Using Temporal Coherence to Build Models of Animals

Deva Ramanan and D. A. Forsyth
Computer Science Division
University of California, Berkeley
Berkeley, CA 94720
ramanan@cs.berkeley.edu, daf@cs.berkeley.edu

Abstract

This paper describes a system that can build appearance models of animals automatically from a video sequence of the relevant animal with no explicit supervisory information. The video sequence need not have any form of special background. Animals are modeled as a 2D kinematic chain of rectangular segments, where the number of segments and the topology of the chain are unknown. The system detects possible segments, clusters segments whose appearance is coherent over time, and then builds a spatial model of such segment clusters. The resulting representation of the spatial configuration of the animal in each frame can be seen either as a track — in which case the system described should be viewed as a generalized tracker, that is capable of modeling objects while tracking them — or as the source of an appearance model which can be used to build detectors for the particular animal. This is because knowing a video sequence is temporally coherent — i.e. that a particular animal is present through the sequence — is a strong supervisory signal. The method is shown to be successful as a tracker on video sequences of real scenes showing three different animals. For the same reason it is successful as a tracker, the method results in detectors that can be used to find each animal fairly reliably within the Corel collection of images.

1. Introduction

In learning circles, supervised data is usually thought of as data where each data item comes with information identifying its class. However, useful supervisory information can come in a variety of important forms — for example, in multiple instance learning, one is given a collection of packets of examples, and told which packets contain positive examples, but not which example is positive. It is still possible to learn under these circumstances [9, 10]. It has been shown that, given sentences in two languages, where one knows the sentences are translations of one another but not the exact correspondence between words, one can build a lexicon [11]. Similarly, given bags of data that consist of representations of image regions and words — here one knows that the words come from some subset of the regions but not which word comes from which — it is possible to learn to map words onto regions [4]. All this follows from observations that really useful supervisory information can appear in hidden forms [14, 16].

It is commonly the case that, when building models of objects from images, it is relatively easy to associate objects with names but relatively difficult to determine which pixels in the image are associated with the object. This is why, for example, it is quite common to model objects by placing them on known, simple backgrounds. This isn't always possible, but we do not wish to have to manually segment the relevant object in a large set of training examples. This manual segmentation is an important, often forgotten, part of the supervision process. In this paper, we show that temporal coherence can be used as supervisory information to supersede the manual segmentation process.

1.1. Tracking as Model Building

An alternative view of this paper is that it describes a **generalized tracker**, which is able to build models of objects automatically and then track the object by detecting the presence of the model. In this respect, our work exposes a connection between tracking and object detection; namely, tracking an object is easier if we can detect it. Conversely, a good tracker should yield a good object detector.

The tracking literature is too large to review in detail here; a brief discussion and review appears in [6, 7]. Typically, current trackers have a dynamical model which is applied to object appearance and movement, and use this model to obtain some maximum *a posteriori* (MAP) estimate of object appearance and pose in each frame. The dynamical model of appearance is Markovian, which is extremely convenient for inference but a poor model of actual object appearance — more consistent with experience is a model that says that, *expressed in appropriate coordinates*, appearance hardly changes at all over a motion se-

quence. We examine this constant appearance model further in section 1.2. Inference is now harder because one needs to determine a single appearance that works for most or all frames. However, approximate inference is possible, as described in [13]. We repeat some information here for the convenience of the reader. The basic assumption of this tracker is that appearance is a stronger cue to the configuration of a person in the next frame than dynamics, because the body can move very fast but it takes some time to change clothes. This tracker assumes that the image of the body consists of some set of segments of known scale, assembled according to a known plan (i.e. lower leg attached to upper leg attached to torso, etc.). The tracker works by building an appearance model of putative actors, detecting instances of that model, and linking the instances across time.

The **appearance model** approximates a view of the body as a puppet built of colored, textured rectangles. The model is built by applying detuned body segment detectors to some or all frames in a sequence. These detectors respond to roughly parallel contrast energies at a set of fixed scales (one for the torso and one for other segments). A detector response at a given position and orientation suggests that there may be a rectangle there. For the frames that are used to build the model, we cluster together segments that are sufficiently close in appearance — as encoded by a patch of pixels within the segment — and appear in multiple frames without violating upper bounds on velocity. It is possible to determine whether these bounds are violated from minimal camera calibration information (the scale in a scaled orthographic model). Clusters that contain segments that do not move at any point of the sequence are then rejected. The next step is to build assemblies of segments that lie together like a body puppet. The torso is used as a root, because our torso detector is quite reliable. One then looks for segments that lie close to the torso in multiple frames to form arm and leg segments. Note that this procedure does not require a particularly reliable initial segment detector, because we are using many frames to build a model — if a segment is missed in a few frames, it can be found in others which will yield the appearance model. We are currently assuming that each individual is differently dressed, so that the number of individuals is the number of distinct appearance models, a reasonable assumption for many but not all application domains. It is straightforward to instance the model, though this complicates counting people and separating individual tracks.

Detecting the appearance model is straightforward, and uses the method of Felzenschwalb and Huttenlocher [5]. This yields, for each frame in the sequence, the best match to each known individual, laid out as a puppet. If the match cost exceeds some threshold, we assume that the individual is absent.

In this paper, we show that it is not necessary to know anything about the segment model to track an object with this scheme (section 2). Once one has done so, one possesses an appearance model of the object *that can be used to detect it in other images* (section 4). As a result, the assumption of motion coherence is being used as supervisory information to identify which pixels in the image belong to the object being learned.

1.2 Constant Appearance Model

The constant appearance model from [13] is unusual in the tracking literature; most trackers assume a Markovian appearance model to make inference easier (by conditioning on the current frame, we can forget about the past when predicting the future). We argue this is wrong because all past frames tell us something about the underlying appearance of the object. Consider drawing sample trajectories from two toy models of state; Markov $\{x^i \sim N(x^{i-1}, \sigma^2), x^1 \sim N(0, \sigma^2)\}$ versus i.i.d. from a constant distribution $\{x^i \sim N(0, \sigma^2)\}$. The former is essentially a random walk where our final state might be arbitrarily far from our initial one. If our state encodes position, this is perfectly reasonable – given the absence of image observations, we might believe an object travels far from its starting point. If our state encodes appearance, this is a poor model because it allows a blue shirt to turn yellow. In practice, with good image observations, we can constrain this random walk by reweighting samples with the image likelihood. But once we hit a frame with an ambiguous image observation, the random walk tendency resumes, causing the track to drift (an illustrative example is figure 6 from Sickenbladh *et al* [15]). With a constant appearance model, we force our initial and final appearance to be similar – the blue shirt stays blue, even given an ambiguous image.

Turning to our analogy between tracking and object detection, we find our constant appearance model is quite standard in the detection literature. Implicit in most unsupervised learning algorithms is the claim that images of different objects from the same class look similar [4, 14, 16]. One learns what a zebra looks like by finding the common bits in a collection of images of different zebras. We observe the same applies for a collection of images of the *same* zebra. A convenient place to find such data is a video of a zebra, where in fact we get the added constraint that the configuration of zebra bits from one frame to the next cannot change too fast.

1.3. General Approach

Assume we are given a video sequence and told that, say, a zebra is present. We wish to build a visual model of a zebra that can be used to find zebras in this sequence — so the animal can be tracked — and to find zebras in other images. We would like to do so using the minimum of information

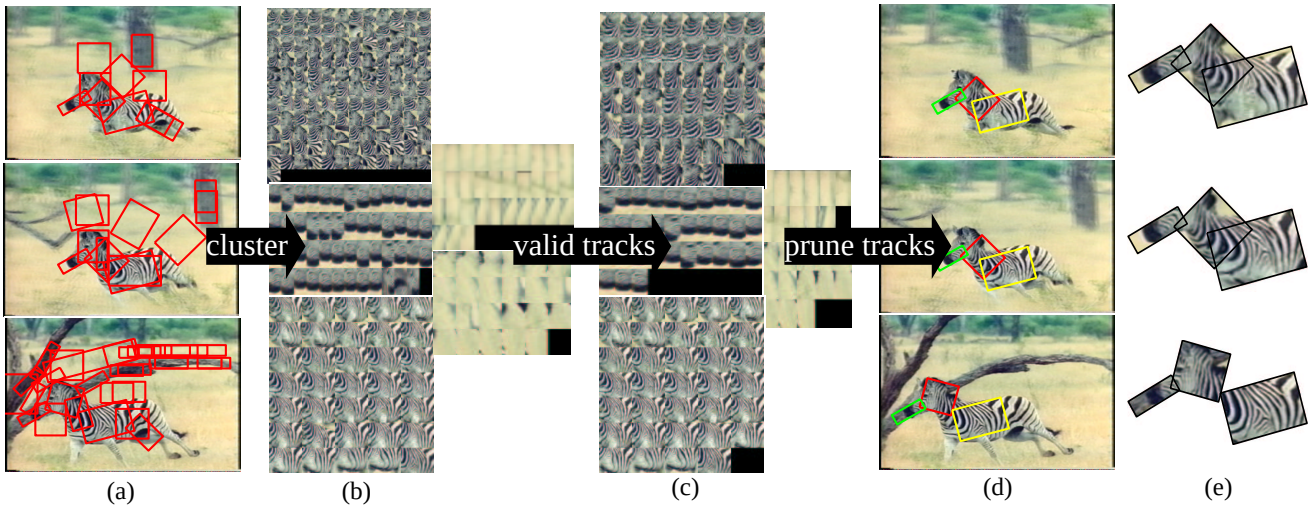

Figure 1. *Learning an object model by clustering. We first search for candidate segments using local detectors (we show 3 sample frames in (a)). We cluster the image patches together in (b). From each cluster we extract a valid sequence obeying our motion model in (c). We prune away the short tracks to retain the final segments in (d). We are left with the final object model in (e), consisting of the head, neck and body of the zebra.*

about zebras, so that the method can be used for other animals. If we know that the zebra is the only animal present, there are two powerful cues that can be used to establish what it looks like: First, the zebra is, rather roughly, assembled out of body segments. Second, the segments will look the same from frame to frame — their appearance is coherent in time. This suggests the following strategy:
Detect candidate segments with a detuned segment detector.
Cluster the resulting segments to identify body segments that look similar across time
Prune segments that move too fast in some frames.
Assemble a spatial model from these segments.
In what follows, we show that this results in a satisfactory model for a variety of animals.

2. Detecting and Clustering Segments

We model segments as cylinders and generate candidates by convolving the image with a template that responds to parallel lines of contrast (at a variety of orientations and scales), suppressing the non-maximum responses. We used 15 orientations, and 25 scales (5 lengths crossed with 5 widths). We expect our low-level segment detectors to perform poorly with many false positives and missed detections, such as those in figure 1-(a).

2.1. Clustering Segments

Since we do not know the number of segments in our model (or for that matter, the number of segment-like things in the background), we do not know the number of clusters *a priori*. Hence, clustering segments with parametric methods like gaussian mixture models or k-means is difficult. We opted for the mean-shift procedure [2], a non-parametric density estimation technique.

We create a feature vector for each candidate segment, consisting of a normalized color histogram in the Lab color space, appended with shape information (in our case, simply the length and width of the candidate patch). Note that this feature vector is to be used for *clustering*, for which it is sufficient. The representation of appearance is not limited to this feature vector.

The color histogram is represented with projections onto the L, a, and b axis, using 10 bins for each projection. Hence our feature vector is $10 + 10 + 10 + 2 = 32$ dimensional. We scale the histogram and scale dimensions so as to obtain a meaningful $\mathcal{L}_2$ distance for this space. Further cues — for example, image texture — might be added by extending the feature vector, but appear unnecessary for clustering.

Identifying segments with a coherent appearance across time involves finding points in this feature space that are (a) close and (b) from different frames. This is difficult to do; we drop requirement (b), which can be imposed on clusters *post hoc*, and concentrate on (a). The mean-shift procedure is an iterative scheme where we find the mean position of all feature points within a hypersphere of radius h, recenter the hypersphere around the new mean, and repeat until convergence. We initialize this procedure at each original feature point, and regard the resulting points of convergence as cluster centers. For example, for the zebra sequence in figure 1, starting from each original segment patch yields five points of convergence (denoted by the centers of the five

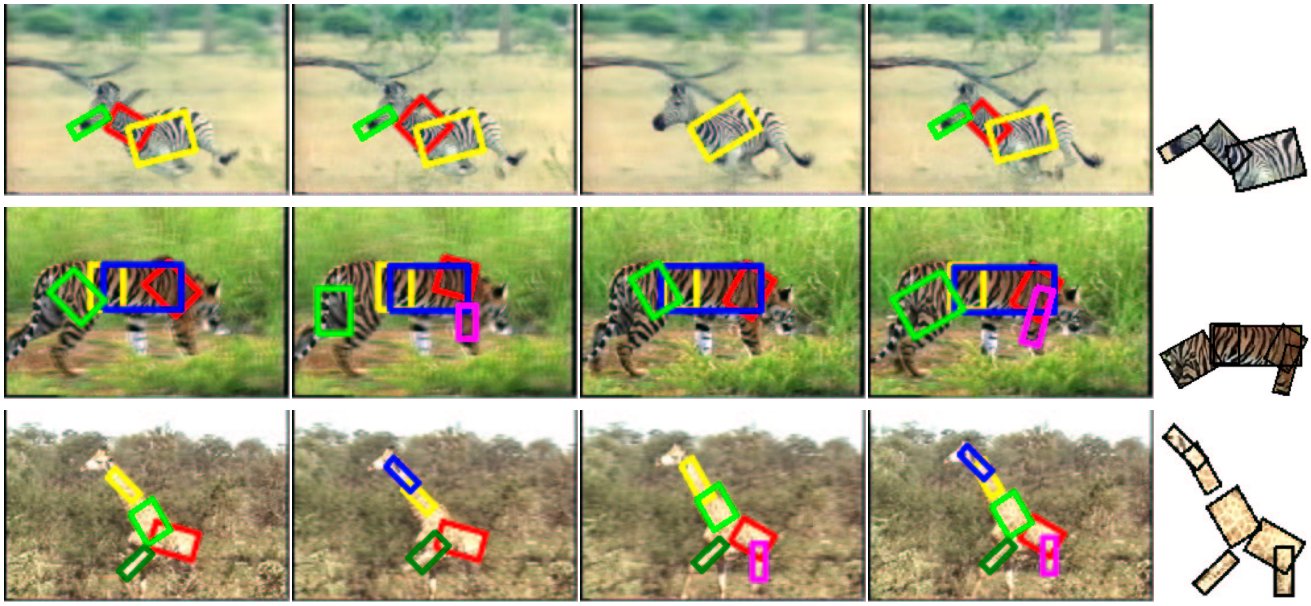

Figure 2. *Our activities could be viewed as building a generalized kinematic tracker, using temporal coherence in segment appearance. From this view, they are successful, as the figure indicates. Each* **row** *shows four frames from a sequence of a moving animal. Superimposed on each frame is a segment model, the color of the block indicating the temporal correspondence of segments. Notice that these segments stay in about the right place, and correspond appropriately, despite the animal's movement. The appearance model is built* automatically, *and is shown on the* **right**.

clusters in (b)).

As a post-processing step we merge clusters which contain members located within h of each other (in a greedy fashion, starting with the two closest clusters). We account for over-merging of clusters by extracting multiple valid sequences from each cluster during step (c) (for each cluster during the third step in figure 1, explained further in the following section, we keep extracting sequences of sufficient length until none are left). Hence for a single arm appearance cluster, we might discover two valid tracks of a left and right arm.

2.2. Velocity Bounds and Track Requirements

As figure 1 indicates, not every coherent patch is associated with a moving figure. The second column of clusters in 1-(b) are background regions. However, at this point cluster elements are neither constrained to move with bounded velocity nor required to form a sequence — there might be several elements from the same frame.

We now find the most likely sequence of candidates for each cluster that obeys the velocity constraints. By fitting an appearance model to each cluster (typically a Gaussian, with mean at the cluster mean and standard deviation computed from the cluster), we can formulate this optimization as a straightforward dynamic programming problem. Let P^i be the position of a segment in the i^{th} frame. Since these variables represent *position* (rather then appearance), we can model them as Markovian; i.e. $Pr(P^i|P^{1:i-1}) = Pr(P^i|P^{i-1})$. The reward for a given candidate is its likelihood under the gaussian appearance model, and the temporal rewards are '0' for links violating our velocity bounds and '1' otherwise. We add a dummy candidate to each frame to represent a "no match" state with a fixed charge. By applying dynamic programming, we obtain a sequence of segments, at most one per frame, where the segments are within a fixed velocity bound of one another and where *all* lie close to the cluster center in appearance. As figure 1-(c) demonstrates, this results in a somewhat smaller set of segments associated with each cluster, particularly the second column of background clusters. Background segments which happen to cluster together often do not move like true segments.

We now discard those tracks which are not long enough. In figure 1-(c), this results in pruning away the second two clusters. Note we could impose other tests of validity beyond the length of a track; we might require that a segment move at some point, and so we would prune away a track which is completely still. Alternatively, if we are given two different videos of the same animal, we might prune away those clusters which do not appear in both.

The segments belonging to the remaining three clusters are shown in Fig 1-(d). These constitute our learned object model depicted in (e); we can now learn the spatial con-

straints between the three segments and a more precise temporal motion model (along with the appearance model from the clustering step).

2.3. Building a Spatial Model

We now have a set of spatio-temporal tracks. Each track contains instances of a segment of known appearance — obtained from the clustering — where the instances appear in many frames and move with bounded velocity. This set of tracks is a track of the animal (figure 2), established without information about its appearance.

This set of tracks can also be seen as a set of segment groups, one in each frame, where frame to frame correspondence is known (figure 2). Each track is a candidate segment for a body model; currently, we simply accept all tracks.

We build a spatial model using the same procedure as [8]. Note in our case, we obtain the set of hand-labeled training examples automatically from the tracker; it outputs a variety of valid configurations of the zebra head, neck, and body (we do not need the precise segment labels so long as we know their correspondence between frames). For the readers convenience, the procedure is briefly outlined here. We can imagine a fully connected graphical model of segment positions, $P_{head}, P_{neck}, P_{body}$. Each link represents a joint distribution $\Pr(P_{seg1}, P_{seg2})$ of pairwise segment positions, to which we fit a gaussian using the tracker data. We then find the minimum entropy spanning tree, disregarding the extraneous edges. This yields a tree spatial model of segment positions ([8] used a mixture of trees, but we found a simple tree model to suffice).

Using the learned spatial model and the appearance model from the clustering, we now have a zebra detector. We use the method of Felzenschwalb and Huttenlocher [5] to find the best match for a given image efficiently. We can use our zebra detector on the original tracked frames to yield more accurate tracks [13], or use it to query a collection of images to find zebra pictures.

3. Clustering as Approximate Inference

The algorithm discussed above is, in fact, an approximate inference procedure for the graphical model shown in Fig 3. For simplicity, assume we are dealing with only one segment. We write the 32 dimensional feature vector extracted from the patch centered at pixel (x, y) from the i^{th} image, oriented at angle θ with length l and width w as $Im^i(x, y, \theta, l, w)$. Hence we can interpret the i^{th} image as a *set* of feature vectors $\mathbf{Im^i} = \{Im^i(\ldots)\}$; this is the only observed quantity in our model (Fig.3-a). We assume there is an unobserved variable encoding the configuration of the true segment patch at frame i, which we write as $P^i = [x^i y^i \theta^i l^i w^i]$ (we also call this the segment "position"). Let C be the true, *constant* underlying segment

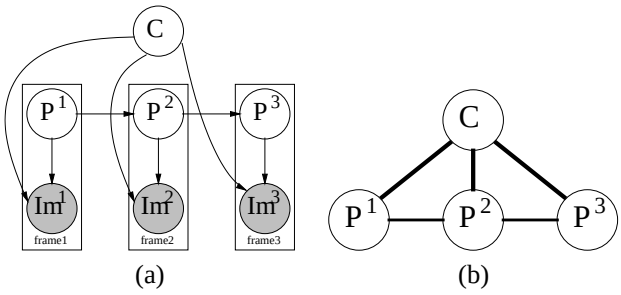

Figure 3. *The graphical model for segment inference. The model in (a) encodes the fact that each image instance of a segment has the same appearance (encoded in C) but appears at different places. In (b), the simplified undirected form of the model.*

appearance and shape (represented as a 32 dimensional feature vector). Hence we can say that $Im^i(P^i)$ is distributed as

$$\phi(||Im^i(P^i) - C||) \tag{1}$$

where ϕ is a kernel capturing appearance variation. We use the Epanechnikov (triangle) kernel with bandwidth h. We assume the background image patches are unstructured (i.i.d. uniform).

We emphasize that our model still allows segment appearance to change (slightly) from frame to frame – the appearance in each frame is an i.i.d. sample from a distribution which is constant (equation 1). Most other trackers model temporal changes in appearance by replacing the constant C in figure 3 with a temporally varying copy C^i in each frame plate. We argued in section 1.2 that this Markovian appearance model is a poor choice.

We can simplify our model by turning to the undirected case in Fig.3-b. Note that since we observe $\mathbf{Im^i}$ we only use a 2-dimensional slice of the 3-dimensional "table" $\Pr(\mathbf{Im^i}|P^i, C)$. Hence the image observations specify a particular potential between P^i and C (i.e., this is the standard moralization that results from conversion of a directed graphical model to an undirected one). Note our image observations are now implicitly represented in the potentials $\psi_i(C, P^i)$, while our motion model lives in the potentials $\psi_{link}(P^i, P^{i-1})$.

The algorithm described in section 2 is a loopy-type inference procedure for the model in Fig.3-b (see also [13, 3, 12]). In particular, we are passing messages along a set of different subtrees of this model. The first subtree is shown in figure 4.

While clustering does not immediately seem like an inference procedure, it is an approximate procedure to obtain likely values of C. In particular, the domain of C is continuous, which is awkward for inference. Assume that we wish to obtain good values of C from the tree of fig-

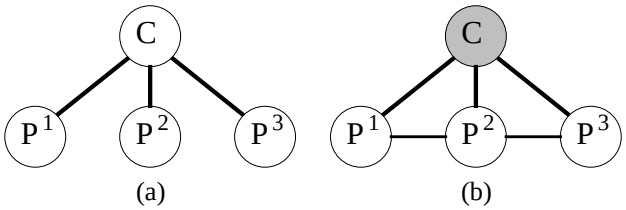

Figure 4. *Approximate inference on the model of figure 3 proceeds by inference on embedded trees, as in [12]. We use two; tree (a) is the most difficult, as it is practically very difficult to apply dynamic programming because C is a continuous variable. Instead, we use an approximation by clustering, described in greater detail in the text, which identifies the appearance of segments that occur many times; such segments are likely to be close to extrema of the dynamic programming criterion, which would look for a value of C that was like many image instances. Once C has been determined, we must do inference on tree (b), which is relatively straightforward.*

ure 4-(a). We wish to obtain values of C and P^i that maximize $\psi_1(C, P^1)\psi_2(C, P^2)\dots\psi_k(C, P^k)$, where the image information is implicit in the ψ_i's (whence the subscript).

Now this corresponds to choosing a C and a set of P^i such that the image segments identified by P^i all look like C. If C was defined over a discrete domain, all we'd be doing is dynamic programming: for each value of C, we'd choose the best P^i for each i, form the product, and then choose the C with the best product — we label this $\hat{C}$. This search is not easy for a continuous domain (e.g. [1]).

However, we know that we are looking for, in essence, a point in the domain of C such that there are many image segments that look like that point. Now assume we have a detuned but usable segment detector. It will then detect many, but not all, instances of the relevant segment and some background segments too. The instances of the relevant segment will look like one another. This means that, by clustering the representations of the segment appearances, we are obtaining a reasonable approximation to $\hat{C}$. In particular, finding local modes of the posterior on C using a Parzen's window estimate with a kernel ϕ is equivalent to the mean-shift algorithm. Using a more sophisticated appearance kernel ϕ reduces to using a weighted mean in each iteration of the mean-shift procedure. We possess no formal information on the quality of the approximation.

3.1. Multiple Segments

We now incorporate a multi-segment object model by interpreting each cluster as a *unique* segment, instantiating multiple copies of the model Fig 3-(b), one for each cluster. We can partly justify this procedure by our aggressive post-clustering merging of clusters; any left-over clusters which remain separate are likely to be different segments, and not multiple modes of a single segment.

The mean-shift procedure yields, in addition to likely values of C, a collection of nearby good values; these are the clusters corresponding to each convergence point. We exploit this fact to learn a new appearance kernel ϕ fit to the clustered points; this was our previous step of learning a gaussian for each cluster. We now can treat C as an observed quantity, for each instantiation of figure 3-(b). Inferring $\{P^i\}$ from such a model (figure 4-(b)) is straightforward; this is just our dynamic programming solution to find the most likely sequence of candidates given a known appearance. Note our initial claim of segment positions $\{P^i\}$ being Markovian is only true when we condition on C. Finally, we disregard those instantiations we deem invalid (i.e., not existing for enough frames).

4. Results

Tracking: Our activities could be described as building a generalized kinematic tracker. Taking this view, our system is successful, as figure 2 indicates. These show typical frames for three sequences depicting different moving animals. The same program was used in each case. In each case, the tracks were not hand initialized; the program builds an appearance model and a spatial model for the animal automatically, then identifies instances automatically. In each sequence, the animal's body deforms considerably, the zebra because it is moving very fast, the giraffe and the tiger because giraffes and tigers deform a lot when they move. Nonetheless, the program is able to build an appearance model that is clearly sufficient to capture the essence of the moving animal, but lacks some details. In particular, legs are narrow and fast, and very difficult to track. Furthermore, the temporal correspondences for the segments — which are indicated by colored outlines in the figures — are largely correct. Finally, the tracker has been able to identify the main pool of pixels corresponding to the animal in each frame.

Following [13], we evaluate our tracker using detection rates (figure 5) obtained from the original video. Our algorithm builds a representation of each animal as a collection of segments. We define a correct detection for a segment to occur when it is found anywhere on the animal body. Defining precise ground truth for segment location is hard because many segments do not correspond to actual animal limbs. In practice, the tracked segments tend not to drift on the animal body, so we believe the localization to be largely accurate. Our correct detection and false positive rates are quite good for all the segments for the 3 animals.

Object Detection: Our activities could be described as using motion coherence information to build an appearance model of an animal. Taking this view, our system is successful, as figure 6 demonstrates. We took the appearance mod-

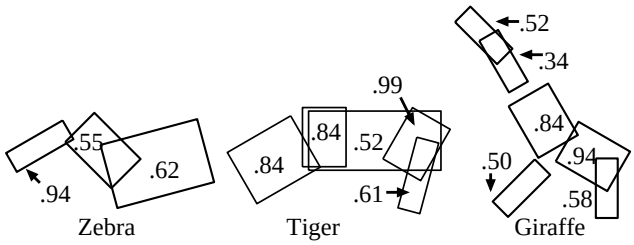

Figure 5. *As in [13], we evaluate our trackers using detection rates for the original videos. Our algorithm builds representations of animals as a collection of segments. We overlay the probability of correct detection over each segment. We define a correct detection to occur when a segment is found on the animal body. Since all our tracked segments lie on the animal, our false positive rate for each segment is 0. Our correct detection rates are also quite good.*

els and the spatial models built during the tracking phase and used them to match to images of the relevant animals from the Corel collection of royalty-free images. We built a test pool with 100 random images, not labeled with either "zebra", "tiger", or "giraffe" plus 50 "zebra", 120 "tiger", and 34 "giraffe" images. To test a given detector, we use all the corresponding animal images as positives, and randomly select 100 images from those remaining in the pool as negatives. We repeat this experiment 200 times, each time sampling with replacement, to generate the mean performance curves and standard deviation error bars. The red (dashed) line suggests that the segment appearance representation used for clustering (a color histogram) many not be sufficient for building an animal detector.

Our model building algorithm tracks the animal in a video sequence. This means we know where the animal pixels are in each frame of that sequence, and can use them to build more sophisticated representations. By appending our color histogram with a texture descriptor learned from the pools of animal pixels, we can increase our performance. This is particularly true for the zebra. We use a texture model similar to [14], which is discussed further in upcoming work. We train this model using 100 random Corel images (not in our test set) as negatives. Our tiger detectors perform reasonably well and equivalently to each other, probably because color is a sufficient cue for detection. The giraffe detectors perform poorly, because the color and texture models learned from the washed-out giraffe video are quite bad. In order to build good detectors we need video with resolution and spatial frequency content similar to our test images.

4.1. Evaluation of detection results

The error bars for our operating curves are quite large, indicating performance is heavily dependent on the test set. Telling zebras apart from other striped animals such as tigers is much harder than telling zebras apart from cars. There is no standard test set for animal detection; the best comparison is with the work of Schmid [14], who shows precision vs. recall for an automatically constructed model of a zebra (and cheetah). Comparing performance is hard because the test set is different, Schmid does not show error bars, and selects negative training images by hand. In particular (1) zebras co-occur frequently with grass backgrounds in the training set, and so Schmid uses negative training images with grass to disambiguate the two; and (2) the negative training images reflect the statistics of the negative test images. They include cheetahs and other animals. Our model building algorithm by itself does not require any negative training samples. The texture models we add to the detectors need a baseline of "non-zebra" textures, so we obtain a negative training set by *randomly* selecting 100 Corel images (not in our test set). We seem to obtain a comparable zebra detector; this is in part because temporal coherence naturally segments out the zebra, circumventing the first issue. Results for tigers and giraffes were not shown in [14].

5. Discussion

We have shown there exists a strong analogy between tracking and object detection; both involve exercises in model building.

Tracking: This paper has shown that the constant appearance model introduced in [13] can be used to learn spatial models on-the-fly. The ability to create a data-driven spatial model might prove useful for tracking people wearing long skirts or other difficult clothing.

We demonstrate that constant state models can be useful for tracking. We show an approximate inference procedure for a constant model which naturally recasts tracking as *model building* (inferring a MAP estimate of appearance) followed by *detection* (using the model to infer a MAP estimate of segment configuration).

Object Detection: This paper has also demonstrated that videos are a useful source of data for the unsupervised learning of object models. By learning with videos as opposed to image collections [14, 16], we exploit the additional constraint that object parts cannot move too fast from image to image. This constraint helps address one notable difficulty with unsupervised learning – disambiguating objects that co-occur frequently [4]. We can disambiguate zebra and grass image patches using motion constraints.

Our temporal coherence criterion also produces object models with appealing spatial qualities. Building a kinematic model for a giraffe is difficult (even given a collection of giraffe pictures with the background masked out) because a giraffe deforms non-rigidly. Our algorithm suggests a good kinematic model is one where the segments appear in a lot of giraffe pictures. This is why one should

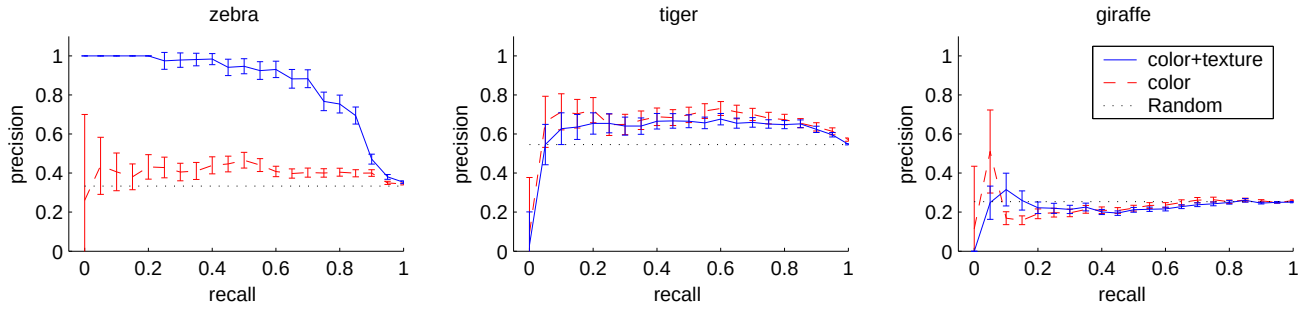

Figure 6. *Our activities could be viewed as automatic production of segment models using the implicit supervisory information in temporal coherence of appearance. In this view, our work is successful, too. We took the animal models (consisting of an appearance model for each segment and a spatial model between the segments) built during the tracking phase and used them to match to images of the relevant animals from Corel. Our testing pool consists of 100 random (not labeled with "zebra", "tiger" or "giraffe") plus 50 "zebra", 34 "giraffe", and 120 "tiger" images. We show precision/recall curves for 2 types of detectors, comparing them with a baseline of random guessing (the black dotted line). We plot error bars representing the standard deviation in precision for a given recall rate by repeatedly sampling (with replacement) 100 negative images from our image pool when testing each detector. In the red (dashed) line, we model segment appearance with a color histogram, the same representation used for clustering. Since our algorithm identifies pools of animal pixels in each video, we can learn more sophisticated appearance models. The blue (solid) line is a detector which adds a texture descriptor to each segment feature vector. We use 100 random Corel images (not in our test set) as negative examples to train the descriptors. Our zebra detector performs significantly better when texture is used. Our tiger detectors both perform moderately well and near equivalent to each other, indicating color is sufficient for detection. Our giraffe detectors are poor, performing at chance. Even though our giraffe shape model seems accurate (as suggested by figure 2), the video is washed out and generates a poor appearance model. To build good detectors, we need video with resolution and spatial frequency content similar to our test images. The large error bars in all cases indicate performance can heavily depend on the negative test pool, implying comparisons of experiments on different data sets [14] need to made cautiously. Our zebra detector performance seems to be equivalent to [14], without the need for hand picked negative training images.*

model the giraffe neck as three segments rather than one; some of the giraffe pictures have a deformed neck.

Acknowledgments

This work was supported by NSF award no. 0098682 and by Office of Naval Research grant no. N00014-00-1-0890, as part of the MURI program. D. R. is supported by a NSF fellowship.

References

[1] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

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

[3] J. Coughlan and S.J. Ferreira. Finding deformable shapes using loopy belief propogation. In *Proc ECCV*, 2002.

[4] P. Duygulu, K. Barnard, N. de Freitas, and D.A. Forsyth. Object recognition as machine translation. In *Proc. European Conference on Computer Vision*, pages IV: 97–112, 2002.

[5] P. Felzenschwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *Proc CVPR*, 2000.

[6] D.A. Forsyth and J. Ponce. Tracking with non-linear dynamic models, online chapter *http://www.cs.berkeley.edu/∼daf/bookpages/pdf/*.

[7] D.A. Forsyth and J. Ponce. *Computer Vision: a modern approach*. Prentice-Hall, 2002.

[8] S. Ioffe and D. Forsyth. Human tracking with mixtures of trees. In *Int. Conf. on Computer Vision*, 2001.

[9] O. Maron. *Learning from Ambiguity*. PhD thesis, MIT, 1998.

[10] O. Maron and A.L. Ratan. Multiple-instance learning for natural scene classification. In *The Fifteenth International Conference on Machine Learning*, 1998.

[11] I. Dan Melamed. *Empirical Methods for Exploiting Parallel Texts*. MIT Press, 2001.

[12] M.Wainwright, T. Jaakola, and A.Willsky. Tree-based reparameterization for approximate inference on loopy graphs. In *NIPS*, 2001.

[13] D. Ramanan and D.A. Forsyth. Finding and tracking people from the bottom up. In *Proc CVPR*, 2003.

[14] C. Schmid. Constructing models for content-based image retrieval. In *Proc CVPR*, 2001.

[15] H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *European Conference on Computer Vision*, 2000.

[16] Markus Weber, Max Welling, and Pietro Perona. Unsupervised learning of models for recognition. In *ECCV (1)*, pages 18–32, 2000.