Controlling Your TV with Gestures

Ming-yu Chen[1], Lily Mummert[2], Padmanabhan Pillai[2],
Alex Hauptmann[1], Rahul Sukthankar[2,1]

[1] Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213
+1 412 268 1448
{ mychen,alex,rahuls}@cs.cmu.edu

[2] Intel Labs Pittsburgh
4720 Forbes Ave, Suite 410
Pittsburgh, PA 15213
+1 412 297 4020
{lily.b.mummert,padmanabhan.s.pillai}@intel.com

ABSTRACT
Vision-based user interfaces enable natural interaction modalities such as gestures. Such interfaces require computationally intensive video processing at low latency. We demonstrate an application that recognizes gestures to control TV operations. Accurate recognition is achieved by using a new descriptor called MoSIFT, which explicitly encodes optical flow with appearance features. MoSIFT is computationally expensive — a sequential implementation runs 100 times slower than real time. To reduce latency sufficiently for interaction, the application is implemented on a runtime system that exploits the parallelism inherent in video understanding applications.

Categories and Subject Descriptors
C.3 [**Computer Systems Organization**]: Special-Purpose and application base systems. D.2 [**Software**] Software engineering.

General Terms
Algorithms, Performance, Design.

Keywords
Parallel Computing, Cluster Applications, Multimedia, Sensing, Stream Processing, Computational Perception.

1. INTRODUCTION
Video is becoming ubiquitous in daily life and the rate at which video is being generated has accelerated demand for machine understanding of rich media. Systems for processing video have been traditionally evaluated according the accuracy with which they can recognize events of interest, and the rate at which data can be processed. However, as interactive video applications become more prominent, latency is becoming increasingly important. Latency directly impacts many applications because they require that the results of video understanding be made immediately available to the user. Examples of latency-sensitive applications include surveillance scenarios where the operator must be quickly alerted in an emergency, and vision-based user interfaces or immersive environments where even moderate latencies unacceptably degrade the user experience. There has been extensive research on frame-rate processing of video, but simply achieving the desired throughput does not necessarily lead to any improvement in latency. A major barrier to widespread deployment of video understanding algorithms has been their computational expense. For instance, current methods for recognizing human activities in surveillance video typically involve spatio-temporal analysis such as computing optical flow and 3D SIFT descriptors at multiple scales for every frame in a high-resolution stream. Fortunately, the increasing availability of large-scale computer clusters is driving efforts to parallelize video applications. The majority of these efforts, such as MapReduce [9] and Dryad [11], focus on efficient batch analysis of large data sets; and while such systems accelerate offline indexing of video content, they do not support continuous processing. A smaller set of systems provide support for continuous processing of streaming data [1, 3, 8] but most focus on queries using relational operators and data types, or are intended for mining applications where throughput is optimized.

Figure 1: Setup of TV/camera for gestural control system.

Our demonstration application involves a situation where the television set is actively observing the viewers all the time. This enables any viewer to control a TV's operations, such as channel selection and volume, without additional devices such as remote controls, motion sensors or special clothing, simply by gesturing to the TV set. We demonstrate an implementation of a low-latency gesture recognition system that processes video from a

MIR'10, March 29–31, 2010, Philadelphia, Pennsylvania, USA.

commodity camera to identify complex gestures in real time and interpret them to control the TV set. Detailed examples of different user gestures used in our interface can be found at http://lastlaugh.inf.cs.cmu.edu/MIR10Demo and Figures 1 - 3. While this demo uses a commodity webcam, our proposed approach can be applied to video from depth-enhanced cameras that will soon become available. Such sensors offer increased resiliency to background clutter, and initial reports indicate that they are well suited for natural user interfaces [16].

Our demonstration allows any user standing or sitting in front of a TV set to control its operations through gestures. The TV is equipped with a camera that observes the users watching the programs. When a user gives an 'attention' signal by raising both arms, the control application then observes this user more carefully for a few seconds to recognize a control command. Examples of control commands can be hand and arm motion upward or outward, as well as crossing hands/arms. In the current interface, e.g., a left hand moving upwards indicates a channel should be switched up, a left hand moving outwards signifies that the channel should be switched down. Analogously we use the right hand to control the volume of the audio. Crossing gestures are used to shut off the TV. User tests showed that downward motions cannot be effectively executed by seated users; therefore we avoided downward motions in the current gesture command set.

In this demonstration, we highlight two novel aspects of our human-activity recognition research. First, we employ a novel and robust descriptor called MoSIFT, which exploits continuous object motion explicitly calculated from optical flow and integrates it with distinctive appearance features. Although computationally more expensive, this approach significantly outperforms state-of-the-art approaches [12, 14, 18] on standard action recognition data sets [7]. These results validate our belief that the added computational complexity of sophisticated descriptors is warranted.

Second, we utilize a cluster-based distributed runtime system called Sprout that achieves low latency by exploiting the parallelism inherent in video understanding applications to run them in interactive time scales. In particular, although straightforward sequential implementations of MoSIFT can process relatively small collections of videos, such as the popular KTH dataset [21], they cannot process data at the speed required for the real-world applications that are the primary focus of our research. Our system implements the computationally challenging, but highly accurate MoSIFT descriptor on top of the Sprout runtime, and parallelizes execution across a cluster of several 8-core machines, to detect TV control gestures in full-frame-rate video with low latency.

2. ACTIVITY RECOGNITION IN VIDEO

Activity recognition forms the core of video understanding systems for interactive applications. In this section, we briefly review current approaches to the problem and describe a feature representation, MoSIFT, that we employ in our system for extracting semantic content to control TV operations. This descriptor matches (or exceeds) state-of-the-art descriptors in terms of recognition accuracy on established action recognition datasets and has performed well on the challenging Gatwick airport surveillance collection. These experiments validate our decision to use MoSIFT in this demonstration.

Figure 2: User gesturing "Channel Up".

Current approaches to action recognition in video are typically structured as follows: (1) identify a set of semantically-interesting regions in the video; (2) characterize the spatio-temporal neighborhood at each interest point as a feature vector, which is often quantized using a codebook; (3) aggregate the set of features extracted in a video snippet to generate a histogram of their occurrence frequencies; (4) treat this histogram as a high-dimensional vector and classify it using a machine learning technique trained on human-annotated video sequences.

Interest point detection reduces video data from a large volume of pixels to a sparse but descriptive set of features. Ideally, an interest point detector should densely sample those portions of the video where events occur while avoiding regions of low activity. Therefore, our goal is to develop a method that generates a sufficient but manageable number of interest points that can capture the information necessary to recognize arbitrary observed actions. Popular spatio-temporal interest point detectors [10, 13] are spatio-temporal generalizations of established 2D operators developed for image processing, such as the Harris corner detector. Although mathematically elegant, these approaches treat motion in an implicit manner and exhibit limited sensitivity for sooth gestures, which lack sharp space-time extrema [12]. By contrast, the philosophy behind the MoSIFT detector is to treat appearance and motion separately, and to explicitly identify spatially-distinctive regions in a frame that exhibit sufficient motion at a variety of spatial scales.

The information in the neighborhood of each interest point is expressed using a descriptor that explicitly encodes both an appearance and a motion component. We are not the first to do this; several researchers [14, 20] have augmented spatio-temporal representations with histograms of optical flow (HoF). However, unlike those approaches, where the appearance and motion information is separately aggregated, MoSIFT constructs a single feature descriptor that concatenates appearance and motion. The former aspect is captured using the popular SIFT descriptor [15] and the latter using a SIFT-like encoding of the local optical flow. MoSIFT is a superset of the Laptev et al. detector [14] since MoSIFT not only detects velocity changes but also smooth

movements. Additional implementation details of MoSIFT are given in our technical report [6].

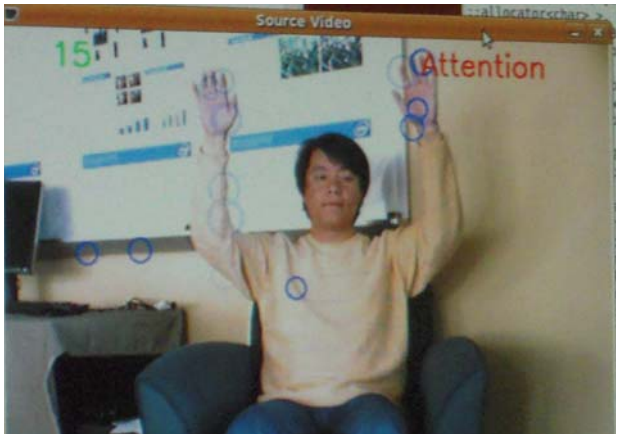

Figure 3: TV view of the user gesturing to control operations

We adopt the popular bag-of-features representation for action recognition, summarized as follows: Interest points are extracted from a set of training video clips. K-Means clustering is then applied over the set of descriptors to construct a codebook. Each video clip is represented by a histogram of occurrence of each codeword (bag of features). This histogram is treated as an input vector for a support vector machine (SVM) [8], with a Chi-Square kernel [22]. Since the SVM is a binary classifier, to detect multiple actions we adopt the standard one-vs-all strategy to train separate SVMs for multi-class learning.

While MoSIFT significantly improves activity recognition, the gain in accuracy is computationally expensive because it not only scans though different spatial scales but also calculates corresponding optical flows. A straightforward implementation of MoSIFT would exhibit unacceptable performance in terms of throughput and latency. On a single processor computing all the MoSIFT descriptors in a standard definition video stream would take 100 times slower than real time. With a naively parallelized implementation in which frames are processed in a pipelined fashion, the system would incur a delay of more than 2.5 seconds between an activity and a result action.

3. A PARALLEL IMPLEMENTATION OF GESTURE-BASED TV CONTROL

Our application is implemented on the Sprout runtime system, a distributed stream processing system designed to enable the creation of interactive perception applications [19]. Interaction requires low end-to-end latency, typically well under 1 second [4, 5, 17]. Sprout achieves low latency by exploiting the coarse-grained parallelism inherent in such applications, executing parallel tasks on clusters of commodity multi-core servers. Its programming model facilitates the expression of application parallelism while hiding much of the complexity of parallel and distributed programming.

Sprout applications are structured as data flow graphs. The vertices of the graph are coarse-grained processing steps called stages, and the edges are connectors which represent data dependencies between stages. The data flow model is particularly well suited for perception, computer vision, and multimedia processing tasks because it mirrors the high-level structure of these applications, which typically apply a series of processing steps to a stream of video or audio data. Concurrency in the data flow model is explicit — stages may execute in parallel, constrained only by their data dependencies and the availability of processors.

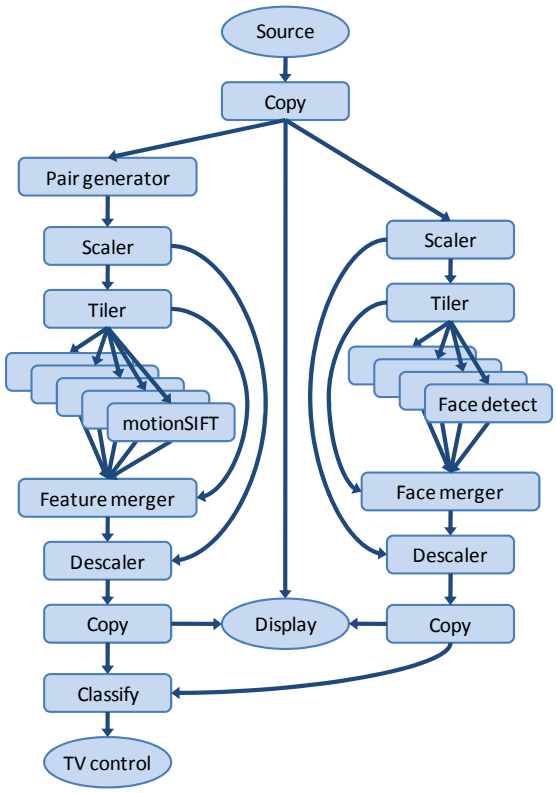

Figure 4 illustrates our application flow.

Figure 4 illustrates our application data flow. Each video frame from a camera that observes the user is sent to two separate tasks, face detection and MoSIFT detection task. The incoming frame is duplicated (Copy stage) and sent to two different stages which initialize tasks. The face detection task starts from a scale stage (Scaler) which scales the frame to a desired size. The tiling stage (Tiler) is an example of coarse-grained parallelization. The tiler divides each frame into configurable number of uniformly sized overlapping sub-regions. The tiles are sent to a set of stages to be processed in parallel. The tiler also generates meta-data that includes positions and sizes of the tiles, for merging the results. The face detected in the scaled frame is de-scaled via Descaler stage to recover the resolution. The face detection result is then sent to the display stage to display and a classify stage which will further fuse face detection result with MoSIFT features to detect gestures. The MoSIFT detection task accumulates frame pairs, and then extracts MoSIFT features that encode optical flow in addition to appearance. These features, filtered by the positions of detected faces, are aggregated over a window of frames to generate a histogram of their occurrence frequencies. The histogram is treated as an input vector to a set of support vector machines trained to detect gestures in video streaming. These

processes are included in Classify stage. The gesture detection result is further sent to TV control stage to perform associated TV controlling.

4. CONCLUSIONS

Efficient and automatic processing of streaming video content at low latencies is critical for a large class of emerging applications in surveillance, gaming, intelligent environments and vision-based user interfaces. Our novel representation for video content (MoSIFT) significantly improves the accuracy of human activity recognition but it is so computationally expensive that naive implementations on a single processor are impractical for complex real-world video control and gaming applications. Thus, we developed a novel method for leveraging clusters of multi-core processors to significantly improve latency and throughput. This method exploits both the coarse- and the fine-grained parallelism inherent in the algorithm. The system enables video understanding algorithms to process streaming video for a wide variety of applications, such as gestures for TV control described here. In future work, we plan to extend the approach to work with depth-augmented video to improve the robustness of our system.

5. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under Grants Nos. IIS-0917072, IIS-0812465, and IIS- 0751185. Any opinions, findings, and conclusions expressed in this material are those of the author(s) and do not reflect the views of the National Science Foundation.

6. REFERENCES

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Çetintemel, M. Cherniack, J. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, and S. Zdonik. The design of the Borealis stream processing engine. In Proc. Innovative Data Systems Research, 2005.

[2] J. K. Aggarwal and Q. Cai. Human motion analysis: a review. In Proc. Nonrigid and Articulated Motion Workshop, 1997.

[3] L. Amini, H. Andrade, R. Bhagwan, F. Eskesen, R. King, P. Selo, Y. Park, and C. Venkatramani. SPC: A distributed, scalable platform for data mining. Workshop on Data Mining Standards, Services, and Platforms, 2006.

[4] J. Brady. A theory of productivity in the creative process. IEEE Computer Graphics and Applications, 6(5):25–34, May 1986.

[5] S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In CHI ’91: Human factors in computing systems, 181–186, 1991.

[6] M.-Y. Chen and A. Hauptmann. Mosift: Recognizing human actions in surveillance videos. In CMU-CS-09-161, 2009.

[7] M.-Y. Chen; L. Mummert; P. Pillai; A. Hauptmann; R.Sukthankar, Exploiting Multi-level Parallelism for Low-latency Activity Recognition in Streaming Video; Proc. ACM Multimedia Systems (MMSys) Conference, 2010,

[8] M. Cherniack, H. Balakrishnan, M. Balazinska, D. Carney, U. Çetintemel, Y. Xing, and S. Zdonik. Scalable distributed stream processing. In Proc. Innovative Data Systems Research, 2003.

[9] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. CACM, 51(1), 2008.

[10] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In IEEE Workshop on PETS, 2005.

[11] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: distributed data-parallel programs from sequential building blocks. Proc. European Conference on Computer Systems, 2007.

[12] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. Proc. Int’l Conference on Computer Vision, 2005.

[13] I. Laptev and T. Lindeberg. Space-time interest points. In Proc. Int’l Conference on Computer Vision, 2003.

[14] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In Proc. Computer Vision and Pattern Recognition, 2008.

[15] D. Lowe. Distinctive image features form scale-invariant keypoints. Int’l Journal on Computer Vision, 60(2), 2004.

[16] Microsoft, "Project Natal in detail". Microsoft. June 2009. http://www.xbox.com/en-GB/news-features/news/Project-Natal-in-detail-050609.htm. Retrieved Jan 26, 2010.

[17] R. B. Miller. Response time in man-computer conversational transactions. In AFIPS ’68: Proc. of the Dec. 9-11, 1968, joint computer conference (Fall, part I), pages 267–277, 1968.

[18] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. In Proc. British Machine Vision Conference, 2006.

[19] P. Pillai, L. Mummert, S. Schlosser, R. Sukthankar, and C. Helfrich. SLIPStream: scalable low-latency interactive perception on streaming data. In Proc. NOSSDAV, 2009.

[20] K. Schindler and L. Van Gool. Action snippets: How many frames does human action recognition require? In Proc. Computer Vision and Pattern Recognition, 2008.

[21] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local SVM approach. In Proc. ICPR, 2004.

[22] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. Int’l Journal on Computer Vision, 73(2), 2007.