OpenTrack - Automated Camera Control for Lecture Recordings

Benjamin Wulff
Universität Osnabrück
Zentrum virtUOS
bwulff@uni-osnabrueck.de

Rüdiger Rolf
Universität Osnabrück
Zentrum virtUOS
rrolf@uni-osnabrueck.de

Abstract

In this paper we present the current state of our research project which aims to develop an open-source framework for real-time scene analysis and automatic camera control in lecture recording scenarios. The system is designed to run in alliance with the Opencast Matterhorn lecture capture system as well as stand-alone. A GPU-based scene segmentation technique using motion cues and background modeling has been implemented using OpenCL. Moving objects are tracked by their centroids and bounding boxes and a dynamic appearance model is used to give persons a relative identity. Development started on a scriptable virtual camera operator module that will drive PTZ-cameras. Other applications of the scene analysis are possible.

1 Introduction

The recording of lectures and the distribution of these recordings over the web have become quite common at many universities within the last 10 years [9]. Open-source lecture recording systems, like Opencast Matterhorn [10], have helped to automate these recordings to a very high degree and make it affordable even for small universities, with not much staff for media production [14]. However, within the classroom a camera operator is still needed if a good classroom experience with interesting and engaging camera views is wanted. Instead static camera perspectives are often the only choice when it comes to a broad adoption of lecture recording. A close shot on the lectern and wide-angle overview perspectives are mostly used, both show specific advantages and down-sides.

Research on context aware environments for lecture recordings started in the 1990. One offspring of this research is the commercial AutoAditorium system [1] [2]. Contributions to the field haven been made by Microsoft Research [16] [12] [15] and others [4] [5].

Our aim is to create an open-source framework for real-time scene analysis and camera control. The system should be easily deployable on common low cost PC hardware. A high degree of modularity and extendability is our ambition for the design of the system. This paper mainly describes the scene analysis component that has been implemented so far, since this was our research focus during the last month. The outlook section 3 describes our plans for further development.

OpenTrack is an OpenCast community project and is designed to work in alliance with the Matterhorn capture agent (though it can also run stand-alone). Opencast Matterhorn is an open-source lecture capture system that already offers a high degree of automation. Events can be scheduled manually or imported from a learning management system. The so called *capture agents* in the classrooms conduct the scheduled recordings automatically without requiring the presenter to do anything but her usual work. Finished recordings are processed using configurable workflows and published to different distribution channels [10]. OpenTracks functionality should be as unobtrusive as the Matterhorn system already is for the lecturer.

2 System

2.1 The Domain

The system is intended to be a tool for automated lecture recording. This defines the domain in which it has to provide its functionality in a stable manner.

One can assume that the recording will take place indoors in a lecture hall or seminar room. For a normal lecture the area observed by the system can be assumed to be a relatively stable scene with a mostly static background and moving objects like persons or equipment (blackboards, projectors, chairs etc). Thus the system normally won't have to deal with complex background scenes like it is the case in outdoor surveillance scenarios, where a vision system has to deal for example with vegetation moved by wind.

From the fact that the system is used in the context of video production it can further be assumed that lighting conditions will be prepared in a way that allows for the captur-

978-0-7695-4589-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ISM.2011.97

ing images of a certain degree of quality. But though artificial lighting adjusted for video production should be dominant, changes in illumination due to natural light cannot be assumed to be absent.

We can expect that there are regions of interest in the scene, e.g. close to the blackboards, and that there are regions that we would like to ignore, e.g. the doors where the students enter the room.

We usually will try to follow a single person that resides in a region of interest, this person might change during a session, i.e. in a seminar with several speakers, but we must not expect that there are several people in our region of interest that we will have to separate from each other and follow one specific subject. If we encounter several people in our region of interest, it will be appropriate to show the whole scene without separating persons standing close to each other. When they go in different directions the individual tracking can be restored again.

2.2 Architecture

2.2.1 Camera Setup

We decided to use a setup similar to the one described in [15]. We have a pan-tilt camera that can be moved around and we have a separate webcam mounted close to the pan-tilt camera to get an overview of the room. We analyze the video from the webcam with some computer-vision techniques that we will describe later in this paper to evaluate the scene and to create the commands to move the pan-tilt camera.

2.2.2 Implementation

Like Opencast Matterhorn the system is designed as Service Orientated Architecture (SOA) implemented in Java as OSGi bundles [10]. The OSGi-standard describes an open software modularization framework for the Java programming language [18].

The concept of *Bundles* is one key point in OSGi. Functional components of a software are encapsulated completely into these bundles. In runtime each bundle has its own Classloader which makes the use of the different versions of the same library in one software system possible. A bundle can also include its own set of native libraries [18] which need not to be installed in the host system. Physically a bundle is embodied by a single JAR file.

The bundles of our implementation fall into two categories we call *Infrastructure* and *Business Logic*. The infrastructure bundles serve as the technical basis for the implementation of the systems functionality. An OpenCL bundle for parallel computations, the FrameSource bundles for the acquisition of video streams and a camera control bundle currently based on the Sony VISCA protocol fall into this category. The scene segmentation, object analysis and tracking and the camera operator bundles fall into the business logic category as they implement the systems intended functionality. While the infrastructure modules already define a stable basis for the system, it is our intention that the focus of further research and development will be on the business logic components.

The image processing functionalities are implemented using OpenCL. It is an open standard for general purpose parallel programming [11]. As of now most modern graphics cards support this standard. OpenCL describes an API for communication with the *compute device* (GPU) and for device code (the portions of code that are executed in parallel on the GPU) OpenCL "utilizes a subset of ISO C99 with extensions for parallelism"[11]. We have chosen to shift the image processing portions of the system to the GPU mainly because of the fact that many image processing techniques have an obvious (i.e. pixel-vise operations like image convolution) or at least immanent parallel nature and thus their implementation can benefit from harvesting the parallel computation abilities of modern GPUs. Also, due to their specialization for graphics processing, certain abilities like image sampling are implemented directly by the hardware and can be used very efficiently. Lastly shifting the very time consuming image processing operations to the graphics processor takes a big load of work from the shoulders of the CPU and hence makes execution of other tasks like video stream encoding possible on the same machine.

2.3 Scene Analysis

A lot of research has been done on people detection and tracking for video surveillance applications. A system like W^4 [7] that is meant for out-door surveillance has to deal with much rougher preconditions than a system in the lecture recording domain (see section 2.1). Several background modeling techniques have been proposed to overcome those problems, an overview is given in [13]. Surveillance systems for use indoors like ADVISOR [17] face more stable conditions and use simpler background subtraction techniques.

In our scene segmentation we use a very simple subtraction method much like the one proposed in [6]. A pixel in the input image is considered a foreground pixel if the absolute difference of the input pixel and the pixel in the background model is above a threshold θ_{BG}. Instead of intensity images RGB color images are used. The differences in each color channel are summed before the thresholding. The resulting binary image D_{BG} is first eroded with a small structuring element, then dilated with a larger structuring element. A background pixel is modeled by its mean RGB values. The background model is also updated like pro-

posed in [6], but we use a high value for the parameter α that determines how fast changes in the scene are incorporated into the model. The model is only updated in those regions that are not marked as foreground.

Systems that use background modeling like [7] and [17] use the generated binary image as the foreground mask in each frame (after cleaning with morphological operations). In our scene analysis the actual foreground mask is build over time using the results of the background subtraction together with the detection of change in consecutive frames. Regions of change are detected with the method described above, summing the absolute differences of the color channels and thresholding, but with a lower threshold θ_{IF} than for the background subtraction. The resulting binary image D_{IF} is dilated. Then for each pixel in D_{IF} it is decided if it changed from background to foreground or vice versa based on the corresponding element in D_{BG}. The foreground image F is an intensity image. Pixels that became foreground are set to 255, pixels that became background are set to 0 in F.

Figure 1. Update map, light regions are added to the foreground image, dark regions are removed

As the next step the parallelized connected component analysis that was proposed in [8] is conducted on F. The blobs size computation also proposed in [8] is applied too. Since the analysis also determines the number of pixels in each blob, very small and too big blobs can be erased from F at this point. The remaining blobs are assigned with sequential numbers. They are the blobs Id in the current frame and the index of this particular blob in the result arrays generated by further analysis steps.

As proposed in [8] we extended the algorithm with further analysis functionalities. For each blob in F the number of pixels that changed since the last frame is counted. If the ratio of total pixel conunt to "active" pixels is above a threshold θ_{act} all pixels in the blob are set back to 255. At the start of each iteration the values of all non-zero pixels in F are diminished by the value δ_F. This way blobs that show little or no activity fade and disappear,δ_F controls how fast. This idea was born out of the observation that persons giving a talk virtually never stand completely still. On the other hand in many cases false detections resulting from changes in illumination appear once but stay inactive. The sames hold for pieces of equiment that are moved in the scene.

Another extension to the blob coloring algorithm is the computation of the bounding boxes and the centroids for each blob in F. The system tracks the objects found by the foreground segmentation by trying to correlate the centroids in the current frame with those from the last. If there is no match found for a centroid, the bounding box data from the last frame is used to infer splitting or merging of blobs. To re-identify targets when they re-appear, either because they separate from a blob or when they re-enter the scene, textural templates similar to those described in [7] are computed and constantly updated for each target.

3 Outlook

The scene analysis can not only be used for directing a pan-tilt-camera in real time, it will also be a useful feature in the analysis of high resolution videos that could be cropped to be more focused on the lecturer in the recording like proposed in [5]. Because of the OSGI architecture the image processing functionalities of OpenTrack could easily be used for post-processing in Opencast Matterhorn.

With the analysis of recorded material there is no need for the system to make every decision automated. The system could ask an administrator later on which of the identified persons should be tracked in the video and which can be ignored. The identified people can be labeled with their names and it can be added to the mpeg7 metadata of the recording, when a person shows up in the recording and when the person leaves the scene. This can be used as additional chapter marks in the player for example.

Our approach will not only be able to track people. There are more elements in a room that can be important to a student in a recording. If the lecturer writes something on the chalkboard this should be shown to the viewer for a while like depicted in [5]. The analysis module could even make a screenshot before a board will be cleaned up, so that this screenshots can be browsed by the viewer later on. The MPEG4 specification allows different layers within a video, *Video Object Planes*[1], that could be turned on and off for example. If we would turn off such a layer with a person we can replace him with the image data from the background model and the viewer would only see a vague shadow of the removed person. This option could be useful if students have been recorded accidentally, which you were not allowed to record.

We don't expect that first versions of this tool will work

[1] http://mpeg.chiariglione.org/standards/mpeg-4/mpeg-4.htm#10.6

in a perfect way. There are several situations in a live lecture that may confuse our software and it will produce unexpected results. There have to be backup strategies for such cases like showing the whole room instead. Even if the scene analysis works as expected the director module will need to be improved iteratively. Not every lecture follows the same rules. Therefore we started developing a scriptable camera operator module. A description language for camera strategies is described in [3]. With such a platform it would be possible to treat a seminar, with changing speakers during a session, different from a regular lecture where every other persons except the lecturer should be ignored.

4 Conclusion

The goal of OpenTrack is not to completely replace human camera operators, but to allow universities to produce lecture recordings, with a video that contains all important information in enough detail, in an efficient way. If the universities want to offer lecture recordings for many courses they cannot afford the human workforce that is needed.

When applied to already recorded material the analysis functions of OpenTrack can help to enrich the metadata of the recording and improve the post-production of the video.

OpenTrack is intended to be an open-source project. We hope that our initial afford to create an architecture and a framework for an automated lecturer tracking system will allow others to create the automated camera operator that they need for their lectures and that they will contribute their improvements back to the community.

References

[1] M. Bianchi. Autoauditorium: a fully automatic, multi-camera system to televise auditorium presentations. 1998.

[2] M. Bianchi. Automatic video production of lectures using an intelligent and aware environment. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, MUM '04, pages 117–123, New York, NY, USA, 2004. ACM.

[3] D. Christianson, S. Anderson, L. He, D. Salesin, D. Weld, and M. Cohen. Declarative camera control for automatic cinematography. In *Proceedings of the National Conference on Artificial Intelligence*, pages 148–155, 1996.

[4] S. Drucker and D. Zeltzer. Camdroid: A system for implementing intelligent camera control. In *Proceedings of the 1995 symposium on Interactive 3D graphics*, pages 139–144. ACM, 1995.

[5] M. Gleicher, R. Heck, and M. Wallick. A framework for virtual videography. In *Proceedings of the 2nd international symposium on Smart graphics*, pages 9–16. ACM, 2002.

[6] J. Heikkila and O. Silvén. A real-time system for monitoring of cyclists and pedestrians. In *Visual Surveillance, 1999. Second IEEE Workshop on,(VS'99)*, pages 74–81. IEEE, 1999.

[7] D. H. I. Haritaoglu and L. Davis. W^4: Real-Time Surveillance of People and Their Activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809 – 830, Aug. 2000.

[8] O. Kalentev, A. Rai, S. Kemnitz, and R. Schneider. Connected component labeling on a 2d grid using cuda. *Journal of Parallel and Distributed Computing*, 2010.

[9] M. Ketterl and K. Morisse. User generated web lecture snippets to support a blended learning approach. In G. Siemens and C. Fulford, editors, *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2009*, pages 2886–2893, Honolulu, HI, USA, June 2009. AACE.

[10] M. Ketterl, O. A. Schulte, and A. Hochman. Opencast matterhorn: A community-driven open source software project for producing, managing, and distributing academic video. *Interact. Techn. Smart Edu.*, 7(3):168–180, 2010.

[11] Khronos OpenCL Working Group. The OpenCL specification, version 1.0, revision 29. http://www.khronos.org/opencl/, 2008.

[12] Q. Liu, Y. Rui, A. Gupta, and J. Cadiz. Automating camera management for lecture room environments. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449. ACM, 2001.

[13] A. McIvor. Background subtraction techniques. *Proc. of Image and Vision Computing*, 1(3):155–163, 2000.

[14] R. Mertens, M. Ketterl, and O. Vornberger. The virtpresenter lecture recording system: Automated production of web lectures with interactive content overviews. *Interactive Technology and Smart Education*, 4(1):55–65, 2007.

[15] Y. Rui, A. Gupta, and J. Grudin. Videography for telepresentations. In *Computer Human Interaction*, pages 457–464, 2003.

[16] Y. Rui, L. He, A. Gupta, and Q. Liu. Building an intelligent camera management system. In *Proceedings of the ninth ACM international conference on Multimedia*, MULTIMEDIA '01, pages 2–11, New York, NY, USA, 2001. ACM.

[17] N. Siebel and S. Maybank. The advisor visual surveillance system. In *ECCV 2004 workshop Applications of Computer Vision (ACV)*, volume 1. Citeseer, 2004.

[18] The OSGi Alliance. OSGi service platform core specification, release 4, version 4.2. http://www.osgi.org/Specifications, 2009.