumetric max-pooling setup, which extracts the single feature.

The selected action detectors are simple, template-based detectors with invariance to appearance changes [12]. While these detectors are invariant to appearance changes, they are *not* invariant to scale, viewpoint, or tempo changes. To account for these phenomena, the set of action detectors must be very large, with each detector encoding a separate scale, tempo, and viewpoint for each action. The requirement for a larger set of detectors is a disadvantage to Action Bank, yielding very slow processing times. Processing a single video of the UCF0 database requires 204 minutes, not including detector construction. Furthermore, it is not clear how to automatically select the action detectors, which results in the user hand-selecting the detectors a priori. Regardless of these disadvantages, Action Bank achieves significant accuracy gains over other state of the art methods. State of the art recognition performance is reported on 4 standard benchmark datasets (KTH [41], UCF Sports [37], UCF50, and HMDB51 [23]).

Action Recognition by Learning Mid-Level Motion Features (Fathi and Mori)

Fathi and Mori [16] use tracking to localize an action performed by a human, limiting the effect of background motion on the action recognition. At a high level, the approach constructs a series of mid-level features by combining spatio-temporally close low-level motion features (optical flow). The optical flow field is treated as a set of weak classifiers, and classifiers that are spatio-temporally close are combined (with Adaboost [40]) to form a more powerful feature representing a spatio-temporal cuboid. These more powerful features are then reconsidered as weak classifiers, allowing the set of weak classifiers representing separate spatio-temporal cuboids to be combined into a single overarching classifier for the entire image sequence.

Like other high-level approaches to action recognition, this algorithm suffers from a high computational cost. Reported results claim a running time of 0.2 to 4 seconds per frame, which is significantly slower than real time (most videos run at approximately 25 frames per second). The crutch of this approach lies in requiring both person tracking and classifier construction at run-time.

4.1.2 Trajectory Analysis

Tracking interest points and analyzing the motion trajectory has recently been shown to be another effective approach for action recognition [33, 15, 48]. Such methods are popular due to the simplicity of tracking keypoints, with methods such as the KLT feature tracker [30]. However, these approaches tend to perform poorly without a very large set of well-distributed trajectories to track. With that in mind, dense trajectories provide favourable performance, but poor computational performance. Meanwhile, sparse trajectories, while more efficient, may not produce satisfactory recognition results.

Velocity Histories of Tracked Keypoints (Messing *et al.*)

Messing *et al.* [33] use the KLT feature tracker [30] on 3-dimensional Harris corners [24] to build descriptors based on the velocity history of keypoints. Each tracked keypoint generates a trajectory which is binned with a log-polar histogram, with bins for velocity magnitude and direction. While standard trajectory analysis methods use short trajectories to avoid noise, this approach prefers long trajectories to compute rich velocity history descriptors. The assumption is that an employed mixture model will handle noise caused by long trajectory drift.

Dense Trajectories (Wang *et al.*)

Wang *et al.* [48] take a dense sampling approach for trajectory analysis. At multiple spatial scales, features are sampled on a dense grid and a dense optical flow field is computed. The points are tracked between two consecutive frames t and $t+1$ by median filtering on the dense optical flow field, $\omega = (u_t, v_t)$. For a point $P_t(x_t, y_t)$, the tracked point is computed in the next frame by

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * \omega)|(\bar{x}_t, \bar{y}_t) \tag{4.1}$$

in which $(M * \omega)$ denotes convolving the optical flow field with a median filtering kernel, and $(\bar{x}_t, \bar{y}_t)$ denotes the rounded position of (x_t, y_t). These points are concatenated up to a predetermined length to create a series of trajectories. The trajectories are kept short to avoid the problem of trajectory drifting.

From these trajectories, spatio-temporal features are constructed and binned in a bag-of-words representation, which can then be classified by an SVM. Each trajectory is described by a combination of its normalized displacement vectors and a HOG/HOF [25] descriptor. Histogram of Oriented Gradients (HOG) describes a local image patch by counting the occurrences of gradient orientations from a dense sampling within that patch. These are binned into a histogram representation, thus the descriptive name of “histogram of oriented gradients”. In a similar fashion, Histogram of Optical Flow (HOF) describes a local space-time cube by densely computing the optical flow in an image patch and binning the optical flow vectors by their orientation. The normalized displacement vector describes the trajectory’s shape, and is computed by

$$S = \frac{\Delta P_t, ..., \Delta P_{t+L-1}}{\sum\limits_{j=1}^{t+L-1} ||\Delta P_j||} \tag{4.2}$$

where L is the length of the trajectory. The HOG/HOF descriptor is extracted from a space-time cube around the trajectory, combining a histogram of oriented gradients (HOG) descriptor for appearance modelling and a histogram of optical flow (HOF) descriptor for motion modelling.

This approach demonstrates high recognition on several benchmark datasets, with state of the art performance on a subset. Nonetheless, the approach requires significant tuning of parameters (trajectory length, sampling size, feature balancing, etc). Furthermore, the approach is infeasible for large datasets due to the complexity of computing the Euclidean distance to match descriptors in a dense sampling paradigm.

4.1.3 Local Features

The contribution in this thesis falls under the domain of recognition with local spatio-temporal features. Using local spatio-temporal features with a bag-of-words model has remained an effective method for producing efficient and accurate action recognition algorithms. These approaches are appealing due to their increased efficiency over more complex algorithms, their competitive accuracy, and their ease of implementation.

Space-Time Interest Points (Laptev and Lindeberg)

Laptev and Lindeberg [24] introduce the idea of extending spatial descriptors into the temporal domain by extending the Harris corner detector. Rather than accepting image patches with significant change in the (x, y) dimensions, image patches are only accepted as keypoints with sufficient change in the (x, y, t) dimensions, appending time to the standard Harris formulation. The intuition to such an approach is that interesting points contain a strong reversal of direction, such as clapping hands moving in a forward-backward motion pattern. By this criteria, the number of keypoints returned in a video sequence tends to be very low, resulting in a lack of discriminative information and poor classification results. Nevertheless, the simplicity of the approach grants very efficient computational performance.

Learning Realistic Human Actions from Movies (Laptev *et al.*)

Laptev *et al.* [25] improve upon the previously described approach by detecting keypoints at multiple scales. From these detections, a descriptor is built based on a coarse histogram of oriented gradients (HOG) and histogram of optical flow (HOF). Describing a space-time keypoint by these histogram features yields a significant improvement on accuracy, at the cost of computational performance, due to the optical flow and gradient computations.

MoSIFT (Chen and Hauptmann)

Further extending upon the work previously described, Chen and Hauptmann [6] continue with this methodology in the construction of MoSIFT, a spatio-temporal feature detection/description scheme. The contribution in this thesis is most similar to this approach, and can be seen as an improvement on this work. At each frame throughout a video sequence, the SIFT detector [29] is used to detect spatially interesting keypoints at each frame. Alongside the SIFT detection, optical flow [30] is computed at each frame. Each detected SIFT keypoint is only kept if there is sufficient optical flow at that location. This keypoint pruning is aimed at disregarding keypoints belonging to static objects in the background, keeping only keypoints with significant motion. From the remaining keypoints, a descriptor is constructed by computing the 128-floating point value SIFT descriptor and 128-floating point value optical flow descriptor at each remaining keypoint. The two descriptors are amalgamated into a single one, named MoSIFT, which independently encodes both the motion and appearance of a scene. Due to its relevance in the context of this thesis, the pipeline for computing MoSIFT descriptors is shown in Figure 4.1. The boxed components in the figure are the main causes of slow computations, which we target for replacement by more efficient means.

While very similar to the HOG/HOF approach by Laptev *et al.* [25], the explicit use of SIFT and optical flow yields improved recognition accuracy. This technique has been one of the most successful methods for event detection at NIST TRECVID workshops on video surveillance [7]. However, beyond the improvement in recognition accuracy is an added computational cost. MoSIFT remains too computationally heavy for use in real-time systems, such as streaming video surveillance applications.

Motion Interchange Patterns (Kliper-Gross *et al.*)

Kliper-Gross *et al.* [21] capture the local geometry of motion with self-similarity [43] computations throughout the temporal domain. Each frame is densely sampled, where each pixel centers a series of self-similarity computations with a geometric pattern in one frame from

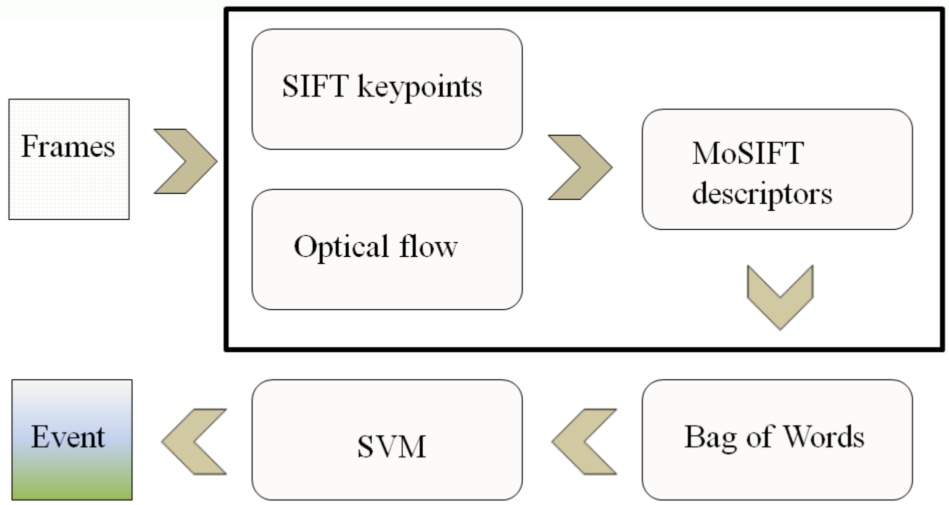

Figure 4.1: The pipeline for detecting events with MoSIFT [7]

the past and one frame from the future, generating a 64 trinary digit descriptor which encodes the motion of the pixel's local neighbourhood. By using frames from both the present and future, there is more information available to capture how a pixel is moving throughout time, in comparison to approaches such as optical flow which only use the past frames.

This approach, called Motion Interchange Patterns, currently reports the highest recognition accuracy on several standard benchmark datasets. Like other methods, this approach is infeasible for real time action recognition, due to the dense sampling used. Each dense sampling computation returns a feature descriptor. The cost of matching these feature descriptors under standard action classification frameworks, using a bag-of-words model, requires an excessive number of Euclidean distance computations, slowing down the approach drastically. The contribution in this thesis also draws inspiration from this approach of capturing the geometric structure of motion through self-similarity computations.

Chapter 5

MoFREAK

5.1 MoFREAK

Several space-time descriptors, such as [25] and [7], leverage gradient-based methods to encode the appearance of an action, while using optical flow to capture its motion. While these works have demonstrated encouraging recognition accuracy, they can falter in practice due to the computational complexity of computing these descriptors. A pure binary setup is desirable for efficient keypoint detection, description, and matching.

Alahi *et al*. recently presented FREAK [1], a binary descriptor for image recognition with the potential to replace gradient-based appearance encodings. FREAK is constructed with short binary strings, based on intensity comparisons within a sampling pattern inspired by the human retina. This sampling pattern is seen in Figure 5.1. FREAK descriptors are efficient to compute and compare, while presenting competitive accuracy compared to SIFT [29] in the object recognition domain. These results inspire us to explore its potential in the action recognition domain.

We present MoFREAK, a compact binary spatio-temporal feature descriptor which is inspired by recent advances in descriptor representation for both object and action recognition [1].

5.1.1 Appearance Modelling

Several approaches for action recognition have successfully encoded action appearance by directly porting successful feature descriptors from object recognition into the action recognition

[1] Source code available at `http://github.com/ChrisWhiten/MoFREAK`

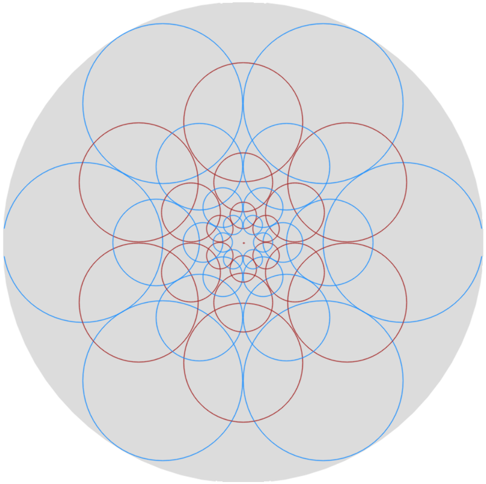

Figure 5.1: A visualization of the FREAK sampling pattern

domain [47, 24, 14]. We adopt a similar methodology in constructing MoFREAK, employing the recently proposed FREAK descriptor [1] as an appearance component. FREAK is advantageous by virtue of its binary nature, making it efficient to compute and match descriptors, while retaining high recognition accuracy.

FREAK

Alahi *et al*. [1] have proposed a local spatial feature descriptor with the aim of efficient feature extraction and descriptor matching. FREAK achieves both of these goals by avoiding costly gradient computations, instead opting for simple intensity comparisons along a pre-defined sampling pattern for a selected image patch. Each comparison yields a binary decision, indicating the greater of the two compared intensity values, leading to a complete binary descriptor.

An image patch is sampled densely towards the center of the patch, with the sampling density decreasing as the distance from the patch center increases. Samples near the center are smoothed with narrow Gaussian kernels, having a small standard deviation. The width of the Gaussians also increases with sampling distance from the patch center. FREAK refers to these samples as receptive fields, due to the sampling pattern's inspiration from the human retina. A series of intensity comparisons among the receptive fields constructs the binary string.

Formally, an n-bit FREAK descriptor F is constructed from a sequence of one-bit Difference of Gaussians comparisons between pairs of receptive fields, P_a.

$$F = \sum_{a=0}^{n} 2^a T(P_a) \tag{5.1}$$

where $T(P_a)$ returns a binary decision based on the Difference of Gaussian comparisons.

$$T(P_a) = \begin{cases} 1 & \text{if}(I(P_a^{r_1}) - I(P_a^{r_2})) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

In this context, $I(P_a^{r_1})$ denotes the smoothed intensity of receptive field r_1, the first receptive the field of pair P_a.

Appearance Component Construction

At each time step t throughout a query video sequence, we compute the absolute difference image $A_t(x, y)$ between $F_t(x, y)$, the current frame, and $F_{t-5}(x, y)$, the frame 5 time steps in the past.

$$A_t(x, y) = |F_t(x, y) - F_{t-5}(x, y)| \tag{5.3}$$

On $A_t(x, y)$, we detect the location and scale of numerous keypoints with the BRISK keypoint detector [26]. This process is demonstrated in Figure 5.2. From these detections, we extract FREAK descriptors within the difference frame $A_t(x, y)$ at each of the detected locations and scales. Using the difference image allows the detector to find keypoints that are interesting in both the spatial and the temporal domains, owing to the difference image's ability to implicitly encode both appearance and motion. Using the difference image has the added side-effect of performing rudimentary background subtraction, which assists in avoiding spurious features and avoids overfitting to static environments.

One common issue with employing the difference image in recognition tasks is its sensitivity to appearance variations, where the difference image yields a high intensity value when a moving actor's colour and the background colour are of significant difference. While this phenomena does not disappear within the context of constructing this descriptor, it is of little relevance. The use of FREAK on the difference image makes the descriptor invariant to these intensity variations, by quantizing the intensity differences on the difference image to a binary decision.

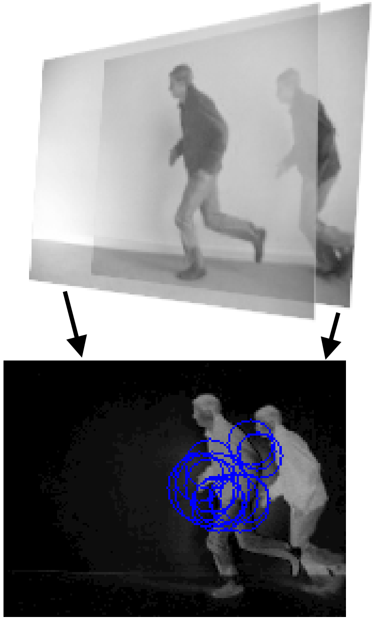

Figure 5.2: Keypoints are detected on the difference image between each frame and the frame 5 time steps in the past, permitting implicit motion encoding and robustness to static environments.

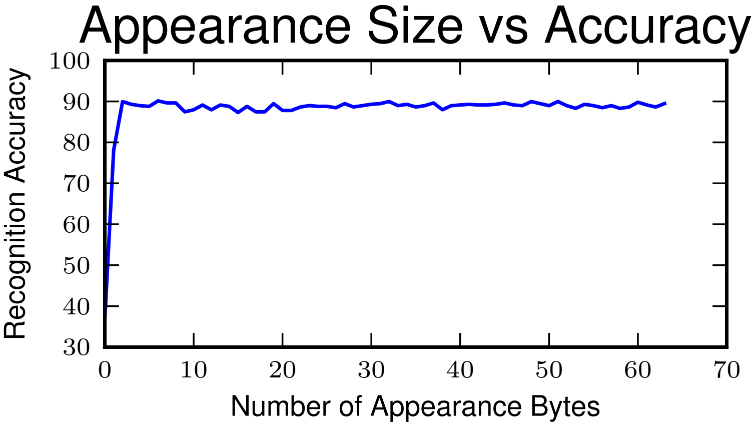

Figure 5.3: We compare the effect of the appearance component size on recognition accuracy. Significant increases are noted between 0 bytes and 1 byte of appearance data, and between 1 and 2 bytes of appearance data. The effect levels out when the appearance component is greater than 2 bytes, leaving us with a larger descriptor with no significant performance increase.

Selecting the Optimal Descriptor Size

As described in [1], the context of the bytes in the FREAK descriptor increases from coarse to fine as the byte index increases. The first 16 bytes mainly involve the perifoveal receptive fields, roughly corresponding to a human's peripheral vision. The remaining bytes help distinguish between finer details, which are of less interest in action recognition since we do not want to match specific appearances, such as a specific actor in a scene. By discarding the bytes related to finer details, we increase invariance to changes in appearance, constructing a descriptor that generalizes well across the set of possible actors in a scene. We have experimented with different descriptor sizes, the results of which can be viewed in Figure 5.3. We find that maintaining only the first 2 bytes of the FREAK descriptor, while discarding the remaining 62 bytes, leaves us with an appearance model that is compact, efficient to match, and discriminative enough to robustly recognize actions.

5.1.2 Motion Modelling

While the first bytes of the descriptor implicitly encode motion through the absolute difference image, it is insufficient for most applications due to its inability to capture the directionality of the motion at each keypoint. A common approach for encoding this information is optical flow, as can be seen in many popular action recognition algorithms, such as [7, 17, 16, 15]. While optical flow yields reasonable results, it is usually described by a set of integers or floating point values, which are expensive to match and impede on progress towards real-time recognition.

Recently, Motion Interchange Patterns have been proposed for modelling motion by a series of self-similarity patch computations [21]. Each frame is densely sampled such that each pixel yields a 64 trinary digit descriptor, where each trinary digit corresponds to the result of SSD computations in the local space-time neighbourhood. We adopt a binary variation on this technique with an extended neighbourhood pattern to build an 8 byte motion descriptor.

To construct the motion component for a MoFREAK descriptor, the detected keypoint is resized to be a 19×19 patch, on which we compute a series of self-similarity computations across the spatio-temporal domain. A self-similarity approach is taken to remain appearance invariant [43], which enables the model to learn the geometric structure of the motion, rather than encoding the details of the specific actor or environment. From this 19×19 patch, we perform identical computations centered at 8 spatial locations. The computations at each of the 8 spatial locations return a single byte of descriptor data, leaving us with a full 8-byte motion descriptor at the end of the process. These computations are centered at pixel locations $(5, 5)$, $(5, 9)$, $(5, 13)$, $(9, 5)$, $(9, 13)$, $(13, 5)$, $(13, 9)$, and $(13, 13)$, forming the pattern shown on the left-hand side of Figure 5.4.

We denote the patch $p_t(x, y)$ as the 3×3 patch in the current frame F_t centered at spatial location (x, y). We wish to evaluate how the intensity values in $p_t(x, y)$ have changed, relative to its spatial neighbourhood in past frames. Moving five frames into the past, we define eight separate 3×3 patches at the frame F_{t-5} which encode possible locations in F_{t-5} of the structure in p_t(x, y). These eight new patches, $p_i(x, y)\ \forall\ i \in [1, 8]$, are defined at the following spatial locations relative to the location of p_t: $(-4, 0)$, $(-3, 3)$, $(0, 4)$, $(3, 3)$, $(4, 0)$, $(3, -3)$, $(0, -4)$, and $(-3, -3)$. Each p_i will be included in an SSD computation against p_t, leaving us with 8 calculations. This process is visualized in the right-hand side of Figure 5.4. We convert the resulting SSD value into a binary decision by comparing its value against threshold θ, giving rise to a natural 8-bit descriptor for this set of computations. We define $b(i)$, the value of bit i, by the following equation:

$$b(i) = \begin{cases} 1 & : SSD(p_i, p_t) < \theta \\ 0 & : SSD(p_i, p_t) \geq \theta \end{cases} \tag{5.4}$$

The construction of this 8-byte descriptor is a simple set of independent SSD computa-

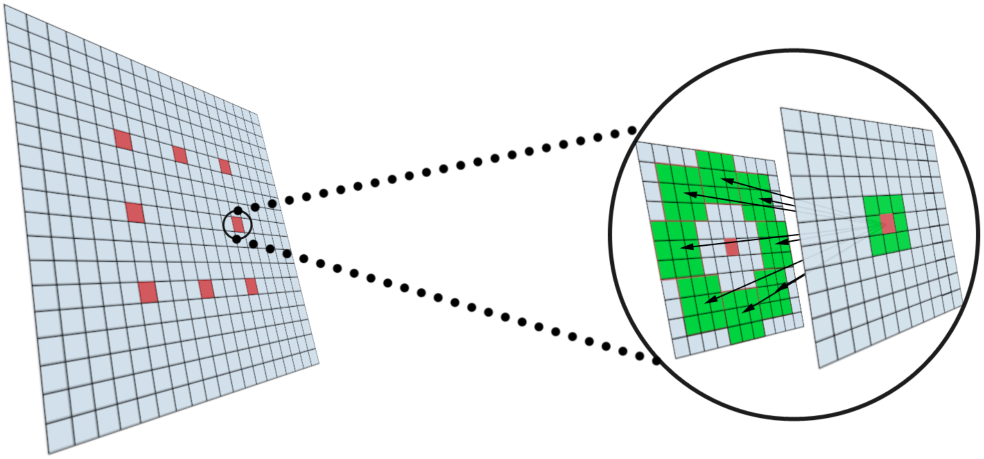

Figure 5.4: **Left:** Self-similarity computations are computed in 8 neighbouring locations around a detected keypoint. **Right:** A 3×3 patch is evaluated against a set of image patches in the past frame to compute 1-byte of the descriptor's motion component.

tions, making it efficient to compute and highly parallelizable. Encoding the intensity changes over several image patches within a keypoint's neighbourhood is advantageous compared to analyzing the movement of a single location, since complex motions over a larger area can be evaluated with our compact encoding.

5.2 Classification

Once MoFREAK points are computed throughout a video sequence, the features are processed and forwarded to a classifier to return a final decision on the action's class. While the classification process is a very important step of action recognition (and may be the step that impacts recognition accuracy the most), the focus in this work is on descriptor representation, rather than classification. For that reason, a simple setup was taken, where MoFREAK features are fed into a bag-of-words model, which returns a single feature for classification. This single feature is then passed into a Support Vector Machine (SVM) for a final classification decision. This setup was selected to allow for easy comparison between similar published methods, as many approaches in the current action recognition literature follow this pipeline. The approach is popular due to its implementation simplicity and favourable results.

5.2.1 Bag of Words

Bag-of-words [11] is a popular technique in the computer vision literature for simplifying and quantizing extracted features from a large, unbounded domain to a simple low-dimensional feature. Originally borrowed from the field of natural language processing, a set of spatio-temporal features is encoded as a distribution representing the occurrence frequency of a pre-defined set of “codewords’ (pre-selected features)’. Each detected feature increments the frequency count for the *closest* codeword within the codebook (the set of pre-defined codewords). The definition of “closest” is dependent on the distance metric used to compare features. Common distance metrics are the Euclidean distance, cosine similarity, and the Hamming distance.

The set of spatio-temporal features extracted within a given space-time cube all combine to give a single bag-of-words feature, which is the occurrence frequency distribution of that space-time cube, with respect to the pre-defined codebook. This single feature is then used for classification of the space-time cube, resulting in a single decision. The advantage to this approach lies in its simplicity. Not only is the approach simple to understand and implement, it also simplifies the feature space. This makes the problem of classification significantly more stable and tractable. Classification becomes more stable by becoming more robust to data noise and outliers. Similar features “snap” to a common codeword, while spurious features each contribute to only a single increment for another codeword’s count. As the number of features increases, the effect of such features further diminishes. Classification also becomes more tractable, since the dimensionality of the feature space is drastically reduced. For a codebook with 1000 codewords, an entire set of image frames is represented by a single 1000-dimensional integer feature.

One commonly cited problem with the bag-of-words model lies in the fact that the spatial and temporal relationships associated with each feature is thrown away, being replaced with a simple frequency distribution. While this problem still exists, no spatial localization is required within the scope of this thesis, diminishing the effect of this issue.

Codeword Selection

The task of selecting the original pre-defined codebook remains an open problem. It is preferable that the codebook accurately reflects the data distribution, to allow for the bag-of-words features to accurately capture the relevant nuances of the data. To understand why that is the case, imagine a codebook where a single codeword lies within a populous area of the data, where 99 % of the features within the data will quantize to that single codeword. Then, the remaining codewords become representative of features with low probability of occurrence, and every constructed bag-of-words feature will be very similar, with high density at the single probable codeword. Such features have low discriminative power, making it difficult for classifiers to find good decision boundaries to classify future features.

A common approach to codeword selection is to extract several features from training data and to cluster them with k-means clustering [46]. However, studies have shown [19] that using k-means to define the codebook often overfits to the densest regions of the feature space, making the resultant codebook no better than a codebook constructed by random cluster selection. Experiments on sample data verified that hypothesis within the domain of video action recognition, leading to the use of random cluster selection for codebook construction. The random cluster selection is class-balanced, so features from each classified labeling are equally represented in the codebook.

5.2.2 Support Vector Machines

The single feature output from the bag-of-words model is classified with a Support Vector Machine classifier [10]. SVMs are among the most popular classifiers for action recognition, with an efficient open source implementation available for use [5]. SVMs aim to discover the separating hyperplane with the maximal margin (maximized distance from the closest points to the separating hyperplane). For complex data, a kernel is used to map inputs into high-dimensional feature spaces.

In the context of this work, the histogram intersection kernel is employed. For bag-of-words features a and b, the histogram intersection kernel is defined as

$$K_{\cap}(a,b) = \sum_{i=1}^{n} min(a_i, b_i) \tag{5.5}$$

Although the χ^2 kernel is very popular in the action recognition literature, we found the difference in accuracy between the two kernels to be negligible, while the histogram intersection is more computationally efficient.

5.3 Extension to Event Detection

5.3.1 Problem Description

While most benchmark datasets tackle the problem of *action recognition*, a related problem with relevant application to video surveillance is the *event detection* problem. While event detection requires classifying video sequences (as action recognition requires), it also requires temporally segmenting the target video sequence to determine the temporal location that a classified event occurs at. This extension of temporal segmentation makes event detection significantly more difficult than action recognition.

More formally, given a collection of surveillance data files (videos from airport surveillance cameras) for preprocessing, select a pre-determined topic and, at test time, return a set of tuples containing the event's beginning and end times, along with the video file in which the event as located from the surveillance data files, ranked by likelihood of meeting the need described in the topic [35]. By ranking the retrieved segments by likelihood, a human inspector can quickly verify the most likely events by searching at the top of the returned list.

A preliminary version of the MoFREAK descriptor has been applied to the task of event detection, within the context of TRECVID 2012 [35]. The target application was to detect a pre-defined event in many hours of video data, returning the temporal locations where the event occurred in the data.

5.3.2 Data Description

All training and event detection was completed on the Gatwick dataset, which contains surveillance footage captured at the London Gatwick airport. The data is recorded with MPEG2 compression at a resolution of 700×480 with 25 frames per second. 5 separate camera views are present, each containing video data from several different days. This results in a dataset with a great deal of diversity, with each camera capturing different viewpoints and degrees of "crowdedness". The spatial locations of the cameras, as well as a sample frame captured from each camera, is seen in Figure 5.5. The cameras capture: a controlled access door, a crowded waiting area, a debarkation area, close-up view of an elevator, and a crowded transit area. In particular, the waiting area and transit areas are extremely crowded, making for very difficult data to correctly classify. In contrast, the elevator close-up view rarely contains even a single person.

The dataset is split into 100 hours of training data and 15 hours of test data. The ground truth for the temporal segmentation has been provided for the training data, granting sufficient data to train classifiers. The ground truth is not provided for the testing set.

5.3.3 Approach

A priori, we decided to detect *PersonRuns* events within this data. A query video sequence will incrementally build new bag of words features, which can be fed into an SVM to classify

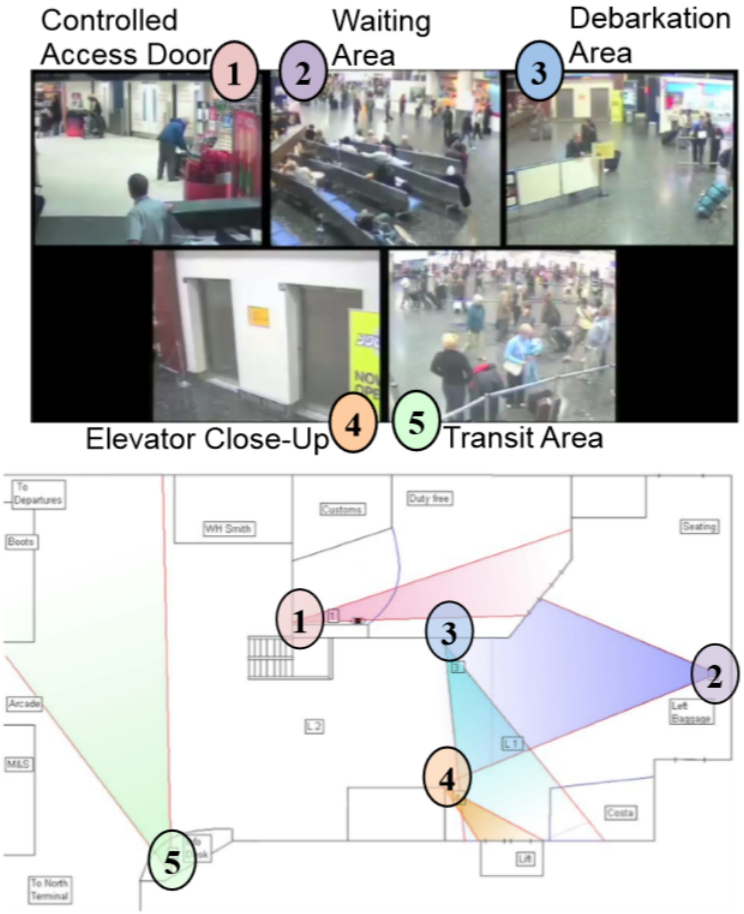

Figure 5.5: The spatial locations of each camera, along with a sample frame from each camera, from the Gatwick dataset provided as part of TRECVID [35].

that specific feature. With that in mind, we pose the event detection problem as the problem of finding large local maxima in the SVM response space.

Each bag of words feature is encoded as a normalized histogram. To classify this histogram, we use a support vector machine with the *histogram intersection* kernel.

Each of these SVM responses gives a floating point value corresponding to how likely the feature belongs to the *PersonRuns* event. The set of all bag of words features over an interval gives a distribution with several peaks and valleys, visualized in Figure 5.6. We are interested in SVM responses that are much higher than the local neighbourhood of that specific response. We begin with a single pass over each response and extract each response x_i where $x_i > \tau$, for some threshold τ, $x_i > x_{i-1}$ and $x_i > x_{i+1}$. This leaves us with a small number of candidate events, the areas with a direct local peak.

With the subset of candidate events extracted, we finalize the *PersonRuns* decision by centering a window around each candidate point i with SVM response x_i and scanning the distribution in that window to test whether x_i is a window-wise maximum. If it is, then the *PersonRuns* event has occurred at location i. The size of the window is relative to the assumed length of a *PersonRuns* event. In experiments, the *PersonRuns* event is defined as a 50 frame event, so the window would be for 25 frames before and after i. Experiments were also completed defining the event as 25 frames in duration, rather than 50.

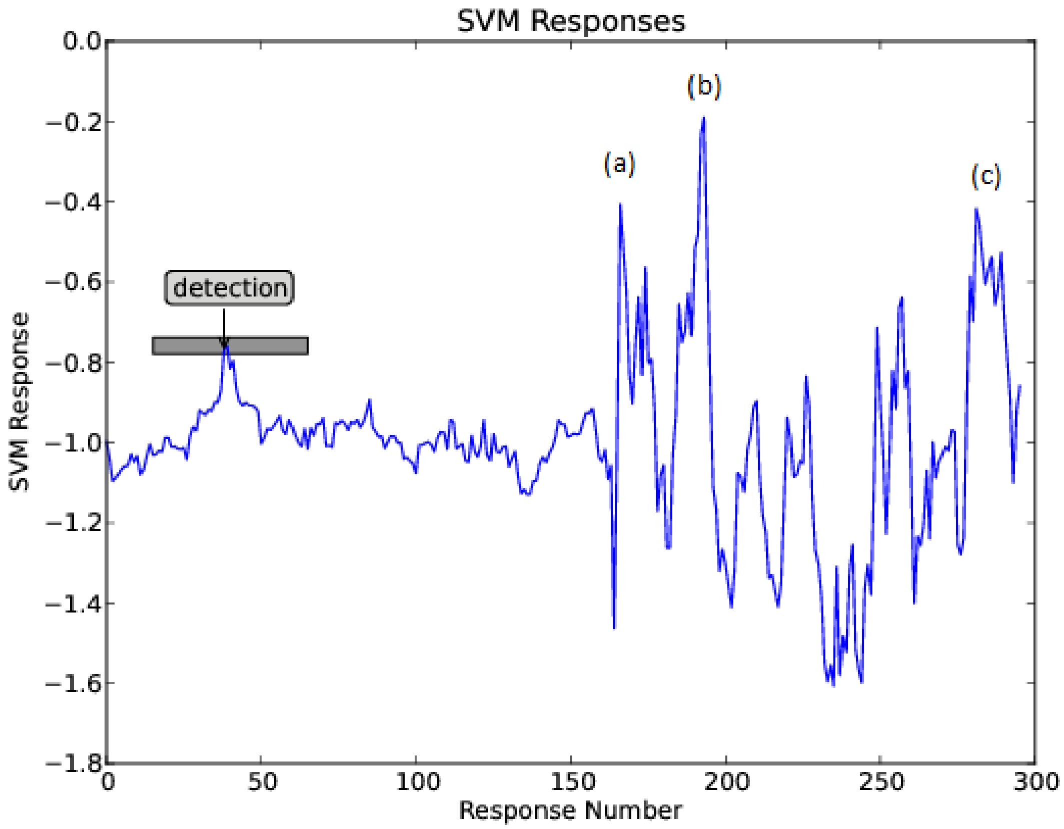

Figure 5.6: We test whether a peak in the SVM response space is a detected event by checking the window around that peak for local maxima with a greater SVM response. In this case (the gray box), there are no greater peaks; the point would be detected as a *PersonRuns* event. Peak (a) would not be a detection, since there is a larger peak (b) in its local vicinity. In contrast, peak (b) would be a detection, since there are no larger nearby peaks. The decision on peak (c) would need more information (from later frames) to be completed.

Chapter 6

MoFREAK: Experimental Results

6.1 Experimental Results

Performance of the MoFREAK descriptor has been evaluated on two standard benchmark datasets for action recognition. KTH [41] is a simple, single actor dataset that has been benchmarked by several approaches. In contrast, HMDB51 [23] is a recent dataset that is among the most difficult datasets due to its varying complexities. Furthermore, we compare the running times required by alternative approaches to process KTH, showcasing the significant computational advantage of MoFREAK.

In the context of event detection, the performance of an early version of the MoFREAK descriptor was evaluated for the TRECVID 2012 Surveillance Event Detection task [35]. With a much less sophisticated classification scheme than other submissions to TRECVID, the descriptor still performed extremely well, ranking among the top methods in this year's submissions. The Gatwick dataset is used for evaluation, which is very different than the action recognition datasets that MoFREAK has been benchmarked against. A great deal of the data deals with dense, crowded scenes in an airport surveillance setting. This allows for evaluation of its robustness to crowded scenes, and its applicability in the context of event detection rather than action recognition.

All experiments were conducted on standard consumer hardware with unoptimized code. The experiments were run on Windows 7 with an Intel Core i7 870 2.93 GHz CPU. No GPUs or other hardware optimizations were used, and the C++ code is not highly optimized.

	box	clap	wave	jog	run	walk
box	97	0	2	0	0	1
clap	5	95	0	0	0	0
wave	7	3	90	0	0	0
jog	0	0	0	79	9	12
run	0	0	0	11	83	6
walk	1	0	0	3	0	96

Table 6.1: KTH Confusion matrix

Approach	**Accuracy**
Schüldt et al. [41]	71 %
Dollár et al. [14]	81 %
Laptev et al. [25]	91 %
MoSIFT [6]	87 %
Kovashka & Grauman [22]	95 %
Kliper-Gross et al. [21]	93 %
MoFREAK	90 %

Table 6.2: Recognition results on the KTH dataset [41] for similar spatio-temporal feature-based approaches. MoFREAK retains competitive accuracy with similar state of the art methods, despite the significant increase in computational efficiency.

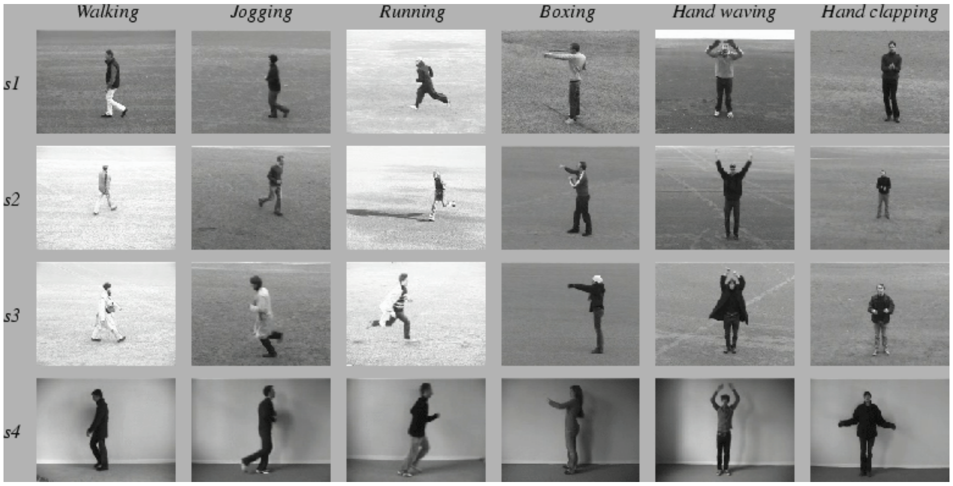

Figure 6.1: Sample frames from each KTH action. [41]

6.1.1 Action Recognition Experiments

KTH dataset: The KTH dataset [41] consists of 599 video sequences, each of which contains a single actor performing one of six actions (hand waving, clapping, boxing, walking, jogging, and running). Sample frames from each of the KTH actions are viewable in Figure 6.1. 25 separate actors are used, each performing all six actions under different conditions, such as camera scaling, outdoors versus indoors scenes, and actors changing clothing. Each sequence is low resolution (160×120), and the average sequence length is 20 seconds. The dataset has been well benchmarked and is approaching being “solved”, with algorithms achieving and surpassing 90 % accuracy in recent years.

MoFREAK features are detected and computed on every frame in each sequence, beginning at the sixth frame to allow computation of the absolute difference image. Then, the entire sequence is represented by a bag-of-words model with 600 randomly selected clusters, 100 clusters for each action class. Each video yields only a single bag-of-words feature, which is then classified by an SVM. We evaluate the performance of MoFREAK on this dataset through leave-one-out cross validation. For each of the 25 actors, we remove one actor for testing and train on the remaining 24 actors, computing the recognition performance on that actor. We then average the results across all possible actors to achieve our final reported accuracy.

The confusion matrix for classification on KTH is displayed in Table 6.1. The most significant areas of confusion are between walking, jogging, and running. Specifically, our system appears to occasionally confuse jogging with these two similar actions (walking and running).

To evaluate the effect of the MoFREAK descriptor, we compare its performance to similar methods which employ spatio-temporal features with a bag-of-words model in Table 6.2. We find that MoFREAK has comparable performance to the state of the art for comparable methods. For instance, we perform better than MoSIFT in accuracy while being significantly more efficient, which we expand upon in Section 6.1.3.

HMDB51 dataset: The HMDB51 dataset [23] is a significantly more difficult dataset than KTH [41]. HMDB51 consists of 51 actions, each of which have at least 101 video representations, with a total of $6,766$ video clips to recognize. A sample frame from each action is viewable in Figure 6.2. Due to the numerous sources that the videos are sampled from, such as Hollywood movies and YouTube clips, the quality varies greatly from video to video. The distribution of clip quality for each action can be seen in Figure 6.3. This dataset is currently among the most difficult datasets for benchmarking action recognition, with most algorithms achieving recognition accuracy in the neighbourhood of 20 %. Its difficulty comes from many separate sources. As mentioned, the video quality varies greatly. Furthermore, the videos have large variations in camera motion (Figure 6.4). This is likely one of the main causes for the disparities between the results between MoFREAK and other algorithms on this dataset, as MoFREAK employs no scheme for camera motion stabilization or correction. Other sources of difficulties include wide variations in camera viewpoint (Figure 6.5), scale, cluttered backgrounds, and illumination conditions.

To evaluate our descriptor on this dataset, the $70/30$ training/testing split presented in [23] is followed. This evaluation methodology selects 70 videos for training and 30 videos for testing from each of the 51 actions. The videos were hand-selected to contain videos of varying quality, environments, and camera motion. When computing features, we use a codebook of 5100 randomly selected features, 100 from each action, for the bag of words representation.

A summary of reported results are presented in Table 6.3. These results are all reported on the more difficult, original HMDB51 dataset, rather than the stabilized alternative. The current state of the art is achieved with Motion Interchange Patterns [21], which includes a motion stabilization component within the algorithm. We note that, while we do not have greater accuracy than the remaining approaches, we remain close while achieving significant performance speedups, as discussed in Section 6.1.3.

6.1.2 Event Detection Experiments

All training and event detection was done on the Gatwick dataset, surveillance footage captured at the London Gatwick airport recorded with MPEG2 compression and a resolution of 720x480

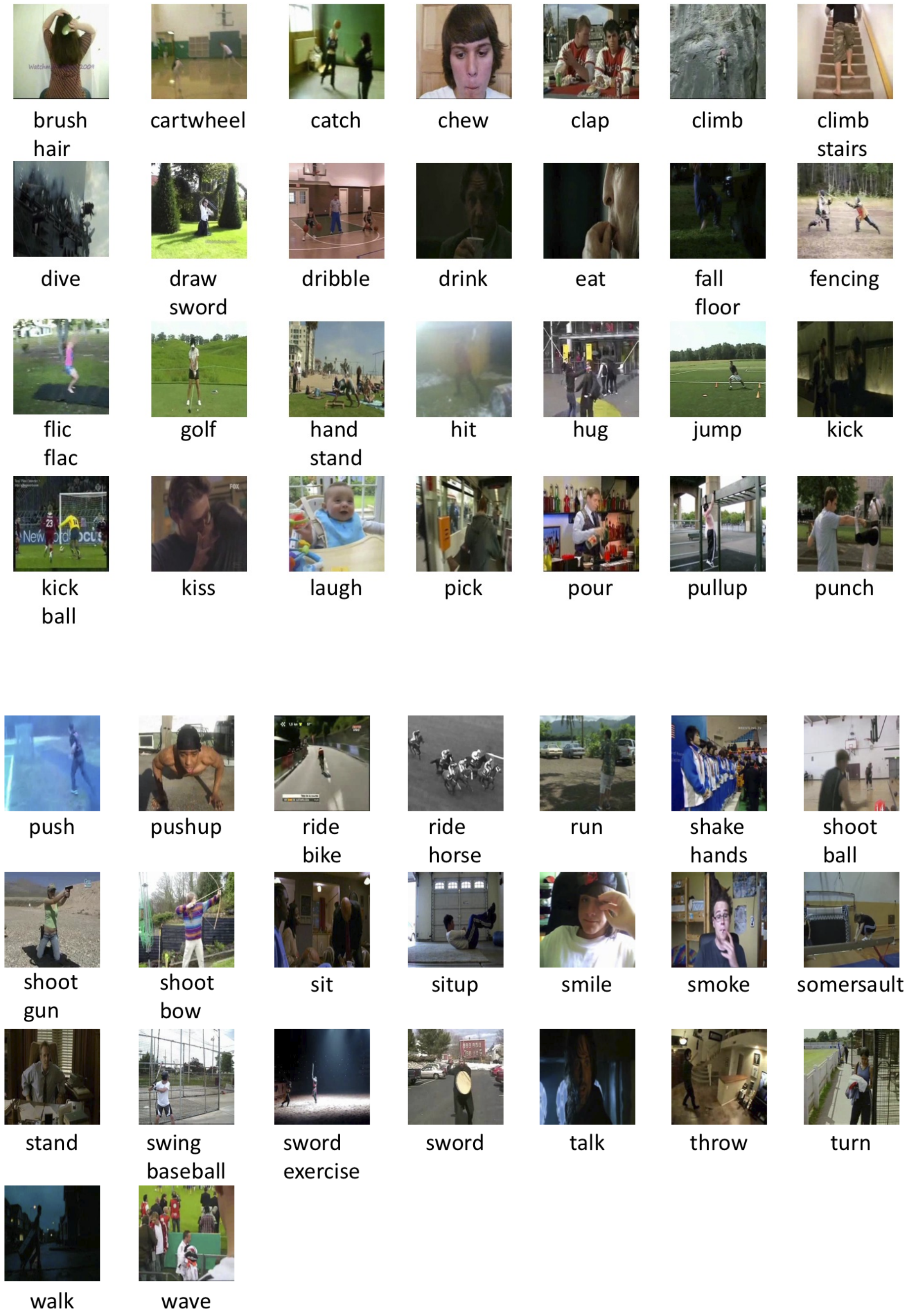

Figure 6.2: Sample frames from each HMDB51 action. [23]

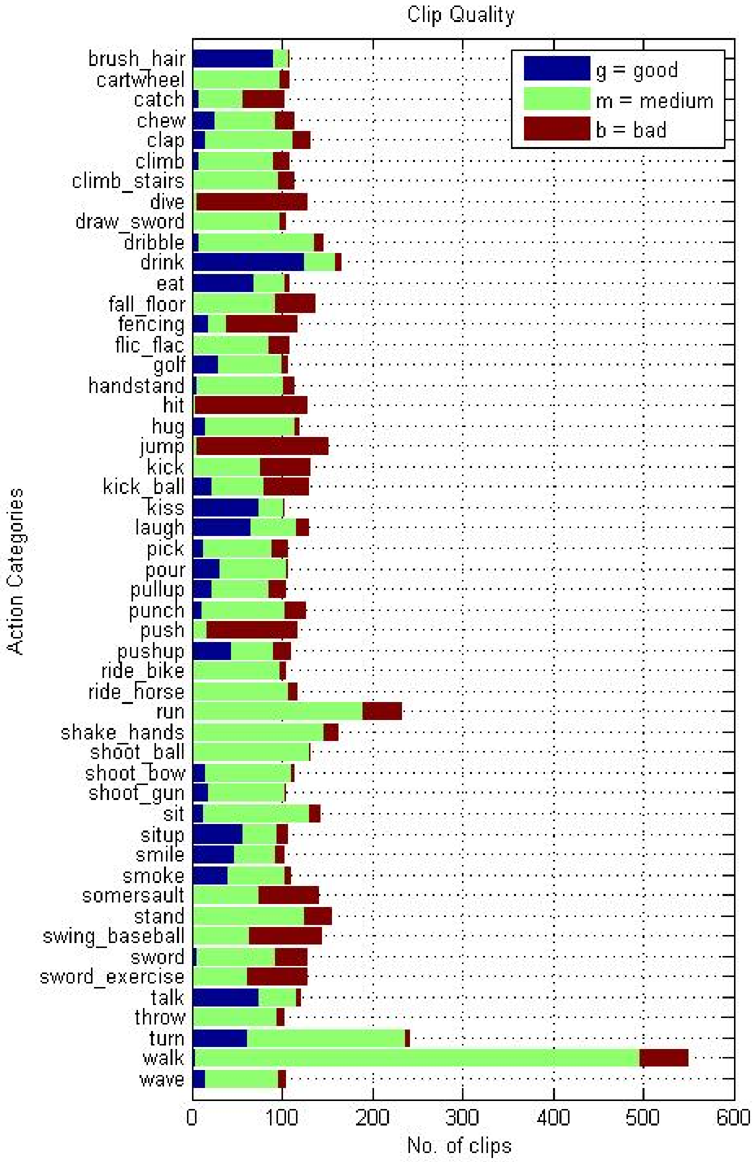

Figure 6.3: Distribution of video quality for each HMDB51 action. [23]

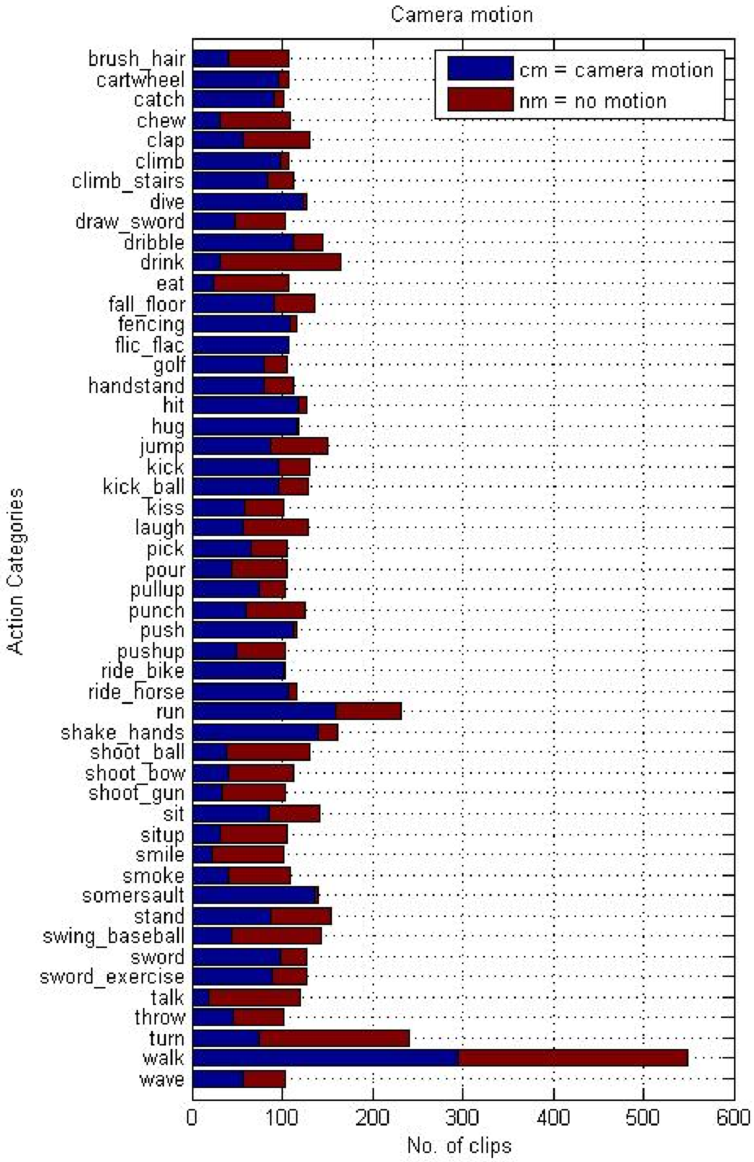

Figure 6.4: Distribution of camera motion for each HMDB51 action. [23]

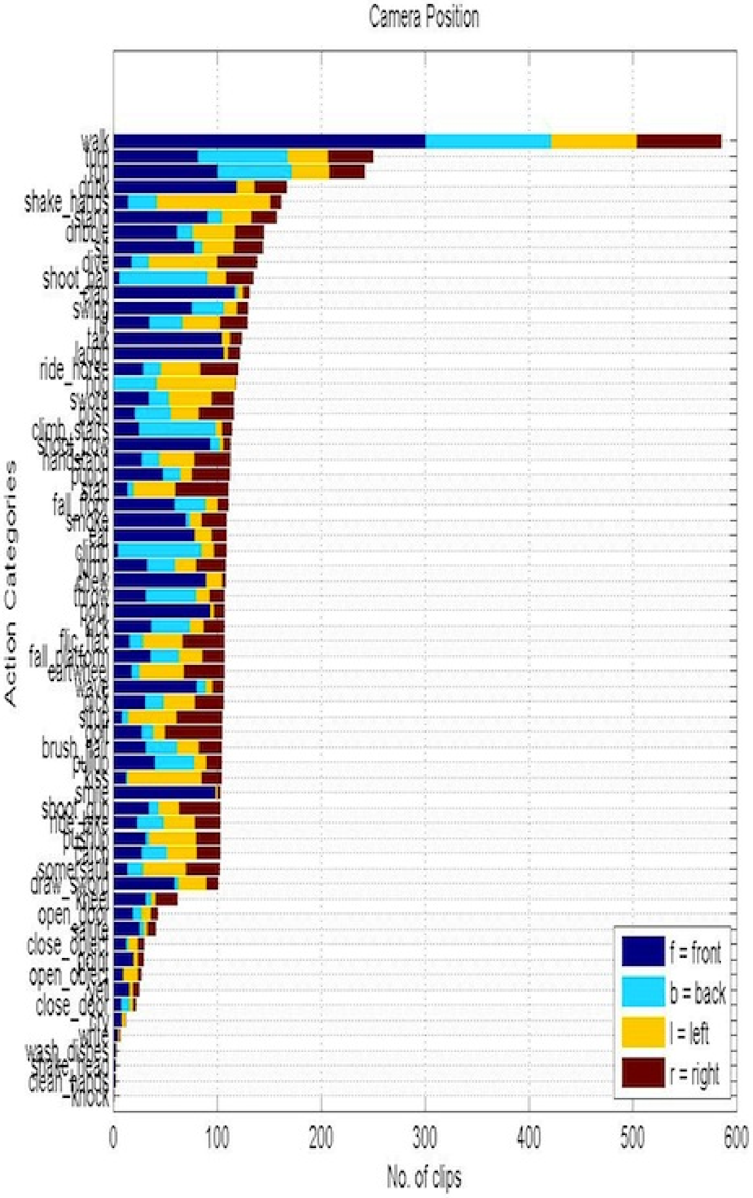

Figure 6.5: Distribution of camera viewpoint for each HMDB51 action. [23]

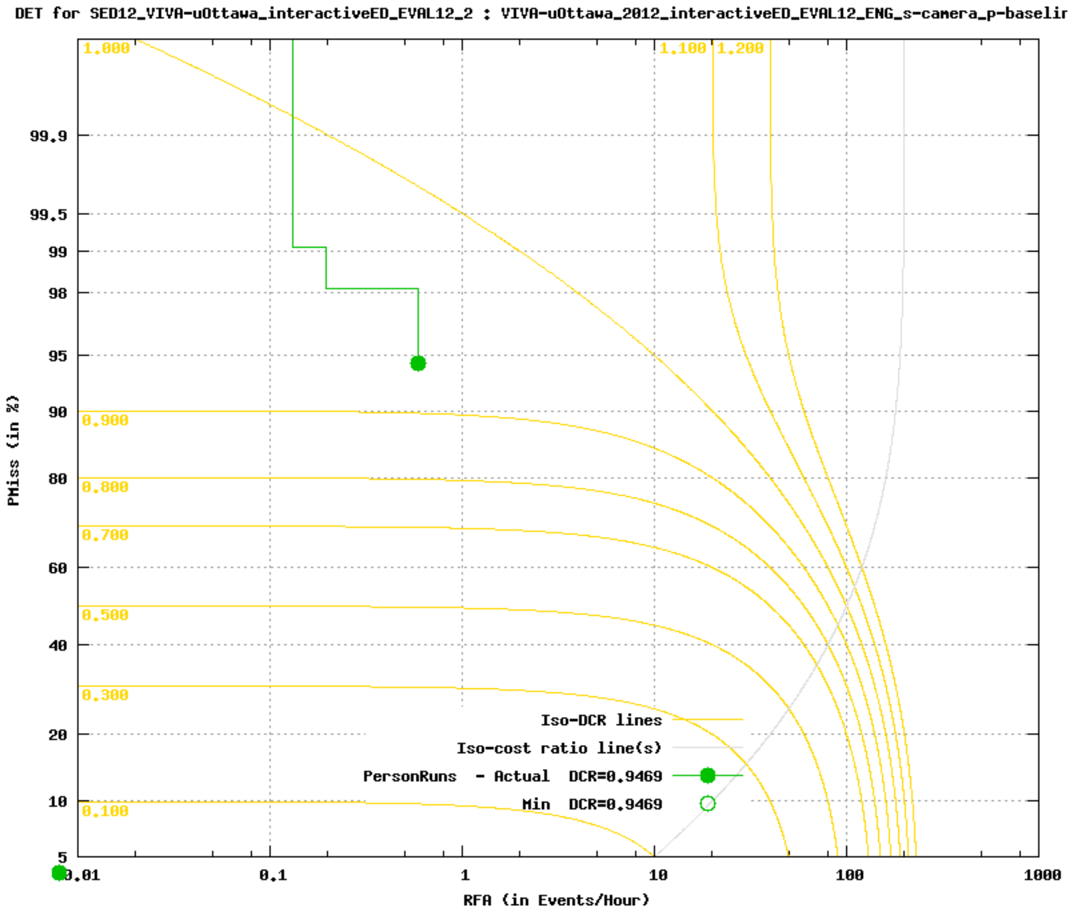

Figure 6.6: DET curve for our submission for the *PersonRuns* event on the Interactive Surveillance Event Detection task. This curve was generated and provided by TRECVID [35] based on our results. The DET curve plots the missed detection percentage against the rate of false alarms. The ideal location on this graph is at the very bottom-left, with 0 missed detections and 0 false alarms. The closer to the bottom of the graph, the more true positives are found. In contrast, being further left on the graph implies less false positives. These are both desirable results.

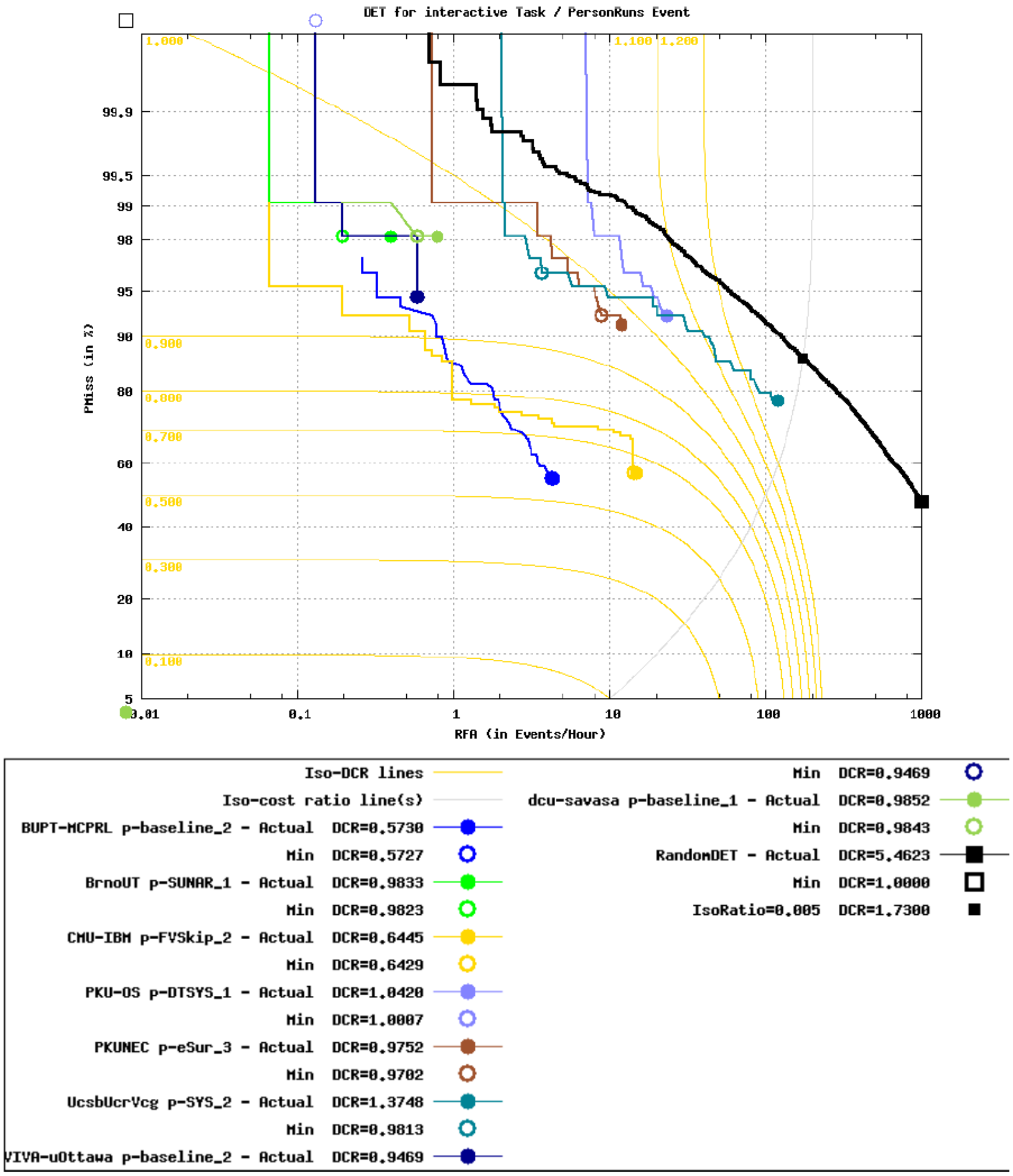

Figure 6.7: DET curve for our submission for the *PersonRuns* event on the interactive SED task. The DET curve plots the missed detection probability against the rate of false alarms. Our result is the darkest blue curve, as shown in the legend (VIVA-uOttawa p-baseline_2). The actual DCR value is equivalent to the minimum DCR value, so only one node is visible due to the minimum and actual DCR values overlapping. This curve was generated by the TRECVID [35] organizers based on the submissions by all teams.

Approach	**Accuracy**
Laptev et al. [25]	20.4 %
Jhuang et al. [17]	22.8 %
Sadanand & Corso [39]	26.9 %
Kliper-Gross et al. [21]	29.1 %
MoFREAK	18.4 %

Table 6.3: Recognition comparison to previous results on the difficult HMDB51 dataset [23].

Title	#Targ	#NTarg	#Sys	#CorDet	#FA	#Miss	RFA	PMiss	DCR
PersonRuns	107	9	15	6	9	101	0.59027	0.944	0.9469

Figure 6.8: Details of our submission for the *PersonRuns* event on the interactive SED task. There were 107 true events (#Targ), while we detected 15 (#Sys). 6 of those 15 events were deemed to be true events (#CorDet), while 9 were deemed false alarms (#FA), giving us 101 missed detections (#Miss). Our rate of false alarms is 0.59027 (RFA), and our percentage of missed detections is 0.944 (PMiss). The weighted linear combination of the false alarm rate and probability of a missed detection is 0.9469 (DCR).

at 25 frames per second. As described in Chapter 5, the data comes from 5 separate cameras with overlapping but different viewpoints. Each camera is broken up into several hours of video across multiple days, which ensures that systems do not overtrain on specific persons and allows for different variations on the same event. Training was performed on selected portions of 100 hours of the development video sequences. Event detection was performed on a final 15 hours of test data.

The interactive surveillance event detection task allows for a user to spend 25 minutes with a visual analytics tool to select true positive events from the automated event detection process. This places an emphasis on the SVM's ability to correctly assign a likelihood to each returned detection, so the most likely events are ranked with the highest likelihood. Within this 25 minute time-frame, our test user selected 15 "true positive" events for the *PersonRuns* task. As described in Figure 6.8, nine of these events selected by the user were deemed to be false positives, while six of them were deemed to be true positives. Due to the nature of the underlying event detection system, all events processed were of two fixed lengths as processed, 50 frames or 25 frames. As such, it is possible that some of the false positives were a result of the intersection between the selected event and the ground truth being too small to be considered a true detection. A possible remedy for such a situation would be to add sliders to the visual analytics tool to modify the start and end frames of each event. While this would slow down the human processing speed, it would increase the precision of each detection, allowing us to accurately capture the entire event and less spurious data.

The DET curve for the returned results is displayed in Figure 6.6. Our returned DCR score is 0.9469. A DCR score is a weighted linear combination of the missed detection probability (the number of missed detections divided by the number of events) and the rate of false alarms. A DCR score of 0 is indicative of a perfect detection system, so lower DCR scores are deemed to be representative of a more accurate event detection system, with respect to false alarms and detected events. The DCR score balances the missed detections and the rate of false alarms, giving the optimal point on the DET curve [45]. Formally, DCR is defined as

$$DCR = P_{Miss} + \beta * R_{FA} \tag{6.1}$$

where $\beta = \frac{Cost_{FA}}{Cost_{Miss} * R_{Target}}$, $Cost_{Miss}$ is a constant for the cost of a missed detection, $Cost_{FA}$ is a constant for the cost of a false alarm, and R_{Target} is a constant for the a priori rate of event observations [35].

Our submission is placed in the context of the other submissions to TRECVID in Figure 6.7. In the context of other TRECVID teams, two groups achieved lower DCR scores, while five teams have higher DCR scores reported than our DCR of 0.9469. The mean DCR is

0.9406, lowered by the best DCR of 0.573, while the highest DCR was 1.3748. Our rate of false alarms (RFA) is 0.59027, the second lowest in this year's TRECVID task. The average RFA is 22.0040075, significantly higher than our own. The disparity in RFA scores indicates that there are multiple methodologies that were used to determine how to prune the automated detection results. The largest RFA value is 119.82510, indicating that rather than manually filtering events sequentially, an approach of tuning parameters to optimize on a subset of the results may have been taken. Conversely, the lowest RFA score is 0.39351. That score corresponds to a submission with very few (8) detected events, likely indicating that a similar approach to our own was taken, through manual sequential filtering.

6.1.3 Computational Cost

One of the main focal points of our approach is ensuring a low computational cost, making it feasible to quickly process large datasets. Action Bank [39] reports extremely high accuracy on several datasets, but its applicability is limited, due to the overhead involved in building its action detectors and the amount of time required to process each video. Although the time required to build the detectors is not reported, processing a single UCF50 video requires, on average, 204 minutes. Processing a comparable video with our unoptimized implementation of MoFREAK requires approximately 1 minute, a $200\times$ speed increase over Action Bank.

The computational advantage to MoFREAK is two-fold, as a result of the representation of the descriptor. In contrast with descriptors represented as complex floating point values, representing gradients or detection responses, MoFREAK is represented as a binary string, representing simple intensity comparisons between pixels. While this is much quicker to compute than detector responses or gradients, it is also much quicker to compare descriptors under this representation, as the distance metric (Hamming distance) is simple and can be highly optimized by modern CPUs.

To showcase the performance gains of MoFREAK, we compare the running time of MoFREAK against similar spatio-temporal descriptor approaches with reported running times, outlined in Table 6.4. We have also run an implementation of MoSIFT [6] for our comparisons. Fathi and Mori [16] report a computationally efficient method in which classification takes between 0.2 and 4 seconds per frame on KTH [41]. Using the midpoint of 2.1 for computation on the $11,576$ seconds of KTH footage at 25 frames per second, this adds up to over $10,000$ minutes to process this small, artificial dataset. Assuming a best-case scenario of 0.2 seconds for each frame, the method still takes 965 minutes to process. Jhuang *et al.* [17] report that a typical run of their approach takes over 2 minutes per video sequence, processing only 50

Approach	**Running Time (mins)**	Accuracy
Fathi and Mori [16]	10,129	90%
Jhuang et al. [17]	1,198	90 %
MoSIFT [6]	449	87 %
MoFREAK	185	90 %

Table 6.4: Comparing the running times of processing KTH [41]. MoFREAK and MoSIFT [6] were computed with our C++ implementation, while the remaining approaches are reported results.

frames per sequence, summing to over 1198 minutes for KTH, without even processing every frame. In contrast, processing the entirety of KTH is accomplished in 185 minutes with MoFREAK, which is a shorter time period than the actual length of the KTH dataset. It is interesting to note that no other state of the art approaches are applicable for real-time action recognition, even on the low-resolution artificial sequences of KTH.

Chapter 7

Conclusion

7.1 Conclusion

This thesis has presented two main contributions: an approach for shape parsing and matching, and a new binary spatio-temporal feature descriptor for action recognition in video sequences.

We have proposed a model for parsing shapes that allows for different parameterizations and measures of shape similarity, based on arbitrary representations of shape parts. The main requirement for instantiating the model is to give a parameterization such that the desired parses of a shape are within its k most probable parses, which is a benefit granted by our proposal of keeping not only the single most probable parse of a shape, but the k most probable ones. Experiments on a challenging dataset showed that we are able to achieve 95% accuracy based on a simple parameterization. We expect that more sophisticated models for selecting shape cuts will yield even better performance.

We have also presented a compact, 10-byte binary spatio-temporal keypoint descriptor, MoFREAK, which simultaneously encodes the local appearance and the geometric structure of the motion of a local space-time neighborhood. The descriptor takes advantage of recent advances in binary feature descriptors to avoid costly gradient computations that are typically present in space-time descriptors.

We have shown that this approach has significant computational advantages, being faster than the state of the art by several orders of magnitude, while retaining competitive recognition accuracy. On the KTH dataset [41], we achieved 90 % recognition accuracy, while being several times faster than previous methods.

Throughout this paper, our focus has been on the descriptor rather than the overall recognition system, leading us to use a very simple bag-of-words representation with a simple SVM classifier. A more extensive classification setup may lead to recognition accuracy greater than what we have reported, and is an area of future work.

While the presented approaches both tackle the problem of recognition, they are clearly quite distinct, with complimentary advantages and disadvantages. One of the main areas of concern for bag-of-words action recognition models is that they are inherently global, capturing little-to-no specific local information. In these approaches, any specificities gained by more descriptive local feature descriptors is eventually lost through quantization in the bag-of-words model. Nonetheless, this often becomes an advantage instead of a disadvantage. One example of it being advantageous is when over-specificity causes instances that should be classified as the same class to be incorrectly classified to different categories. Such a scenario is succinctly described by the adage of "missing the forest for the trees". The bag-of-words approach allows for more generality in classifiers, leading to resistance against perturbations in similar instances. In contrast, the presented approach to shape parsing takes a global representation of shape and decomposes it to find similarities to previously seen instances on a local scale. By taking this local decomposition approach, we can detect subtle nuances that differentiate instances in classes with a great deal of similarity. When scale normalized, it is often very difficult for algorithms to differentiate between shapes of dogs and horses when performing evaluations at a global level. It is then helpful to search for a lower-level representation where smaller, more local properties can be compared. For example, while the torso is similar between horses and dogs, the shape of their heads have subtle differences which can be captured in a local representation. These core differences between global and local approaches are among the main factors that drive algorithm selection for a given application domain. When the core differentiators between classes are subtle, it is often a necessity to follow a local representation. However, when intraclass variation is high and there are no obvious local indicators, it may be advantageous or necessary to follow a global approach.

7.1.1 Future Work

Shape Parsing

There is a significant amount of room for improvement with respect to computational efficiency. Currently at query time, k matches must be computed against every SPG saved in the database. It would be beneficial if SPGs could be embedded into an alternative space where efficient nearest-neighbour computations could be performed. Locally linear embeddings [38] compute a low-dimensional embedding of high-dimensional data points that preserves neighbourhoods. Alternatively, BoostMap [2] performs a similar embedding by combining a series of simple one-dimensional embeddings with AdaBoost [40]. This preserves a significant amount of the proximity relationships between SPGs, and would greatly increase query efficiency.

MoFREAK

Throughout this work, the focus has been on the descriptor rather than the overall recognition system, leading to the use of a very simple bag-of-words representation with a simple SVM classifier. A more extensive classification setup may lead to greater recognition accuracy than what we have reported.

One such extension would be the replacement of the bag-of-words model with Fisher Vectors [36]. Fisher Vectors replace the traditional codebook encoding of videos with a Gaussian mixture model. Then, rather than the hard-assignments present in the bag-of-words model, the set of spatio-temporal features extracted from a video sequence are softly assigned to each of the Gaussian components within the GMM. Recent work [9] has demonstrated that the use of Fisher Vectors over a bag-of-words model increases both computational efficiency and recognition accuracy.

References

[1] Alexandre Alahi, Raphal Ortiz, and Pierre Vandergheynst. FREAK: Fast Retina Keypoint. In *CVPR*, 2012.

[2] Vassilis Athitsos, Jonathan Alon, Stan Sclaroff, and George Kollios. Boostmap: An embedding method for efficient nearest neighbor retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(1):89–104, 2008.

[3] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Neural Information Processing Systems Conference (NIPS)*, pages 831–837, December 2000.

[4] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, 2002.

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[6] Ming-Yu Chen and Alex Hauptmann. Mosift: Recognizing human actions in surveillance videos. Technical Report CMU-CS-09-161, Carnegie Mellon University, 2009.

[7] Ming-Yu Chen, Huan Li, and Alex Hauptmann. Informedia at trecvid 2009: Analyzing video motions. In *TRECVID*.

[8] Dmitry Chetverikov and Zsolt Szabo. A simple and efficient algorithm for detection of high curvature points in planar curves. In *Workshop of the Austrian Pattern Recognition Group*, pages 175–184, 1999.

[9] Yang Cia, Qiang Chen, Lisa Brown, Ankur Datta, Quanfu Fan, Rogerio Feris, Shuicheng Yan, Alex Hauptmann, and Sharath Pankanti. Cmu-ibm-nus at trecvid 2012: Surveillance event detection. In *Proceedings of TRECVID*, 2012.

[10] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[11] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cdric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[12] Konstantinos G. Derpanis, Mikhail Sizintsev, Kevin J. Cannons, and Richard P. Wildes. Efficient action spotting based on a spacetime oriented structure representation. In *CVPR*, pages 1990–1997, 2010.

[13] Pavel Dimitrov, Carlos Phillips, and Kaleem Siddiqi. Robust and efficient skeletal graphs. In *CVPR*, pages 1417–1423, 2000.

[14] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, 2005.

[15] Alexei A. Efros, Alexander C. Berg, Er C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *ICCV*, 2003.

[16] Alireza Fathi and Greg Mori. Action recognition by learning mid-level motion features. In *CVPR*, 2008.

[17] Hueihan Jhuang, Thomas Serre, Lior Wolf, and Tomaso Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007.

[18] Ralf Juengling and Lakshman Prasad. Parsing silhouettes without boundary curvature. In *ICIAP*, pages 665–670, 2007.

[19] Frederic Jurie and Bill Triggs. Creating efficient codebooks for visual recognition. In *ICCV*, 2005.

[20] Hae-Kwang Kim and Jong-Deuk Kim. Region-based shape descriptor invariant to rotation, scale and translation. *Sig. Proc.: Image Comm.*, 16(1-2):87–93, 2000.

[21] Orit Kliper-Gross, Yaron Gurovich, Tal Hassner, and Lior Wolf. Motion interchange patterns for action recognition in unconstrained videos. In *ECCV*, 2012.

[22] Adriana Kovashka and Kristen Grauman. Learning a hierarchy of discriminative spacetime neighborhood features for human action recognition. In *CVPR*, 2010.

[23] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: A large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011.

[24] Ivan Laptev and Tony Lindeberg. Space-time interest points. In *ICCV*, 2003.

[25] Ivan Laptev, Marcin Marszaek, Cordelia Schmid, Benjamin Rozenfeld, Inria Rennes, Irisa Inria Grenoble, and Lear Ljk. B.: Learning realistic human actions from movies. In *CVPR*, 2008.

[26] Stefan Leutenegger, Margarita Chli, and Roland Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[27] Li-Jia Li, Hao Su, Eric P. Xing, and Fei-Fei Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, pages 1378–1386, 2010.

[28] Haibin Ling and David W. Jacobs. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Mach. Intell*, 29:286–299, 2007.

[29] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[30] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. 1981.

[31] Diego Macrini, Sven J. Dickinson, David J. Fleet, and Kaleem Siddiqi. Object categorization using bone graphs. *Computer Vision and Image Understanding*, 115(8):1187–1206, 2011.

[32] Diego Macrini, Chris Whiten, Robert Laganière, and Michael Greenspan. Probabilistic shape parsing for view-based object recognition. In *International Conference on Pattern Recognition (ICPR)*, 2012.

[33] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV*, 2009.

[34] Xiaofeng Mi and Doug DeCarlo. Separating parts from 2d shapes using relatability. In *ICCV*, 2007.

[35] Paul Over, George Awad, Martial Michel, Jonathan Fiscus, Greg Sanders, Barbara Shaw, Wessel Kraaij, Alan F. Smeaton, and Georges Quenot. Trecvid 2012 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID 2012*. NIST, USA, 2012.

[36] Florent Perronnin and Christopher R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[37] Mikel D. Rodriguez, Javed Ahmed, and Mubarak Shah. Action mach a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.

[38] Sam T. Roweis and Lawrence K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *SCIENCE*, 290:2323–2326, 2000.

[39] Sreemananth Sadanand and Jason J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, pages 1234–1241, 2012.

[40] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Machine Learning*, 1999.

[41] Christian Schüldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.

[42] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. Shock-based indexing into large shape databases. pages 731–746, 2002.

[43] Eli Shechtman and Michal Irani. Matching local self-similarities across images and videos. In *CVPR*, 2007.

[44] Manish Singh, Gregory Seyranian, and Donald Hoffman. Parsing silhouettes: The short-cut rule. *Perception and Psychophysics*, 61(4):636–660, 1999.

[45] Alan F. Smeaton, Paul Over, and Wessel Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[46] Hugo Steinhaus. Sur la division des corp materiels en parties. *Bull. Acad. Polon. Sci*, 1:801–804, 1956.

[47] Xinghua Sun, Ming-Yu Chen, and Alex Hauptmann. Action Recognition via Local Descriptors and Holistic Features. In *CVPR*, 2009.

[48] Heng Wang, Alexander Kläser, Cordelia Schmid, and Liu Cheng-Lin. Action Recognition by Dense Trajectories. In *CVPR*, 2011.

[49] Chris Whiten, Robert Laganière, and Guillaume-Alexandre Bilodeau. Efficient action recognition with mofreak. In *Tenth Conference on Computer and Robot Vision*, 2013.

[50] Chris Whiten, Robert Laganière, Ehsan Fazl-Ersi, Feng Shi, Guillaume-Alexandre Bilodeau, Dmitry Gorodnichy, Jean-Philippe Bergeron, and Ehren Choy. Viva-uottawa / cbsa at trecvid 2012: Interactive surveillance event detection. In *Proceedings of TRECVID*, 2012.