Dynamic Processing Allocation in Video

Daozheng Chen, *Student Member, IEEE,* Mustafa Bilgic, Lise Getoor, *Member, IEEE,* and David Jacobs, *Member, IEEE,*

Abstract—Large stores of digital video pose severe computational challenges to existing video analysis algorithms. In applying these algorithms, users must often trade-off processing speed for accuracy, as many sophisticated and effective algorithms require large computational resources that make it impractical to apply them throughout long videos. One can save considerable effort by applying these expensive algorithms sparingly, directing their application using the results of more limited processing. We show how to do this for retrospective video analysis by modeling a video using a chain graphical model and performing inference both to analyze the video and to direct processing. We apply our method to problems in background subtraction and face detection, and show in experiments that this leads to significant improvements over baseline algorithms.

Index Terms—Video processing, resource allocation, graphical models, optimization, background subtraction, face detection, dynamic programming.

1 Introduction

New technology is giving rise to large stores of digital video. Their size has increased much faster than the computational resources needed to effectively process them. At the same time, as we develop increasingly sophisticated and accurate vision algorithms, they also demand greater computational resources. Consequently, it is important to develop strategies for applying vision algorithms with greater efficiency to video data.

The scope of the problems we face is evident. Surveillance systems can contain thousands of cameras. Real time processing of huge data sets is extremely challenging; retrospective or forensic analysis creates even greater problems when one must rapidly examine hours or days of video from thousands of cameras. For example, British police were required to examine 80,000 CCTV tapes from a network of 25,000 cameras [1] to discover the image of a bomber after the terrorist attack in London in 2005 [2]. Automatic processing is needed to speed up this analysis, but one cannot hope to process all available video in such cases; it is essential to direct processing to portions of video most likely to be informative.

In this paper, we develop a new method for controlling processing, so that available resources are directed at the most relevant portions of the video. In our proposed approach, we initially perform some inexpensive processing of a video by applying a cheap but less accurate algorithm combined with sparse application of a more expensive and accurate algorithm. We then use an inference algorithm to determine to which frames we should apply further expensive processing.

- *D. Chen, L. Getoor, and D. Jacobs are with the Department of Computer Science and the Institute for Advanced Computer Studies (UMIACS), University of Maryland, College Park, College Park, MD 20742. E-mail: {dchen, getoor, djacobs}@cs.umd.edu*
- *M. Bilgic is with the Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616. E-mail: mbilgic@iit.edu*

Our work makes two critical assumptions. First, that expensive algorithms exist that can perform a task quite accurately (e.g., >90% accuracy). While many real-world vision tasks are still too challenging for this, recent growth in the number of vision companies and applications illustrate that high accuracy is often achievable in simpler tasks (eg., face detection in cameras [3]) or in controlled environments (eg., detection and tracking in stores[4]). Moreover, in vision systems with a human in the loop, a human analyst may be regarded as a very accurate, and very expensive algorithm. Second, we assume that a much cheaper, but less accurate algorithm is available, and that it is desirable to use the output of this algorithm to direct the attention of the expensive algorithm most profitably. We want to stress that our work does not aim to solve vision problems that are beyond the reach of existing algorithms, but rather to speed up the solution to problems that are currently solvable, albeit only at considerable cost.

To combine information from features produced by cheap and expensive algorithms, we present a graphical model for video analysis. We use a second-order Markov model with a node for each frame, and a state variable that indicates whether this frame is relevant to a query. For example, the state might indicate whether the frame contains a visible face. Each state has two potential observations. The first observation is always given; it is obtained by running a cheap algorithm on all frames. For example, cheap background subtraction might provide a clue as to whether people are currently visible. The second observations is only obtained if a more expensive and accurate algorithm is applied to that frame (in this example, a face detector). As in a Hidden Markov Model (HMM), each observation directly depends on the current state. In addition, in our model each observation directly depends on the previous observation. This captures the phenomenon that errors made by an algorithm are often correlated from one state

to the next. This model allows us to effectively combine information from cheap and expensive algorithms to improve performance.

Our primary contribution is a new algorithm that uses this model to determine where in a video to apply the expensive algorithm. We build on prior work by Krause and Guestrin [5] that shows that one can use a dynamic programming algorithm to determine the optimal places at which to make observations in a first-order Markov chain. While this work is readily extended to our graphical model, it requires $\Theta(B^2n^3)$ computation time, where n is the number of nodes in the Markov chain, and B is the number of places at which we will apply the expensive algorithm. In our setting n is the number of frames in the video and B is also $\mathrm{O}(n)$, so this algorithm is not practical for video analysis.

We solve this problem with a new algorithm that produces an approximately optimal answer efficiently. More precisely, we make an additional assumption about the concavity of the reward from observations as the budget increases, a law of diminishing returns that we show is generally valid in our setting. Then, we show that by applying part of the total budget to make observations at a uniform step size, we can find an allocation of the remaining observations that will be at least as good as the optimal batch allocation, and that requires a modest amount of computation. This allocation makes use of rewards computed by Krause and Guestrin's algorithm, applied to small sections of the video.

Our approach is quite general, and can be applied to a wide range of scenarios in which multiple algorithms are combined into a single system. Our final contribution is to experimentally demonstrate the value of this algorithm in two very different vision tasks: motion and face detection. To detect motion efficiently we combine a very cheap and a more expensive background subtraction algorithm. For the second task, we use background subtraction to trigger face detection. We show that our algorithm can be used to significantly improve performance.

The paper is organized as follows. In Section 2 we discuss related work. In particular, we describe a dynamic programming algorithm [5] that determines the optimal place to make observations. We then describe our new algorithm in the context of Markov Chains in Section 3. Then, we introduce our graphical model for video analysis and describe how to apply the new algorithm to this model in Section 4. In Section 5, we show experiments.

2 Prior Work

We first review background subtraction and face detection algorithms. Next, we describe work on visual computing that deals with issues of resource constraint. We then discuss work on feature and label acquisition. Finally, we describe the algorithm by Krause and Guestrin [5] in detail.

2.1 Background Subtraction

Background subtraction detects moving objects in video, usually taken by static cameras. This typically involves building a background model. Wren et al. [6] use a Gaussian estimate of the background distribution of each pixel. Lo and Velastin [7] use the median of the previous n frames as the background model. Elgammal et al. [8] propose a non-parametric model based on kernel density estimation to approximate the pdf of each pixel. In Kim et al. [9], background values are quantized into codebooks to handle periodic motion. Rittscher et al. [10] represent the background using a hidden Markov model, which can discriminate between foreground, background, and shadow. Cheung et al. [11], Piccardi [12], and Yilmaz [13] give a general review of this problem. We describe in more detail two methods we use in our experiments.

In frame differencing (FD), the background model of the frame at time t, f_t, is the frame in the previous time step, f_{t-1} (Jain and Nagel [14]). Given a threshold, Th, $|f_t - f_{t-1}| > Th$ gives the foreground region of f_t.

Stauffer and Grimson [15] use a mixture of Gaussians (MoG) for the background model. With the assumption that a more compact distribution with a higher mode is more likely to be the background, MoG selects background components whose ratio between its peak value and standard deviation is greater than a certain threshold. Finally, it uses recent pixel values to update the model parameters. This method is much more sophisticated than the FD method, and requires significantly more computational resources. Zivkovic [16] improves this work by using recursive equations that can also simultaneously select the appropriate number of components in the mixture model. We call this method the improved adaptive Gaussian mixture model (IAGMM), and we use it in our experiments.

2.2 Face Detection

Yang et al. [17] provide a comprehensive survey of face detection methods and organize them into four major categories. First, top-down knowledge based methods represent a face using human knowledge, which usually captures relationships between facial features such as eyes, nose, and mouth. Yang and Huang [18], for example, use a hierarchical knowledge-based method to detect faces. Second, bottom-up feature based methods seek invariant features, such as eyebrows, hair texture, and skin color, for detection. Hsu et al. [19] propose a skin-tone color model which they use to generate face candidates for verification by facial features (see also Jones and Rehg [20]). The third category is template based methods, in which correlation between an input image and the template is used to detect faces. Sinha [21], for example, builds a face template by capturing the invariance between the relative brightness of facial regions. The last category includes appearance based methods, which in general use statistical analysis and

machine learning techniques. Various methods, such as support vector machines [22] and neural networks [23], and naive Bayesian classifiers [24] have been proposed.

Viola and Jones [25] achieve a breakthrough in performance that has been widely adopted. With integral images for fast computation, their scheme uses a set of features that are similar to Haar wavelets. They then construct classifiers by selecting a small number of important features using Adaboost. Finally, the scheme detects faces inside an image region by applying classifiers in a cascade. At each level of the cascade, one uses a classifier with a very low false negative rate, although the false positive rate might be high. Subsequent classifiers are run only when previous classifiers indicate a positive result. Lienhart and Maydt [26] extend this work by adding an efficient set of 45° rotated features to the original feature set and by using a new post-optimization procedure for a given boosted classifier. Their work shows significantly lower false alarm rates, and we use this method to detect faces in our experiment.

There have been many other extensions to the Viola-Jones method. For example, Huang et al. [27] extend the cascade of classifiers structure to a Width-First-Search (WFS) tree structure. Mita et al. [28] introduce a new feature, called the joint Haar-like feature, for detection. Xiao et al. [29] use a boosting chain to integrate historical knowledge of successive learning of strong classifiers.

2.3 Visual Computing under Resource Constraints

Many methods have been developed to handle resource constraints in computer vision. Weiss and Taskar [30] generalize the approach of Viola and Jones and apply it to a range of applications, including handwriting recognition. Felzenszwalb et al.[31] develop cascades for object detection using deformable models such as pictorial structures. Vijayanarasimhan [32] recently introduce a novel framework for object detection and classification in still images under resource constraints. They design a grid based model that is used to determine the best image regions to look at and the best features to be extracted. This process is guided by the principle of value-of-information (VOI) to find the most evidence at the least cost.

In video processing, performance is often an issue, as many effective algorithms are too slow to run in real-time, and even fast algorithms may require enormous amounts of time when used to perform retrospective analysis of large quantities of video. One common strategy is to run cheap, lower level algorithms such as motion detectors to determine when something interesting might be happening. When these detect motion, higher level algorithms are then deployed. While this approach is used heuristically, but very effectively in a wide range of applications, we will mention two representative works that formalizes this. First, Krishna et al. [33] propose an algorithm switching approach to handle background subtraction. The system starts by processing each frame with a uni-modal model. When the system shows poor segmentation quality, it switches to use the MoG model. Second, Barotti et al. [34] use algorithm switching to handle lighting changes and solve bootstrapping problems in motion detection. When the system detects sudden global illumination variation, the motion detection switches from background subtraction using a single Gaussian to FD.

2.4 Feature and Label Acquisition

The machine learning community has looked at the problem of determining which features to acquire in order to correctly classify instances. In this setup, instances are described by a set of features each of which has an associated acquisition cost, and a total budget limits feature acquisition. Some example strategies are [35], [36]. The biggest difference between this line of work and ours is that the feature-value acquisition community treats each instance as independent. However, in our case, the information we want to extract in nearby frames of video is highly correlated, and we should be able to do better if we take these correlations into account.

Another related area of work is label acquisition: instead of obtaining features, we can query an oracle to determine an instance's label directly. Given a network of instances, such as a sequence of frames, a network of friends, etc, acquiring the label for an instance helps in correctly classifying the rest of the network. The question is then which instances should be queried in order to get the best performance on the remaining ones. Rattigan et al. [37] queries the instances that are structurally important, e.g. highly connected instances, central instances, etc. Bilgic and Getoor [38] build a classifier that can predict which instances might be misclassified and query a central instance only if it is predicted as misclassified. Active learning work [39] is very related to this problem, and it has been applied to visual recognition [40], [41]. However, active learning acquires labels to construct training data to learn a model, whereas label acquisition described here is applied to an already learned model to guide probabilistic inference.

These methods have been quite successful in practice, but they are heuristic approaches and have no theoretical guarantees (partly because they are applicable to general networks). However, for the class of chain graphical models such as HMMs, Krause and Guestrin [5] show how to solve the label acquisition problem optimally. We describe their work next.

2.5 Optimal Observation Plans

Krause and Guestrin [5] present VoIDP, an Dynamic Programming to optimize Value of Information, for selecting observations for the class of chain graphical models. Since we build on this method, we now describe it in some detail, although we consider a special case of their work that is suitable to our problem.

They optimize an objective function based on a class of reward functions, R, that are defined using the probability distribution of a set of random variables $S = \{X_1, \cdots, X_n\}$. This set of variables forms a chain graphical model, that is, $i < j < k$, implies that X_i is conditionally independent of X_k given X_j. For example, consider a HMM unrolled for n time slices. Then the n hidden state variables form a chain graphical model. Suppose that for each of these variables, it is possible to observe its hidden state at a fixed cost. This corresponds, in our problem, to the supposition that an expensive algorithm is extremely accurate, and reveals the hidden state. Let O be the set of observed variables and o be the values of these variables. $O = o$ is used to denote each variable in O takes its corresponding value in o.

The reward function R is built upon a local reward R_j, which is a functional on the probability distribution $P(X_j|O = o)$. While this reward could be quite general, in this paper we consider only X_j that are binary variables, and use

$$\begin{aligned} R_j(P(X_j|O=o)) = \\ max(P(X_j=1|O=o), P(X_j=0|O=o)) - \\ min(P(X_j=1|O=o), P(X_j=0|O=o)) \end{aligned} \quad (1)$$

That is, given a set of observations, we receive a greater reward as we become more certain of the value of each state. This reward is equivalent to considering the expected number of correct classifications. We will also use the notation:

$$R_j(X_j|O) \triangleq \sum_o P(O=o) R_j(P(X_j|O=o)), \quad (2)$$

That is, given the choice of a set of variables, O, to observe, $R_j(X_j|O)$ denotes the expected reward we will receive from these observations.

Assume that there is a fixed budget B for selecting observations, we then must select observations O to

$$\begin{aligned} &\text{maximize } J(O) = R(O) = \sum_j R_j(X_j|O), \\ &\text{subject to } \|O\| = b \leq B, \end{aligned} \quad (3)$$

where j is the index over the state variables S, b is the number of observed state variables O, and B is the total budget for the whole chain. Observations can include variables at any time step in the chain since we consider processing a video after it is recorded. This corresponds to the "smoothing" version of the problem [5].

The conditional independence property in the chain graphical model simplifies the local reward. With this property, the local reward $R(X_j|O) = 1$ in the case that $X_j \in O$. In the case that $X_j \notin O$, we have $R(X_j|O) = R(X_j|O_j)$, where O_j is a subset of O containing two observations. These are the last observation preceding X_j and the first observation in O that follows X_j.

Furthermore, Krause and Guestrin [5] consider both a conditional planning setting of this problem, in which the best observation is made and then the optimal next observation is computed, and this is repeated k times, and a subset selection setting of the problem, in which one decides on the locations of the best observations with a total cost of k first, and then makes these observations. Our algorithm uses the conditional planning variant since it in general produces the best performance.

They solve this problem by noting that once an observation is made, it splits the problem of determining future observations into conditionally independent components before and after the observations. This allows for a dynamic programming solution. They define a value, $J_{a:b}(x_a, x_b; k)$, which denotes the reward produced by the optimal plan with a budget of k over the interval from variables X_a to X_b, given that these variables have been observed to have states x_a and x_b. Then they note that $J_{a:b}(x_a, x_b; k)$ can be recursively computed given the value of $J_{c:d}(x_c, x_d; l)$ for all $a \leq c \leq d \leq b$ and $l < k$. The recursive formula is

$$\begin{aligned} J_{a:b}(x_a, x_b; k) = \max\{J_{a:b}(x_a, x_b; 0), \max_{a<j<b}\{ \\ \sum_{x_j} P(X_j = x_j | X_a = x_a, X_b = x_b)\{R_j(X_j|X_j = x_j) + \\ \max_{0 \leq l \leq k-1} [J_{a:j}(x_a, x_j; l) + J_{j:b}(x_j, x_b; k-l-1)]\}\}\}, \end{aligned} \quad (4)$$

where the base case is

$$J_{a:b}(x_a, x_b; 0) = \sum_{j=a+1}^{b-1} R_j(X_j|X_a = x_a, X_b = x_b). \quad (5)$$

The recursive formula basically iterates over each split point j between a and b to find one that returns the highest reward (or performs no further observations if they do not increase the reward). For each split point, the reward is the expectation taken over all possible assignments of value to the split point. All possible budget allocations between the two split subsequences are considered when the value of split point is fixed.

To initialize, the algorithm adds two independent dummy variables, X_0 and X_{n+1}, which have no reward and observation cost but have default states, to the head and tail positions of the chain. Thus the optimal reward for a chain of n variables with a budget of B is computed as $J_{0:n+1}(x_0, x_{n+1}; B)$.

This algorithm corresponds to the smoothing version of VoIDP in a conditional plan setting, and we refer it as VoIDP-SCP in this paper. According to Theorem 2 in [5], the complexity of this algorithm in terms of number of evaluations of local rewards for our binary state variables is

$$\Theta(B^2 n^3). \quad (6)$$

3 Efficient Observation Plans

We now present a novel algorithm, Dynamic Progamming Allocation (DPA), that is efficient enough to apply to problems with very large values of n. DPA approximates the optimal algorithm and is much faster. In the next sections, we will show how this algorithm

can be applied to video processing. DPA first uses B' observations from the total budget, B, to make uniform observations. This splits the Markov chain into $M = B' + 1$ consecutive intervals where the first and last variables of each interval are observed. This breaks our problem up into a series of smaller problems of the same size. These problems are not independent, however, since the remaining $B'' = B - B'$ budget must be parceled out between all these intervals. We show that, with an additional, reasonable assumption, this can be done optimally. Once the budget is allocated to intervals, observations can be allocated within intervals using VoIDP-SCP.

DPA maximizes the sum of rewards over all intervals to allocate budget between them. That is, let k_i be the budget allocated to interval i. Let $J_i(k_i)$ be the reward of VoIDP-SCP for interval i with a budget of k_i. We want to find $k_1, k_2, \cdots, k_M$ such that they

$$\text{maximize } \sum_{i=1}^{M} J_i(k_i), \text{ subject to } \sum_{i=1}^{M} k_i = B''. \quad (7)$$

To perform this optimization efficiently, we rely on the empirical observation that for each interval, the optimal reward typically forms a concave curve as the budget increases. That is, the plot of $J_i(k_i)$ against k_i as k_i increases is concave in general. This is a kind of law of diminishing returns property. We denote these kinds of curves as Reward-Budget (RB) curves, and the assumption that these curves are concave will allow us to optimally allocate our budget. We will experimentally verify that this assumption is reasonable. Given this assumption we can compute the k_i with our proposed algorithm, Dynamic Progamming Allocation (DPA).

Dynamic Processing Allocation (DPA)

1) Use B' observations to make uniform observations to break the chain model into consecutive disjoint intervals separated by the observed variables;
2) Allocate the remaining budget B'' to these intervals. No observation is taken in this step;
3) Use VoIDP-SCP to determine the observation locations within each interval; observations are taken in a conditional mode.

It remains to describe how we perform the second step of this algorithm. Let N_i be the maximum possible budget for interval i, typically the number of unobserved state variables in the interval. We first compute the reward curve for each subsection up to its maximum budget. That is, we compute $J_i(k)$, for $1 \leq k \leq N_i$, using VoIDP-SCP. Next, we define the reward increment as

$$\Delta J_i(k) \equiv J_i(k) - J_i(k-1), \quad (8)$$

where $k = 1, \cdots, N_i$. Assuming concavity, we have

$$\Delta J_i(k) \leq \Delta J_i(k-1) \quad (9)$$

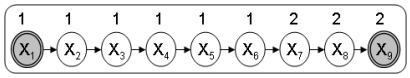


Fig. 1: The example chain graphical model. We assume that X_1 and X_9 are observed in advance, shown with gray shading. The state of a node can be 1 or 2, and we show the actual state on the top of each node.

for all ks. Next, we sort all the reward increments in descending order. Finally, the budget for each interval is set to the number of increments it has in the top B'' positions of the sorted list. We call this allocation method batch budget allocation. Intuitively, we can see that this always assigns observations to the sequences where they will create the most incremental benefit. We prove this in Section 3.2 and discuss the complexity of the algorithm in Section 3.3.

We expect this algorithm to do much better than an optimal batch algorithm, since the B' observations we use to break the problem into intervals also provide very useful information, and because we can use an optimal conditional plan within each interval, which can be much better than the optimal batch plan.

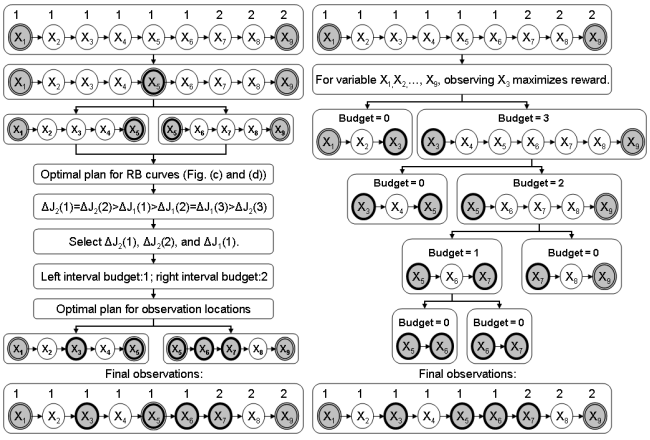
However, we note that it is possible to improve the running time of budget allocation at the cost of some additional memory. This is because DPA requires us to compute $J_i(k)$ for all k, while in practice, most intervals are allocated small budgets, and we only need to compute the RB curve up to this budget. This leads to the algorithm incremental budget allocation, which performs step 2 of DPA a bit differently. This method returns the same output as the batch allocation method. In section 3.3, we show that it is asymptotically faster, especially when we have a small budget. In addition, when the subsection size is large, we can avoid computing the RB curve up to its maximum budget for each subsection, this will save a significant amount of processing time. We use DPA with incremental budget allocation in our experiments.

Incremental Budget Allocation - Step 2

1) Initialize the budget of each interval to be zero;
2) Compute the reward increment $\Delta J_i(1)$ for $i = 1, 2, \cdots, M$;
3) Select the highest increment, and add one to the budget of the corresponding interval I;
4) If the total budget, B'', has been used, terminate and use the current budget allocation for each interval as the final budget allocation;
5) If not, compute the next reward increment for interval I, and use it to replace the current reward increment for this interval. Go back to step 4.

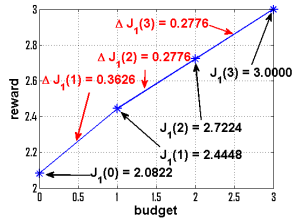
3.1 An Example

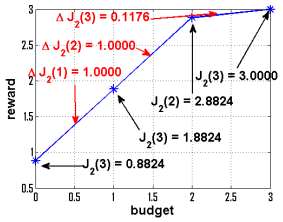
We use an example to illustrate how DPA works and how VoIDP-SCP is different. Consider a chain graphical



(a) DPA with batch budget allocation

(b) VoIDP-SCP





(c) RB curve - interval 1 (left interval)

(d) RB curve - interval 2 (right interval)

Fig. 2: (a) shows how DPA with batch budget allocation determines the observation locations. We use gray to highlight observed nodes. The interval size is 5, so the initial observation is X_5, which is highlighted by a double-line boundary with a thick inner line. (c) and (d) show the RB curves for left and right intervals respectively. The observation locations for each of these two intervals are determined by VoIDP-SCP, and are highlighted by a single-line thick boundary. (b) shows how VoIDP-SCP determines observations locations, which are also highlighted by a single-line thick boundary. It shows how the chain is split into sub-chains by sequential observations.

model with nine state variables whose value can be either 1 or 2. The prior probability of being in each state is 0.5. The transition probability of switching from one state to the other is 0.2. Figure 1 shows this model and displays one set of states generated by it on the top of each variable. For simplicity, we assume the first and last states are known in advance. Fig. 2 shows how DPA and VoIDP-SCP determine the observation locations with a budget of 4.

First, we note that for this example, it is most likely that the initial states have a value of 1, and that at some point in the chain there is a single transition from 1 to 2. To correctly determine the state values, the main goal is to find the location of this transition. It is also possible that there are really three transitions, and a secondary goal will be to check on that. Next, we note that VoIDP-SCP is able to perform a binary search to find such transitions. It turns out that the optimal strategy for this situation involves first determining the value of X_3. If this state is 1, then the remaining budget is sufficient to allow a binary search to be performed on states X_4-X_8, to find the transition from 1 to 2. This strategy therefore guarantees that the transition from 1 to 2 will be found, and maximizes the chances that any additional transitions will be found.

Suppose instead we run DPA, with a budget of 4, and $B' = 1$. The algorithm begins by determining the state of X_5, to break the problem into two equal parts. When this state is found to be 1, the algorithm then allocates its budget between these two subsequences. To allocate the remaining budget to each interval, it computes the RB curves up to its maximum budget for both intervals, as shown in Fig. 2(c) and 2(d). Notice that both curves satisfy the concavity property. In addition, the reward increment under a small budget for the right interval is higher than those for the left interval. This is because the state of the first and last nodes for the right interval indicate a state transition. As a result, the right interval obtains a higher budget. In fact, DPA allocates two observations to the right side of the chain, which is enough to perform a binary search for the state transition, and one observation to the left side. This final observation on the left side is more likely to find something interesting than if allocated to the right side, once the binary search has occurred.

We now consider how incremental budget allocation works in this example. After the observation of X_5, the chain is split into two intervals as shown in Fig. 2(a) and the remaining budget becomes 3. The incremental algorithm then initializes the budget for both intervals to be 0 and computes $J_1(0), J_1(1), J_2(0)$, and $J_2(1)$ for $\Delta J_1(1)$ and $\Delta J_2(1)$. Since $\Delta J_1(1) = 0.3626 < \Delta J_2(1) = 1.0000$, it increases the budget for the right interval by 1. The remaining budget becomes 2 and it computes $J_2(2)$ for $\Delta J_2(2)$. It again increases the budget for the right interval by 1 because $\Delta J_1(1) < \Delta J_2(2) = 1.0000$. The remaining budget becomes 1 and it computes $J_2(3)$ for $\Delta J_2(3)$. After this, because $\Delta J_1(1) > \Delta J_2(3) = 0.1176$, it increases the budget for the left interval by 1. All the budget has been allocated, and the algorithm terminates with a budget of 1 for the left interval and 2 for the right interval. Notice that compared with batch allocation, incremental allocation does not compute $J_1(2)$ and $J_1(3)$.

DPA is not optimal in two ways. First, an optimal set of observations may not include X_5. Second, allocating one observation to the left subsequence and two to the right subsequence may not be optimal; future observations could determine that a different allocation would be better. On the other hand, in VoIDP-SCP shown in Fig. 2(b), the initial observation X_3 is determined after computation and comparison of the expected reward of observing $X_1, \cdots, X_9$ with all possible budget distributions. This is a considerable amount of computation. In DPA, this interval is split into two shorter intervals, and reward can be more cheaply computed for each subchain separately.

3.2 Proof of Correctness

We now provide a proof that the batch budget allocation is correct based on the problem formulation in (7); it follows directly that the incremental budget allocation must also be correct.

We use the following new notations, definitions, and facts. With a budget of B, we let $\hat{k}_1^B, \hat{k}_2^B, \cdots, \hat{k}_M^B$ be an optimal budget allocation, where M indicates the number of subsequences among which we must divide the budget. Let $\bar{k}_1^B, \bar{k}_2^B, \cdots, \bar{k}_M^B$ be the budget allocation by the algorithm. Since the algorithm picks only the top B reward increments from Z to distribute budget, it is clear that $\sum_{i=1}^M \bar{k}_i^B = B$. The following theorem proves the correctness of the allocation algorithm by showing that summation of reward from each subsection under the batch budget budget allocation is equal to that under the optimal budget allocation.

Theorem 1:

$$\sum_{i=1}^{M} J_i(\bar{k}_i^B) = \sum_{i=1}^{M} J_i(\hat{k}_i^B). \tag{10}$$

To prove Theorem 1, we introduce Lemma 1 and Lemma 2. Lemma 1 shows that the sum of all reward increments for each subsection using batch budget allocation is equal to the sum of the top B reward increments in Z. Lemma 2 proves the the sum of all reward increments for each subsection under the optimal allocation cannot be greater than that under batch allocation. Finally, we prove Theorem 1 by adding the reward increment to the reward with zero budget for each subsection to establish the equality.

Denote the sorted list of rewards gained by an additional observation as Z and the sum of the top B reward increments in Z as ΔL. In addition, we define $\Delta J_i(0) \equiv 0$ to handle the case that some interval has a budget of 0. Then we define $\Delta \hat{J}_i^B \equiv \sum_{j=0}^{\hat{k}_i^B} \Delta J_i(j)$, which means $\Delta \hat{J}_i^B$ is the sum of all reward increments for interval i with a budget of $\hat{k}_i^B$. Similarly, we define $\Delta \bar{J}_i^B \equiv \sum_{j=0}^{\bar{k}_i^B} \Delta J_i(j)$.

Lemma 1:

$$\sum_{i=1}^{M} \Delta \bar{J}_i^B = \Delta L. \tag{11}$$

Proof: For an interval i, by the procedure of the algorithm, there must be $\bar{k}_i^B$ reward increments from interval i in ΔL. Let the sum of these increments be ΔL_i. In case that no such increment exists, we let $\Delta L_i \equiv 0$. Suppose $\Delta \bar{J}_i^B \neq \Delta L_i$. By concavity, we know that ΔL_i must include some reward increment $\Delta J_i(x)$ such that $x > \bar{k}_i^B$ and $\Delta J_i(x) < \Delta J_i(\bar{k}_i^B) \leq \Delta J_i(\bar{k}_i^B - 1) \leq \cdots \leq \Delta J_i(1)$. Thus, there must be at least $\bar{k}_i^B + 1$ reward increments in the top B positions from interval i. But this conflicts with the fact that the top B positions only contain $\bar{k}_i^B$ such increments. Thus, it can only be that $\Delta \bar{J}_i^B = \Delta L_i$. Finally, because $\sum_{i=1}^M \bar{k}_i^B = B$, we have $\sum_{i=1}^M \Delta L_i = \Delta L$. Therefore, $\sum_{i=1}^M \Delta \bar{J}_i^B = \sum_{i=1}^M \Delta L_i = \Delta L$. □

Lemma 2:

$$\sum_{i=1}^{M} \Delta \bar{J}_i^B \geq \sum_{i=1}^{M} \Delta \hat{J}_i^B. \tag{12}$$

Proof: Any reward increment not in the top B positions of Z must be less than or equal to any reward increment in the top B positions because Z is sorted. So by Lemma 1, $\sum_{i=1}^M \Delta \bar{J}_i^B = \Delta L$ must be greater than or equal to the sum of any B reward increments from Z. Because $\sum_{i=1}^M \hat{k}_i^B = B$ and Z contains all reward increments from all intervals, we know that $\sum_{i=1}^M \Delta \hat{J}_i^B$ is also the sum of B reward increments from Z. Therefore, $\sum_{i=1}^M \Delta \bar{J}_i^B \geq \sum_{i=1}^M \Delta \hat{J}_i^B$. □

By the definition of $\Delta J_i(k)$, we know that $J_i(k) = \Delta J_i(k) + J_i(k-1)$. Using induction, it is trivial to show that $J_i(k) = \sum_{j=1}^k \Delta J_i(j) + J_i(0)$. Because we define $\Delta J_i(0) \equiv 0$, then

$$J_i(k) = \sum_{j=0}^{k} \Delta J_i(j) + J_i(0). \tag{13}$$

With this formula, we can finally prove Theorem 1.

Proof: By equation (13), we have

$$\begin{aligned}\sum_{i=1}^{M} J_i(\bar{k}_i^B) &= \sum_{i=1}^{M} [\sum_{j=0}^{\bar{k}_i^B} \Delta J_i(j) + J_i(0)] \\ &= \sum_{i=1}^{M} [\Delta \bar{J}_i^B + J_i(0)] = \sum_{i=1}^{M} \Delta \bar{J}_i^B + \sum_{i=1}^{M} J_i(0). \end{aligned} \tag{14}$$

Similarly,

$$\sum_{i=1}^{M} J_i(\hat{k}_i^B) = \sum_{i=1}^{M} \Delta \hat{J}_i^B + \sum_{i=1}^{M} J_i(0). \tag{15}$$

Then by lemma 2, it follows that $\sum_{i=1}^M J_i(\bar{k}_i^B) \geq \sum_{i=1}^M J_i(\hat{k}_i^B)$. Finally, because the budget allocation, $\hat{k}_1^B, \hat{k}_2^B, \cdots, \hat{k}_M^B$, is optimal, we have $\sum_{i=1}^M J_i(\bar{k}_i^B) = \sum_{i=1}^M J_i(\hat{k}_i^B)$. □

3.3 Complexity Analysis

We now determine the complexity of DPA in terms of the number of local reward evaluations.

Theorem 2: With batch budget allocation, the number of local reward evaluations computed with DPA is:

$$\mathrm{O}\big(\frac{1}{\epsilon^4} n\big), \tag{16}$$

and with incremental budget allocation, it is

$$\mathrm{O}\big(\frac{1}{\epsilon^4} B\big), \tag{17}$$

where ϵ is a number less than 1 such that $\epsilon = \frac{M}{n}$.
The inverse of ϵ reflects the subsection length. This complexity is a vast improvement over $\Theta(B^2 n^3)$ for VoIDP-SCP. A brief outline of the proof is as follows.

Proof: Let the total budget be $B = B' + B''$, where B'' is the remaining budget allocated to each subsection after the initial uniform sampling with a budget of B'.

The proof with batch allocation is straight forward. In the budget allocation stage, VoIDP-SCP is run with a budget up to the number of unobserved state variables, which is $\mathrm{O}(\frac{1}{\epsilon})$ for each subsection. Step 1 of the algorithm requires no computation of rewards, and Step 3 can reuse the rewards computed in Step 2. With $M = \epsilon n$ and using the complexity of VoIDP-SCP, we have the complexity of DPA as:

$$\mathrm{O}\big(M \cdot \mathrm{O}\big(\mathrm{O}\big(\frac{1}{\epsilon}\big)^2 \cdot (\frac{1}{\epsilon})^3\big)\big) = \mathrm{O}\big(\epsilon n (\frac{1}{\epsilon})^2 (\frac{1}{\epsilon})^3\big) = \mathrm{O}\big(\frac{1}{\epsilon^4} n\big) \tag{18}$$

For incremental budget allocation, the complexity is similarly determined by the budget allocation step. However, VoIDP-SCP in this case is run on each subsection with a budget up to its allocated budget plus one. For analysis, we can ignore the constant one and the complexity is

$$\mathrm{O}\big(\sum_{j=1}^{M} (\tilde{k}_j^2 (\frac{1}{\epsilon})^3)\big) = \mathrm{O}\big(\frac{1}{\epsilon^3} \sum_{j=1}^{M} \tilde{k}_j^2\big) \tag{19}$$

where $\tilde{k}_j$ is the allocated budget for interval j. Because $\sum_{j=1}^{M} \tilde{k}_j = B''$ and $\tilde{k}_j \leq \frac{1}{\epsilon}$ for all j, $\sum_{j=1}^{M} \tilde{k}_j^2$ reaches its maximum by letting as many intervals have budget $\frac{1}{\epsilon}$ as possible. Each of the other intervals either has 0 budget or has the remainder of the total budget. As a result, the number of intervals that have nonzero budget is $\lceil B''/\frac{1}{\epsilon} \rceil = \mathrm{O}\big((B''/\frac{1}{\epsilon})\big) = \mathrm{O}\big((\epsilon B'')\big)$. Thus, we conclude that the number of local reward evaluation is in

$$\mathrm{O}\big(\frac{1}{\epsilon^3} \sum_{j=1}^{M} \tilde{k}_j^2\big) = \mathrm{O}\big(\frac{1}{\epsilon^3} \cdot \epsilon B'' \cdot (\frac{1}{\epsilon})^2\big) = \mathrm{O}\big(\frac{1}{\epsilon^4} B\big) \tag{20}$$

□

4 Data Fusion for Frame Sequences

In this section, we first describe how we model a video as a graphical model. This model is a Markov chain which emits cheap and expensive features. Therefore, we can apply DPA to this model and Section 4.2 describes this.

4.1 A Graphical Model for Video

The graphical model is a Markov model, where each frame of video corresponds to a node. Each node contains a state variable that represents the property we wish to infer, such as whether a face or a moving object is present. Each node can emit two observable quantities, corresponding to cheap and expensive features extracted from the frame. This is similar to a hidden Markov model (HMM), but here we also model dependencies between observations. In our model, the value of a cheap feature at time t is not conditionally independent of the rest of the model given the state at time t, but is also dependent on the cheap feature at times $t-1$ and $t+1$. We do this to capture the fact that when an algorithm makes an error in one frame, it is quite likely to make a similar error at an adjacent frame. This model can be considered a type of autoregressive hidden Markov model [42]. We have experimentally verified in Section 5.5 that if we assume conditional independence between consecutive cheap features, the model will become overconfident about the evidence of the cheap features, resulting in less accurate inference.

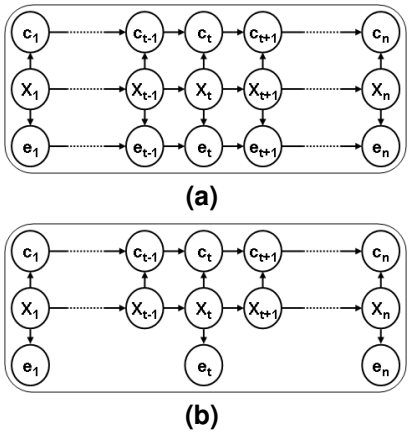


Fig. 3: Markov models for video sequences. State variables are labeled "X", cheap observations are labeled "c", and expensive observations are labeled "e". They all have numbered subscripts indicating their time steps. (a) The model we use when expensive features are not available at every frame; (b) The model we use when all frames have expensive features.

We typically have expensive features for a small fraction of frames, so it is less important to model the dependency between them. In addition, we assume the expensive feature is accurate when predicting the state. Therefore, we assume expensive features depend only on the state. However, in cases where we have an expensive feature at every frame, we model their dependencies the same way we do with cheap features (see Figure 3).

4.2 Applying DPA

We have described DPA for the case of simple, chain graphical models. However, it is straightforward to apply it to the model in the previous section, since it has a chain structure and the same conditional independence property. We assume that the expensive feature is accurate in predicting the state of a node. This allows an expensive feature to play the role of an observation in our algorithm. In practice, expensive features do make mistakes. This means that the states before and after the frame at which we apply an expensive feature are not truly conditionally independent, but only approximately so. We experimentally evaluate the consequences of this approximation in the next section.

Finally, cheap features also provide useful information. Thus, we make the recursive formulas for computing the optimal reward be conditioned on applicable cheap features. Denoting $c_{a:b}$ as the cheap feature over the interval from variable X_a and X_b, the formula (4)

and (5) for computing the optimal reward become

$$J_{a:b}(x_a, x_b; k) = \max\{J_{a:b}(x_a, x_b; 0), \max_{a<j<b}\{\sum_{x_j} P(X_j = x_j | X_a = x_a, X_b = x_b, c_{a:b})\{ R_j(X_j | X_j = x_j, c_{a:b}) + \max_{0\le l\le k-1}[J_{a:j}(x_a, x_j; l) + J_{j:b}(x_j, x_b; k-l-1)]\}\}\}, \quad (21)$$

where the base case is

$$J_{a:b}(x_a, x_b; 0) = \sum_{j=a+1}^{b-1} R_j(X_j | X_a = x_a, X_b = x_b, c_{a:b}). \quad (22)$$

Finally, when predicting the state of frames and using formula (21) and (22) to determine where to sample expensive features, we need to determine the probability distribution of each state based on observations. We can easily extend the standard Forward-Backward algorithm [43] to do this.

5 Experiments

We now apply DPA to two vision tasks involving motion detection and face detection. Our main goal is to show that inference can be used to efficiently allocate processing in two very different tasks. We begin by first describing some common characteristics of our experiments in the next section. We then present the results for the two tasks in section 5.2 and 5.3. Section 5.4 and 5.5 discuss how the expensive and cheap features can affect DPA. Section 5.6 shows how the subsection size affects the performance and running time of the algorithm. Finally, section 5.7 discusses how the concavity assumption holds for our sampling method.

5.1 General Experiment Setup

We use DPA described above to determine the locations at which to run the expensive algorithm. We uniformly sample the expensive feature at every 20 frames to break the sequence, while also running the cheap algorithm at every frame. We compare to several baseline algorithms. For all algorithms, when all the cheap and the necessary expensive observations have been made, we predict the state of each frame using the inference model in Fig. 3(a) since expensive features are not available in every frame. Competing algorithms are always provided the same total budget. We describe the budget in terms of its percentage of the total number of frames.

- The first baseline method is *uniform sampling*, which runs the expensive algorithm at a uniform step size. This method is in essence equivalent to running the expensive algorithm at a lower frame rate.
- The second baseline method is *most-relevant sampling*. We first run the cheap algorithm at each frame and perform inference using the model in Fig. 3(a) to obtain the conditional probabilities of all state variables. We then run the expensive algorithm on the frames that are most likely to satisfy our query. This is equivalent to using the cheap algorithm to prune the least interesting frames.
- The third and last baseline method is *most-uncertain sampling*. Similar to the most-relevant sampling method, we again run the cheap algorithm at each frame and then perform inference to obtain conditional probabilities. Then, we run the expensive algorithm on frames that have the greatest uncertainty, measured by the entropy of the conditional probability.
- Finally, to calibrate the performance of algorithms on different tasks, we compared to an idealized method, *ceiling sampling*, in which we run the cheap and expensive algorithms at all the frames, i.e, the budget is equal to 100%. We use Fig. 3(b) to model the frame sequence because the expensive feature is available everywhere. This method should provide an upper bound on performance.

To compare these methods, we used 4-fold cross validation on each video. We recorded each video at 30 frames per second. We then uniformly sampled 3 frames per second to generate the training and testing sequences, as this is a reasonable frame rate for real world surveillance videos [44]. By beginning sampling at different locations, we produced 10 different sequences for training and also for testing. All ten sequences are used as training data. We also use all ten for testing, helping to smooth the results a bit. The performance measure we used was the 11-point average precision of the precision recall (PR) curves [45]. That is, we take the average precision for 11 uniformly spaced levels of recall. This is averaged over all 40 testing sequences from the 4 folds. We varied the total budget from 5% to 25% of n. Because the subsection size is 20, DPA and uniform sampling have the same performance at a budget of 5%. We next provide more details about the experiments for both tasks.

In addition to these three baseline methods, we also considered a greedy selection scheme. This acquires one expensive observation at a time [38], [46], choosing the observation that provides the greatest increase in overall expected reward. That is, given that we have already applied the expensive algorithm to all frames in O, we next choose the expensive observation, E_j, that maximizes $R(O \cup E_j) - R(O)$. Unfortunately, this approach is not suitable for our problem. First, to implement this approach, at each iteration we must perform inference for each possible value of E_j over the whole video to determine $R(O \cup E_j)$, and repeat this for each j. This requires an impractical amount of run time. We have performed preliminary experiments in which we choose ten observations at a time after each round of inference. However, not only is this still slow, but this approach performs poorly compared with other baseline methods. So we omit this method from further experiments.

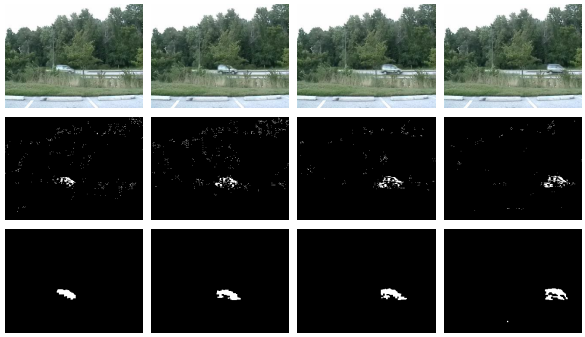


Fig. 4: Top: examples from video used in the motion detection task. Middle: output of FD. Bottom: output of IAGMM.

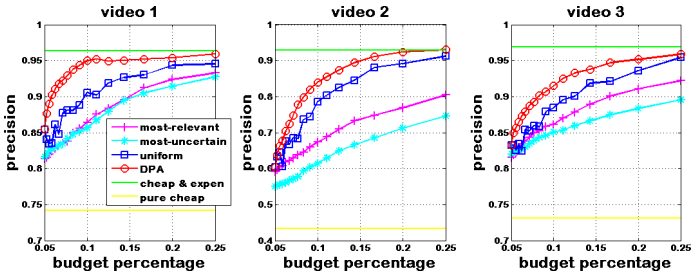


Fig. 5: 11-point average precision values for the background subtraction task.

TABLE 1: Concavity of RB curves for the motion detection tasks.

Video	Video 1	Video 2	Video 3
Number of intervals[a]	2720	2720	2720
concave (%)	56.21	67.39	59.93
concave or ≈ concave[b](%)	94.34	97.28	94.45
Median of the rest	0.0468	0.0181	0.0834

[a] The total number of intervals over the 40 testing sequences.
[b] RB curves which are not concave but with a nonconcavity measure not greater than 0.01.

5.2 Motion Detection

We first evaluated these algorithms in a simple background subtraction task. We collected three half hour videos at thirty frames per second, for a total of $n \approx 55,000$ per video, with each frame at 240×320 resolution. We hand-labeled each frame as "interesting" if it contained a moving object, such as a person or car, "uninteresting" otherwise.

As a cheap algorithm, we used FD [14] and we used IAGMM [16] as the expensive algorithm. For FD the feature was the number of foreground pixels in a frame after applying a threshold of 10. This avoided postprocessing, saving a significant amount of time. It is an interesting question for future work to determine how best to build a background model suitable for the expensive algorithm based on sparsely sampled frames. However, in this experiment we wish to focus on the effectiveness of our algorithm in directing application of IAGMM. Therefore, we build a background model using all recent frames and then apply IAGMM only at sampled locations. After applying IAGMM, we performed an opening and a closing morphological operation. We then extracted the area of the largest connected component to generate features, which were then discretized. We had tried 8 different features, the area of the component, the width of its Bounding Box, and the diameter of a circle with the same area as the component. They all produced similar performance, and we chose the area of the component to show results. Figure 4 shows some examples; the other video and output are similar.

Next, we tested our concavity assumption on these videos, since DPA assumes that the reward curves were concave. Table 1 shows the results. We can see that over a half of all intervals produce concave reward curves, while most of the non-concave ones have very small convexities.

We show 11-point average precision results on the three videos in Fig. 5. In all three videos, our method outperforms the baseline methods. We also observe from the plots that uniform sampling outperforms both most-relevant sampling and most-uncertain sampling. We postulate that the reason may be that the cheap algorithm does not produce high quality features and so decisions based purely on the cheap algorithm are unreliable. In these videos, FD faces difficulties because leaves often move in the background.

5.3 Face Detection

Next, we applied our approach to the problem of identifying frames containing a face. As with the last task, we collected three half-hour videos. We hand-labeled each frame as "interesting" if there is a frontal or profile face in it and labeled it as "uninteresting" otherwise.

For the cheap algorithm, we used IAGMM with the area of the largest connected component as a feature since it is relatively good at detecting the motion of a human and is still computationally cheap compared to the face detector. The expensive algorithm was the face detection algorithm based on OpenCV [47], using the scheme in [26]. We used both frontal and profile face detectors and the expensive feature was a binary indicator of whether the detectors found a face. Fig. 6

Fig. 6: Top: frames from video used in face detection. Middle: output of IAGMM. Bottom: output of the face detector.

TABLE 2: Concavity of RB curves for the face detection task.

Video	video 4	video 5	video 6
Number of subsections	2720	2840	2800
concave (%)	80.22	66.16	68.61
concave or ≈ concave (%)	98.16	98.45	99.07
Rest median	0.0970	0.1014	0.1014

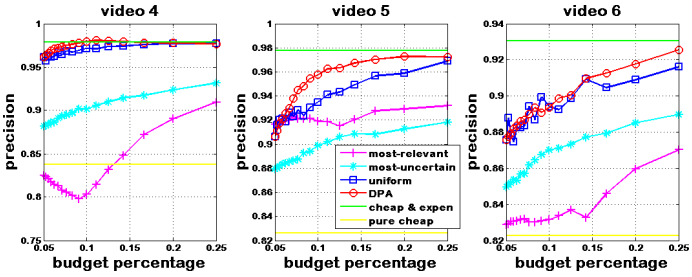


Fig. 7: 11-point average precision values for the face detection task.

shows examples from one of these videos, the others are similar.

We again first measured the concavity of the reward curves for the face detection videos. The results are given in Table 2. Again, the reward curves are largely concave. We show the 11-point average precision results on the three videos in Figure 7. Our method outperforms the baseline methods in two videos, video 4 and 5, under all budget percentages. However, in video 6, our method has no advantage over uniform sampling when the budget is small, but as the budget increases, the advantage of our method becomes clear.

5.4 Accuracy of Expensive Features

The accuracy of the expensive feature can cause variations in performance of DPA, since our decisions are based on the assumption that the expensive feature is very accurate. The prediction error rates of the expensive feature in the six videos are .0147, .0417, .0391, .0582, .0828, and .2339 respectively. Note that the error rate

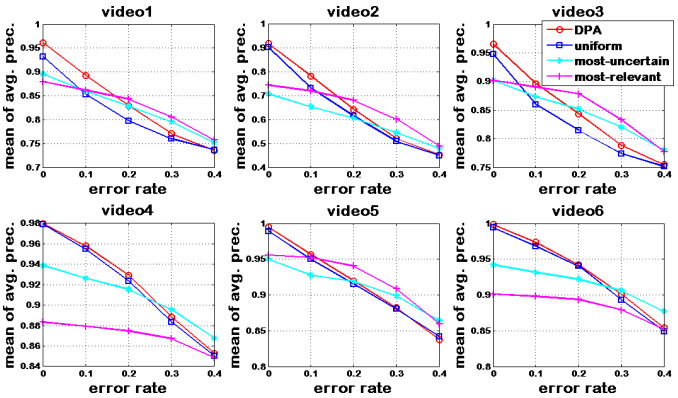


Fig. 8: Mean of average precision as the accuracy of the expensive features decreases.

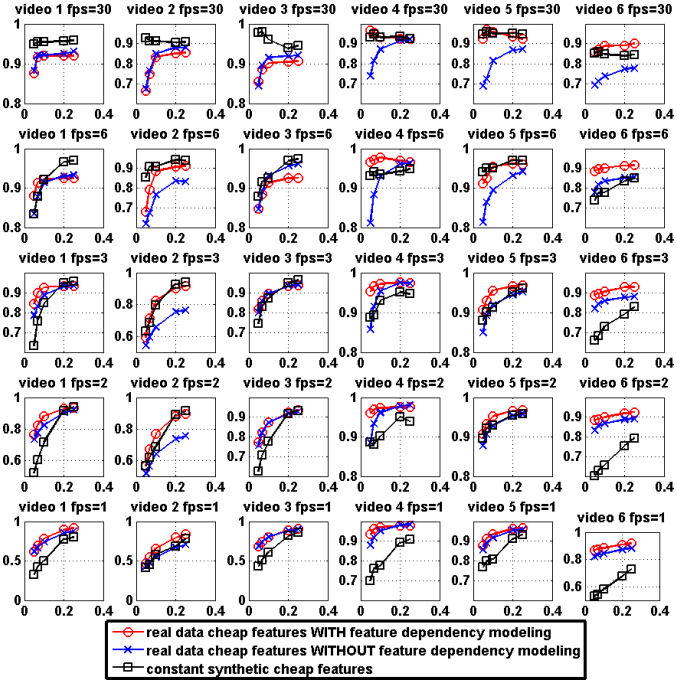


Fig. 9: Comparison of using constant synthetic cheap features, and cheap features from real data with and without feature dependency modeling, using a frame rate of 30, 6, 3, 2, and 1 frames per second. As in Fig. 5, the x axis is budget and the y axis is average precision.

is highest in the sixth video, the video on which our method has no clear advantage over uniform sampling when given a small budget.

Radovilsky et al. [48] shows that inaccurate observations lead to bounded loss on the subset selection version of VoIDP. To investigate how the accuracy of the expensive features affects our method, which is based on a conditional plan setting of VoIDP, we compare performance of different methods using synthetic expensive features with different accuracies for each video. Based on ground truth, we generate expensive features by randomly choosing a set of frames to make its value incorrectly reflect the actual states. We varied the size of the set to be 0%, 10%, 20%, 30%, and 40% of the total number of frames, and a smaller set is always the proper subset of a larger set. To measure the performance of a method, we use the mean of average precision for budget going from 5% to 25%. Fig. 8 shows the performance of different methods as the accuracy of the expensive features decreases. We observe that an accuracy over 85% in general is needed to give DPA an advantage over other methods.

5.5 Usefulness of Cheap Features and Feature Dependency Modeling

Ideally, combining both features in DPA to determine the expensive feature sampling locations should give better results than purely using expensive features. To show this, we replace the cheap feature with a constant

synthetic feature. Under this setting, the sampling locations and inference only depends on the expensive features. In addition, to show the importance of feature dependency modeling, we also compare to the result of using both types of features but without modeling the dependency between cheap features. That is, we remove the link between consecutive cheap features in Fig. 3(a), and the model becomes a standard HMM. We also experiment with varying frame rates, hoping to capture feature dependency at different levels.

Fig. 9 shows the performance when the frame rate is 30, 6, 3, 2, and 1. Under 3 frames per second, which is the frame rate in the experiments above, we observe that combining both features and modeling dependency has a clear advantage over using only the expensive features. Without dependency modeling, however, inference performance can sometimes be worse. For other frame rates, modeling feature dependency still has a clear advantage than that with no modeling. But we do observe that combining both features has no advantage when the frame rate is 30 or 6 frames per second. In particular, using purely expensive features has better performance for the first three videos for background subtraction at 30 frames per second. However, when the frame rate is 2 or 1 frame per second, the advantage of combining both features becomes clear again. In particular, even using the model without feature dependency is better than that using just expensive features when the frame rate is 1 frame per second. This is likely due to the issue of how well our model fits the data. When the frame rate becomes higher and higher, the dependency between errors made by cheap features is stronger and stronger, and our model cannot capture this well enough. As a result, the cheap feature is overconfident in some locations, and suppresses the correct decisions by the expensive features. Using a better model that captures this dependency adequately is one possible solution, and we discuss this more in section 5.8. Real world videos, such as surveillance videos, need to run for a long time in general. A small frame rate is more suitable for storage and power issues [44]. At the same time, when cheap features are not helpful, DPA will still provide significant benefits by using information from expensive features to control processing.

5.6 Size of Subsections and Running Time

The size of subsections determines B', the budget used for uniform sampling. With larger subsections, more budget can be used within the subsections. However, a smaller subsection size does have advantages in terms of running time. Fig. 10 shows performance of DPA as the subsection size decreases. In general, we observe that a larger subsection size has better performance at the same budget level. This is because more budget can be allocated by VoIDP-SCP with a larger subsection size, and the allocation is affected less by the uniform sampling. However, at a high budget level, performance of

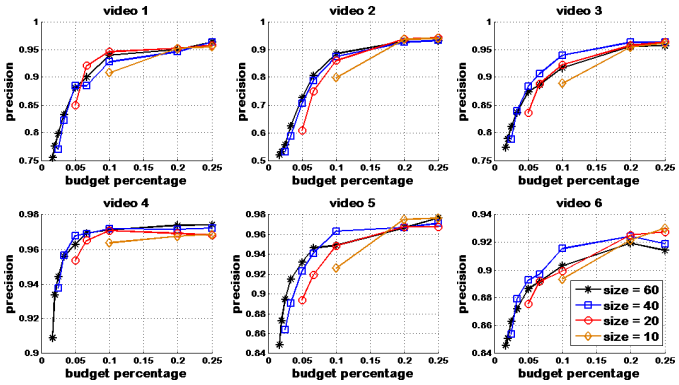


Fig. 10: 11-point average precision values as size of subsections varies in DPA.

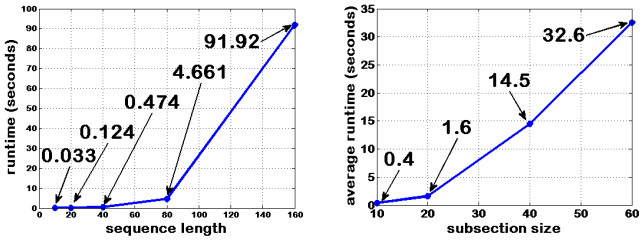


Fig. 11: Left: running time of VoIDP-SCP at a budget of 25% of the sequence length. Right: running time of DPA at a budget of 25% averaged over all testing sequences from all videos. We use the multi-dimensional array to store the memory table in the dynamic programming. This allows the fastest table lookup speed. These measures are taken using a server with two 2.66GHz quad-core Xeon processors with 48GB of memory)

different subsection sizes start to converge. In addition, the performance at size 60 has no advantage over that of size 40 in many cases. This may indicate a saturation of performance as the size increases. The left side of Fig. 11 shows an example of the running time of VoIDP-SCP as the size of the subsections increases. The plot shows that we cannot afford to run VoIDP-SCP on the whole testing sequence in our experiment, which has length over 1000. The right side shows the running time of DPA with increasing subsection size averaged over all testing sequences from all videos. Each testing sequence is about one fourth of a 30-minute video at 3 frames per second. A large subsection size, such as 40 or 60, requires a fair amount of running time. A small size, such as 10, runs very fast, but suffers from disadvantages discussed above. In our experiments, we choose 20 as the subsection size, which has reasonable performance and fast running time.

5.7 Concavity

Our algorithm assumes that the RB curve is concave. Empirically, we find that this assumption holds well in our two, real-life domains. We have also performed experiments with synthetic data to get a sense of when this assumption might fail. We generate data from a

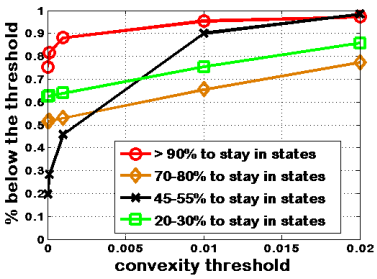


Fig. 12: Percentage of subsections from all sampled sequences whose nonconcavity measure is not greater than a varying threshold. Each sampled sequence has a length of 2000, producing 100 subsections with a subsection size of 20. The threshold values are 0, 0.0001, 0.001, 0.01 and 0.02. The legend indicates probability for each state to stay in its current value. All other parameters of the model are sparsely sampled to cover their value ranges. Going from top to down in the legend, the number of sampled sequences used to generate the curve is 1728, 1750, 1956, and 1960 respectively.

Markov process as shown in Figure 3(b). Our experiments indicate that the key factor in determining the concavity of the RB curve is the probability of a state change from one node to the next. When states persist, RB curves tend to be concave. Figure 12 shows the variation in concavity with the probability that a state persists from one time step to the next. Each curve pools results from a large number of simulations in which other parameters of the model vary. We also explored other state transition cases, such as when one state tends to persist while the other does not. However, their results are not as good as the case that both states tend to persist. We note that in our tasks, and in many other video analysis tasks, node states will persist; once an interesting event begins in a video it will tend to last for at least a few seconds. Therefore, we expect our results to be applicable in many settings.

5.8 Discussion

Our experiments demonstrate that DPA can make two potential contributions to video processing. First, we have shown that it is possible to use a Markov model to integrate cheap and expensive features to improve system accuracy. Second, we have shown that by sparsely applying expensive features, our algorithm can use the results of inference to direct processing to portions of the video where further processing is most beneficial.

At the same time, we note that these potential benefits are dependent on some important assumptions. First, inaccurate expensive features will affect the performance of DPA. The advantage of DPA will decreases as the accuracy of the expensive features decreases. The results in section 5.4 indicate that an 85% accuracy is in general needed to allow DPA to outperform other baseline methods. In many complex video analysis problems in unconstrained environments this accuracy is not yet achievable. For this reason we feel that DPA will be most relevant in two situations. First, in many controlled, or partially controlled environments, vision algorithms can achieve high accuracy. Second, in some applications a human analyst may serve as an expensive feature. It will be an interesting problem for future research to explore the use of DPA in integrating algorithmic output with human analysis.

It might also be of interest to modify VoIDP-SCP so that rather than considering all possible state assignments for each potential split point, it considers all possible expensive feature assignments. Such a method requires modification of formulas 4 and 5 for dynamic programming such that they are based on values of features rather than values of states. Since the feature space is in general much larger than the state space, this will be more computationally intensive. A similar idea has been exploited in the work by Radovilsky et al. [48]. However, they consider a subset selection setting of the problem which determines all sampling positions before samples are made, while we consider a conditional plan setting in which the next sampling position is conditioned on the sampled observations. These two problems have different recursive formulas for dynamic programming [5], and their approach may not be directly applicable to our problem.

A second issue that deserves further exploration is the modeling of dependency between features. Section 5.5 shows that this can improve inference. However, at high frame rates (eg., 30 fps) our model is still not able to properly capture these dependencies. It will be interesting to consider more sophisticated models, such as Conditional Random Fields (CRFs) [49]. CRFs can capture arbitrary dependencies among input observation variables, by conditioning on all inputs.

Finally, we have demonstrated our approach on relatively simple sample video that we have collected. It would be of interest to define tasks for which expensive features can achieve high accuracy in real-world surveillance datasets, such as i-LIDS MCTTR dataset [50]. On these challenging datasets, it might, for example, be of interest, to consider problems in which state-of-the-art algorithms act as a cheap feature and a human analyst serves as an expensive feature.

6 Conclusion

Our main goal has been to design inference algorithms that can be used to direct video processing. This allows us to replace simplistic methods such as reducing the frame rate with principled decisions that carry theoretical performance guarantees. We believe that this is a quite general framework that can be applied to many video processing tasks and may be extended in the future to more complex graphical structures.

To this end, we have made two more detailed contributions. First, we propose a graphical model that maps onto a video frame sequence and allows us to combine features from expensive and cheap algorithms to do

inference. We show that in practical situations, there is much to be gained by this combination. Second, we have shown how to build on an existing algorithm that was designed for short chains to create an algorithm that runs efficiently on long video sequences. Specifically, we show that by applying an expensive algorithm in some extra locations, we can determine future sensing locations efficiently. Experiments with two concrete video processing tasks, low-level background subtraction and the higher level task of face detection, show that these can be mapped onto our framework. The effectiveness of DPA's inference algorithm in these tasks illustrates the potential of our approach for general video processing.

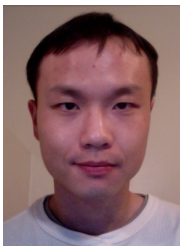
ACKNOWLEDGMENTS

The authors would like to thank reviewers for their helpful comments. This work was supported by the Army Research Office, ARO #W911NF0810466. Mustafa Bilgic was also supported under NSF Grant #0746930.

REFERENCES

[1] L. Telindus Surveillance Systems, "http://www.telindussurveillance-us.com/," 2005.
[2] N. Dean, "Bombers staged dry run before london attacks," The Independent, online edition, 20 Sept. 2005.
[3] A. J. R. Westrop, "Face detection technology on digital cameras works," http://ezinearticles.com/?Face-Detection-Technology-on-Digital-Cameras-Works&id=1598427.
[4] "Brickstream launches video analytics managed services program," http://www.prweb.com/releases/2008/05/prweb955134.htm, 2008.
[5] A. Krause and C. Guestrin, "Optimal value of information in graphical models," *Journal of Artificial Intelligence Research*, vol. 35, pp. 557–591, 2009.
[6] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, 1997.
[7] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," in *Proceedings of International Symposium on Intelligent Multimedia, Video and Speech Processing*, 2001, pp. 158–161.
[8] A. M. Elgammal, D. Harwood, and L. S. Davis, "Non-parametric model for background subtraction," in *European Conference on Computer Vision*, 2000.
[9] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. S. Davis, "Real-time foreground-background segmentation using codebook model," *Real-Time Imaging*, vol. 11, no. 3, pp. 172–185, 2005.
[10] J. Rittscher, J. Kato, S. Joga, and A. Blake, "A probabilistic background model for tracking," in *European Conference on Computer Vision*, 2000.
[11] S.-C. S. Cheung and C. Kamath, "Robust techniques for background subtraction in urban traffic video," vol. 5308, no. 1. SPIE, 2004, pp. 881–892.
[12] M. Piccardi, "Background subtraction techniques: a review," in *IEEE International Conference on Systems, Man and Cybernetics*, 2004.
[13] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Survey*, vol. 38, no. 4, p. 13, 2006.
[14] R. Jain and H. Nagel, "On the analysis of accumulative difference pictures from image sequences of real world scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, pp. 206–213, 1979.
[15] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, 1999.
[16] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *International Conference on Pattern Recognition*, 2004.
[17] M. hsuan Yang, D. J. Kriegman, S. Member, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 34–58, 2002.
[18] G. Yang and T. Huang, "Human face detection in a complex background," *Pattern Recognition*, vol. 27, no. 1, pp. 53–63, January 1994.
[19] R. Hsu, M. Abdel-Mottaleb, and A. Jain, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, 2002.
[20] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, 2002.
[21] P. Sinha, "Object recognition via image invariants: A case study," *Investigative Ophthalmology and Visual Science*, vol. 35, no. 1, pp. 1,735–1,740, May 1994.
[22] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines: an application to face detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.
[23] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998.
[24] H. Schneiderman and T. Kanade, "Probabilistic modeling of local appearance and spatial relationships for object recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*.
[25] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2002.
[26] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *IEEE International Conference on Image Processing*, 2002.
[27] C. Huang, H. Ai, Y. Li, and S. Lao, "Vector boosting for rotation invariant multi-view face detection," in *IEEE International Conference on Computer Vision*, vol. 1, oct. 2005, pp. 446 – 453 Vol. 1.
[28] T. Mita, T. Kaneko, and O. Hori, "Joint haar-like features for face detection," in *IEEE International Conference on Computer Vision*, vol. 2, oct. 2005, pp. 1619 –1626 Vol. 2.
[29] R. Xiao, L. Zhu, and H.-J. Zhang, "Boosting chain learning for object detection," in *IEEE International Conference on Computer Vision*, oct. 2003, pp. 709 –715 vol.1.
[30] D. Weiss and B. Taskar, "Structured prediction cascades," in *International Conference on Artificial Intelligence and Statistics*, 2010.
[31] P. F. Felzenszwalb, R. Girshick, and D. McAllester, "Cascade object detection with deformable part models," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
[32] S. Vijayanarasimhan and A. Kapoor, "Visual recognition and detection under bounded computational resources," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.
[33] R. Krishna, K. McCusker, and N. O'Connor, "Optimising resource allocation for background modeling using algorithm switching," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2008.
[34] S. Barotti, L. Lombardi, and P. Lombardi, "Multi-module switching and fusion for robust video surveillance," in *International Conference on Image Analysis and Processing*, 2003.
[35] V. Bayer-Zubek, "Learning diagnostic policies from examples by systematic search," in *Conference on Uncertainty in Artificial Intelligence*, 2004.
[36] P. D. Turney, "Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm," *Journal of Artificial Intelligence Research*, vol. 2, pp. 369–409, 1995.
[37] M. Rattigan, M. Maier, and D. Jensen, "Exploiting network structure for active inference in collective classification," in *ICDM Workshop on Mining Graphs and Complex Structures*, 2007.
[38] M. Bilgic and L. Getoor, "Effective label acquisition for collective classification," in *International Conference on Knowledge Discovery and Data mining*, 2008.
[39] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.
[40] S. Vijayanarasimhan and K. Grauman, "Whats it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
[41] S. Vijayanarasimhan, P. Jain, and K. Grauman, "Far-sighted active learning on a budget for image and video recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[42] K. P. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, UC Berkeley, Computer Science Division, 2002.
[43] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.
[44] C. Loy, T. Xiang, and S. Gong, "Multi-camera activity correlation analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
[45] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.
[46] R. Howard, "Information value theory," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 2, no. 1, pp. 22 –26, aug. 1966.
[47] Intel, "Opencv open source computer vision library," http://www.intel.com/technology/computing/opencv/index.htm.
[48] Y. Radovilsky, G. Shattah, and S. Shimony, "Efficient deterministic approximation algorithms for non-myopic value of information in graphical models," 2006.
[49] C. Sutton and A. McCallum, "An introduction to conditional random fields for relational learning," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. Cambridge, MA, USA: MIT Press, 2007, pp. 93–127.
[50] J. Fiscus, J. Garofolo, T. Rose, and M. Michel, "Avss multiple camera person tracking challenge evaluation overview," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009, pp. 219 –219.



Daozheng Chen is a PhD student in Computer Science from the University of Maryland, College Park. He received his B.S. and M.S. degree in computer science from the University of Maryland, College Park, in 2007 and 2009 respectively. His research interests include computer vision, such as object recognition and video analysis, and related topics in machine learning, such as probabilistic graphical models. He is a member of the IEEE.

Mustafa Bilgic is an Assistant Professor in the Computer Science Department at Illinois Institute of Technology. He received his BS degree in 2004 from University of Texas at Austin and his PhD degree in 2010 from University of Maryland at College Park. His research interests include machine learning, active learning, experimental design, probabilistic graphical models, and decision theory. His work on active inference won the best student paper award in ACM KDD 2008.

Lise Getoor is an Associate Professor in the Computer Science Department at the University of Maryland, College Park. She received her undergraduate degree from University of California, Santa Barbara, her Masters degree in computer science from University of California, Berkeley, and her PhD from Stanford University, all in computer science. Her research areas include machine learning, and reasoning under uncertainty; in addition she works in data management, visual analytics and social network analysis. She is well-known for work in statistical relational learning. She is a board member of the International Machine Learning Society, a Machine Learning Journal Action Editor, an Associate Editor for the ACM Transactions of Knowledge Discovery from Data, a former JAIR Associate Editor, and she has served on the AAAI Council. She has served on the PC of many major conferences including the senior PC of AAAI, ICML, KDD, UAI and the PC of SIGMOD, VLDB, and WWW. She is PC co-chair of ICML 2011. She is a recipient of an NSF Career Award and was awarded a National Physical Sciences Consortium Fellowship.

David Jacobs received the B.A. degree from Yale University in 1982. From 1982 to 1985 he worked for Control Data Corporation on the development of data base management systems, and attended graduate school in computer science at New York University. From 1985 to 1992 he attended M.I.T., where he received M.S. and Ph.D. degrees in computer science. From 1992 to 2002 he was a Research Scientist and then a Senior Research Scientist at the NEC Research Institute. In 1998 he spent a sabbatical at the Royal Institute of Technology (KTH) in Stockholm. Since 2002, he has been at the University of Maryland, College Park, where he is a Professor.

Dr. Jacobs' research has focused on human and computer vision, especially in the areas of object recognition and perceptual organization. He has also published articles in the areas of motion understanding, memory and learning, computer graphics, human computer interaction, and computational geometry. He has served as an Associate Editor of IEEE Transactions on Pattern Analysis and Machine Intelligence, and has assisted in the organization of many workshops and conferences, including serving as Program co-Chair for CVPR 2010. He and his co-authors received honorable mention for the best paper award at CVPR 2000. He also co-authored a paper that received the best student paper award at UIST 2003.