Distributed Smart Cameras for Hard Real-Time Obstacle Detection in Control Applications

Herwig Guggi
Pervasive Computing Group
Institute of Networked and Embedded Systems
Klagenfurt University
Email: herwig.guggi@aau.at

Bernhard Rinner
Pervasive Computing Group
Institute of Networked and Embedded Systems
Klagenfurt University
Email: bernhard.rinner@aau.at

***Abstract*— This paper describes the integration of distributed image analysis in a smart camera network with a control application. The distributed smart cameras analyze the environment of the controlled object and transfer accurate position and orientation information of all detected obstacles to the controller within guaranteed time bounds. With this information the controller can optimize the trajectory of the load. We present two different approaches for estimating the bounding boxes for detected obstacles and compare their benefits and drawbacks.**

***Index Terms*—Smart camera networks; hard real-time; obstacle detection;**

I. Introduction and Motivation

Fast and reliable obstacle detection is important for many real world control applications where the controller determines the behavior of the controlled object based on the observed internal state. In this work we use multiple cameras to observe the *environment* of the controlled object with the goal to acquire information of the environment's current state. The controller exploits this *auxiliary* information, i.e., the position, shape and velocity of obstacles, to optimize the control behavior. The delivery of this information within guaranteed time bounds—in *hard real-time*—is an important precondition for the control optimization. Thus, our ultimate objective is to integrate real-time image analysis, adaptive motion control and synchronous communication between the imaging and control subsystems.

In this work, we deploy distributed smart cameras [1] to monitor the environment of a model crane. The smart cameras perform local image analysis to identify the shape of potential obstacles in their field of view (FOV). Shape information from multiple cameras is then fused together to estimate the 3D structure of the obstacles. The smart cameras are connected over a time-triggered network with the crane's control system and deliver the position and velocity of all obstacles within the predefined frame rate. By knowing the position of obstacles, the controller can adapt the trajectory of the payload appropriately and increase the safety and efficiency of the crane system.

This paper expands our previous work on such integration of distributed image analysis, communication and control in a hard real-time setting [2]. Our contribution comprises the distributed real-time image processing in the smart camera network, i.e., the image analysis, the obstacle reconstruction, the object motion prediction and the communication with the control system. Other aspects of this integrated approach such as robustness and trajectory planning can be found in [3], [4], [5] and [6]

The remainder of this paper is organized as follows. Section II sketches related work. Section III describes the system overview, the components of the system and the logical dataflow of the distributed smart camera application. In section IV we present details about the algorithms that are executed in this setup and in section V we provide information about timing, accuracy and delay of the system and single components. Section VI concludes the results of the paper and section VII shows suggestions for speed improvements of the system based on the measured data.

II. Related Work

There exists a large body of research in vision-based object (and obstacle) detection. In the following we focus our discussion to real-time detection on multiple cameras.

Wang et al. [7] introduced a scalable peer-to-peer wireless smart camera system for detecting and tracking multiple objects. Each camera autonomously performs tracking of objects in the local FOV and record object information in its local storage. Short messages are used to exchange label and position of objects between cameras with a joint view. The network is evaluated by tracking remote controlled cars and by defining regions-of-interest for raising events.

Henrich et al. [8] presented a multi-camera system which is used to guide a robot in a region with moving obstacles. Obstacle detection is performed with four calibrated cameras connected to a single computer where the position and structure of the obstacles are calculated. After the planning, each configuration (movement step of the robot) is tested for collision with obstacles. If a collision is detected, the trajectory is re-planned. The paper provides no information about the update rate of the scene or if images are captured while the robot is moving.

Lampert [9] presented a high-speed multi-camera system that is able to detect colored objects in cluttered scenes. They demonstrated a robotic table tennis deployment to estimate ball trajectories through 3D space simultaneously from four cameras images at a speed of 200 fps. However, their image

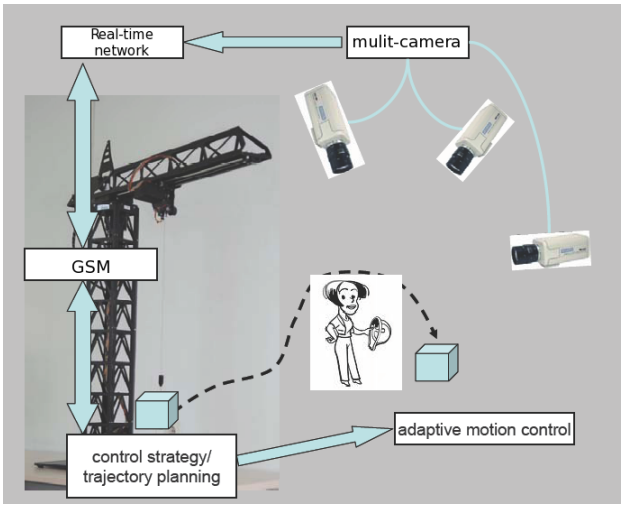

Fig. 1. System architecture and high-level data flow

processing is purely centralized, and the system has been implemented on commercial-off-the-shelf hardware (standard PC with GPU).

Work by Ladikos et al. [10] used a similar setup where four cameras are connected to a PC. This PC calculates the 3D model with the help of a GPU (graphics processing unit). Potential collisions are checked based on bounding boxes around detected obstacles. To prevent collisions these bounding boxes are extended by a security distance. The cameras operate at a resolution of 1024×768 and at a frame rate of 30 fps.

III. System Overview

Figure 1 depicts the system overview as well as the high-level data flow. The main components include the multi-camera subsystem, the control subsystem (including the crane with its sensors and actuators), the real-time network for connecting the components, and a ground state monitor (GSM). The distributed smart cameras analyze the crane environment, i.e., the cameras perform object detection individually and fuse this local information within the camera network to determine the obstacle's position and velocity in the 3D space. This fused information is checked for plausibility by the GSM to increase the reliability of the overall system. In case of an implausible state the scene analysis is re-triggered—or if re-triggering is not possible the whole system is re-started. The checked position and velocity data of all detected obstacles are then transferred to the control subsystem over the real-time network.

A. Multi-camera subsystem

The model crane has a height of approx. 1.3 m and its jib has a length of approx. 1 m. This results in an operation region of the crane in form of a cylinder with radius and height of about 1 m (i.e., only a cylindrical segment with angle of 270° is accessible by the crane). The ground plane of the observation area is 2 m × 2.25 m which we consider as the observation space. The model crane is operated in an indoor environment with stable ambient light.

The arrangement of the cameras is important for deployment. We mounted three cameras to cover the observation space of the crane model, providing three different object views. In this setup, almost the entire ground plane is in the FOV of all cameras. Due to the available space in the laboratory, the cameras were mounted at a rather high position of 1.5 m above the ground plane resulting in a downward angle of about 60°. A mounting at a lower height would lead to a more accurate height estimation.

The processing in the multi-camera subsystem is partitioned into two parts: the local pre-processing and the fusion processing. During the pre-processing the images from the cameras are analyzed. In this process the contours are abstracted from the obstacles. These are then forwarded to be used by the fusion process. The fusion part estimates size, position and orientation of the obstacles. The pre-processing is executed on each camera while the fusion is executed only on one of the distributed cameras.

B. Control subsection

The task of the control subsection is to autonomously move a load from a starting point to a destination point. This paper will only give a rough overview on the processes that are executed in the control system. For more details about this system, see [5] and [6]. The trajectory is planned based on some optimization goal before the movement of the load is started. If an obstacle is detected—either before or during the movement process—it is checked whether the obstacle intersects the planned trajectory. If so, the trajectory is re-planned by the controller in order to avoid a collision.

The control system is responsible for calculating a trajectory from the start-position to the destination. The position and current orientation of the crane (position of trolley, angle of jib, ...) is measured using sensors. These sensor values are only available to the control system. The other components provide detailed information about obstacles that are present in the environment space of the crane. This additional information is used by the planner to find a trajectory that is low-cost - in the sense of timing or power consumption - and avoids collisions with obstacles that intersect the optimal path. The control subsystem is based on a real-time work station equipped with an Intel Core 2 Duo with 2.13 GHz CPU and 1024 MB RAM. The smart cameras and the control subsystem are connected via TTEthernet [11].

C. Real-time network

Another part of the system is the real-time communication network which has to guarantee that every data packet is transferred within a strict period of time. The data is transferred using the TTEthernet protocol [11] which extends classical Ethernet with services based on time-triggered protocols to meet time-critical, deterministic or safety-critical requirements. A dedicated software driver was developed to support TTEthernet on our smart cameras.

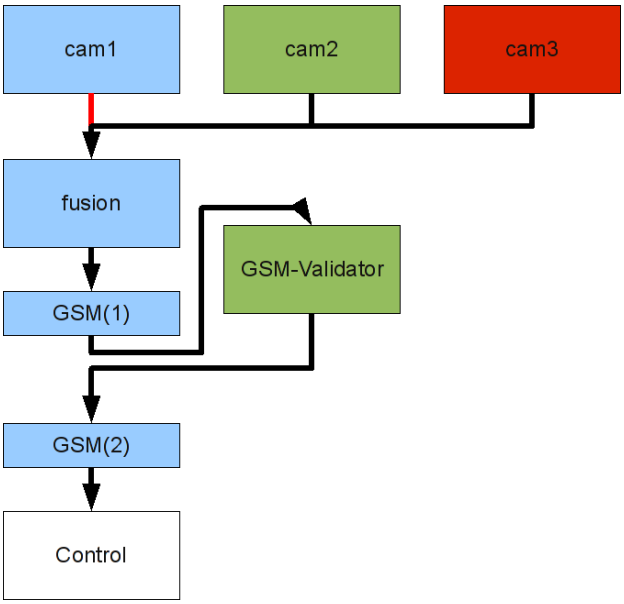

Fig. 2. System architecture (logical connections) The black lines represent the connections using TTEthernet, where the red line represents a connection using the local loopback-device.

D. Data flow

Figure 2 depicts the data flow of the overall system. The black lines represent a TTEthernet connection where the red line represents a local loop back connection. The colors of the boxes represent the cameras where the application modules are executed. Blue corresponds to camera 1, green to camera 2 and red to camera 3. The white square represents the control system including the crane. Each camera performs the pre-processing and provides the result to the fusion unit. The fusion unit reads the dataset and estimates the 3D scene. As soon as the estimation is completed, the GSM (Ground State Monitor) forwards the current ground state of the fusion unit to the GSM-Validator. Not every detail of the GSM is described in this document. The ground state consists of the transmitted data plus some timing information. The GSM-Validator checks the ground state of the fusion unit for plausibility by comparing the received output with the expected output. The expected output is calculated based on the previous ground-states. If the current output seams reasonable, the validator informs the GSM (2) that the estimated scene information can be forwarded to the control system. If the output is not as expected, the GSM has the option of restarting the fusion unit. This is done by pre-loading an expected ground state. Another important function of the GSM is the option of re-starting the fusion unit if the software fails for any reason. Hence, the GSM operates as a filter for the data and a monitoring device for the fusion process. More details about the GSM can be found in [3] and [4].

We use custom-built smart cameras from SLR Engineering which are equipped with an Intel Atom processor running at 1.6 GHz and an 100 MBit Ethernet interface, see figure 3. The camera has a CCD image sensor with a native resolution of

Fig. 3. Image of the used smart camera from SLR Engineering

1360×1024 pixels. The same cameras are also used for traffic surveillance, see [12].

IV. Collaborative processing in distributed smart cameras

The implemented system relies on following assumptions. All cameras are calibrated and their extrinsic and intrinsic parameters are known (see section III). All obstacles that have to be detected have cuboid structure and are placed on the ground plane (plane with real-world height 0). For the scene analysis a perfect object reconstruction is not required; the detected obstacles must only be described by bounding boxes.

A. Local pre-processing

The local pre-processing captures new frames and detects the obstacles and abstracts their contours. Obstacles are segmented by background subtraction. The contours of these segments are then represented by a list of corner points. This list is forwarded to the fusion unit. Each scene (set of contours) is transmitted within a single Ethernet frame (1500 Bytes). Each contour is represented as a polygon with a variable number of corner points.

As the number of points for a contour is limited by the size of an Ethernet frame, it may happen that in complex scenes, the contour representation has to be simplified. The simplification works in a way that the points of the polygons that have the smallest impact on the shape of the contour are removed from the polygons. This way the quality of the representation is dynamically fitted to the number of contours. Even contours with a small number of corner points are sufficient to perform a reasonable reconstruction of the objects in the fusion step. Note that a perfectly segmented cuboid would result in general in a contour of six corner points.

The quality of a polygon representation is defined by the maximum distance between any point of a segmented contour and the abstracted polygon. The maximum distance is set to 6 pixels, a lower distance would lead to slower processing with no further improvement of the results in later phases.

B. Fusion

The fusion process grabs the preprocessed scene data from the single cameras. These scene representations are analyzed and bounding boxes of detected objects are generated.

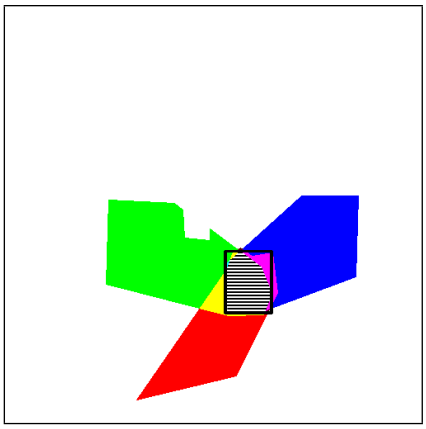

Fig. 5. Combination of the projections from the single cameras to a combined view. Blue: camera1, Green: camera 2, Red: camera3. The region where all three overlap can be determined as ground area of the object

The estimation of the obstacles' bounding boxes is performed in two steps. The first step determines the ground area of the obstacles. The detected ground area is used to calculate the height in a second step.

1) Ground area determination: The ground area can be assumed as the part of the object which overlaps in the projection of the contours to the ground plane. With the help of the ground area it is already possible to determine the length, the width, translation (in x and y) and the orientation of the object.

For determining the ground area of the objects, the contours from all cameras are projected to the ground plane. This mapping is done based on the intrinsic and extrinsic camera parameters which are provided by each camera. Figure 4 shows an example of a single scene where one object is present. This object is detected by all cameras. The contours mapped to the ground plane are displayed for each camera. Figure 5 shows the combined view of all cameras and the estimated ground area is visualized as black rectangle.

In figure 4 the image in the center (the one of camera 2) has an edge on the upper part of the object. This part was falsely classified as background. It is actually a part of the object. Effects like these happen due to poor segmentation. To further explain this effect, the polygon was drawn to the background image of camera 2. Figure 6 shows that the black stand of the crane is the reason for the miss-classification.

As there might be regions of the ground plane which cannot be seen by any camera (as this part is occluded by the object, there might be a wrong estimation of the ground area. However, this estimation will always be larger than the real ground area. This means that a perfect representation is not possible. Figure 7 shows the top view of a scene where an object (blue) is detected by two cameras. The contours of these two cameras (green and red) overlap in a region that is larger than the ground area of the object.

2) Height calculation: For the height calculation, two different approaches have been implemented and tested. Both implementations rely on the estimated ground area.

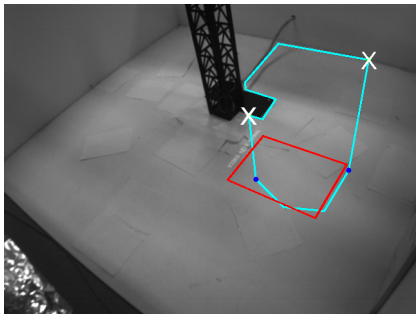

Fig. 6. Contour of camera 2 projected to background image from camera 2. (The background image is not available for the fusion unit). red: Projection of the ground area to the image, blue: detected points on ground plane for height calculation, white crosses: corresponding points in the height, turquoise: contour received from camera 2

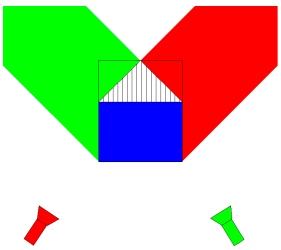

Fig. 7. Top view of scene: An objects ground area is miss-detected in the region where the ground plane is occluded from the cameras (the hatched area)

The first implementation is based on vector intersection. The height is determined by calculating the height of the point where the vector from the camera center to the projection of the "head point" (white cross in figure 8) intersects the vector in z direction from the projected head point on ground plane ("foot point", i.e., blue point in figure 8). Since we assume a cuboid structure of the obstacles, head point and foot point are corner pairs of the cuboid. Each contour offers two chances to calculate the height this way, since in general there is a corner pair on the most left and right edge of the contour. It turned out that in some cases, these vectors do not show the correct result as they might have an edge due to bad contour detection (see figure 6, left - white cross) or due to miss-detection of the point on ground (see Figure 8, right - blue point). We used the maximum calculated value as effective height of the object.

The second approach relies on the fact that all contours represent boxes with cuboid shape with already known ground area. To calculate the height, a cuboid with the known length and width and different height values is generated and mapped to the contour. The mapping that covers most of the contour is suggested as the best assumption for the height. Figure 9 shows example images. The top left image shows a contour (blue) and a projection of a model (green) with a small height value. The height of the model is increased until the contour is well covered by the projection of the model (bottom right).

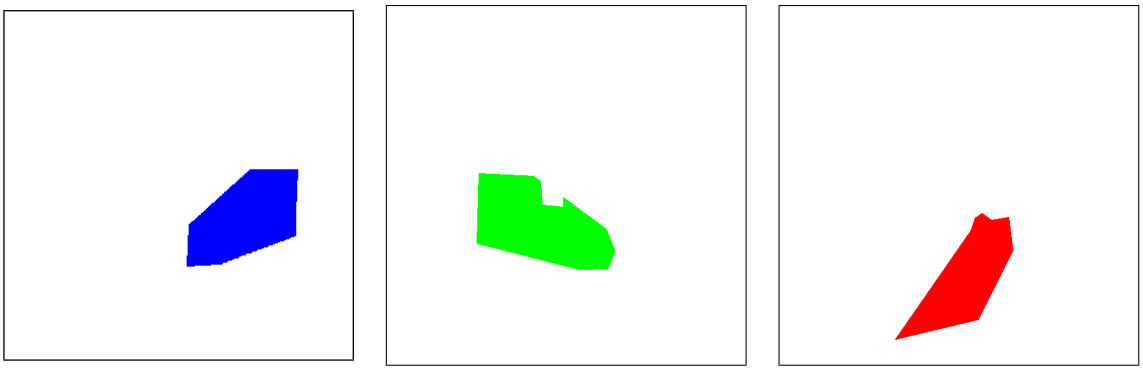

Fig. 4. Projection of contours to ground plane left: camera 1, middle: camera 2 and right: camera 3

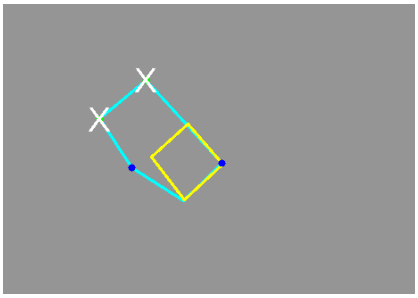

Fig. 8. False detection of the point on the ground. As the line along the ground area has no edge at the ground point at the back, the ground point is assumed on the wrong position. This leads to a too high calculation of the height of up to 50% note: this figure is not from the scene in Figure 5

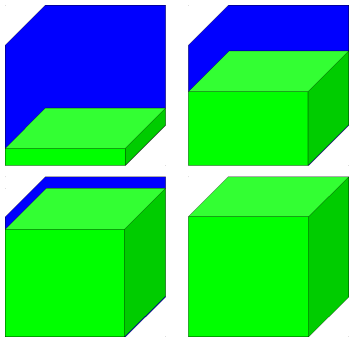

Fig. 9. Example for model fitting: Contour (blue), model with different heights (green)

V. Results

The presented results focus on the collaborative processing in the distributed smart camera system. All other components (such as control unit and ground state monitor) are not considered here.

A. Single camera application

In [2] we presented preliminary results on single camera processing where we used a different resolution and contour algorithm. This method turned out to be too inaccurate to reconstruct 3D objects.

In the current implementation, the cameras were set to a resolution of 640×480 pixels. The contours are represented by bounding polygons as described in section IV-A. The time measurements were taken from the camera where also the fusion process is executed. This means that this is the worst case situation. The other cameras execute within a shorter time period. A measurement of 200 scenes (frames from each camera) showed that the maximum processing delay is 70 ms. With this setup it is possible to handle more than 10 contour lists per second to the fusion process.

B. Fusion process

In section IV-B the two different methods for calculating the bounding contours of the objects are described. Both methods have the determination of the ground area in common. The ground area estimation is performed in less than 2 ms. The accuracy is depending on the location of the obstacle and the placement of the cameras, but lies within 3 to 5 cm. This value refers to the accuracy of the position and the dimensions (width and length) of the objects.

The two methods for calculating the height differ mainly in two aspects. The first aspect is the accuracy and the second aspect is the calculation speed. Including the time for the detection of the ground area, the first method, the calculation of the height using the vector of the contour that represents the height is executed within less than 10 ms. The achieved height results were always equal or larger than to the true height. In some cases, however, a large height error of about 50% was estimated which corresponds to an absolute error of more than 40 cm. This is caused by adverse orientations of the object (see figure 8) and our conservative height estimation, i.e., selecting the largest estimate.

The second method for calculating the height resulted in an accuracy of 5 cm for the height but has the drawback of a slower execution. Each detected ground area is tested against all heights starting with 0 cm up to 1000 cm. The step-size was chosen to be 3 cm. If a contour has already been detected by a different camera, the already detected height is chosen as starting-value. During a series of more than 400 measurements with a contour coverage of up to 25% of the whole field of view of the camera, the algorithm took up to 200 ms to calculate the scene. The median time for calculation was 66 ms and the standard deviation was 27.9 ms. Further analysis of the

timing values showed, that the most time consuming part was the transformation of the real-world 3D model to the single camera views.

C. System performance

The main parameters that describe the whole performance of the system are the delay from the change of the environment to the notification of the control system and the update frequency. The delay depends on the the slowest path for data calculation. When looking at figure 2, this path starts with the local pre-processing on the 3 cameras. This takes a maximum of 100 ms. The next path for the data is the transmission to the fusion unit. Due to real-time communication (TTEthernet), this time is guaranteed to be limited to 20 ms. The next component is the fusion unit. The maximum delay of this unit depends on the implementation. As shown in this paper, the first approach finishes within a maximum time of 10 ms. The alternative (model-fitting) takes up to 200 ms. The final transmission to the control (including the path via the ground state monitor) takes another 50 ms. This results in a maximum delay of 180 ms for the first method for height calculation and 370 ms for the model fitting approach.

The worst case update time depends on the time and frequency of the slowest component. Using the fast implementation for the fusion unit, the distributed pre-processing is the slowest component and the update time is 100 ms (10 Hz) in the worst case. In the second case (model fitting), the fusion process is the slowest component and the worst case update time is 200 ms (5 Hz).

VI. Conclusion

This paper shows the results of a 3D obstacle detection with the use of Intel Atom processors. Two different implementations are described for the height calculation. The local pre-processing is done in a distributed manner and the data-size is reduced to a single Ethernet frame per image (1500 Bytes). This pre-processing is executed within less than 100 ms for the worst case.

For the fusion process, two different implementations are shown. One implementation offers high speed with an accuracy of 3 cm for the ground area and an error in height of some times more than 50% (or 40 cm) of the obstacle height. In any case the height is larger than the real obstacle height. The second implementation offers an assumption of the ground area with an equal accuracy of 3 cm. The height is calculated more exactly and the accuracy is about 5 cm.

With the use of more powerful hardware like GPUs, FPGAs or DSPs, it is still possible to reach better results in both aspects, the local pre-processing and the fusion part.

VII. Future work

The provided results for the two algorithms show that one algorithm is relatively fast for the height calculation where the other is more accurate.

There are a number of ways to reduce the height calculation time. One way is to combine the two algorithms. The first algorithm is used as described in this paper, where the second algorithm is modified to use a top-down instead of a buttom-up approach. The fast algorithm is used to determine the upper bound for the height of the object. This upper-bound is then used as a starting point for the second algorithm to obtain a more accurate result.

Another improvement would be to replace the second (slow) algorithm by an adaption of a binary search. The binary search algorithm works in a way that the search-space is cut into halves and it is checked which half includes the result. The half that includes the result is then again cut into two halves. This process continues recursively (always taking the new boundaries into consideration) until a defined accuracy is reached. This binary search algorithm can only be applied, if the upper bound is known. Therefore the first algorithm is again used to determine the first estimation for the height of the object.

Acknowledgment

This work was supported by the Austrian Science Promotion Fund (FFG) under grant 819482 and by Lakeside Labs GmbH, Klagenfurt, Austria and funded by the European Regional Development Fund (ERDF) and the Carinthian Economic Promotion Fund (KWF) under grant KWF 20214/18354/27107.

References

[1] B. Rinner and W. Wolf, "A Bright Future for Distributed Smart Cameras," *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1562–1564, October 2008.

[2] H. Guggi and B. Rinner, "Distributed Smart Cameras for Hard Real-Time Control," in *Proceedings of the ACM/IEEE Conference on Distributed Smart Cameras*, Atlanta, USA, 2010, pp. 1–4.

[3] V. Mikolasek and H. Kopetz, "Roll-Forward Recovery with State Estimation," in *14th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing*, Mar. 2011.

[4] V. Mikolasek and M. Zolda, "Towards Distributed Robustness in Embedded Systems," *Dependable Systems and Networks*, Jun. 2009.

[5] K. Graichen, M. Egretzberger, and A. Kugi, "Ein suboptimaler Ansatz zur schnellen modellpraediktiven Regelung nichtlinearer Systeme," in *at - Automatisierungstechnik*, vol. Vol. 58, no. No. 8, 2010, pp. 447–456.

[6] ——, "Suboptimal model predictive control of a laboratory crane," in *Preprints of the 8th IFAC Symposium on Nonlinear Control Systems*, 2010, pp. 397 – 402.

[7] Y. Wang, M. Casares, and S. Velipasalar, "Cooperative Object Tracking and Event Detection with Wireless Smart Cameras," in *Proc. Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS '09)*, Sep. 2–4, 2009, pp. 394–399.

[8] D. Henrich and T. Gecks, "Multi-camera collision detection between known and unknown objects," in *2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2008)*, vol. 2. Stanford/USA: ACM/IEEE, Sept 7-11 2008, pp. 1–10.

[9] C. H. Lampert and J. Peters, "Real-Time Detection of Colored Objects In Multiple Camera Streams With Off-the-Shelf Hardware Components," *Journal of Real-Time Image Processing*, pp. 1–11, 2010. [Online]. Available: http://springerlink.com/content/x328117h1557076v/fulltext.pdf

[10] A. Ladikos, S. Benhimane, and N. Navab, "Real-Time 3D Reconstruction for Collision Avoidance in Interventional Environments," in *Proceedings of the 11th International Conference on Medical Image Computing and Computer-Assisted Intervention, Part II (MICCAI '08)*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 526–534.

[11] www.tttech.com.

[12] F. Pletzer, R. Tusch, B. Böszörményi, B. Rinner, O. Sidla, M. Harrer, and T. Mariacher, "Feature-based Level of Service Classification for Traffic Surveillance," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2011.