4-2016

Learning in vision and robotics

Daniel P. Barrett
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Daniel Barrett

Entitled: Learning in Vision and Robotics

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

JEFFREY SISKIND

ROBERT L. GIVAN

T. N. VIJAYKUMAR

THOMAS M. TALAVAGE

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

JEFFREY SISKIND

Approved by Major Professor(s): _____

_____

Approved by: V. Balakrishnan 04/15/2016

Head of the Department Graduate Program Date

LEARNING IN VISION AND ROBOTICS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Daniel P. Barrett

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2016

Purdue University

West Lafayette, Indiana

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Figure Page

ABSTRACT

Barrett, Daniel P. PhD, Purdue University, May 2016. Learning in Vision and Robotics. Major Professor: Jeffrey M. Siskind.

I present my work on learning from video and robotic input. This is an important problem, with numerous potential applications. The use of machine learning makes it possible to obtain models which can handle noise and variation without explicitly programming them. It also raises the possibility of robots which can interact more seamlessly with humans rather than only exhibiting hard-coded behaviors. I will present my work in two areas: video action recognition, and robot navigation. First, I present a video action recognition method which represents actions in video by sequences of retinotopic appearance and motion detectors, learns such models automatically from training data, and allow actions in new video to be recognized and localized completely automatically. Second, I present a new method which allows a mobile robot to learn word meanings from a combination of robot sensor measurements and sentential descriptions corresponding to a set of robotically driven paths. These word meanings support automatic driving from sentential input, and generation of sentential description of new paths. Finally, I also present work on a new action recognition dataset, and comparisons of the performance of recent methods on this dataset and others.

1. INTRODUCTION

I present my work on machine learning from video and robotic input. This work has resulted in several published papers and several others which are still in review. Specifically, I am first author on two accepted IEEE journal papers and on two journal papers currently in review, as well as coauthor on two accepted conference papers and one currently in review, and coauthor on three technical reports. The journal papers are:

1. Daniel P. Barrett and Jeffrey M. Siskind, "Action Recognition by Time-Series of Retinotopic Appearance and Motion Features," in IEEE Transactions on Circuits and Systems for Video Technology, 2015.

2. Daniel P. Barrett, Andrei Barbu, N. Siddharth, and Jeffrey M. Siskind, "Saying What You're Looking For: Linguistics Meets Video Search," in IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.

3. Daniel P. Barrett, Scott A. Bronikowski, Haonan Yu, and Jeffrey M. Siskind, "Driving Under the Influence (Of Language)," *manuscript in review.*

4. Daniel P. Barrett, Ran Xu, Haonan Yu, and Jeffrey Mark Siskind, "Collecting and Annotating the Large Continuous Action Dataset," *manuscript in review.*

The conference papers are:

1. Andrei Barbu, Daniel P. Barrett, Wei Chen, Narayanaswamy Siddharth, Caiming Xiong, Jason J, Corso, Christiane D. Fellbaum, Catherine Hanson, Stephen Jose Hanson, Sebastien Helie, Evguenia Malaia, Barak A. Pearlmutter, Jeffrey Mark Siskind, Thomas Michael Talavage, and Ronnie B Wilbur, "Seeing is Worse than believing: Reading People's Minds Better than Computer-Vision

Methods Recognize Actions," in the Proceedings of the European Computer Vision Conference (ECCV), 2014.

2. Yu Cao, Daniel Barrett, Andrei Barbu, Siddharth Narayanaswamy, Haonan Yu, Aaron Michaux, Yuwei Lin, Sven Dickenson, Jeffrey Mark Siskind, Song Wang, "Recognize Human Activities from Partially Observed Videos," in the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

3. Scott A. Bronikowski, Daniel P. Barrett, and Jeffrey M. Siskind, "Object Codetection from Mobile Robot Video," *in review.*

The technical reports are:

1. Daniel P. Barrett, Scott A. Bronikowski, Haonan Yu, and Jeffrey M. Siskind, "Robot Language Learning, Generation, and Comprehension," on Arxiv, 2015.

2. Haonan Yu, Daniel P. Barrett, and Jeffrey M. Siskind, "A Faster Method for Tracking and Scoring Videos Corresponding to Sentences," on Arxiv 2014

3. Daniel P. Barrett and Jeffrey M. Siskind, "Felzenszwalb-Baum-Welch: Event Detection through Changing Appearance," on Arxiv 2013

In all of these ten documents, my technical contribution was significant enough that the paper would not exist without it. In many of these, I am the first author, performed the majority of the work, and am responsible for the majority of the technical contribution.

The majority of my work consists of that described in the two journal papers: "Action Recognition by Time-Series of Retinotopic Appearance and Motion Features,", which constitutes Chapter 2, and "Driving Under the Influence (Of Language),' which constitutes Chapter 3. In the first, 100% of the technical contribution of the paper is solely my work: the learning algorithm, the action recognition method, the new dataset, and all code, experiments, figures, and text. In the second, the vast majority

of the technical contribution of the paper is my work, including the sentence parsing method, sentence and word representations, learning algorithm, sentence comprehension and planning algorithm, experiments, and the majority of the code, text and figures. The material from these two journal papers constitutes the bulk of this document, with the addition of a second journal paper currently in review "Collecting and Annotating the LCA dataset", (Chapter 4), for which I am also responsible for the majority of the technical contribution: the annotation of the new dataset and the experiments demonstrating its difficulty through the poor performance of state-of-the-art action recognition methods. The final two chapters include material from my contributions to other published papers, specifically ECCV 2014 (Chapter 5), for which I conducted all the experiments with video action recognition methods, and PAMI 2015 (Chapter 6), in which I conducted large scale video retrieval experiments and evaluation of results through Amazon Mechanical Turk.

2. ACTION RECOGNITION BY TIME-SERIES OF RETINOTOPIC APPEARANCE AND MOTION FEATURES

We present a method for recognizing and localizing actions in video by the sequence of changing appearance and motion of the participants. Appearance is modeled by histogram of oriented gradients (HOG) object detectors while motion is modeled by optical-flow motion-pattern detectors. Sequencing is modeled by a hidden Markov model (HMM) whose output models are these appearance and motion detectors. The HMM and associated detectors are simultaneously trained, learning the sequence of detectors that match the most distinctive temporal subsequences of the action represented in the training data. Training uses both positive and negative samples of a given action class and is accomplished without need for annotation of the correspondence between training-video frames and the state-conditioned detectors, by minimizing a discriminative cost function through gradient descent. Trained models are used to perform recognition and localization by simultaneous detection, tracking, and action-recognition. In contrast to many prior methods, our approach learns intuitively meaningful models that represent action as a sequence of retinotopic models. We demonstrate such by rendering these models on unseen test video. This method was found to perform competitively on three standard datasets, Weizmann, KTH and UCF Sports, as well as on video from the DARPA Mind's Eye program and a newly filmed dataset. Action recognition in video is a growing field, applicable to areas such as surveillance, robotics, and video retrieval. This field largely focuses on forced choice one-out-of-K classification of video clips, rather than binary classification or detection in long streaming videos, which can, in theory, be done by extension of a classification system. The most prominent datasets [1–6] reflect this focus. Every video clip in these datasets contains a single instance of an action; no clip contains multiple

actions or fails to depict any action. The most prominent current methods [1, 6–15] generally employ a bag of spatio-temporal visual-words approach (BOW). They generally extract feature vectors, such as spatio-temporal interest points (STIP) [4] or Dense Trajectories [13], at a subset of space-time points, build a codebook by pooling such, vector quantize such feature vectors on this codebook, compute a histogram of codebook-entry occurrences on the pooled frames of a video, and classify these histograms with temporally invariant models, such as support vector machines (SVM). However, such methods leave something to be desired, as they do not represent the coarse-grained temporally variant appearance and motion that is characteristic of an action class in a human discernible fashion, relying instead on aggregation of very fine-grained properties. Such methods can learn models for actions based upon low-level features which have correlation with the classes in the particular dataset used, but which are unrelated to the meaning of the action class that a human might understand. For example, they might associate *diving* with a blue Olympic swimming pool [16], or *basketball* and *volleyball* with a wooden gym floor [17]. The aggregated properties by which a video is classified, therefore, may not even have come from the part of the video in which the action takes place. Such methods therefore generally do not localize the recognized actions.

These methods are akin to distinguishing lasagna from spaghetti by blending them up and analyzing the ratios of elements with a mass spectrometer. Fundamentally, spaghetti and lasagna differ only in the shape of the pasta, but are usually associated with different sauces, resulting in correlations with different microscopic properties. Thus, such a blender approach would generally have high performance, but completely fail to distinguish lasagna and spaghetti without sauce or fail to distinguish pesto or Alfredo lasagna from spaghetti when trained on tomato lasagna and spaghetti, and would not be well suited to localizing individual pieces of pasta. Similarly, models for *diving* and *basketball* which rely on the backgrounds would have difficulty recognizing a person shooting a *basketball* in a swimming pool. Perhaps for this reason, many datasets focus on classes which might be better thought of as scenes than actions.

For example, the UCF50 [1] dataset depicts classes, such as *military parade* and *horse race*, which are not localizable atomic actions, but rather general scene categories.

One reason that much recent work focuses on BOW-style methods is precisely because they do not involve localization, which itself has proven to be quite difficult. Localization of the action participants generally involves the use of some kind of detector or tracker. However, detection and tracking are themselves unsolved problems, with no totally reliable solution. Background subtraction (*e.g.*, Lin *et al.* [18]) avoids explicit object detection, and aims to identify those pixels which are in the "foreground" by building a model of the background, and assigning areas that move or otherwise deviate from this model to the foreground. General tracking methods like that of Kalal *et al.* [19] take a bounding box as input, and attempt to track the entity bounded by that box through the rest of a video. However, none of these methods have a way to track the particular person performing an action and fail in the presence of occlusion or, in the case of background subtraction, in the presence of several moving objects. Any method which relies on such preprocessing steps to provide the locations of the action participants is subject to cascading failure when the detector or tracker fails to track the relevant participants. The localization performance therefore sets an upper bound on recognition performance.

However, the benefits of localizing the action participants in addition to recognizing the action class have been shown by the recent work of Yu & Siskind [20] and Siddharth *et al.* [21]. They track action participants and use them to recognize and generate natural-language sentences complete with adjectives, adverbs, and prepositional phrases describing the properties of and interactions between multiple participants. This kind of deep analysis is impossible without localization. The key to both of these methods is the Event Tracker of Barbu *et al.* [22] which is a framework for performing simultaneous tracking and action recognition. It allows a high-level action model to influence the tracking process, preventing the issue of cascading failure, and making it more reliable than two step independent tracking and action-recognition methods.

The Event Tracker is not an action recognition model, nor a learning method, but a way to combine an existing action recognizer which satisfies certain properties with existing object-detectors to perform simultaneous tracking and action recognition. The action models used by Barbu *et al.* [22], Yu & Siskind [20], and Siddharth *et al.* [21] represent actions only through simple features such as the gross motion and relative position of the action participants. They therefore can only support rather simple action classes, such as *pick up* and *approach*, which can be represented by such features. They rely on pretrained generic person and object detectors to provide detections to the simultaneous tracking and action-recognition process. However, the actions in many standard action-recognition datasets, like UCF Sports [2], depict people in rather unusual poses which result in failure of generic person detectors. For these reasons, the Event Tracker framework has not been shown to work on standard action-recognition datasets.

2.0.1 Contribution

We present a new method which performs action recognition and localization using intuitively meaningful models, while still achieving competitive performance on complex actions and on standard action-recognition datasets. This is made possible by a novel learning method, which integrates detection, localization and action recognition, training a combined model to maximize action classification. Further, it does so in a way compatible with the Event Tracker, allowing it to be used as a tool, and showing that it can be extended to complex actions. This new method models the time series of the appearance and motion of the people and/or objects that participate in an action. This is done by learning, for each action class, a temporal sequence of actor-centered Histograms of Oriented Gradients (HOG) object detectors [23] and optical-flow-based motion-pattern detectors. These models are retinotopic. That is, like the receptors of the retina, they are spatially organized in a dense grid. They are therefore able to represent coarse spatial organization which is lost in sparse interest-

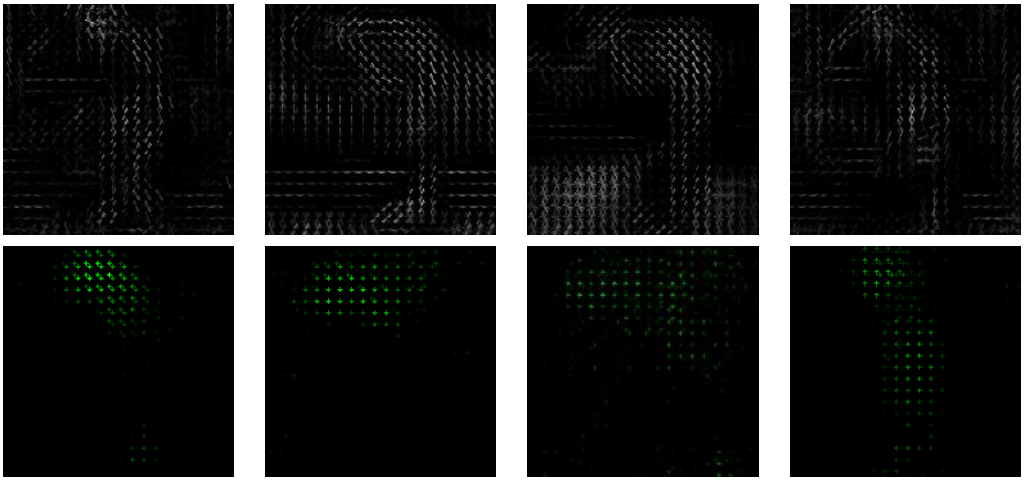


Fig. 2.1. Visualization of appearance and motion models for the *bend* action. Appearance (top row) and motion (bottom row) are represented as dense grids of edge and optical-flow orientation histograms (HOG and HOF) centered on the action. These models depict a person bending over and standing back up.

point-based methods, and can be meaningfully visualized and understood by humans as we demonstrate in Fig. 2.1.

Each of these models represents the motion or appearance of the actor during a highly distinguishing temporal subsequence of an action. A sequence of such models represents the changing characteristics of an actor in a video as they evolve over time during that action. Each action class is modeled with a hidden Markov model (HMM) [24] whose state-conditioned output models are the appearance and motion detectors. Thus each HMM state represents one of the characteristic appearance and/or motion patterns, and the sequence of HMM states models the changing appearance and motion over time. Such an HMM is learned automatically through a novel discriminative training procedure on both positive and negative samples of an action class. This results in simultaneously learning which subsequences of the action are useful for recognition, the order in which they take place, and the appearance and motion detectors associated with each, without the need for manual key-framing or other extensive manual annotation. We require only bounding boxes around the actor in the training videos. We further show how our learned models allow for im-

provements to the Event Tracker to perform simultaneous object detection, tracking, and recognition of actions.

2.1 Related Work

There is recent prior work which also attempts to model time-series in general [25], which model actions in video as temporal sequences, or which attempt to model the appearance or pose of the participants. Tang *et al.* [26] break videos into fixed-length segments, on which BOW features are computed, and then use an HMM to model the sequence of segments. However, the segment length was manually specified, varying by dataset, and was very long, reducing each video to a very small number of segments, and largely providing the HMM with the latent state-assignment information. Niebles *et al.* [5] model actions as sequences of motion segments at varying time scales. However, they represent each segment as a histogram of sparse STIP features, whereas our method uses a dense sampling grid to represent the coarse-scale appearance and motion. Like Tang *et al.* [26], Liang *et al.* [27] also break videos into fixed length segments. They employ a Spatio-Temporal-And-Or-Graph, which models actions as a sequence of disjunctions of local BOW models over HOF and HOF features computed at STIP interest points. They attempt to account for spatial structure by explicitly modeling the relative positions of these local models. Wang *et al.* [8] explicitly model human joint pose, mine for clusters of pose sequences, and then reduce videos to BOW histograms for classification with a temporally invariant SVM. Wu & Shao [28] also classify video with explicitly modeled human pose. They model the sequence of human skeleton estimates extracted from 3D RGB-D sensor data through a Hidden Markov Model with a series of neural networks. In contrast, our method implicitly models pose through appearance and motion models, allowing application to appearance change of non-human objects, and models the temporal sequence of changing appearance rather than reducing it to a histogram like Wang *et al.* [8]. Barbu *et al.* [22] also employ an HMM model, but do not represent the

appearance or motion patterns of the action. They instead rely on features computed directly from bounding boxes themselves, ignoring the content of the video inside these boxes. They obtain such boxes from pretrained object models which are used only for detection and are not state conditioned. Thus, while they do model the sequence of changing gross object motion and of spatio-temporal relationships among action participants, they do not model the sequence of changing object appearance or motion patterns which cannot be captured by the gross movement of the bounding detection boxes. Banerjee & Nevatia [29] present a method based on key-pose filters which model both appearance and motion. However, whereas we model the temporal sequence of action state and infer the state at each frame with an HMM, they explicitly model and infer the position of the key frames with a hidden conditional random field (HCRF). This model cannot localize the action, and instead depends on a separate pedestrian tracker, whereas we learn models which enable simultaneous detection, tracking, and action recognition.

The two methods which are most similar in spirit to our work are Tian *et al.* [30] and Yao *et al.* [31]. Tian *et al.* [30] present a spatio-temporal extension to the deformable part model (DPM) [32] intended to serve as an action detector. They train a template for each action which includes a cuboid root filter and a number of displaceable cuboid part filters. Because the filters are cuboids, they must be large enough to encompass the entire space traversed by a moving object over the course of the entire action, whereas our models track the object through time, allowing the filters to be centered on the image region where an object is present in a specific frame in the video. Their method must also be provided with annotation of the temporal extent of a single cycle of each training action, such as a single *jumping jack*, whereas our method can automatically handle such repetition without any annotation. Yao *et al.* [31] present another recent method which extends the ideas of the DPM object detector to action recognition. They use an HMM to model the sequence of such models, each of which contains a HOG root filter and a set of explicitly labeled body-part models which each consist of a HOG model and a Histograms of Oriented Flow

(HOF) model [33]. For training, they require manual annotation and labeling of the body parts. We require no such part annotations. Whereas our training procedure can automatically cluster the training video frames into a sequence of discriminative poses to be modeled, they must do so by performing clustering on these manual part annotations.

2.2 Action Model

The sequence of each action class is modeled by an HMM with N states. An HMM assumes the existence of a hidden state sequence X_t, for $t = 1, ..., T$, and that a series of observations D_t were sampled from state-conditioned output distributions $b_i(D_t) = P(D_t|X_t = i)$. An HMM's parameter set λ consists of an initial state distribution π, specifying the probability of beginning in each state, an N by N transition matrix A, whose elements a_{ij} specify the probability of transitioning from state i at time t to state j at time $t+1$, and the parameters of the set of state output models b_i, for $i = 0, ..., N$, which depend on the particular output models used.

The likelihood $l = P(D_1, \ldots, D_T|\lambda)$ of the data given the HMM can be computed efficiently via the forward algorithm [24]. It is computed as $l = \sum_{i=1}^{N} \alpha_T(i)$, where $\alpha_t(i) = P(D_1, \ldots, D_t, X_t = i|\lambda)$, and is computed recursively via $\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1} a_{ij} b_j(D_t)$. Given a uniform prior over action classes, a video can be classified by computing the likelihood of the HMM for each class and choosing the class with the highest likelihood.

In the present method, as illustrated in Fig. 2.2, the observations D_t of a video are a sequence of dense $n \times n$ grids of HOG and/or HOF feature vectors extracted from each frame of the video along a sequence of bounding boxes around the actor. For training, we assume that such boxes are available by some means, either manually specified or produced automatically. Such a sequence can be produced automatically by a variety of methods: background subtraction, adjacent-frame image differencing, or more powerful tracking methods. For simple datasets such as Weizmann [3],

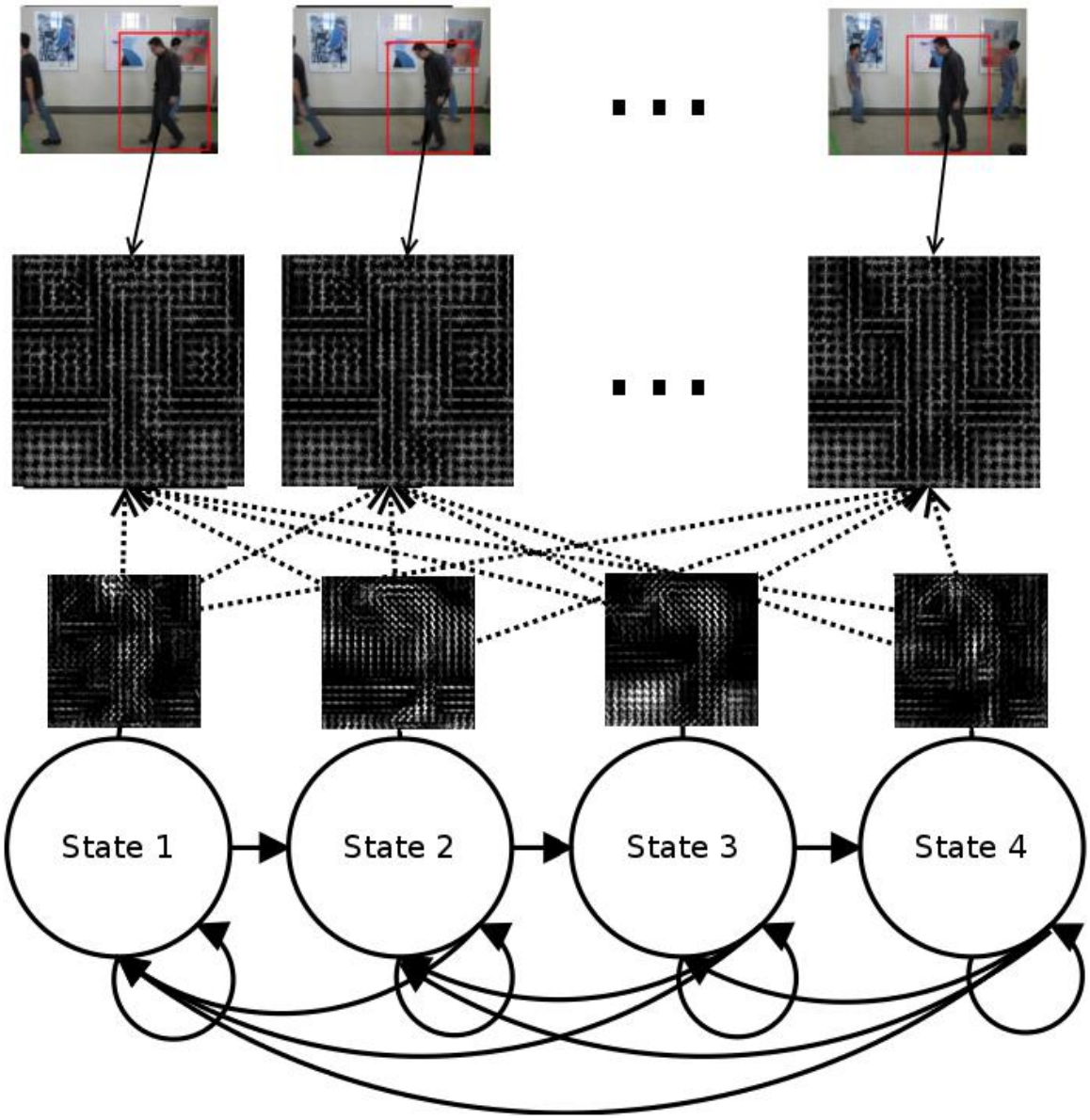


Fig. 2.2. Diagram illustrating the presented method. Appearance (and motion, not shown) features (second row) are extracted from the images within bounding boxes in the video frames (top row). These features are matched against the output models (third row) associated with each state (bottom row) of an HMM which models a particular action class. The sequence of states, and thus the sequence of appearance and motion associated with that action class, is modeled with a transition distribution.

background subtraction is sufficient to find the actor, and such background masks are provided with the dataset. We automatically extract bounding boxes from these masks for use in training. Other datasets, such as UCF Sports, UCF11 [16], and UT-interaction [34], provide such boxes directly. At test time, we obtain such bounding boxes automatically. Our trained models allow us to employ a modified version of the Event Tracker [22] to perform simultaneous detection-based tracking and action recognition, which is described further in Section 2.3.

Following Dalal & Triggs [23], our HOG features use 9 orientation bins. However, rather than using 4 such histograms normalized according to different blocks, which would result in a 36 element vector for each block, we only normalize each histogram once in each block in order to reduce the feature dimensionality. Thus each HOG grid position consists of a 9 element vector. Our flow feature is similar to HOF. While HOF involves computing differential flow, breaking the image region into overlapping patches, binning the differential flow by orientation in each patch, and then normalizing each patch, our feature, in contrast, is computed for each patch by taking the average flow and then binning its horizontal and vertical components each into three bins, resulting in a 6 element vector for each grid position. The feature vectors at the grid positions are concatenated into a single HOG vector and single HOF vector per frame. Thus, the feature vector D_t for each frame is of length $9n_h^2$ for an $n_h \times n_h$ HOG grid, and is of length $6n_f^2$ for an $n_f \times n_f$ HOF grid.

The HMM state output model $b_j(D_t)$ for a particular state j and the normalized HOG or HOF feature vector for a particular frame t is computed with a sigmoided dot product,

$$b_j(D_t) = \frac{1}{1 - \exp\left(-h \sum_i w_{ji} D_{ti}\right)} \tag{2.1}$$

where h is a smoothing parameter of the sigmoid and w_j is a weight vector of length $9n_h^2$ for a HOG model and $6n_f^2$ for an HOF model.

We further take symmetry into account when computing $b_j(D_t)$. This is done by computing both $b_j(D_t)$ using the original features D_t and using those features D'_t,

which were computed from the video after reflecting it along the vertical axis. For each state j, and for each frame t, the maximum of these two scores is taken to be $b_j(D_t)$. This provides a degree of viewpoint invariance, allowing a single model to match an action when viewed from either side, or to switch from one orientation to another, as in row two of Fig. 2.3.

These output models do not satisfy the sum-to-one constraint, and are therefore are not probability distributions, but rather single-layer neural networks. Niles & Silverman [35] showed that HMMs can be viewed as a special case of neural networks and that, in this context, relaxing the sum-to-one constraints is natural, and may result in a better classifier. However, doing so does remove the probabilistic interpretation of the HMM. Therefore, we refer to the output of the HMM forward algorithm as an HMM score, rather than a likelihood.

HOG object detectors generally also use a dot product model. Thus, each state's HOG output model can be thought of as an object detector, and its HOF output model as a motion-pattern detector. The use of object and motion detectors as HMM output models allows an action to be recognized by a sequence of appearances, by a sequence of motion patterns, or by a combination of the two. For example, an *open drawer* action could be recognized by the appearance of a closed drawer, as determined by a high scoring closed-drawer detector in one state, followed by a transition to the appearance of an open drawer, as determined by a high scoring open-drawer detector in another state. Fig. 2.3 depicts a visualization of such a model on an unseen test video. This figure was generated automatically by rendering, on a given frame t, the HOG output model from the state j with the highest $P(X_t = j|D_1, \ldots, D_T, \lambda)$ as computed by the forward-backward algorithm [24]. The white lines indicate HOG model weight at an orientation and location. Brighter lines indicate more weight, while those with non-positive weight are not drawn. Fig. 2.3 also shows a similar visualization of the HOF model for *jumping jack* on an unseen test video. A green arrow indicates the weight of the model at a particular position and orientation. Brighter arrows have higher weight, while those with negative or zero weight are not

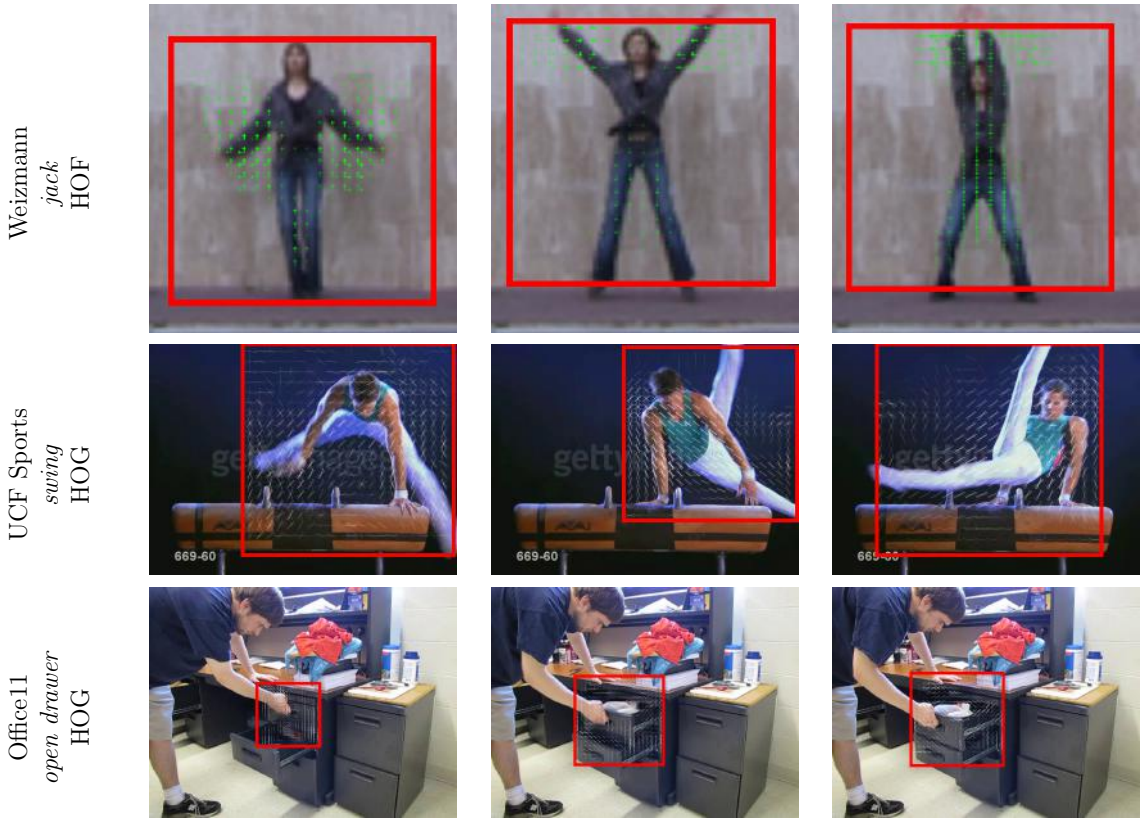


Fig. 2.3. Visualization of the learned HOF models from *jumping jack* (Weizmann) and learned HOG models from *swing* (UCF Sports), and *open drawer* (Office11). The *jumping jack* models closely follow the person's sequence of body and limb movements, while the *swing* and *open drawer* models closely follow the sequence of edge-structure characteristics of each action. Each visualization was produced on an unseen test video by using the forward-backward algorithm to produce a weighted assignment of each frame to each HMM state, and rendering onto each frame that model which belongs to the HMM state with the highest weight.

drawn. Fig. 2.4 shows visualizations of a number of example output models showing both HOG and flow.

Because the opening of a drawer involves simple linear motion which depends on the viewpoint, such an action might not be easily recognizable with motion features. Other actions, such as the *skateboarding* and *walking* classes in UCF Sports, which have very similar appearance, might be more easily distinguished by the difference in motion of the person's legs. The use of both appearance and motion detectors allows

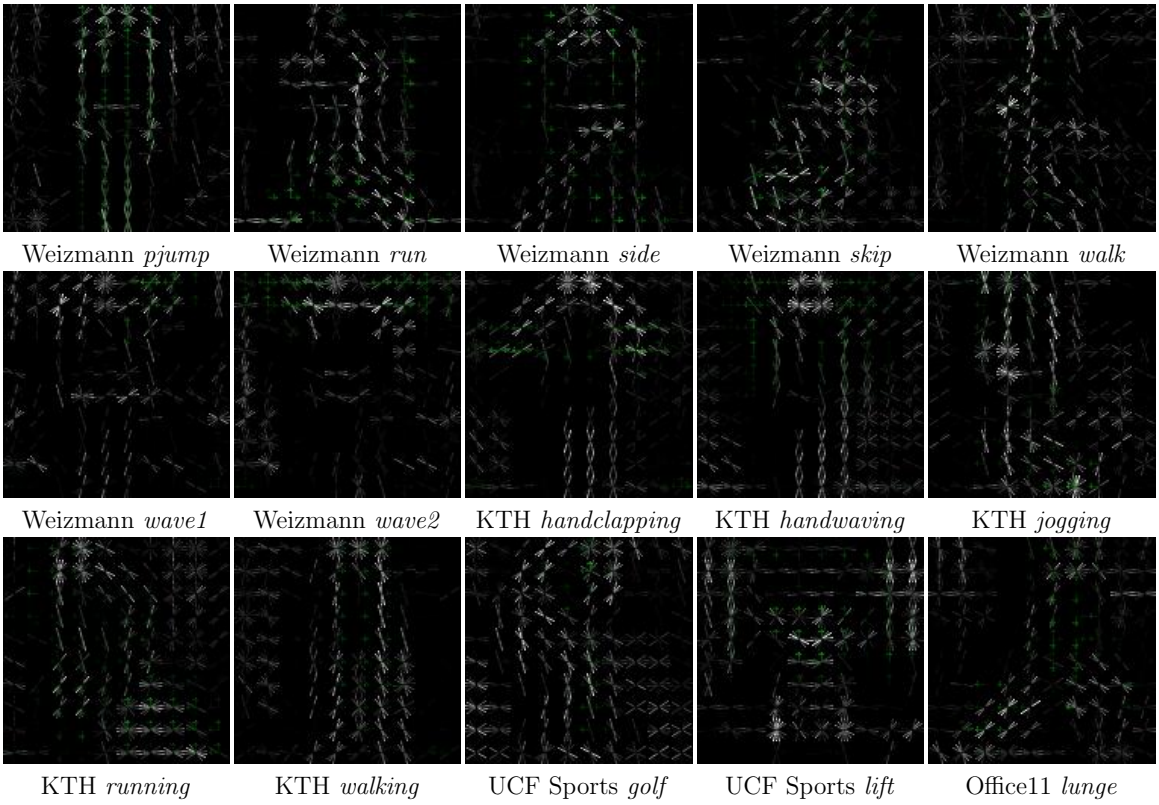


Fig. 2.4. Visualization of the learned appearance and motion models from example states from a variety of action models. These each depict the edge orientations (white) and motion orientations (green arrows) associated with a particular state of a particular action. These examples all show the general form of a person, but in different postures, and with motion in different places and orientations. For example, the UCF Sports *golf* model (bottom row, third column) shows the form of a person bent over while putting, with horizontal motion at the position of the club, while the Weizmann *wave1* model (second row far left) shows a person standing up straight, with motion corresponding to waving a single hand in the air.

such differences to be modeled. Other actions might involve the same appearance or motion, but in a different sequence, such as a *close drawer* action, which involves the same appearances as an *open drawer* action, but in the opposite order. The combination of the HMM's state transition matrix with the detectors in its output models allows such actions to be distinguished by the order in which these appearances take place. BOW approaches would not be able to make such a distinction.

2.3 Simultaneous Tracking and Action Recognition

At test time, the goal of our system is to both recognize and localize the actions occurring in video. We use a modification to the Event Tracker [22] to combine our action model and its internal object detectors into an integrated simultaneous detection, tracking and action recognition system. Given a video, a set of scored object-detection boxes in each video frame, and an HMM action model, the Event Tracker produces a track composed of a single detection in each frame along with a score indicating how well the track depicts the action based upon the HMM action model.

The Event Tracker operates by globally optimizing a joint tracking and action-recognition objective function.

$$\max_{\substack{q^1,\ldots,q^T \\ j^1,\ldots,j^T}} \sum_{t=1}^{T} f(p^t_{q^t}) + \sum_{t=2}^{T} g(p^{t-1}_{q^{t-1}}, p^t_{q^t}) + \sum_{t=1}^{T} e(j^t, p^t_{q^t}) + \sum_{t=2}^{T} a(j^{t-1}, j^t), \tag{2.2}$$

where the four terms are: $f(p)$, the detection score of each box p in the sequence, $g(p, p')$, the track-coherency score between each pair of boxes p and p' in adjacent frames, $e(j, p)$, the HMM state output-model score of the features in box p scored against the state j, and $a(j, j')$, the state-transition score between the states j and j' assigned to each pair of adjacent frames. Thus, it simultaneously finds the best possi-

ble sequence of detections and the best possible sequence of HMM states, producing the MAP estimate of the HMM score given the optimal track.

The Event Tracker has three steps:

1. Obtain an over-generated set of scored object-detection boxes.
2. Construct a lattice through the cross-product of detections and HMM states.
3. Use the Viterbi algorithm [36] to find the optimal path through the lattice.

The Viterbi algorithm uses dynamic programming to efficiently find the optimal path through a lattice with unary vertex costs and binary edge costs. Because the HMM and tracking cost functions each consist of unary and binary terms, a combined lattice with unary and binary costs acting on the cross-product of all box-state pairs can be generated. The Viterbi algorithm then finds the optimal combined sequence of detections and HMM states, along with the corresponding cost.

Joint optimization of the combined cost function allows the action model to bias the tracker. This is advantageous when compared to the use of tracking and action recognition in independent steps. For instance, only one of several people in a particular video might be performing a given action. If that person is also difficult to track, perhaps partially occluded or in the background, an independent tracker would have difficulty in finding that person amidst the other, more easily tracked people. However, the Event Tracker allows the high level information reflected in the action model to bias the low-level tracker towards that person which best exhibits the characteristics of that action.

Our use of HMMs allows us to make use of the Event Tracker. In addition, the availability of HMM state conditioned object models produced by our method allows us to improve upon it.

In step (1) of the Event Tracker, Barbu *et al.* [22] use generic pretrained DPM object detectors to generate a large number of detection boxes. Then they discard all but the top P detections in each frame based upon their detection scores. In contrast, we produce the detections using the action-specific HMM state output models which

have been trained by our system to maximize recognition performance. We do so by running our state-conditioned output models as object detectors, producing, as in DPM, a score pyramid in each frame corresponding to the scores of boxes at each possible box position at a variety of scales. Thus, for each position in the pyramid, which corresponds to a box with a particular position and scale, we have the output score for each HMM state. Next, the top P boxes in each frame are kept for each HMM state based upon this score. This differs from Barbu *et al.* [22], where the top P boxes are kept based solely on a generic object-detector score. Step (2) differs in that our detection boxes were produced using state-conditioned models, and thus each box has an associated HMM state. We therefore add a constraint to the lattice which forces the chosen HMM state in a given frame to match the state associated with the chosen box in that frame.

These modifications allow further influence of the high-level action model onto to not only the tracker, but also the object-detection process. This has two advantages. First, the action-specific HMM state-conditioned object models can detect people or objects in poses or configurations which are generally unusual, and thus score poorly with a generic detector, but which might be common, or even characteristic of the action in question. Thus, actions which are very difficult to track with generic models because the person takes an unusual pose become straightforward to track with our method. Further, the inclusion of other features like motion into the score prior to thresholding the boxes avoids the loss of detections which have comparatively poor object appearance score, but which exhibit the desired motion so well that they are part of the optimal track.

We run our modified Event Tracker using the model for each action class. For each action, this yields the optimal track, along with a score indicating how well each such optimal track depicts the action. If there is no person or other agent performing a given action in the video, the associated score will be low. We perform classification and localization by choosing the class and track from the model with the highest score.

2.4 Training the Models

The action models are discriminatively trained through gradient descent with a maximum-margin objective function. Such an objective encourages each model to score highly on training samples of its own class, and poorly on others. It also encourages each model to have higher score on training samples with matching class than the other models on those same samples. Thus optimizing such a function aims to improve classification performance.

The objective function used, O, is a sum over individual training-video costs O_v. Each video cost O_v is the square of a soft version of the multi-class hinge-loss function used by Crammer & Singer [37] for multi-class SVMs:

$$O_v = \max(0, 1 + \textsc{SoftMax}(\{s_{vr} | r \neq k_v\}) - s_{vk_v})^2 \tag{2.3}$$

where s_{vr} is the HMM score (in log space) of video v with HMM r, k_v is the index of the HMM whose class matches the class of video v, and

$$\textsc{SoftMax}(S) = y \log \left(\sum_{i=1}^{U} \exp \left(\frac{S_i}{y} \right) \right) \tag{2.4}$$

where U is the dimension of S and y is a smoothing constant. Thus, when the correct HMM score s_{vk_v} on a video v is greater than the SoftMax of the incorrect scores s_{vr} by a margin greater than 1, the video is classified correctly with a margin of at least 1, and the cost is zero. If the margin is less than 1, then

$$O_v = \left(1 + y \log \left(\sum_{\substack{r=1 \\ r \neq k_v}}^{C} \exp \left(\frac{s_{vr}}{y} \right) \right) - s_{vk_v} \right)^2 \tag{2.5}$$

where C is the number of classes.

To optimize O, we compute its gradient with respect to the HMM parameters, and perform gradient descent. We use the chain rule to compute the gradient as a

function of the gradients of the individual video costs with respect to the parameters of individual HMMs. The derivative $\frac{\partial O}{\partial x_{rd}}$ of O with respect to x_{rd}, the dth parameter of HMM r, is

$$\frac{\partial O}{\partial x_{rd}} = \sum_{v} \frac{\partial O_v}{\partial x_{rd}} \tag{2.6}$$

where by the chain rule,

$$\frac{\partial O_v}{\partial x_{rd}} = \frac{\partial O_v}{\partial s_{vr}} \frac{\partial s_{vr}}{\partial x_{rd}} \tag{2.7}$$

and $\frac{\partial O_v}{\partial s_{vr}}$ depends on whether $r = k_v$, *i.e.*, whether this HMM's class matches that of video v.

If $r = k_v$, then $\frac{\partial O_v}{\partial s_{vr}} = -2\sqrt{O_v}$, otherwise $\frac{\partial O_v}{\partial s_{vr}} = 2\sqrt{O_v}\frac{\partial \textsc{SoftMax}_u(s_{vu})}{\partial s_{vr}}$. After simplifying, we obtain

$$\frac{\partial O}{\partial x_{rd}} = \sum_{v} 2Z\sqrt{O_v}\frac{\partial s_{vr}}{\partial x_{rd}} \tag{2.8}$$

where $Z = -1$ if $r = k_v$, otherwise

$$Z = \frac{s_{vr}\exp(s_{vr})}{\sum\limits_{\substack{u=1 \\ u \neq k_v}}^{C} \exp\left(\frac{s_{vu}}{y}\right)} \tag{2.9}$$

and where $\frac{\partial s_{vr}}{\partial x_{rd}}$ is the derivative of the score computed by the HMM forward algorithm on video v with HMM r, with respect to parameter d. Using the techniques of reverse-mode automatic differentiation [38], which is a systematic way to apply the chain rule, the entire gradient of s_{vr} with respect to all HMM parameters can be computed efficiently within a small constant factor of the time needed to compute the score s_{vr}. This is done by taking a forward pass through the program, computing the value of the function while storing intermediate values, and then taking a backward pass, computing the derivatives using these stored intermediate values. This process is analogous to back propagation in neural networks [39].

The parameters of an HMM are composed of three parts, π, a, and w. The derivative $\bar{a}_{ij}$ of an HMM's score on a video with features D_t with respect to the elements of its transition matrix a_{ij}, and $\bar{\pi}_i$, the derivative with respect to the elements of its initial state distribution, π_i, are given by

$$\begin{aligned} \bar{a}_{ij} &= \sum_t \bar{\alpha}_{tj}\alpha_{(t-1)i} \\ \bar{\pi}_i &= \bar{\alpha}_{0i}b_i(D_0) \end{aligned}$$

where $\bar{\alpha}_{ti}$, the derivative of the HMM score with respect to α_{ti}, is given by

$$\bar{\alpha}_{ti} = \sum_j \bar{\alpha}_{(t+1)j}b_j(D_t)a_{ij} \tag{2.10}$$

The derivative $\bar{w}_{jm}$ of the HMM score with respect to w_{jm}, the mth weight parameter of the output model of state j, is given by

$$\bar{w}_{jm} = \sum_t \frac{hD_{tm}\bar{b}_{tj}\exp(-h\gamma_{tj})}{(1+\exp(-h\gamma_{tj}))^2} \tag{2.11}$$

where h is the smoothing parameter of the sigmoid function, D_{tm} is the value of the mth element of the feature vector D_t at frame t, γ_{tj} is the value of the dot product between D_t and w_j, and where $\bar{b}_{jt}$, the derivative of the HMM score with respect to $b_j(D_t)$ is given by

$$\bar{b}_{jt} = \begin{cases} \bar{\alpha}_{tj}\sum_i a_{ij}\alpha_{(t-1)i} & t > 0 \\ \bar{\alpha}_{0j}\pi_j & t = 0 \end{cases} \tag{2.12}$$

During optimization, the output models are constrained to have unit magnitude. This is accomplished in two ways. First, the gradient of the objective function is made to take the normalization into account. This is done by creating an augmented objective function which normalizes the output models before passing them into the original

objective function. We then take the gradient of this augmented objective function. We let

$$w_{jm} = \frac{w'_{jm}}{\sum_z w'_{jz}} \tag{2.13}$$

Then, the desired derivative becomes

$$\bar{w}'_{jm} = \frac{\bar{w}_{jm}}{\sum_z w'_{jz}} - \frac{\sum_z \bar{w}_{jz} w_{jm}}{\left(\sum_z w_{jz}\right)^2} \tag{2.14}$$

The optimization procedure breaks the parameter space into two subspaces, the space of initial-state parameters π and transition parameters a_{ij} and the space of output-model parameters w_{jm}. It alternates between taking steps in each space. The initial-state parameters π and transition parameters a_{ij} are updated using the growth transform [40], while the output models are updated using gradient descent with a dynamic step size. While the above gradient points in a direction for which an infinitesimally small step will maintain the magnitude of the output models, the finite step size taken by gradient descent results in small changes to the magnitude. To compensate, the output models are also renormalized after each gradient-descent step.

The effect of this optimization process is that the HMM parameters are updated in a way that maximally improves the discriminative power of the HMM score. The state-conditioned output models each gradually become more distinct and specialized towards a particular temporal subsequence of the action as the weighted assignment of frames to states becomes less uniform. This results in the HMM states for a given action model matching the sequence of the most distinct appearance and/or motion that occurs in that action class, and rejecting the appearance and motion that occurs in other actions as encoded in negative training samples. The latent HMM states are automatically assigned to these distinctive moments in the video without need for manual annotation of such. Fig. 2.3 shows visualizations of the models learned for

jumping jack from Weizmann, for *swing* from UCF Sports, and for *open drawer* from Office11, a new dataset that we have filmed.

In contrast to pretraining object models and treating the detector output scores as features, which would require manual assignment of training frames to each model, the use of detectors as HMM state-conditioned output models allows the sequence of detectors to be trained automatically as part of the action model. Thus the latent state assignment is learned without supervision in conjunction with the detectors themselves.

Training the detectors in conjunction with an HMM also makes it possible for the detectors to be trained in a discriminative fashion specifically to maximize action-class discrimination. In contrast, an object detector is generally trained to maximize detection of that object, and therefore seeks to score highly on all positive training frames. This is achieved in methods such as DPM by the use of a disjunction of root filters. However, some poses are not useful for discriminating between actions, such as a person standing upright, who might be preparing to either *bend*, *kick*, or *jump*. Training a DPM model for each action would result in each model including a root filter corresponding to a person standing before the action. In contrast, the simultaneous training of the detectors and the HMM to maximize discrimination between the actions allows non-discriminative poses to be ignored automatically as the detectors learn to model the most discriminative sequence.

2.4.1 Mining for Difficult Negatives

The HMM scores s_{vr} depend upon the sequence of boxes on which the features have been computed. For the positive samples, *i.e.*, when $r = k_v$, the provided track of bounding boxes around the actors are initially used to compute s_{vr}. After the models have been partially trained, the Event Tracker is used to find the highest scoring track through each training video with the model whose class matches that video. The score s_{vr} for $r = k_v$ is then taken to be a weighted mean between that computed

on the provided track, and that computed on the automatically determined track. This allows the system to handle errors and misalignments in the tracks provided for training.

We obtain additional negative samples by running our partially trained action detector on the training videos similar to the way Felzenszwalb *et al.* [32] obtain such for training an object detector by running it on training images. To produce difficult negative samples, our modified Event Tracker is used to find the highest scoring tracks in each video v using all non-matching models of class r such that $r \neq k_v$. These tracks are used to compute the HMM scores s_{vr} for the non-matching models for each training video. Thus the objective O pushes each model to score as high as possible on the tracks through the positive videos and as low as possible on all tracks through other videos. The use of such difficult negative tracks is very important. Without them, the system can easily classify the training videos on the provided tracks and will not learn models that can classify videos with automatically produced tracks, as will be shown in Section 2.6.

2.4.2 Implementation Details

In addition to the output parameters w and the transition parameters a, which are fully learned, our system contains eight hyper-parameters, which are not learned. There are four hyper-parameters in the training procedure: the iteration delay until the onset of retracking the training videos, the frequency of retracking, the automatic track queue size, and the smoothing parameter of the SoftMax in the objective function. There are four hyper-parameters in the model: the number N of states, the grid sizes n_h and n_f for the HOG and HOF models, and the smoothing parameter h to the output model sigmoid.

SoftMax is used in place of Max in the objective function in order to make the objective smooth. It removes the discontinuities in the gradient that would otherwise be caused by Max. The smoothing constant y in SoftMax, which controls how

closely it approximates MAX, was fixed at 0.1. This small value was chosen so that the value of SOFTMAX is close to the true MAX.

At the start of training, the models are initialized in the following way. The output models are initialized randomly with elements sampled uniformly in $[-1, 1]$. These are then shifted so that each output model sums to zero and subsequently normalized to have unit magnitude. The shift is done in order to prevent some output models from having lower mean, and thus scoring poorly on all feature vectors. This would be undesirable and potentially cause the model to learn to skip that state before it has a chance to adapt to the data.

The transition matrices are initialized to a chain structure which allows arbitrary backward transitions but only allows transitions forward by a single state, *i.e.*, the states are labeled $0, \ldots, N-1$ and state i can transition to all states $j \leq i+1$, where the probabilities are uniform among allowed transitions. Thus skipping states is initially disallowed, but arbitrary looping is possible. The HMMs are constrained to begin in the first state 0, and end in the last state $N-1$. This is done so that the HMM cannot return a high score on a video which depicts only a subsequence of an action, but must find the full sequence. On such a video, the low score of states corresponding to undepicted parts of the action will result in a poor overall score. The number of states N was set to 4 by looking at the actions in the datasets and observing that there are generally fewer than 4 major human-discernible poses. Because the model can learn that not all states are necessary, we used this upper bound for all datasets and for all classes.

The computation needed to mine a training video for high-scoring negative tracks, $O(TNB(n_f^2 + n_h^2))$ (where B denotes the number of locations in the image pyramid), dominates that needed to compute the gradient and perform a parameter update, $O(TN(n_f^2 + n_h^2) + TN^2)$, because of the comparatively compute-intensive step of running the HMM state-conditioned models as detectors at a large number B of positions and scales in each frame. Therefore, new negative tracks are not produced every iteration of training. This is done with a retracking frequency of once per

1000 iterations instead, so that the models are will have been substantially modified and will produce a new set of negative tracks each time. To avoid cyclic behavior caused by abrupt changes in the set of negatives, we do not discard the previous set of negative tracks each time new tracks are found. Instead, we maintain a queue of three such sets, discarding the oldest set when the queue becomes full. Thus each set of negative tracks is used for 3000 updates. Positive mined tracks are not used until the model has been substantially trained in order to avoid using garbage tracks as positive samples. In practice, they have been used after an onset delay of 2000 iterations, and the weight given to the mined positives has been half the accuracy on the training set.

As in Dalal & Triggs [23], our HOG features are computed using 8×8 pixel blocks. During training, the image inside each bounding box is scaled so that the number of 8×8 pixel blocks in each box matches the size of the grid of the output models. When performing automatic detection and tracking, the computation of the image scale pyramid controls the size of the boxes. Larger values of the grid sizes n_h and n_f allow the modeling of finer grained spatial information, but increase the runtime, particularly for running all the output models as detectors. The balance between detail in the model and computational efficiency is thus determined by the scale of the participants in the dataset. We want the grid size to be large enough to match the scale of the participants in the training videos: if it is smaller, the boxes will be scaled down and details lost; if it is larger, the boxes will be scaled up, and the extra detail will be redundant and add unnecessary computation. For all datasets except the LCA dataset, we used $n_h = 10$ and $n_f = 10$, yielding 10×10 grids of HOG and HOF features. For the LCA dataset, we used $n_h = 6$ because the participants tend to be at a smaller scale, and the larger dataset makes computational efficiency more important. The smoothing parameter of the output model sigmoid h controls its steepness. It was chosen to be 10 so that the output for the dot product of two random unit vectors is usually close to zero. Since our method performs well with these intuitively chosen values on all the datasets, we have not done a search through

the space of these parameters to maximize performance. It is possible that some other values produce better results.

2.5 New Dataset

We filmed a new dataset, Office11, to highlight the power of the current method. We are interested in recognizing and localizing specific actions, such as might be used for a robotics or surveillance application. Many other methods are focused on web video and general pattern recognition. These different focuses result in different challenges. Video taken from movies or the from web sites like YouTube were produced for human consumption, and contain scene transitions, highly zoomed in camerawork, and unknown camera movement. However, a surveillance camera or robot would not be presented with these difficulties, but instead have other challenges, like occlusion and multiple simultaneous actions. Localization of detected objects and actions would be also be more important than in a web-based setting. The types of actions to recognize in these two contexts also differ. Whereas it might be useful to categorize and retrieve YouTube videos based on general scene categories like *military parade*, *wedding*, or *horse race*, specific actions like a person *marching*, *giving* an object, or *riding* a horse would be more relevant to a surveillance system or robot.

Our dataset is meant to better reflect these challenges, and attempts to remove two shortcomings of prior standard datasets. First, most action-recognition datasets focus on people, and do not include actions which involve the manipulation of objects. Second, many action-recognition datasets have strong correlation between the background and the action taking place. For example, in UCF Sports, *diving* always occurs in an Olympic swimming pool. No other actions take place in a pool, so identification of the background provides a powerful cue to the action: "For instance, *v_spiking* normally happens in a crowd of people, and *diving* happens in a pool. This is common for professional sport actions which take place in highly structured environments" [16]. Similarly, in UCF11, "Basketball shooting and volleyball actions

are also confused in some cases: this is largely because most of the time, the basketball and volleyball sports use very similar courts" [17]. While some consider the use of contextual information to inform the classification process to be a virtue, we desire to recognize actions in a semantically meaningful way, solely by recognizing and localizing the action itself, which is more true to the goal of video action recognition.

Office11 was filmed to remove these two shortcomings. It avoids spurious correlation between actions and backgrounds and includes actions which involve both manipulation of objects and human body-posture changes. The dataset includes 11 action classes: *bend*, *kick*, *lunge*, *wave*, *open drawer*, *close drawer*, *answer phone*, *hang up phone*, *talk on phone*, *operate hole punch*, and *knock over cup*. Some actions are the same or similar to those in other datasets, such as *bend*, *kick*, and *wave*, but they all take place in the same background, with several people constantly walking around in the field of view, often occluding each other and depicting parts of other actions as one person performs the entirety of the desired action. The *open drawer* and *close drawer* actions take place in the same cluttered office with 7 different drawers, which are opened and closed. The remaining actions also take place in a cluttered office. In this dataset, the background cannot be used to distinguish *bend*, *kick*, *lunge*, and *wave* from each other, nor to distinguish *open drawer* from *close drawer*, or discriminate *answer phone*, *hang up phone*, *talk on phone*, *operate hole punch*, or *knock over cup* from each other. The dataset is slightly bigger than UCF Sports, with 179 videos.

2.6 Experiments

The performance of the current method was compared with state-of-the-art prior methods on three standard datasets (Weizmann, KTH [4], and UCF Sports), as well as on the LCA dataset [41] and Office11, and was found to be competitive with recent methods. We also compare to several baseline methods: the HOG baseline, the HOF baseline, and the 3-Stage Baseline. The HOG and HOF baselines use the method described in the current manuscript, but with only a single HMM state and using

only the HOG or HOF feature, respectively. Both baselines use the same combined automatic tracking method described here. Their poor relative classification accuracy shows that the performance of our method is not due simply to the strength of the features, but due to their combination together with the sequence model. The 3-Stage baseline classifies each video by detecting the person, tracking them, and running the classifier of the current manuscript in sequence. The DPM object detector produces the top person detection in the first frame of the video, initializing the Tracking-Learning-Detection (TLD) tracker of Kalal *et al.* [19]. Finally, this track is passed to the same trained action recognition models used to evaluate our method. The poor relative performance of this baseline shows the importance of the integration of the detection, tracking, and classification systems. The tracks produced by this baseline are also used to compare localization performance.

Classification results are reported in Table 2.1 and Fig. 2.5, and localization accuracy results are reported in Fig. 2.6. The standard leave-one-actor-out evaluation protocol from prior publication was used for the Weizmann dataset. The KTH dataset was run with the train-test split associated with the dataset and used by Yao *et al.* [31]. The UCF Sports dataset was run with 16-fold cross-validation. To compare against prior work on the LCA dataset, we used the same 70:30 train-test split used to produce the results in Barrett *et al.* [41]. For Weizmann, UCF Sports, and Office11, the bounding boxes for training were determined using information provided with the dataset. For KTH, we used the boxes made available by Lin *et al.* [42]. For the LCA dataset, as no such information was provided, the boxes used as input for training were generated automatically. For each dataset, we report the accuracy of our method on the test videos with boxes produced automatically by our learned models through simultaneous tracking and action recognition. We also report classification accuracy with manually annotated boxes on the Weizmann, KTH, UCF Sports, and Office11 datasets, in order to observe performance in the absence of difficulties caused by tracking. As no such manual boxes are available for the LCA dataset, we also report results using boxes obtained with the same method used to

obtain the training boxes. We also report results with models which were trained without the use of additional bounding boxes produced by mining for high scoring tracks with the Event Tracker during training. This last number is included in order to show the importance of such negative training data. Without it, performance is much worse than any of the other baselines on almost every dataset.

2.6.1 Classification Accuracy

Weizmann Dataset

The Weizmann dataset consists of 90 low resolution video clips. There are 10 classes, each depicted once by each of 9 different actors, wearing a variety of different clothing. Experiments are done with 9-fold leave-one-actor-out cross validation. These videos have a clean white background and a single visible person, who performs the action. As noted in Tian *et al.* [30], many methods which report results on this dataset depend on this clean background to obtain a clean mask of the actors' silhouette by performing background subtraction as a preprocessing step.

Like Tian *et al.* [30] we do not rely on such properties. We instead perform simultaneous recognition and localization with our general-purpose system. Table 2.1 compares our results against recent methods on this dataset. Our results with automatic tracking are almost identical to those with manual tracks, both making very few mistakes and outperforming several recently published methods. While we do not achieve perfect results like Tian *et al.* [30], we significantly outperform the result they report without deformable parts. As Tian *et al.* [30] is among the most similar methods to this work, particularly the version without parts, this is an interesting result.

Table 2.1.

Classification accuracy of the current method compared with recent prior methods (all published since 2009) on the Weizmann, KTH, UCF Sports, LCA, and Office11 datasets. Our numbers are uniformly reported to one decimal place. Prior results are uniformly reported to the published precision.

Weizmann	Accuracy (percent)
Tian *et al.* (2013) [30] (with parts)	100
Oreifej & Shah (2014) [12]	92.8
Tian *et al.* (2013) [30] (without parts)	92.4
Nagar & Agrawal (2014) [43]	92
current manuscript (automatic tracks)	96.7
current manuscript (manual tracks)	97.8
current manuscript (without negative mining)	12.2
HOG baseline	75.5
HOF baseline	92.2
3-Stage-Baseline	43.3

KTH	Accuracy (percent)
Yao *et al.* (2014) [31] (with annotated parts)	94.53
Yao *et al.* (2014) [31] (without annotated parts)	84.70
current manuscript (automatic tracks)	94.9
current manuscript (manual tracks)	91.2
current manuscript (without negative mining)	44.9
HOG baseline	62.5
HOF baseline	82.8
3-Stage-Baseline	59.7

UCF Sports	Accuracy (percent)
Sadanand & Corso (2012) [44]	95.0
Yuan *et al.* (2013) [10]	92.67
Wu *et al.* (2011) [7]	91.3
Wang *et al.* (2013) [8]	90.22
Oreifej & Shah (2014) [12]	89.7
Wang *et al.* (2013) [13]	89.1
Everts *et al.* (2013) [9]	85.7
Yuan *et al.* (2013) [11]	87.33
Tian *et al.* (2013) [30] (with parts)	75.2
Tian *et al.* (2013) [30] (without parts)	64.9
current manuscript (automatic tracks)	82.0
current manuscript (manual tracks)	94.0
current manuscript (without negative mining)	38.6
HOG baseline	48.6
HOF baseline	48.6
3-Stage-Baseline	38.3

LCA	Accuracy (percent)
Sadanand & Corso (2012) [44]	16.667
Wang *et al.* (2013) [45]	15.556
Wang *et al.* (2013) [13]	14.074
Kuehne *et al.* (2011) [6]	9.259
Cao *et al.* (2013) [46]	7.592
Le *et al.* (2011) [47]	6.667
Ryoo (2011) [48] as reimplemented by [46]	6.667
Messing *et al.* (2009) [49]	6.296
current manuscript (automatic tracks)	14.3
current manuscript (pregenerated tracks)	11.8
current manuscript (without negative mining)	5.5
HOG baseline	6.3
HOF baseline	12.0
3-Stage-Baseline	7.6

Office11	Accuracy (percent)
Sadanand & Corso (2012) [44]	87.2
Kuehne *et al.* (2011) [6]	73.2
current manuscript (automatic tracks)	93.3
current manuscript (manual tracks)	98.3
current manuscript (without negative mining)	21.8
HOG baseline	33.5
HOF baseline	73.2
3-Stage-Baseline	24.5

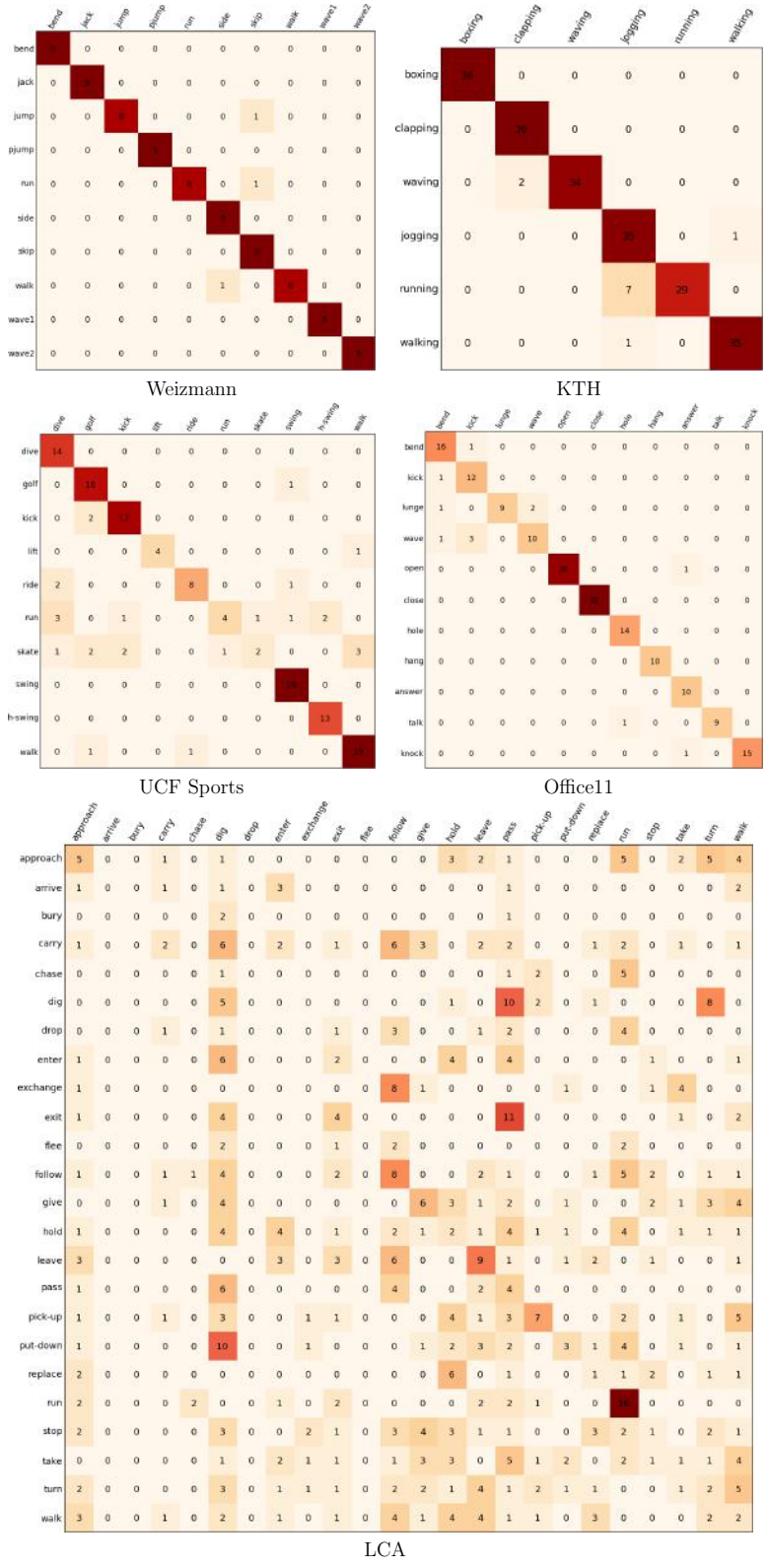


Fig. 2.5. Confusion matrices for the current method with automatic tracks on each dataset.

KTH Dataset

We evaluate our method on the KTH dataset in order to compare against Yao *et al.* [31], the most recent and most similar method. The KTH dataset consists of videos depicting six human action classes: *boxing*, *hand clapping*, *jogging*, *running*, *walking*, and *hand waving*. Each action is performed several times by 25 different people each in four scenarios: indoors, outdoors, outdoors while wearing different clothes, and outdoors with scale variation caused by camera zoom. This yields a total of 600 videos and 2391 action instances. The dataset provides a split into training, validation, and testing videos. Like Yao *et al.* [31], we use the training and validation sets for training, and test on the test videos. While the training videos each contain several instances of each action, they are quite similar, so we need only use the first from each video.

Table 2.1 compares our performance against that of Yao *et al.* [31]. The method of Yao *et al.* [31] depends on manual annotation of bounding boxes in the training videos, as we do, but also uses manual annotation of part locations. They report results both with manual annotation of part locations in the training videos, and without. We outperform both numbers, despite not using all the training data. In particular, we greatly outperform them when they only have bounding-box annotations for training, as we do. Since they use clustering on the part locations to determine how the frames of training videos should be broken up into distinct poses, the fact that we greatly outperform them given the same level of manual supervision shows that our method is better able to automatically learn the relevant characteristics of the actions.

UCF Sports Dataset

The UCF Sports dataset consists of 150 videos obtained from the web. This dataset also contains 10 classes. The videos in UCF Sports are much more difficult than those of Weizmann, containing substantial camera movement, widely variant backgrounds and viewpoints, and occasionally low temporal resolution. It also in-

cludes scene changes which result in the actor sometimes jumping significantly in the image from frame to frame. As noted earlier, it contains significant correlation in background between videos of the same class.

Previous work often uses leave-one-out cross-validation. It has been shown in Lan *et al.* [50] that the leave-one-out setting allows methods to take advantage of scene correlation between training videos. We instead use 16-fold cross-validation. Doing so reduces the amount of training data available, but our method performs well despite this. Table 2.1 compares our results to recently published work. Our result with manual tracks outperforms all previous work except Sadanand & Corso [44], but the result with automatic tracks only outperforms Tian *et al.* [30]. The camera movement, scene changes, and sometimes low temporal resolution make tracking in this dataset very difficult. Many of these other methods are BOW based: they do not perform localization, and can take advantage of the correlation in background, which we ignore by design. However, our results with automatic tracking are still good, and critically, outperform Tian *et al.* [30].

Because Tian *et al.* [30] also attempt to localize actions and recognize them based on the appearance and motion of the actor, this comparison is worth discussing further. As on the Weizmann dataset, they report results both with part models, and without, showing that they perform much better with such part models. As our method does not use part models, it is more similar to the version that lacks parts. On this dataset, we outperform both numbers. This suggests that our method of closely tracking the actor so that the models are centered on the action better handles the variation present in UCF Sports than the cuboid models of Tian *et al.* [30].

LCA Dataset

The LCA dataset is very difficult, consisting of a 13 hour subset of the video produced by DARPA for year 2 of the Mind's Eye program, designed to simulate a ground-based surveillance task. It contains 24 classes (*approach*, *arrive*, *bury*, *carry*,

chase, *dig*, *drop*, *enter*, *exchange*, *exit*, *flee*, *follow*, *give*, *hold*, *leave*, *pass*, *pick up*, *put down*, *replace*, *run*, *stop*, *take*, *turn*, and *walk*) in a number of outdoor environments depicted from a variety of viewpoints and scales, and often includes several people simultaneously moving in the same video, partially occluding one another, and performing other actions not in the list of 24 classes. Being designed to emulate surveillance, there is no camera movement. All the action classes occur in each background environment, so that the background gives little to no clue as to the action. This is a very difficult dataset, despite the lack of camera motion. A recent paper [41] compares the results of 8 recent methods. No method surpasses 17% accuracy.

As no bounding box information is included with LCA, we used automatically generated boxes to initialize training. This was done using our tracker with a generic object proposal method [51] and a motion prior. To bias the tracker towards moving objects, the mean optical-flow magnitude of each proposal was added to its score to produce the detection score f used by the tracker. The single highest scoring track as determined by the Viterbi algorithm was used as the initial positive track for each video. This simple tracker has no information about the actions, and therefore has no way to determine which visible person to track when, as is often the case in the LCA dataset, several people are simultaneously visible and moving. Therefore, these initial tracks are quite poor, often tracking the wrong person. However, our method can handle these noisy training detections. The use of the Event Tracker to repeatedly resample the positive tracks according to the partially learned action models during training allows our method to recover from this, and perform competitively, as can be seen in Table 2.1.

We report results both using tracks for the test videos which were pregenerated in the same manner described for the training videos and using the learned models to automatically localize the action with the Event Tracker. The results using the Event Tracker outperform those with pregenerated tracks, because the learned action models make it possible to determine which person is performing the action. Our method performs better than several methods, including Dense Trajectories [13], and

performs at a similar level with Improved Trajectories [45] and Action Bank [44]. These four top methods significantly outperform the rest, with all others besides C2 [6] performing at a near chance level. This shows that our method is able to match the state of the art on this difficult dataset, even with no manual annotation on the training videos, using only noisy automatically generated boxes.

Office11 Dataset

As described previously, Office11 consists of 179 videos, depicting 11 classes. It contains both human-pose related actions as well as manipulation of objects. We performed a comparison experiment between our method and two top-performing methods for which software is available: Action Bank [44] and C2 [6]. Action Bank performs the best on both the UCF Sports and LCA datasets, and so provides a strong comparison. We performed an identical 5-fold cross-validation experiment on this dataset with both these methods and our own. Results are shown in Table 2.1. Our method outperforms both other methods on this dataset.

2.6.2 Localization

We evaluated the localization accuracy of our method, separate from classification accuracy. It is evaluated by computing the Intersection-over-Union (IoU) measure between the track produced by the highest scoring model and the ground-truth track. We report the accuracy of localization as the fraction of test videos with IoU above a certain threshold. Fig. 2.6 shows the localization accuracy of our method on each of the datasets as a function of this threshold, and compares this against that produced by the TLD tracker. Results are not reported for LCA because ground-truth is not available. Fig. 2.7 visualizes localization examples which cannot be produced by low-level tracking systems. Our method outperforms the TLD tracker on every dataset. The performance of our system is somewhat similar to that of TLD on the easier to

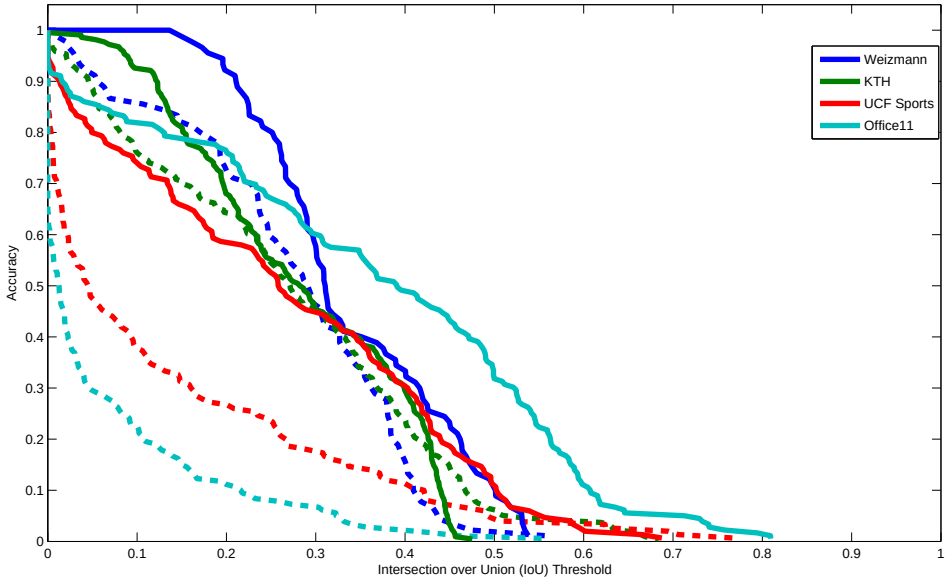


Fig. 2.6. Localization accuracy as a function of Intersection-over-Union (IoU) threshold. Comparison between the presented method (solid lines) and the TLD tracker initialized with the DPM object detector (dashed lines)

track Weizmann and KTH datasets, but is dramatically superior on the more difficult UCF Sports and Office11 datasets.

2.7 Discussion

Our method outperforms both Tian *et al.* [30] and Yao *et al.* [31], the two most similar methods, both of which were published recently. It also outperforms two state-of-the-art methods on our new dataset Office11, and performs competitively in the very difficult LCA dataset. It performs well with both manual and automatic tracking. On the KTH dataset, classification performance is actually higher than with manual tracking. This is likely due to the manually annotated tracks being aligned poorly with the person in some videos, a problem which is solved by performing automatic tracking. On the LCA dataset, it performs well despite noisy bounding boxes used for training, and also performs better with automatic tracking than with

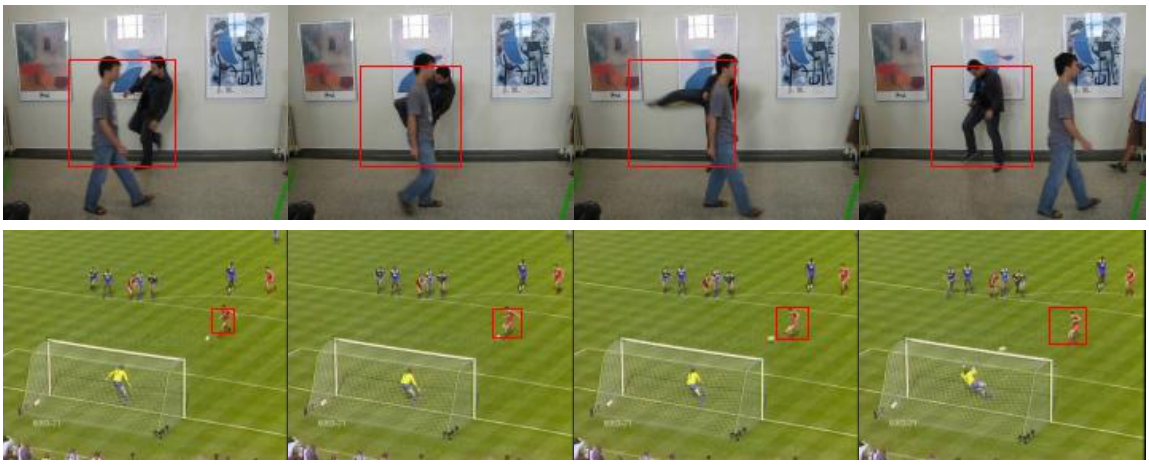


Fig. 2.7. Example tracks automatically produced by our system on unseen test video. The top example, from Office11, shows successful tracking of a *kick* despite complete occlusion. A tracking system unaware of the action would not produce this track because it is highly sub-optimal according to low-level criteria. The bottom example, from UCF Sports, shows successful tracking of the single person performing the *kick* action out of many other visible and moving people. Low-level tracking systems have no mechanism to choose that particular person to track.

predefined tracks obtained with a tracker unaware of the action class. Evaluation of localization shows that lower classification performance on the UCF Sports dataset with automatic tracks when compared to manual tracks may be due to the difficulty in tracking the people performing the actions in that dataset. Our results also show the importance of mining for difficult negative tracks in the training data, rather than using only instances of other actions as negative samples. Without this additional negative training data, our method learns models which perform extremely poorly. This suggests that other methods such as Tian *et al.* [30] or Yao *et al.* [31] may benefit from a similar step during training.

2.8 Conclusion

We have presented a method for automatically learning the changing appearance and motion patterns of actions in video. It accomplishes this by automatically identifying the most discriminative temporal subsequences of the action classes, and training sequences of appearance and motion detectors to maximize the training classification margin. The learned sequences of detectors are shown by their visualizations to be intuitively meaningful representations of the actions. These learned models can then be used to perform simultaneous recognition and localization of actions in new video. This method has been shown to perform competitively with the state of the art in action recognition on several datasets. In particular, it outperforms two recent methods which also attempt to recognize and localize actions by appearance and motion.

Acknowledgments

This research was sponsored, in part, by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0060. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either express or implied, of the Army

3. DRIVING UNDER THE INFLUENCE (OF LANGUAGE)

We present a unified framework which supports grounding natural-language semantics in robotic driving. This framework supports acquisition (learning grounded meanings of nouns and prepositions from human sentential annotation of robotic driving paths), generation (using such acquired meanings to generate sentential description of new robotic driving paths), and comprehension (using such acquired meanings to support automated driving to accomplish navigational goals specified in natural language). We evaluate the performance of these three tasks by having independent human judges rate the semantic fidelity of the sentences associated with paths. Overall, machine performance is 74.9%, while the performance of human annotators is 83.8%. With recent advances in machine perception and robotic automation, it becomes increasingly relevant and important to allow machines to interact with humans in natural language in a *grounded fashion*, where the language refers to actual things and activities in the world. Here, we present our efforts to automatically drive—and learn to drive—a mobile robot under natural-language command. Our contribution is summarized in Fig. 3.1. A human teleoperator is given a set of sentential instructions designating robot paths. The operator then drives a mobile robot under radio control according to these instructions through a variety of floorplans. The robot uses onboard odometry and inertial guidance sensors to determine its location in real time and saves traces of the driving paths to log files. Visualizations of these paths are then placed on Amazon Mechanical Turk (AMT), where anonymous workers are asked to provide sentential descriptions. From a training corpus of these paths paired with their corresponding sentential descriptions and floorplan specifications, our system automatically learns the meanings of nouns that refer to objects in the floorplan and prepositions that describe both the spatial relations between floorplan objects and

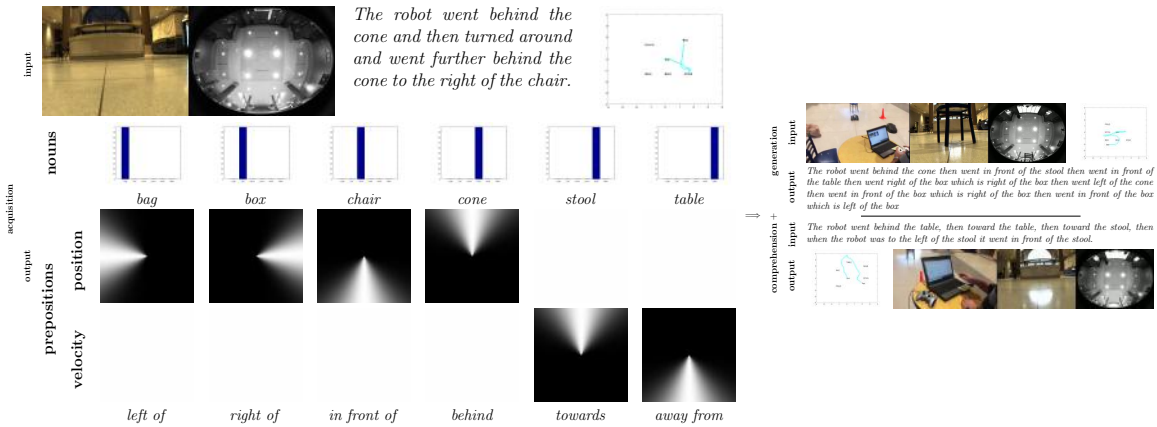


Fig. 3.1. (left) A human drives the mobile robot through paths according to sentential instruction while odometry reconstructs the robot's paths. Natural-language descriptions of these paths are obtained from AMT. This allows the robot to learn the meanings of the nouns and prepositions. Hand-designed word models are shown here for illustration; actual learned word models are shown in Fig. 3.12. Note that the distributions are uniform in velocity angle (bottom row) for *left of*, *right of*, *in front of*, and *behind* and in position angle (top row) for *towards* and *away from*. These learned meanings support generation of English descriptions of new paths driven by teleoperation (top right) and autonomous driving of paths that meet navigational goals specified in English descriptions (bottom right).

between such objects and the robot path. With such learned meanings, the robot can then generate sentential descriptions of new driving activity undertaken by a teleoperator. Moreover, instead of manually controlling the robot through teleoperation, one can issue the robot natural-language commands which induce fully automatic driving to satisfy the path specified in the natural-language command.

We have conducted experiments with an actual radio-controlled robot that demonstrate all three of these modes of operation: acquisition, generation, and comprehension. We demonstrate successful completion of all three of these tasks on hundreds of driving examples. We evaluate the fidelity of the sentential descriptions produced automatically in response to manual driving and the fidelity of the driving paths induced automatically to fulfill natural-language commands, by presenting the pairs of sentences together with the associated paths to anonymous human judges on AMT. For machine generated results overall, the average "sentence correctness" (the de-

gree to which the sentence is true of the path) reported is 74.2%, the average "path completeness" (the degree to which the path fully covers the sentence) reported is 76.0%, and the average "sentence completeness" (the degree to which the sentence fully covers the path) reported is 74.5%, for an average of 74.9%.

3.1 Related Work

While there is prior work which learns word meanings, both in the context of robot navigation and in other contexts such as robotic manipulator arms, there is no prior work which learns word meanings from robotically driven paths annotated only with sentences, combines these words to form new sentences correctly describing newly driven paths, and paths and physically drives new paths satisfying newly input sentences.

Those prior works [52–59] which perform word learning do so within discrete simulations and make use of discrete symbolic primitives from those simulations, such as MOVE FORWARD N STEPS or DRIVE TO LOCATION 1. In contrast, our method operates in continuous space, and learns intuitive word meanings in terms of the relative positions and velocities as measured from sensor data, rather than simple mappings to predefined logical primitives. Since not every point in a path is described, and the correspondence between phrases and path segments is unknown, our system performs automatic alignment between the temporal sequence of measured robot path positions and the sequence of temporal segments in a sentence, without the additional annotation required by prior work.

Some prior work (*e.g.*, [60,61]) perform learning in the context of a physical robot, but do not perform all three tasks, and still make use of pre-defined logical primitives like *A is near B*. These methods do not learn word meanings that are combined to produce or describe new paths. Some other work on natural language interaction with robots (*e.g.*, [62–67]) does not perform any learning.

There is other work on the topic of natural-language interaction with robots (*e.g.*, [62–67]), both within and outside the realm of robotic navigation, however, such work does not involve any learning. Some prior work (*e.g.*, [68–71]) performs learning in the context of robotics, but not robotic navigation. For example, McGuire *et al.* [68] interact with a robotic arm via speech and hand gestures, and learn to associate certain tasks with certain commands, and Matuszek *et al.* [70] learn a mapping between color and shape classifiers operating on RGB-D camera data and words like *red* and *square*. These are static properties of individual objects, rather than dynamic properties and complex relationships among objects and the robot as our system uses.

3.2 Experimental Platform

Experiments were conducted using a small wheeled robot. This robot makes use of odometry, inertial guidance, and an Extended Kalman Filter [72] to localize itself in real-time, and to log a densely sampled sequence of such positions for later use. Real-time localization is used in conjunction with the sentence-comprehension and planning algorithm to automatically drive paths given an English-language sentential description, while the logging of the position information supports the learning and path description algorithms.

3.3 Extracting Meaning from a Sentence

This paper concerns itself with semantics, not syntax, and only addresses issues relating to the grounding of word meanings. We represent the meaning of a sentence describing a robot path as a sequence of graphical models composed of probability distributions corresponding to word meanings. Each such graphical model represents the meaning of a clause describing the path at a particular point in time. Each graphical model is a factorized joint distribution over a set of variables: a *path variable*, which is a pair of 2D vectors representing the position and velocity of the robot at

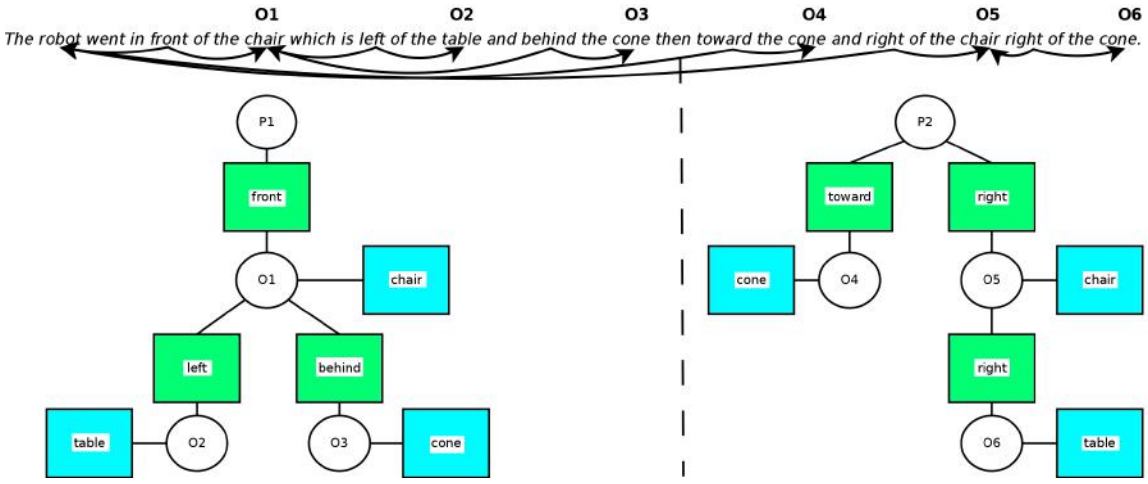
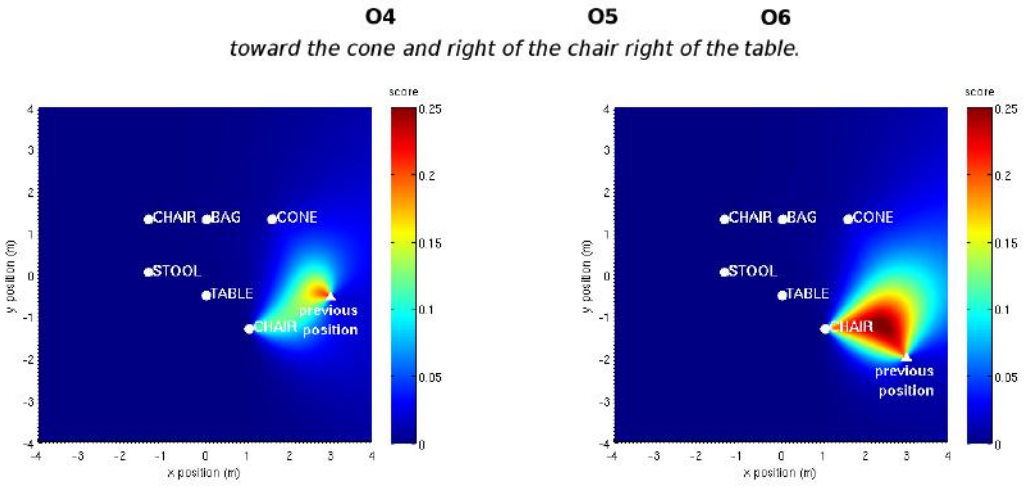


Fig. 3.2. Illustration of the sequence of graphical models induced by a sentence. The sentence is broken into sequential segments, and a path variable (P1, P2) is created for each segment. Next, a floorplan variable (O1–O6) is created for each noun in each segment, applying the noun's label distribution (in blue) to the variable's set of labels. Finally, the arguments of each preposition are found, and each preposition's distributions (in green) over relative positions and velocities are applied between its arguments.



previous position: (3.0, -0.55) previous position: (3.0, -2.0)

Fig. 3.3. Illustration of the score function induced by the sentence-segment graphical model from the right side of Fig. 3.2, using the word models obtained from the learning process. The graphical model is marginalized over all possible mappings from floorplan variables to objects, yielding a scoring function over the position and velocity of the path variable. The velocity is computed as the difference between the position at two adjacent time steps, so that, given the position of the robot at the previous time step, the function can be plotted at each point in space. The score function corresponding to the graphical model is plotted for two different previous positions: the point (3.0, -0.55) (left) and the point (3.0, -2.0) (right). Note that the differing positions for the previous position drastically change the function. In the image on the right, the function prefers points directly between the cone and the previous position, thus satisfying the *towards* requirement, which are also to the *right* of the bottom-most chair. In the left image, the previous position is such that there is no point both between it and the cone and directly to the right of the chair. The optimal point is therefore to move toward the cone, biased somewhat to the right, thus partially satisfying *right of the chair*. This point has a much lower score that the optimal point on the right plot. Also note that the scoring function correctly prefers points to the right of the chair described in the phrase, and not the other chair or other objects. This is because those mappings of floorplan variable O5 to other objects have a score close to zero at all points. The noun distribution associated with O5 results in low score when the label of the object mapped to O5 is not CHAIR, and the distribution induced by the phrase *right of the table*, results in low score for any mapping for which the object mapped to O5 is not to the right of O6, and for any mapping for which the label of the object mapped to O6 is not TABLE. Therefore, with the learned word models, only those mappings of floorplan variables to the proper objects significantly influence the score.

that particular time, and a set of *floorplan variables*, which are labeled 2D Cartesian coordinates representing the class and position of each object in a floorplan.

3.3.1 Constructing graphical models from a sentence

We automatically generate such a sequence of graphical models directly from a sentence. Fig. 3.2 illustrates this process. Each noun induces a floorplan variable with a univariate distribution over possible object labels that it may take. Each preposition induces a joint distribution between its target and referent objects. Prepositions may be used adverbially to describe the motion of the robot in relation to objects in the floorplan, or adjectivally to describe the static position of an object relative to other objects.

The task of constructing a sequence of graphical models from a sentence has three parts: breaking the sentence into its temporally sequential parts, identifying nouns and prepositions, and determining the arguments to each preposition. Off-the-shelf semantic parsers such as the Stanford parser [73] produce largely erroneous parse trees on our corpus of sentences, and thus cannot be used directly for this purpose. We do, however, use the Stanford parser to perform part-of-speech tagging. Fasola & Mataric [67] similarly use the Stanford parser solely for part-of-speech tagging.

To break a sentence into its temporal segments, we identify all verbs that do not immediately follow a WH-determiner (*i.e.*, *which*, *that*, *etc.*), as well as adverbial transition words (*i.e.*, *then*) and use them as the segment boundaries. We next identify prepositions and nouns of interest using a combination of the Stanford parser and the list of prepositions from Wikipedia [74]. For precise details on how the lexicon is determined, see Section 3.5.2.

Identifying the arguments to prepositions is done with a small number of rules. The first argument of a preposition may be either a path variable or a floorplan variable corresponding to a noun within the same temporal segment and preceding the preposition. If neither preceded by a conjunction, nor a comma, or if preceded by

a WH-determiner, but not a conjunction, the first argument is considered to be the immediately preceding noun. If the preposition is immediately preceded by a conjunction and there are no preceding WH-determiners in the temporal segment, then the first argument is considered to be the path variable. If the preposition is immediately preceded by a conjunction followed by a WH-determiner (*e.g.*, *and which*) or is immediately preceded by a conjunction and there is no WH-determiner preceding the preposition in the temporal segment, then the first argument is considered to be the noun prior to the immediately preceding noun. There are cases where attachment ambiguity exists. In such cases, the first argument is assumed to be the path variable. The second argument to a preposition is considered to be the noun immediately following the preposition. If there is a temporal break or another preposition between a preposition and the nearest following noun, the preposition is ignored.

Once the arguments to each preposition in a temporal segment have been found, the graphical model is formed as a product of the factors associated with each of the nouns and prepositions. Given a floorplan specifying the positions and class labels of objects, along with the distributions for each noun and preposition, each such graphical model represents the probability that a point in 2D space satisfies the semantics of the corresponding temporal segment of the sentential description. Fig. 3.3 illustrates the distribution over points in 2D space induced by such a graphical model.

3.3.2 Representation of the lexicon

The lexicon specifies the meanings of the nouns and prepositions as a set of probability distributions. The nouns are represented as discrete distributions over the set of class labels. These labels are abstract symbols corresponding to object classes, such as might be obtained by grouping object detections according to class with a clustering algorithm on sensor data. In this work, we do not perform such detection and clustering, but provide such data in the form of a floorplan. For a given floorplan,

the robot is provided a list of objects, each of which has a class label and 2D location. The class labels are consistent across floorplans. For example, objects of class **bag**, might have class label CLASS0, while objects of class **stool**, might have label CLASS4. A discrete distribution of weights w_{ij} is learned for each noun i in the lexicon which scores the mappings between it and each possible label j.

Each floorplan variable generated from a sentence can be mapped to one of the objects in a floorplan. When mapped to the to the kth object, whose label is l_k and which resides at location (x_k, y_k), the score of the noun distribution i applied to that variable is w_{i,l_k} Because a given floorplan often has multiple instances of objects of the same class, these class labels do not uniquely identify the objects. These would be disambiguated with complex noun phrases such as *the chair which is right of the stool* and *the chair which is left of the cone*. Such disambiguating prepositional phrase modifiers of noun phrases can be nested and conjoined arbitrarily. Similarly, the locations of path variables can be disambiguated by conjunctions of prepositional phrase adjuncts.

Prepositions specify relations between target objects and reference objects. When used adjectivally, the reference object is the object of the preposition while the target object is the head noun, thus applying the preposition distribution between two floorplan variables. For example, in *the chair to the left of the table*, *chair* is the target object and *table* is the reference object. When used adverbially, the target object is a waypoint in the robot path while the reference object is the object of the preposition, thus applying the preposition distribution between a path variable and a floorplan variable. For example, in *went towards the table*, *table* is the reference object, and the target object is a waypoint in the robot path. The lexical entry for each preposition is specified as the location μ and concentration κ parameters for two independent von Mises distributions [75] over angles between target and reference objects. One, the *position angle*, is the orientation of a vector from the coordinates of the reference object to the coordinates of the target object (Fig. 3.4 left).[1] The second, the *velocity*

[1]Without loss of generality, position angles are measured in the frame of reference of the robot at time zero, which is taken to be the origin.

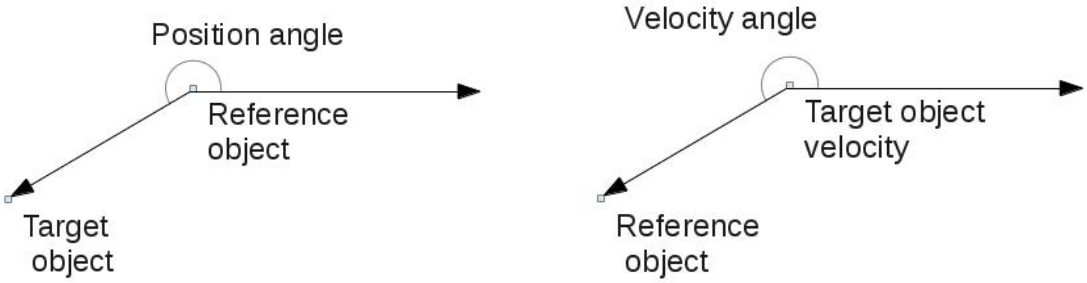


Fig. 3.4. How position angles (left) and velocity angles (right) are measured.

angle, is the angle between the velocity vector at a waypoint and a vector from the coordinates of the waypoint to the coordinates of the reference object (Fig. 3.4 right). This second angle is only used for adverbial uses, because it requires computation of the direction of motion, which is determined from temporally adjacent waypoints, and is undefined for stationary objects. This angle is thus taken from the frame of reference of the robot. The von Mises distribution $f(x|\mu,\kappa)$ is given by

$$f(x|\mu,\kappa) = \frac{e^{\kappa cos(x-\mu)}}{2\pi I_0(\kappa)}, \tag{3.1}$$

where I_0 is the modified Bessel function of order 0.

Fig. 3.1 (bottom left) illustrates how this framework is used to represent the meanings of prepositions. We render the angular distributions as potential fields around the reference object at the center for the position angle, and the target object at the center for the velocity angle. The intensity of a point reflects its probability density. Note that in the idealized distributions of Fig. 3.1, the distributions are uniform in velocity angle for *left of*, *right of*, *in front of*, and *behind* and in position angle for *towards* and *away from*.

When the ith preposition in the lexicon is applied between two variables, whose physical relationship is specified by the position angle θ and velocity angle γ between them, its score s_i is given by

$$s_i(\theta, \gamma) = \left(\frac{e^{\kappa_{i,1} cos(\theta - \mu_{i,1})}}{2\pi I_0(\kappa_{i,1})} \right) \left(\frac{e^{\kappa_{i,2} cos(\gamma - \mu_{i,2})}}{2\pi I_0(\kappa_{i,2})} \right), \tag{3.2}$$

where $\mu_{i,1}$ and $\kappa_{i,1}$ are the location and concentration parameters of the position angle distribution of the ith preposition, and $\mu_{i,2}$ and $\kappa_{i,2}$ are the location and concentration parameters of the velocity angle distribution.

3.3.3 Computing the graphical model score

Once constructed from a sentence segment, each graphical model induces a distribution over the path variable $p = (p^x, p^y, p^{v_x}, p^{v_y})$, conditioned on the O objects in the floorplan $\mathbf{f} = (o_1, o_2, \ldots, o_O)$ and the mapping m from the N floorplan variables to floorplan objects. Each element of the mapping m_n is the index of the floorplan object mapped to floorplan variable n. Let a be $\{p, o_{m_1}, \ldots, o_{m_N}\}$, a set consisting of the path variable and the floorplan objects mapped to each of the N floorplan variables. Further, let $b_{c,1}$ and $b_{c,2}$ be the indices in a of the target and referent, respectively, of the cth preposition in the graphical model. The 2D world position of the target and referent of the cth preposition can then be referenced with $(a^x_{b_{c,1}}, a^y_{b_{c,1}})$ and $(a^x_{b_{c,2}}, a^y_{b_{c,2}})$, respectively. The velocity vector of the target can similarly be referenced with $(a^{v_x}_{b_{c,1}}, a^{v_y}_{b_{c,1}})$. Therefore, the position angle of the target and referent of the cth preposition in the graphical model is given by

$$\theta_c = \tan^{-1} \frac{a^y_{b_{c,1}} - a^y_{b_{c,2}}}{a^x_{b_{c,1}} - a^x_{b_{c,2}}} \tag{3.3}$$

and the velocity angle γ_c between them is given by

$$\gamma_c = \tan^{-1} \frac{a^{v_y}_{b_{c,1}}}{a^{v_x}_{b_{c,1}}} - \tan^{-1} \frac{a^y_{b_{c,2}} - a^y_{b_{c,1}}}{a^x_{b_{c,2}} - a^x_{b_{c,1}}} \tag{3.4}$$

A sentence-segment graphical model's conditional probability $P(p|\mathbf{f}, m, \Lambda)$ of the path variable given an object mapping m, floorplan $\mathbf{f}$, and lexicon parameters Λ is therefore given by the product of preposition and noun scores:

$$P(p|m, f, \Lambda) = \prod_{c=1}^{C} s_{d_c}(\theta_c, \gamma_c) \prod_{n=1}^{N} w_{e_n, l_{m_n}} \tag{3.5}$$

where c indexes into the C prepositions in the graphical model, d_c is the index in the lexicon of the cth preposition in the graphical model, n indexes into the N nouns in the graphical model, e_n is the index in the lexicon of the nth noun in the graphical model, and l_{m_n} is the class label of the object mapped to the nth noun.

3.4 Tasks

We formulate sentential semantics as a variety of relationships between a sentence $\mathbf{s}$, or more precisely its corresponding sequence of graphical models, a path $\mathbf{p}$, which is a sequence of path waypoints, a floorplan $\mathbf{f}$, which is a set of labeled points, and a lexicon Λ, which is the collective μ and κ parameters for the angular distributions for each of the prepositions and the discrete distributions for each of the nouns. This allows us to accomplish the following tasks:

acquisition Learn a lexicon Λ from a collection of observed paths $\mathbf{p}_i$ taken by the robot in the corresponding floorplans $\mathbf{f}_i$ as described by human-generated sentences $\mathbf{s}_i$.

generation Generate a sentence $\mathbf{s}$ that describes an observed path $\mathbf{p}$ taken by the robot in a given floorplan $\mathbf{f}$ with a known lexicon Λ.

comprehension Generate a path $\mathbf{p}$ to be taken by the robot that satisfies a given human-generated sentence $\mathbf{s}$ issued as a command in a given floorplan $\mathbf{f}$ with a known lexicon Λ.

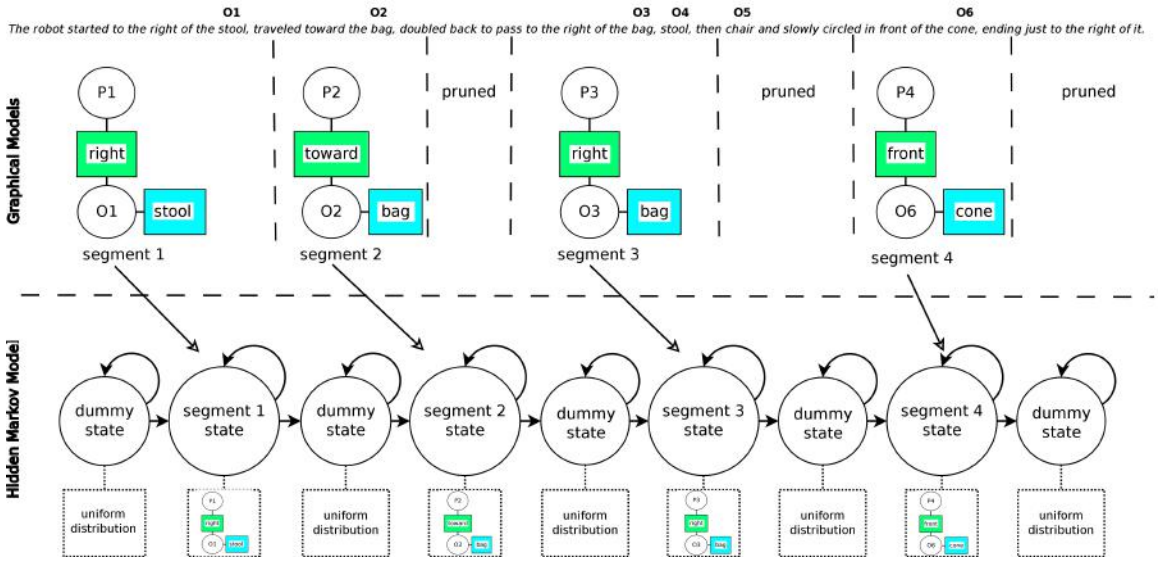


Fig. 3.5. A hidden Markov model is created representing the semantics of a sentence. The sentence is broken into segments, and a graphical model is created representing each segment. When a segment cannot be understood, it is pruned, and no graphical model is created. Next, an HMM state is created for each remaining segment. The output model of each such state represents the distribution over the possible positions and velocities of the robot at a given point in time. These output distributions are the graphical models associated with each segment, marginalized over the possible labelings of the floorplan variables. Additional dummy states with uniform output distributions are added at the beginning, end, and between each state. These dummy states allow the HMM to match paths for which the semantics of the sentence are true, but for which their are points in time where the robot does not fulfill any of the stated conditions. The HMM transition distribution encodes the sequence of the sentence by forcing each state to self transition or pass to the next state, as well as by requiring that the model begin in the first state and end in the last.

The robot started to the right of the stool, traveled toward the bag, doubled back to pass to the right of the bag, stool, then chair and slowly circled in front of the cone, ending just to the right of it.

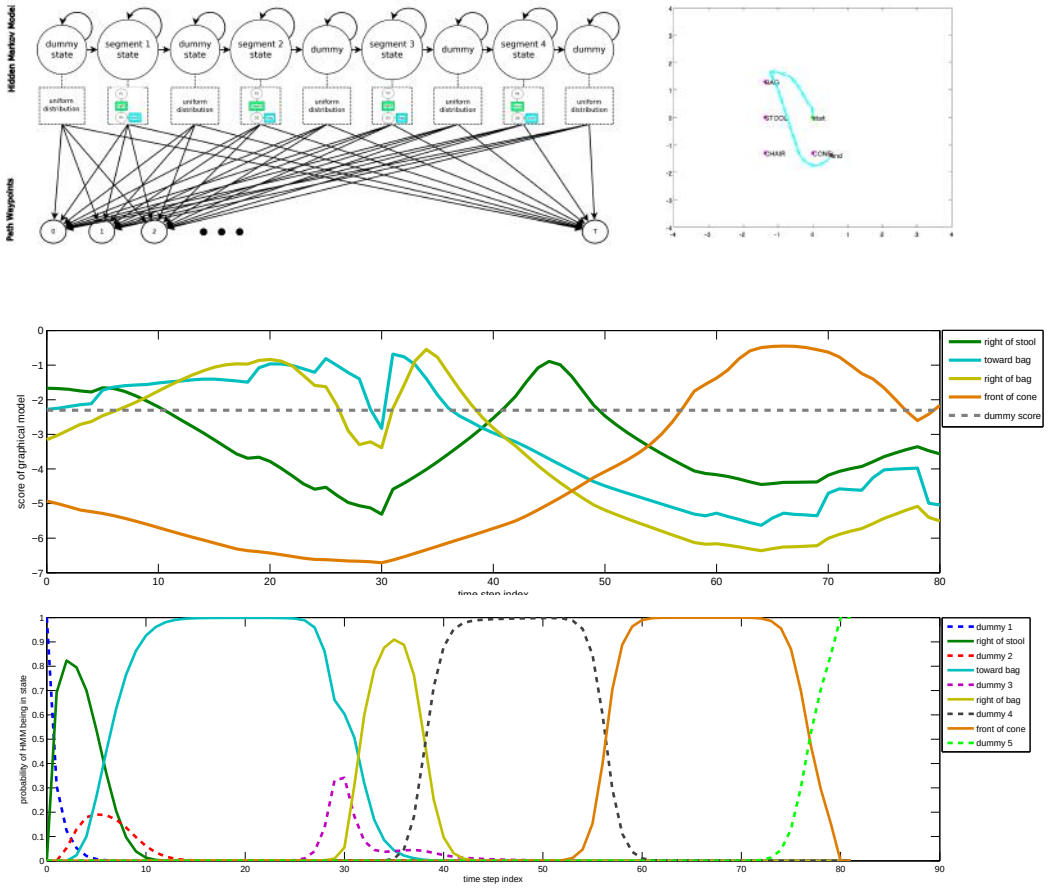


Fig. 3.6. Illustration of aligning an example sentence-path pair from the training set. An HMM is produced to represent the semantics of the sentence (top left). Given a floorplan and path (top right), the HMM is used during learning to perform an alignment between the states (and therefore the temporal segments of the sentence) and the densely sampled waypoints of the path. Each HMM output model computes the score of the position and velocity at each path waypoint (middle row). Because the preposition and noun distributions are unknown at the start of the learning process, the labels of the floorplan variables, which map nouns in the sentence to objects in the floorplans, are also unknown. Therefore each state's output score is the likelihood of the associated graphical model, marginalized over all possible mappings of floorplan variables to labels. These scores, along with the HMM transition model, are used with the forward-backward algorithm to compute the probability of the HMM being in each state (bottom row), along with the HMM likelihood. Prior to learning the word meanings, all preposition and noun distributions are random. During acquisition of such meanings, the model is updated to increase the overall HMM likelihood summed over all training samples. At each iteration, this concentrates the probability mass of the distributions associated with the prepositions for each HMM-state output model at those angles seen among waypoints and objects at those time steps at which the probability of the HMM being in that state is high. It also concentrates the probability mass of the object-label distributions in those bins associated with the mappings corresponding to high HMM likelihoods. This example shows the scores and HMM state-probability assignments using the word models after the learning process is complete.

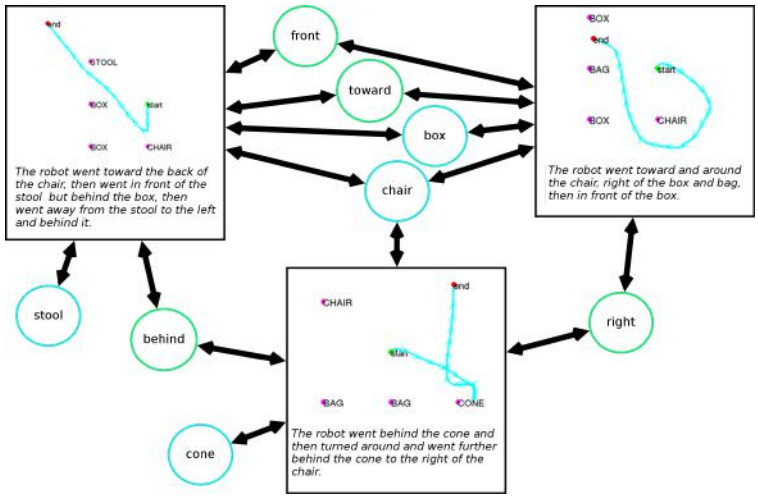


Fig. 3.7. Viewing the learning process as a constraint-satisfaction problem. Individual words appear across multiple path-sentence pairs. This allows inference across different words in the same sentence, where knowledge about one word constrains which points in the path or objects are described by or referred to by another. It also allows inference across multiple instances of the same word in the descriptions of different paths, where relationships among waypoints and objects in one sentence-path pair, whose description includes a particular word, constrain which relationships are referred to by that same word in the description of another path.

toward the cone and right of the chair right of the table

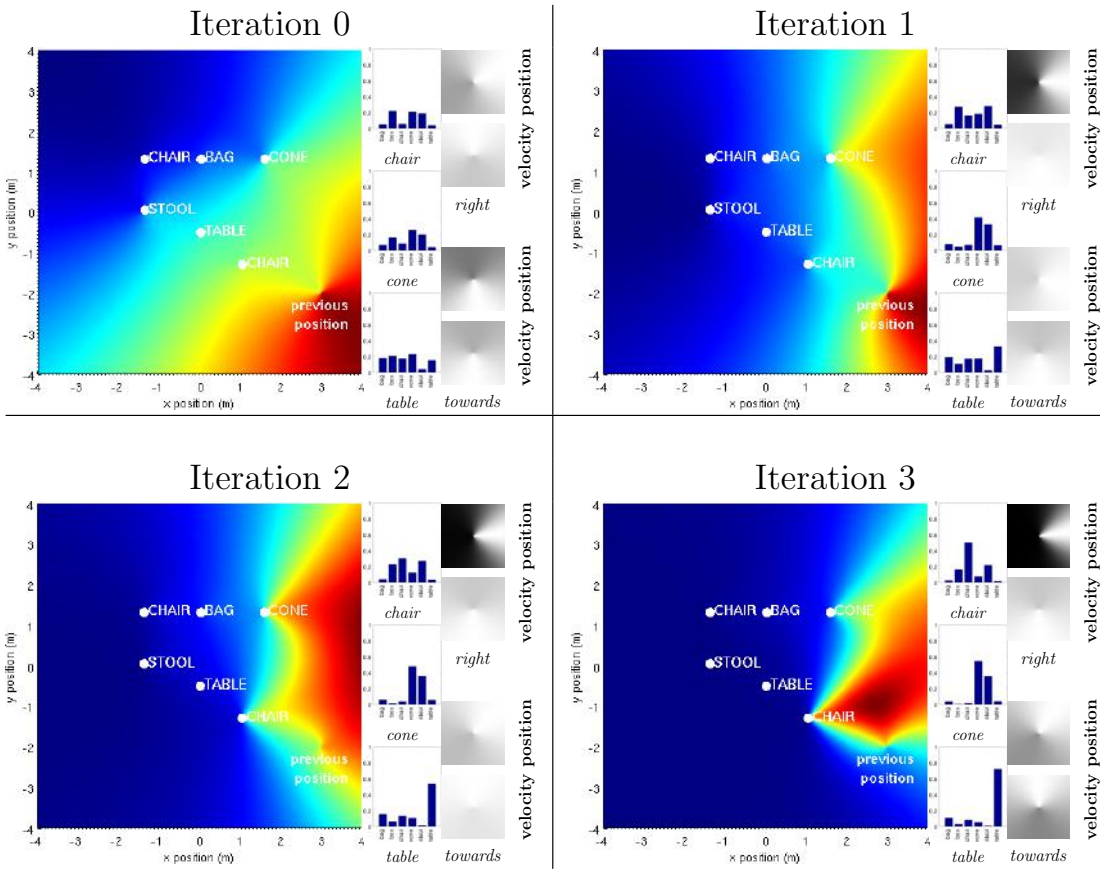


Fig. 3.8. Illustration of the word meanings and resulting scoring functions at different steps in the learning process. The scoring function corresponding to the same phrase illustrated in Fig. 3.3 is shown for the first four iterations of the learning process, along with the word models used in interpreting that phrase. Iteration 0 (top left) shows the randomly initialized word models, and the resulting score surface, which does not encode the meaning of the phrase at all. The noun distributions are largely uniform, resulting in no visible correlation between the score and the individual object positions. After the first iteration (top right), the noun models have just begun to concentrate in the correct bins, and the position distribution of the *right* model is beginning to concentrate in the correct direction. This change is evident in the score surface, which shows that it depends upon the position of the cone, but not in the correct way, as the *towards* model is still completely wrong. After the second iteration (bottom left), the noun distributions are further concentrated in the correct bins, and the *towards* velocity distribution is now pointed in the correct direction, although still almost uniform. The score surface now clearly depends on both the cone and the proper chair. After the third iteration (bottom right), the noun distributions are further concentrated, as are both the position angle distribution of the *right* model and the velocity angle distribution of the *towards* model. The cost surface now largely represents the meaning of the phrase, and already looks quite similar to that in Fig. 3.3, which is the result after convergence.

3.4.1 Acquisition

To perform acquisition, we formulate a large set of hidden Markov models (HMMs), one for each path-sentence pair in the training corpus. Each such HMM has a state k corresponding to every temporal segment its corresponding training sentence. The observations for each such HMM consist of the sequence of path waypoints in the path-sentence pair. The output model R_k for each state is the graphical model constructed from that temporal segment, given the current estimate of the parameters in Λ and marginalized over all mappings m between *floorplan variables* in the graphical model and objects in the floorplan.

$$R_k(p_t, \mathbf{f}, \Lambda) = \sum_m P_k(p|m, f, \Lambda) \tag{3.6}$$

The transition matrix for each HMM is constructed to allow each state only to self loop or to transition to the state for the next temporal segment in the training sentence. The HMM is constrained to start in the state associated with the first temporal segment in the sentence associated with each path. Dummy states, with a small fixed output probability, are placed between the states for each pair of adjacent temporal segments, as well as at the beginning and end of each sentence, to allow for portions of the path that are not described in the associated sentence. Fig. 3.5 illustrates the automatic construction of such an HMM from a sentence.

The HMMs are used to infer the alignment between the densely sampled points in each path and the sequence of temporal segments its corresponding sentence. This process is illustrated in Fig. 3.6.

The output models for the HMMs are all parametrized by the word meanings from the lexicon Λ. Thus, the meaning of each word is constrained by many path-sentence pairs. As illustrated in Fig. 3.7, this can be thought of as a large (soft) constraint-satisfaction problem. This mutual constraint allows the learning system to gradually infer the unknown mappings between points in the paths and the segments of sentences, and between nouns in the sentences and objects in the floorplans,

Fig. 3.9. Illustration of the generation algorithm. A disambiguating noun phrase is generated for each floorplan waypoint. Path waypoints are described by prepositional phrases, and then sets of identical phrases are merged into intervals, which are combined to form the sentence.

while simultaneously learning the parameters of the lexicon. Thus, it uses its current estimate of the word meanings to infer which physical relationships between the robot and the objects, or between several objects, are being described, and uses this knowledge to further update the word meanings in order to match the described relationships.

This learning is accomplished by maximizing the summation of the log likelihoods of all HMMs on their corresponding paths through Baum-Welch [24, 76, 77]. This trains the distributions for the words in the lexicon Λ as they are tied as components of the output models. Specifically, it infers the latent alignment between the large number of noisy robot path waypoints and the smaller number of temporal segments in the training descriptions while simultaneously updating the meanings of the words to match the relationships between waypoints described in the corpus. In this way, the meanings of both the nouns and the prepositions are learned. Fig. 3.8 illustrates the gradual learning process by showing how the scoring function corresponding to an example phrase begins in a completely meaningless state, but gradually changes to represent the meaning of that phrase as the meanings of the words are gradually learned.

3.4.2 Generation

Language generation takes as input a path $\mathbf{p}$ obtained by odometry during human teleoperation of the robot. This path consists of a collection of 2D floor positions sampled at 50Hz. To generate a sentence, one must select a subsequence of this dense sequence worthy of description.

During generation, we care about three properties: "correctness," that the sentence be logically true of the path, "completeness," that the sentence differentiate the intended path from all other possible paths, and "conciseness," that the sentence be the shortest that does so. We attempt to find a balance between these properties with the following heuristic algorithm (Fig. 3.9). First, we sample path waypoints in a way that the sampled points evenly distribute along the path. To this end, we downsample the path by computing the integral distance traveled from the initial position for each point in the dense path and selecting a subsequence whose points are separated by 5cm of integral path length. We then produce a prepositional phrase to describe each path waypoint by selecting that preposition with maximum posterior probability with the path waypoint as its first argument and with a floorplan waypoint as its second argument. Identical such choices for consecutive sets of waypoints in the path are coalesced and short intervals of such path prepositional phrases are discarded. We then generate a noun phrase for the object of each path waypoint preposition that refers to that referenced floorplan object. For each floorplan object, we take that noun with maximum posterior probability given the class of the floorplan object. Similarly, for each pair of floorplan objects, we take that preposition with maximum posterior probability to be true of that pair and all other prepositions applied to that pair to be false. Thus when the floorplan contains a single instance of a class, it can be referred to with a simple noun. But when there are multiple instances of a class, the shortest possible noun phrase, with one or more prepositional phrases, is generated to disambiguate.

More formally, let $q(o)$ be the most probable noun for floorplan object o given Λ. For each pair of floorplan objects (o, o'), there exists only one preposition ϕ that is true of this pair. Let $u(o)$ be the noun phrase we want to generate to disambiguate the floorplan object o from others o'. Then o can be referred to with $u(o)$ unambiguously if **(a)** $u(o) = (q(o), \{\})$ is unique; or **(b)**, there exists a collection of prepositional phrases $\{\phi(o, o')\}$ such that formula $u(o) = (q(o), \{(\phi, u(o'))\})$ is unique. To produce a concise sentence, we want the size of the collection of prepositional phrases in step **(b)** above to be as small as possible. However, finding the smallest collection of modifiers is NP-hard [78]. To avoid exhaustive search, we use a greedy heuristic that biases towards adding the least frequent pairs $(\phi, u(o'))$ into the collection until $u(o)$ is unique. This results in a tractable polynomial algorithm. After we get $u(o)$, we map it to a noun phrase by simple realization, for example:

$$\big(\textsc{table}, \big\{\big(\textsc{leftOf}, \textsc{chair}\big), \big(\textsc{behind}, \textsc{table}\big)\big\}\big)$$

$$\downarrow$$

the table which is left of the chair and behind the table

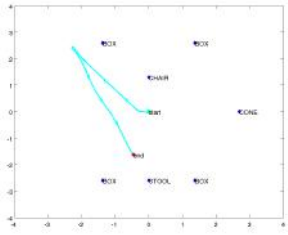
3.4.3 Comprehension

To perform comprehension, we use gradient ascent to optimize a scoring function with respect to an unknown path $\mathbf{p}$

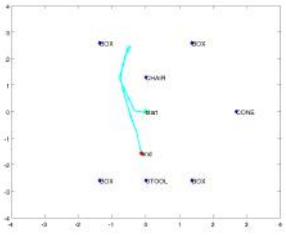
$$\mathbf{p}^* = \arg \max_{\mathbf{p}} \mathcal{R}(\mathbf{s}, \mathbf{p}, \mathbf{f}, \Lambda) \tag{3.7}$$

where $\mathcal{R}(\mathbf{s}, \mathbf{p}, \mathbf{f}, \Lambda)$ is the product of the graphical model likelihoods $P_k(p_k|m, f, \Lambda)$ from (3.5) constructed from the temporal segments of the sentence $\mathbf{s}$. The unknown path $\mathbf{p}$ is constructed to contain one path waypoint p_k for each temporal segment k in the sentence, whose locations are optimized to maximize the scoring function,

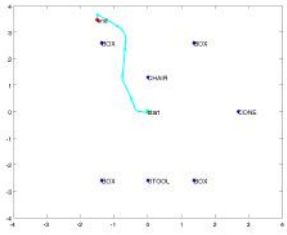
*The robot went away from the cone then went **left** of the box which is left of the chair and behind the cone then went towards the stool.*



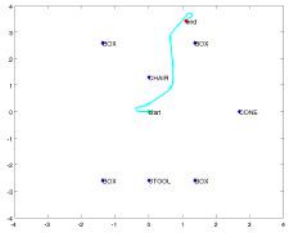
*The robot went away from the cone then went **right** of the box which is left of the chair and behind the cone then went towards the stool.*



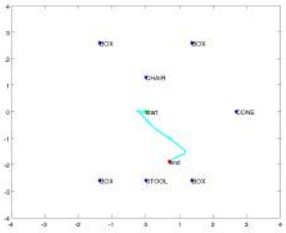
*The robot went away from the cone then went **behind** the box which is left of the chair and behind the cone then went towards the stool.*



*The robot went away from the cone then went behind the box which is **right** of the chair and which is **behind** the cone then went towards the stool.*



*The robot went away from the cone then went behind the box which is **right** of the chair and in **front** of the cone then went towards the stool.*



*The robot went away from the cone then went behind the box which is **left** of the chair and in **front** of the cone then went towards the stool.*

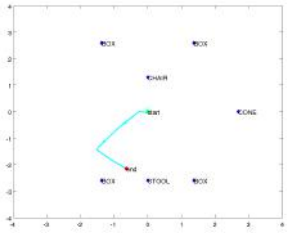


Fig. 3.10. Illustration of the effect on the comprehension system of single-word changes to the input sentence. The top row shows the effect of changing the preposition specifying the relation between the robot and a reference object between *left* (left), *right* (center), and *behind* (right). The bottom row shows the effect of changing the prepositions specifying which object is is being referred to. This allows the system to correctly distinguish between many objects of the same type, such as each of the four boxes in the example. As the prepositions *away from* and *toward* only specify the direction of motion, and are not specific about how far to go, the comprehension system sometimes chooses points which result in rather short distances traveled, particularly when there is an obstacle in the way, such as in the paths shown at the top-right and bottom-left.

toward the chair left of the bag

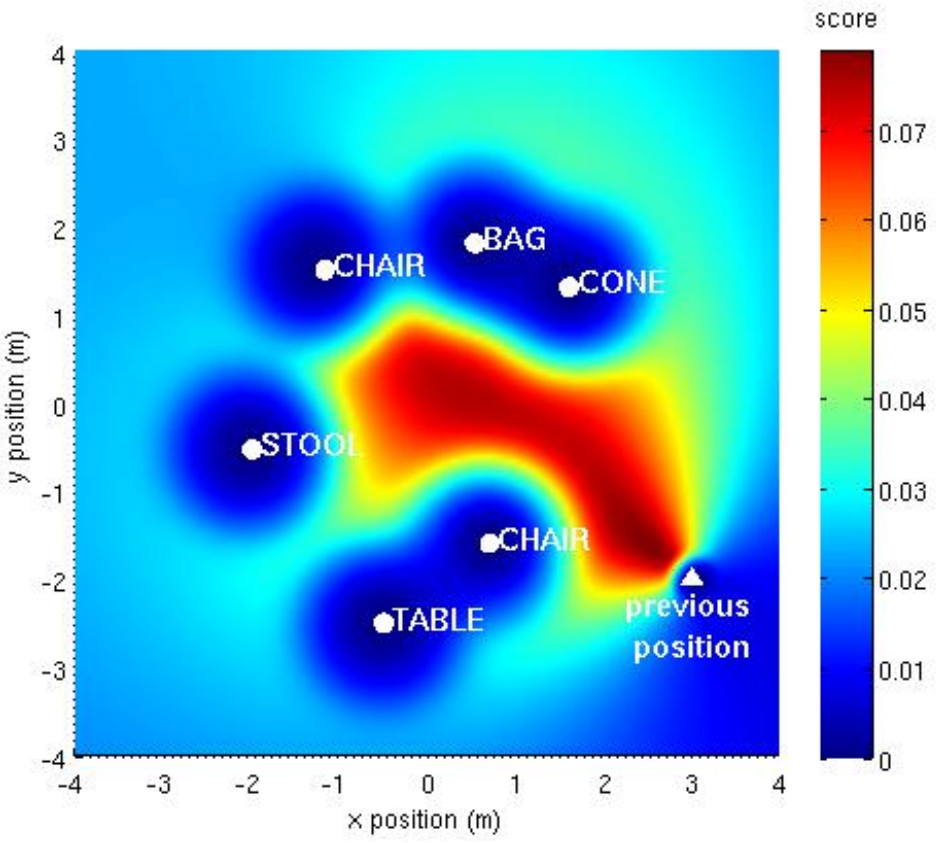


Fig. 3.11. Illustration of the scoring function after the addition of the barrier penalties, which keep the comprehension path waypoints away from the objects and from each other, and attraction terms, which encode preference for proximity to the target object.

and thus maximize the degree to which these waypoints satisfy the semantics of the sentence.

The above optimization computes a MAP estimate of the product of the likelihoods of the graphical models associated with the sentence $\mathbf{s}$. These graphical models represent the semantics of the sentence, but do not take into account constraints of the world, such as the need to avoid collision with the objects in the floorplan. Further, the scoring function as stated can be difficult to optimize because the velocity angle computed between two waypoints becomes increasingly sensitive to small changes in their positions as they become close together. To remedy the problems of the path waypoints getting too close to objects and to each other, additional factors are added to the graphical models. A barrier penalty $B(r)$ is added between each pair of a path waypoint and floorplan waypoint as well as between pairs of temporally adjacent path waypoints to prevent them from becoming too close. We use the formula

$$B(r) = \textsc{smoothMax}\left(1, 1 + \frac{2r_1 + r_2}{r}\right)^{-1} \tag{3.8}$$

where r is the distance either between a path waypoint and an object or between two path waypoints, and where r_1 and r_2 are the radii of the two things being kept apart, either the robot or an object. This barrier is approximately 1 until the distance between the two waypoints becomes small, at which point it decreases rapidly, pushing them away from each other by approximately the robot radius. For the penalty between the path waypoints and objects, meant to prevent collision, both the robot radius and object radii are assumed to be 40cm. For the penalty between temporally adjacent path waypoints, meant to ease the optimization problem, r_1 and r_2 are set to 10cm. Finally, because our formulation of the semantics of prepositions is based on angles but not distance, there is a large subspace of the floor that leads to equal probability of satisfying each graphical-model factor, *i.e.*, the cones in Fig. 3.1. This allows a path to satisfy a prepositional phrase like *to the left of the chair* while being very far away from the chair, which, while technically correct, can result in

paths which appear to a human to be infelicitous. To remedy this, we encode a slight preference for shorter distances by adding a small attraction $A(r) = \exp(-\frac{r}{100})$ between each path waypoint and the floorplan waypoints selected as its reference objects, where r is the distance between the path waypoint and the target object of a preposition. The score optimized is the product of the graphical-model factors for each path waypoint along with the barrier and attraction terms. An example of the scoring function corresponding to the example phrase *toward the chair left of the bag*, together with the additional terms, is shown in Fig. 3.11. Its gradient with respect to the path waypoint locations is computed with Automatic Differentiation [38]. The sequence of path waypoints maximizing this product is then found with gradient ascent. The individual points cannot be optimized independently because each graphical model score depends on the velocity, and thus the previous point. The score is optimized repeatedly with subsets of the waypoints increasing in size. The waypoints are added sequentially, each time intitializing the newest point 10cm from the last point in such a way that direction of motion is maintained. Then, the product of scores corresponding to the current set of points is optimized. In the final stage of optimization, all points have been added, and the entirety of the score is optimized. This process helps to prevent the optimization procedure from becoming stuck in local optima.

Fig. 3.10 shows the effect of small differences in the input sentence on the resulting paths for a series of example sentences. A single-word change can greatly alter the path by changing where the robot goes with respect to one object or completely changing which object is referenced. This leads to changes in other parts of the path.

The barrier penalties prevent the path waypoints from being chosen close to objects, but do not prevent the paths between them from doing so. Therefore, a post-processing step performs obstacle avoidance by adding additional path waypoints as needed to ensure that the straight-line path between any two adjacent path waypoints does not pass too close to any floorplan object. This is done by looping over all line segments between temporally adjacent waypoints, checking whether the distance between each object and that point v on the line closest to that object is less than the

sum of the object and robot radii. If so, a new waypoint is created at the point closest to v which is separated from the object by the sum of the two radii. This process is then repeated recursively on all modified line segments until no segment passes too close to any object.

3.5 Experiments

We conducted an experiment as outlined in Fig. 3.1. A corpus of robot paths paired with sentences describing those paths was first collected and used to learn a lexicon. This lexicon was then used to automatically generate sentences from a new set of paths and to produce and follow paths satisfying a new set of sentences. Fig. 3.12 illustrates examples of input to the learning system, the learned lexicon, and examples of the input to and output from the generation and comprehension systems.

All sentences were obtained from independent, anonymous workers on AMT. All paths were recorded using odometry and sensor data from the robot. A second set of judgments were obtained from AMT workers in which they judged the degree to which the sentences and paths matched. Such judgments were obtained for the robot paths automatically driven according to human sentences, for automatically produced sentences describing human-driven paths, and also for the sentences produced by AMT workers to describe human-driven paths. This allows us to compare the performance of the automatic systems against that of AMT workers. Fig. 3.13 depicts the performance of both AMT workers and of our system.

3.5.1 Dataset collection

Three sets of robot paths and sentences were collected. The first, called the *acquisition* corpus, consisted of 750 path-sentence pairs, 3 sentences for each of 250 robot paths. The second, called the *generation* corpus, consisted of 100 robot paths. The third, called the *comprehension* corpus, consisted of 300 path-sentence pairs, 3 sentences for each of 100 robot paths.

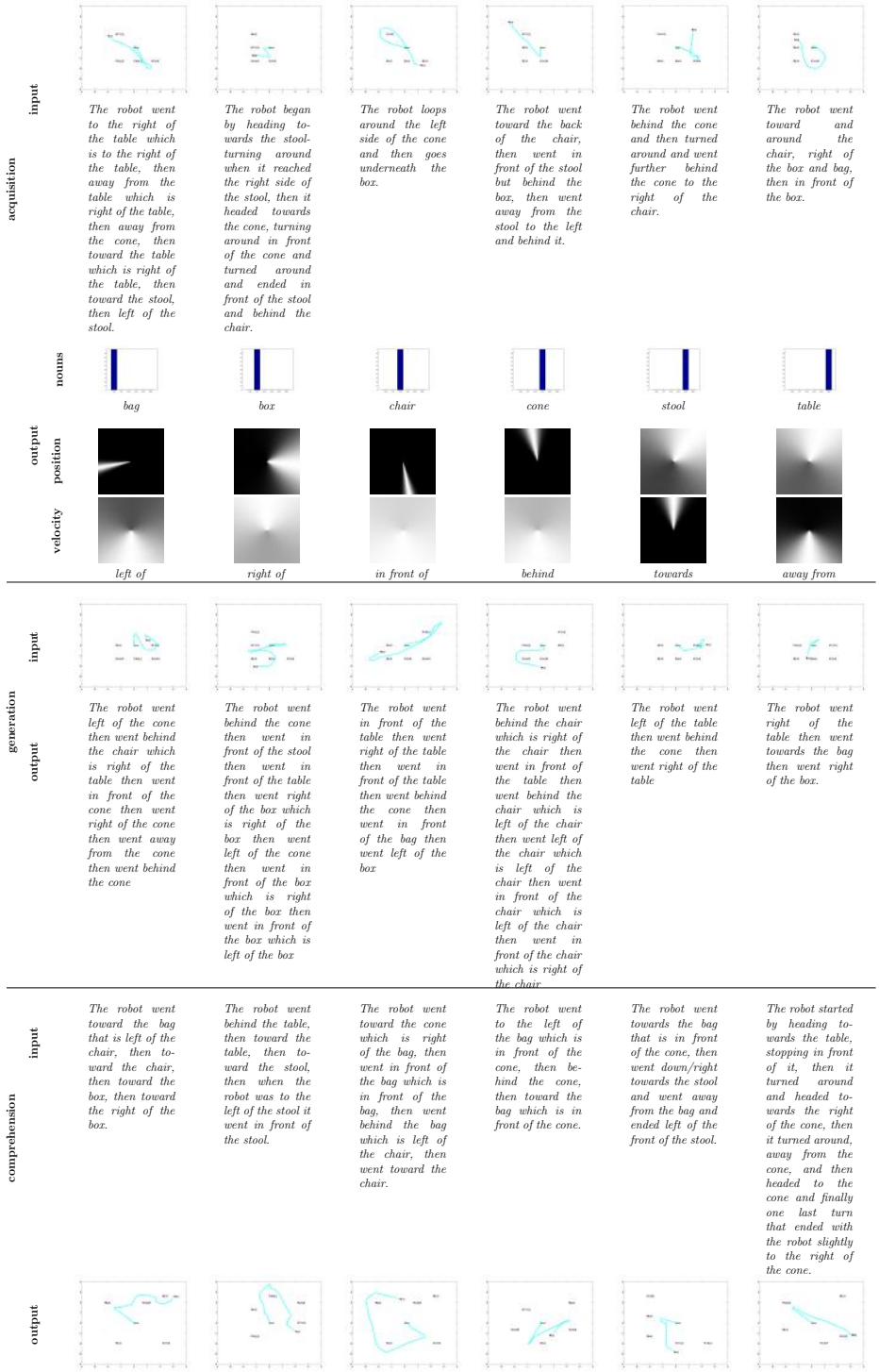


Fig. 3.12. Example experimental runs, 6 for each of acquisition, generation, and comprehension. Our source code and dataset will be available upon publication.

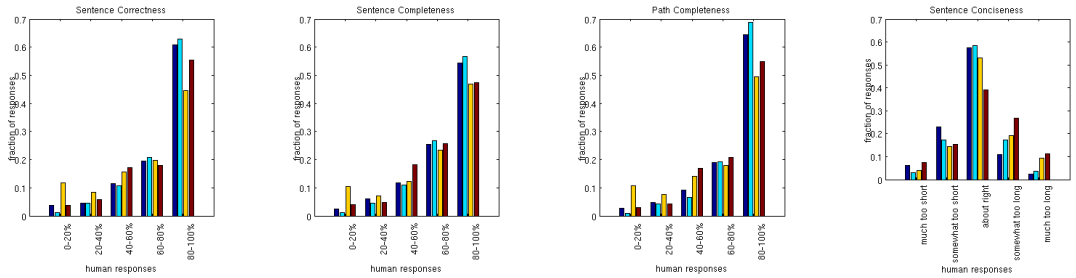


Fig. 3.13. Bar graphs showing the distribution of responses given by AMT workers for each of the four questions: sentence correctness (far left), sentence completeness (middle left), path completeness (middle right), and sentence conciseness (far right). The distributions are shown for sentences elicited from AMT workers and judged against the acquisition (dark blue) and comprehension (light blue) paths used to elicit the sentences, as well as for paths produced by the comprehension system judged against the human sentences used as input (yellow), and for machine-generated sentences judged against the paths used as input (red).

All sentences were obtained from anonymous workers on AMT. These workers were asked to provide a sentence describing a robot path as depicted in an image. They were asked to describe the path in terms of its relation to objects in the floorplan, but were not given any restrictions on the syntax of the sentence. Neither were they told what the sentences were for. Therefore the sentences are not artificially mechanical or specific, as they might have been had they known that they were to be used by a robot. Rather, they are representative of human path descriptions in this domain. Each of the paths used to elicit sentences from workers was obtained by a human driving the robot in a path through a randomly generated floorplan in accordance with a randomly generated sentence.

For the *acquisition* corpus, 10 floorplans were generated, each with 25 random sentences. Each floorplan contained four random objects, with up to one duplicate object. The objects were placed randomly at the corners of floor tiles in a large room. The random sentences used to guide the driving of the *acquisition* paths used to elicit sentences contained a sequence of two or three instructions to move according to random prepositions chosen from *left*, *right*, *front*, *behind*, *towards*, and *away* with respect to randomly chosen objects in the floorplan. For the *generation* and *comprehension* corpora, the floorplan contained five random objects, also limited to one duplicate, which were placed randomly on the corners, centers, or edge centers of the floor tiles. The randomly generated sentences for these corpora were generated similar to those of the *acquisition* corpus, only longer, with a sequence of five or six instructions. For each such sentence, the robot was driven in a path according to the instructions of the random sentence. The human driver was allowed to drive freely, so long as the sentence remained true of the path. Therefore, while these paths do contain, in the proper order, portions which depict the described physical relationships, the driven paths are generally more complex than the original randomly generated sentence.

Images of these paths were given to the AMT workers to elicit sentences. Workers were shown paths in this way in order to elicit sentences which describe a variety of

paths. Many of these elicited sentences contain ambiguity, misspellings, grammatical errors, or describe relations which are untrue or impossible, such as describing a chair as *the chair to the right of the chair*, when there is, in fact, only a single chair. We corrected for obvious misspellings, but did not otherwise make modifications to the sentences. Quantitative evaluation of the quality of the sentences from a second round of AMT judgments can be seen in Fig. 3.13, where the results of the human-generated sentences judged against the paths used to elicit them are shown in dark blue (acquisition sentences) and light blue (comprehension sentences). Only about 60% of human-generated sentences received the highest possible rating from human judges on AMT, with the rest being judged less than 80% correct or complete. Our methods are robust enough to handle such errors gracefully: the learning system learns the correct meaning of words despite the noisy and ambiguous training data, and the comprehension system ignores parts of sentences it cannot understand.

3.5.2 Experimental evaluation

The path-sentence pairs in the *acquisition* corpus were used to learn a lexicon of word models. The paths in the *generation* corpus were used to test the robot's ability to use its learned lexicon to generate a sentence which describes a given path. The sentences in the *comprehension* corpus were used to test the robot's ability to use its learned lexicon to automatically generate and follow a path described by a sentence.

We determined the nouns and prepositions to learn as follows. We identified prepositions of interest as words which appear in the Wikipedia list of prepositions, and which appear more than 100 times in our combined corpus of 1050 sentences, (excluding *in*, *on*, and *to*). This resulted in the following list of prepositions: *left*, *right*, *front*, *behind*, *towards*, and *away*. We identified nouns of interest as words which have not been identified as prepositions, and which have been tagged as nouns by the Stanford parser more than 100 times in our corpus (excluding the word *robot*). This resulted in the following list of nouns: *bag*, *box*, *chair*, *cone*, *stool*, and *table*.

Among the 750 sentences in the acquisition corpus are a number of extremely long sentences which approach the limit of the Stanford parser, and which slow down the learning process. Therefore, the sentences were sorted according to the number of references to objects, and the 600 shortest sentences were used for training. This resulted in the exclusion of sentences with more than 16 object references. The acquisition system used the resulting 600 path-sentence pairs to learn the lexicon of word meanings.

After learning the meanings of the words, the recorded paths from the *generation* corpus were used to automatically produce sentential descriptions and the sentences from the *comprehension* corpus were used to automatically drive the robot, recording its path through odometry. A second round of AMT judgments were obtained, judging the degree to which the sentence and path in each pair match. Such judgments were obtained for the robot paths automatically driven according to human sentences, for the automatically produced sentences describing human-driven paths, and also for the sentences produced by AMT workers to describe human-driven sentences. This allowed us to compare the performance of the automatic systems with that of AMT workers through four multiple-choice questions:

Sentence Correctness: Approximately how much of the sentence is true of the path?

Sentence Completeness: Approximately how much of the path is described by the sentence?

Path Completeness: Approximately how much of the sentence is depicted by the path?

Sentence Conciseness: Rate the length of the sentence.

For the first three questions, the possible answers were *0–20%*, *20–40%*, *40–60%*, *60–80%*, and *80–100%*. The first two questions allow us to evaluate how true (sentence correctness) each sentence is and how fully it describes its corresponding path (sentence completeness). The third question allows us to evaluate how fully a driven path executes the sequence of actions described in its corresponding sentence. For the last question, the possible answers were *much too short*, *somewhat too short*, *about*

right, *somewhat too long*, and *much too long*. This allows us to evaluate the verbosity of the sentence-generation system, compared to human-generated sentences.

We obtained three independent judgments for each path-sentence pair in order to evaluate the reliability of the human judgments. For 26.5% of the path-sentence pairs, all three judgments agreed, for 54.1% of the pairs, two of the judgments agreed, and for 19.4% of the pairs, all three judgments differed.

Fig. 3.13 shows details of the distributions of judgments given to each of the sets of path-sentence pairs for each question. The fraction of human judgments in each of the possible responses for each of the questions is shown for the output of human annotators (dark and light blue), the comprehension system (yellow), and the generation system (red), each judged against the paths or sentences given as input. Sentence length was judged *about right* 58.1% of the time for humans and 39.1% of the time for our sentence generation system. The *about right* and *somewhat too short/long* judgments combine to 92.4% for humans and 81.3% for our generation system. Averaging the judgments for the first three questions yields 82.4% and 85.3% for the human sentences on the *acquisition* and *comprehension* corpora, respectively, when judged against the paths used to elicit those sentences. Averaging the judgments for the automatically driven paths judged against the human sentences used as input yields 71.1%. Averaging the judgments for the automatically produced sentences judged against the paths used as input yields 78.6%. Overall, machine performance is 74.9%, while the performance of human annotators is 83.8%.

Our system learns the proper meanings of the words despite the fact that the human sentences used as input are far from perfect, and using these learned meanings, can produce paths and sentences whose quality averages 89.2% of the way towards human performance. The mistakes made by the automatic driving system were due to the input sentences containing words which were too rare to learn (*e.g.*, *zig-zagged*, *circled*, *between*, *past*, *near*, *above*, and *underneath*), and occasionally sophisticated grammar usage not handled by our framework for mapping sentences to sequences of graphical models. While the generation system generally produces sentences which

are technically correct, such sentences are not always the most intuitive or helpful from a human perspective, resulting in lower scores than for human-generated sentences.

3.6 Conclusion

We demonstrate a novel approach for grounding the semantics of natural language in the domain of robot navigation. Sentences describe paths taken by the robot relative to other objects in the environment. The meanings of nouns and prepositions are trained from a corpus of paths driven by a human teleoperator annotated with sentential descriptions. These can then support both automatic generation of sentential descriptions of new paths as well as automatic driving of paths to satisfy navigational goals specified in provided sentences. This is a step towards the ultimate goal of grounded natural language that allows machines to interact with humans when the language refers to actual things and activities in the real world.

Acknowledgments

This research was sponsored, in part, by the Army Research Laboratory, accomplished under Cooperative Agreement Number W911NF-10-2-0060, and by the National Science Foundation under Grant No. 1522954-IIS. The views, opinions, findings, conclusions, and recommendations contained in this document are those of the authors and should not be interpreted as representing the official policies, either express or implied, of the Army Research Laboratory, the National Science Foundation, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

4. THE LARGE CONTINUOUS ACTION CORPUS

We make available to the community a new dataset to support action-recognition research. This dataset is different from prior datasets in several key ways. It is significantly larger. It contains streaming video with long segments containing multiple action occurrences that often overlap in space and/or time. All actions were filmed in the same collection of backgrounds so that background gives little clue as to action class. We had five humans replicate the annotation of temporal extent of action occurrences labeled with their class and measured a surprisingly low level of intercoder agreement. A baseline experiment shows that recent state-of-the-art methods perform poorly on this dataset. This suggests that this will be a challenging dataset to foster advances in action-recognition research. This manuscript serves to describe the novel content and characteristics of the LCA dataset, present the design decisions made when filming the dataset, and document the novel methods employed to annotate the dataset.

4.1 Introduction

There has been considerable research interest in action recognition in video over the past two decades [2,5–10,12,14–17,20–22,26,30,44,45,49,79–115]. To support such research, numerous video datasets have been gathered. Liu *et al.* [116] summarize the available datasets as of 2011. These include KTH (6 classes, [93]), Weizmann (10 classes, [84]), CMU Soccer (7 classes, [117]), CMU Crowded (5 classes, [110]), UCF Sports (9 classes, [2]), UR ADL (10 classes, [49]), UM Gesture (14 classes, [42]), UCF Youtube (11 classes, [16]), Hollywood-1 (8 classes, [118]), Hollywood-2 (12 classes, [119]), MultiKTH (6 classes, [120]), MSR (3 classes, [121]), and TRECVID (10 classes, [122]). These datasets contain short clips, each depicting one of a small

number of classes (3–14). Several more recent datasets also contain short clips, each depicting a single action but with a larger number of action classes: UCF50 (50 classes, [103]), HMDB51 (51 classes, [6]), and UCF101 (101 classes, [123]). The VIRAT dataset [124] has 12 classes and longer streaming video.

Here, we introduce a new dataset called the *Large Continuous Action Corpus* (LCA). This dataset contains depictions of 24 action classes. A novel characteristic of this dataset is that rather than consisting of short clips each of which depicts a single action class, this dataset contains much longer streaming video segments that each contain numerous instances of a variety of action classes that often overlap in time and may occur in different portions of the field of view. The annotation that accompanies this dataset delineates not only which actions occur but also their temporal extent.

Many of the prior datasets were culled from videos downloaded from the internet. In contrast, the LCA dataset contains videos that were filmed specifically to construct the dataset. While the video was filmed with people hired to act out the specified actions according to a general script, the fact that the video contains long streaming segments tends to mitigate any artificial aspects of the video and render the action depictions to be quite natural. Moreover, the fact that all of the video was filmed in a relatively small number of distinct backgrounds makes the dataset challenging; the background gives little clue as to the action class.

A further distinguishing characteristic of the LCA dataset is the degree of ambiguity. Most prior action-recognition datasets, in fact most prior datasets for all computer-vision tasks, make a tacit assumption that the labeling is unambiguous and thus there is a 'ground truth.' We had a team of five human annotators each annotate the entire LCA dataset. This allowed us to measure the degree of intercoder agreement. Surprising, there is a significant level of disagreement between humans as to the temporal extent of most action instances. We believe that such inherent ambiguity is a more accurate reflection of the underlying action-recognition task and

hope that the multiplicity of divergent annotations will help spur novel research with this more realistic dataset.

Another distinguishing characteristic of the LCA dataset is that some action occurrences were filmed simultaneously with multiple cameras with partially overlapping fields of view. While the cameras were neither spatially calibrated nor temporally synchronized, the fact that we have multiple annotations of the temporal extent of event occurrences may support future efforts to perform temporal synchronization after the fact. Furthermore, while most of the video was filmed from ground-level cameras with horizontal view, some of the video was filmed with aerial cameras with bird's eye view. Some of this video was filmed simultaneously with ground cameras. This may support future efforts to conduct scene reconstruction.

Some datasets are provided with specific tasks and evaluation metrics. We refrain from doing so for this dataset. *Inter alia*, we do not provide official sanctioned splits into validation sets. Instead, we leave it up to the community to make use of this dataset in a creative fashion for as many different tasks as it will be suited.

In particular, the evaluations conducted by Mind's Eye included a specific set of tasks, namely Recognition (REC), Description (DES), Gap Filling (GAP), and Anomaly Detection (ANM) with specific evaluation metrics. Such tasks and metrics are expressly *not* part of LCA. The evaluations conducted under Mind's Eye make use of material that is not included in LCA and metrics that are not public. Likewise, LCA contains material that was not available for use during the Mind's Eye evaluations. Thus said evaluations could not be replicated outside of the context of Mind's Eye. Any potential future evaluations conducted with LCA would thus be incomparable to the results obtained under Mind's Eye.

The entire LCA dataset, including the video and the annotations, has been cleared for release by DARPA. The remaining material gathered by DARPA for the Mind's Eye Year 2 evaluation that is not included in LCA may not have been cleared for release. As part of the release process, some video was edited to remove certain portions. Furthermore, the annotation process was performed with the particular

versions of the videos included in LCA as provided by DARPA. These may have been transcoded from the original as filmed by the camera. Thus, the time alignment of the annotations can only be guaranteed with the versions of the videos included in LCA. The time alignment may not be correct for any other versions of these videos that may be residual from the DARPA Mind's Eye program.

4.2 Collection

The video for this dataset was filmed by DARPA in conjunction with Mitre and several performers from the Mind's Eye program.[1] The bulk of the video was filmed as part of the Mind's Eye Year 2 evaluation. Within the Mind's Eye program, that video was referred to as C-D2a, C-D2b, C-D2c, and the Y2 Evaluation dataset. This video is disjoint from that gathered by Janus Research Group as part of the Mind's Eye Year 1 evaluation. Within the Mind's Eye program, that video was refereed to as C-D1a, C-D1b, C-D1, and C-E1. As the LCA dataset contains only a subset of that material, we refrain from using all such terminology in reference to LCA.

The LCA dataset was filmed at three different locations over four periods:

1. The Great Plains Joint Training Center (GPJTC), an army training facility in Kansas, on 22 August 2011. Filming took place in two contexts at GPJTC, a simulated country road and a simulated safe house.
2. Strategic Operations, Inc. (STOPS), a training facility in San Diego, on 14–15 December 2011 and on 6–9 March 2012. Filming during the first period took place in three contexts: two different simulated country roads and one simulated safe house. Filming during the second period took place in five contexts: two different simulated country roads, two different simulated safe houses, and one other.
3. Fort Indiantown Gap (FITG), an army training facility in Pennsylvania, on 19–20 June 2012.

[1] `http://www.darpa.mil/Our_Work/I2O/Programs/Minds_Eye.aspx`
`http://www.visint.org/`

A portion of the video was annotated by Mitre for the Mind's Eye Year 2 evaluation. That annotation is not included in LCA. After the completion of the Mind's Eye Year 2 evaluation, we undertook a systematic annotation effort for a portion of the above video. That annotation forms the basis of LCA. LCA contains all and only the portion of the above video that was annotated as part of this process. This video comprises 190 files as delineated in Table 4.1. Eight files are MOV format, 46 are MP4 format, and 136 are AVI format. The MOV files all use the MP4V codec and are 640×384 at 60 fps. The MP4 files all use the H264 codec and are 640×360 at 30 fps. The AVI files all use the XVID codec and are either 640×384 at 60 fps or 1440×1080 at 30 fps. This constitutes 2302144 frames and a total of 12 hours, 51 minutes, and 16 seconds of video. For comparison, UCF50 has 1330936 frames and 13.81 hours, HMDB51 has 632635 frames and 5.85 hours, UCF101 has 27 hours, Hollywood-2 has 20.1 hours, and VIRAT has 8.5 hours. Several frame sequences from this dataset illustrating several of the backgrounds are shown in Fig. 4.1.

The Mind's Eye program specified a set of 48 verbs of interest. Of these, the LCA corpus uses only 24 verbs as annotation labels, as delineated in Table 4.2. Of these, 17 verbs were used as part of the stage directions given to the actors to guide the actions that they performed. The remainder were not used as part of the stage directions but occurred incidentally. Nothing, however, precluded the actors from performing actions that could be described by other verbs. Thus the video depicts many other actions than those annotated, including but not limited to riding bicycles, pushing carts, singing, pointing guns, arguing, and kicking balls. The only restriction, in principle, to these 24 verbs is that these were the only actions that were annotated. Identifying the presence of specific verbs in the presence of many such confounding actions should present additional challenges.

GPTC_20110822_SH_02_CP1_NOACTIVITY
GPTC_20110822_SH_02_CP2_NOACTIVITY
GPTC_20110822_SH_07_CP1_EX&RET
GPTC_20110822_SH_07_CP2_EX&RET
GPTC_20110822_SH_13_CP1_WARYHO
GPTC_20110822_SH_13_CP2_WARYHO
GPTC_20110822_SH_11_CP1_PARAHO
GPTC_20110822_SH_11_CP2_PARAHO
GPTC_20110822_SH_12_CP1_HOSTILEHO
GPTC_20110822_SH_12_CP2_HOSTILEHO
GPTC_20110822_SH_06_CP1_SUPCACHRET
GPTC_20110822_SH_06_CP2_SUPCACHRET
GPTC_20110822_SH_09_CP1_GIVE&CONT-BLDG
GPTC_20110822_SH_09_CP2_GIVE&CONT-BLDG
GPTC_20110822_SH_05_CP1_SUPCACHDUMP
GPTC_20110822_SH_05_CP2_SUPCACHDUMP
GPTC_20110822_SH_08_CP1_HO&EX-mixBKG
GPTC_20110822_SH_08_CP2_HO&EX-mixBKG
GPTC_20110822_SH_03_CP1_HO&RET
GPTC_20110822_SH_03_CP2_HO&RET
GPTC_20110822_SH_04_CP1_HO&RET2
GPTC_20110822_SH_04_CP2_HO&RET2
GPTC_20110822_CR_13_CP1_NOACTIVITY
GPTC_20110822_CR_13_CP2_NOACTIVITY
GPTC_20110822_CR_07_CP1_SALESMAN
GPTC_20110822_CR_07_CP2_SALESMAN
GPTC_20110822_CR_12_CP1_BAGDOWNHO&RET
GPTC_20110822_CR_12_CP2_BAGDOWNHO&RET
GPTC_20110822_CR_02_CP1_RoutineActivity
GPTC_20110822_CR_02_CP2_RoutineActivity
GPTC_20110822_CR_05_CP1_CONTHUR-NOHO
GPTC_20110822_CR_05_CP2_CONTHUR-NOHO
GPTC_20110822_CR_04_CP1_EX
GPTC_20110822_CR_04_CP2_EX
GPTC_20110822_CR_08_CP1_SALESMAN_INSIST
GPTC_20110822_CR_08_CP2_SALESMAN_INSIST
GPTC_20110822_CR_11_CP1_DOUBLEAVOIDBADGUY
GPTC_20110822_CR_11_CP2_DOUBLEAVOIDBADGUY
GPTC_20110822_CR_09_CP1_UPTONOGOOD
GPTC_20110822_CR_09_CP2_UPTONOGOOD
GPTC_20110822_CR_06_CP1_DISAGREE_NOGIVE_RETURN
GPTC_20110822_CR_06_CP2_DISAGREE_NOGIVE_RETURN
GPTC_20110822_CR_10_CP1_AVOIDBADGUY
GPTC_20110822_CR_10_CP2_AVOIDBADGUY
GPTC_20110822_CR_03_CP1_HO
GPTC_20110822_CR_03_CP2_HO

STOPS_20120307_SH3_06_C1-edited-01
STOPS_20120307_SH3_06_C3-edited-01
STOPS_20120307_SH3_03_C1-edited-01
STOPS_20120307_SH3_03_C3-edited-01
STOPS_20120307_SH3_02_C1-edited-01
STOPS_20120307_SH3_02_C3-edited-01
STOPS_20120307_SH3_01_C1-edited-01
STOPS_20120307_SH3_01_C3-edited-01
STOPS_20120308_CR1_07a_C1
STOPS_20120308_CR1_07a_C3
STOPS_20120309_VT1_23_C1
STOPS_20120309_VT1_23_C3
STOPS_20120308_CR1_NA_C1
STOPS_20120308_CR1_NA_C3
STOPS_20120306_SH1_06_C1
STOPS_20120306_SH1_06_C3
STOPS_20120306_SH1_NA_C1
STOPS_20120306_SH1_NA_C3
STOPS_20120308_CR1_08a_C1
STOPS_20120308_CR1_08a_C3
STOPS_20120308_CR1_12a_C1
STOPS_20120308_CR1_12a_C3
STOPS_20120308_CR1_01a_C1
STOPS_20120308_CR1_01a_C3
STOPS_20120309_VT1_01_C1
STOPS_20120309_VT1_01_C3
STOPS_20120309_VT1_26_C1
STOPS_20120309_VT1_26_C3
STOPS_20120308_CR1_02a_C1
STOPS_20120308_CR1_02a_C3
STOPS_20120308_CR1_05a_C1
STOPS_20120308_CR1_05a_C3
STOPS_20120309_VT1_04_C1
STOPS_20120309_VT1_04_C3
STOPS_20120306_SH1_07_C1
STOPS_20120306_SH1_07_C3
STOPS_20120309_VT1_21_C1
STOPS_20120309_VT1_21_C3
STOPS_20120307_SH3_07_C1
STOPS_20120307_SH3_07_C3
STOPS_20120308_CR1_09a_C1
STOPS_20120308_CR1_09a_C3
STOPS_20120307_SH3_08_C1
STOPS_20120307_SH3_08_C3
STOPS_20120308_CR1_11a_C1
STOPS_20120308_CR1_11a_C3
STOPS_20120306_SH1_05_C1
STOPS_20120306_SH1_05_C3
STOPS_20120309_VT1_12_C1
STOPS_20120309_VT1_12_C3
STOPS_20120309_VT1_10_C1
STOPS_20120309_VT1_10_C3

STOPS_20120308_CR1_06a_C1 100
STOPS_20120308_CR1_06a_C3 105
STOPS_20120306_SH1_04_C1 114
STOPS_20120306_SH1_04_C3 118
STOPS_20120306_SH1_01_C1 11
STOPS_20120306_SH1_01_C3 121
STOPS_20120308_CR1_13a_C1 128
STOPS_20120308_CR1_13a_C3 133
STOPS_20120309_VT1_07_C1 154
STOPS_20120309_VT1_07_C3 158
STOPS_20120309_VT1_25_C1 162
STOPS_20120309_VT1_25_C3 175
STOPS_20120309_VT1_05_C1 188
STOPS_20120309_VT1_05_C3 195
STOPS_20120308_CR1_01c_C1 196
STOPS_20120308_CR1_01c_C3 200
STOPS_20120309_VT1_NA_C1 211
STOPS_20120309_VT1_NA_C3 226
STOPS_20120308_CR1_03a_C1 22
STOPS_20120308_CR1_03a_C3 230
STOPS_20120306_SH1_02_C1 238
STOPS_20120306_SH1_02_C3 243
STOPS_20120307_SH3_11_C1 252
STOPS_20120307_SH3_11_C3 265
STOPS_20120308_CR1_04a_C1 269
STOPS_20120308_CR1_04a_C3 277
STOPS_20120307_SH3_10_C1 30
STOPS_20120307_SH3_10_C3 39
STOPS_20120309_VT1_22_C1 41
STOPS_20120309_VT1_22_C3 46
STOPS_20120308_CR1_01b_C1 52
STOPS_20120308_CR1_01b_C3 57
STOPS_20120309_VT1_28_C1 59
STOPS_20120309_VT1_28_C3 62
STOPS_20120309_VT1_20_C1 68
STOPS_20120309_VT1_20_C3 69
STOPS_20120309_VT1_03_C1 81
STOPS_20120309_VT1_03_C3 86
STOPS_20120309_VT1_27_C1 88
STOPS_20120309_VT1_27_C3 97
STOPS_20120309_VT1_13_C1
STOPS_20120309_VT1_13_C3
STOPS_20120308_CR1_06b_C1
STOPS_20120308_CR1_06b_C3
STOPS_20120309_VT1_02_C1
STOPS_20120309_VT1_02_C3
STOPS_20120308_CR1_10a_C1
STOPS_20120308_CR1_10a_C3
STOPS_20120309_VT1_24_C1
STOPS_20120309_VT1_24_C3
STOPS_20120308_CR1_03b_C1
STOPS_20120308_CR1_03b_C3

Table 4.1.

Video filenames from the LCA dataset. The original names of the filed provided by DARPA were used. Filenames containing GPTC were filmed at GPJTC. Filenames containing STOPS were filmed at STOPS. Filenames consisting solely of a number were filmed at FITG. Numbers of the form YYYYMMDD indicate filming date. CR indicates country road. SH indicates safe house. Indices on CR, SH, and VT indicate variant backgrounds of the given class. CP1, CP2, C1, and C3 indicate camera. Text indicates the staging directions to guide filming. The remaining numbers serve to uniquely identify the video.

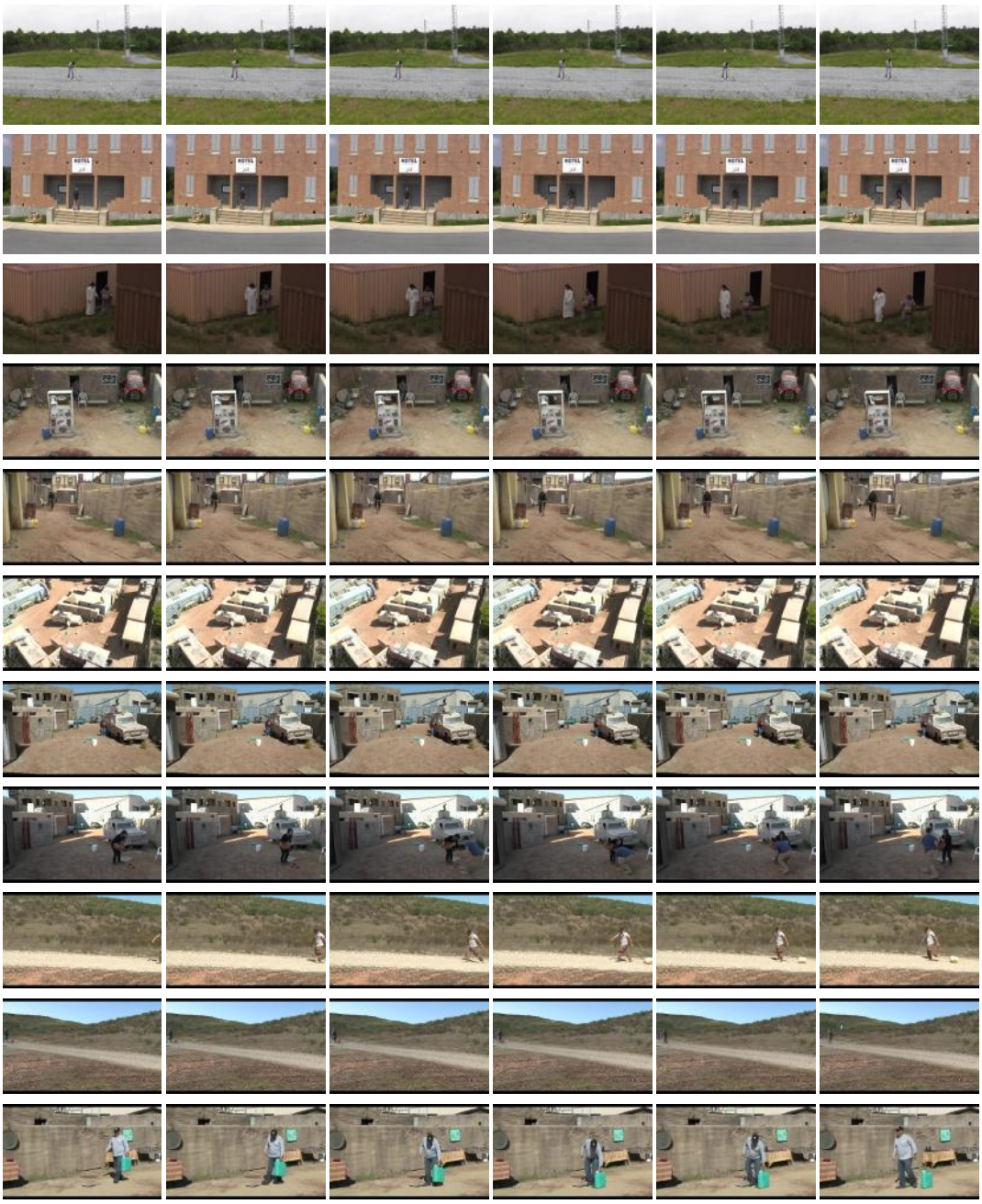


Fig. 4.1. Several frame sequences from the LCA dataset illustrating several of the backgrounds in which they were filmed.

*approach**	*drop**	*give**	*replace**
arrive	*enter**	*hold*	*run*
*bury**	*exchange**	*leave*	*stop*
*carry**	*exit**	*pass**	*take**
*chase**	*flee**	*pick up**	*turn*
*dig**	*follow**	*put down**	*walk*

Table 4.2.

Verbs used as labels in the LCA dataset. The starred verbs were used as part of the stage directions to the actors. The remaining verbs were not used as part of the stage directions but may have occurred incidentally.

4.3 Annotation

We annotated all occurrences of the 24 verbs from Table 4.2 in the videos in Table 4.1. Each such occurrence consisted of a temporal interval labeled with a verb. The judgment of whether an action described by a particular verb occurred is subjective; different annotators will arrive at different judgments as to occurrence as well as the temporal extent thereof. To help guide annotators, we gave them the specification of the intended meaning of each of the 24 verbs as provided by DARPA. Annotators performed the annotation at workstations with dual monitors. One monitor displayed the annotation tool while the other monitor displayed the documentation of intended verb meaning. The documentation of intended verb meaning is included in the LCA distribution.

We also asked annotators to annotate intervals where certain object classes were present in the field of view. These include *bandannas*, *bicycles*, *people*, *vehicles*, and *weapons*. (The *bandannas* were worn by *people* around their head or arms.) For these, a count of the number of instances of each class that were visible in the field of view was maintained. It was incremented each time a new instance became visible and decremented each time an instance became invisible. We instructed annotators that there was no need to be precise when an instance was partially visible. We further instructed annotators that *vehicles* denoted motor vehicles, not push carts, and *weapons* denoted guns, not other things like clubs or rocks that could be used as weapons.

We provided annotators with a tool that allowed them to view the videos at ordinary frame rate, stop and start the videos at will, navigate to arbitrary points in the videos, view individual frames of the videos, add, delete, and move starting and ending points of intervals, and label intervals with verbs. The tool also contained buttons to increment and decrement the counts for each of the object classes and appraised the annotator with the running counts for the objects classes in each frame as the video was played or navigated.

Because of the large quantity of video to be annotated, and the fact that nothing happens during large portions of the video, we preprocessed the video to reduce the amount requiring manual annotation. We first downsampled the video to 5 fps just for the purpose of annotation; the annotation was converted back at the end to the original frame rate. Then segments of this downsampled video where no motion occurred were removed. To do this, we computed dense optical flow on each pixel of each frame of the downsampled video. We then computed the average of the magnitude of the flow vectors in each frame and determined which frames were above a threshold. Stretches of contiguous frames that were above threshold that were separated by short stretches of contiguous frame that were below threshold were merged into single temporal segments. Then such single temporal segments that were shorter than a specified temporal length were discarded.[2] Annotators were only given the remaining temporal segments to annotate. We performed a postprocessing step whereby the authors manually viewed all discarded frames to make sure that no actions started, ended, or spanned the omitted temporal segments. As part of this post processing step the authors manually checked that none of the specified object classes entered or left the field of view during the omitted temporal segments.

We had five annotators each independently annotate the entire LCA corpus. Annotators were given initial instructions. During the annotation, annotators were encouraged to discuss their annotation judgments with the authors. The authors would then arbitrate the judgment, often specifying principles to guide the annotation. These principles were then circulated among the other annotators. The annotator instructions and principles developed through arbitration are included in the LCA distribution.

We performed a consistency check during the annotation process. Whenever an annotator completed annotation of a temporal segment, if that annotator did not

[2]The threshold for average optical flow magnitude was 150. The threshold for ignoring short below-threshold spans when merging contiguous above-threshold frames into temporal segments was 50 frames. The threshold for ignoring short temporal segments was 15 frames.

annotate any intervals during that segment but other annotators did, we asked that annotator to review their annotation.

The LCA dataset contains five verb annotation files for each of the video files in Table 4.1. These have one of the annotator codes `bmedikon`, `cbushman`, `kim861`, `nielder`, and `nzabikh` in the filename, the keyword `verb` in the filename, and the extension `txt`. Each line in each of these files contains a single temporal interval as two zero-origin nonnegative integers specifying the starting and ending frames of the interval inclusive and a text string specifying a verb. The LCA dataset also contains five object-class annotation files for each of the video files in Table 4.1. These have one of the above annotator codes `in` the filename, the keyword `object-class` in the filename, and the extension `txt`. Each line in each of these files contains a single zero-origin nonnegative integer specifying a frame, a text string specifying an object class, and either the text string `enter` or `exit`.

4.4 Analysis

We analyzed the degree of agreement between the different annotators. To do this, we compared pairs of annotators, taking the judgments of one as 'ground truth' and computing the F1 score of the other. An interval in the annotation being scored was taken as a true positive if it overlapped some interval with the same label in the 'ground truth'. An interval in the annotation being scored was taken as a false positive if it didn't overlap any interval with the same label in the 'ground truth'. An interval in the 'ground truth' was taken as a false negative if it didn't overlap any interval with the same label in the annotation being scored. From these counts, an F1 score could be computed.

We employed the following overlap criterion. For a pair of intervals, we computed a one-dimensional variant of the 'intersection over union' criterion employed within the Pascal VOC Challenge to determine overlap of two axis-aligned rectangles [125], namely the length of the intersection divided by the length of the union. We consid-

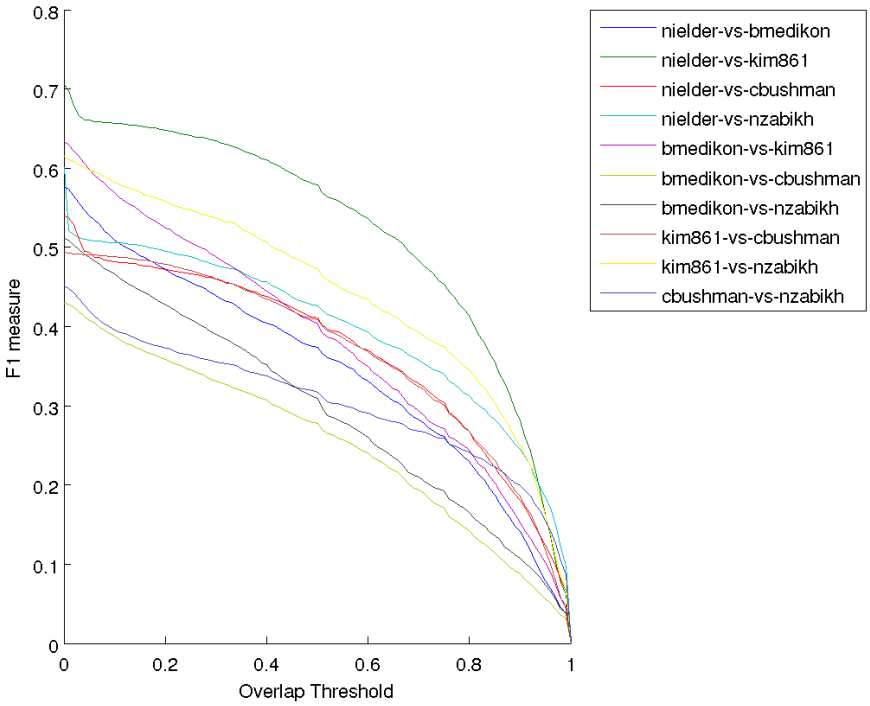


Fig. 4.2. Intercoder agreement on the annotations of the LCA dataset. F1 score for each pair of annotators as the overlap criterion is varied. Overlap of two intervals is measured as the length of their intersection divided by the length of their union.

ered two intervals to overlap when the above exceeded some specified threshold. We then computed the F1 score as this threshold was varied and plotted the results for all pairs of annotators (Fig. 4.2).

Note that there is a surprisingly low level of agreement between annotators. Annotators rarely if ever agree on the precise temporal extent of an action as indicated by the fact that all agreement curves go to zero as the overlap threshold goes to one. At an overlap threshold of 0.5, the F1 score varies between about 0.3 and about 0.6. At an overlap threshold of 0.1, the threshold employed by VIRAT to score machines against humans, the F1 score varies between about 0.38 and about 0.67. This would put an upper bound on machine performance with this dataset using the VIRAT threshold. Even if the overlap threshold is reduced to zero, the F1 score varies be-

tween about 0.43 and about 0.7. This indicates that this dataset should be challenging for computer action recognition.

This difficulty is corroborated by a recent paper [126]. That paper employs a different subset of video from the DARPA Mind's Eye Year 2 evaluation that is extremely similar to that in the LCA dataset. That dataset was annotated with the procedures used to annotate the LCA dataset. The six verbs with highest intercoder agreement were selected: *carry*, *dig*, *hold*, *pick up*, *put down*, and *walk*. For each of these between 23 and 30 clips of 2.5s duration with the highest level of intercoder agreement were selected, yielding 169 distinct clips. Seven different state-of-the-art computer-vision action recognition methods (C2 [127], Action Bank [44], Stacked ISA [47], VHTK [49], Cao's implementation [46] of Ryoo's method [48], Cao's method [46], and our own implementation of the classifier described in [82] on top of the Dense Trajectories [45, 82, 83] feature extractor) were employed on this dataset, performing one-of-out-six classification in an eight-fold cross-validation. Note that for this task, each 2.5s clip was labeled with precisely one of the six verbs as ground truth. All seven methods performed extremely poorly on this dataset (C2 47.4%, Action Bank 44.2%, Stack ISA 46.8%, VHTK 32.5%, Cao's implementation of Ryoo's method 31.2%, Cao's method 33.3%, Dense Trajectories 52.3%), a task with only six classes and chance performance of 16.6%.

4.5 Experiments

We performed a set of baseline experiments to present and compare the performance of all known action recognition systems for which the code for the end to end system is available, as well as implementations of several for which the code is unavailable. We used the same set of seven state-of-the-art event-recognition systems compared in a recent paper [126] (C2 [127], Action Bank [44], Stacked ISA [47], VHTK [49], Cao's implementation [46] of Ryoo's method [48], Cao's method [46], and our own implementation of the classifier described in [82] on top of the Dense

Trajectories [82,83] feature extractor), as well as on top of the more recent Improved Trajectories method [45].

As these methods are designed for classification of video clips, rather than for streaming video, this experiment was performed on a subset of the LCA designed to be similar in character to other event-recognition datasets, and composed of short video clips. It was created as follows. First, we took the human-annotated event instances produced by one of the annotators, cbushman. This annotator was chosen to maximize the number of available event intervals: while cbushman did not have the highest agreement with the other annotators, it was observed to be because this annotator found a large number of event intervals which were missed by the others. Next, a random set of 100 intervals was selected for each event class for which 100 or more were available, and all available intervals were chosen for those classes for which fewer than 100 intervals were available. A 2 second clip was extracted from the original videos centered in time on the middle of each selected event interval, and downsampled in time to 20 fps and in space to a width of 320 pixels, maintaining aspect ratio. This process resulted in a total of 1858 clips used for the baseline experiments.

The class label of each such clip was considered to be the event corresponding to the human-annotated interval from which the clip was derived. The clips for each class were randomly split into a training set with 70 percent of the clips and a test set with 30 percent of the clips, under the constraint that sets of clips extracted from the same video should fall completely into either the training or test set. This was done to avoid having clips (e.g. two clips from the same person digging in the same location) from appearing in both the training and test sets. This resulted in a training set of 1318 training videos and 540 test videos. Each method was trained on the training set and used to produce labels on the test set. All methods were run with default or recommended parameters. These labels were compared to the intended class labels to measure the accuracy of each method. The results of such experiments are summarized in Table 4.3

Table 4.3.
Comparison of Accuracy for state of the art systems on the baseline experiment.

Method	#correct	accuracy (%)
Action Bank [44]	90	16.667
Improved Trajectories [45]	84	15.556
Dense Trajectories [82, 83]	76	14.074
C2 [127]	50	9.259
Cao [46]	41	7.592
Cao's [46] implementation of Ryoo [48]	36	6.667
Stacked ISA [47]	36	6.667
VHTK [49]	34	6.296

There are several things of note in these results. First, all the accuracies are quite low, indicating the difficulty of the task. The highest performing method, Action Bank [44], is correct only 16.667% of the time. The four lowest performing methods have accuracies approaching chance performance (5.555%).

Additionally, the newer methods do not necessarily outperform the older methods. C2 [127] significantly outperforms four more recently published methods, while Action Bank is the best, outperforming even Improved Trajectories [45], which has the highest performance on several well known datasets such as datasets including HMDB (57.2% vs Action Bank's 26.9%) and UCF50 (91.2%). We suspect that this difference in relative performance compared to other datasets is the result of the lack of correllation between background and event class which is often present in other datasets. That the performance is so low and that the highest scoring methods on other datasets are not necessarily the same here shows that this dataset presents new and difficult challenges not present in other datasets.

4.6 Conclusion

We make available to the community a new dataset to support action-recognition research.[3] This dataset has more hours of video than HMDB51, roughly the same amount of video as UCF50, about half as much video as UCF101 and Hollywood-2, but unlike these has streaming video and has about twice as much video and twice as many classes as VIRAT, the largest dataset of streaming video. A distinguishing characteristic of this dataset is that the video is streaming; long video segments contain many actions that start and stop at arbitrary times, often overlapping in space and/or time. A further distinguishing characteristic is that while all actions were filmed in a variety of backgrounds, every action occurs in every background so that background gives little information as to action class. We employed novel techniques to annotate the temporal extent of action occurrences. A multiplicity of

[3] `http://upplysingaoflun.purdue.edu/~qobi/lca.tgz`

human annotations allows measuring intercoder agreement. The above characteristics together with the surprisingly low level of intercoder agreement suggest that this will be a challenging dataset. This is confirmed by the low performance of recent methods on a baseline experiment which also shows that those methods which perform best on other datasets do not necessarily outperform other methods on this dataset. The new difficulties posed by this dataset should spur significant advances in action-recognition research.

5. COMPARISON OF ACTION RECOGNITION WITH FMRI MIND READING

This chapter describes a comparison of the performance of video action recognition methods and fMRI mind-reading techniques in an apples-to-apples experiment. The on the one hand, action recognition methods were used to classify a set of videos. On the other hand, videos were classified using fMRI brain scan data of human subjects while they were watching these same videos.

5.1 Dataset

We employed a small portion of the video dataset gathered as part of the Year 2 evaluation for the DARPA Mind's Eye program.[1] (Note that we did not design the corpus or film the video ourselves; it was designed and filmed by DARPA and provided to all teams funded by the Mind's Eye program.) In particular, we used data from two components of that dataset: the portion known as `C-D2b`, which was intended to be used as training data, and the portion known as `y2-evaluation`, what was used as test data for the actual evaluation. Of `C-D2b`, we used solely the `Country_Road` portion (both `Country_Road_1` and `Country_Road_2`), videos filmed on a rural country road depicting the specified action classes. This portion contains 22 video clips ranging in length from about 13.5 minutes to about 41 minutes totaling about 8.5 hours of video. Of `y2-evaluation`, we used all of the videos employed for evaluating the 'Recognition' and 'Description' tasks that were part of the Year 2 evaluation. This portion contains 11 video clips ranging in length from about 6 minutes to about 13 minutes totaling about 2 hours of video. Two of these video clips were filmed in a country-road setting while the remainder were filmed in a

[1] `http://www.visint.org/datasets#Year_2_Videos`

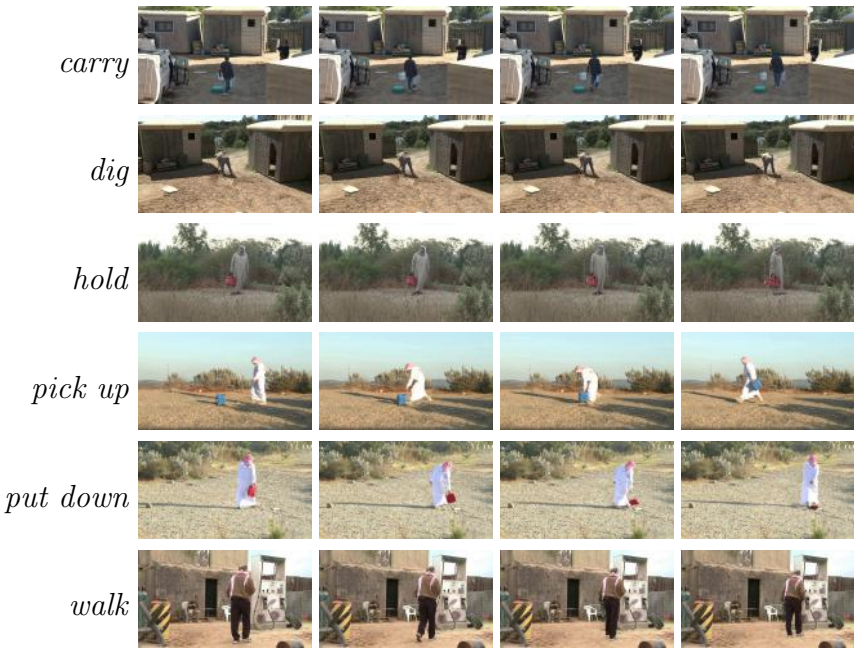
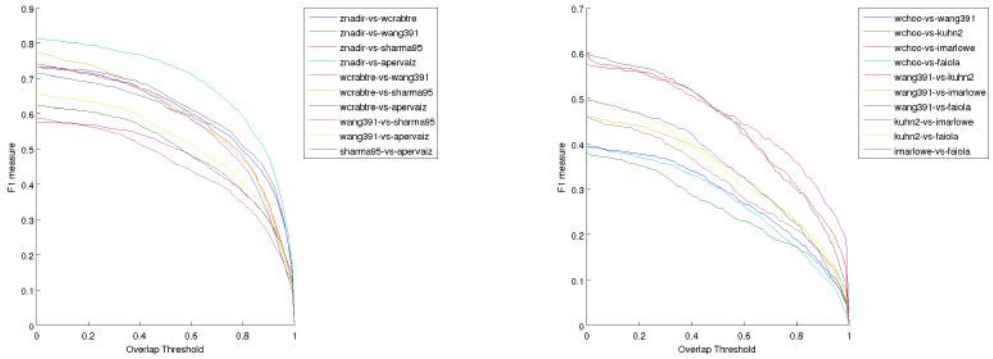


Fig. 5.1. Key frames from sample stimuli for each of the six action classes.

'Safe House' setting, a simulated middle-eastern urban environment. Nominally, this dataset depicts 24 distinct action classes: *approach*, *arrive*, *bury*, *carry*, *chase*, *dig*, *drop*, *enter*, *exchange*, *exit*, *flee*, *follow*, *give*, *hold*, *leave*, *pass*, *pick up*, *put down*, *replace*, *run*, *stop*, *take*, *turn*, and *walk*. However, the video is streaming; action occurrences start and stop at arbitrary points in the time course of the video, and often overlap.

There is no official ground-truth action labeling associated with this dataset. To remedy this, we had five humans annotate the entire `Country_Road` portion of `C-D2b` (both `Country_Road_1` and `Country_Road_2`) and had a different set of five annotators (with one annotator in common) annotate the entire set of videos for the Recognition and Description portions of `y2-evaluation`. Each annotator annotated the entire corpus portion independently, labeling each occurrence of the 24 specified action classes along with the start and end times for each occurrence. Thus we have five complete redundant annotations of the entire corpus. Having multiple



(a) (b)

Fig. 5.2. Intercoder agreement for each annotator pair on (a) the `C-D2b/Country_Road` dataset and (b) the Recognition and Description portions of the `y2-evaluation` dataset that were part of the Year 2 evaluation of the DARPA Mind's Eye program, as a function of requisite temporal overlap.

annotators allows us to measure intercoder agreement, which we did for all pairs of annotators. We considered two annotated action occurrences to match when they were labeled with the same action class and temporally overlapped by a minimum specified amount. The temporal overlap was measured using a 1-dimensional variant of the 2-dimension spatial-overlap metric used in PASCAL VOC [125], namely the ratio of the length of the intersection of the two intervals to the length of their union. We then computed the F1 score for each pair of annotators as a function of overlap. The result is shown in Fig. 5.2. The F1 score naturally decreases monotonically with increasing minimum overlap and goes to zero when the required overlap is 100%, indicating that human annotators never agree on the precise temporal extent of the actions in question. But the F1 score ranged between 0.27 and 0.8 at 50% overlap and between 0.39 and 0.81 at 0% overlap (which still requires temporal adjacency).

This surprisingly low level of human-human intercoder agreement indicates that even in this setting where the actions are easily interpretable by humans and occur largely unoccluded in an outdoor setting with an uncluttered background, the task of delineating temporal extent of action occurrences is ambiguous. Thus we selected a subset of 6 out of the 24 action classes with the highest level of intercoder agreement:

carry, *dig*, *hold*, *pick up*, *put down*, and *walk*. For each of these classes, we selected intervals of at least 2.5 seconds where at least two human annotators agreed on the label with at least 50% overlap. From these, we attempted to select 30 random 2.5-second clips for each of the six classes. The 2.5-second clips were chosen to maximally coincide with the intersection of the human-annotated intervals. However, two classes did not have sufficient clips with the requisite level of intercoder agreement: *dig* with 23 and *hold* with 26. Thus we selected a total of 169 distinct clips across all six action classes with the highest possible level of intercoder agreement.[2] Key frames from sample stimuli are shown in Fig. 5.1.

We employed a technique to further reduce the potential ambiguity in determining the intended action-class label for each stimulus. This technique was borrowed and adapted from the Natural Language Processing community. Natural language exhibits lexical polysemy: words can have multiple senses, which leads to ambiguity in contexts. WordNet represents word meanings with synsets, unordered sets of words that share a same meaning. A polysemous word with n different meanings occurs in n different synsets, along with its synonyms. For example, the verb *break* is found in the synsets $\{break, interrupt\}$ and $\{break, bust\}$. To further reduce the potential ambiguity in the intended class label depicted by each video, we constructed pairs of video clips with the same label, in the spirit of WordNet's synsets. In other words, we constructed longer stimuli as pairs of different video clips with the same intended action-class label, where each might otherwise be mildly ambiguous as to which action class was intended, but where together, the ambiguity is resolved. Sequences of such video-clip pairs constituted both the stimuli presented to human subjects during fMRI as well as training and test sets for computer-vision action recognition.

[2]The supplementary material contains the complete set of 169 video clips.

5.2 Action Recognition Software

We sought to try our corpus with as many published action-recognition methods as possible. We searched all papers on action recognition published in all conferences listed under *Computer Vision Paper Indexes*[3] since 2011, namely ICCV 2011 and 2013, CVPR 2011, 2012, and 2013, ECCV 2012, ACCV 2012, BMVC 2012, SIGGRAPH 2011, EUROGRAPHICS 2011, and IJCAI 2011, for indication that their code was publicly available. We sought **end-to-end** implementations that included both feature extraction and classification. (Some authors release only the code for feature extraction, for example binaries for STIP [111][4] and source for Dense Trajectories [45, 82, 83][5]. The lack of a compatible released classifier makes it difficult to run and further difficult to compare with the precise published method.) The only papers that we found that indicated such were for C2 [6, 127][6] and Action Bank [44].[7] C2 is particularly relevant to our comparison with fMRI as [6] claims that it

> uses a hierarchical architecture modeled after the ventral and dorsal streams of the primate visual cortex for the task of object and action recognition, respectively.

Additionally, we posted a query for available action-recognition software to CVNet which yielded a single response pointing us to the code for Stacked ISA [47].[8] Furthermore, we contacted Rogerio Feris to see if any code was collected for the study in [116]. He pointed us to a website[9] that yielded only one available system that we hadn't already been aware of, namely Velocity Histories of Tracked Keypoints (VHTK) [49].[10] As far as we can tell, these are the only published action-recognition methods for which there are corresponding publicly available **end-to-end** implementations.

[3] http://www.cvpapers.com/index.html
[4] http://www.di.ens.fr/~laptev/download.html
[5] https://lear.inrialpes.fr/people/wang/download/dense_trajectory_release_v1.2.tar.gz
[6] https://github.com/hueihan/Action_Recognition
[7] http://www.cse.buffalo.edu/~jcorso/r/actionbank/
[8] http://ai.stanford.edu/~quocle/video_release.tar.gz
[9] http://rogerioferis.com/VisualRecognitionAndSearch2014/Resources.html
[10] http://www.cs.rochester.edu/~rmessing/uradl/

Note that the released code for Stacked ISA is only able to perform binary classification and so must differ from that used to generate the published results which include evaluation of KTH that requires multi-label classification. Also note that for VHTK, the documentation for the released code states that the released code differs from that used to produce the results in the corresponding publication; the actual code used to produce the results in the corresponding publication has not been publicly released. Thus the only publicly available systems that we are aware of that can replicate the associated published results are C2 and Action Bank.

We also have access to two action-recognition software packages that are not publicly available. Cao [46] reports that they reimplemented Ryoo's method [48] as it is not publicly available. We tested against both Cao's implementation [46] of Ryoo's method [48] as well as Cao's method [46]. Further, we implemented our own classifier using the methods described in [82] on top of the publicly available source code for the Dense Trajectories [45, 82, 83] feature extraction and tested against this as well.

5.3 Computer-Vision Action-Recognition Experiments

We applied C2 [127], Action Bank [44], Stacked ISA [47], VHTK [49], Cao's implementation [46] of Ryoo's method [48], Cao's method [46], and our own implementation of the classifier described in [82] on top of the Dense Trajectories [45,82,83] feature extractor to the same dataset.[11] When running Action Bank, we used the precomputed 205-template bank that was provided with the release. These experiments employed the same eight-fold leave-one-run-out cross validation. One complication arises, however. Since the stimuli were selected randomly from a uniform distribution over the set of available video clips, the same video clip could appear both within a given run and across runs. In the case of computer-vision systems, which directly process the stimuli, this would constitute training on the test data. In particular, several of

[11] These experiments were analogous to the within-subject fMRI experiment. It would be meaningless to perform a computational analog of the cross-subject fMRI experiments because there would be no variation between different runs of the same program.

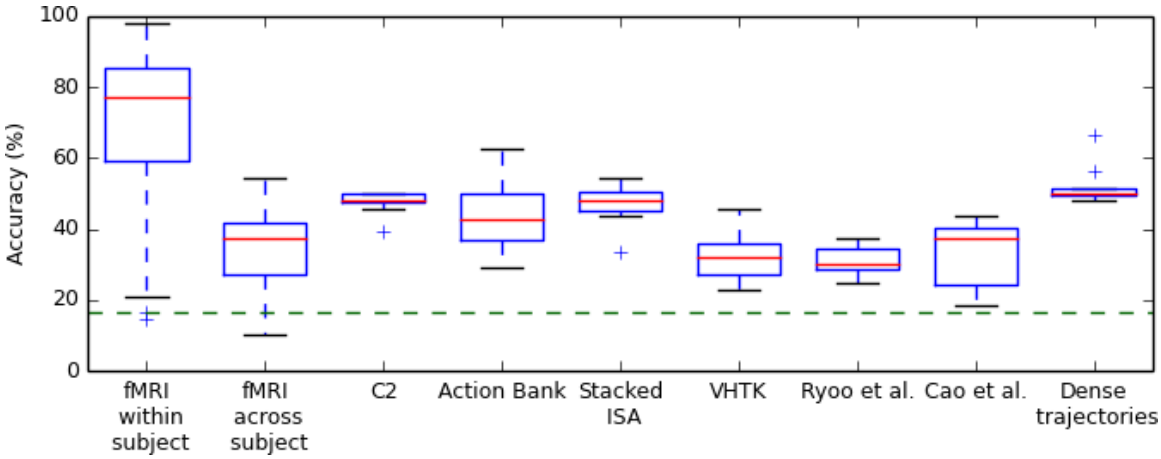


Fig. 5.3. Box plot corresponding to the results in Table 5.1, aggregated across subject and run for fMRI and aggregated across run for the computer-vision methods. Red lines indicate medians, box extents indicate upper and lower quartiles, error bars indicate maximal extents, and crosses indicate outliers. The dashed green lines indicates chance performance.

the computer-vision systems that we evaluated are memory-based and would gain an unfair advantage by recalling from memory the class labels of test videos that occur in the training set. This is not a problem for the fMRI experiments because we did not directly process the stimuli; we process the brain-scan data that was evoked by the stimuli and there is significant natural variation in such.

To ameliorate this problem when performing the computer-vision experiments, we removed from each training set any pair that contained a video clip shared with a pair in the test set. This kept each test set unmodified but resulted in slightly smaller training sets. After removing such pairs, the two video clips from each pair were temporally concatenated in the same order as presented to human subjects to yield the training and test samples for the computer-vision action-recognition experiments. The results are presented in Table 5.1 and Figs. 5.3 and 5.4. Note that all the computer-vision action-recognition systems that we tested on yield similar accuracy to the cross-subject fMRI experiments and *much* lower accuracy than the corresponding within-subject fMRI experiments.

				run							
analysis	**subject**	**mean**	**stddev**	1	2	3	4	5	6	7	8
fMRI within subject	1	0.7943	0.0783	0.8333	0.8125	0.8958	0.8542	0.7292	0.8125	0.7708	0.6458
	2	0.8880	0.0589	0.8750	0.9375	0.9792	0.9167	0.8958	0.7917	0.8333	0.8750
	3	0.7500	0.0568	0.7917	0.7083	0.7292	0.7500	0.7500	0.6458	0.8125	0.8125
	4	0.3828	0.0945	0.4583	0.5417	0.3750	0.3542	0.3750	0.2083	0.3750	0.3750
	5	0.9063	0.0686	0.8750	0.8542	0.9583	0.9583	0.9583	0.9583	0.9167	0.7708
	6	0.8385	0.0348	0.8750	0.8750	0.8542	0.8333	0.8125	0.8542	0.7708	0.8333
	7	0.5104	0.2260	0.1667	0.1458	0.6875	0.5417	0.6875	0.6875	0.6042	0.5625
	8	0.5078	0.1531	0.2083	0.6458	0.5208	0.6458	0.3958	0.4375	0.6042	0.6042
	mean	0.6973		0.6354	0.6901	0.7500	0.7318	0.7005	0.6745	0.7109	0.6849
	stddev		0.2171	0.3092	0.2557	0.2156	0.2061	0.2136	0.2450	0.1734	0.1694
fMRI across subject	1	0.2917	0.1045	0.2708	0.1458	0.2917	0.3750	0.3542	0.2708	0.1667	0.4583
	2	0.4141	0.0901	0.5417	0.5208	0.3750	0.3958	0.2500	0.3958	0.4167	0.4167
	3	0.3698	0.0761	0.4167	0.4375	0.2917	0.3750	0.3333	0.3125	0.2917	0.5000
	4	0.2917	0.1210	0.4167	0.2292	0.4792	0.2500	0.3958	0.1667	0.2292	0.1667
	5	0.3568	0.0550	0.3958	0.4167	0.3125	0.3333	0.3958	0.3750	0.3750	0.2500
	6	0.4036	0.0695	0.4375	0.3750	0.3333	0.3542	0.3333	0.5208	0.4792	0.3958
	7	0.3698	0.1677	0.1042	0.1042	0.4375	0.4792	0.3958	0.4375	0.5000	0.5000
	8	0.2865	0.0770	0.1458	0.2917	0.2917	0.3958	0.2708	0.2500	0.3750	0.2708
	mean	0.3480		0.3411	0.3151	0.3516	0.3698	0.3411	0.3411	0.3542	0.3698
	stddev		0.1068	0.1527	0.1475	0.0725	0.0647	0.0567	0.1135	0.1173	0.1254
C2 [127]		0.4740	0.0348	0.5000	0.4792	0.3958	0.4792	0.4583	0.5000	0.5000	0.4792
Action Bank [44]		0.4427	0.1112	0.5625	0.4583	0.2917	0.6250	0.3958	0.4792	0.3542	0.3750
Stacked ISA [47]		0.4688	0.0649	0.5208	0.5000	0.5417	0.4583	0.3333	0.5000	0.4375	0.4583
VHTK [49]		0.3255	0.0721	0.3750	0.2708	0.2708	0.3333	0.2292	0.3542	0.4583	0.3125
Ryoo's method* [48]		0.3125	0.0459	0.2500	0.2708	0.2917	0.3750	0.3333	0.2917	0.3750	0.3125
Cao's method [46]		0.3333	0.0964	0.3958	0.2292	0.2500	0.4375	0.1875	0.4167	0.3958	0.3542
Dense Trajectories [45, 82, 83]		0.5234	0.0634	0.6667	0.5625	0.5000	0.5000	0.4792	0.4792	0.5000	0.5000

*as implemented in Cao *et al.* [46]

Table 5.1.

Accuracy of within-subject and cross-subject classification of fMRI brain scans of subjects watching video clips on a 1-out-of-6 action-recognition task (chance performance is 0.1666), by subject and run, aggregated across subject, aggregated across run, and aggregated across subject and run. Comparison with seven computer-vision action-recognition methods, by run and aggregated across run.

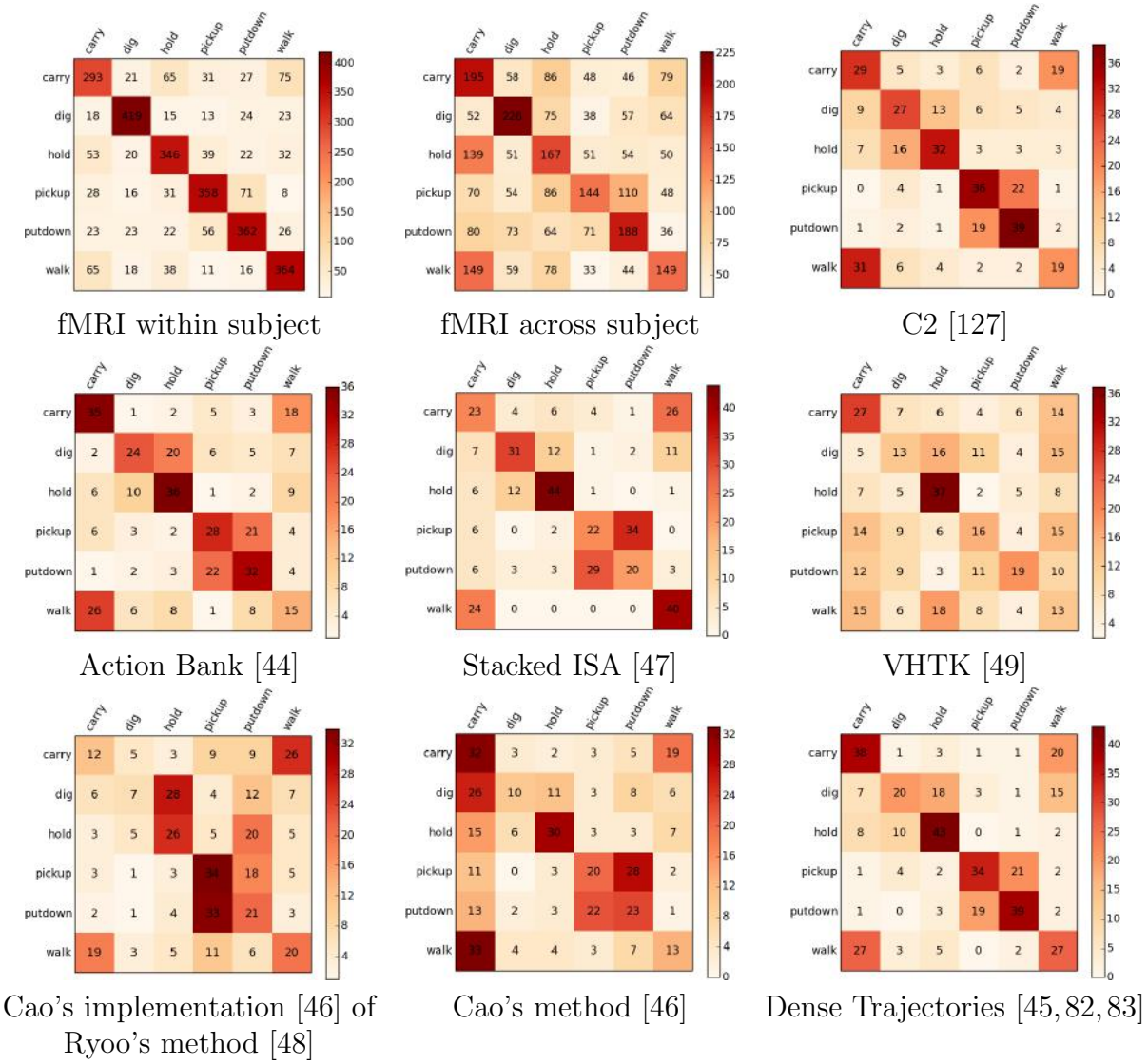


Fig. 5.4. Confusion matrices corresponding to the results in Table 5.1, aggregated across subject and run for fMRI and aggregated across run for the computer-vision methods.

5.4 Discussion

Fig. 5.3 illustrates some interesting issues. It shows that Action Bank [44] has lower median accuracy and a higher variance profile that extends to much lower accuracy than C2 [127] and Stacked ISA [47] which predate it. It shows that Cao's implementation [46] of Ryoo's method [48] and Cao's method [46] have lower median

accuracy and a much lower span of accuracies than C2 [127], Action Bank [44], and Stacked ISA [47] which predate them. It shows that Cao's method [46] has higher variance than Cao's implementation [46] of Ryoo's method [48] which predates it. Thus generally, the newer methods perform worse than the older ones; it shows that the field is basically not progressing.

Fig. 5.4 gives some indication as to why. It shows that all the computer-vision methods tested confuse *carry* and *walk* much more than fMRI, which could be explained if these methods detected these action classes solely by detecting horizontal motion. It shows that all the computer-vision methods tested confuse *dig* and *hold*, which could be explained if these methods detected these action classes solely by detecting the lack of horizontal motion. It shows that all the computer-vision methods tested confuse *pick up* and *put down*, which could be explained if these methods detected these action classes solely by detecting vertical motion, without detecting the object being picked up or put down and without accounting for the temporal ordering of the motion. It also suggests that the semantics of human perception may play a role in action recognition, which the statistical classifiers cannot pick up. This is all to be expected when one considers that, generally, most current computer-vision methods employ techniques that look solely at local image features at very short spatial and/or temporal scales. Even Action Bank ultimately relies on local image gradients to define its templates. And none of the methods, even Dense Trajectories which can incorporate a person detector, detect the objects being interacted with as part of the action class. In other words, they don't detect the object being *carried*, the shovel used to *dig*, the hole in the ground that is *dug*, or the objects being *held*, *picked up*, or *put down*. Moreover, they don't model the time course of the changing human pose and relative position and orientation of the person and the object interacted with. These are the semantic characteristics of the action class. Thus it shows that none of these methods are, in fact, doing action recognition.

5.5 Conclusion

Despite the explosive growth of interest in action recognition over the past three years and the perfect or near-perfect classification accuracies reported on datasets with small numbers of action classes, we show that the problem remains difficult. Uniformly, the newer methods we tried performed *no better* than or even *worse* than the older methods on this new dataset. One potential explanation is that the field as a whole is collectively overfitting to the datasets, *i.e.*, having individual researchers repeatedly hone their methods to a small number of datasets and having the community collectively perform hill climbing on these datasets is tantamount to training on the test data. We advocate ameliorating this problem by testing methods on *read-once data*, data that has never been processed by the method.

6. LINGUISTICS MEETS VIDEO SEARCH

This chapter describes experiments conducted to evaluate the performance of the sentence tracker [21] for video retrieval on a large realistic dataset comprised of ten full length Hollywood movies, cut into a large number of short video clips. Given a sentence drawn from a small grammar and vocabulary, the system returns a set of the top hits for that sentence, those clips which have been determined to best match the semantics of that sentence. An experiment was undertaken using Amazon Mechanical Turk to obtain independent judgments of the truth of query sentences on all clips of this dataset in order to measure the precision and recall of the system. A comparison baseline system was also constructed using state of the art object detectors and action recognition systems, and its performance was also evaluated.

6.1 Experiments

We present three experiments which test video retrieval using sentential queries. All three use the same video corpus but use different query corpora.

6.1.1 The Ten Westerns Video Corpus

Our video corpus consists of 10 full-length Hollywood movies, nominally of the genre westerns. This corpus is very challenging and demonstrates the ability of our approach to handle videos found in the wild and not filmed specifically for this task: Black Beauty (Warner Brothers, 1994), The Black Stallion (MGM, 1979), Blazing Saddles (Warner Brothers, 1974), Easy Rider (Columbia Pictures, 1969), The Good the Bad and the Ugly (Columbia Pictures, 1966), Hidalgo (Touchstone Pictures, 2004), National Velvet (MGM, 1944), Once Upon a Time in Mexico (Columbia Pic-

tures, 2003), Seabiscuit (Universal Pictures, 2003), and Unforgiven (Warner Brothers, 1992). In total, this video corpus has 1187 minutes of video, roughly 20 hours. The appendix specifies the duration and spatial and temporal resolution of each movie.

We temporally downsampled all videos to 6fps but kept their original spatial resolutions, splitting them into 37186 clips, each clip nominally being 18 frames (3 seconds) long, while overlapping the previous clip by 6 frames. This overlap ensures that actions that might otherwise occur on clip boundaries will also occur as part of a clip. While there is prior work on shot segmentation [128] we did not employ it for two reasons. First, it complicates the system and provides an avenue for additional failure modes. Second, the approach taken here is able to find an event inside a longer video with multiple events. The only reason why we split the videos into clips is to return multiple hits.

6.1.2 Query Corpora

The grammer used allows for queries that describe people interacting with horses, hence our choice of genre for the video corpus, namely westerns. We generated two of the three query corpora from this grammar. The first consisted of 9 SVO queries generated by the grammar. We omitted the 3 SVO queries that involve people riding people, horses riding people, and horses riding horses. We refer to this collection of 9 queries as the SVO queries. The second consisted of the 204 queries generated by the template included in the appendix. This consisted of all queries generated by the grammar except those that involve a PP in an NP and further restricting the lexical $\mathrm{PP_M}$ to be appropriate for the verb. For this query corpus we included all queries, including those that involve people riding people, horses riding people, and horses riding horses. We refer to this collection of 204 queries as the synthetic queries.

The third collection of queries was elicited from 300 distinct, disinterested, independent, and anonymous humans via Amazon Mechanical Turk through a mock up of our system, as described in the appendix. We obtained 3000 unrestricted queries,

completely unconstrained as to grammar and lexicon. We discarded 22 blank queries and 351 that violated the instructions given to workers, as described in the appendix. We processed all remaining 2627 queries by mapping them to synthetic queries using a spelling and grammar correction process based on Levenshtein distance, as described in the appendix. We refer to this collection of 2627 queries as the human queries. The mock up did not expose this spelling and grammar correction process to the workers, who simply entered queries, which were recorded, and obtained search results. We evaluated the truth of the retrieved results relative to the original human queries, not their mapping to the synthetic queries.

6.1.3 Models

A requirement for determining whether a video depicts a query, and the degree to which it depicts that query, is to detect the objects that might fill roles in that query. Previous work has shown that people and horses are among the easiest-to-detect objects, although the performance of object detectors, even for these classes, remains extremely low. To ensure that we did not test on the training data, we employed previously-trained object models that have not been trained on these videos but have instead been trained on PASCAL VOC. We use models provided with the software release associated with Sadeghi & Forsyth [129] which were trained by the UoCTTI_LSVM-MDPM team (the authors of Felzenszwalb *et al.* [32, 130]) for the 2009 Challenge. On the 2009 Challenge, the *person* model achieves an AP score of 41.5% and the *horse* model achieves an AP score of 38.0%. We note that the improvement in AP scores for these object classes in subsequent years of the Challenge has been minor. When running the object detectors, we set the non-maximal-suppression parameter to 0.7 and use at most the top 4 detections returned for each class in each frame.

We also require settings for the 9 parameters which are required to produce the predicates which encode the semantics of the words in this grammar. For this purpose,

we judiciously selected values for these parameters that are consistent with their intent: **far** = 180, **close** = 120, **stationary** = 2, **Δclosing** = 3, **Δangle** = 45°, **Δpp** = 50, **Δquickly** = 30, **Δslowly** = 30, and **overlap** = 0.1. Yu & Siskind *et al.* [20] present a strategy for training the parameters of a lexicon of words given a video corpus.

6.1.4 Baseline

We compared the performance of the sentence tracker against a baseline on the SVO queries. We compare against a baseline only for the SVO queries and not the synthetic and human queries because we know of no other system that can support the more complex syntax and ontology in these query corpora. This baseline employs the same approach that is used in state-of-the-art video-search systems in that it uses a bag-of-words approach to search independently for the subject and object of an SVO query using object detection and the verb of an SVO query using event detection. We do not compare against any particular existing system because no current system employs state-of-the-art object or event detectors and thus any such system would be severely handicapped by its inability to reliably detect people, horses, and the particular verbs we search for.

Our baseline operates as follows. We first apply an object detector to each frame of every clip to detect people and horses. For comparison purposes, we employ the same object detector and pretrained models as used for the experiments with the sentence tracker, including passing the raw detector score through the same sigmoid. We rank the clips by the average score of the top detection in each frame. If the query sentence contains only the word *person*, we rank only by the person detections. If the query sentence contains only the word *horse*, we rank only by the horse detections. If the query sentence contains both the words *person* and *horse*, we rank by the average of the top person and top horse detection in each frame. We then apply a binary event detector to eliminate clips from the ranking that do not depict the event specified by

the verb. For this purpose, we employ a state-of-the-art event detector, namely that of Kuehne *et al.* [6]. We train that detector on 70 positive and 70 negative samples of each verb and remove those samples from the test set. We then report the top 1, 3, 5, and 10 ranked clips that satisfy the event detector and compare those clips against the top 1, 3, 5, and 10 clips produced by our method.

6.1.5 Evaluation Procedure

For each query, we scored every clip paired with that query and return the top 1, 3, 5, and 10 best-scoring clips for that query. Each of these top 10 clips was annotated by a collection of nominally 5 distinct, disinterested, independent, and anonymous humans via Amazon Mechanical Turk. Each judge was presented with a query and associated hit and asked: is this query true of this clip? The precise details of how such assessment was performed are described in the appendix.

6.1.6 Results

Our results are summarized in Fig. 6.1. The left column summarizes the experiments with the SVO queries. Our approach yields significantly higher precision than the baseline on the SVO queries. Precision of the sentence tracker on the SVO queries varies as a function of recall as controlled by the threshold on the sentence-tracker score. Note that it is not possible to achieve high recall with our method, because we employ hard FSMs to model sentential semantics which cannot be overcome by any threshold on sentence-tracker score because such is $-\infty$ when the FSM is violated. Recall is thus limited to about 10^{-2}. Precision is around 0.3 for most of the attainable recall range. It reaches a peak of 0.5 when recall is about 2×10^{-5}. Its lowest value is 0.125 with a similar recall.

The right three columns summarize the experiments with the synthetic and human queries in the top and bottom rows respectively. For the second and third columns, no threshold on sentence-tracker score was employed; we evaluated the top 1, 3, 5,

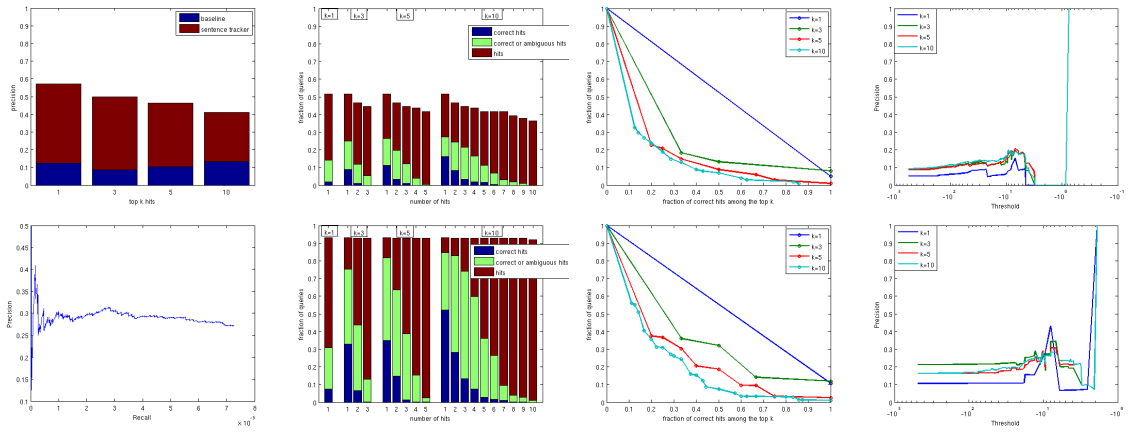


Fig. 6.1. (top left) Comparison of average precision in the top 1, 3, 5, and 10 hits, over the SVO queries for both the baseline and the sentence tracker. (bottom left) Precision/recall curve over the SVO queries for the sentence tracker. Results for synthetic (top row) and human (bottom row) queries in the top 1, 3, 5, and 10 hits (right three columns). (second column) Fraction of queries with at least the the indicated number of hits, correct or ambiguous hits, and correct hits. (third column) Fraction of queries that have at least the indicated fraction of correct hits. (fourth column) Precision of returned hits as a function of threshold.

and 10 hits returned. Because of the stringent FSM model, the sentence tracker can return fewer than the requisite number of hits, even without a threshold. Thus the red bars in the second column depict the fraction of the queries for which at least the indicated number of hits were returned. Because the human judges were sometimes divided as to whether the queries were true of the hits, we classified these hits as correct, ambiguous, or incorrect, as described in the appendix. The green bars depict the fraction of the queries for which the indicated number of correct or ambiguous hits were returned, while the blue bars depict the fraction of the queries for which the indicated number of correct hits were returned.

The third column depicts the fraction of the queries that yield at least the indicated fraction of correct hits. For example, with the synthetic queries, slightly more than 30% of the queries yield 10% or more correct hits in the top 10. As a point of comparison, with the human queries, slightly more than 55% of the queries yield 10% or more correct hits in the top 10. Note that for much of the range, the precision in the top hits requested for human queries exceeds that of the synthetic queries.

The fourth column depicts the variation in average precision as a function of a threshold on the sentence-tracker score. As the threshold nears zero, the sentence tracker becomes very precise. As the threshold tends to $-\infty$, the average precision asymptotes. Again note that overall precision for the human queries is significantly higher than that of the synthetic queries over almost all of the range of thresholds.

We highlight the usefulness of this approach in Fig. 6.2 where we show one of the top few hits for a variety of different synthetic and human queries. Note that two pairs of similar queries, both *The person approached the horse* and *The horse approached the person* as well as *The person approached the horse slowly from the left* and *The horse approached the person slowly from the left*, yield different but appropriate results. With existing systems, both queries in each pair would provide the same hits as they treat sentences as conjunctions of words.

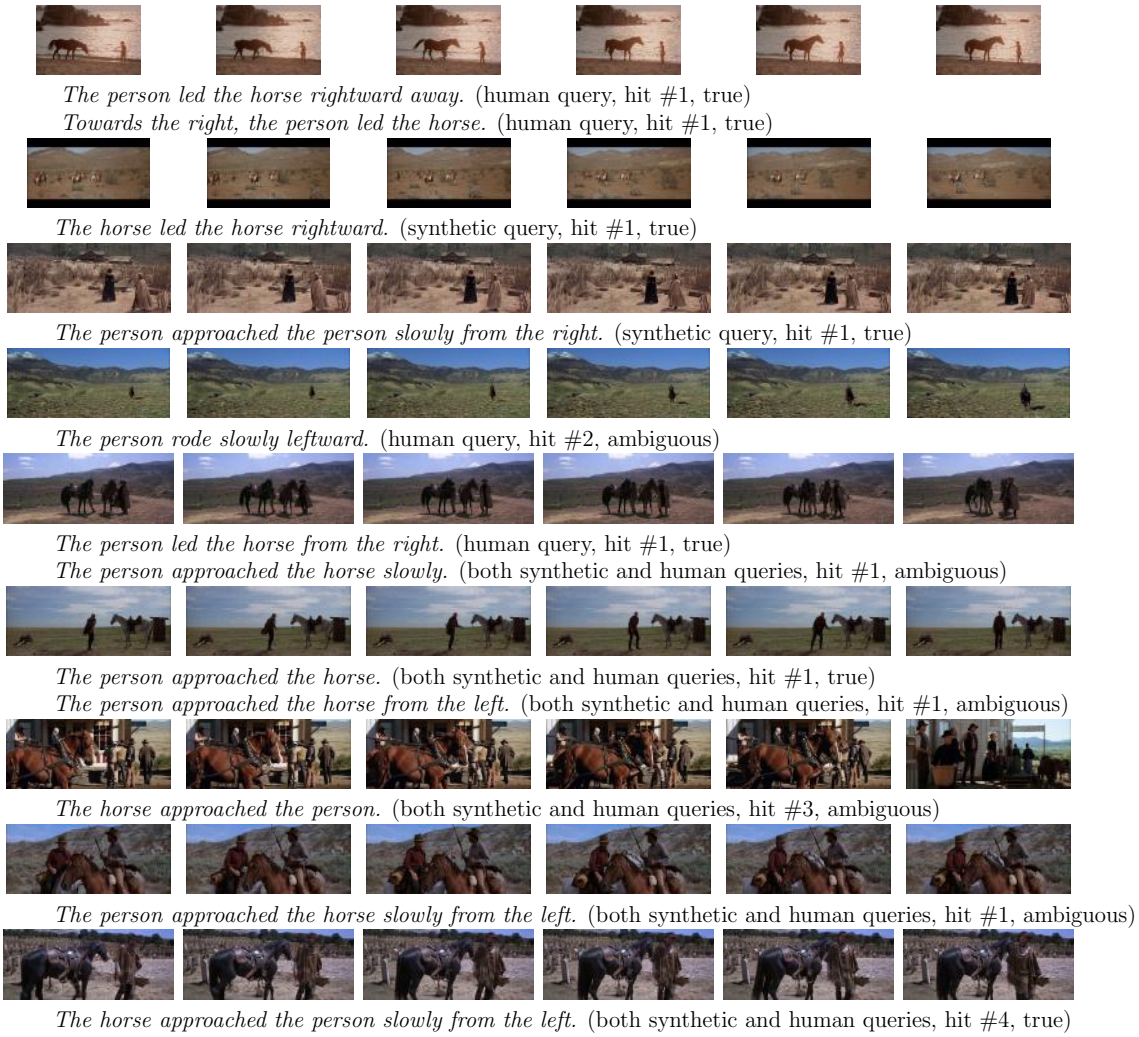


Fig. 6.2. Frames from hits returned for several synthetic and human queries. Some clips are returned for multiple queries. As indicated above, theses hits were judged as correct or ambiguous for the associated query by human judges.

REFERENCES

REFERENCES

[1] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of web videos," *Machine Vision and Applications*, vol. 24, no. 5, pp. 971–981, 2013.

[2] M. D. Rodriguez, M. Sullivan, and M. Shah, "Action MACH: Maximum average correlation height filter for action recognition," in *CVPR*, 2008, pp. 1–8.

[3] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *ICCV*, 2005, pp. 1395–1402.

[4] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: A local svm approach," in *ICPR*, 2004, pp. 32–36.

[5] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification," in *ECCV*, 2010, pp. 392–405.

[6] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, "HMDB: A large video database for human motion recognition," in *ICCV*, 2011, pp. 2556–2563.

[7] X. Wu, D. Xu, L. Duan, and J. Luo, "Action recognition using context and appearance distribution features," in *CVPR*, 2011, pp. 489–496.

[8] C. Wang, Y. Wang, and A. L. Yuille, "An approach to pose-based action recognition," in *CVPR*, 2013, pp. 915–922.

[9] I. Everts, J. C. van Gemert, and T. Gevers, "Evaluation of color STIPS for human action recognition," in *CVPR*, 2013, pp. 2850–2857.

[10] C. Yuan, W. Hu, G. Tian, S. Yang, and H. Wang, "Multi-task sparse learning with Beta process prior for action recognition," in *CVPR*, 2013, pp. 423–429.

[11] C. Yuan, X. Li, W. Hu, H. Ling, and S. Maybank, "3D R transform on spatio-temporal interest points for action recognition," in *CVPR*, 2013, pp. 724–730.

[12] O. Oreifej and M. Shah, *Robust Subspace Estimation Using Low-Rank Optimization*. Springer, 2014, ch. 5, pp. 55–67.

[13] H. Wang, A. Kläser, C. Schmid, and C. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.

[14] J. Zhu, B. Wang, X. Yang, W. Zhang, and Z. Tu, "Action recognition with Actons," in *ICCV*, 2013, pp. 3559–3566.

[15] D. Oneata, J. Verbeek, and C. Schmid, "Action and event recognition with Fisher vectors on a compact feature set," in *ICCV*, 2013, pp. 1817–1824.

[16] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos "in the wild"," in *CVPR*, 2009, pp. 1996–2003.

[17] N. Ikizler-Cinbis and S. Sclaroff, "Object, scene and actions: Combining multiple features for human action recognition," in *ECCV*, 2010, pp. 494–507.

[18] L. Lin, Y. Xu, X. Liang, and J. Lai, "Complex background subtraction by pursuing dynamic spatio-temporal models," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 3191–3202, 2014.

[19] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[20] H. Yu and J. M. Siskind, "Grounded language learning from video described with sentences," in *ACL*, 2013, pp. 53–63.

[21] N. Siddharth, A. Barbu, and J. M. Siskind, "Seeing what you're told: Sentence-guided activity recognition in video," in *CVPR*, 2014.

[22] A. Barbu, N. Siddharth, A. Michaux, and J. M. Siskind, "Simultaneous object detection, tracking, and event recognition," *Advances in Cognitive Systems*, vol. 2, pp. 203–220, 2012.

[23] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005, pp. 886–893.

[24] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state Markov chains," *The Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[25] Z. Chen, W. Zuo, Q. Hu, and L. Lin, "Kernel sparse representation for time series classification," *Information Sciences*, vol. 292, no. 20, pp. 15–26, 2015.

[26] K. Tang, L. Fei-Fei, and D. Koller, "Learning latent temporal structure for complex event detection," in *CVPR*, 2012, pp. 1250–1257.

[27] X. Liang, L. Lin, and L. Cao, "Learning latent spatio-temporal compositional model for human action recognition," in *ACM International Conference on Multimedia*, 2013, pp. 263–272.

[28] D. Wu and L. Shao, "Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition," in *CVPR*, 2014, pp. 724–731.

[29] P. Banerjee and R. Nevatia, "Pose filter based hidden-crf models for activity detection," in *ECCV*, 2014, pp. 711–726.

[30] Y. Tian, R. Sukthankar, and M. Shah, "Spatiotemporal deformable part models for action detection," in *CVPR*, 2013, pp. 2642–2649.

[31] B. Yao, B. Nie, Z. Liu, and S.-C. Zhu, "Animated pose templates for modeling and detecting human actions," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 3, pp. 436–452, 2014.

[32] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[33] N. Dalal and B. Triggs, "Human detection using oriented histograms of flow and appearance," in *ECCV*, 2006, pp. 428–441.

[34] M. S. Ryoo and J. K. Aggarwal, "Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities," in *ICCV*, 2009, pp. 1593–1600.

[35] L. Niles and H. Silverman, "Combining hidden Markov model and neural network classifiers," in *ICASSP*, 1990, pp. 417–420.

[36] A. J. Viterbi, "Error bounds for convolutional codes and an asymtotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–267, 1967.

[37] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, pp. 265–292, 2002.

[38] A. Griewank, "On automatic differentiation," in *Mathematical Programming: recent developments and applications*, M. Iri and K. Tanabe, Eds. Kluwer Academic, 1989, pp. 83–108.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[40] P. S. Gopalakrishnan, D. Kanevsky, A. Nadas, and D. Nahamoo, "An inequality for rational functions with applications to some statistical estimation problems," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 107–113, 1991.

[41] D. Barrett, R. Xu, H. Yu, and J. M. Siskind, "Collecting and annotating the large continuous action dataset," 2015, manuscript in review.

[42] Z. Lin, Z. Jiang, and L. Davis, "Recognizing actions by shape-motion prototype trees," in *ICCV*, 2009, pp. 444–451.

[43] P. Nagar and A. Agrawal, "Geometric invariant model based human action recognition," in *Industrial and Information Systems (ICIIS)*, 2014, pp. 1–6.

[44] S. Sadanand and J. J. Corso, "Action Bank: A high-level representation of activity in video," in *CVPR*, 2012, pp. 1234–1241.

[45] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013, pp. 3551–3558.

[46] Y. Cao, D. Barrett, A. Barbu, S. Narayanaswamy, H. Yu, A. Michaux, Y. Lin, S. Dickinson, J. M. Siskind, and S. Wang, "Recognizing human activities from partially observed videos," in *CVPR*, 2013, pp. 2658–2665.

[47] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *CVPR*, 2011, pp. 3361–3368.

[48] M. S. Ryoo, "Human activity prediction: Early recognition of ongoing activities from streaming videos," in *ICCV*, 2011, pp. 1036–1043.

[49] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *ICCV*, 2009, pp. 104–111.

[50] T. Lan, Y. Wang, and G. Mori, "Discriminative figure-centric models for joint action localization and recognition," in *ICCV*, 2011, pp. 2003–2010.

[51] C. L. Zitnick and P. Dollár, "Edge Boxes: Locating object proposals from edges," in *ECCV*, 2014, pp. 391–405.

[52] M. MacMahon, B. Stankiewicz, and B. Kuipers, "Walk the talk: connecting language, knowledge, and action in route instructions," in *AAAI*, 2006, pp. 1475–1482.

[53] T. Kollar, S. Tellex, D. Roy, and N. Roy, "Toward understanding natural language directions," in *International Conference on Human-Robot Interaction*, 2010, pp. 259–266.

[54] C. Matuszek, D. Fox, and K. Koscher, "Following directions using statistical machine translation," in *Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction*, 2010, pp. 251–258.

[55] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy, "Understanding natural language commands for robotic navigation and mobile manipulation," in *AAAI*, 2011, pp. 1507–1514.

[56] D. L. Chen and R. J. Mooney, "Learning to interpret natural language navigation instructions from observations," in *AAAI*, 2011, pp. 859–865.

[57] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox, "Learning to parse natural language commands to a robot control system," in *International Symposium on Experimental Robotics*, 2012, pp. 403–415.

[58] Y. Artzi and L. Zettlemoyer, "Weakly supervised learning of semantic parsers for mapping instructions to actions," *Transactions of the Association for Computational Linguistics*, vol. 1, no. 1, pp. 49–62, 2013.

[59] S. Tellex, P. Thaker, J. Joseph, and N. Roy, "Learning perceptually grounded word meanings from unaligned parallel data," *Machine Learning*, vol. 92, no. 2, pp. 151–167, 2014.

[60] S. Dobnik, S. Pulman, P. Newman, and A. Harrison, "Teaching a robot spatial expressions," *Proceedings of the Second ACL-SIGSEM Workshop on The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*, 2005.

[61] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein, "Mobile robot programming using natural language," *Robotics and Autonomous Systems*, vol. 38, no. 3, pp. 171–181, 2002.

[62] S. Teller, M. R. Walter, M. Antone, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, J. Glass, J. P. How, A. S. Huang *et al.*, "A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments," in *ICRA*, 2010, pp. 526–533.

[63] A. Koller, K. Striegnitz, A. Gargett, D. Byron, J. Cassell, R. Dale, J. Moore, and J. Oberlander, "Report on the second nlg challenge on generating instructions in virtual environments (GIVE-2)," in *Proceedings of the 6th International Natural Language Generation Conference*, 2010, pp. 243–250.

[64] T. K. Harris, S. Banerjee, and A. I. Rudnicky, "Heterogeneous multi-robot dialogues for search tasks," in *Proceedings of the AAAI Spring Symposium Intelligence*, 2005.

[65] M. R. Marge, A. K. Pappu, B. Frisch, T. K. Harris, and A. I. Rudnicky, "Exploring spoken dialog interaction in human-robot teams," in *Robots, Games, and Research: Success Stories in USARSim, IROS Workshop*, 2009.

[66] A. Pappu and A. Rudnicky, "The structure and generality of spoken route instructions," in *The Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2012, pp. 99–107.

[67] J. Fasola and M. J. Mataric, "Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 143–150.

[68] P. McGuire, J. Fritsch, J. J. Steil, F. Rothling, G. A. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter, "Multi-modal human-machine communication for instructing robot grasping tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2002, pp. 1082–1088.

[69] F. Doshi and N. Roy, "Spoken language interaction with model uncertainty: an adaptive human-robot interaction system," *Connection Science*, vol. 20, no. 4, pp. 299–318, 2008.

[70] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox, "A joint model of language and perception for grounded attribute learning," in *International Conference on Machine Learning*, 2012, pp. 1671–1678.

[71] L. She, S. Yang, Y. Cheng, Y. Jia, J. Y. Chai, and N. Xi, "Back to the blocks world: Learning new actions through situated human-robot dialogue," in *The Annual Meeting of the Special Interest Group on Discourse and Dialogue*, 2014, pp. 89–97.

[72] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[73] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Empirical Methods on Natural Language Processing*, 2014, pp. 740–750.

[74] Wikipedia, "List of English prepositions," 2015, [Online; accessed 4-June-2015]. [Online]. Available: http://en.wikipedia.org/w/index.php?title=List_of_English_prepositions&oldid=662161282

[75] M. Abramowitz and I. A. Stegun, *Handbook of mathematical functions*. Dover New York, 1972.

[76] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–71, 1970.

[77] L. E. Baum, "An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process," *Inequalities*, vol. 3, pp. 1–8, 1972.

[78] R. Dale and E. Reiter, "Computational interpretations of the gricean maxims in the generation of referring expressions," *Cognitive Science*, vol. 19, no. 2, pp. 233–263, 1995.

[79] J. Yamoto, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden Markov model," in *CVPR*, 1992, pp. 379–385.

[80] Y. Wang and G. Mori, "Human action recognition by semilatent topic models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 10, pp. 1762–1674, 2009.

[81] S. Maji, L. Bourdev, and J. Malik, "Action recognition from a distributed representation of pose and appearance," in *CVPR*, 2011, pp. 3177–3184.

[82] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *International Journal of Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.

[83] ——, "Action recognition by dense trajectories," in *CVPR*, 2011, pp. 3169–3176.

[84] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *International Conference on Computer Vision*, vol. 2, 2005, pp. 1395–1402.

[85] A. Gaidon, Z. Harchaoui, and C. Schmid, "Activity representation with motion hierarchies," *International Journal of Computer Vision*, vol. 107, no. 3, pp. 219–238, 2014.

[86] L.-J. Li and L. Fei-Fei, "What, where and who? Classifying events by scene and object recognition," in *ICCV*, 2007, pp. 1–8.

[87] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden Markov models for complex action recognition," in *CVPR*, 1997, pp. 994–999.

[88] Z. Wang, E. E. Kuruoglu, X. Yang, Y. Xu, and S. Yu, "Event recognition with time varying hidden Markov model," in *ICASSP*, 2009, pp. 1761–1764.

[89] J. Liu, B. Kuipers, and S. Savarese, "Recognizing human actions by attributes," in *CVPR*, 2011, pp. 3337–3344.

[90] G. Willems, T. Tuytelaars, and L. Van Gool, "An efficient dense and scale-invariant spatio-temporal interest point detector," in *ECCV*, 2008, pp. 650–663.

[91] A. Kojima, T. Tamura, and K. Fukunaga, "Natural language description of human activities from video images based on concept hierarchy of actions," *International Journal of Computer Vision*, vol. 50, no. 2, pp. 171–184, 2002.

[92] R. Gopalan, "Joint sparsity-based representation and analysis of unconstrained activities," in *CVPR*, 2013, pp. 2738–2745.

[93] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local SVM approach," in *International Conference on Pattern Recognition*, vol. 3, 2004, pp. 32–36.

[94] A. Gupta and L. S. Davis, "Objects in action: An approach for combining action understanding and object perception," in *CVPR*, 2007, pp. 1–8.

[95] P. Das, C. Xu, R. F. Doell, and J. J. Corso, "A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching," in *CVPR*, 2013, pp. 2634–2641.

[96] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis, "Representing videos using mid-level discriminative patches," in *CVPR*, 2013, pp. 2571–2578.

[97] J. M. Siskind and Q. Morris, "A maximum-likelihood approach to visual event classification," in *ECCV*, 1996, pp. 347–360.

[98] P. Hanckmann, K. Schutte, and G. J. Burghouts, "Automated textual descriptions for a wide range of video events with 48 human actions," in *ECCV Workshops*, 2012, pp. 372–380.

[99] M. U. G. Khan and Y. Gotoh, "Describing video contents in natural language," in *Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, 2012, pp. 27–35.

[100] C. Fernández Tena, P. Baiget, X. Roca, and J. Gonzàlez, "Natural language descriptions of human behavior from video sequences," in *Advances in Artificial Intelligence*, 2007, pp. 279–292.

[101] C. Yuan, X. Li, W. Hu, H. Ling, and S. Maybank, "3D R transform on spatio-temporal interest points for action recognition," in *CVPR*, 2013, pp. 724–730.

[102] M. U. G. Khan, L. Zhang, and Y. Gotoh, "Human focused video description," in *ICCV Workshops*, 2011, pp. 1480–1487.

[103] K. K. Reddy and M. Shah, "Recognizing 50 human action categories of web videos," *Machine Vision and Applications*, vol. 24, no. 5, pp. 971–981, 2013.

[104] M. Rohrbach, W. Qiu, I. Titov, S. Thater, M. Pinkal, and B. Schiele, "Translating video content to natural language descriptions," in *ICCV*, 2013, pp. 433–440.

[105] D. J. Moore, I. A. Essa, and M. H. Heyes, "Exploiting human actions and object context for recognition tasks," in *ICCV*, 1999, pp. 80–86.

[106] S. Guadarrama, N. Krishnamoorthy, G. Malkarnenkar, R. Mooney, T. Darrell, and K. Saenko, "Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition," in *ICCV*, 2013, pp. 2712–2719.

[107] P. F. Dominey and J.-D. Boucher, "Learning to talk about events from narrated video in a construction grammar framework," *Artificial Intelligence*, vol. 167, no. 1-2, pp. 31–61, 2005.

[108] M. U. G. Khan, L. Zhang, and Y. Gotoh, "Towards coherent natural language description of video streams," in *ICCV Workshops*, 2011, pp. 664–671.

[109] Y. Song, L.-P. Morency, and R. Davis, "Action recognition by hierarchical sequence summarization," in *CVPR*, 2013, pp. 3562–3569.

[110] Y. Ke, R. Sukthankar, and M. Hebert, "Event detection in crowded videos," in *ICCV*, 2007, pp. 1–8.

[111] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[112] J. M. Siskind, "Visual event classification via force dynamics," in *AAAI*, 2000, pp. 149–155.

[113] G. Xu, Y.-F. Ma, H. Zhang, and S. Yang, "Motion based event recognition using HMM," in *ICPR*, 2002, pp. 831–834.

[114] A. Barbu, N. Siddharth, A. Michaux, and J. M. Siskind, "Simultaneous object detection, tracking, and event recognition," *Advances in Cognitive Systems*, vol. 2, pp. 203–220, 2012.

[115] N. Krishnamoorthy, G. Malkarnenkar, R. J. Mooney, K. Saenko, and S. Guadarrama, "Generating natural-language video descriptions using text-mined knowledge," in *AAAI*, 2013, pp. 541–547.

[116] H. Liu, R. Feris, and M.-T. Sun, *Benchmarking datasets for human activity recognition*. Springer, 2011, ch. 20, pp. 411–427.

[117] A. Efros, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *ICCV*, 2003, pp. 726–733.

[118] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *CVPR*, 2008, pp. 1–8.

[119] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *CVPR*, 2009.

[120] H. Uemura, S. Ishikawa, and K. Mikolajczyk, "Feature tracking and motion compensation for action recognition," in *British Machine Vision Conference*, 2008, pp. 1–10.

[121] J. Yuan, Z. Liu, and Y. Wu, "Discriminative subvolume search for efficient action detection," in *CVPR*, 2009, pp. 2442–2449.

[122] A. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and TRECVID," in *ACM International Conference on Multimedia Information Retrieval*, 2006, pp. 321–330.

[123] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: a dataset of 101 human actions classes from videos in the wild," *Computing Research Repository*, vol. abs/1212.0402, 2012.

[124] S. Oh, A. Hoogs, A. Perera, N. Cuntoor, C.-C. Chen, J. T. Lee, S. Mukherjee, J. Aggarwal, H. Lee, L. Davis, E. Swears, X. Wang, Q. Ji, K. Reddy, M. Shah, C. Vondrick, H. Pirsiavash, D. Ramanan, J. Yuen, A. Torralba, B. Song, A. Fong, A. Roy-Chowdhury, and M. Desai, "A large-scale benchmark dataset for event recognition in surveillance video," in *CVPR*, 2011, pp. 3153–3160.

[125] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[126] A. Barbu, D. P. Barrett, W. Chen, N. Siddharth, C. Xiong, J. J. Corso, C. D. Fellbaum, C. Hanson, S. J. Hanson, S. Hélie, E. Malaia, B. A. Pearlmutter, J. M. Siskind, T. M. Talavage, and R. B. Wilbur, "Seeing is worse than believing: Reading people's minds better than computer-vision methods recognize actions," in *ECCV*, 2014, pp. 612–627.

[127] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *International Conference on Computer Vision*, 2007, pp. 1–8.

[128] M. Cooper, T. Liu, and E. Rieffel, "Video segmentation via temporal pattern classification," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 610–618, 2007.

[129] M. A. Sadeghi and D. Forsyth, "Fast template evaluation with vector quantization," in *NIPS*, 2013, pp. 2949–2957.

[130] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *CVPR*, 2010, pp. 2241–2248.

VITA

VITA

Daniel Paul Barrett received the B.S. degree in Computer Engineering from Purdue University in 2011. He is currently a Ph.D. student in the School of Electrical and Computer Engineering at Purdue University. His research interests include computer vision, robotics, and artificial intelligence, particularly their intersection, where a robot perceives, learns about, and acts on the world through noisy real-world camera and sensor input.