Automatic Real Time Key-frame Detection in Video Stream Using SURF Algorithm

Javier Iparraguirre, Claudio Delrieux, and Lisandro Perez Meyer

Universidad Nacional del Sur, Laboratorio de Ciencias de las Imagenes,
Avenida Alem 1253, Bahia Blanca, Argentina
{javier.iparraguirre,cad,perezmeyer}@uns.edu.ar
http://www.imaglabs.org

Abstract. Real-time video processing is steadily becoming a standard matter in a wide diversity of applications, including computer vision, surveilance, social networks, and many other. Unsupervised video interpretation is required to avoid the operative expense and faultiness of human-in-the-loop interpretation. However, robust, general purpose real-time unsupervised video interpretation and analysis appears to be among the most difficult processing tasks. In this work we present advances in real-time video stream processing, in particular in automatic key-frames detection. This detection procedure is based on feature analysis and evaluation of the features detected by the popular SURF algorithm. We present the implementation details, some interesting experimental results are shown, and we discuss some of the applications that our detection algorithm may have.

Keywords: Computer vision, video processing, robust visual descriptor, key-frame detection, parallel processing

1 Introduction

Contents extraction in video sequences is not a new topic [9], but recently it has become increasingly popular given the wide spectrum of applications related to real-time video processing in general, and with unsupervised video interpretation in particular. This research topic is getting an ever increasing attention, and therefore many major breakthroughs are expected to be achieved in the coming years [6]. However, unsupervised, real-time video interpretation has been addressed many times as a tough research topic, and despite current advances, there is a lot of room for improvements and new achievements.

Contents extraction in video sequences can be divided into several procedures, among which key-frame detection is essential in video interpretation, like splitting, motion analysis, and many other recognition related algorithms. In [4], the authors use the k-means algorithm and they determine the key frames based on frame clustering. Although they can detect key-frames without human intervention, the method is computationally very expensive and cannot be used in real-time video processing.

In [8] a system for parallel tracking and mapping augmented reality workspaces is presented. In this work, key-frames detection is based on SLAM algorithm. In [11] the authors use SURF algorithm to extract features from the video sequence, but the features are not used to detect a key-frame but to contrast them against a database of known frames.

This paper presents a method for real-time, unsupervised key-frame detection using the Speeded-up Robust Features (SURF) algorithm [3, 2]. This method focuses on the similarities among frames in the video stream. It is designed around a dynamic feature variation criterion that takes into account: amount of features, variation in the features number, and features distance in a 64 dimension space. The method calculates for each frame a relevance index. When the relevance index passes a threshold value, it is possible to split the video sequence into takes.

In Section 2 we present the conceptual ideas behind the selection process. The way the application was implemented is detailed and the results in Section 3. Section 4 presents the conclusions and future work of our development.

2 Key-frame Selection Criteria

Given a particular video stream, we process each of the frames as an individual entity. Once a frame is selected, we apply the SURF algorithm [3, 2]. The algorithm evaluates a set of parameters about the frame. This set is a descriptor of the feature set of the frame, which can be regarded as a vector in a 64 dimension phase-space.

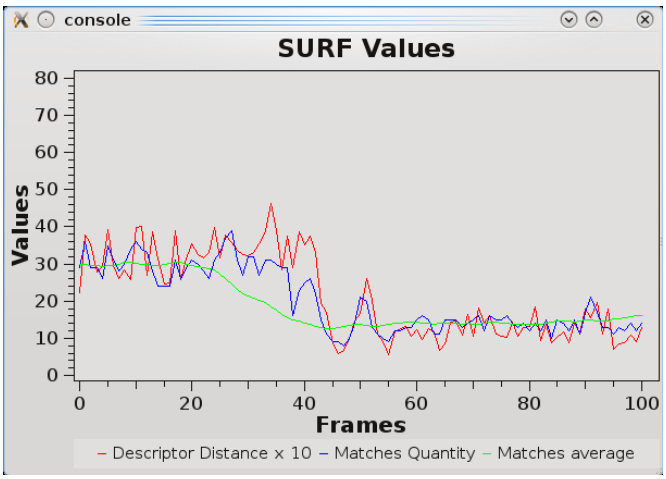

Fig. 1. The algorithm evaluates three parameters in each frame. In this case we show the descriptor distance (red), the amount of matches that a frame produces (blue), and the average of matches in the last 20 frames (green)

Based on the output of SURF, we created three parameters: the amount of features detected per frame, the average value of the features in the last 20 frames, and the Euclidean distance in the descriptor space for all the features in a particular frame. A frame may be a candidate key frame when there is a significant difference among this particular frame and the previous one. This can be expressed in our parameter selection stating that a key frame candidate is a frame with a significant difference between the quantity of features and the average. Following this line of reasoning, we implemented the key-frame selection using a perceptual difference threshold. Changing on the threshold value, it is possible to adjust the *sensitivity* of the system (see Fig. 1).

3 Implementation Details

Our application opens three windows. The main window shows the parameters produced by the SURF algorithm. The second window shows the video stream with a superimposition of the features detected by the algorithm. Finally the third window shows the latest detected key-fame. We build the application based on several open source packages: OpenCV [5] to handle the images, OpenSURF [7] for an implementation of SURF. The plot was built on top of QWT [10]. The whole package was built using C++ and Qt [1] as base tools. Since real-time video processing demands huge computational resources, we developed a multi-threaded architecture. The application runs in two threads, one launches the application and executes the main window, the other thread executes the OpenSURF. We run the application on a dual-core machine that uses the benefits of a multi-threaded application.

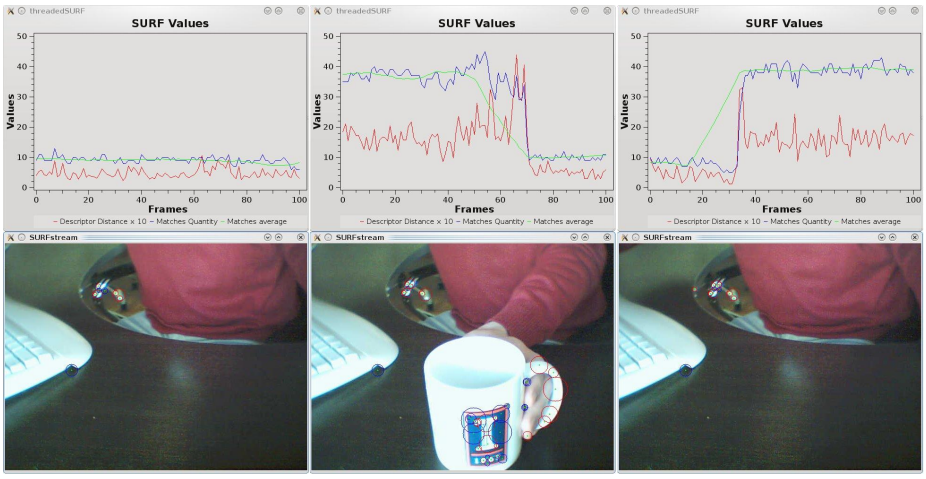

Fig. 2. The figure shows three cases in which a key-frame is detected. In each case it is possible to observe the difference between the number of features and the average

We run the application on a live camera and we observer the response of the system. In general terms the application works as expected. We set a threshold of 30 percent and the sensibility of the systems was reasonable. Figure 2 shows the system in action.

4 Conclusions and Future Work

We presented an unsupervised real-time key-frame detection algorithm for video sequences. The application is based on a parameter set evaluated from the features computed by the SURF algorithm. We developed a proof-of-concept application to exemplify these ideas, which is based on open source packages. The experimental behavior of the application is satisfactory in a wide set of examples. The sensitivity parameter can be adjusted to select a specific detection threshold. The applications takes full advantage of the many-core processing achitecture due a full multithreaded design. In future developments we are planning to improve the performance of the key-frame detection, but also to add information retrieval algorithms. Once we detect a key-frame, it is straightforward to launch a new thread that classifies the information contained in the frame, perhaps using any of the web-services already providing this. This may give rise to a full unsupervised video interpretation system that may have many applications as discussed in the introduction.

References

1. Qt: a cross-platform application and UI framework.
2. H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.
3. H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Lecture notes in computer science*, 3951:404, 2006.
4. D. Borth, A. Ulges, C. Schulze, and T.M. Breuel. Keyframe Extraction for Video Tagging and Summarization. In *Proc. Informatiktage*, pages 45–48, 2008.
5. G.R. Bradski and A. Kaehler. *Learning opencv*. O'Reilly, 2008.
6. R. Datta, D. Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys (CSUR)*, 40(2):1–60, 2008.
7. Christopher Evans. Notes on the opensurf library. Technical Report CSTR-09-001, University of Bristol, January 2009.
8. G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.
9. M.S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):19, 2006.
10. U. Rathmann, G. Vermeulen, M. Bieber, R. Dennington, and J. Wilgen. QwtQt Widgets for Technical Applications.
11. G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod. Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In *Proceeding of the 1st ACM international conference on Multimedia information retrieval*, pages 427–434. ACM, 2008.