Proceedings of the 2008 IEEE
International Conference on Robotics and Biomimetics
Bangkok, Thailand, February 21 - 26, 2009

Real-Time 3D Environment Perception: An Application for Small Humanoid Robots

Denis Klimentjew, Andre Stroh, Sascha Jockel, Jianwei Zhang
Dept. of Informatics, University of Hamburg, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany
{klimentjew, 1stroh, jockel, zhang}@informatik.uni-hamburg.de

Abstract—**This paper presents a modular software architecture based on a stereo camera system that makes real-time 3D perception, interaction and navigation for small humanoid robots possible. The hardware-independent architecture can be used for several purposes, like 3D visualisation, object recognition, selflocalization, collision avoidance *et cetera*. First of all, the intrinsic calibration parameters of the stereo camera system are identified. Then the lens distortions of the input images are revised, and both images are rectified according to the extrinsic parameters to facilitate the subsequent correspondence analysis. The imaged scenery, as well as a region of interest, can be reconstructed either by the block-matching, the Schirai- or the Birchfield algorithm, followed by triangulation back into 3D space. Finally, the above-mentioned aims can be tackled based on the resulting 3D model computed by our modular software architecture.**

Index Terms—**software architecture, real-time, perception of environment, stereo vision, 3D reconstruction, humanoid robot.**

I. INTRODUCTION

In the field of autonomous robots, particularly in the domain of humanoid robots, stereo vision systems are still the most important interface between the robot and the outside world. This is resonable for many humanoid robots, e.g. QRIO by Sony [1] or ASIMO by Honda [2]. Despite the huge number of different sensors, most robots make use of a stereo camera system, too. Comparable to the perception of humans, a stereo camera system offers a large amount of information, while the aquired image data can also be used in many different ways. However, there are a lot of disadvantages: the amount of information within an image is huge, and processing the data reasonable requires considerable calculation[1]. The advantage of visually based systems is the possibility to perceive the environment and, as a result, to react to this environment. The main disadvantage of a stereo camera system is the loss of depth information by projecting a 3D scene onto a 2D image plane in the image acqusition process [3]. If the depth can be reconstructed appropriately, stereo camera systems deliver enough data for the interaction of robots with their environment.

Most architectures for environmental perception are adapted to a special scenario or certain hardware [4][5]. Though dealing with a specific problem, we design a universal architecture which could be used with many different platforms and with any stereo camera system. In additional, it can be used as a module of a further architecture.

This paper is structured as follows: In section II, we present the modular software architecture, discuss its features as well as several modules and their specialised characteristics. In section III, we describe the implementation of our architecture designed for a small humonoid robot. In section IV, we review the extensibility of our architecture. We discuss our experimental results in section V, and demonstrate the efficency of the developed architecture. Finally, we present our conclusion and future work in section VI.

[1]The reason for this becomes clear when we consider the significance of visual perception in humans and primates. The significance derives from the amount and size of areas involved in image analysis. Approximately 60% of of the cerebral cortex is involved in the perception, interpretation and response to visual stimuli. The primary visual cortex alone constitutes 15% of the cerebal cortex http://www.allpsych.uni-giessen.de/karl/teach/aka.htm.

II. MODULAR SOFTWARE ARCHITECTURE

Within the robotics domain, navigation consists of three areas: localisation, path planning and robot control [6]. The navigation of humanoid robots has been an important and diverse topic in the field of robotics for several years and the reason why many scientists are getting involved in robotics.

The complete architecture is based solely on a stereo camera system. In a first step, the images are preprocessed. The pair of stereo images is corrected, since every camera lens causes a distortion. After this, the images are rectified. The parameters of the calibration of each single camera (intrinsic) as well as the relation of both cameras to the external world (extrinsic) of the stereo system are determined, too. We describe the calibration of each camera and the calibration of the stereo camera system in section III. Our software architecture is outlined in fig. 1.

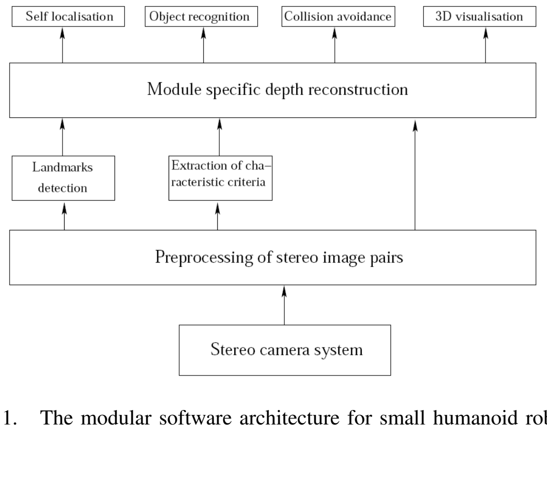


Fig. 1. The modular software architecture for small humanoid robots.

After preprocessing – which consists of the correction of lens distortion, image enhancement, and rectification – the input images, several possibilities for other processing steps are revealed, e.g. object recognition, collision avoidance *et cetera*. For the purpose of self-localisation, the software system can search for passive, active, artificial or natural landmarks [7].

The quality of a depth reconstruction depends on several characteristics, e.g. texture, non-homogenous regions *et cetera*. If many features are of interest in an image, a detailed reconstruction can be accomplished. After a depth reconstruction, a 2.5D depth map can be used for 3D visualisation based on triangulation. Collision avoidance, self-localisation and path planning for humonoid robots can all be derived from depth reconstruction.

The advantage of the present architecture lies in its modularity and universality. To extend our architecture to new stereo camera systems, it only has to be extended with the appropriate camera driver. All the other processing modules are preserved and must not be changed or adapted. Other advantages of the architecture are easier modifications as well as the exchange of single modules. Furthermore, each module encapsulates its algorithms and for the integration within the architecture, only the input and output of the module has to be adapted to the interface. Thus, the basic construction process is always preserved. This facilitates the debugging of a single component as well as that of the complete system. In the next section, we describe the implementation of the system and various integrated algorithms. The performance of the implemented system is investigated in section V.

III. BASIC STEREO PREPROCESSING FOR PROPOSED SOFTWARE ARCHITECTURE

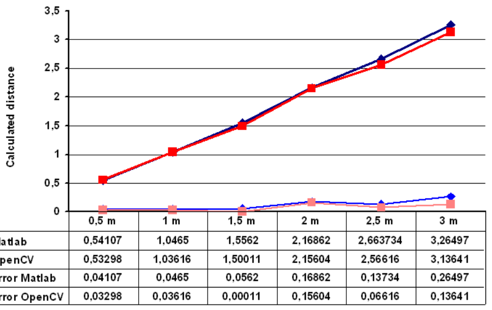
All experiments and tests have been carried out on a HOAP-2 humanoid robot from Fujitsu. The HOAP-2 robot is equipped with two Logitech quickcams. The implementation of the proposed architecture is realised via C/C++. However, we investigated further camera systems to test our software architecture and to prove its portability. The details are given in section V. In this section, we successively present and discuss the single steps of the implementation.

A. CAMERA CALIBRATION

Each of the two Logitech Quickcams of HOAP-2 have a $\frac{1}{4}$ inch CMOS sensor. They are not synchronised and have a resolution of 324×248 pixels [8]. For camera calibration, we considered algorithms by Tsai [9] and Zhang [10]. The camera calibration of Tsai is hard to realise, since at least seven corresponding point pairs are constrained to not lie in the same plane, otherwise calibration will fail. Thus, for our needs, calibration by Tsai turned out to be rather inefficient. Rather we follow the procedure of Zhengyou Zhang, whose methods are partly available in the OpenCV-library version 1.0 [11], a free computer vision library optimised for Intel processors co-developed by Intel. The library is based on Integrated Performence Primitive (IPP) [12] and is adapted for imperative Intel processors. For the calibration, we use a checkered calibration pattern with a single square size of 30×30 mm.

As a result of the calibration of a single camera, we got its intrinsic parameters and the lense destortion. This was done for both cameras seperately. The OpenCV-library does provide methods for calibrating a stereo camera system to get the extrinsic parameters, but the results are extremely unreliable and mostly poor. To improve the results and to determine reasonable rectification matrices based on inctrinsic and extrinsic camera parameters, we implemented a procedure based on the work of Fusiello, Trucco and Verri [13] in C/C++. With the matrices we got from this algorithm, the camara calibration could be completed [14]. The procedure is characterised by its stability, robustness and runtime performance. Runtime aspects are discussed in section V. This leads to a fully-automatic calibration procedure.

Moreover, we used the semi-automatic calibration tool box for Matlab [15] to verify our results from the camera calibration. The comparison of the depth estimations for both procedures along with the number of points for which no correspondance could be found is shown in fig. 2 & 3.



	0,5 m	1 m	1,5 m	2 m	2,5 m	3 m
Matlab	0,54107	1,0465	1,5562	2,16862	2,663734	3,26497
OpenCV	0,53298	1,03616	1,50011	2,15604	2,56616	3,13641
Error Matlab	0,04107	0,0465	0,0562	0,16862	0,13734	0,26497
Error OpenCV	0,03298	0,03616	0,00011	0,15604	0,06616	0,13641

Fig. 2. Comparison of the resulting calibration parameters from two procedures, camera calibration toolbox for Matlab and OpenCV with an algorithm by Fusiello, Trucco and Verri. The latter is based on depth estimates.

The depth test were accomplished by measuring distances from 0.5 m to 3 m between the camera lens and relevant objects, increasing the distance by 0.5 m each time. Matlab and OpenCV methods were applied to the same input images. Image pixels whose correspondance cannot be determined properly are set to an average depth value based on the sourrounding pixels.

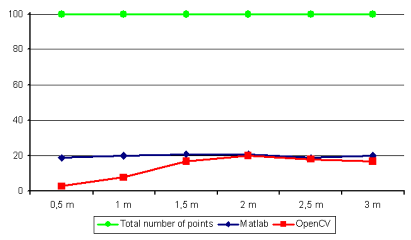


Fig. 3. The total number of corresponding points within an image pair as well as the number of points for which no correspondance has been found.

For calibration, a region of interest (ROI) of 10×10 pixels is selected in the input image. In fig. 3 the green line shows the total number of pixels that are processed to find corresponding points. The red and blue lines represent the number of points with no correspondence assigned by OpenCV and Matlab during this process. It is obvious that with our calibration method, the number of non-assigned corresponding points decreases with a smaller distance. However, it also increases with a growing distance and stabilises to a degree comparable to Matlab values between 18%-20% of non-corresponding assignments.

Thus, we have developed a fully automatic calibration procedure for stereo camera systems which delivers satisfying results and serves as a base for the architecture presented here.

B. RECTIFICATION

Most camera sensors and lenses are not perfect. Therefore, a distortion occurs in every picture and affects the hole camera model. These so-called non-linear effects should be corrected by the aid of inverse transformation. Therefore, the amount of distortions is determined during the calibration of a camera. The distortions only affect the intrinsic parameters of the camera. We distinguish between a radial and a tangential lens distortion. The radial distortion again is divided into cushion-shaped and metric tons-shaped distortion. The camera calibration according to Zhang only takes radial distortion into consideration, since the tangential distortion falsifies the picture only slightly.

After the calibration of the cameras, the lens distortion can be filtered out of the images with OpenCV methods. However, as already mentioned the OpenCV methods deliver inappropriate results for intrinsic and extrinsic camera parameters. Also, the OpenCV implementation of the rectification process, the rotation and translation and skew of the input images to artificial coplanar images, leads to worse results. Additionally, integrating the scale factors of intrinsic camera parameters improve the results only slightly. Thus, we decided on integrating the procedure by Fusiello, Trucco and Verri [13] in our software architecture, since its compact algorithm yields more reliable results within less processing time than OpenCV. Both projection matrices are revealed from the intrinsic and extrinsic parameters of the respective camera after its calibration as in Eq. (1) and together form the projection matrix P.

$$P = A \cdot [RT] \tag{1}$$

where A is a matrix with intrinsic camera parametres, R a 3×3 rotation matrix and T a 1×3 translation vector [16]. R and T are the extrinsic camera parameters and describe the relation of the camera position to the world reference system.

The optical centres of both cameras are preserved with this rectification procedure, merely the rotation of the cameras is changed. With this process, an ideal stereo system is constructed by re-projecting the image planes of both cameras onto new coplanar image planes. Afterwards, the new artificial projection matrices are calculated. This process is shown clearly in fig. III-B. The main advantages of rectification

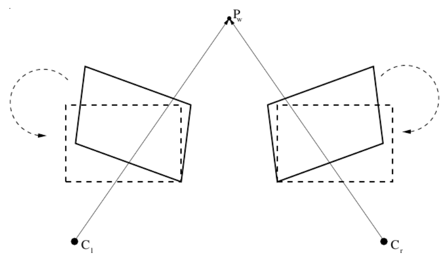


Fig. 4. Image warping within rectification brings the two artificially new images together on the same baseline.

become clear not only in connection with the Birchfield algorithm.

The set of suitable locations for corresponding points within the second image lie on the so-called epipolar line. Since fundamentals on epipolar geometry can be found in most standard books on computer vision, we refer the interested reader to [17], [18], [19]. The essence is that through rectification, the epipolar lines are reprojected onto horizontal lines in the new images, which makes it much easier for succeeding algorithms to detect corresponding pixels.

After rectifying the images, the next step is the depth reconstruction based on geometric triangulation from a 2.5D depth map. It is possible to choose either a small area or the complete image for reconstruction. To test the architecture, we generated edge pictures by a Laplacian of Gaussian with a δ = 0.7 and a window size of 3×3. Then we integrated the algorithm for landmark recognition by Scharstein and Briggs [20]. Landmarks within the images are generated on the basis of similarity functions of black and white brindled patterns with unique spreads between the lines that can be easily distinguished from the background. This allows for a better and unambiguous recognition of landmarks. In addition, the landmarks were provided with a bar code for more reliable detection. An accurate position in space can be assigned to the detected landmarks in combination with the 3D reconstruction system. If two or more landmarks are found, the position of the robot can be determined unambiguously. Additionally, another result of the Sharstein and Briggs algorithm is the length of a single stripe of the landmark pattern. With this length information, the distance to the point can be calculated. This distance is compared to the assigned depth from the disparity map. The integration of the algorithms is fast and causes no problems. The algorithms were integrated into the software architecture through an adaptation of the incoming and the outgoing data.

The size of the area that has to be reconstructed can be chosen by the user. To maintain the same conditions for a test of the architecture with three different camera systems, the complete images are reconstructed. In the following, some depth reconstruction procedures are introduced and compared respectively. Their efficiency and the required time is presented in the section V.

C. DEPTH RECONSTRUCTION

To realise depth estimation in the best way possible, some well-known algorithms are compared, e.g. Shirai, Block-

Matching and Birchfield. From the beginning of this work, it was clear that Birchfield algorithm delivers the best performance concerning runtime aspects. However, it was also important to compare the quality of the algorithms. To test the advantages and disadvantages of the algorithms, their behavior is examined in the context of a table scenario. The input images of the reconstruction system are shown in the experimental setup of fig. 5.

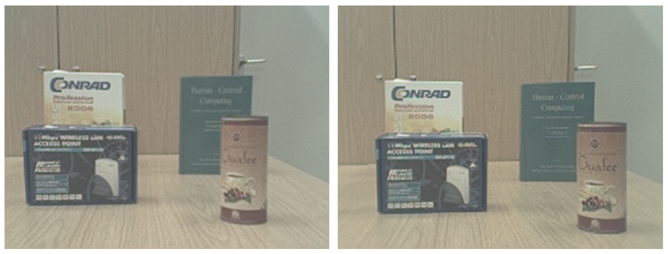


Fig. 5. Pair of stereo images of a table scene.

Objects have been equally distributed in different depths. To allow a fair comparison between different correspondance algorithms, these input images and their rectified image pair (see fig. 6), are used henceforth. For the acceptable time performance we delimited the maximum disparity to 160 pixels for all used algorithms.

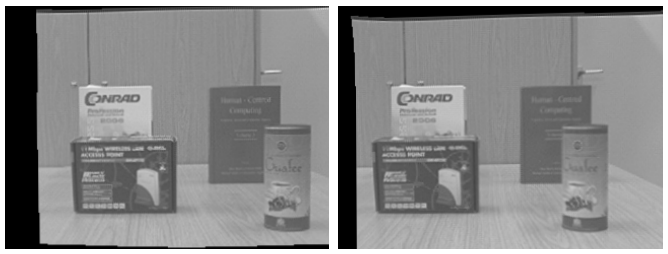


Fig. 6. Corrected and rectified pair of stereo images of a table scene. This pair of images serve as input to successive correspondance analysis procedures.

At first, the Shirai algorithm has been implemented and examined. A theoretical description of this algorithm can be found in [21]. The algorithm was primarily developed for detecting correspondences of edge images. Even if it leads to unequally good results for edges and texturised areas, it fails when used for homogenous regions. It shows worse behaviour in reconstructing the depth of continuous surfaces. Through its advantages for the correspondence of edge-enclosed and texturised areas, the objects on the table are well recognised while the continuous areas clutter, see fig. 7.

Secondly, the Block-Matching algorithm is examined. The computation of the disparity with Block-Matching allows to disassemble itself since it is composed from several processing steps. In a first step, the pair of stereo images is divided into blocks of equal size whose block disparity is calculated. We got the best results with a block size of 5 $\times$ 5 pixels. Then the block disparity on the pixel level is refined. The search for corresponding blocks is affected via different metrics. For our experiments, the Block-Matching algorithm has been

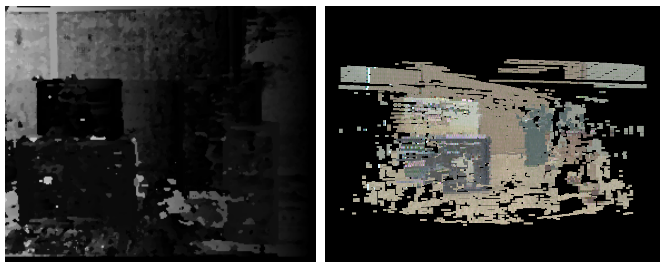


Fig. 7. Disparity map (left) based on the Schirai algorithm and the depth reconstruction (right) by using OpenGL. We used blocks with a maximum size of 20 pixels to calculate an intensity.

implemented with the "sum of the absolute differences" (SAD) metric. This algorithm shows a similar behavior to the Shirai algorithm through comparing intensity values of a certain window and the local searching character. The results are also very similar to the Shirai algorithm concerning object edges, homogenous regions and hidden areas. In contrast to the Schirai algorithm, the edges of objects are rather blurred due to the refinement of the block disparity at pixel level. In areas with steadily recurring patterns, the grain of the algorithm fails completely, e.g. the cupboard or table (see fig. 8). Finally, the algorithm of Birchfield and Tomasi is

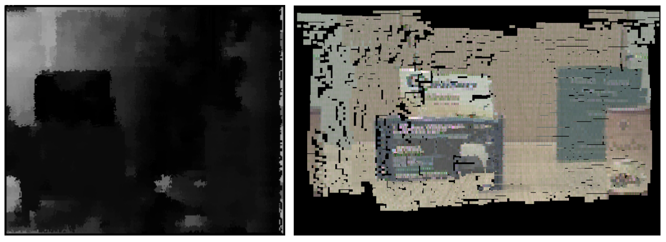


Fig. 8. Disparity map (left) based on Block-Matching. Brightness of an image point represents depth of a real-world point. The brighter, the closer to the camera. A depth reconstruction (right) from the depth map is implemented by using OpenGL.

considered. The algorithm is based on dynamic programming [22] and also part of the OpenCV library, so that this can be used immediately within our architecture. In contrast to the above-mentioned correspondence procedures, the algorithm of Birchfield and Tomasi has a global searching character through its dynamic programming principle instead of local windowing functions.

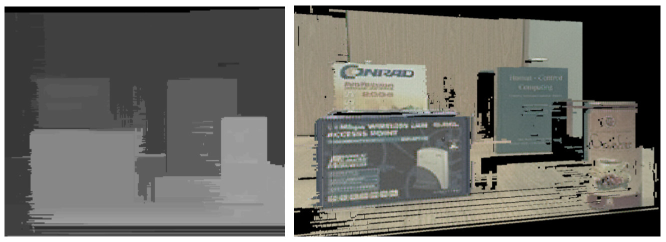


Fig. 9. Disparity map (left) based on the dynamic programming algorithm by Birchfield & Tomasi [22]. Brightness of an image point represents depth of real-world point. The brighter, the closer to the camera. A depth reconstruction (right) from the depth map is implemented by using OpenGL.

The algorithm scans a scanline from a certain point - defined by the coordinate in the left image - for local optima and refines it under consideration of the depth values above and below the optima. This is realised via a compact cost function. For a short but detailed description of the algorithm of Birchfield and Tomasi, see [23]. Note: Only few outliers are found at the object edges as well as the continuous depth difference between the cupboard and the door on the right image side (see fig. 9). The stereo algorithm by Birchfield and Tomasi is not as time-consuming and computationally expensive as classical methods that normally compare the similarity of frames of size $N \times N$ around pixels to determine their correspondence.

Among the tested correspondence analysis procedures, the algorithm of Birchfield and Tomasi deliveres superior results. The method is preferable to both of the other procedures and hence was included in our software architecture. Nevertheless, it is possible that under different test conditions, e.g. real natural scenes, the Schirai algorithm will lead to better results than Block-Matching or Birchfield. This could be caused by the fact that most natural scenes only contain few continuous surfaces or repeating patterns.

After considering the disparity maps of the different algorithms, they were visualised three-dimensionally with OpenGL. The results of the reconstructed scene in 3D are shown in the right images of fig. 7 - 9 respectively.

IV. FURTHER MODULE FOR A STEREO-BASED ROBOT CONTROL ARCHITECTURE

Self-localisation of a robot can be realised by using artifical passive landmarks. Several different guiding points exist that facilitate visual recognition. The visual admission of guiding points is put out constantly to varying lighting conditions, visual distortion and different scaling. In this work, we have searched for a solution nonsensitive to all above-mentioned factors. The authors Scharnstein and Brigs provide a mathematical model for artifical passive landmarks [20] that robustly deals with different lighting conditions, visual distortion and different scaling. The landmarks are based on a self-similar mathematics function. Another advantage of these landmarks is their constant complexity and simple pattern, so that the landmarks with complexity of $O(\frac{nw}{k})$ are easily detected in an image. The calculation of the complexity takes the number of pixels n of an image, the searching interval w and the number of the searched fissures k into account. For unambiguous identification, they are complemented with a barcode stripe [24]. In addition, an unequivocal position can be stored for every guiding point in the space. If two or more landmarks are in the image, the robot can obviously be localised unambiguously. The algorithm is open-source for research purposes and exists in C/C++, thus it was integrated into our architecture. The algorithm quickl yand robustly determines the landmarks in an image. The computed information allows for a relatively precise self-localization of the mobile robot. Since normally a critical distance is maintained around the robot for security reasons and collision avoidance, the navigation takes care of guiding the robot through a known and unknown evironment with obstacles beyound the critical distance, in the case of our HOAP-2 a distance of 0.2 m. In addition, the precision can be improved, e.g. by using a Kalman or Particle filter.

V. EXPERIMENTAL RESULTS

As section III shows, our software architecture is very modular as further specialised modules are easy to integrate. We have combined existing algorithms with our own. In this section, the architecture is tested with different stereo camera systems. We selected three systems that differ in the type of connection and image resolution: two USB Logitech Quickcams with a resolution of 324×248 pixels, two Sony DFW-VL 500 connected via firewire with a resolution of 640×480 pixels, and finally two Sony XC-999P with an image resolution of 704×576 pixels. The latter are connected through a Sensoray 2255S framegrabber. By integrating these three connection types into our architecture, the system supports most common cameras. By analysing the system with different cameras and connection types, the influence of the total number of pixels on the processing time emerged[2].

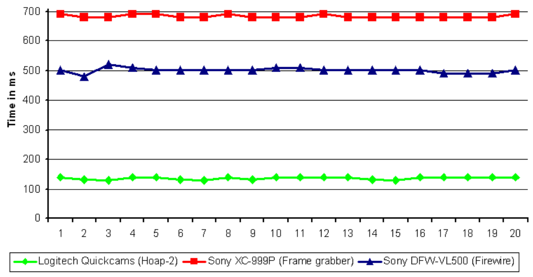
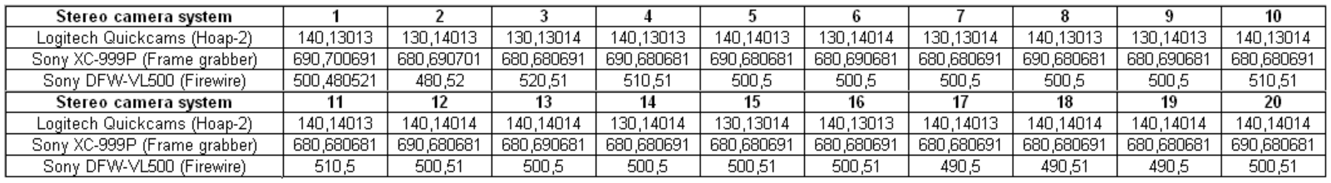


Fig. 11. Measured CPU-time for three different stereo camera systems for the following processing steps: lens destortion, rectification, image enhancement (Gaussian), disparity map including depth calculation.

The total runtime of image acquisition, lense distortion correction, rectification, image enhancement (Gaussian), including depth map valuation has been investigated. The used CPU-time was measured for this purpose and leads to the exclusion of concurrent processes not involved in processing. The results are shown in fig. 11. The CPU-time increases proportionally to the number of pixels, independent of the image structure. The results do not include outliers. The maximal deviation from the average value is as follows: Logitech Quickcams 4,8%, Sony XC-999P with framegrabber 1,02%, and Sony DFW-VL 500 3,99%. Since we are able to predict the time interval needed for processing based on the above-mentioned tests, our implemented software architecture is a real-time system. If the integrated algorithms and moduls meet the requirements of real-time, the complete system will be a real-time system.

If we examined the currently fastest humanoid robot ASIMO with a maximal velocity of $0,75\frac{m}{s}$ [2], the robot

[2]All tests run on a DELL Optiplex 745 with 1 GB ram, Intel Core 2 CPU @ 2.13 GHz, 2MB Cache, and ATI Radeon X1300 equipped with 256 MB internal memory.



Stereo camera system	1	2	3	4	5	6	7	8	9	10
Logitech Quickcams (Hoap-2)	140,13013	130,14013	130,13014	140,13013	140,14013	130,14014	130,13014	140,13013	130,14013	140,13014
Sony XC-999P (Frame grabber)	690,700691	680,690701	680,680691	690,680681	690,680681	680,690681	680,680691	690,680681	680,690681	680,680691
Sony DFW-VL500 (Firewire)	500,480521	480,52	520,51	510,51	500,5	500,5	500,5	500,5	500,5	510,51
Stereo camera system	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Logitech Quickcams (Hoap-2)	140,14013	140,14014	140,14014	130,14014	130,13014	140,13013	140,14013	140,14014	140,14014	140,14014
Sony XC-999P (Frame grabber)	680,680681	690,680681	680,690681	680,680691	680,680691	680,680691	680,680691	680,680691	680,680681	690,680681
Sony DFW-VL500 (Firewire)	510,5	500,51	500,5	500,5	500,51	500,51	490,5	490,51	490,5	500,51

Fig. 10. Measured CPU-time for three different stereo camera systems for the following processing steps: lens destortion, rectification, image enhancement (Gaussian), disparity map incluing depth calculation.

would be able to walk a distance of approximately 0.52 m during a depth estimation of our proposed system with maximum resolution. If a smaller resolution or a region of interest (ROI) suffices, the processing speed will increase and the covered walking distance will decrease, e.g. the robot will walk no more than 0.1 m at a resolution of 324×248 pixels. Thus, robots can be operated by stereo camera based environment perception and even respond in unknown environments.

However, image-based systems still have limitations regarding reliable environment perception and robot control, e.g. lighting conditions are fraught with problems. Furthermore, the cameras have to be synchronised to represent the environment at a particular time, e.g. during walking. Stiff and jerking movements of humanoid robots will cause the robot itself to swing. Even if the staggering problem could easily be solved by a triggered synchronisation , the preceding problems still need sophisticated solutions.

VI. Conclusion and Future Work

The presented software architecture can be operated at real-time, is universal and offers a lot of possibilities for 3D perception of and interaction with the environmant, not only for small humanoid robots. The architecture facilitates integration of different stereo camera systems or algorithms for different tasks or scenarios. Each researcher's own algorithms can be integrated and tested together with existing ones without any trouble. With this architecture, the algorithms can be integrated, tested and, if required, improved. The developer does not need to know the complete system specification and thus, can concentrate on his own tasks. It is also possible for several developers to work on different parts of the system. The algorithms to be integrated into the system have only to fit the architecture, not the system or other algorithms, e.g. camera. Beyond our fast and modular stereo system architecture we developed a command interface and an initialisation and calibration system for a HOAP-2 humanoid robot. Future work will be the navigation of the robot within known and unknown environments based on the proposed stereo camera architecture. Furthermore, the proposed architecture should be extended with a human-machine interface (HMI) for interaction, as much as memory and learning capabilities. Moreover, corporate robotics with two or more HOAP-2 robots will be investigated in the near future.

References

[1] Sony. Qrio. *http://www.sony.net/Products/cx-news/vol32/sideview.html, last visited 11/07*, 2008.

[2] Honda. Asimo. *http://world.honda.com/ASIMO/, last visited 11/07*, 2008.

[3] O. Faugeras. *Three-dimensional computer vision: a geometric visited-point.* 1993.

[4] J.-S. Gutmann, M. Fukutchi and M. Fujita. *3D Perception and Environment Map Generation for Humanoid Robot Navigation. In: The International Journal of Robotics Research 2008; 27; 1117*, 2008.

[5] C. Tessier, C. Cariou, C. Debain, F. Chausse, R. Chapuis and C. Rousset. *A Real-Time, Multi-Sensor Architecture for fusion of delayed observations: Application to Vehicle Localisation. In: First National Workshop on Control Architectures of Robots - Montpellier*, 2006.

[6] J. J. Leonard and H. F. Durrant-Whyte. *Mobile robot localization by tracking geometric beacons*. 1991.

[7] J. Borenstein, H. Everett, L. Feng, and D. Wehe. Navigating Mobile Robots: Systeme and Techniques. *In: Journal of Robotic Systems, Special Issue on Mobile Robots, pp. 231-249 14*, 1996.

[8] Fujitsu Automation Co. Ltd. Hoap-2 instruction manual. *http://www.jp.fujitsu.com/group/automation/downloads/en/service/humanoid-robot/ hoap2/instructions.pdf, last visited 14/07*, 2003.

[9] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-self tv cameras and lens. *In: IEEE Journal of Robotics and Automation 21*, 1987.

[10] Z. Zhang. A flexible new technique for camera calibration. *In: IEEE Trans. on Pattern Analysis and Machine Intelegence*, 2000.

[11] Intel. OpenCV library. *http://www.intel.com/technology/computing/-opencv/index.htm, last visited 14/07*, 2008.

[12] Intel Software Network, Intel Integrated Performance Primitives [IIPP]. *http://www.intel.com/software/products/ipp/ippvm20/index.htm*, 2007.

[13] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *In: Machine Vision and Applications, Springer*, 2000.

[14] A. Gruen and T. S. Huang. Calibration and orientation of cameras in computer vision. *Springer, Berlin, Heidelberg*, 2001.

[15] J.-Y. Bouguet, K. Strobl, W. Sepp, C. Paredes, and K. Arbter. Camera calibration toolbox for matlab. *http://www.vision.caltech.edu/bouguetj/*, 2008.

[16] O. Faugeras. Stratification of 3-dimensional vision: Projective, affine and metric representations. *In: Journal of the Optical Society of America 19*, 1995.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2^{nd} ed., 2004.

[18] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, Berlin, Heidelberg, 2004.

[19] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, Cambridge, MA, USA, 1993.

[20] D. Scharstein and A. Briggs. Fast recognition of self-similar landmarks. *In: In Workshop on Perception for Mobile Agents, pp. 74-81*, 1999.

[21] R. Klette, A. Koschan, and K. Schluens. *Computer Vision.* 1996.

[22] S. Birchfield and C. Thomasi. *Depth Discontinuities by Pixel-to-Pixel Stereo.* 1998.

[23] S. Jockel, T. Baier-Löwenstein, and J. Zhang. Three-dimensional monocular scene reconstruction for service-robots: An application. *Proc. of 2^{nd} Int. Conf. on Computer Vision Theory and Applications (VISAPP), Vol. Special Sessions*, pp. 41–46, 2007, INSTICC Press.

[24] D. Scharstein and A. Briggs. Real-time recognition of self-similar landmarks. *In: Image Vision Comput., 2001, pp. 763-772*, 2001.