Development of an Intelligent Omnivision Surveillance System

Hannes Bistry and Jianwei Zhang

Department of Informatics, University of Hamburg
Vogt-Koelln-Strae 30, 22527 Hamburg, Germany
{bistry,zhang}@informatik.uni-hamburg.de
http://www.informatik.uni-hamburg.de

Abstract. This paper describes an innovative intelligent omnivision video system, that stitches the images of four cameras together, resulting in one seamless image. This way the system generates a 360 degree view of the scene. An additional automatically controlled pan-tilt-zoom-camera provides a high resolution view of user defined regions of interest (ROI). In addition to the fusion of multiple camera images, the system has intelligent features like object detection and region-of-interest detection. The software architecture features configurable pipelines of image processing functions. It is easily possible to rearrange the pipeline and add new functions to the overall system. The pan-tilt-zoom camera is controlled by an embedded system, that has been developed for this system. GPU-accelerated processing of elements allows real-time panorama stitching. We're showing the application of our system in the field of surveillance, but the system can also be used for robots.

1 Introduction

For many applications, a wide view of the scene yields significant advantages. In the field of surveillance, wide angle and omnidirectional camera systems can provide all significant information in one video-stream. Having one video stream is much easier to supervise for the staff than having multiple streams.

The challenge of the research work was to set up a surveillance system for the use on a ship. The system has to be installed on a pole on board of the ship and shall provide an omnidirectional view of the scene. The image of a pan-tilt-zoom-camera shall be overlayed onto the video stream in order to display image details of a region of interest. In addition to that, intelligent feature detection algorithms will be implemented into the system.

In our previous research work we have focused on intelligent cameras for robot systems ([1, 2]) and surveillance ([3]). Due to the underlying architecture that we developed, it is easy to set up image processing systems and reuse functions that have already been implemented([4]).

The remainder of this paper is organized as follows: In section 2 we introduce approaches of related research projects. Section 3 gives an overview about the developed system, the hardware setup and the software architecture. In section 4

we introduce the special features of the system and show experimental results. A conclusion on the achievements and an outlook to future research is given in section 5.

2 Related research

There are different approaches of generating wide angle images using digital cameras. One possibility, that is also applied in the system described in this paper, is to use multiple cameras and to stitch the images together into a panoramic image. The FlyCam is a ring of five inexpensive colour cameras ([5]). Each of the images is transformed and the tiles are combined to a panoramic image. At the borders of the image, a cross fading algorithm is applied. In this research project, the resolution of the cameras has to be reduced due to the computational load.

Another method of generating panoramic images is the use of hyperboloidal mirrors ([6]). One camera is placed below the mirror. It is possible to create a seamless panoramic image by back-transformation, but the resolution is quite limited.

A surveillance system based on a fish-eye lens is described in [7]. The system features DSP-based processing and an advanced background subtraction algorithm.

Many research projects focus on adding intelligent functions to surveillance systems. In [8] a surveillance system that features real-time detection and tracking of multiple people is introduced. The authors showed their approach of classifying the activities of the people. Objects that are carried and placed by the people can also be detected. In the evaluation, the system achieved real-time performance on a regular PC. People tracking is also a well known topic in the field of robotics ([9–11]).

There are also research projects on the system architecture of image processing systems. One example - not directly related to surveillance - is shown in [12]. A framework is presented where image processing tasks can be configured in a graphical editor and can be carried out on distributed systems. During runtime, the execution time of the steps is monitored, and considered in the distribution of tasks. As an application, a setup for depth reconstruction using a stereo camera system is presented.

[13] introduces a system, where multiple image processing tasks are carried out on the data stream from a camera (face detection, object detection, background subtraction, compression). Using classical scheduling, not all tasks can be carried out in time. The authors propose an approach, where the different tasks are classified according to the maximum acceptable latency. During test runs, the execution time is measured and the system tries to find a scheduling strategy that meets the requirement of all tasks.

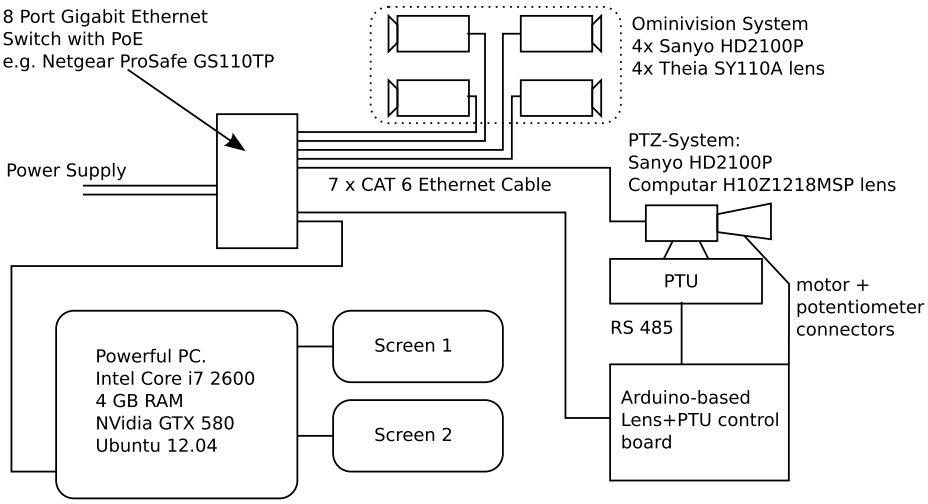

Fig. 1. Hardware Setup of the Intelligent Omnivision System. The five cameras, the embedded system for PTZ control and the control PC are connected to a Power-Over-Ethernet-Switch. This way only (neglecting power supply) one network cable need to be installed between the control PC and the rest of the hardware. The link between the embedded system and the pan-tilt-unit is a serial RS485 connection. The zoom-lens provides connections to the zoom/focus motors and potentiometers; these are also connected to the embedded system.

3 System Setup

This chapter describes the system setup in terms of hardware and software setup.

Description of Hardware The setup of the system is shown and explained in figure 1. The 360 degree image is captured by four IP-cameras (Sanyo HD2100P). Each of these cameras covers a $> 90°$ field of the scene.

As all cameras provide compressed video streams, there are no bandwidth issues, as the Gigabit uplink to the PC can easily handle 5 video streams and control data.

Due to certain image quality and zoom range requirements, it was necessary to use a FullHD IP camera and a zoom lens. For this combination there is no ready-to-use solution for zoom and focus control. An embedded system has been designed to control the zoom lens as well the movement of the pan-tilt-unit. There are few zoom lenses for the C-mount standard featuring motorized zoom and focus control. One of them is the Computar H10Z1218MSP, which features a focal range of 12 to 120 mm, motor driven zoom and focus, and potentiometers to measure the current zoom and focus setting. This way it becomes possible to implement a system that approaches a desired setting automatically.

The embedded system is based on an Arduino Ethernet microcontroller, extended by custom hardware. The motors are connected using an H-bridge circuit

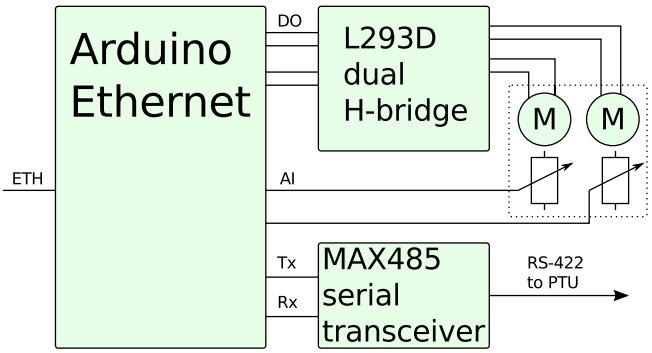

Fig. 2. This figure shows the structure of the embedded system for PTZ control. In the figure the power supply of the different components is omitted for a better clarity. (ETH=ethernet, DO=digital output, AI=analogue input, Rx= serial receive, Tx=serial transmit)

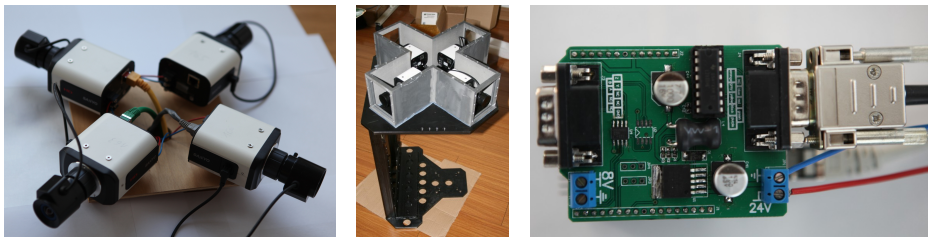

Fig. 3. These images show a test setup of the four cameras (left), the housing (middle, designed by Kunshan Robotechn Intelligent Technology Co.), and the embedded system for PTZ control (right, circuit developed by Johannes Liebrecht, board layout by Kunshan Robotechn Intelligent Technology Co.)

and the potentiometers are read out using the analogue input ports of the microcontroller.

One additional feature of the embedded controller is the possibility to connect a PTU via a RS485/422 based serial interface. Therefore it is not necessary to have a direct serial connection between the control PC and the PTU. The commands can be send via Ethernet, so the cabling effort will be reduced. In order to support the RS485/422, an additional transceiver chip is integrated in the embedded system. On the control PC a virtual COM port is created. The PTU control software can access the PTU just as if it was connected directly to the PC.

The structure of the embedded system is shown in figure 2. Images of the cameras, the housing and the developed embedded system are shown in image 3.

Software Setup For the development of the intelligent omnivision camera system we are using a lot of modular processing functions developed for the use in smart camera systems and service robots ([4]). This is possible, since all software components are implemented as modules, that can be re-used in different context. In the following, we describe the basics of our software system.

Our general idea is to provide module-based image processing functions that are connected to a pipeline. Those pipelines can also be set up across network connections.

The software is written for Linux, but could be transferred to other operating systems, too. Transfer and processing of image data will be done within the GStreamer [14] framework. This open-source multimedia framework is used by many multimedia applications under Linux. Many functions needed for this application are already implemented in GStreamer, like format conversion, image resizing, encoding, decoding, timing issues and network data transmission. The GStreamer framework is plugin-based, so the functionality can be expanded by new elements, which can also define their own data types. There is also the possibility to set up branched pipelines where data of one image is processed by many elements in parallel.

4 Features of the system

In the following subsections we will introduce the main features of the system and their implementation.

GPU accelerated stitching of panorama image In this processing step, the system needs to generate a panorama image from the four raw images acquired by the cameras. The information about the geometry of the camera setup and the types of lenses is included in a configuration file generated by the open source panorama stitching application Hugin[1].

The performance of the CPU-based stitching was evaluated using the command line tool from Hugin with the defined configuration file. In the following table, the duration of the stitching procedure is shown.

System	Execution time
Intel Core i5 750	0.90 sec
Intel Xeon E31245	0.55 sec
Intel Atom N270	11.47 sec

With these tests it can be shown that a real-time capable solution using these algorithms is not possible. Even with more powerful CPUs it seems impossible to reach execution times in the magnitude of 0.05 seconds (for 20 fps). Therefore we had to investigate an alternative solution for real-time panorama stitching. We implemented a GPU-based stitching algorithm, that generates a lookup-table (LUT) at startup. The whole procedure works as follows:

[1] Hugin - Panorama photo stitcher; hugin.sourceforge.net

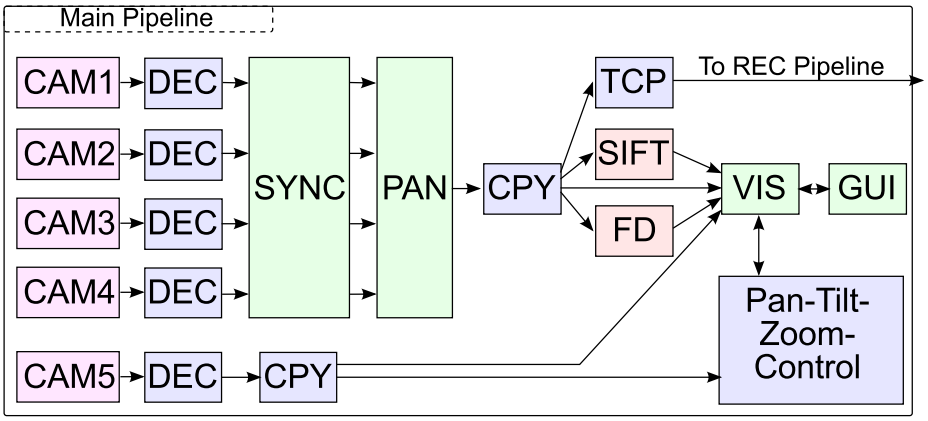

Fig. 4. Simplified pipeline setup of the intelligent surveillance system; the most important elements are their connections, as shown in this figure. There are many more elements involved than those shown in this figure. As decoding is done by several elements, some elements need different colourspace and there are measures to ensure that the PTZ image is only decoded if it is activated in the GUI. Elements that are used in this setup:
CAMx: access to the IP cameras
DEC: parsing of RTSP protocol, decoding of h264
SYNC: forward a group of 4 frames, drop otherwise
CPY: duplicate image to be used in multiple downstream elements
PAN: panorama generation, calculate one image from 4 images
TCP: send data via a TCP connection
VIS: overlay PTZ image, mark detected features
GUI: provide control widgets, control recording

- Offline Calibration (once)
  - Take 4 Snapshots
  - manually/semi-automatically configure panorama stitching in Hugin, save configuration file
  - generate LUT with sub-pixel accuracy from configuration file using a script
- Online Initialization (once at startup)
  - allocate GPU memory (4+1 image buffers)
  - upload LUT
- Online Processing (each frame)
  - 4x upload YUV buffers
  - 4x colourspace conversion to RGB
  - panorama stitching, subpixel accuracy
  - download panorama

Executing all processing steps for each frame takes 7 ms on an NVidia GTX580 (11 ms on an NVidia GTS450) including upload and download. The

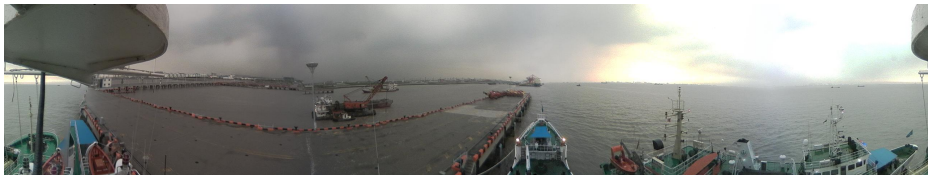

Fig. 5. This image shows the result of panorama stitching and has a resolution of 2556 x 470 pixel. The camera system is mounted on the pole of a ship. The image also shows a colour correction algorithm that has not been implemented in the GPU-accelerated code yet. In some areas, the blending is not perfect. This could be improved by a better calibration of the lens distortion.

algorithm is parallelized very efficiently, one output pixel can be computed by one of the shaders of the GPU (NVidia GTX580 has 512 shaders clocked at about 1.5 GHz). Theoretically, this GPU is capable of stitching together > 100 panoramas per second. Due to an OpenCL based implementation, the code could also be used for devices of other vendors. A generated PTZ image is shown in 5

Object Detection In order to detect known objects in the image, we implemented an advanced algorithm for object detection. This can either be used to generate user notifications if a known object is detected, or to supervise whether one object of the scene has been moved. In several research projects, Scale Invariant Feature Transform (SIFT) has proven to be a powerful, but also computationally intensive algorithm [15–17]. The general idea of the SIFT algorithm is to find significant points in an image and describe them in a way so that the description is invariant to rotation, translation and scaling. Extracted features of images can be compared to features from a sample image in order find corresponding points. If multiple corresponding points can be found, a transformation matrix can be calculated. We implemented a processing element, that compares the detected features of the current image to the features of all image files in a folder. Detected objects can be displayed in the GUI.

Person Detection Detecting persons is another important feature of an intelligent surveillance system. Unlike in the task of object detection, the targets are not known exactly, as the persons may be unknown. Therefore a person detection element has been set up using Haar classifiers [18] that are implemented in the OpenCV framework [19]. Prior tests have shown that running the face detection algorithms with a reduced resolution is sufficient for a robust detection. This way we can save processing time and thus computing capacity for real-time critical tasks. Within this element we can load different training data sets, e.g. for faces, faces in profile view or upper bodies.

Scheduling In order to achieve real-time stitching and display of the panoramic video, we need to take some measures in our image processing pipeline. Some

algorithms like object- and face detection may not run in real-time. Thus, we have to drop frames, if processing of a previous frame is still in progress. After the processing has finished, the newest available frame will be processed. This way, it is ensured, that the frame processing queue does not grow. The recording function is scheduled to allow frames queueing up to a certain point, in order to compensate for load variation of the system and to avoid frame drops.

Recording As we are using the GStreamer-framework, it is easy to implement recording functions. Recording is implemented as a separate pipeline, that runs detached from the main pipeline. This way, recording can be started and stopped independently. Several tests have been performed in order to find a good compromise between data rate and CPU load. It is possible to choose different codecs (e.g. h264, MJPEG, MPEG2, Theora, h263) and there is also the possibility to reduce resolution and framerate. Trials showed that h264 yielded acceptable results with the following settings:

- recording resolution 1880 x 384
- 10 frames per second
- tuned to be less CPU-intensive
- bitrate 3000 kBit per second

Using these settings, it is possible to save one hour of video in 1.35 GByte. Considering the size of modern HDDs, this is quite moderate.

Autofocus There are several options to control the functions of the PTZ camera. Beside the possibilities of controlling the focus with a slider in the GUI or choosing predefined values from a lookup table, an autofocus algorithm is implemented. Within this mode, the sharpness of the image is analysed while the focus setting is changed. We take the difference between two neighbouring pixels as a measurement for image sharpness. As a first step, the image is filtered with a Sobel operator and the absolute of the value will be determined. This way, points with high contrast will be detected in the image. In the next step we sum up all the pixel values of the filtered image and take the resulting integer as a measure. The higher this value, the sharper the image. During the procedure, the focus will move from the lowest setting to the highest setting. The received values are assigned to the focus settings. In the last step, the focus setting with the maximum value is chosen.

PTZ image One of the main features of the system is to provide close-up views of user defined regions-of-interest (ROI). These can be selected in the main software by dragging a rectangle in the main image or by double-clicking on a detected image feature. This way, the system will automatically try to calculate the setting for the PTU-unit and the zoom lens.

$$pan = (\frac{x_r + x_l}{2} - \frac{width}{2}) * \frac{hfov}{width}$$

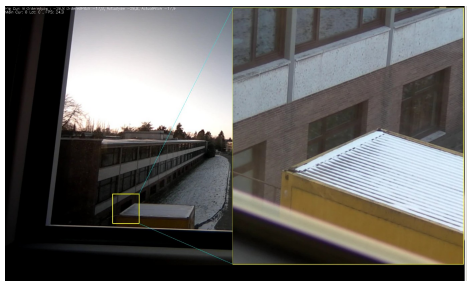

Fig. 6. In this image the PTZ-overlay function is shown. The ROI is choosen by the user. After the PTU, zoom and focus have been adjusted, the image is overlayed.

In this equation width is the width of the panorama image, hfov is the horizontal field of view, in our case 360°, x_r and x_l are the right and left border of the ROI.

Analogue to this we can define

$$tilt = (\frac{y_d + y_u}{2} - \frac{height}{2}) * \frac{hfov}{width}$$

given that the aspect ratio of one pixel is 1 : 1, height is the height of the panorama image, x_u and y_d are the upper and lower border of the ROI.

The desired horizontal field of view of the PTZ-camera is calculated the following way:

$$PTZfov = \begin{cases} \frac{x_r - x_l}{width} * hfov & \text{if } \frac{x_r - x_l}{y_d - y_u} \geq \frac{PTZwidth}{PTZheight} \\ \frac{y_d - y_u}{height} * hfov & \text{else} \end{cases}$$

where $PTZwidth$ and $PTZheight$ are width and height of the pan-tilt-zoom camera, the other variables are defined like in the prior equations. As the aspect ratio of the user-drawn rectangle will probably differ from the aspect ratio of the PTZ-camera, we need to distinguish between two cases. This way the desired horizontal field of view is calculated in a way that it always covers the full user-drawn rectangle.

Having the desired horizontal field of view, it is possible to calculate the corresponding focal length in the following way:

$$f = \frac{d}{2.0 * tan(\frac{\pi * fov}{360.0})}$$

where d is the horizontal width of the image sensor, and fov is the desired horizontal field of view. This value will be provided to the PTZ control routine. An image where the PTZ-camera automatically adjusted to a user-defined region is shown in 6.

5 Conclusion

In this paper we showed our research on an intelligent omnivision camera system. We showed how this system can be used in a surveillance scenario. One of the most important findings was, that there are a lot of parallels between surveillance and robotics and many implemented functions can be reused. Due to the fact that we are using the modular framework GStreamer, it was easy to access the IP cameras and it was also possible to add state of the art features like h264 recording. Using an OpenCL accelerated panorama stitching algorithm and our scheduling functions, it became possible to get a real-time view of the scene, while all other detection algorithms run as fast as possible.

There are several options to enhance the system. Due to the modular pipeline setup, it is also possible to add dedicated PCs that are only used for detection of image features. This way, computationally intensive image processing functions can be executed without the risk of disturbing real-time critical tasks.

The system can also be used for different types of camera systems. They could be equipped with wide-angle lenses or spherical mirrors. In the software setup, only the elements for generating the image need to be exchanged, the rest of the software as well as the GUI can be reused.

One possibility to have a next-generation surveillance systems would be to access even more cameras and to fuse the image of the different cameras with awareness of the three-dimensional structures in the scene.

Acknowledgment

The authors would like to thank our student assistant Johannes Liebrecht for designing the PTZ control board during his thesis, Andreas Maeder for the calibration of the panorama stitching routine, Gang Cheng for contributing the PTU control code and the Group Kunshan Robotechn Intelligent Technology Co., Ltd. for performing real-world testing in a maritime scenario.

References

1. Hannes Bistry and Jianwei Zhang. Task oriented control of smart camera systems in the context of mobile service robots. *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3844–3849, October 2009.
2. Hannes Bistry and Jianwei Zhang. A cloud computing approach to complex robot vision tasks using smart camera systems. *Intelligent Robots and Systems, 2010. IROS 2010. IEEE/RSJ International Conference on*, pages 3195–3200, October 2010.
3. Hannes Bistry, Frank Vietze, and Jianwei Zhang. Towards intelligent high resolution surveillance cameras. In M. Knzel and B. Michel, editors, *Safety and Security Systems in Europe*, number 10 in Micromaterials and Nanomaterials, pages 76–79. Micro Materials at Fraunhofer IZM Berlin and Fraunhofer ENAS Chemnitz, June 2009.

4. A. Maeder, H. Bistry, and J. Zhang. Intelligent Vision Systems for Robotic Applications. *International Journal of Information Acquisition (IJIA)*, 5(3):259 – 267, September 2008.
5. J. Foote and D. Kimber. Flycam: Practical panoramic video and automatic camera control. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 3, pages 1419–1422. IEEE, 2000.
6. K. Yamazawa, Y. Yagi, and M. Yachida. Omnidirectional imaging with hyperboloidal projection. In *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 2, pages 1029–1034. IEEE, 1993.
7. J. Wu, K. Yang, Q. Xiang, and N. Zhang. Design of object surveillance system based on enhanced fish-eye lens. *Chinese Optics Letters*, 7(2):142–145, 2009.
8. I. Haritaoglu, D. Harwood, and L.S. Davis. W¡ sup¿ 4¡/sup¿: real-time surveillance of people and their activities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):809–830, 2000.
9. Martin Weser, Daniel Westhoff, Markus Hüser, and Jianwei Zhang. Real-time fusion of multimodal tracking data and generalization of motion patterns for trajectory prediction. In *Proc. of the IEEE Int. Conf. on Information Acquisition (ICIA)*, Shandong, China, August 2006.
10. Y.H. Huang, M. Hu, H.L. Chong, X.M. Jia, J.X. Ma, and W.L. Liu. A survey of robot visual tracking algorithm. *Key Engineering Materials*, 501:577–582, 2012.
11. J. Huang, S.P. Ponce, S.I. Park, Y. Cao, and F. Quek. GPU-accelerated computation for robust motion tracking using the CUDA framework. In *Proceedings of the IET International Conference on Visual Information Engineering*, 2008.
12. F. Torres, F. Candelas, S. Puente, L. Jiménez, C. Fernández, and R. Agulló. Simulation and scheduling of real-time computer vision algorithms. *Computer Vision Systems*, pages 98–114, 1999.
13. R. Xu and J. Jin. Scheduling latency insensitive computer vision tasks. *Parallel and Distributed Processing and Applications*, pages 1089–1100, 2005.
14. W. Taymans, S. Baker, A. Wingo, R. Bultje, and S. Kost. Gstreamer application development manual (0.10.21.3), October 2008.
15. D.G. Lowe. Object recognition from local scale-invariant features. *International Conference on Computer Vision*, 2:1150–1157, 1999.
16. D. Schleicher, L.M. Bergasa, R. Barea, E. Lopez, M. Ocaña, and J. Nuevo. Real-Time wide-angle stereo visual SLAM on large environments using SIFT features correction. In *Proceedings of the IEEE/RSJ International Conference on International Robots and System, Oct*, pages 3878–3883, 2007.
17. J. Kuehnle, A. Verl, Zhixing Xue, S. Ruehl, J.M. Zoellner, R. Dillmann, T. Grundmann, R. Eidenberger, and R.D. Zoellner. 6d object localization and obstacle detection for collision-free manipulation with a mobile service robot. pages 1 –6, June 2009.
18. P. Viola and M. Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features. *IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION*, 1, 2001.
19. G. Bradski. The openCV library. *DOCTOR DOBBS JOURNAL*, 25(11):120–126, 2000.