A Maximum-Likelihood Approach to Visual Event Classification

Jeffrey Mark Siskind[⋆] and Quaid Morris

Department of Computer Science, University of Toronto, Toronto Ontario M5S 1A4 CANADA

Abstract. This paper presents a novel framework, based on maximum likelihood, for training models to recognise simple spatial-motion events, such as those described by the verbs *pick up*, *put down*, *push*, *pull*, *drop*, and *throw*, and classifying novel observations into previously trained classes. The model that we employ does not presuppose prior recognition or tracking of 3D object pose, shape, or identity. We describe our general framework for using maximum-likelihood techniques for visual event classification, the details of the generative model that we use to characterise observations as instances of event types, and the implemented computational techniques used to support training and classification for this generative model. We conclude by illustrating the operation of our implementation on a small example.

1 Introduction

People can describe what they see. Not only can they describe the objects that they see, they can also describe the events in which those objects participate. So if you were to see a person pick up a pen, you could describe that event by saying *The person picked up the pen*. In doing so you classify the two participant objects as a person and a pen respectively. You also classify the observed event as a picking-up event. Almost all recognition work in machine vision has focussed on object classification. In contrast, this paper is an attempt to address the problem of event classification.

While we are not the first to address this problem, our work differs from prior approaches (Badler, 1975; Nagel, 1977; Tsuji, Morizono, & Kuroda, 1977; Okada, 1979; O'Rourke & Badler, 1980; Rashid, 1980; Tsotsos, Mylopoulos, Covvey, & Zucker, 1980; Abe, Soga, & Tsuji, 1981; Marburger, Neumann, & Novak, 1981; Waltz, 1981; Marr & Vaina, 1982; Neumann & Novak, 1983; Thibadeau, 1986; Yamamoto, Ohya, & Ishii, 1992; Pinhanez & Bobick, 1995) in a number of important ways. First, we apply our methods to camera input, in contrast to synthetic input. Second, we group naturally occurring events into classes that correspond to pre-theoretic notions described by simple spatial-motion verbs like *pick up*, *put down*, *push*, *pull*, *drop*, *throw*, and so forth. While people may be able to perceive many other kinds of differences between two motion sequences,

[⋆] Current address: Department of Electrical Engineering, Technion, Haifa 32000, ISRAEL we are only interested in detecting those differences that can be described using ordinary verbs. Finally, we use an approach based on maximum likelihood. We determine the parameters of a general model empirically from training data instead of formulating detailed logical and geometric descriptions of event classes by hand.

The vision community has done much prior work on processing image sequences, particularly sequences involving motion. Examples of such work include optical flow, 2D and 3D object tracking, and shape from motion, to name a few. While our work bears superficial resemblance to such work, we wish to stress that we are concerned with a fundamentally different problem that is orthogonal to the ones addressed by such prior work. For example, we are interested in event classification and not 3D tracking. For us, techniques like 3D tracking are relevant only in so far as they might facilitate—or be facilitated by—event classification. In fact, the techniques that we describe in this paper do not perform detailed shape recovery, object classification, or 3D pose estimation.

Siskind (1992, 1995) proposed a technique for classifying events by recovering changing support, contact, and attachment relations between participant objects using a kinematic simulator driven from the output of 3D tracking. A tacit assumption behind this prior work was that object recognition is a prerequisite for event recognition. While attempting to implement the aforementioned techniques, we conducted a simple experiment that calls into question the validity of this assumption.

We took several movies of simple spatial-motion events in ordinary desk-top environments. These events included picking-up, putting-down, pushing, and pulling boxes, dropping erasers, and various collisions between objects. These movies were filmed using an off-the-shell SunVideo system recording image sequences with a resolution of 320×240 at 30 frames-per-second compressed using Sun's CellB format. We then applied an edge detector and line finder to each of the images in each movie and animated the resulting output. One original movie and the edge-detected images that correspond to that movie are shown in figure 1.

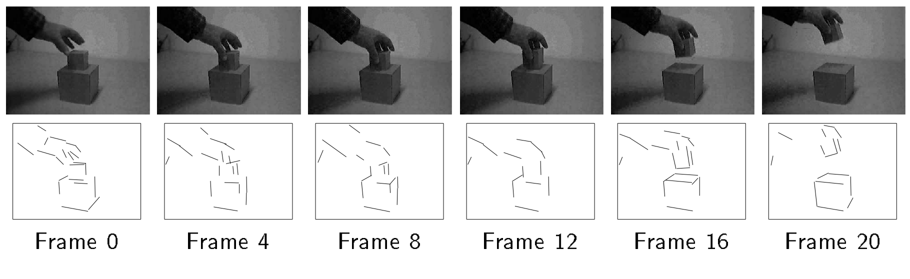

Fig. 1. Several frames from a movie depicting a *pick up* event along with the result of applying an edge detector and line finder to that movie.

The results of this experiment are striking. Edge detection and line finding collectively reduce the amount of information in the movie from 76,800 bytes per frame to roughly 25 line segments, or 100 bytes, per frame. This corresponds to a factor-of-768 lossy data compression. When watching such edge-detected output, either as isolated frames or as animated line movies, humans cannot reliably recognise the objects that appear in the images. Yet despite such lossy data compression, humans can still reliably recognise the depicted events when viewing an animation of the line drawings.

In retrospect, these results are not surprising. They are very much in-line with the point-light studies of animate-body motion performed by Johansson (1973) and others. They do, however, call into question the assumption of prior work, namely that event recognition presupposes object recognition. The information required to classify objects and recover their 3D pose over time is simply not available in the animated line drawings that we constructed. Yet even without such information, event recognition is a very robust process.

This leads us to conjecture, instead, that human event perception does not presuppose object recognition. We conjecture that event recognition is performed by a visual pathway that is separate from object recognition. Furthermore, we conjecture that this pathway requires far lower information bandwidth than object recognition. If this is true, then it may be the case that event recognition is an *easier* problem than object recognition and more amenable, in the short term, to synthetic engineered implementations. The rest of this paper offers precisely that: one possible framework for building an event-recognition engine.

2 The Framework

Linguistic evidence indicates that humans characterise events in terms of characteristic changes in the properties of, and relations between, objects that participate in those events. For example, a *pick up* event typically consists of a sequence of two subevents. During the first subevent, the hand of the *agent* moves toward the *patient*,[2] the object being picked up, while the patient rests on the *source* object. The agent then comes into contact with, and grasps, the patient. During the second subevent, the agent moves together with the patient away from the source, while supporting the patient. Similarly, a *throw* event typically consists of a sequence of two subevents. During the first subevent, the patient moves with the agent while the agent grasps, supports, and applies force to the patient. This subevent ends when the agent releases the patient. During the second subevent, the patient continues in unsupported motion after leaving the patient's hand in a trajectory that results, in part, from the force applied by the agent during the first subevent. The types and properties of the participant objects have little importance in defining these event types. Any agent can throw any patient or pick up any patient from any source. Objects simply fill *roles* of an event type. While some event types are characterised by the types

[2] The term 'patient' is being used here to denote the object affected by an action.

and potentially changing properties of participant objects, events described by simple spatial-motion verbs are largely characterised by the changing spatial and force-dynamic relations between objects. This paper focuses solely on such event types. Our long-term goal is to characterise and recognise simple spatial-motion events by recovering changing force-dynamic relations in addition to changing spatial relations. Mann, Jepson, and Siskind (1996) present some work along these lines. This paper, however, describes techniques for event recognition that are based solely on modelling the characteristic motion profiles, changing over time, of objects that participate in different simple spatial-motion events.

We partition the event recognition task into two independent subtasks. The lower-level task performs 2D object tracking, taking image sequences as input and producing as output a stream of 2D position, orientation, shape, and size values for each participant object in the observed event. This 2D pose information takes the form of a set of parameters for ellipses that abstractly characterise the position, orientation, shape, and size of the participant objects. This lower-level tracking is done without any constraint from event models. The upper-level task takes as input the 2D pose stream produced by lower-level processing, without any image information, and classifies such a pose stream as an instance of a given event type. We use a maximum-likelihood approach to perform the upper-level task. In this approach we use a supervised learning strategy to train a model for each event class from a set of example events from that class. We then classify novel event observations as the class whose model best fits the new observation.

3 Tracking

Our tracker uses a mixture of colour-based and motion-based techniques. Colour-based techniques are used to track objects with uniform distinctive colour, such as the blocks, even though such objects might not be in motion, or might be in motion for only part of the event. Motion-based techniques are used to track moving objects, such as the hand of the agent, even though the colour of such objects might not be uniform or distinctive and thus would not be detected by the colour-based techniques.

Our tracker operates on a frame-by-frame basis, tracking coloured objects and moving objects independently. To track coloured objects it first determines a set of ‘coloured pixels’ in each frame. Pixels are considered to be coloured if their saturation and value are above specified thresholds. Figure 2(b) shows the coloured pixels derived from the input image in figure 2(a). Such pixels are then classified into bins using a histogram clusterer based on hue.

After finding coloured regions in each frame, our tracker then finds moving regions. To do so it determines a set of ‘moving pixels’ for each frame by thresholding the absolute value of the difference between the grey scale values of corresponding pixels in adjacent frames. Figure 2(d) shows the set of moving pixels recovered by this technique for the image in figure 2(a).

Each hue cluster might, however, be spread over noncontiguous regions of the image. There might also be multiple moving objects, so the set of moving pixels

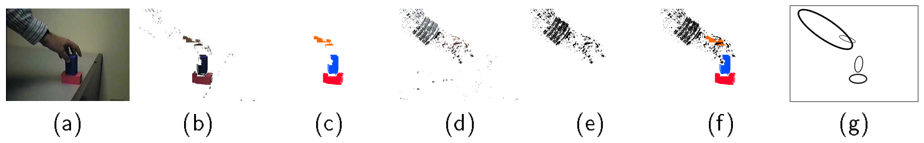
Fig. 2. The processing stages of our tracker. (a) shows an input image. (b) shows the coloured pixels. (c) shows the output of the region grower on (b). (d) shows the moving pixels. (e) shows the output of the region grower on (d). (f) shows the combination of (c) and (e). (g) shows the ellipses that are fit to the regions from (f).

might also be spread over noncontiguous regions. We apply a proximity clustering technique to divide each hue cluster and the set of moving pixels into contiguous subclusters. We employ a simple region-growing algorithm that groups pixels into equivalence classes. Two pixels are placed in the same equivalence class when the Euclidean distance between the $\langle x, y \rangle$ coordinates of those two pixels is less than a specified threshold. We then discard equivalence classes that have fewer than a specified number of pixels. This eliminates small spurious regions, such as those that appear in figures 2(b) and 2(d). Figure 2(c) shows the result of applying our region-grower to the hue clusters in the image in figure 2(b). Figure 2(e) shows the result of applying our region-grower to the image in figure 2(d) while figure 2(f) shows the combined colour-based and motion-based output of our tracker. At this point, a movie is represented as a set of regions for each frame, where each region is a set of pixels.

We now abstract each region in each frame as an ellipse. To do so, we compute the mean and covariance matrix of the two-dimensional $\langle x, y \rangle$ coordinate values of the pixels in a given region. We take the ellipse for that region to be centered at the mean and to follow a contour one standard deviation out from the mean. Thus the orientation of the major axis of the ellipse is along the primary eigenvector of the covariance matrix and the lengths of the major and minor axes of the ellipse are given by the eigenvalues of the covariance matrix. Figure 2(g) shows the ellipses generated for the regions in figure 2(f). All subsequent processing ignores the underlying image pixel data and uses only the derived ellipse parameters.

The operation of fitting ellipses to image data is done independently for each frame in the movie. Such independent processing suffers from two limitations. First, it does not recover the intra-frame correspondence between ellipses. We require this *internal correspondence* in order to track object position over time. Second, different frames can contain different numbers of ellipses. There may be situations, as is the case in figure 2(g), where the tracker produces spurious ellipses that do not correspond to objects that participate in events in the movie. There may also be situations where the tracker fails to produce an ellipse to represent an object that does participate in an event. Such drop outs can happen for a variety of reasons, such as inappropriate settings for the various threshold parameters. These drop outs and spurious ellipses make the intra-frame correspondence task more difficult. Subsequent processing can overcome the limitations of the tracker by robustly determining intra-frame correspondence between ellipses.

We employ a simple technique to determine intra-frame correspondence between ellipses in a single movie. First, we group ellipses believed to track the same object between frames into ellipse chains. We use a weighted five-dimensional Euclidean distance metric on ellipse parameters to determine object continuity between adjacent frames. The chains are contiguous sequences that track the motion of a single ellipse for a subrange of frames in the movie. Due to noise, chains might not span the entire movie. Thus we subsequently relax the frame-adjacency requirement and attempt to attach the ends of the chains together to produce contiguous sequences of ellipses that span the entire movie. Relaxation of the frame-adjacency requirement reduces the problem of ellipse drop outs, while elimination of short chains reduces the problem of spurious ellipses. The final result of this technique is a set of ellipse sequences that track all of the participant objects in the event through the entire movie.

The result of applying our tracker to several complete movies is shown along with the original movies in figure 3. Only a small subset of key frames for each movie is shown. These movies depict *pick up*, *put down*, *push*, *pull*, *drop*, and *throw* events respectively. In these movies, the intra-frame ellipse correspondence is indicated by line thickness. Notice that it is fairly easy for a human to recognise the depicted event solely from the ellipse data. Our event recogniser attempts to mimic this ability.

4 Event Recognition

The output of our tracker consists of a stream of five parameters for each ellipse in each frame of each movie. Since movies typically have from two to five objects, this constitutes roughly 10 to 25 floating point numbers per frame. From this data stream we compute a larger feature vector. This feature vector contains both absolute, and relative, ellipse positions, orientations, velocities, and accelerations. More specifically, our feature vector includes the following features for each frame:

- Absolute features
  1. the magnitude of the velocity vector of the centre of each ellipse,
  2. the orientation of the velocity vector of the centre of each ellipse,
  3. the angular velocity of each ellipse,
  4. the first derivative of the area of each ellipse,
  5. the first derivative of the eccentricity of each ellipse,
  6. the first derivatives of each of the above five features,
- Relative features
  1. the distance between the centres of every pair of ellipses,
  2. the orientation of the vector between the centres of every pair of ellipses,

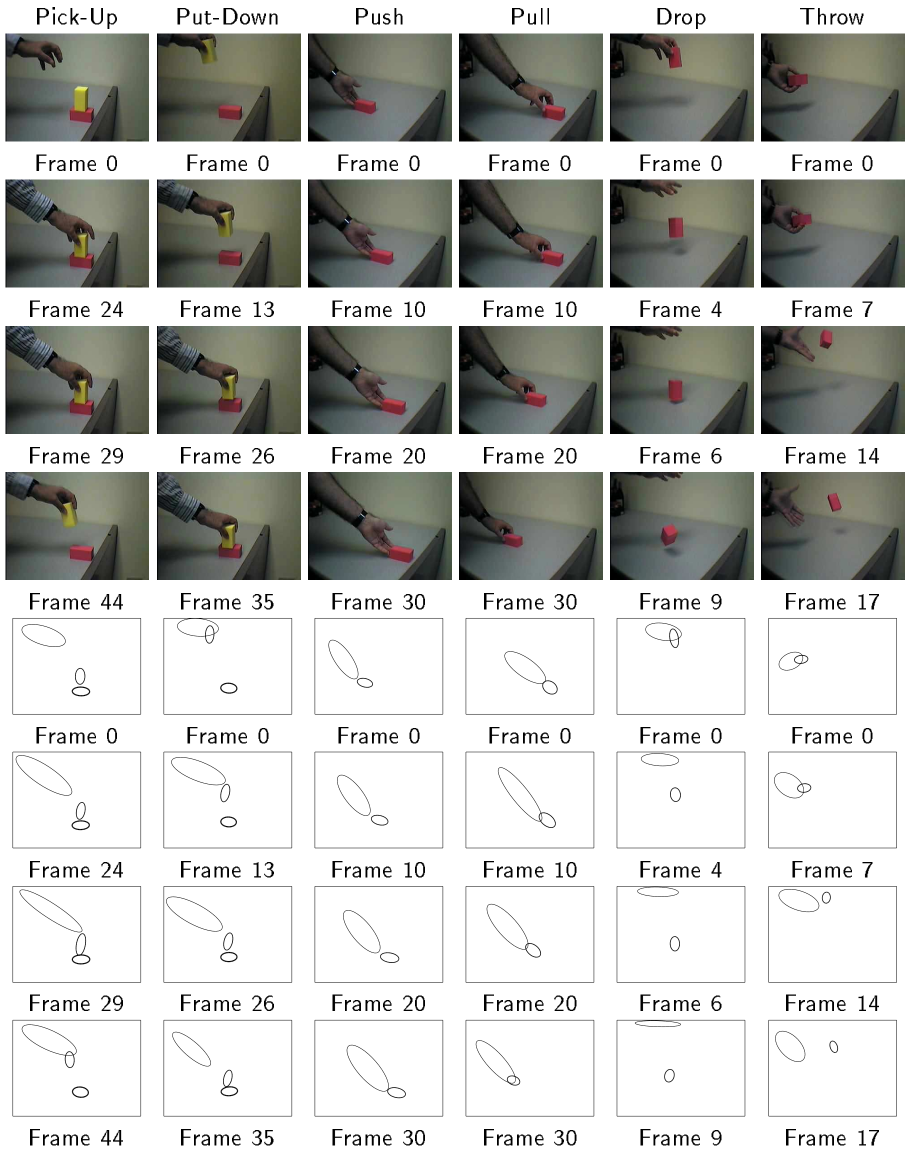
Pick-Up
Put-Down
Push
Pull
Drop
Throw
Frame 0 Frame 0 Frame 0 Frame 0 Frame 0 Frame 0
Frame 24 Frame 13 Frame 10 Frame 10 Frame 4 Frame 7
Frame 29 Frame 26 Frame 20 Frame 20 Frame 6 Frame 14
Frame 44 Frame 35 Frame 30 Frame 30 Frame 9 Frame 17
Frame 0 Frame 0 Frame 0 Frame 0 Frame 0 Frame 0
Frame 24 Frame 13 Frame 10 Frame 10 Frame 4 Frame 7
Frame 29 Frame 26 Frame 20 Frame 20 Frame 6 Frame 14
Frame 44 Frame 35 Frame 30 Frame 30 Frame 9 Frame 17

Fig. 3. Sample frames from several movies depicting six different events along with the result of applying our tracker to those movies.

3. the difference between the orientations of the major axes of every pair of ellipses,
4. for every pair of ellipses, the difference between the orientation of the major axis of the first ellipse and the orientation of a vector from the centre of that ellipse to the centre of the second ellipse, and
5. the first derivatives of each of the above four features.[3]

In the above, all derivatives are calculated as finite differences between two adjacent frames.

Using this feature vector, we adopt a maximum-likelihood approach to event recognition. We use supervised learning techniques to train a generative model for each class of events, from a set of training examples for each class, and subsequently use the derived generative models to classify new observations into existing classes. More specifically, if $p_1, \ldots, p_L$ constitute the feature vector sequences for a set of L movies, we find the parameters ψ for the generative model that maximise the joint likelihood of generating all of the training sequences.

$$\textsc{Train}(\{p_1, \ldots, p_L\}) \stackrel{\triangle}{=} \operatorname*{argmax}_{\psi} \prod_{l=1}^{L} P(p_l|\psi) \tag{1}$$

Then, if $\psi_1, \ldots, \psi_J$ constitute the parameters for J different event types, we classify a new feature vector sequence p as the class j that is most likely to have generated p.

$$\textsc{Classify}(p) \stackrel{\triangle}{=} \operatorname*{argmax}_{j} P(p|\psi_j) \tag{2}$$

We adopt Hidden Markov Models (HMMs) as the generative model within the above maximum-likelihood framework. Each event class j is described by a u_j-state model.[4] We intend these states to represent the major subevents of each event type. For instance, a *pick up* event might consist of two states. During the first state, the agent characteristically will move toward the patient while the patient is stationary above the source. During the second state, the agent will move together with the patient away from the source. Similarly, our HMMs can partition the other event types into distinct states representing easily interpretable subevents.

The HMMs in our implementation assume independent normal probability distributions over each feature in the feature vector. The mean and variance of each feature in each state of each model are adjusted in order to maximise the likelihood of that model generating the training data. The assignment of high and low variances to different features is tantamount to learning which features

[3] The results that we report in this paper are based on a simpler feature vector that contains only the first two of the aforementioned absolute and relative features along with their first derivatives. While we obtain good results with this smaller feature set, when classifying a small set of different event types, we believe that the larger feature set will be necessary to classify a larger set of event types.

[4] Currently, the number of states used to represent each event type is specified manually as a parameter to the training process.

are relevant to each subevent of each event type, and which are not, since the likelihood is more sensitive to changes in the values of those features having low variance than it is to those features having high variance. For example, the variance of the first derivative of the distance between the agent and the patient during the first state of a *pick up* event will be low, indicating that that feature is relevant, while the variance of the orientation of the velocity vector of the agent will be high, since the angle of approach during a *pick up* can vary significantly, indicating that that feature is not relevant. In contrast, the variance of that feature, namely the orientation of the velocity vector of an object, will be low during a *drop* event, since objects typically fall straight downward.

We train the parameters of the HMMs with the Baum-Welch reestimation procedure (Baum, Petrie, Soules, & Weiss, 1970) and use the Viterbi procedure (Viterbi, 1967) for classification. During training, we restrict the state-transition matrix to be upper triangular, thus disallowing non-self cycles in the state-transition graph. Our experience shows that such non-ergodic models generalise much better to new observations. This is not a severe restriction, since none of the event types that we have considered require repetitive subevents.

The training and classification procedures are somewhat more complex than the standard Baum-Welch and Viterbi procedures for a number of reasons. First, our tracker does not provide object-identity information. While our tracker does track the position of objects through time in the movie with ellipse sequences, it does not identify the objects that it is tracking. Second, our tracker can produce an ellipse sequence for an object that does not participate in the event or an object that does not really exist, for example a shadow. This is problematic because we need to know which tracked objects in the movie are participating in the event and what role those objects play within the event. Each event type specifies a fixed number of participant objects and each object plays a well-defined role in a given event type. Roles must be assigned to tracked objects both during training, to group together ellipse sequences from different movies that play the same role, and during classification, to assign ellipse sequences to existing roles in a model. We refer to this grouping problem as the *external correspondence* problem.

We determine the external correspondence among members of the training set by examining a set of candidate correspondences. Each candidate correspondence contains a subset of the ellipse sequences for each movie in the training set. Each subset is ordered to match corresponding ellipse sequences across the training movies. Conceptually, an HMM can be trained on each candidate correspondence, and the best correspondence can be chosen to be that which leads to a model that has the highest likelihood of generating the event instances in the training set.

Since evaluating all possible candidate correspondences would be intractable, we use a variant of the greedy algorithm to choose which candidate correspondences to evaluate. We choose one particular movie from the training set to be the canonical event. The movie chosen to represent the canonical event must contain the same number of ellipse sequences as there are participant objects in the given event type. An arbitrary order is chosen for the set of ellipse sequences in the movie representing the canonical event. Ordered subsets of the set of ellipse sequences in every other movie are then chosen to establish a candidate correspondence with the ordered set of ellipse sequences in the canonical event. This choice is done incrementally. First, a candidate correspondence is established between the canonical event movie and one other movie chosen from the training set. This correspondence is chosen by training an HMM on every possible correspondence between the canonical event and the other movie and choosing the HMM that has the highest likelihood of generating these two event instances. We then explore successively larger candidate correspondences. At each stage we maintain a set of r candidate correspondences for n movies and choose as the candidate correspondences for $n + 1$ movies the r best ways of augmenting a previous candidate with a new movie. At the last iteration we choose the best HMM among the r candidates correspondences constructed for the entire training set.

During classification, ellipse-sequence role assignments are determined in the following manner. We examine all possible permutations of all subsets of the set of ellipse sequences derived from a new observation. We then compute the likelihood that that sequence was generated by each of the models that specifies the same number of participant objects as the number of ellipse sequences in that candidate. The new observation is classified as being generated by the model that assigns the highest likelihood to some permutation of some subset of the set of ellipse sequences derived from that observation.

5 Experiments

To test our techniques, we filmed 72 movies using a SunVideo system to record image sequences with a resolution of 320×240 at 30 frames-per-second compressed in JPEG format. These 72 movies comprised twelve instances for each of the six event types *pick up*, *put down*, *push*, *pull*, *drop*, and *throw*. All of these movies were filmed in a relatively uncluttered desk-top environment, though some movies contain extraneous objects. The camera position changed somewhat from movie to movie. Four coloured foam blocks (one red, one blue, one green, and one yellow) were used as props to enact the six different event types. We clipped these movies by hand so that the beginning and end of the event coincided with the beginning and end of the movie. The *pick up* and *put down* movies were uniformly clipped to be 50 frames long, the *push* and *pull* movies were uniformly clipped to be 35 frames long, while the *drop* and *throw* movies were uniformly clipped to be 20 frames long. We then processed each of the 72 movies with our tracker. Of these 72 movies, 36 were randomly selected as training movies, six movies for each of the six event types. We constructed six five-state non-ergodic Hidden Markov Models, one for each of the six event classes. We then classified all 72 movies, i.e. both the original training data as well as the data not used for training, against all six event classes. Our model correctly classified all 36 of the training movies and correctly classified 35 out of the 36 test movies. One *drop* movie was misclassified as a 'throwing' event. Our classifier, however, selected 'drop' as the second-best choice for this movie. This misclassification appears to result from poor tracking.[5]

6 Discussion

We are cautiously optimistic with our results. We plan to test our techniques with a larger set of event types using a wider range of participant objects filmed in different length movies in a wider range of environments. We also plan to evaluate the robustness of these techniques in the face of event occurrences that exhibit larger variance in motion profile. Furthermore, while our event recogniser can work with output from any tracker, the class of events that we can classify is to some extent dependent on the properties of our current tracker. We have built a variety of trackers to drive our recogniser and are currently tuning a more sophisticated tracker that we believe will allow us to apply our event recogniser to a wider class of events and environments. A longer-term objective is to eliminate the need for manually specifying the beginning and end of each event by automatically performing temporal segmentation among events that potentially overlap in space and time.

Our work has surprising similarities with earlier work on event recognition reported in Siskind (1992, 1995). That work suggests that event classes can be described with temporal-logic formulas, such as

$$\textsc{PickUp}(x, y) \overset{\triangle}{=} \exists w \left(\begin{array}{c} \textsc{Part}(w, x) \wedge \\ \left[\left(\exists z \begin{pmatrix} \textsc{Disjoint}(z, w) \wedge \\ \textsc{Supported}(y) \wedge \\ \textsc{Supports}(z, y) \wedge \\ \textsc{Contacts}(z, y) \end{pmatrix} \right) ; \begin{pmatrix} \textsc{Translating}(w) \wedge \\ \textsc{Contacts}(w, y) \wedge \\ \textsc{Attached}(w, y) \wedge \\ \textsc{Supports}(x, y) \wedge \\ \textsc{Translating}(y) \end{pmatrix} \right] \end{array} \right)$$

over a set of primitives that describe spatial relations, motion, and force-dynamic relations such as support, contact, and attachment. On the surface, the approach taken in that work might appear to be fundamentally different from the approach taken here. There are deep similarities, however. The features used as input to our HMMs are analogous to conceptual-structure representations proposed by Jackendoff (1990). For example, the orientation of the velocity of the centre of each ellipse can be viewed as a quantified representation of the conceptual structures GO(x,UP) and GO(x,DOWN). Similar analogies can be drawn for all of the remaining features.

Temporal-logic formulas can be viewed as regular expressions over symbols that represent the occurrence of primitive events described by these Jackendovean representations. Such symbols can be viewed as thresholded feature values, while the temporal-logic formulas can be viewed as finite-state recognisers over these thresholded values. Our HMMs can be viewed as probabilistic finite-state recognisers over the same features. Thus, while Siskind (1992, 1995) only presents techniques for using event descriptions to recognise events, assuming that event descriptions are produced by hand, the work presented here

[5] All of our training and test data, as well and source code for our implementation is available from `http://www.cs.toronto.edu/~qobi`.

extends this approach to learn event descriptions in a supervised fashion. The work presented here, however, uses only spatial motion features and not force-dynamic ones. We pursued this approach because of our initial concern that it would be difficult to robustly recover force-dynamic relations from camera input. Mann et al. (1996), however, present some encouraging results that show that recovery of force-dynamics relations from camera input is feasible. In the future we hope to integrate the force-dynamics recovery techniques described in that paper with the event-recognition techniques described here.

7 Conclusion

Event perception is a fundamental cognitive faculty. Any solution to the ultimate goal of artificial intelligence, namely endowing a computer with human-level intelligence, must embody a mechanism for perceiving events. Event recognition plays a central role in human cognition, perhaps even a more central role than object recognition. Consider, for instance, the fact the sentence structure of natural languages is built around verbs rather than nouns. Furthermore, event recognition appears to be easier than object recognition, and to rely on less information. It may even be the case that event recognition is an evolutionarily more primitive faculty.

Earlier work adopted the tacit assumption that object recognition precedes event recognition. It is interesting to consider the reverse assumption. If event recognition is easier and evolutionarily more primitive than object recognition, perhaps object recognition can be facilitated by information provided by prior event recognition. For instance, it might be difficult to reliably segment, recognise, and track a hammer on the basis of an object model, geometric or otherwise. Yet, it might be easy to recognise the characteristic motion of a hammering event even without detecting a hammer. Having hypothesised a hammering event, and produced a course abstract object pose sequence in the process of forming such a hypothesis, it might be possible to use that knowledge to guide the search for a hammer to confirm that hypothesis. Prior event recognition could prune the space of object models to consider and also provide course grouping and tracking data, using a novel source of top-down constraint.

We have applied a particular strategy to event classification, namely maximum likelihood. To the best of our knowledge this approach has not been previously applied to this task. Maximum-likelihood methods, however, have proven extremely successful in machine analysis of speech. These methods provide several important methodological advantages. First, they provide graded levels of performance rather than monolithic success or failure. This is crucial when working on hard problems for which there is no unequivocally successful method in sight. Second, they provide two well-defined methods for improving performance: additional training data and more accurate generative models. The course of speech recognition research over the past few decades can be seen as long intervals of shallow but steady performance growth punctuated by a small number of discontinuous paradigm changes. Third, systems that maintain graded decisions internally, delaying all categorical decisions as long as possible, are more robust than those that operate as a pipe of categorical decision processes. Systems in the later category are brittle, since a minor inaccuracy can affect a categorical decision early in the pipe and that decision can irrevocably affect the ultimate outcome. Visual event perception shares much in common with speech analysis in that both deal with time-varying signals. It is not surprising that techniques that have been applied to speech could also be applied to visual event perception. The success of maximum-likelihood methods should not be taken to imply that they are the only feasible approach to performing event classification. Rather, we intend the results of this paper to be taken simply as encouragement that the problem of event classification can be solved. Given this realisation, we encourage other researchers to explore alternate techniques for solving this problem.

Acknowledgments

We wish to thank Peter Dayan and Sven Dickinson for invaluable assistance on this project and Richard Mann for offering his comments on an earlier draft of this paper. Any remaining errors are, of course, the sole responsibility of the authors. This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

References

Abe, N., Soga, I., & Tsuji, S. (1981). A Plot Understanding System on Reference to Both Image and Language. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 77–84, Vancouver, BC.

Badler, N. I. (1975). Temporal Scene Analysis: Conceptual Descriptions of Object Movements. Tech. rep. 80, University of Toronto Department of Computer Science.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A Maximization Technique Occuring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, *41*(1), 164–171.

Jackendoff, R. (1990). *Semantic Structures*. Cambridge, MA: The MIT Press.

Johansson, G. (1973). Visual Perception of Biological Motion and a Model for its Analysis. *Perception and Psychophysics*, *14*(2), 201–211.

Mann, R., Jepson, A., & Siskind, J. M. (1996). The Computational Perception of Scene Dynamics. In *Proceedings of the Fourth European Conference on Computer Vision*, Cambridge, UK: Springer-Verlag.

Marburger, H., Neumann, B., & Novak, H. (1981). Natural Language Dialogue About Moving Objects in an Automatically Analyzed Traffic Scene. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 49–51, Vancouver, BC.

Marr, D., & Vaina, L. (1982). Representation and Recognition of the Movements of Shapes. *Proc. R. Soc. Lond. B*, *214*, 501–524.

Nagel, H. (1977). Analysing Sequences of TV-Frames. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, p. 626, Cambridge, MA.

Neumann, B., & Novak, H. (1983). Event Models for Recognition and Natural Language Description of Events in Real-World Image Sequences. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 724–726, Karlsruhe.

Okada, N. (1979). SUPP: Understanding Moving Picture Patterns Based on Linguistic Knowledge. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, pp. 690–692, Tokyo.

O'Rourke, J., & Badler, N. I. (1980). Model-Based Image Analysis of Human Motion Using Constraint Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *2*(6), 522–536.

Pinhanez, C., & Bobick, A. (1995). Scripts in Machine Understanding of Image Sequences. In *AAAI Fall Symposium Series on Computational Models for Integrating Language and Vision*.

Rashid, R. F. (1980). Towards a System for the Interpretation of Moving Light Displays. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *2*(6), 574–581.

Siskind, J. M. (1992). *Naive Physics, Event Perception, Lexical Semantics, and Language Acquisition*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Siskind, J. M. (1995). Grounding Language in Perception. *Artificial Intelligence Review*, *8*, 371–391.

Thibadeau, R. (1986). Artificial Perception of Actions. *Cognitive Science*, *10*(2), 117–149.

Tsotsos, J. K., Mylopoulos, J., Covvey, H. D., & Zucker, S. W. (1980). A Framework for Visual Motion Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *2*(6), 563–573.

Tsuji, S., Morizono, A., & Kuroda, S. (1977). Understanding a Simple Cartoon Film by a Computer Vision System. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 609–610, Cambridge, MA.

Viterbi, A. J. (1967). Error Bounds for Convolutional Codes and an Asymtotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, *13*, 260–267.

Waltz, D. L. (1981). Toward A Detailed Model of Processing for Language Describing the Physical World. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pp. 1–6, Vancouver, BC.

Yamamoto, J., Ohya, J., & Ishii, K. (1992). Recognizing Human Action in Time-Sequential Images Using Hidden Markov Model. In *Proceedings of the 1992 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 379–385. IEEE Press.

This article was processed using the LaTeX macro package with ECCV'96 style