Efficient Resource-constrained Retrospective Analysis of Long Video Sequences

Daozheng Chen, Mustafa Bilgic, Lise Getoor, David Jacobs
Department of Computer Science
University of Maryland, College Park
{dchen, mbilgic, getoor, djacobs}@cs.umd.edu

Introduction: New technology is creating large stores of digital video. Real time processing of these 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. We develop a new method for controlling processing, so that available resources are directed at the most relevant portions of the video. In our 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 a new probabilistic inference algorithm to determine to which frames we should apply further expensive processing.

Approach: We model the video sequence with a Markov model that is similar to an HMM. Each frame corresponds to a node with a binary variable, X_i that indicates whether the state has a specific property, such as containing a face or moving object. There is a cheap feature, c_i, observed for every node, and an expensive feature, e_i, that *may* be observed at every node. As in an HMM, a directed edge connects each node X_i to X_{i+1}. In addition, X_i connects to c_i and e_i, and a directed edge connects each cheap observation c_i to c_{i+1}. These last connections model the fact that when a cheap feature is in error in one frame, a similar error is likely in the next frame. Because we assume that the expensive feature is quite accurate, it is less necessary to model this dependency for them. This model can be considered a type of autoregressive hidden Markov model [1], or conditional random field, and we can use an extension of the standard forward-backward algorithm to do inference.

Next, we assume that we can apply the expensive algorithm to a fixed number of frames, B. We choose these expensive observations to maximize a reward that depends on our ability to correctly infer the state of each frame. Observations can include variables at any time step in the chain since we consider a retrospective analysis of a video.

If we assume that observing the expensive feature at a node fully determines that node's state, then we can apply an algorithm of Krause and Guestrin [2] to solve this problem, because, conditioned on the cheap observations, our model becomes a Markov chain. They note that in this case, 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 present an optimal solution that runs in time $O(B^2n^3)$, where n is the number of states in the chain. For their applications, n is small, but in our setting n corresponds to the number of frames in hours(or day)-long video, making it impractical to directly apply this algorithm to retrospective video analysis.

Building on that work, we present a novel algorithm to find observation plans that is efficient enough to apply to problems with very large values of n. The main insight of this approach is that rather than just allocating observations to optimize reward, in practice it may make sense to allocate some observations in a way that makes the inference algorithm more tractable. This leads to an algorithm we refer to as *Dynamic Processing Allocation* (DPA):

1. We use B' observations to make uniform observations to break the chain model into consecutive disjoint intervals separated by the observed variables;
2. We allocate the remaining budget $B'' = B - B'$ between these intervals in a batch mode;

3. We use the optimal observation algorithm of Krause and Guestrin to determine the observation locations within each interval in a conditional mode, in which the placement of each observation depends on all prior observations in that interval.

To perform step 2, we observe that for each interval, the optimal reward typically forms a convex curve as the budget increases. This means that we suppose the reward function exhibits submodularity [3]. By assuming convexity, we are able to show that the optimal allocation of observations between intervals is found by greedily assigning observations to whichever interval will benefit most from an additional observation.

The complexity of DPA is $O(\frac{1}{\epsilon^4}B + Blog(B))$, where $\epsilon = \frac{B'}{n}$. This algorithm is guaranteed to perform at least as well as an algorithm that optimally allocates B'' observations in a batch mode. In practice, it should be much better than that, because the B' observations used to break the video into intervals are very useful for inference, and because the allocation of observations in Step 3 is conditional, and therefore more effective than a batch allocation.

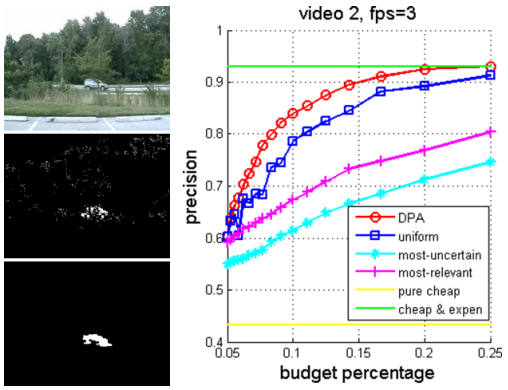
Experiments: We applied DPA to motion detection and face detection. Given a video we first use B' budget to run the expensive algorithm uniformly with a step size of 20, while also running the cheap algorithm on all frames. Then we use our budget allocation algorithm to optimally distribute the remaining budget B'' among the resulting intervals. Finally, when all observations have been made, we perform inference to predict the states.

We compared our results to several baseline algorithms supplied with the same total budget. The first baseline method is *uniform sampling*, which runs the expensive algorithm at a lower frame rate. The second baseline method is *most-relevant sampling*. In this sampling method, we perform inference using the cheap features and then run the expensive algorithm on the frames that are most likely to satisfy our query. The third and last baseline method is *most-uncertain sampling*. This is like the previous method, except we run the expensive algorithm on frames that have the greatest uncertainty. 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% of n. This method should provide an upper bound on the performance of any method given that the model fits the data well enough.

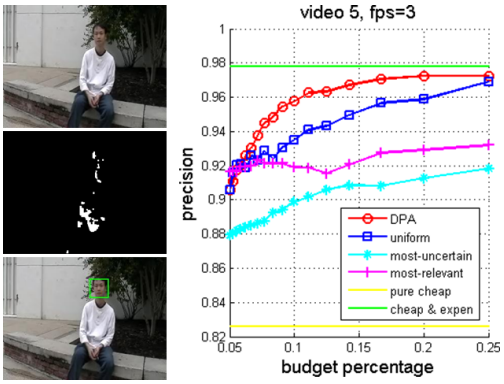
To compare these methods, we used 4-fold cross validation on each video, using three-quarters of the data to train the model and one quarter to test. We recorded each video in 30 frames per second. We then uniformly sampled 3 frames per second to generate the training and testing sequences; this is a reasonable frame rate for real world surveillance videos [4]. By beginning sampling at different locations, we produced ten different sequences with which we train and test. The performance measure we used was the 11-point average precision of the precision-recall (PR) curves [5]. This is averaged over all 40 testing sequences from the 4 folds. We used the conditional probabilities for each frame's state variable to generate PR curves. To compare the sampling methods under varying conditions, we varied the total budget from 5% to 25% of n.

We first evaluated these algorithms on a simple motion detection task. We collected three half hour videos at thirty frames per second, referred to as videos 1, 2, and 3. We hand-labeled each frame as "interesting" if it contained a moving object, "uninteresting" otherwise. As a cheap motion detection algorithm, we used frame differencing (FD) [6] and we used Gaussian mixtures (IAGMM) [7] as the expensive algorithm. With FD, the cheap feature was the number of foreground pixels. With IAGMM, we extracted the largest connected component to generate more accurate and expensive features. The left side of Figure 1(a) shows frame examples and outputs by the two algorithms for video 2, and the right side shows the corresponding 11-point average precision results. We can see that our method outperforms the baseline methods, and we observe similar results for video 1 and 2.

Next, we applied our approach to the problem of face detection. As with the last task, we collected three half-hour videos, referred to as videos 4, 5, and 6. We hand-labeled each frame as "interesting" if it contained a frontal or profile face 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 motion provides a clue that a person is present, and it is still computationally cheap compared to the face detector. The expensive algorithm was the face detection algorithm based on OpenCV [8], using the scheme in [9]. We used both frontal and profile face detectors and the expensive feature was a binary indicator of whether the detectors found a face. The left side of Figure 1(b) shows frame examples and outputs by IAGMM and the face detection algorithms for video 5. The right



(a) Video 2: frame examples and performance



(b) Video 5: frame examples and performance

Figure 1: Frames examples and performance for video 2 and 5. Within each subfigure, frame examples are on the left side, and performance are on the right side. For the frame examples in (a), from top to bottom, they are the original frame, output from FD, and output from IAGMM. For the frame examples in (b), from top to bottom, they are the original frame, output from IAGMM, and output from face detection.

side of Figure 1(b) shows the corresponding 11-point average precision results, which indicates that our method outperforms the baseline methods. We observed similar results for video 4. Video 6 shows an example in which our method does not outperform uniform sampling. In this video, the face detector performs at only about 80% accuracy. Our algorithm assumes that the expensive feature is very accurate and does not perform well if this assumption is grossly wrong.

Conclusions: 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.

Acknowledgments: This work was supported by ARO #W911NF-08-1-0466.

References

[1] K. P. Murphy, "Dynamic bayesian networks: Representation, inference and learning," Ph.D. dissertation, UC Berkeley, Computer Science Division, 2002.

[2] A. Krause and C. Guestrin, "Optimal nonmyopic value of information in graphical models - efficient algorithms and theoretical limits," in *International Joint Conference on Artificial Intelligence*, 2005.

[3] A. Krause, B. McMahan, C. Guestrin, and A. Gupta, "Robust submodular observation selection," *Journal of Machine Learning Research*, vol. 9, pp. 2761–2801, 2008.

[4] C. Loy, T. Xiang, and S. Gong, "Multi-camera activity correlation analysis," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[5] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[6] 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.

[7] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *International Conference on Pattern Recognition*, 2004.

[8] Intel, "Opencv open source computer vision library," http://opencv.willowgarage.com/wiki/.

[9] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *IEEE International Conference on Image Processing*, 2002.