Evaluating clustering in smart camera networks

Bernhard Dieber*†
*Video and Security Technology
Austrian Institute of Technology, Vienna
Email: bernhard.dieber@uni-klu.ac.at

Bernhard Rinner† (Advisor)
†Institute of Networked and Embedded Systems
Klagenfurt University, Austria
Email: bernhard.rinner@uni-klu.ac.at

***Abstract*—Clustering is an important concept for structuring large networks of cameras. In this research we investigate the various trade offs of clustering in networks of smart cameras. Major research questions in this context are (i) how to model clustering, (ii) how to deal with the heterogeneity in large camera networks, and (iii) how to integrate clustering in real-world networks. We have developed a flexible and scalable software suite that supports clustering in camera networks. We present first results in a multi-camera person tracking case study.**

I. Introduction

Many surveillance networks consist of a large number of cameras—some of them with more than thousand cameras. Recent surveillance systems not only capture and record visual data, they abstract and fuse data from the individual cameras to obtain a compact global view of the monitored area [1]. This data abstraction and fusion can be performed *centralized* by a dedicated node, fully *distributed* in the network or within *clusters* of cameras.

Clustering lies somehow in between the other two approaches wrt. the degree of distribution. It is a very natural approach for multi-camera applications because (i) the underlying network is often hierarchically structured, (ii) processing data in small groups of cameras is typically sufficient to detect specific activities in the monitored area and (iii) it offers a good trade off between scalability and flexibility.

Figure 1 depicts an example for clustering in a camera network. Clustering can be performed hierarchical at the level of rooms, floors or buildings. Within each cluster, a dedicated node fuses the data streams from the cluster cameras and generates an abstracted output potentially delivered to other clusters.

II. Research Questions

There are various ways how cameras can be organized into clusters. For example, clustering can be based on the network topology, the cameras' field of view or their locations. Clusters can have different properties such as the number of cameras, the processing performed within the cluster and the delivered output. In this research project, we are interested in evaluating clustering in smart camera networks [2], i.e., we investigate different the methods and provide tools for rapid prototyping and monitoring of clustered camera networks. We address the following research questions.

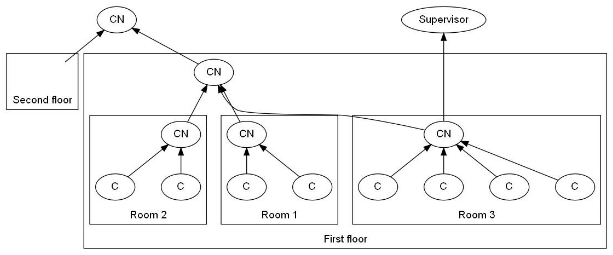

Fig. 1: Clustering a set of cameras (C) using cluster nodes (CN).

Modeling Cluster-based Networks: For modeling cluster-based camera networks we adopt a data-flow approach. A cluster node merges the output from a set of smart cameras, processes this data and generates an abstracted output. Various parameters of this simple model can change from cluster to cluster or over time. Examples of these parameters include the number and type of inputs, the processing within the cluster node and the number and type of generated outputs. Thus, an important question is how to model individual clusters as well as the overall camera network at a sufficient level of detail.

Flexible and Scalable Cluster Processing: In many surveillance systems the individual cameras have different capabilities. Some cameras can simply capture images and transfer the raw data to the cluster node. Other (smart) cameras can perform advanced image analysis onboard and deliver high-level results. Naturally, the heterogeneity of the cameras effects the processing within the cluster. Thus, we want to exploit this heterogeneity and evaluate the trade off between local processing at smart cameras and centralized processing at the cluster node. The scalability of clusters is another important aspect of this research.

Prototyping and Case Studies: An important goal is to apply, demonstrate and evaluate clustering in real camera networks. This requires a flexible software tool for the cluster nodes. This tool includes a plug-in mechanism for easy integration of various SW modules, a composable processing pipeline for the cluster node and support for SW deployment. Performance monitoring is required for the evaluation.

III. A software suite for cluster nodes

To evaluate different clustering scenarios we have implemented a special software suite which can run on all cluster nodes in the network. As briefly discussed above the main

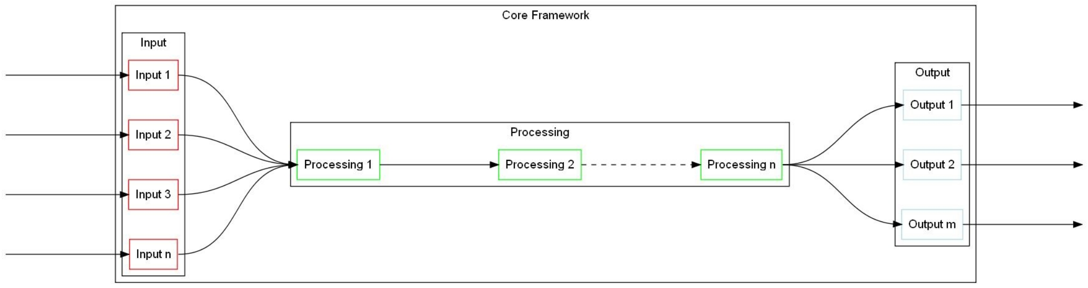

Fig. 2: Data-flow oriented structure of a cluster node

requirements for this software suite include flexibility, composability, as well as support for deployment and monitoring.

The design is based on a plug-in based structure which enables quick exchange of individual SW modules to adapt the processing pipeline to the current requirements of the cluster node (cp. Figure 2). We support multiple parallel input and output plug-ins as well as a sequential chain of processing plug-ins. With this design we also aim for high code reuse. When deploying software to cluster nodes the effort for reconfiguration and recompilation must be minimized. Therefore, we generate the software configuration using modules. Modules for common tasks such as data input or background subtraction can be easily reused in different applications.

Another important requirement is the automation and simplification of common tasks during development and evaluation of new algorithms. Testing an application in a real network of cameras and processing nodes usually means a lot of effort on deployment and configuration of all network nodes. Therefore, we provide facilities to remotely deploy software to all cluster nodes.

A prototype of our software suite has been implemented in C# under Windows .NET; a C++ version for improved performance is currently under development.

IV. Case Study and Preliminary Results

In a first case study, we have tested our software suite on multi-camera person tracking. The goal of this case study is to demonstrate the feasibility of our software suite in a single cluster. The cluster cameras have an overlapping FOV and are calibrated on the common ground plane. We deploy both standard cameras streaming raw data as well as smart cameras performing onboard analysis such as background subtraction and blob detection. To increase the robustness of the person detector the inputs of the overlapping cameras are used to generate a joint occupancy map [3]. The main processing modules for this case study are: background subtraction ($[BG]$), object detection ($[OD]$) and multi-view person tracking ($[MV]$). We have tested our software suite with a different set of overlapping cameras (1 to 4) as well as different types of cameras. In this paper, we briefly compare three settings.

Centralized Processing: This setting represents a traditional multi-camera network where all cameras stream their raw data to the central (cluster) node. The entire processing pipeline ($\langle [BG] \rightarrow [OD] \rightarrow [MV] \rangle$) is executed on the cluster node. Each camera requires a communication bandwidth of XXX MB/s; the central cluster node was able to process at most 2 input stream in real-time.

Smart Cameras with BG: In this setting background subtraction is performed on each smart camera and lower resolution differential images are transferred. Thus, only ($\langle [OD] \rightarrow [MV] \rangle$) is performed at the cluster node. The communication bandwidth for each camera was reduced to XXX MB/s; the cluster node performance was XXX.

Smart Cameras with BG and OD: In the third setting $[BG]$ and $[OD]$ is performed at the smart cameras, and only the bounding boxes of detected objects are transferred to the cluster node. The communication bandwidth for each camera was further reduced to XXX MB/s; the cluster node performance was XXX.

These three settings exploit the capabilities of smart cameras in different ways. By using our software suite the different settings were easy to deploy and to evaluate. Important performance parameters such as bandwidth, memory consumption and computing performance were automatically measured by the software suite.

Further work include ...

Acknowledgments

This research is funded by the Austrian Institute of Technology.

References

[1] Henry Detmold, Anton van den Hengel, Anthony Dick, Alex Cichowski, Rhys Hill, Ekim Kocadag, Katrina Falkner, and David S. Munro. Topology Estimation for Thousand-Camera Surveillance Networks. In *Proceedings of the ACM/IEEE Conference on Distributed Smart Cameras*, Vienna, Austria, 2007.

[2] Bernhard Rinner and Wayne Wolf. Introduction to Distributed Smart Cameras. *Proceedings of the IEEE*, 96(10):1565–1575, October 2008.

[3] Francois Fleuret, Jerome Berclaz, Richard Lengagne, and Pascal Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Transactions and Pattern Analysis and Machine Intelligence*, 30(2):267–282, 2008.