Detecting social actions of fruit flies

Eyrun Eyjolfsdottir[1], Steve Branson[1], Xavier P. Burgos-Artizzu[1], Eric D. Hoopfer[1], Jonathan Schor[1], David J. Anderson[1,2], and Pietro Perona[1]

[1]California Institute of Technology, Pasadena, California, USA
[2]Howard Hughes Medical Institute (HHMI)

Abstract. We describe a system that tracks pairs of fruit flies and automatically detects and classifies their actions. We compare experimentally the value of a frame-level feature representation with the more elaborate notion of 'bout features' that capture the structure within actions. Similarly, we compare a simple sliding window classifier architecture with a more sophisticated structured output architecture, and find that window based detectors outperform the much slower structured counterparts, and approach human performance. In addition we test our top performing detector on the CRIM13 mouse dataset, finding that it matches the performance of the best published method. Our Fly-vs-Fly dataset contains 22 hours of video showing pairs of fruit flies engaging in 10 social interactions in three different contexts; it is fully annotated by experts, and published with articulated pose trajectory features.

1 Introduction

Machine understanding of human behavior is potentially the most useful and transformative application of computer vision. It will allow machines to be better aware of their environment, enable rich and natural human-machine interaction, and it will unleash new applications in a number of industries including automotive, entertainment, surveillance and assisted living. Development of automated vision systems that can understand human behavior requires progress in object detection, pose estimation, tracking, action classification and detection, and activity analysis. Progress on the latter (actions and activities) is hampered by two difficulties. First, tracking and pose estimation is very difficult in humans due to variation in clothing, the amount of occlusion in natural environments and in social conditions, and the sheer complexity and number of human body motions. Second, it is difficult (both technically and legally) to film large numbers of humans acting spontaneously while they perform interesting activities. As a result, human action datasets are small and unrepresentative, especially when social behavior is concerned (see Table 1).

A good strategy for computer vision researchers to make progress on behavior analysis is to shift their attention to the simpler world of laboratory animals [1]. We collaborate with behavioral neurobiologists who are interested in measuring and analyzing behavior across genotypes, in order to understand the link between

genes, brains and behavior. One of their most popular model organism is the fruit fly, Drosophila melanogaster; it is easy to care for, has a fast life cycle, and exhibits a wide range of behaviors despite having merely 10^5 neurons. Through this collaboration we have put together a large annotated dataset of fruit flies interacting spontaneously in controlled environments. This dataset allows us to study natural actions and develop insight into how to represent, segment and classify them. If our effort is successful, we can both advance the state of the art in human action analysis and provide biologists with tools for automatic labeling of actions, enabling them to do experiments at a scale which would otherwise be extremely expensive or impossible.

In this paper we describe an end-to-end approach for detecting the actions of fruit flies from video. The main contributions of our study are:

1. We consider two different action detection architectures: sliding window detectors and structured output detectors. By comparing five variants of the two architectures on our dataset, we find that sliding window detectors outperform the structured output detectors, in spite of being orders of magnitude faster.

2. We describe *bout features* that extract statistical patterns from frame-level features over an interval of time, and emphasize the similarities of bouts within an action class. Our experiments show that actions cannot be well detected using frame-level features alone, and that bout features improve performance by 28%.

3. We discuss pitfalls of measures commonly used for benchmarking action detection in continuous video and demonstrate which measures are most suitable, suggesting a protocol for comparing the performance of different algorithms.

4. We introduce Caltech Fly-vs-Fly Interactions (Fly-vs-Fly for short), a dataset containing 22 hours of fruit flies interacting spontaneously and sporadically. It comes with complete labeling of 10 actions, annotated by neurobiologists, and a second layer of annotations that can be used as a reference point for action detection performance. Along with the videos and annotations we publish a number of time-varying trajectory features, computed from the tracked pose (position, orientation, wing angles, etc.) of the flies. The dataset is available at `www.vision.caltech.edu/Video_Datasets/Fly-vs-Fly`.

2 Related work

Datasets – A large number of human action datasets have been published. KTH [2] and Weizmann [3] are early contributions that have been extensively used, but they are very small and consist of pre-segmented clips of acted actions. Hollywood 2 [4], Olympic Sports [5], HMDB51 [6], and UCF-101 [7], contain pre-segmented clips of natural actions, making them suitable for action classification, but not for detection and segmentation of actions, while UT-interactions [8] contains continuous social interactions that are acted. VIRAT [9] contains hours of continuous video of humans behaving naturally and intermittently, lending itself well to action detection research; however, the pose of the subjects cannot yet

be robustly tracked and the human motion that can be explored is limited; furthermore, VIRAT does not contain social actions. HumanEva [10], HDM05 [11], TUM Kitchen [12], CMU MMAC [13], and CAD-60/120 [14,15] are continuous and come with fully tracked skeletons which makes them useful for analyzing a range of human motions; however, these datasets are small, and their actions are acted. Table 1 compares details of the mentioned datasets.

The publicly available datasets of animal behavior video are Honeybee Dance [16], UCSD mice [17], Home-cage behaviors [18], and CRIM13 [1]. The latter two are suitable for action detection, containing long videos of spontaneous mouse actions, but both are parameterized with only the tracked centroid of the subject and spatial-temporal features. A large and well annotated dataset containing unsegmented, spontaneous, social actions, that includes tracking of articulated body motion has not yet been published. Our dataset aims to fill that place.

Dataset	Year	#Citations	Duration	#Actions*	Natural	Social	Continuous	Articulated pose
KTH	2004	1634	3 hours*	6	x	x	x	x
Weizmann	2005	986	5 minutes*	10	x	x	x	x
Hollywood 2(1)	2009	436(1327)	20 hours	12	✓	✓	x	x
Olympic Sports	2010	196	2 hours*	16	✓	x	x	x
UT Interactions	2010	41	20 minutes	6	x	✓	✓	x
HMDB51	2011	137	2 hours	51	✓	✓	x	x
VIRAT	2011	91	29 hours	23	✓	x	✓	x
UCF-101(50,11)	2012	40(57,477)	27 hours	101	✓	✓	x	x
HumanEva	2006	583	22 minutes	6	x	x	✓	✓
HDM05	2007	107	3 hours	70	x	x	✓	✓
TUM Kitchen	2009	98	1 hour*	13	x	x	✓	✓
CMU MMAC	2009	97	6 hours*	16**	x	x	✓	✓
CAD-60/120	2011/13	175/34	2 hours*	22**	x	x	✓	✓
UCSD mice	2005	1458	2 hours	5	✓	x	x	x
Honeybee	2008	61	3 minutes	3	✓	x	✓	x
Home-cage	2010	60	13 hours	8	✓	x	✓	x
CRIM13	2012	15	37 hours	12	✓	✓	✓	x
Fly-vs-Fly	2014	-	22 hours	10	✓	✓	✓	✓

*estimated upper limit, **sub-activities (actions/verbs), *excluding the null category

Table 1. Synoptic table of action datasets, with properties desired for detecting realistic social actions from articulated pose marked in green.

Action detection – A common approach to action detection is frame-by-frame classification, where each frame is classified based on features extracted from the frame itself, or from a time window around it: Dankert et al. detected actions of fruit flies using manually set thresholds on frame level features, along with nearest neighbor comparison [19]; Burgos et al. used boosting and auto-context on sliding windows for detecting actions between mice [1]; and Kabra et al. also use window based boosting for detecting actions of fruit flies in their interactive behavior annotation tool, JAABA [20]. More sophisticated approaches globally optimize over possible temporal segmentations, outputting structured sequences of actions: Jhuang et al. used an SVMHMM, described in [21], for detecting actions of single housed mice [18]; Hoai et al. used a multi-class SVM with

structured inference for segmenting the dance of the honeybee [22]; and Shi et al. used a discriminative semi-Markov model for segmenting human actions [23].

We implemented three variants of the above approaches, specifically comparing a sliding window SVM detector against two structured output SVM detectors, expecting the latter to improve frame-wise consistency and better capture structured actions. For reference, we compare our results with the methods described in [20] and [1] and with the performance of trained novice annotators.

3 Fly-vs-Fly

In collaboration with biologists we have collected a new dataset, Fly-vs-Fly, which contains a total of 22 hours (1.5m frames recorded at 200Hz and 2.2m frames at 30Hz) of 47 pairs of fruit flies interacting. The videos are organized into three subsets, each of which was collected for a different study:

Boy meets boy is designed to study the sequence of actions between two male flies, whose behaviors range from courtship to aggression. The flies are placed in a 4x5 cm^2 chamber with a food patch in its center and walls coated with Fluon, constraining the flies to walking on the floor [24]. It contains six 20 minute videos recorded at 200Hz with 12 pix/mm (24 pixels covering the 2mm fly body length).

Aggression contains videos of two hyper aggressive males [25] and is used to quantify the effect of genetic manipulation on their behavior. The flies are placed in a circular 16mm diameter chamber with uniform food surface [26]. It consists of ten 30 minute videos recorded at 30Hz with 8 pix/mm.

Courtship videos contain a female and a male, which in some cases are wild type and in the rest are so-called hyper courters [25]. This set of videos was used to study how genetic manipulation affects male courtship behavior. It consists of 31 videos recorded with the same chamber and video settings as Aggression.

The filming setups for these experiments are shown in Supplementary Figure 2.

3.1 Annotations

The entire dataset was annotated by, or under the supervision of, biologists, with 10 action classes that have been identified for the study of fruit fly interactions [27,28,19]: *wing threat*, *charge*, *lunge*, *hold*, and *tussle* (aggressive), *wing extension*, *circle*, *copulation attempt*, and *copulation* (courtship), and *touch* (neutral). Each action is described and visualized in Supplementary Figure 1.

Annotating a video involves finding all time intervals that contain an action of interest, also referred to as action *bouts*, and requires recording the start frame, end frame, and class label of each detected bout. The dataset is annotated such that actions can overlap, for instance tussling usually includes lunging, wing threat sometimes includes a charge, and wing extension and circling tend to overlap. Each action class takes up less than 2% of the total frames, apart from touch which takes up 7% of Boy meets boy and copulation which takes up 57% of the Courtship videos, and some classes have substantial intraclass variation, both in terms of duration and appearance. Figure 1 summarizes the dataset.

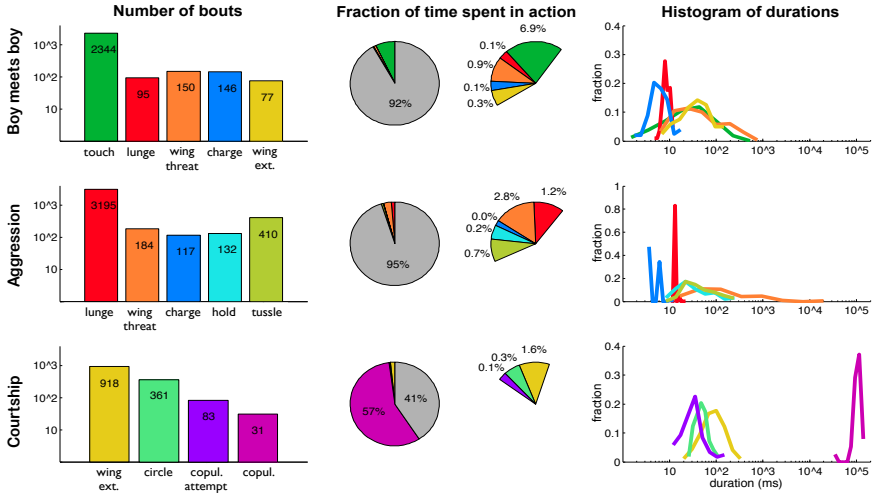


Fig. 1. Action statistics: *Left:* Number of bouts for each action. *Center:* Fraction of time a fly spends in each action, where the gray area represents the grab-bag category *other*. *Right:* Distribution of bout durations for each action class.

3.2 Feature representation

For action classification, data representation is half the challenge. An ideal classifier is invariant of any intra-action variation, but to train such a classifier a complex model or large amounts of training data may be needed. Alternatively, this invariance can be encoded into the features. Following prior art [19,29,20] we have implemented a tool that tracks individual flies and segments them into body, wing, and leg pixels, which are parameterized further by fitting an oriented ellipse to the body component and line segments to the wing components.

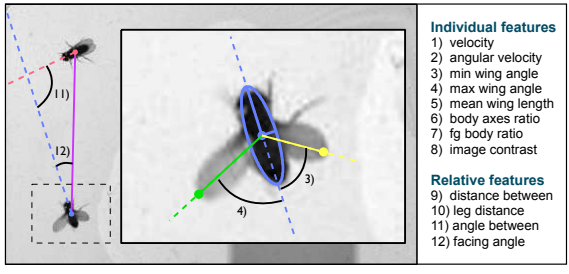


Fig. 2. Illustration of features derived from the tracked fly skeletons, which are invariant of absolute position and orientation of the fly and relate the pose of the fly to that of the other fly.

From the tracking output we derive a set of features that are designed to be invariant of the absolute position and orientation of a fly, and relate its pose to that of the other fly. The features (illustrated in Figure 2) can be split into two categories: individual features which include the fly's *velocity*, *angular velocity*, *min and max wing angles*, *mean wing length*, *body axis ratio*, *foreground-body ratio*, and *image contrast* in a window around the fly; and relative features

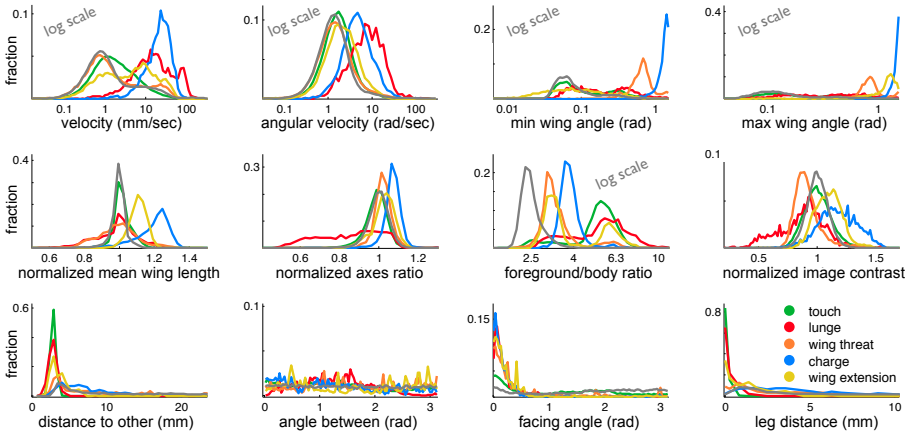


Fig. 3. Frame-wise feature distribution for the actions of the Boy meets boy sub-dataset, and the grab-bag action *other* shown in gray.

which relate one fly to the other with *distance between* their body centers, *leg distance* (shortest distance from its legs to the foreground of the other fly), *angle between*, and *facing angle*. Analysis of the feature distributions showed that the velocities, wing angles, and foreground-body ratio are better represented by their log values, becoming more normal distributed. Figure 3 shows the distribution of each feature, for all actions in the Boy meets boy sub-dataset, giving an idea of which features are important for which action. In addition, we take the first two time derivatives of each feature, resulting in a feature space of 36 *per-frame* features. The features are computed from the reference frame of each fly, yielding two asymmetrical feature vectors, and the actions we consider are always performed by a single fly. Hence, an hour of video effectively results in 2 hours of labeled data.

Our software for tracking flies and annotating their actions is available, along with documentation, at `www.vision.caltech.edu/Tools/FlyTracker`.

4 Action detection

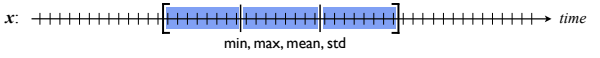
In this paper we focus on detection by exhaustive classification, in particular we compare two different architectures: *Sliding window detection* which refers to classifying fixed size windows that move frame-by-frame over a video sequence, and *structured output detection* which refers to detection by optimizing over all possible segmentations of a sequence into actions. Both schemes involve a training algorithm that learns an action classifier from n labeled sequences, $\{(x_i, y_i)\}_{i\in\{1,\ldots,n\}}$, and an inference algorithm that takes a new sequence x and predicts $y := \{y^j\} = \{(s^j, e^j, c^j)\}$, where y^j is the jth bout in the segmentation of x, s^j and e^j mark the start and end of the bout and c^j is its class label. We

treat the problem of detecting different actions as disjoint detection problems, mainly because the data that we are interested in has many overlapping actions. Before describing the detection architectures in detail we define *bout features* that aggregate per-frame features over an interval of frames, and are used in our implementation of both detection schemes.

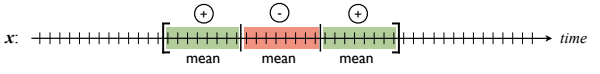
4.1 Bout features

We define a number of bout-level features that are designed to extract statistical patterns from an interval and emphasize the similarities of bouts within an action class, independent on bout duration. The following bout features, $\psi_k(x, t_{start}, t_{end})$, are functions of sequence x and interval $[t_{start}\ t_{end}]$:

Temporal region features capture statistics of frame-features over subintervals, and are meant to emphasize patterns within an action composed of r subactions. They can be expressed as: $\{\text{op}(x(t_{start}+(i-1)\delta t : t_{start}+i\ \delta t-1))\}_{i\in\{1,\ldots,r\}}$, where $\delta t = (t_{end} - t_{start} + 1)/(r-1)$, $r \geq 1$, and op $\in$ {min, max, mean, std}.



Harmonic features are meant to capture harmonic actions and can be expressed as: $\sum_{i=1}^{r}(-1)^i$ mean$(x(t_{start} + (i-1)\delta t : t_{start} + i\ \delta t - 1))$, where $\delta t = (t_{end} - t_{start} + 1)/(r-1)$ and $r \geq 1$.



Boundary features emphasize the change in features at the start and end of a bout, and help with locating boundaries. For a fixed δt, they can be expressed as: mean$(x(t_{start/end} : t_{start/end} + \delta t))$ - mean$(x(t_{start/end} - \delta t : t_{start/end}))$.

Bout change features capture the difference in features between the beginning and end of a bout, expressed as: $x(t_{end}) - x(t_{start})$.

Global difference features compare the mean of a bout to global statistics of data, expressed as: mean$(x(t_{start} : t_{end}))$−op(x), where op $\in$ {min, max, mean}.

Histogram features represent the normalized distribution of each feature within the bout, expressed as: hist$(x(t_{start} : t_{end})$, bins), where bins are extracted from the training data, such that an equal number of frames falls into each bin.

In our experiments we use three temporal region splits, $r \in \{1, 2, 3\}$, and set the number of histogram bins to be 2^3, resulting in a total of $K = 48$ bout features. With K bout features applied to each of the N per-frame features, the feature representation for a bout ends up being a $D = KN$ dimensional vector, ψ.

4.2 Sliding window framework

Our sliding window implementation has 4 main components: a *training* algorithm that learns a classifier from labeled sequences, a *classifier* module, an *inference* algorithm that predicts labels for unseen sequences, and a *post processing* module that promotes continuity in the prediction labels.

Training: The training algorithm converts each sequence of input labels, $\{y_i\} = \{(s^j, e^j, c^j)_i\}$, to indicator vectors, $\{z_i\}$, that specify whether a frame belongs to an action or not. It extracts normalized bout features over fixed sized windows surrounding each frame of all sequences, obtaining high dimensional data points whose labels are the same as those of the frames around which the windows were placed. With this data it trains a classifier using a bootstrapping scheme that overcomes memory limitations that may be associated with large data, and allows us to indirectly optimize with respect to performance measures that involve the number of predicted positives. At each iteration it learns a classifier from a subset of the data, using a learning algorithm suitable for the classifier type, applies it to all of the data and adds misclassified samples to the training set - repeating until the desired performance measure stops increasing.

Inference: The inference algorithm extracts bout features from a window around each frame in x, normalized with statistics from the training data, and classifies each window using the classifier obtained from the training step. The resulting sequence of scores is thresholded to obtain an action indicator vector, $\hat{z}$, whose connected components make up the predicted label sequence, $\hat{y}$, assigning each bout the label, start frame, and end frame of its component.

Post processing: Classifying a sequence frame-by-frame often results in noisy labels, that is, within a bout of an action a few frames may be just below a threshold and therefore split the bout into multiple bouts. To account for this we fit an HMM to the scores to achieve smoother transitions: we convert scores to posterior probabilities, $P(x(t)|z(t) = 1) := 1/(1+\exp(-\text{score}(t)))$, $P(x(t)|z(t) = 0) := 1-P((t)|z(t) = 1)$, compute prior probabilities, $P(z(1) = c)$, and transition matrix, $P(z(t+1) = c_i|z(t) = c_j)$, from the training data, and run the Viterbi algorithm [30] to find the most probable frame-wise sequence of actions.

Classifier: The classifier module consists of a binary classifier and its associated learning algorithm. For comparison with our structured SVM implementation, we choose to use a linear SVM classifier, learnt using the LIBLINEAR implementation described in [31]. The classifier can be substituted by any other binary classifier, such as boosting, regression, neural net, or a generative model.

This approach can be converted to a frame-based detector, by simply substituting the bout features around a frame with its per-frame features.

4.3 Structured output framework

Structured output detectors differ from sliding windows in that they optimize over all possible segmentations of a sequence into action intervals, finding the best start and end frame of all bouts, allowing for varying sized intervals.

Structured SVM We extend the structured SVM [32] to train a model that can be utilized for segmenting sequences into actions, by defining a *score function*, $f(x, y)$, which assigns high scores to good segmentations, and a *loss function*, $\mathcal{L}(y, \hat{y})$, which penalizes poor segmentations.

Training: The goal is to learn the weights w of a score function from a given training set, such that for each training example the score of the true segmentation y_i is higher than the score of any other segmentation y by at least $\mathcal{L}(y_i, y)$. If these constraints cannot be satisfied, a hinge loss is suffered. To learn these weights we use the primal structured SVM objective:

$$w^* \leftarrow \arg\min_w \|w\|^2 + C\frac{1}{n}\sum_{i=1}^{n}\left(\max_y \left[f(x_i, y) + \mathcal{L}(y_i, y)\right] - f(x_i, y_i)\right),$$

which we minimize using a cutting plane algorithm [32] that iteratively finds the most violated constraint: $\hat{y} = \arg\max_y \left[f(x_i, y) + \mathcal{L}(y_i, y)\right]$. Searching over all possible segmentations is intractable, but since our score- and loss functions are linear in the bouts of y, dynamic programming [33] can solve for the optimal y.

Score function: We define a score function $f(x, y)$, which measures how well y segments x into actions and can be represented as the sum of a bout score, unary cost, transition cost, and duration cost, over all bouts in the segmentation:

$$f(x, y) = \sum_{(s^j, e^j, c^j) \in y} [w_{c^j} \cdot \psi(x, s^j, e^j) - \tau(c^j) - \lambda(c^{j-1}, (c^j)) - \gamma(c^j, s^j, e^j)].$$

Weights w_{c^j} are used to calculate the score for a bout of class c^j, $\tau(c^j)$ is the cost of detecting a bout of class c^j, $\lambda(c^{j-1}, c^j)$ is the cost of moving from action c^{j-1} to c^j, and $\gamma(c^j, s^j, e^j)$ is the cost of spending $e^j - s^j + 1$ frames in action c^j. These terms are inspired by a hidden semi Markov model, comparable to [23].

Loss function: The loss function penalizes discrepancies between ground truth segmentation y and a predicted segmentation $\hat{y}$, and should be constructed such that a small loss indicates satisfactory results. We define it as:

$$\mathcal{L}(y, \hat{y}) = \sum_{(s,e,c) \in y} \frac{\ell^c_{fn}}{e - s + 1}\left(\bigcap_{\hat{y}, \hat{c} \neq c}(s, e)\right) + \sum_{(\hat{s}, \hat{e}, \hat{c}) \in \hat{y}} \frac{\ell^c_{fp}}{\hat{e} - \hat{s} + 1}\left(\bigcap_{y, c \neq \hat{c}}(\hat{s}, \hat{e})\right),$$

where $\bigcap_{\hat{y}, \hat{c} \neq c}(b, e)$ is the number of frames in $\hat{y}$ intersecting with $[b\ e]$ with different action class $\hat{c} \neq c$, ℓ^c_{fn} is the cost for missing a bout of class c, and $\ell^{\hat{c}}_{fp}$ is the cost for incorrectly detecting a bout of class $\hat{c}$. This loss function softly penalizes predictions where the start or end of the bout is slightly incorrect. On the other hand, since the loss is normalized by the bout duration, it effectively counts the number of incorrectly predicted bouts and, unlike a per-frame loss, long actions are not deemed to be more important than short ones.

Inference: Given a score function, $f(x, y)$, and an input x, the optimal segmentation can be found by solving $\hat{y} = \arg\max_y f(x, y)$. Again, similarly to the learning phase, searching over all possible segmentations is intractable but we can solve for y using dynamic programming.

Semi-structured SVM This approach is a hybrid of the sliding window SVM and the structured SVM; its inference algorithm optimizes over possible segmentations of a sequence, using dynamic programming, but the classifiers are trained using a linear SVM on fixed bouts from the training set, similar to [22].

Training: We extract bout features from the positive bouts, $\{(s^{j_i}, e^{j_i})\}_i$, for each sequence x_i in the training set, and from randomly sampled negative bouts. We consider a bout as negative if its intersection with a positive bout is less than half of their union, so that large intervals containing positive bouts and small intervals that are parts of a positive bout are still considered as negatives. Inference involves considering all possible intervals of any duration as potential action bouts, however training on all such possible intervals would be intractable. Instead, we generate a limited number of randomly sampled negatives and use a bootstrapping training process that gradually adds useful negative samples. At each iteration we train a classifier on the current training data, run inference with the learnt classifier, and add falsely detected positives to the set of negative training samples - repeating until no new false positives are detected.

Inference: Here the goal is the same as in the structured SVM approach, to find the optimal segmentation of a new input sequence x, $\hat{y} = \arg\max_y f(x, y)$, but with a simpler score function: $f(x, y) = \sum_{(s^j, e^j, c^j) \in y} w_{c^j} \cdot \psi(x, s^j, e^j)$. Again, we solve this using dynamic programming. We speed up the inference by setting upper limits on the duration of an action, which we obtain from the training set.

5 Experiments and Analysis

5.1 Measures

When evaluating algorithms, the performance measure should favor desirable outcomes; in the case of action detection for behavior analysis it is important that there are few false hits and misses compared to the number of true action instances, which becomes difficult the more sparsely actions occur in the data. We have generated a synthetic ground truth sequence with 5 sporadic action classes, and two different prediction sequences, to demonstrate the difference between three common measures: a *confusion matrix*, *ROC* curves, and *precision-recall* curves. This comparison shows that precision-recall most effectively emphasizes the large performance discrepancy between the two predictions (see Figure 4).

Precision-recall curves, used for measuring detection performance for a single class, plot *precision* against *recall*, favoring minimum number of false positives and false negatives with respect to the number of positives. **ROC** curves are similar but instead of precision they plot the *false positive rate*, which places little emphasis on false positives when negatives take up vast majority of the frames. A **confusion matrix**, used in multi-class classification, is a square matrix whose entry (i, j) represents the fraction of ground truth instances of class i that are predicted as class j, and is commonly summarized by its diagonal mean. However, its diagonal effectively measures the recall of each class and fails to emphasize false positive instances which get absorbed into the grab-bag class

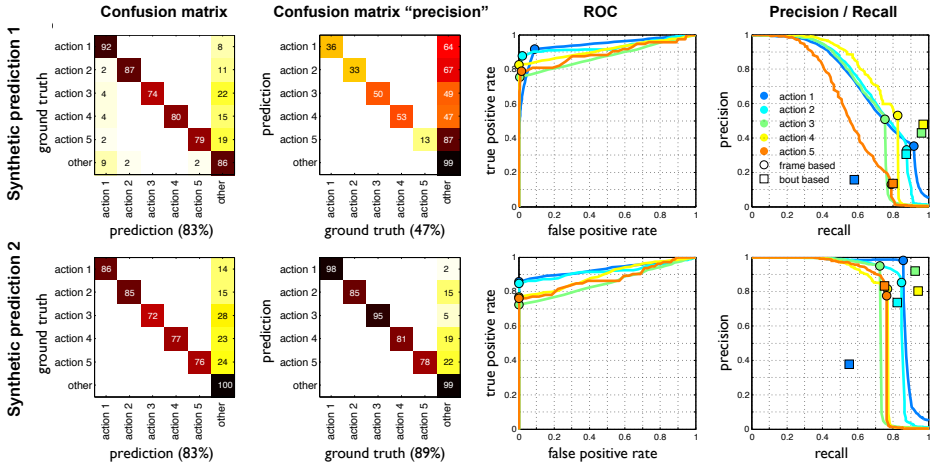


Fig. 4. Confusion matrices and ROC are unreliable diagnostics for assessing experimental results. Each row shows the result of a different synthetic experiment. The confusion matrices (first column) and the ROC (third column) suggest that both experiments yield the same result and hide the large difference in the number of false detections. This fact is revealed by the "precision" confusion matrix (second column) and by the precision-recall curve (fourth column). The last column also shows how bout-wise and frame-wise measurements can differ.

other. To account for this, one must also look at the 'dual' confusion matrix, where entry (i, j) represents the fraction of predicted instances of class i that belong to class j according to ground truth, in which case the diagonal effectively measures the precision of each class. We conclude that "precision" and "recall" confusion matrices are good measures for multi-class detection problems, where classes are mutually exclusive, but for experiments such as ours, where classes overlap and false positives are expensive, precision-recall curves are the best performance measurement tools.

For behavior analysis, correctly counting the number of action instances is equally important as correctly measuring the duration spent in an action, hence we must also measure the **bout-wise performance**. To do that we use an overlap criteria, that deems a ground truth bout (s_g, e_g, b) and predicted bout (s_p, e_p, b) to match only if, $\frac{min(e_g,e_p)-max(s_g,s_p)}{max(e_g,e_p)-min(s_g,s_p)} > threshold$. If multiple bouts fit that criteria, we match the one with the highest ratio. Figure 4 shows that there can be large discrepancies between frame-wise and bout-wise performance. This is the case when predicted bouts are more fragmented than ground truth bouts, or when bouts are consistently predicted to be shorter than, or offset from, the ground truth (see more detail in Supplementary Figure 3).

In order to rank different methods we combine precision and recall into a single value using the *F-score*, defined as $\mathrm{F}_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$, which for $\beta = 1$ represents the harmonic mean that favors balanced precision-recall combinations. To further combine bout-wise and frame-wise performance we define the *F*-score* as the harmonic mean of F1-frame and F1-bout.

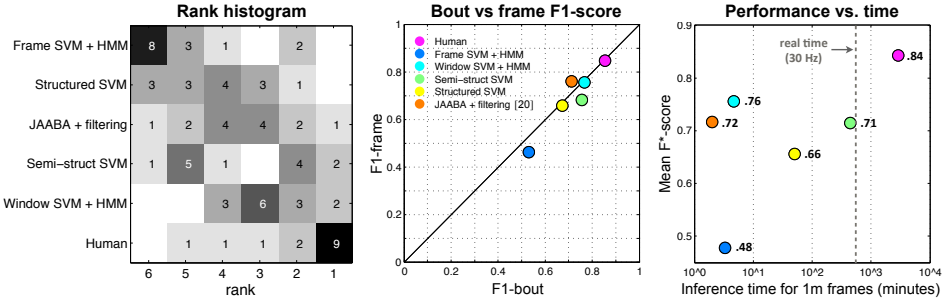


Fig. 5. Method comparison on the Fly-vs-Fly dataset. *Left*: Histogram of method ranks over all actions, based on their F*-score, ordered by mean rank. *Center*: Comparison of F1-scores of each method, averaged over all actions. *Right*: F*-score of each method as a function of inference time.

5.2 Method comparison

Here we explore how a window based SVM compares to structured, and semi-structured SVMs, which we find very interesting as they all make use of linear classifiers and the same bout features, but differ in their training and inference procedures. In addition, we compare them with a frame based SVM to get a sense for how much bout features contribute to performance, and to JAABAs back-end [20], another window based detector, for comparison with methods currently deployed in action detection systems.

Each method's free parameters were optimized using a subset of the training data for validation, and we found that HMM post processing improved the mean F*-score of the window- and frame based SVMs by 11% and 3% respectively. For comparison with JAABA we trained detectors by substituting their boosting classifier implementation into the learning and inference modules of our window based framework. JAABA as presented in [20] does not include post processing, but here we apply a box filter suggested on their project website [34] for a fair bout-wise performance comparison, improving its mean F*-score by 6%.

To measure the performance of our action detectors, we computed bout- and frame-wise precision, recall and F1-scores, and the F*-score which can be used to rank the different methods. These measures, broken down for each behavior in Supplementary Figures 6-8, show considerable variation in method rank depending on the action. Here we summarize the results in a detector rank histogram (Figure 5), which shows the number of times each detector achieved each rank and orders methods according to their mean rank. For a finer resolution view of how the methods line up we show the mean F1-scores, averaged over all actions, and the mean F*-score as a function of time it takes to run the detector on 1 million frames. This view mostly preserves the rank observed in the rank histogram, but it also shows that most methods cluster around 70% performance, apart from humans at 84% and frame based SVM at 48%. In addition, it shows that the window based methods perform slightly better than the structured output counterparts, in spite of being orders of magnitude faster.

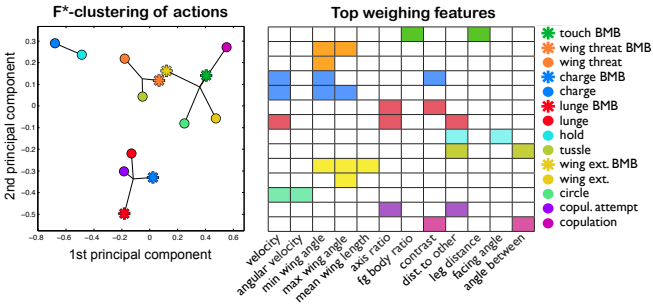


Fig. 6. *Left*: clustering of actions based on F*-score of all methods. *Right*: top weighted features determined by the trained SVM detectors.

These summary measures abstract away information about performance patterns between actions that may give insights into the different types of actions. To explore that, we cluster actions based on their F*-score for each method, by applying principal component analysis to the F*-matrix, and fitting k-means to the dimension-reduced matrix, splitting actions into 4 groups. Figure 6 shows that this clustering groups together lunge, charge, and copulation attempt, which all share the characteristic of being short and concise but poorly captured by the frame based detector, and, as one might expect, it groups actions (wing threat and wing extension) from different sub-datasets together. From the learnt detectors of the three different SVM approaches we found that the window based detector made most use of the bout statistics and histogram features, while structured ones used boundary dependent features to a similar extent, and that the top per-frame features used by all methods are those listed in Figure 6, showing that each feature is the highest contributing feature to at least one action.

5.3 Performance on CRIM13

Finally, to give a better idea of where these methods place within state of the art, we test the top ranked detector on the most recently published animal dataset, CRIM13, and compare our results with those presented in [1]. Actions in CRIM13 are non-overlapping, and the detection problem is treated as multi-class. To make a similar comparison we covert our binary action detectors to a single multi-class detector by fitting them to an HMM with 13 states. By shifting output scores of individual binary classifiers, before converting them to posterior probabilities, we can trade off the performance of different classes. We obtain the optimal shift-parameters by greedily maximizing w.r.t. the diagonal mean of the "recall" confusion matrix, to match the measure used in [1], and since we are interested in high precision-recall combination, we also optimize w.r.t. the mean F1-score of the "recall" and "precision" matrix diagonals. Figure 7 shows the confusion matrices produced for each of our optimization criteria, and Figure 8 shows our results compared with those presented in [1]. We ran our algorithm only on tracking features (TF) provided with the CRIM13 dataset, obtaining performance just above the top results reported in [1], which includes spatial-temporal features (STF), and 3.2% higher than their performance on tracking features alone. Optimizing w.r.t. F1-score results in approximately 6% F1-performance gain over the "recall" optimization.

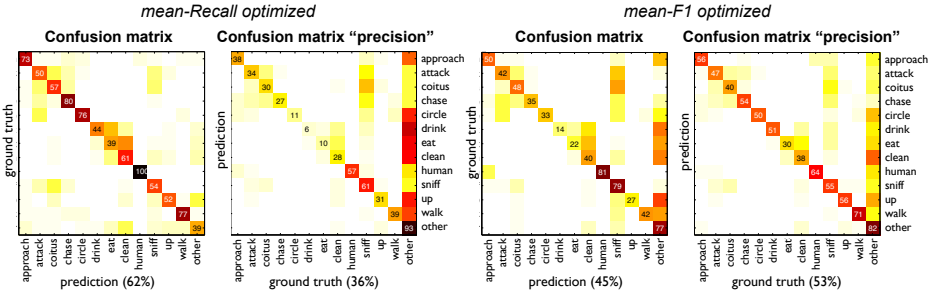


Fig. 7. Confusion matrices for Window SVM + HMM on the CRIM13 test dataset. *Left*: performance optimized w.r.t. the diagonal mean of confusion matrix. *Right*: performance optimized w.r.t. "recall" and "precision" confusion matrices.

Method	mean recall	mean F1
Boosting (TF) + Autocontext [1]	58.30%	-
Boosting (TF + STF) + Autocontext [1]	61.20%	-
Window SVM+HMM (recall shift)	**61.66%**	40.76%
Window SVM+HMM (F1 shift)	45.42%	47.22%

Fig. 8. Comparison of the window based SVM to the methods used in [1], showing performance on the CRIM13 test dataset.

6 Conclusions

We collected a large dataset of fruit fly videos that, with its natural and sporadic interactions and rich set of articulated pose features, fills a gap in existing datasets. We developed a framework for comparing action detection performance, showing that precision and recall are the best suited measures for evaluating detection algorithms, and that results should be reported both in terms of bout- and frame-wise performance. Using these measures, we showed that bout features highly improve performance upon frame-level features. We compared sliding window classifiers to the more sophisticated structured output detectors, and found that window based classifiers outperformed their structured counterparts, despite having much lower time complexity. This was surprising to us, as the structured output methods allow for elastic sized windows which should better capture structure within bouts. A caveat is that the more complex actions in our dataset have low duration variation, therefore fixed sized window classifiers with good bout features may suffice. Our results also show (Supplementary Figures 6-8) that the structured output methods suffer from over-segmenting long bouts of actions that do not have much structure, which leads to a lower bout-wise performance. We believe this may be overcome by incorporating higher order Markov terms in the score function, and will explore that in future work. In our experiments, the top performing algorithm, a window based SVM + HMM, reached a 76% F*-score on Fly-vs-Fly, compared to 84% achieved by humans, and matches the performance of the best published method on CRIM13.

Acknowledgements: This work was supported by the ONR MURI N00014-10-1-0933 and the Gordon and Betty Moore Foundation.

References

1. Burgos-Artizzu, X.P., Dollár, P., Lin, D., Anderson, D.J., Perona, P.: Social behavior recognition in continuous video. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. pp. 1322–1329. IEEE (2012)
2. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: a local svm approach. In: Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on. vol. 3, pp. 32–36. IEEE (2004)
3. Gorelick, L., Blank, M., Shechtman, E., Irani, M., Basri, R.: Actions as space-time shapes. Transactions on Pattern Analysis and Machine Intelligence 29(12), 2247–2253 (December 2007)
4. Marszałek, M., Laptev, I., Schmid, C.: Actions in context. In: IEEE Conference on Computer Vision & Pattern Recognition (2009)
5. Niebles, J.C., Chen, C.W., Fei-Fei, L.: Modeling temporal structure of decomposable motion segments for activity classification. In: Computer Vision–ECCV 2010, pp. 392–405. Springer (2010)
6. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: Proceedings of the International Conference on Computer Vision (ICCV) (2011)
7. Soomro, K., Zamir, A.R., Shah, M.: Ucf101: A dataset of 101 human actions classes from videos in the wild. arXiv preprint arXiv:1212.0402 (2012)
8. Ryoo, M., Aggarwal, J.: Ut-interaction dataset, icpr contest on semantic description of human activities (sdha) (2010)
9. Oh, S., Hoogs, A., Perera, A., Cuntoor, N., Chen, C.C., Lee, J.T., Mukherjee, S., Aggarwal, J., Lee, H., Davis, L., et al.: A large-scale benchmark dataset for event recognition in surveillance video. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 3153–3160. IEEE (2011)
10. Sigal, L., Black, M.J.: Humaneva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Brown Univertsity TR 120 (2006)
11. Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., Weber, A.: Documentation mocap database hdm05 (2007)
12. Tenorth, M., Bandouch, J., Beetz, M.: The tum kitchen data set of everyday manipulation activities for motion tracking and action recognition. In: Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on. pp. 1089–1096. IEEE (2009)
13. De la Torre, F., Hodgins, J., Montano, J., Valcarcel, S., Forcada, R., Macey, J.: Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. Tech. rep., Citeseer (2009)
14. Sung, J., Ponce, C., Selman, B., Saxena, A.: Unstructured human activity detection from rgbd images. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on. pp. 842–849. IEEE (2012)
15. Koppula, H.S., Gupta, R., Saxena, A.: Learning human activities and object affordances from rgb-d videos. arXiv preprint arXiv:1210.1207 (2012)
16. Oh, S.M., Rehg, J.M., Balch, T., Dellaert, F.: Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. International Journal of Computer Vision 77(1-3), 103–124 (2008)
17. Dollár, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: VS-PETS (October 2005)
18. Jhuang, H., Garrote, E., Yu, X., Khilnani, V., Poggio, T., Steele, A.D., Serre, T.: Automated home-cage behavioural phenotyping of mice. Nature communications 1, 68 (2010)

19. Dankert, H., Wang, L., Hoopfer, E.D., Anderson, D.J., Perona, P.: Automated monitoring and analysis of social behavior in drosophila. Nature methods 6(4), 297–303 (2009)
20. Kabra, M., Robie, A.A., Rivera-Alba, M., Branson, S., Branson, K.: Jaaba: interactive machine learning for automatic annotation of animal behavior. nature methods (2012)
21. Altun, Y., Tsochantaridis, I., Hofmann, T., et al.: Hidden markov support vector machines. In: ICML. vol. 3, pp. 3–10 (2003)
22. Hoai, M., Lan, Z.Z., De la Torre, F.: Joint segmentation and classification of human actions in video. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 3265–3272. IEEE (2011)
23. Shi, Q., Cheng, L., Wang, L., Smola, A.: Human action segmentation and recognition using discriminative semi-markov models. International journal of computer vision 93(1), 22–32 (2011)
24. Hoyer, S.C., Eckart, A., Herrel, A., Zars, T., Fischer, S.A., Hardie, S.L., Heisenberg, M.: Octopamine in male aggression of¡ i¿ drosophila¡/i¿. Current Biology 18(3), 159–167 (2008)
25. Hoopfer, E.D., Anderson, D.J.: Unpublished
26. Asahina, K., Watanabe, K., Duistermars, B.J., Hoopfer, E., González, C.R., Eyjólfsdóttir, E.A., Perona, P., Anderson, D.J.: Tachykinin-expressing neurons control male-specific aggressive arousal in¡ i¿ drosophila¡/i¿. Cell 156(1), 221–235 (2014)
27. Chen, S., Lee, A.Y., Bowens, N.M., Huber, R., Kravitz, E.A.: Fighting fruit flies: a model system for the study of aggression. Proceedings of the National Academy of Sciences 99(8), 5664–5668 (2002)
28. Hall, J.C.: The mating of a fly. Science 264(5166), 1702–1714 (1994)
29. Branson, K., Robie, A.A., Bender, J., Perona, P., Dickinson, M.H.: High-throughput ethomics in large groups of drosophila. Nature methods 6(6), 451–457 (2009)
30. Viterbi, A.: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. Information Theory, IEEE Transactions on 13(2), 260–269 (1967)
31. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: A library for large linear classification. The Journal of Machine Learning Research 9, 1871–1874 (2008)
32. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. In: Journal of Machine Learning Research. pp. 1453–1484 (2005)
33. Bellman, R.: Dynamic programming and lagrange multipliers. Proceedings of the National Academy of Sciences of the United States of America 42(10), 767 (1956)
34. http://jaaba.sourceforge.net