

# Fast Development of a Robust Calibration Method for Service Robot FRIEND using *ImageNets*

U. LANGE, H. KAMPE, A. GRÄSER

Institute of Automation, University of Bremen, Germany  
lange@, kampe@, ag@iat.uni-bremen.de

Keywords: *calibration, configuration instead of programming, image processing, robot control, feedback control*

## Abstract

*Developing intelligent behavior for assistive robots like FRIEND is a complex and time-consuming task. The robot vision system has to robustly perceive the environment for collision-free robot motion and successful object grasping. Accurate calibration between camera sensors and the robot builds the basis for this. A robust calibration method has been developed with a framework called ImageNets using several layers to improve the accuracy. The accuracy itself has been evaluated on the one hand with generated marker images and on the other hand with 100 complete calibration cycles showing low standard deviations. With ImageNets the developers of assistive robots need only a short time to create robust high performance algorithms like the proposed calibration method. This article discusses the ImageNets framework and one of its applications in detail, namely camera calibration.*

## 1. Introduction

Developing intelligent behavior for assistive robots like FRIEND<sup>1</sup>, which operate under varying illumination conditions, is a complex and time-consuming task. The robot vision system has to robustly perceive the environment for collision-free robot motion and successful object grasping. Accurate calibration between camera sensors and the robot builds the basis for this.

Since it is time-consuming to develop intelligence for service robots, a tool which supports the developer to achieve his task quickly is a fundamental advantage. Moreover, reduction of time delay in robotic applications is another advantage. Due to that, two fundamental requirements are stated to be achieved by the ImageNets framework:

1. Requirement: Fast development
2. Requirement: High execution performance

The following section describes comparable frameworks and discusses their fulfillment of the two requirements. In section 3 ImageNets is described in detail and in section 4 a calibration application using ImageNets as its basis is presented. Section 5 illustrates a performance evaluation for the proposed method. Finally, section 6 demonstrates the outcome and future work.

---

<sup>1</sup> FRIEND – Functional Robot arm with user-fRIENdly interface for Disabled people

## 2. State of the Art

Representing mathematical problems or code as graphical elements exploits the human capability of understanding complexity better in a graphical form. Thus, the international standard IEC 61131-3 includes a graphical programming language for programmable logic controllers. These function block diagrams enable the programmer to directly use and parameterize pre-compiled functional blocks, which fulfills the first requirement. Based on this, a number of graphical programming tools are available today, each tailored for a particular industry. Moreover, the safety industry considers graphical programming as a safe way of programming (Pilz, 2012). The industry standard MATLAB (The Mathworks, 2012) has with SIMULINK a function block diagram interface. Its extension “Image Processing Blockset” provides the possibility for rapid prototyping of image processing algorithms. Due to the fact that SIMULINK is based on the interpreted MATLAB language, the execution performance is weak with respect to a compiled language like C++. The more commonly used method of using MATLAB is command line programming. Since MATLAB code does not have to be compiled, it is also suitable for fast developments. However, interpreted languages can have magnitudes of lower performance compared to compiled programming languages like C++. Therefore, the combination of rapid prototyping with MATLAB with afterwards conversion to C++ for real time execution is desirable. This can be achieved with the open image processing library OpenCV (Bradsky & Kaehler, 2008). Directly developing algorithms with OpenCV on the other hand is error prone and slow which would break the first requirement. Furthermore, MATLAB and OpenCV both do not have simple access to various hardware components which need to be accessed in service robotic applications. The medical image processing toolbox MeVisLab (Heckel, F. et al., 2009) combines the advantages of graphical programming with high execution speed by using the compiled language C++. Due to its major focus on medical image processing, this toolbox does not fit the given application. Thus, a new framework called ImageNets has been developed which uses the advantages of MeVisLab to fulfill both requirements while fitting the service robot domain.

**Table 2.1** Comparison of image processing frameworks

Category	Property	MATLAB	SIMU-LINK	OpenCV	MATLAB & OpenCV	ImageNets
General	Available functionality	+	○	+	+	○
	Hardware access	–	–	+	+	+
	Degree of brand awareness	+	+	+	+	–
	Cost	–	–	+	+	+
Image Processing Development Steps	Image acquisition	+	+	+	+	+
	Preprocessing (filtering, contrast enhancement)	+	+	+	+	+
	Segmentation	+	+	○	○	+
	Feedback segmentation	–	+	–	–	+
	Feature extraction (contour extraction)	+	+	–	–	+
	3D reconstruction (stereo correspondence computation, point cloud operations)	+	+	–	–	+
	3D visualization (point cloud)	+	+	–	–	+
	3D visualization (object models, robot environment and point cloud combined)	–	–	–	–	+

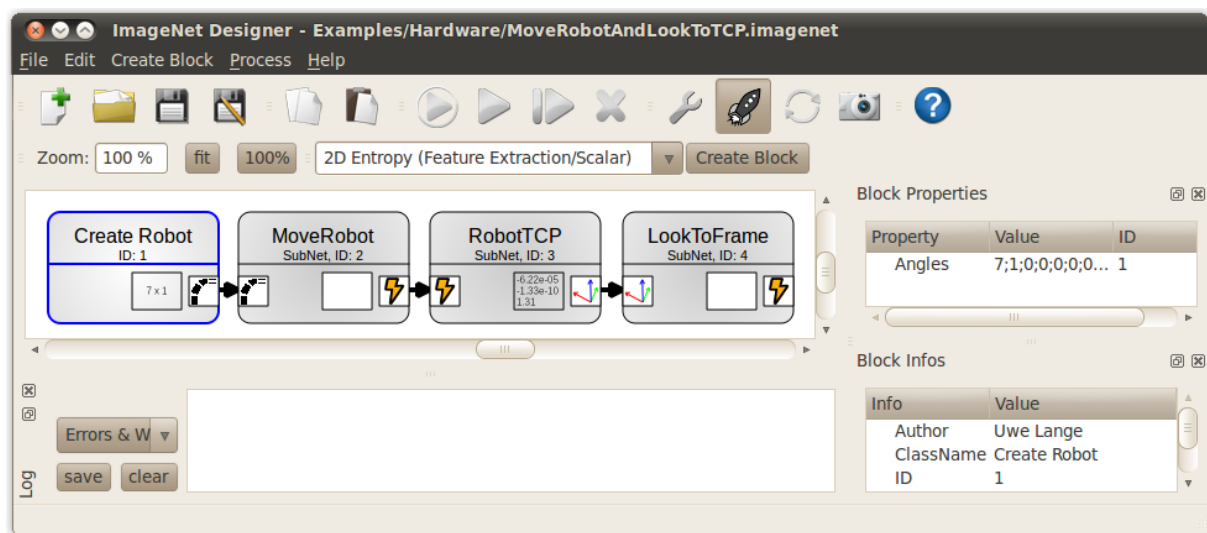
The various features of existing image processing toolboxes are documented in Table 2.1. It is obvious that the new ImageNets framework lacks brand awareness and functionality but it has been optimized for the development steps of image processing for service robotic applications. Details about this framework can be found in the following section.

### 3. ImageNets

Graphical function block networks (FBNs) enable the user to model algorithms with predefined blocks, which can be used as components-of-the-shelf. ImageNets is a tool to model FBNs with a main focus on robotic vision. As one tool in the chain of the process model for the development of the FRIEND robot, called FRIEND::Process, ImageNets has been mentioned in (Prenzel et al., 2012) and is open source<sup>2</sup> to be a benefit for the robotics community. The following subsections describe the core structure of this framework.

#### 3.1 Execution Logic

A function block network consists of blocks, which are interconnected with directed links. These links define the event flow in the net and pass data between the blocks. Fig. 3.1 shows a small example network, in which the execution order of ImageNets shall be explained. In the beginning of this ImageNet's execution, only the block *Create Robot*, which has no input ports can be executed, as its precondition is fulfilled. Afterwards, the block right of it can be executed, because the block to its incoming link has been executed already. This block actually accesses hardware and moves the robot joints of a robot system like FRIEND. Afterwards, FRIEND's stereo camera mounted on a pan-tilt-module (PTH) is moved so that the manipulator's tool center point (TCP) is centered in the stereo cameras' views.



**Fig. 3.1** ImageNet Designer displaying an ImageNet, which moves a robot and afterwards moves the PTH so that the stereo camera looks to the TCP.

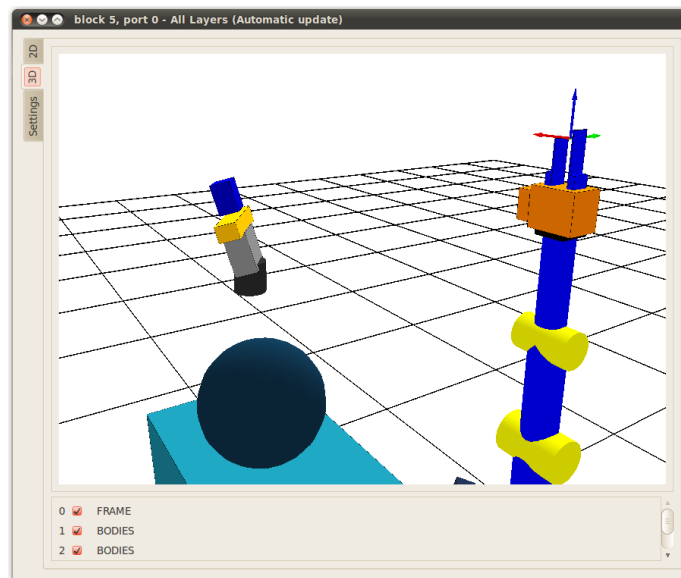
The function blocks are loaded via a plug-in system to support extensibility of available functionality without changing the base framework. Each plug-in holds a separate set of blocks, which may depend on further requirements like external libraries or certain hardware.

<sup>2</sup> [www.imagenets.sourceforge.net](http://www.imagenets.sourceforge.net)

### 3.2 ImageNet Designer

ImageNets networks are modeled using the ImageNet Designer, which is displayed in Fig. 3.1. Blocks can be selected and inserted in different ways to easily find desired functionality. They are displayed to the user sorted by image processing category (e.g. preprocessing or segmentation), alphabetically, by input and output data type. Links are established by dragging a link from an output of one block into an input of another block. To support the user in correct linking, the ports have a colored icon depending on their data type. Blocks are configured for specific tasks by setting their *properties*, like “Angles” in the block *Create Robot*. In addition to run the complete net, it is also possible to execute only single blocks. When just the parameters of a single block have been changed, only this block and its following blocks need a fresh execution to have updated data in the whole net.

The output ports of blocks contain data which can be displayed directly in the designer for an easy inspection. Depending on the data type, the data can be shown in 2D, 3D or both. By joining data with a *Joiner* block, it can be visualized in a common 3D environment, to manually verify the correctness of the algorithm. Fig. 3.2 visualizes the result of the pan-tilt-head movement relative to the manipulator displayed in the 3D environment. Whenever the data changes, the visualization is updated.



**Fig. 3.2** Data visualization dialog of the Designer with the manipulator (right), the PTH (top left) and the system (bottom left). The coordinate system in the top right corner shows the TCP.

### 3.3 Hardware connection

The goal of ImageNets is to facilitate modeling of robot vision algorithms. For these algorithms it is a necessity to access the hardware of the robotic system, for example to acquire new image data from the camera or to move the camera to look at a defined position. Fig. 3.1 already showed the ease of using the manipulator and pan-tilt-head (PTH) together. Normally interaction with hardware components requires deeper knowledge about the system from the user. With this tool, everything is hidden in the function blocks and only requires valid linking and well defined parameters. Section 4 will explain in more detail how this direct access to the hardware can be used to calibrate the stereo-camera relative to the manipulator in the FRIEND system, leading to improved grasping accuracy.

All in all, on the time writing this text, it is possible to access the stereo-camera, the hand camera on the manipulator, the pan-tilt-head, the manipulator, the force-torque-sensor on the manipulator and also the world model of the system in which run-time data is stored.

### 3.4 Feedback Structures for Enhanced Robustness of Algorithms

A major problem in image processing is to create algorithms that are robust against noise in the input, like different illumination or slight changes to the camera position (Grigorescu et al., 2010). Even small changes can require relevant adjustments to the parameters to achieve a decent result. Fixed parameters therefore may often lead to a very limited range of conditions, the algorithm works properly with. Ideally, the algorithm can adapt the parameters by itself to accommodate to changes. In practice, this would often require to test a range of plausible parameters and to keep the best. An additional problem is, that the original programmer of a function block does not know, if this block will later be used with fixed or varying parameters. For this reason, a flexible feedback system to change the parameters of any block, based on definable rules, can be applied to any block in the ImageNet Designer without changing the code of a block.

Fig. 3.3 illustrates the concept of feedback with a small example. The results of the different steps are displayed above the corresponding blocks. In the beginning, an image is loaded and converted to gray scale. Afterwards follows the segmentation which is the input to the marker detection. Outputs of the marker detection are 8 defined points of the found marker and a confidence score for the found points. This confidence score must be maximized for an optimal result. By enabling feedback on the segmentation block, it gets an additional feedback input port which is the lower one on the block in Fig. 3.3. The confidence score is then used as an estimator in the feedback port of the segmentation block. By trying out a range of values for the segmentation, the optimal value for the threshold will be found. In the lower right part of Fig. 3.3 a block collects the confidence scores which will be subsequently drawn as a plot. This renders it easy to see the influence of the parameter variation by the feedback and to choose a range for the parameter to test. In the figure it can be seen that the best threshold is at the value 72. A useful range to test is from 40 to 80. A tradeoff between execution speed and marker detection quality has to be chosen by limiting the test range.

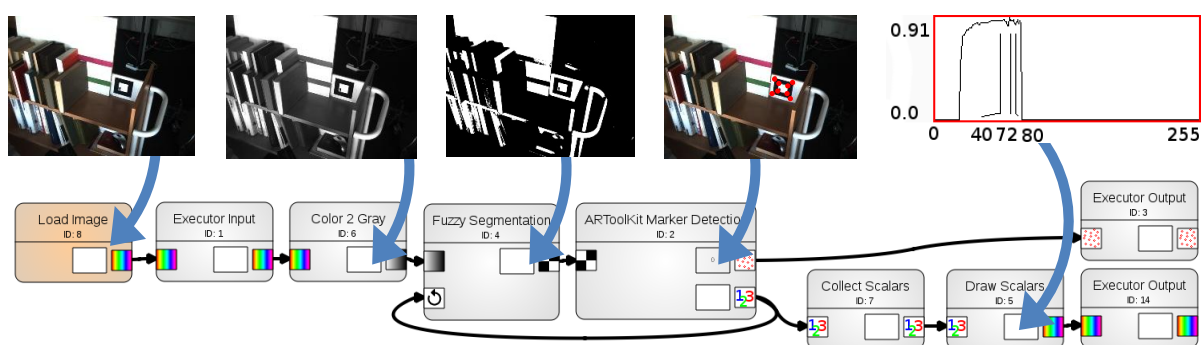
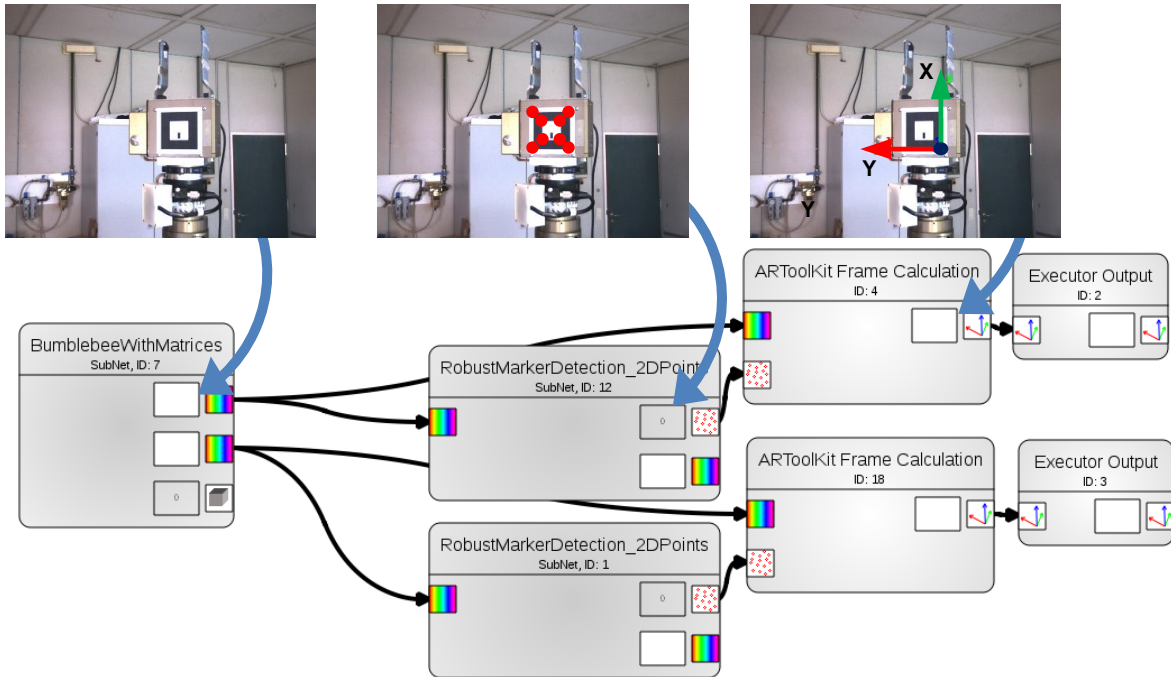


Fig. 3.3 ImageNet of robust 2D marker detection.

### 3.5 Hierarchical Modeling

When developing an algorithm, a standard engineering approach is to subdivide it into smaller, reusable parts. This kind of hierarchical modeling is also possible with ImageNets. By loading an ImageNet into the special block *ImageNet Executor*, a net can be executed inside another net. Connection between the inner and the outer net takes place with the help of

special input and output blocks inside the inner net. The net of Fig. 3.3 has as second block from the left an *Executor Input* block and on the right two *Executor Output* blocks. These blocks are mapped to ports in the outer ImageNet. Fig. 3.4 depicts the net from Fig. 3.3 used as a sub-net. On the left, images of a stereo-camera are acquired from the stereo-camera. These images are afterwards processed in parallel to find a marker in them and to calculate its exact location. Parallel executable blocks are really executed in parallel by using different threads for each block. This feature is provided by the framework to increase processing speed of the net to support the second requirement. Right in Fig. 3.4 these results are forwarded to output blocks so that this net is also possible to be used as a sub-net, which is described in section 4.



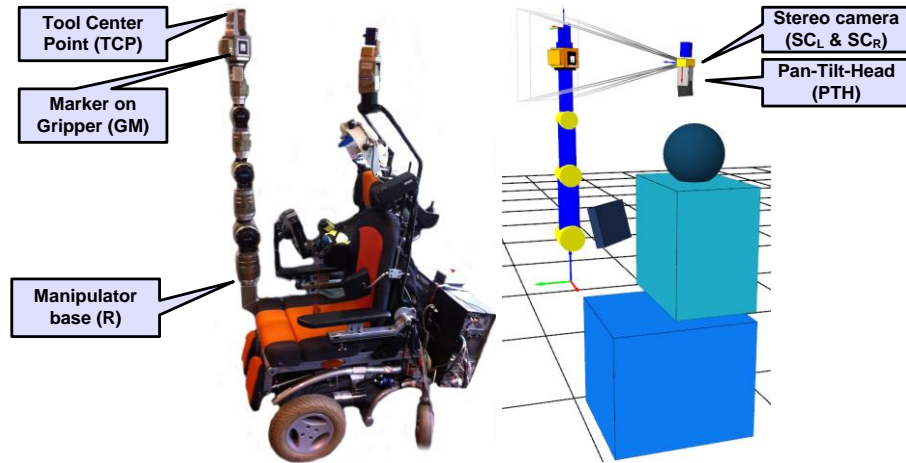
**Fig. 3.4** ImageNet of parallel robust 2D marker detection in both stereo camera images and calculation of the markers' 3D transformation matrices.

With the class of the *ImageNet Executor* block it is also possible to directly embed ImageNets into C++ code. Therefore, no graphical user interface has to be displayed to execute an ImageNet, which supports the second requirement of providing high execution performance.

#### 4. Camera calibration with ImageNets

In any robotic system like the FRIEND system, the relation between its robot and its camera sensors is very important. Obstacles and objects to be grasped are perceived by the camera and handed over to the robot motion planning. For collision-free robot movement a precise model of the robot environment is necessary (e.g. Fig. 4.1), which is based on an accurate calibration.

The previously presented ImageNets framework has been used to create a camera calibration method different to Heyer's method (Heyer et al., 2010) to show the advantages of the new framework like 3D visualization of the results. Both methods could later be combined to create an even more accurate and robust calibration method.

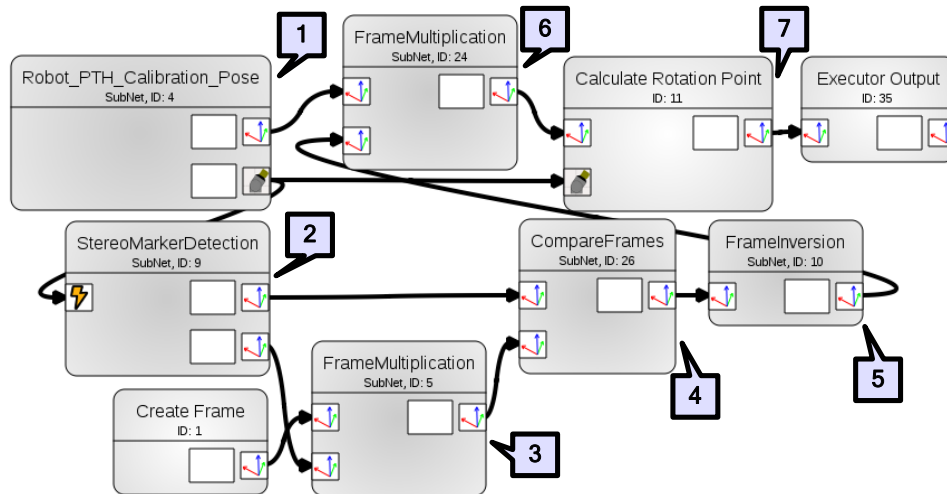


**Fig. 4.1** Overview of the FRIEND components, which are relevant for calibration based on gripper marker. Left: real system, right: 3D-model of the system.

For high robustness with respect to illumination changes, three layers of robustness enhancement have been developed using ImageNets, which will be presented in the following.

#### 4.1 Robust stereo camera 3D marker detection

Based on the previously described robust 2D marker detection in both stereo camera images (sections 3.4 and 3.5), this section discusses the robust marker detection in 3D. Basically, the idea is to detect the marker at the gripper, which is known in robot coordinates for any robot configuration. From that detection, the extrinsic matrix of the stereo camera with respect to the robot can be calculated. Afterwards, the PTH fixed point is calculated to be able to calculate the extrinsic matrices of the stereo camera for any pan and tilt angle.



**Fig. 4.2** ImageNet of the robust stereo camera calibration, based on gripper marker detection.

The ImageNet depicted in Fig. 4.2 gives an overview about the developed calibration algorithm. While every ImageNet can be executed in the ImageNet Designer, the intermediate execution steps can be visualized to quickly manually verify the correctness of the frame operations.



In the initial block *Robot\_PTH\_Calibration\_Pose*, the robot and the PTH are moved to their calibration configurations to make sure the gripper marker is visible in both images of the stereo camera. In Fig. 4.2 (1) the gripper marker in robot coordinates is calculated by multiplying the tool center point (TCP) in robot coordinates  ${}^{TCP}_R T$  with the gripper marker offset to the TCP  ${}^{GM}_{TCP} T$ , using the robot's DH<sup>3</sup>-parameters. The multiplication is denoted as  ${}^{GM}_R T = {}^{GM}_{TCP} T \cdot {}^{TCP}_R T$ . Afterwards, the previously described ImageNet from Fig. 3.4 is used to detect the gripper marker from both stereo camera images in the respective camera coordinates  ${}^{GM}_{SC_L} T$  and  ${}^{GM}_{SC_R} T$ . In (3) the detected marker in the right camera coordinates is then transformed to the left camera frame by the following equation:  ${}^{GM}_{SC_L} T = {}^{GM}_{SC_R} T \cdot {}^{SC_R}_{SC_L} T$ . Subsequently, in (4) the mean of both marker detections is calculated if their similarity is high. In case one of the two marker detections is invalid, only the other one is considered for output. The next step is to invert the gripper marker in left camera coordinates (5) by calculating  ${}^{SC_L}_G T = ({}^{GM}_{SC_L} T)^{-1}$ . By multiplying (1) and (5), the left camera extrinsic matrix in robot coordinates (6) can be calculated like this:  ${}^{SC_L}_R T = {}^{SC_L}_G T \cdot {}^{GM}_R T$ . Since the stereo camera of the FRIEND system is located on a pan-tilt-head, it is not enough to calculate the current transformation matrix of the camera. A pan and tilt rotation invariant frame inside the PTH has to be determined to be able to calculate any new extrinsic matrix of the stereo camera based on the current pan and tilt angles. This is done by  ${}^{PTH_FPP}_R T = {}^{PTH_FPP}_{SC_L} T \cdot {}^{SC_L}_R T$  using the DH-parameters of the PTH. The formula for the whole algorithm is expressed in the following way:  ${}^{PTH_FPP}_R T = {}^{PTH_FPP}_{SC_L} T \cdot {}^{SC_L}_G T \cdot {}^{GM}_R T$ .

## 4.2 Improvement of the calibration robustness by redundancy

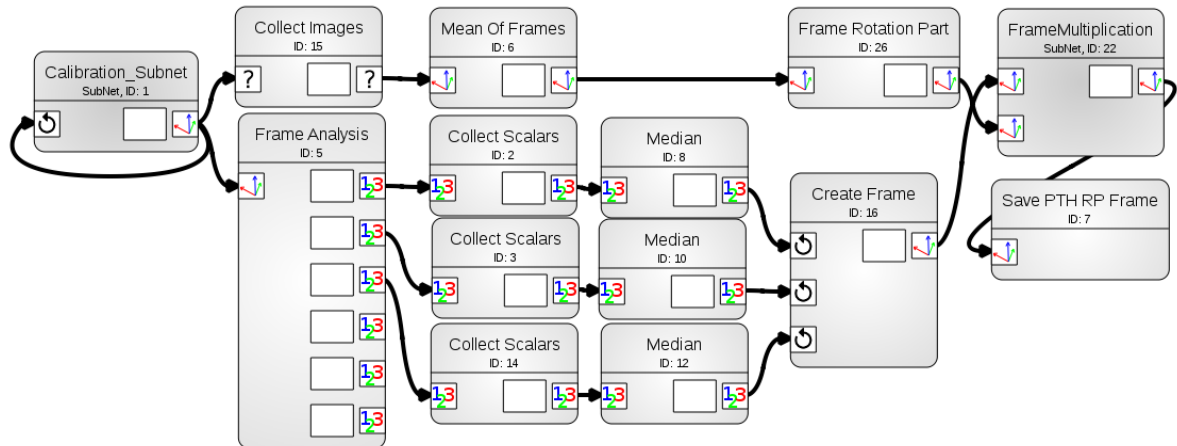


Fig. 4.3 ImageNet of redundant camera calibration to eliminate outliers.

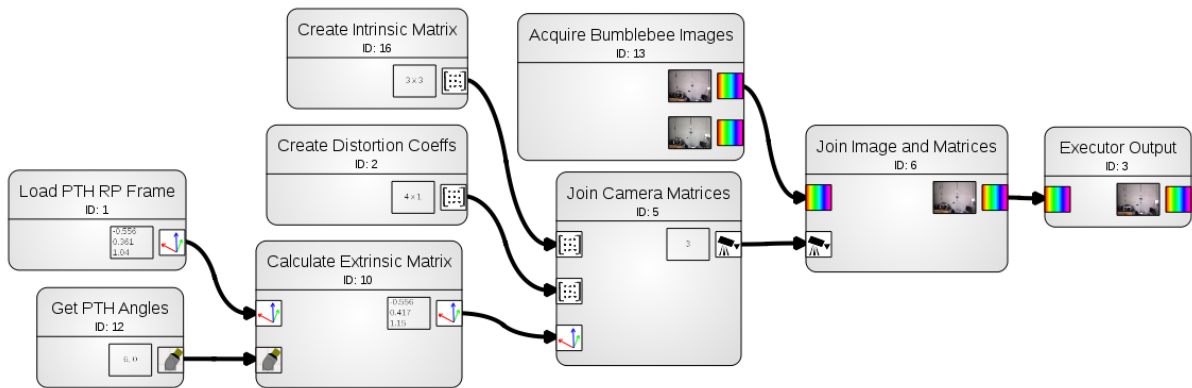
A fundamental concept of statistics is the probable existence of outliers. For a very high amount of samples in a set the standard deviation  $\sigma$  can be calculated which is a range around the mean value of the set.  $1\sigma$  contains 68.27% of all samples;  $2\sigma$  already contains 95.45% and  $3\sigma$  contains almost all samples with 99.73% but there is always a small probability for an outlier. Having this in mind, the robust calibration from the previous section can be further enhanced by running it multiple times to determine an even more accurate frame.

<sup>3</sup> DH – Denavit-Hartenberg parameters are used in the standard approach to describe robot configurations



The ImageNet in Fig. 4.3 gives an overview about the redundant robust camera calibration. On the left side the ImageNet from Fig. 4.2 runs in a loop for a definable amount of times. This amount should be an odd number to calculate meaningful median values. In every loop of the calibration, the resulting frame is collected to calculate the mean rotation of all frames (top part of the ImageNet). Moreover, on the bottom part of the ImageNet each loop's frame is split up into X-, Y-, Z-translation and rotation parts but only the translations are collected to calculate a median value for each. Afterwards, a frame is created from the median translations. This frame is then multiplied with the mean rotation frame to save the combined calibration frame on the hard disc.

### 4.3 Using the calibration data



**Fig. 4.4** ImageNet of left stereo camera image and matrices acquisition. The current extrinsic matrix is calculated by using the current PTH angles and PTH fixed point, obtained by calibration.

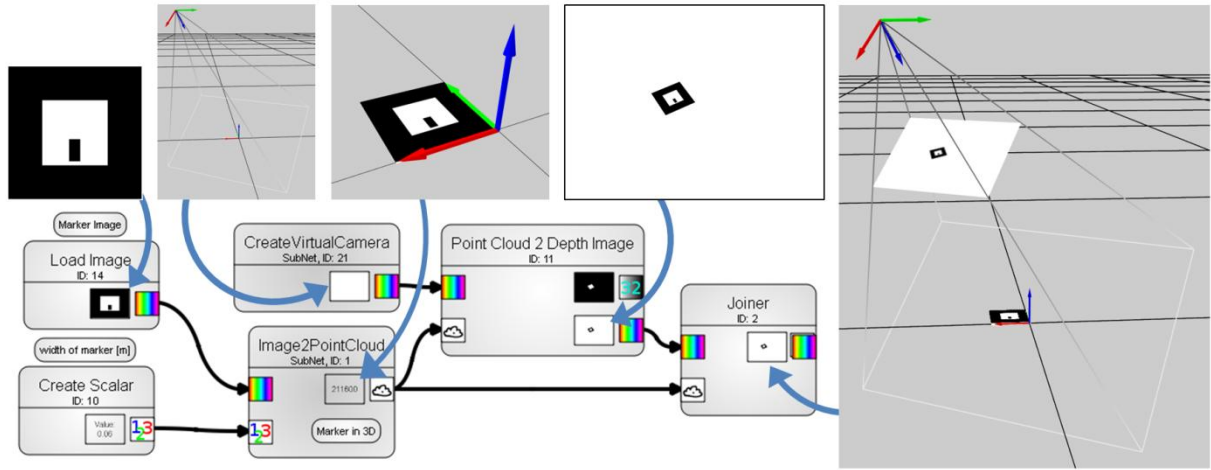
The ImageNet in Fig. 4.4 shows the usage of the calibrated PTH fixed point and the current PTH angles on the left side to calculate the current extrinsic matrix for any given pan-tilt angles. By adding the intrinsic matrix, distortion coefficients and the extrinsic matrix to the current stereo camera image, the combined matrix and image acquisition is completed. The presented ImageNet shows the image and matrix acquisition for the left camera image. This can be applied for the right camera image respectively using the same calibration data.

## 5. Performance Evaluation

The performance evaluation of the camera calibration of FRIEND with ImageNets is done in two steps. At first, the position accuracy of the general marker detection of the ARToolKit (Lamb, 2012) library is simulated. Second, the complete calibration process is performed 100 times on the FRIEND system to find meaningful values for the standard deviation in position and orientation.

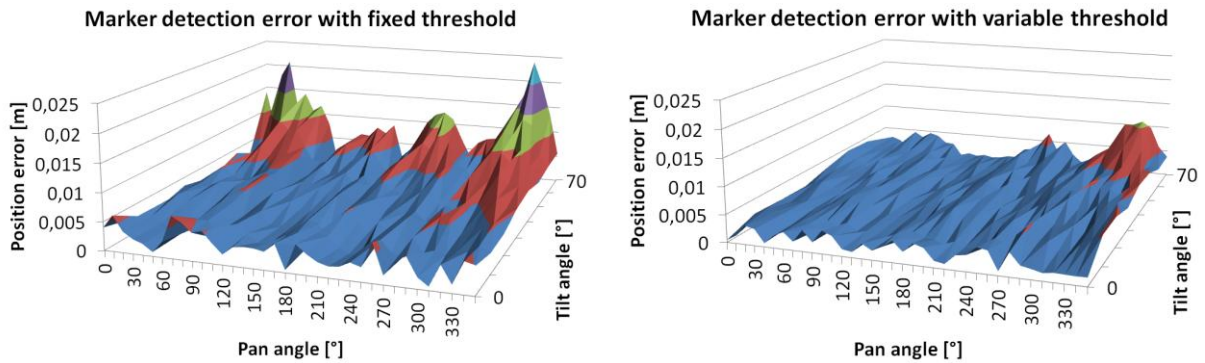
### 5.1 Accuracy of ARToolKit with simulated marker images

As a need for augmented reality applications, the accuracy of the ARToolKit marker detection has to be evaluated, which has been done with camera images (Abawi et al., 2004) but also with generated images (Pentenrieder et al., 2007). The latter experiments have been redone using ImageNets but especially for this application's parameters like camera resolution, marker size and camera to marker distance.



**Fig. 5.1** ImageNet to generate 3D projected marker images for known virtual camera positions.

To find out the location error of the marker detection library ARToolKit has been tested with a set of 288 generated images. The images have been created for a marker size of 60mm and a marker to camera distance of 600mm with a camera image resolution of 1024x768. In this setup, the projected markers have side lengths of at least 50 and up to 140 pixels. To create the simulated marker images, the ImageNet of Fig. 5.1 has been used. At first, it loads the marker image and places it in 3D at the origin. Afterwards, a virtual camera with the parameters of FRIEND's stereo camera is placed in the 3D environment at a fixed distance of 600mm. By altering the tilt angle of the camera between  $0^\circ$  and  $70^\circ$  and the pan angle between  $0^\circ$  and  $350^\circ$  with step widths of  $10^\circ$  a total of 288 images with known camera positions have been created.



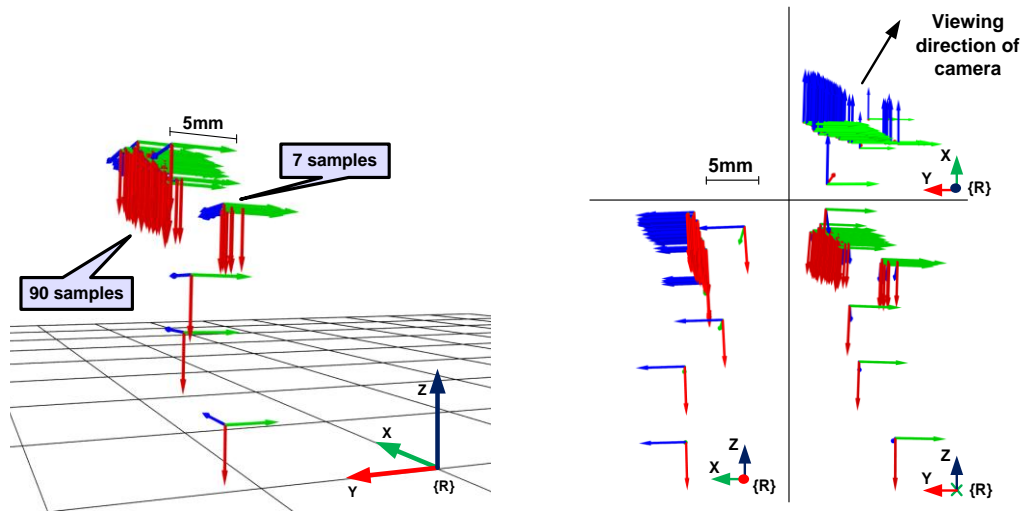
**Fig. 5.2** Position accuracy of marker detection with generated images. Left: marker detection with fixed threshold value of 40. Right: marker detection with variable threshold values between 40 and 80.

After that, these generated images and their known camera parameters can be compared with the ARToolKit marker detection to calculate theoretical errors for a camera-marker setup. This comparison has been done on the one hand with a fixed threshold value of 40 and on the other hand with a variable threshold between 40 and 80 using the ImageNet of Fig. 3.3. The variable threshold is selected by the ARToolKit quality measure. Fig. 5.2 shows position error graphs for a fixed threshold value of 40 (left) and a variable threshold value between 40 and 80 (right). The resulting graphs show both higher position errors for steeper tilt angles, which was expected as the marker becomes smaller in the image. When the variable threshold value

method is applied, the maximum position error declines from 24mm to 11mm. Moreover, the overall position error is decreased with the improved method and the most exact marker detection is expected for tilt angles between  $0^\circ$  and  $40^\circ$ . For the presented calibration application the pan angle will be at  $270^\circ$  and the tilt angle around  $30^\circ$ . In this area of the graph, the theoretical position error is optimal with 0 to 5mm, which can be considered as base noise from the ARToolKit library.

## 5.2 Repeatability of the proposed calibration method

The proposed calibration method has been executed 100 times for different redundancy amounts to find out the repeat accuracy. Capturing 100 data sets (without redundancy) needs around 10 minutes.



**Fig. 5.3** Repeatability results of the robust calibration method for 100 samples acquired without redundancy.

Fig. 5.3 shows a 3D view of the calculated PTH fixed point frames. Three obvious outliers can be seen and a group of 7 frames which are 5mm away from the median frame inside the main group. This result clearly shows the importance of using a set of calibration loops to calculate the median of the results to remove outliers, like it is described in section 4.2.

**Table 5.1** Standard deviations of the calibration frame for 100 calibrations using different redundancy levels.

Redundancy loops	X	Y	Z	X-rot	Y-rot	Z-rot	Execution time
1	1.03 mm	5.42 mm	6.06 mm	0.42 °	0.40 °	0.40 °	3.93 s
3	2.98 mm	3.66 mm	3.00 mm	0.16 °	0.15 °	0.27 °	12.83 s
5	0.88 mm	0.58 mm	2.10 mm	0.12 °	0.11 °	0.73 °	21.54 s

Table 5.1 lists the standard deviations along and around the three main axes. Obviously, more redundancy loops result in longer execution time but also in lower standard deviations and thus higher precision.

## 6. Conclusion

This paper shows that the requirements from section 1 have been fulfilled by ImageNets. In section 3 the advantages of ImageNets with respect to fast development and high execution performance have been shown. A Robust calibration method has been implemented using several layers to improve the accuracy. The accuracy itself has been evaluated on the one hand with generated marker images and on the other hand with 100 complete calibration cycles showing low standard deviations.

## References

- Abawi, D. F., Bienwald, J. and Doerner, R. (2004). "Accuracy in Optical Tracking with Fiducial Markers: An Accuracy Function for ARToolKit", In Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)
- Bradsky, G. and Kaehler, A. (2008). "Learning OpenCV Computer Vision with the OpenCV Library," O'Reilly, ISBN 978-0-596-51613-0
- Grigorescu, S.M., Natarajan, S., Mrona, D., Graeser, A. (2010). "Robust Feature Extraction for 3D Reconstruction of Boundary Segmented Objects in a Robotic Library Scenario", Proceedings of the 2010 IEEE-RSJ International Conference on Intelligent Robots and Systems IROS, Taipei, Taiwan, October 18-22
- Heckel, F., Schwier, M., Peitgen H.-O. (2009). "Object Oriented Application Development with MeVisLab and Python", in Lecture Notes in Informatics (Informatik 2009: Im Focus das Leben), 154:1338-1351
- Heyer, T., Grigorescu, S. M., Gräser, A. (2010). "Camera Calibration for Reliable Object Manipulation in Care-Providing Robot FRIEND", ISR/Robotik, Munich
- Lamb, P. (2012). "ARToolKit software library", <http://www.hitl.washington.edu/artoolkit/>
- The Mathworks (2012). MATLAB&SIMULINK Video and Image Processing Blockset, <http://www.mathworks.de/products/viprocessing>
- Pentenrieder, K., Meier, P., Klinker, G. (2006) "Analysis of Tracking Accuracy for Single-Camera Square-Marker-Based Tracking", In Dritter Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR
- Pilz (2012). PNOZmulti Configurator, <http://www.pilz.de/products/>
- Prenzel, O., Lange, U., Kampe, H., Martens, C., Gräser, A. (2012). "Programming of Intelligent Service Robots with the Process Model "FRIEND::Process" and Configurable Task-Knowledge"; Robotic Systems – Applications, Control and Programming, Ashish Dutta (Ed.), ISBN 978-953-307-941-7, InTech