Demonstrating Autonomous Handover in Heterogeneous Multi-camera Systems

Jennifer Simonjan, Lukas Esterle, Bernhard Rinner Alpen-Adria-Universität Klagenfurt, Austria
Email: {firstname.lastname@aau.at}
Klagenfurt, Austria

Georg Nebehay, Gustavo Fernández Domínguez Austrian Institute of Technology, Austria
Email: {firstname.lastname@ait.ac.at}
Vienna, Austria

ABSTRACT

We have developed a self-aware multi-camera multi-object tracking application for distributed heterogeneous smart camera networks. This application is able to track multiple persons through the network whereby the cameras are able to transfer the tracking responsibility between each other in a self-organizing manner without any central control. This application runs on top of a fully distributed middleware, designed for extensibility and robustness as well as heterogeneous hardware and network technologies. Our demonstrator has been deployed in a laboratory environment consisting of different hardware platforms and network technologies.

1. INTRODUCTION

Distributed smart camera networks have been in the focus of research for several years [1, 2, 3]. Application domains range from traffic control, elderly and home monitoring, to surveillance of public areas. In practice, extending distributed smart camera networks after the initial deployment is not trivial. Due to the fast pace in technological progress, changes in hardware, operating systems and networking have to be considered when extending a distributed smart camera network. Thus, applications executed in smart camera networks need to be able to deal with those heterogeneities. A distributed middleware can help to tackle these challenges. The middleware as well as the applications on top should allow network extension and should be able to run on various hardware platforms with different network connections. To reduce resource consumption of the smart cameras in the network, only a single camera should track the object of interest. The camera with the best view of the object should be the one responsible for tracking. Thus, a handover is required to transfer the tracking responsibility to another camera, whenever the object leaves the field of view (FOV) of the camera currently tracking it.

In this paper, we present a distributed multi-camera tracking application, which is able to track multiple objects through a heterogeneous smart camera network. The main benefits of this application are manifold: the camera network (i) continuously tracks objects in a distributed fashion, (ii) is robust and adaptive to camera failure and newly added cameras, and (iii) is self-aware, with respect to the available resources and the current context, and therefore able to adapt to changes during runtime. The main objective of our demonstrator is to show self-awareness within smart camera networks.

The remainder of this paper is organized as follows: In Section 2 we describe our demonstrator, including the network architecture, the software modules and features. Some evaluation results are shown in Section 3. Section 4 briefly describes our planned ICDSC demo presentation and Section 5 concludes the paper.

2. THE DEMONSTRATOR

Our demonstrator is able to track multiple objects through a heterogeneous smart camera network and serves as case study for self-awareness in computing systems $http : www.epics-project.eu$. If a person leaves the FOV of the camera that is currently tracking it, the tracking responsibility is passed to the camera with the best view of the object. This process, known as handover, is based on an auction mechanism used to determine the camera with the best view of the object [4]. Upon the end of the auction, the tracking responsibility is transferred to the winning camera, which tries to continue tracking immediately. Further, the camera images and tracking information are periodically transmitted to a host PC, where they are visualized using a graphical user interface (UI).

The distributed publish-subscribe middleware system Ella [5] is used to facilitate this distributed tracking application on heterogeneous hardware. Since the application runs upon the middleware Ella, all modules are realised as publishers and/or subscribers.

2.1 Smart Camera Network

Fig. 1 shows the current setup of the smart camera network which consists of six cameras. Four of them are mounted in a laboratory room (cameras 1–4), with overlapping FOV. The other two cameras are placed outside in the corridor (cameras 5 and 6). This smart camera network was used to run and test the multi-object tracking application and the underlying middleware.

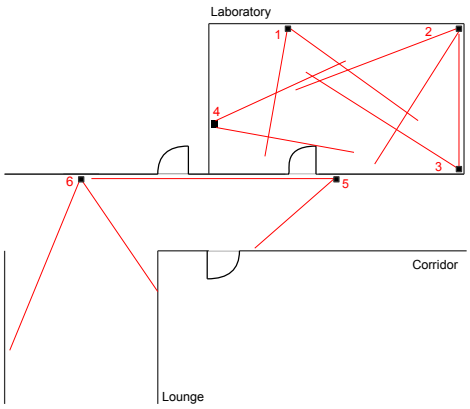

Figure 1: The deployment of our demonstrator. The smart cameras are depicted by a black square and the FOV is indicated by red lines.

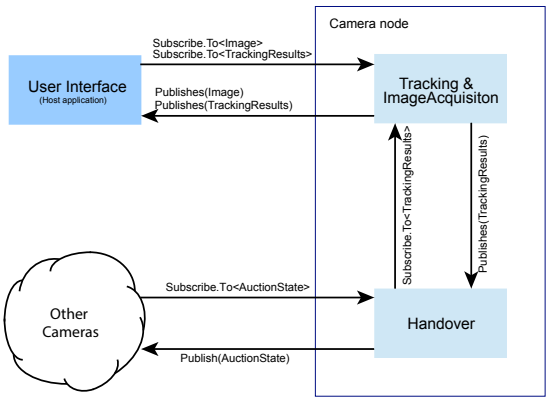

Figure 2: The modules of our application. The host PC runs a UI and each camera runs a tracking and a handover module.

Cameras 1 to 4 are connected via wired Ethernet, while the other two cameras are connected via WiFi. Our demonstrator is composed by different hardware platforms. Cameras 1 to 4 are Atom-based platforms (1.6 GHz processor, 2 GB RAM, 30 GB internal SSD hard disk) from SLR Engineering[1]. Cameras 5 and 6 are ARM-based platforms composed by Pandaboards[2] and Logitech C920 web cameras. The Pandaboard is an Open OMAP 4 mobile software development platform. The OMAP 4 is a System-on-a-Chip, featuring a dual-core 1,2 GHz ARM Cortex-A9 MPCore CPU. Both camera systems are Linux based and run Ubuntu 10.04 LTS.

The middleware and the demonstrator application, running on these six cameras, are implemented in C#.Net. C#.Net programs allow their execution on all major operating systems, supported by Mono.

2.2 Multi-object tracking application

The application is organised in three main modules: (i) image acquisition and tracking, (ii) handover, and (iii) user interface (UI). The interaction of these three modules is depicted in Fig. 2. All modules are realised as publishers or subscribers. A subscriber shows its interest in specific data by subscribing to any data of the corresponding type. Publishers transmit their data to all subscribers in the network for the corresponding type.

The UI visualises the video feed acquired by each camera as well as the tracking results, which are transmitted by the tracking module operating on each camera. The tracking results contain the location of the tracked object in the image, its size, and the tracking confidence. This information enables the user interface to draw a bounding box around the tracked object. Additionally, each camera operates a handover module responsible for deciding on the tracking responsibility. If the object leaves the FOV of the camera currently tracking it, the handover module recognizes that and initiates an auction for this object. Thus, the handover module of this camera becomes an auctioneer. During the auction, the auctioneer accepts bids from all cameras. Cameras determine their valuation of objects, and hence their bid, based on the respective tracking confidence. With the arrival of the first bid, an auction timer is started. The winner of the auction is determined as soon as the auction timer has elapsed. Thereafter, the tracking responsibility for the corresponding object is transferred to the camera with the highest bid. Since the application is capable of multi-object tracking, the user can choose additional objects to track.

2.2.1 *Camera module - Tracking and Image Acquisition*

The tracking module is responsible for both, acquiring images as well as tracking objects within the field of view. Additionally, the tracking module transmits images and tracking results to other interested components in the system (i.e.: UI). After start-up of this module, it begins to periodically transmit images. It manages a list of all objects, the corresponding camera is currently tracking. Every time the user chooses a new object, the user interface informs the tracking module of the corresponding camera about the object to be tracked by transmitting a model description. Upon receiving this model description, the tracking module starts tracking and transmitting tracking results.

The tracker itself is implemented in C++ and is embedded into this module as native code. On each camera, tracking is realized by performing an association between the objects that were selected in the user interface and candidate objects in the current frame, which are identified by performing a foreground-background segmentation. The tracking approach is based on the method described in [6].

The association between templates and candidates is established by interpreting the problem of associating templates and candidates as a transportation problem, where the distances between the respective feature vectors are the transportation costs and the goal is to minimize all transportation costs. This problem can be solved optimally by employing the well-known Hungarian algorithm. Additionally, the reciprocal of the transportation cost for a successful association is reported to other components as the tracking confidence.

2.2.2 *Camera module - Handover*

The main goal of the handover module is the coordination

9[1]http://www.slr-engineering.at/smart_camera
9[2]http://www.pandaboard.org/

of tracking responsibilities for objects of interest in order to track them continuously and in a robust fashion within the network of distributed smart cameras.

During a handover, an auction is performed, as described by Esterle et al. [4]. The camera, having lost an object, becomes an auctioneer and solicits bids from all available cameras in the network. This solicitation contains a description of the object enabling other cameras to search for the object of interest. The advertisement is sent out when the current tracking confidence of a tracked object falls below a certain threshold. This happens whenever the object is going to leave the FOV of the camera. The bidders submit their instantaneous tracking confidence as bids; the bidder with the highest bid wins the auction and thus the tracking responsibility.

To reduce the network traffic, each camera builds a vision graph during runtime to learn about its direct neighbours. The handover module can then decide to inform just a few of the cameras about the auction. The vision graph assigns each neighbour a link strength. Taking inspiration from ant foraging and their employed pheromones, whenever a camera wins an auction the link strength (artificial pheromone) between the auctioneer and the winning camera is increased. Moreover, the pheromones evaporate over time in case no handover occurs. Using this information, each camera knows which of its neighbouring cameras most often won the auction and is likely to win the auction again. The handover module can then decide to just inform the cameras with a high link strength of an auction, since one of those will win with high probability. In a network with a high number of cameras this mechanism can reduce the amount of message drastically. A detailed description of how this vision graph is build can be found in [4].

2.2.3 *Host module - User Interface*

The user interface runs on the desktop computer and receives images and tracking results of all available cameras. Upon the arrival of an image, the user interface visualises it. For every camera, a new window opens to show the captured images of the corresponding camera. The user can choose one or more objects to be tracked, by drawing a bounding box around the desired objects into the UI. Afterwards, the UI informs the tracking module of the corresponding camera by sending a message which contains the selected model. The tracking module receiving this message immediately starts with the tracking process and transmitting tracking results.

3. EVALUATION RESULTS

Table 1 shows an overview of the average CPU and memory usage on each hardware platform in the network, doing single and a multi object tracking. According to this table, there is no significant difference on the performance when doing single or multi object tracking. Regarding the CPU utilisation, the SLR cameras perform a little better than the Pandaboards. However, the Pandaboard systems are designed for energy-efficient usage and consume only 3 Watts. Compared to that, the SLR cameras consume up to 20 Watts, which means that developers have to find the best trade-off between computational power and energy consumption for their application.

4. ICDSC DEMO SETUP

Hardware	Tracking mode	avg. CPU Utilisation	avg. Memory Usage
SLR Camera	Single object	78,50%	$5{,}89*10^7$ B
SLR Camera	Multi object	77,19%	$7{,}07*10^7$ B
Pandaboard System	Single object	81,38%	$8{,}06*10^7$ B
Pandaboard System	Multi object	84,37%	$8{,}1*10^7$ B
Desktop PC	UI	2,53%	$6{,}5*10^7$ B

Table 1: Overview on the performance measurement

For the ICDSC demo we plan to present the completely functional multi-camera multi object tracking application which performs the following tasks: (i) tracking of objects in a single camera, (ii) tracking of multiple objects autonomously in a network of distributed smart cameras, (iii) self-awareness in the context of network congestion, and (iv) robustness and adaptivity in terms of added and removed cameras from the network. To meet the challenge of heterogeneity, we will track multiple objects through a smart camera network consisting of four SLR Engineering cameras and two Pandaboard based web cameras. All tasks will be shown via remote connection to the laboratory at the Alpen-Adria-Universität Klagenfurt and streaming of the contents to the conference location.

5. CONCLUSION

In this paper, we have presented the multi-camera multi-object tracking demonstrator. Our demonstrator supports various hardware platforms using different network technologies. Possible future extensions can include an autonomous initialization of objects to be tracked, an implementation of the user interface for portable devices such as tablet PCs and smartphones, or increasing the number and heterogeneity of network nodes.

Acknowledgement

This work is supported by Lakeside Labs GmbH, Klagenfurt, Austria and is funded in part by the European Union Seventh Framework Programme under grant agreement n° 257906.

6. REFERENCES

[1] M. Reisslein, B. Rinner, and A. Roy-Chowdhury, "Smart camera networks [guest editors' introduction]," *Computer*, vol. 47, no. 5, pp. 23–25, 2014.

[2] H. Aghajan and A. Cavallaro, *Multi-camera networks: principles and applications*. Academic press, 2009.

[3] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf, "The evolution from single to pervasive smart cameras," in *Proceedings of the second ACM/IEEE International Conference on Distributed Smart Cameras 2008. ICDSC 2008.*, 2008, pp. 1–10.

[4] L. Esterle, P. R. Lewis, X. Yao, and B. Rinner, "Socio-economic vision graph generation and handover in distributed smart camera networks," *ACM Transactions on Sensor Networks*, vol. 10, no. 2, 2014.

[5] B. Dieber, J. Simonjan, L. Esterle, G. Nebehay, R. Pflugfelder, G. Fernandez, and B. Rinner, "Ella: Middleware for Multi-camera Surveillance in Heterogeneous Visual Sensor Networks," in *Proceedings*

of the Seventh ACM/IEEE International Conference on Distributed Smart Cameras, Palm Springs, USA, 2013.

[6] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–577, May 2003. [Online]. Available: http://dx.doi.org/10.1109/tpami.2003.1195991