Endoscope Calibration Setup

Moritz Heckscher
Im Schilf 11
CH–8044 Zürich
<moritzh@student.ethz.ch>
+41-76-509 04 08

March 2005

Advisors:
Mireille Reeff, Christian Wengert
Computer Vision Laboratory
The useful tips and support given throughout the course of this project by my two advisors, Mireille Reeff and Christian Wengert, and especially their patience while waiting for the results of this work is highly acknowledged.
Contents

1 Endoscope Calibration ................................................. 1
  1.1 Purpose ..................................................................... 1
  1.2 Calibration Algorithm ............................................. 1
  1.3 Drawbacks of the Current Calibration Setup ................. 3
  1.4 Goal of This Work .................................................. 7
    1.4.1 Mechanical Setup .............................................. 7
    1.4.2 Software and Calibration Pattern ......................... 7
  1.5 General Procedure ................................................ 8

2 Mechanical Image Acquisition Setup ............................... 10
  2.1 Requirements ....................................................... 10
  2.2 Choosing the General Design ................................... 10
  2.3 Geometric Requirements ......................................... 11
  2.4 Final Design .......................................................... 11
    2.4.1 Material .......................................................... 11
    2.4.2 Water Container ............................................... 13
    2.4.3 Shaft ............................................................... 13
    2.4.4 Base Plate and Endoscope Holding ....................... 13
    2.4.5 Bearings ........................................................ 14

3 Calibration pattern .................................................... 16
  3.1 Requirements ....................................................... 16
  3.2 Choosing the Layout ............................................... 16

4 Calibration Data Extraction Software ............................ 19
  4.1 General Procedure .................................................. 19
  4.2 Preprocessing ....................................................... 19
    4.2.1 Finding the Locations of the Grid Points............... 19
    4.2.2 Determining the Orientation of the Grid Coordinate System 22
    4.2.3 Difficult Situations ........................................... 22
  4.3 Mapping the Image Coordinates to Grid Coordinates ........ 25
    4.3.1 Concepts ....................................................... 25
    4.3.2 Obtaining Search Data ...................................... 26
## Contents

4.3.3 Measuring the ‘Fitness’ of Neighbor Points .......................... 32  
4.3.4 Strategy to Traverse the Grid ......................................... 34

5 Results .............................................................................. 42  
5.1 Mechanical Setup .............................................................. 42  
5.2 Software and Calibration Pattern ......................................... 43  
5.3 Calibration ................................................................. 45

6 Conclusion and Outlook ..................................................... 51

A MATLAB Source Code ....................................................... 54  
A.1 Overview ................................................................. 54  
A.2 Operation Modes .......................................................... 55
List of Figures

1.1 Optic of operating endoscope ........................................ 2
1.2 Distorted and undistorted endoscope image .......................... 2
   (a) Distorted .................................................. 2
   (b) Undistorted (reconstructed) .................................. 2
1.3 Current image acquisition ............................................. 3
   (a) Setup ...................................................... 3
   (b) Example images ............................................ 4
   (c) Example images (details) ................................... 4
1.4 Data extraction with the current software ............................ 5
   (a) Selecting the region ........................................ 5
   (b) (Semi-)automatic square counting ............................ 5
   (c) Entering a (virtual) distortion factor $k_c$ .................. 6
   (d) Incorrectly selected region .................................. 6
2.1 Complete mechanical setup ............................................ 12
   (a) Completely assembled ....................................... 12
   (b) In use at Universitätsspital Zürich .......................... 13
2.2 Water container ..................................................... 14
   (a) Top view, with shaft ....................................... 14
   (b) Bottom/side view .......................................... 14
2.3 Shaft to hold the calibration plate, axial shaft fixation .......... 15
   (a) Axial fixation .............................................. 15
   (b) Shaft ..................................................... 15
2.4 Bearings and base plate ............................................. 15
   (a) Side of container with scale ................................ 15
   (b) Opposite side of container ................................. 15
   (c) Base plate (with rotation scale, rods) ..................... 15
   (d) Bottom of container ..................................... 15
3.1 Three calibration pattern designs ................................... 17
   (a) With origin and several distance markers ................... 17
   (b) With many markers ........................................ 17
   (c) Chosen design ............................................ 18
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Image preprocessing steps (examples)</td>
<td>20</td>
</tr>
<tr>
<td>(a)</td>
<td>Original image (cropped)</td>
<td>20</td>
</tr>
<tr>
<td>(b)</td>
<td>Contrast-adapted</td>
<td>20</td>
</tr>
<tr>
<td>(c)</td>
<td>Blurred</td>
<td>21</td>
</tr>
<tr>
<td>(d)</td>
<td>Binary</td>
<td>21</td>
</tr>
<tr>
<td>4.2</td>
<td>Connectedness for a pixel</td>
<td>22</td>
</tr>
<tr>
<td>(a)</td>
<td>4-connectedness</td>
<td>22</td>
</tr>
<tr>
<td>(b)</td>
<td>8-connectedness</td>
<td>22</td>
</tr>
<tr>
<td>4.3</td>
<td>Problematic situations when determining the coordinate system</td>
<td>23</td>
</tr>
<tr>
<td>(a)</td>
<td>Big and small bar initially swapped</td>
<td>23</td>
</tr>
<tr>
<td>(b)</td>
<td>Merged points form a too large region</td>
<td>24</td>
</tr>
<tr>
<td>(c)</td>
<td>Bar merged with background</td>
<td>24</td>
</tr>
<tr>
<td>4.4</td>
<td>Distance differences among image regions and orientations</td>
<td>27</td>
</tr>
<tr>
<td>4.5</td>
<td>Obtaining search data (example)</td>
<td>28</td>
</tr>
<tr>
<td>(a)</td>
<td>Wanted (example): Horizontal search data in negative direction</td>
<td>28</td>
</tr>
<tr>
<td>(b)</td>
<td>Case 1: Desired search data stored in ( p ) itself</td>
<td>28</td>
</tr>
<tr>
<td>(c)</td>
<td>Case 2a: Next point in search direction along search orientation has been found already</td>
<td>29</td>
</tr>
<tr>
<td>(d)</td>
<td>Case 2b: Next point opposite to search direction along search orientation has been found already</td>
<td>29</td>
</tr>
<tr>
<td>(e)</td>
<td>Case 3a: Information for search orientation stored in preceding and succeeding point along perpendicular orientation</td>
<td>29</td>
</tr>
<tr>
<td>(f)</td>
<td>Case 3b: Information for search orientation stored in succeeding point along perpendicular orientation</td>
<td>29</td>
</tr>
<tr>
<td>(g)</td>
<td>Case 3c: Information for search orientation stored in preceding point along perpendicular orientation</td>
<td>30</td>
</tr>
<tr>
<td>(h)</td>
<td>Case 4: Information for orientation perpendicular to search orientation stored in ( p )</td>
<td>30</td>
</tr>
<tr>
<td>(i)</td>
<td>Case 5a: Information for orientation perpendicular to search orientation stored in preceding and succeeding point along perpendicular orientation</td>
<td>30</td>
</tr>
<tr>
<td>(j)</td>
<td>Case 5b: Information for orientation perpendicular to search orientation stored in succeeding point along perpendicular orientation</td>
<td>30</td>
</tr>
<tr>
<td>(k)</td>
<td>Case 5: Information for orientation perpendicular to search orientation stored in preceding point along perpendicular orientation</td>
<td>31</td>
</tr>
<tr>
<td>4.6</td>
<td>Traversing the grid</td>
<td>35</td>
</tr>
<tr>
<td>(a)</td>
<td>Find ( (0, -dY) )</td>
<td>35</td>
</tr>
<tr>
<td>(b)</td>
<td>Find ( (dX, -dY) )</td>
<td>35</td>
</tr>
<tr>
<td>(c)</td>
<td>Find ( (0, -2dY) )</td>
<td>36</td>
</tr>
<tr>
<td>(d)</td>
<td>From ( (dX, -dY) ) on, complete “top” horizontal line in positive direction</td>
<td>36</td>
</tr>
<tr>
<td>(e)</td>
<td>From ( (0, -dY) ) on, complete “top” horizontal line in negative direction</td>
<td>36</td>
</tr>
</tbody>
</table>
List of Figures

(f) From $(0, -2dY)$ on, complete “bottom” horizontal line in both directions ........................................... 37
(g) Complete lower part of vertical line through $(0, -2dY)$ in negative direction ........................................... 37
(h) Complete upper part of vertical line through $(3dX, -dY)$ in positive direction ........................................... 37
(i) Search perpendicular to longer line in positive direction ................................................................. 38
(j) Search perpendicular to longer line in negative direction ................................................................. 39
(k) Search perpendicular to shorter line (upper and lower part, in both directions) ........................................... 40
(l) Try to find missing points ..................................................................................................................... 40
(m) Final result ........................................................................................................................................... 41

4.7 Results for different settings of the parameter allowTolerantFitness ....................................................... 41
(a) allowTolerantFitness=0 ....................................................................................................................... 41
(b) allowTolerantFitness=1 ....................................................................................................................... 41

5.1 Examples of detection results .................................................................................................................. 46
(a) Example 1 ............................................................................................................................................. 46
(b) Example 2 ............................................................................................................................................. 46
(c) Example 3 ............................................................................................................................................. 47
(d) Example 4 ............................................................................................................................................. 47
(e) Example 5 ............................................................................................................................................. 48
(f) Example 6 ............................................................................................................................................. 48

5.2 Sample calibration results for old and new setup and software ................................................................ 49
(a) Old setup, old pattern, old software ....................................................................................................... 49
(b) New setup, new pattern, new software ................................................................................................... 49

5.3 Reprojection example ............................................................................................................................. 50
(a) Reprojection .......................................................................................................................................... 50
(b) Reprojection error .................................................................................................................................. 50
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>Overview of the source code files</td>
<td>54</td>
</tr>
<tr>
<td>A.2</td>
<td>Settings for verbosity to control program output</td>
<td>55</td>
</tr>
<tr>
<td>A.3</td>
<td>Warning message identifiers and corresponding messages</td>
<td>56</td>
</tr>
</tbody>
</table>
1 Endoscope Calibration

1.1 Purpose

Endoscopes such as the one used for this work (figure 1.1) are frequently used in medical applications to acquire information about a patient’s intestines. Their small size provides an enormous advantage for the patient because only a tiny body opening is needed to insert the endoscope which means that large operation wounds can be avoided.

Unfortunately, however, the fish-eye lenses in the optical setup which are needed to obtain a large field of view despite of the small size of the endoscope generally induce strong nonlinear distortions in the images (figure 1.2a). For biomedical applications such as 3D reconstruction based on sequences of 2D endoscopic images – or any computer vision task that makes use of quantitative image analysis –, it is crucial to know the model-based parameters that characterize the camera for a useful processing of the images: If the distortion parameters are known, the lens distortion effects can be removed from the images (figure 1.2b).

1.2 Calibration Algorithm

Usually the needed camera parameters such as focal length, radial and tangential lens distortion or location of the center point in the camera are computed using images of a predefined pattern that are taken from different viewpoints.

The calibration algorithm used [2] can calculate these so-called intrinsic parameters of the camera if the dimensions of the grid (grid spacing in both directions) and, most important, the mapping between the image coordinates and the grid coordinates of the data points are known. I.e. it is necessary to know for each of the points on the calibration grid (that is visible on the images and to be used for calibration) which image position it has and how far it is in reality from the origin, both in x and y direction (in a coordinate system attached to the plate). The origin can be chosen freely but obviously needs to be attached to the same grid point for all images in a sequence.

The calibration algorithm [2] is a gradient-descent method which uses a least squares fitting and calculates the needed parameters by looking at the correspondences of the data points between the different images. Although it is ideal to have the exact same set of grid points visible in all of the calibration images (which is enforced by the currently used software, see section 1.3), the algorithm itself does not require this strict condition.
Figure 1.1: The optics of the mini-operating laparoscope used in this work. It has a diameter of 3.8 mm and an optical axis tilted by 12°. It is produced by Richard Wolf (ref. 8746.401).

Figure 1.2: (a) The images from the endoscope are severely distorted, both radial and tangential lens distortion occur. (b) After the calibration process, the distortion parameters are known and can be used to compute an undistorted reconstruction.
1.3 Drawbacks of the Current Calibration Setup

The enormous advantage of the algorithm is that the so-called extrinsic parameters need not be known, i.e. it is not necessary to know the exact position and orientation of camera and plate relative to each other.

There are also ways to compute the camera model parameters dynamically during the image reconstruction process, but this introduces more uncertainties, so it is advantageous to calculate them in a separate calibration step before the actual image reconstruction.

1.3 Drawbacks of the Current Calibration Setup

The calibration procedure employed so far has three major drawbacks:

1. The images required for the calibration algorithm are taken using a very simple mechanical setup (figure 1.3a) or freehand. The image quality is not satisfying as camera shakes can easily happen. Camera settings such as focus length may not be altered in one calibration sequence, but with the current setup accidental changes in these parameters can easily occur when changing the plate position between images. Furthermore, not all desired plate positions can be reached with the setup as the endoscope holding obstructs the way for the plate to move. This may result in poor images in which the calibration pattern that is ideally to cover the whole image region is only partly visible (figure 1.3b). Generally speaking: It is very difficult to obtain images with many data points in good quality for the subsequent use with the calibration software.

Figure 1.3: Current image acquisition. (a) In the current very simple setup, a glass bowl is used to hold the plate with the pattern and ‘adjust’ the appropriate angle. . .
Chapter 1. Endoscope Calibration

Figure 1.3: Current image acquisition (continued). (b) Since bowl and endoscope holder block each other, often the plate cannot be positioned as desired, the pattern does not cover the whole image region and the surroundings are visible which are useless for calibration. (c) The pattern is imprinted on glass and the strong, punctual internal light source of the endoscope is used, so reflections are visible on most images. The calibration software uses the location of the corners on the chessboard as data points and requires using the same corners in all images. As the reflection spot appears on different regions (orientation of plate changes from image to image), the choice of a common region not affected by reflection is limited. Furthermore, the corner locations can be obtained only imprecisely because they either appear disjoint or 'grow together'.

4
2. The calibration pattern that needs to be used with the current software has proven to be not ideal because the detection algorithm relies on certain image features of this pattern that are unfortunately not clearly recognizable in the endoscopic images, both because the features are strongly affected by the image quality and the distortions and because a reflecting plate material is used (figure 1.3c).

3. The calibration software \[1\] requires the user to manually and precisely select several points (usually more than 15) in each of many images (usually more than eight) to extract the necessary input data for calibration (mapping between image and grid coordinates) which is a very time-consuming (usually more than ten minutes for a sequence of eight images), tedious and error-prone process (figures 1.4a-d).

![Figure 1.4: Data extraction with the current software.](image)

(a) Selecting the region  
(b) (Semi-)automatic square counting

The crosses should be close to the image corners

Figure 1.4: Data extraction with the current software. (a) The user needs to click on four corners in each of the calibration images to define a rectangular region. The first point clicked on must represent the same corner on the physical calibration grid and defines the origin of the coordinate system. The order in which the points are clicked must also be identical as it defines its orientation. Since the chessboard pattern is symmetric, it is often hard to identify origin and direction on the images and to comply with these rules. (b) The software then tries to count the squares in the region. This practically always fails with the endoscopic images as the image quality is not good and the straight line borders defined by the four points cannot capture the appropriate curved image region. The user then enters the number of squares in each direction and the software marks guesses for the corners which represent the data points to be used in the calibration. . .
Figure 1.4: Data extraction with the current software (continued). (c) Since the obtained corner positions are almost always too imprecise, a guess for the distortion factor $k_c$ needs to entered, and the crosses are redrawn at positions calculated using this (virtual) distortion. Several guesses need to be entered most of the times to reach somewhat satisfying results. Most often, however, the results are too poor, so the user must click on every single square corner in the region for every single image to define the precise corner locations by hand. (d) If a wrong point is accidentally clicked, e.g. by missing a row, the whole image needs to be corrected in manual mode.
1.4 Goal of This Work

Therefore the task of this project is to improve the calibration process by developing both a mechanical setup for image acquisition and a data extraction software.

1.4.1 Mechanical Setup

The mechanical setup should fulfill the following requirements:

1. It must allow to present a pattern printed on a plate to the camera in different positions. The different plate positions should cover enough space in 3D within the range of the endoscope.

2. The plate with the pattern must be exchangeable.

3. The device will be used in hospitals. Therefore disinfection or sterilization must be possible (the device must be able to withstand the temperature and the pressure in an autoclave).

4. It must be possible to take the images with the plate and the endoscope immersed in water. This is to simulate the use of the endoscope after the calibration process for some applications more realistically, taking into account the different refraction index of water: For instance, water is used to fill and stretch the placenta when taking endoscopic images of it.

5. The setup should facilitate an easier and more precise image acquisition process than the currently used manual procedure.

6. It is intended to be a prototype. It should be simple and can be designed for manual operation. Future motorization of the moving parts without fundamental design changes should be kept in mind, however.

7. The device should primarily be suitable for the endoscope used in the research group whose characteristics such as the inclined field of view must be taken into consideration; but it should also be possible to use other endoscopes with the same setup without many modifications.

1.4.2 Software and Calibration Pattern

The software to be developed and the pattern design to be used with the software are to fulfill the following purpose:

1. The pattern design should provide the necessary information for the automated data extraction such as plate orientation and allow precise data extraction.
2. The software must extract the needed data and make it available for the calibration algorithm in the required format.

3. It should require as little user interaction or a-priori knowledge about the optical parameters of the system as possible and process the images automatically.

4. The calibration results obtained with the data should be as precise as possible.

1.5 General Procedure

The main steps involved in the calibration process are:

**Image acquisition:** Several images need to be taken from a predefined pattern from several viewpoints. It is crucial to get a good image quality in this step since all remaining steps rely on this first step. The pattern must be well-focused and the lighting must be good. On the one hand, strong lighting improves the image quality because a larger region is visible and illuminated, but it is also likely to cause reflections on the pattern. These should be avoided because then the pattern features are not visible in this region anymore.

The endoscope, just like any other optical component, has a limited depth range in which objects are in focus. Since the calibration algorithm requires the plate with the pattern to cover as much space in 3D as possible to get good calibration results, this poses a problem for the image acquisition. Since the focus settings may not be altered between the different images (otherwise the camera parameters would change and one would have different parameters for each image), a common setting has to be found that provides good results for as many of the positions used as possible. This is not very easy to accomplish since the plate generally covers a large depth range in different regions in the field of view of the camera. Furthermore, the optical axis of the endoscope is tilted, so the combination of a tilted field of view and a tilted pattern can cover large depth ranges. This generally means that a compromise has to be found between focusing on the center region of the field of view to get a good image quality there (the most important part of it) and focusing on the outer areas rather than the center to make as many features of the image recognizable as possible.

In addition, the cameras used in endoscopy usually have only PAL resolution (768×576 pixels) and provide interlaced image frames, i.e. two separate half-frames are taken sequentially with a small time delay of about 20ms and then combined to one image. While this is not disturbing for a human observer when watching movements through the camera, it can cause poor quality in still images such as the ones needed for calibration if the camera is not fixed well enough and moves a little:
1.5 General Procedure

The half-images do not fit together well. It is therefore crucial to avoid motion of the camera during image acquisition as much as possible. For this, a mechanical setup to fixate the endoscope is desirable.

**Data extraction:** In a second step, the data points that are to be used for the calibration algorithm must be extracted from the images. The mapping between image and grid coordinates that is necessary for the algorithm (section 1.2) needs to be achieved in some way.

Generally, the special features of the pattern serve as an aid in this process. The images may need to be preprocessed in some way, e.g. with contrast or brightness adaptation or low-pass noise filtering, to enhance the recognizability of the pattern features. Again a compromise needs to be found between too much and too little preprocessing since it can make certain characteristics more clearly visible, but at the same time also alter the image enough that the information that can be gained from it becomes too imprecise.

The extraction of the data points to be used for calibration can be done manually or automatically. Both ways have advantages and disadvantages. While manual data extraction can deal especially well with difficult situations such as not clearly visible features or very different circumstances in different images (e.g. mixture of very strong and weak distortions effects because of very different plate orientations), it may also be imprecise because the data point extraction cannot always be exact. The largest disadvantage is clearly that manual data extraction generally takes considerably longer than automated extraction. The speed advantage of automated processes can, however, be meaningless if the automatization cannot handle difficult situations at all or – even worse – results in incorrect data extraction that is not observable or controllable by the user.

**Calibration:** Finally, the extracted data is used to calculate the desired camera model parameters with a suitable calibration algorithm.

With the exception of the last point, this work tries to improve the currently used method in all respects: To obtain images of better quality in an easier and more reproducible manner, a mechanical setup is designed (section 2), and to ease data extraction, a pattern suitable for automated detection is chosen (section 3) and a special software algorithm for automated data processing is developed (section 4).
2 Mechanical Image Acquisition Setup

2.1 Requirements

The requirements for the mechanical setup can be grouped into three categories:

1. Suitability for use in hospitals;
2. sterilization
3. water contact.

The most problematic requirement was the combination of water contact and sterilization. Since water has a very low viscosity and any sealing must therefore be rather good, one would ordinarily use materials such as rubber or silicone for the sealings. This, however, is not possible if the device is to be sterilized since these materials cannot withstand the high autoclave temperatures (121°C).

Therefore, a simple setup with as few places as possible requiring sealing needs to be chosen. For the use in the autoclave, the device will have to be disassembled so that all regions can be reached by the steam. Thus a design with few simple parts is advantageous.

2.2 Choosing the General Design

Initially it was also thought about using e.g. a screen on which points were to be displayed by a laser beam (or another kind of display) instead of a mechanism to present a plate to the camera in several positions. The advantage of this approach would have been that much less moving mechanical parts would have been necessary since the movement of the pattern would have been simulated. Also it would have been easier to allow pictures to be taken underwater.

It is, however, not possible to use this approach. The object to be captured by the camera needs to be 3-dimensional because the distortion is dependent on all three dimensions. Alternatively a 2D projection on a screen could be used, but the problem is that this projection would have to be calculated using the intrinsic camera parameters – which are unknown.

Therefore, a mechanical solution to present the plate to the camera was sought. Different possible designs were conceived. They mainly differed in which part is moved and...
which is kept still. In theory, it is possible to either move the plate by itself in a water container, to move the whole water container or to move the endoscope. Moving the endoscope instead of the plate was considered inappropriate since there are cables for the camera and the light attached to the endoscope. These cables prevent an easy movement, and if they are touched, the camera orientation relative to the optics of the endoscope is easily changed – which must be avoided during calibration since then the calibration parameters (center point of camera) are changed.

Thus two alternatives remained: To move only the plate in the water container or to move the whole water container. Finally, a mixture of both was picked: One degree of freedom (plate tilt angle) is adjusted in the water, the other one (plate rotation) is achieved by rotating the whole container. This variation was chosen because only few sealings are required and because the required movements can be performed using two independent shaft to collar connections. By keeping the two rotational degrees of freedom separate, the required shaft to collar connections and the bearings can be kept very simple. The connection is more likely to be watertight and can easily be disassembled.

2.3 Geometric Requirements

As the field of view of the endoscope is tilted, this poses a problem. The criterion that the field of view should not reach the edges of the plate could not be fulfilled for the given initial requirements: A tilted field of view up to 30°, a plate inclination of up to 45°, a distance of 3 cm between endoscope and plate as well as rotational angles from 0 to 360° do not allow to fulfill this; the plate would have to be infinitely large.

Therefore a compromise based on previous experiences about the used endoscopes was made. The goal was to have enough of the pattern in the field of view for the most important settings only. Therefore, the plate size was fixed to be 15 cm × 15 cm.

2.4 Final Design

The resulting complete device can be seen in figure 2.1.

2.4.1 Material

The requirement of using the setup in hospitals allowed only non-toxic, non-corroding materials. Therefore stainless steel, aluminum (production of large parts out of aluminum is easier than out of stainless steel) and bronze are the main materials used in the developed device. Lead, which is contained in some bearings, was not used.

The high temperature in the autoclave also meant that glue cannot be used as even special high-temperature components (such as Loctite Hysol M-21HP, 31CL, or 121HP)
Chapter 2. Mechanical Image Acquisition Setup

Figure 2.1: Complete mechanical setup.

(a) Completely assembled
2.4 Final Design

(b) In use at Universitätsspital Zürich

Figure 2.1: Complete mechanical setup (continued)

can only withstand the temperatures about three or four times. Therefore all parts must be mounted with screws, by welding or soldering.

The main components of the design are the shaft to hold the calibration plate, the water container, the bearings and a base plate with an endoscope holding.

2.4.2 Water Container

The water container (figure 2.2) is produced by folding a steel sheet and welding plates to it to form the sides. The hole for shaft and bearings is made after welding because only then the required tolerances to achieve a watertight fitting can be complied with.

2.4.3 Shaft

The shaft to hold the calibration pattern (figure 2.3b) is milled out of one piece and includes a handle and a scale needle to adjust the plate tilt angle. The middle part is made even on one side to hold the plate. The plate is fixed with two small steel blocks that are screwed to the shaft. This way, various plates can be attached to the shaft. To fix the shaft axially in the bearings in the water container, a screw (figure 2.3a) is used.

2.4.4 Base Plate and Endoscope Holding

For easier production and to reach less weight the base plate is designed out of aluminum instead of stainless steel. Two 12 mm rods are screwed vertically to it to which standard laboratory equipment can be attached to hold the endoscope (figure 2.1). Thanks to this approach holdings for several endoscope types can be connected to the device, it is not only suitable for a special type. Tow rods instead of one provide more flexibility when trying to fix the endoscope, additional light sources or other devices and also more
stability. The scale on the base plate for the rotational angle is engraved and one edge of the water container is used as the needle. This means less parts.

2.4.5 Bearings

Between base plate and water container, a friction bearing is used consisting of a bronze ring in the plate and a steel shaft at the container (figures 2.4c,d). To reduce the friction between the base plate and the water container, three Teflon plates are used. These plates are also fixed with screws since gluing is not possible.

For the shaft, a special friction bearing type was chosen that makes the shaft to collar connection watertight and is resistant to high temperatures (figures 2.4a,b). It allows easy disassembly of the machine for disinfection/sterilization in regular intervals. When designing the fitting for it, it had to be taken into account that enough friction must be present for the shaft to remain in the selected position.

Using friction bearings is not a problem as rotational speeds, axial and radial stress and expected number of rotation cycles are all very low. Some types of friction bearings contain lead; these are not used since they are not suitable for use in operating conditions.

Advantages of the chosen bearings are that they need no maintenance and no lubricants (oil, fat) which would have problems with the high autoclave temperature.
Figure 2.3: Shaft to hold the calibration plate (b), axial shaft fixation (a)

Figure 2.4: Bearings and base plate
3 Calibration pattern

3.1 Requirements

The task of the calibration pattern is to allow an easy and robust extraction of the data points needed for calibration. The representation of the data points on the pattern should be as ‘immune’ to the effects of distortion as possible so that the positions can be determined precisely.

For the automatic mapping between image and grid coordinates which is to follow afterwards, additional features besides the actual data points need to be included in the pattern to allow the determination of the orientation of the pattern. These markers must be large and unique enough to be safely identified, but should at the same time not cover too much space on the pattern since the field of view of the endoscope is rather small and too large markers mean unnecessary loss of actual data points.

3.2 Choosing the Layout

It was known from the experience with the old chessboard calibration pattern that it is problematic to use a pattern with intersecting or connected components or components that are very close to each other. This is because it may be difficult to distinguish the components or determine their positions accurately (figure 1.3c, p. 4). In addition, the recognition of separate components is more easy to achieve. Therefore choices such as a simple grid built of many intersecting lines (with the data points being the intersections) were not considered and it was decided to rather choose a pattern with distinct, spatially separate units.

Filled circles on the calibration pattern are not affected as severely by the distortion effects as, e.g., polygons. They are basically turned into ellipses, but their centroids are not changed as much as those of polygons. For the latter, the centroid shift effect much more depends on the camera perspective and the distortion. Therefore a grid of simple filled black circles on a white background was picked.

To resolve the orientation ambiguity of the symmetric pattern on the images and to allow the determination of location and orientation of the grid coordinate system, additional markers need to be included. Three main layouts that were considered as well as the respective ideas behind them are illustrated in figure 3.1.
3.2 Choosing the Layout

(a) With origin and several distance markers
(b) With many markers

Figure 3.1: Three calibration pattern designs. (a) In this pattern, a cross marks the origin of the coordinate system and additional markers on the diagonal (missing points in the grid) are used to determine the orientation and distance from the origin. The main reason why this pattern was not chosen is that it is more complicated and unreliable to detect the ‘holes’ as markers, i.e. the white regions on white background between the black circles, than to detect other black objects. If the origin marker is furthermore visible in all images (which was fixed as an requirement), it is not necessary to encode information about the distance from the origin in the markers as this information can be gained from the grid itself. (b) The basic idea behind this design was that several markers besides those for the origin itself should be included in order to obtain an initial guess about the image distortions based on the markers. This information was to be used to allow the detection of the actual data points, the circles, with the help of an algorithm that uses the initial distortion guesses. Because the choice fell for an algorithm that does not use distortion guesses and because practical tests revealed that too much space on the images was wasted by the many markers and that it was basically impossible to ensure that all markers are visible on all images for the required plate positions, this design was not chosen. . . .
Chapter 3. Calibration pattern

Figure 3.1: Three calibration pattern designs (continued). (c) The chosen design is a simple grid of points with a bigger and a smaller rectangular bar as markers to tag the two coordinate axes. The larger bar identifies the orientation of the X axis, the smaller one that of the Y axis, the direction of the axes is determined by the fact that they build a right-hand-system. The distance from the origin is determined by counting the number of grid points along the X and Y axes and knowing the grid spacing.

The design in figure 3.1c was chosen because it is simple, contains all necessary information, its markers (the bars) are distinguishable by shape and size from each other and the points and do not cover much room on the pattern and because it was decided to use a search algorithm not based on distortion estimations (section 4.3.1).

The grid spacing was determined to be 2 mm with a point diameter of 1 mm based on previous experiments with the used endoscopes in order to have enough data points which are still well-separated. The larger of the two bars covers three points so that its centroid appears close to that of the middle point on the image. This centroid can then be used as a good starting point for the search.

Both bars are rectangular, therefore their centroids are altered by perspective, distortion and binarization more than those of the points. But it was decided to rather make the markers very small but still distinguishable by shape and size and only use them to determine the coordinate axes than to make them larger, given them better geometric properties and also use them as data points (this would waste space for other data points). The smaller bar therefore only covers two points and has a centroid which comes to lie ‘between’ the grid points.

Based on the results that could be obtained with the software (section 5), the pattern design and the dimensions of the grid and the bars are well-chosen. Even if in rare situations the bars are not well-distinguishable by, e.g., their size, the pattern design still allows to correctly detect the coordinate system (figure 4.3a, p. 23).
4 Calibration Data Extraction Software

4.1 General Procedure

The task of the software is to achieve the mapping between image points and grid coordinates that is needed for the calibration algorithm. There are two main steps involved to achieve this:

1. First some preprocessing of the images needs to be done so that the special features of the pattern can be recognized and the orientation of the pattern as well as the locations of the actual data points on the image can be identified (section 4.2).

2. Knowing the image coordinates of the data points and the orientation of the grid coordinate system on the image, the mapping between the image and the grid coordinates needs to be done (section 4.3).

4.2 Preprocessing

The goal of the operations grouped together here under the term of “preprocessing” is to identify the grid points and the bars on the pattern as distinct units, to obtain their locations and to determine the orientation of the grid coordinate system on the image.

4.2.1 Finding the Locations of the Grid Points

In order to do this, the original image is first cropped and converted into a binary image, i.e. an image which only contains black and white pixels, in several image conversion steps for which examples are shown in figure 4.1. Each connected region of black pixels (cf. figure 4.2) is then considered an individual object whose centroid (‘the center of gravity’) defines its location on the image. The method used to identify these regions is known under the term connected-component labeling or connected-compound labeling.

The objects are then filtered by their area (= number of pixels) so that very small black regions that arise from picture artifacts (noise) or belong to very off-center points with an area below the parameter minArea are not considered. The image locations of the grid points which are to be considered in the mapping are thus defined.
Figure 4.1: Image preprocessing steps (examples). (a) The image is first converted to gray-scale (color-information discarded; effect barely visible as calibration pattern is black and white only) and the large black borders on the original image outside of the region captured by the endoscope are cut off since the following operations (e.g. determination of a threshold) depend on the relative numbers of pixels for each gray level (else there would be too much black in the image). (b) The image is then contrast-adapted. ...
4.2 Preprocessing

Figure 4.1: Image preprocessing steps (examples; continued).

(c) Blurring is applied to achieve an averaging filter effect (noise as single white pixels in large black regions is filtered out).

(d) A gray level threshold is then determined to make the image binary (only black and white). To even increase the separation between black and white, the gray-scale value for each pixel (in a range from 0 to 1) is squared before comparing it to the threshold, making ‘black more black’ and ‘white more white’. Note that some points in the off-center regions can merge with other points or the surroundings and are thus not recognizable as individual points.
4.2.2 Determining the Orientation of the Grid Coordinate System

To distinguish the two bars from the points and each other, another property of the pixel regions besides area is additionally taken into account: the “major axis length” which is the length of the major axis of the ellipse that has the same normalized second central moments as the pixel region. With the help of this parameter and the orientation of the bars, the grid coordinate system can be identified: The origin is the centroid of the big bar, the X axis runs parallel to the orientation of the big bar, the Y axis runs parallel to the orientation of the small bar; the direction of the X and Y axis is determined by the fact that they build a right-hand-system (using the cross product).

4.2.3 Difficult Situations

One problem involved in identifying the points and bars appropriately is the noise in the images: The regions that are ideally black and white only on the calibration grid appear in several gray-scale levels on the images that are not always smoothly distributed: Sometimes bright pixels appear in very dark regions and vice-versa. This noise is removed to obtain filled objects (e.g. black regions without white ‘holes’) in order to calculate their centroids. Of course, the obtained locations of the centroids depend on the way the noise is filtered.

The centroid locations are in general affected by any of the steps to make the image binary, e.g. thresholding. In the binary images, a centroid can be shifted only slightly from its ‘true’ location (which cannot be determined precisely anyway), but it is also possible that depending on the circumstances one object in the original image is split.
4.2 Preprocessing

up into several separate objects in the binary image (less common) or that originally separate objects ‘grow together’ (more common; this frequently happens in the off-center regions where the image quality is poor and the grid points are very close to each other; see figure 4.1d).

Another problem is that the big and the small bar and thus the image orientation may not safely be identified by looking at the size, the major axis length or the dimensions of the bounding box of the objects alone. Tilted perspectives, distortions and image preprocessing (binarization) may in rare cases result in the smaller bar (smaller in reality) being detected as the bigger one (and vice versa; see figure 4.3a). Therefore additional checking of the obtained coordinate system became necessary to correctly identify the image orientation. Other possible obstacles in determining the coordinate system correctly are depicted in figure 4.3b and 4.3c.

If the coordinate system cannot be determined correctly, obviously the following mapping results will be incorrect. If these rare cases happen, the user can ignore the detection results for that particular image.

![Figure 4.3: Problematic situations when determining the coordinate system. The binary image and the resulting coordinate system are shown.](image)

(a) Big and small bar initially swapped

In very rare cases, the larger bar (larger in reality) may be considered the smaller one judging by area or major axis length. In the example, the bar corresponding to the larger bar in reality has a smaller major axis length (72.4 pix) on the picture than the other bar (81.1 pix). Despite of this, the orientation can be determined correctly using a special test.

---

*aThe test is based on the fact that the angle for the connection line between the centroids of the big and the small bar should be more similar to the orientation of the small bar than that of the big bar (in all cases ignoring the direction, i.e. allowing flips by 180°).*
Figure 4.3: Problematic situations when determining the coordinate system (continued). (b) In other rare cases, points may merge to form a large region which is erroneously considered one of the two bars. In the shown example, three points are barely connected (by just about one pixel) to form a longish region which is regarded as the big bar. Note that the ‘real’ big bar is treated as the small bar and the ‘real’ small bar is taken to be an ordinary point (see red mark). (c) If either or both bars have merged with the background due to the binarization process or are not visible at all, the coordinate system can obviously not be determined correctly.
4.3 Mapping the Image Coordinates to Grid Coordinates

4.3.1 Concepts

Two different approaches to develop the algorithm to map image points to grid coordinates were initially considered: A method which somehow makes use of distortion in the images in the mapping process and another one which does not.

For the first alternative, a distortion-based method could have, for example, gained initial information about the approximate distortion coefficients by looking at the markers, could have produced a distorted image of the undistorted known grid in the proper orientation as determined by the grid coordinate system, could have compared this simulated distorted image with the real distorted endoscopic image and made the mapping by finding correspondences between the points. Possibly this process would have to be iterated; or different distortions would have to be tried out and the results for those for which the correspondences are the best would have to be chosen.

This seemed to be rather complicated to develop, would probably not be very reliable and would possibly require some a-priori knowledge about the images (distortions) which is not desired for automated processing. In addition, as stated before, the pattern shown in figure 3.1b intended to be used for this approach would not have been a good choice because proper image acquisition would have been difficult and less data could have been used for calibration.

Therefore a different approach that is not dependent on distortion guesses, needs very few parameters and practically no a-priori knowledge about the image conditions and is applicable to a wide range of problems (image conditions) was developed. Its main characteristics are:

- It is based on a local search and local similarity on the image;
- it uses information about relationships between previously found points to advance further and infer about the positions of unknown points;
- it takes special provisions for cases when nothing or little is known about these relationships.

The main components of the mapping procedure can be summarized as follows: 1

Obtaining search data: Given the location of a point and some knowledge about a “reference relationship” – i.e. how the point can be reached from another known point or how other known points in the local neighborhood are related to each other –, determine the most appropriate data to begin the search in a given search orientation and direction to find the next point in the grid (section 4.3.2).

1To find out which step is implemented in which MATLAB source code file, see table A.1 on p. 54.
Chapter 4. Calibration Data Extraction Software

Determining relationships to neighbors: Find all neighbors within a certain range of the current point (in a square region extending scope pixels from the current location in all four directions). Determine the relationships between them and the current, given point (how to reach the neighbors from the current point).

Measuring the ‘fitness’ of neighbors: Measure the similarity between the reference relationship and the relationships to the neighbors with a ‘fitness’ function. Accept the ‘fittest’ neighbor (also having a certain minimum fitness, minFitness) to be the next point in the search orientation and direction (section 4.3.3).

Traversing the grid: Traverse the grid in a smart way so that information about the relationship between the current point, the just discovered point and other previously found points can be used as well as possible to continue the search (section 4.3.4).

4.3.2 Obtaining Search Data

For a successful search, appropriate search data to compare to is vital. The question arises: Which data needs to be collected because it provides useful information and characterizes the relationships appropriately?

Since the curvature of the grid coordinate lines on the image is the main difference to an ordinary, undistorted rectangular grid, information about the angle of the connection line between two points and the horizontal \((x\) axis on image coordinate system) was chosen as the key characteristic from the beginning on.

The relationship between two points is further described by the distance between them. As the distance between grid points also varies considerably between different image regions (figure 4.4), it seemed appropriate to also collect this kind of information and use it as search data.

Examination of the endoscope images and several longer tests also showed that both angle and distance information can be very different between the horizontal and the vertical orientation; i.e., the distance and angle for a point and the previous one in a horizontal grid line can be very different than those values for the same point and a previous one in a vertical grid line. This can also be observed in figure 4.4.

Therefore distinguishing between data for the horizontal and the vertical direction also became necessary so that each point can store up to four values:

1. The angle between the horizontal image axis \(x\) and the connection line between a horizontal “predecessor point” – i.e. a point with a smaller \(X\) grid coordinate, but the same \(Y\) grid coordinate – and the point itself.

\(^{2}\text{"Very different" in this context is supposed to mean that the deviations between horizontal and vertical orientation were too large to obtain useful results when applying the various fitness functions tried out (section 4.3.3) without distinguishing between horizontal and vertical orientation.}\)
4.3 Mapping the Image Coordinates to Grid Coordinates

Figure 4.4: Distance differences among image regions and orientations. It can be clearly seen that the distances between points vary considerably between different image regions which led to including distance information into the search data. In addition, the distances between one point and its predecessor in horizontal and vertical direction can be quite different due to perspective and distortion effects, so a distinction between horizontal and vertical search data is necessary. The same applies to angular search data.

2. The distance between the point itself and the horizontal predecessor point.

3. The angle between the horizontal image axis $x$ and the connection line between a vertical “predecessor point” – i.e. a point with a smaller $Y$ grid coordinate, but the same $X$ grid coordinate – and the point itself.

4. The distance between the point itself and the vertical predecessor point.

Please note the following conventions in this context:

1. The terms “horizontal” and “vertical” always refer to the grid coordinate system, not the image coordinate system: The $X$ axis of the grid coordinate system has “horizontal” orientation, the $Y$ grid axis is “vertical”. These terms by themselves only denote the orientation, so the terms “positive” and “negative” are additionally used to describe the search direction with “positive” meaning “in the direction of the grid axis”.

2. Each point only stores information on how it can be reached from a predecessor point, not from a successor point (which has a larger grid coordinate for the respective axis). This is to maintain consistency and simplicity. But the way search data is obtained makes sure that information about known successor points is also taken into account if appropriate (see figure 4.5).
Chapter 4. Calibration Data Extraction Software

The way the search data is obtained is best explained graphically with the help of an example: Please refer to figure 4.5 for details and note that the data is obtained using the first possible method in the list of the shown cases. They are sorted by quality in descending order, i.e. if it is possible to apply both case 1 and case 2a, case 1 is used as this will give better search data.

Figure 4.5: Obtaining search data. For the example images shown, the search data for a horizontal search in negative direction (green arrow in \( \text{[a]} \)) starting from \( p = (X, Y) \) (green point) is wanted. Points that are marked blue indicate other known points whose data is used in the respective sub-figure. The red straight arrows denote the search data (e.g. angle). If a red arrow is drawn on a point, the data is stored in that point. Then the orientation of the arrow drawn complies with the convention that each point stores information on how it can be reached from a predecessor, i.e. a point with a smaller grid coordinate in the respective orientation. Red curved arrows denote the rotation operations needed to convert the stored angle to the desired orientation and direction. If a red straight arrow is drawn between \( p \) and another point, then it denotes search data which is calculated using the positions of these two points. – The captions right below the images for each case contain descriptions specific to this particular example. Universal descriptions (suitable not only for this example, therefore generally more awkward to read) are listed in the following: \( \text{[b]} \) Case 1: Desired search data stored in \( p \) itself.

\footnote{Also note that distance and angle information are always returned using the same method and that only the availability of angle information is checked when deciding which method to use. If angular information can be obtained by one method, distance information can generally also be obtained by the same method. Only some special cases involving initial points of the search require additional more or less complicated handling in the code which will not be elaborated here for simplicity's sake.}
4.3 Mapping the Image Coordinates to Grid Coordinates

Figure 4.5: Obtaining search data (example; continued). Case 2a: Next point in search direction along search orientation has been found already. Case 2b: Next point opposite to search direction along search orientation has been found already. Case 3a: Information for search orientation stored in preceding and succeeding point along perpendicular orientation. Case 3b: Information for search orientation stored in succeeding point along perpendicular orientation. . .
Chapter 4. Calibration Data Extraction Software

Figure 4.5: Obtaining search data (example; continued). 

(g) Case 3c: Data for horizontal orientation stored in lower vertical neighbor.

(h) Case 4: Information for vertical orientation stored in p.

(i) Case 5a: Data for vertical orientation stored in lower and upper vertical neighbor.

(j) Case 5b: Data for vertical orientation stored in upper vertical neighbor.

Case 3c: Information for search orientation stored in preceding point along perpendicular orientation. Case 4: Information for orientation perpendicular to search orientation stored in p. Case 5a: Information for orientation perpendicular to search orientation stored in preceding and succeeding point along perpendicular orientation. Case 5b: Information for orientation perpendicular to search orientation stored in succeeding point along perpendicular orientation. . . .
4.3 Mapping the Image Coordinates to Grid Coordinates

Case 5a: Data for vertical orientation stored in lower vertical neighbor

Figure 4.5: Obtaining search data (example; continued).

Please also note the following for the cases shown in the figure:

- As stated before, each point only stores information on how it can be reached from a predecessor. The algorithm nonetheless makes sure that if a point is found, the appropriate point (the newly found point or the current, already known point from which the search was started) receives the updated data that can be gained from the relationship between the two points.

- A point can be found starting from several other points in several directions. E.g., the point (4,2) can be found from (2,2) in positive horizontal direction, from (4,4) in negative vertical direction and so on. This means that even if the information contained in the relationship between a newly found point and the search start point is stored in the respective point immediately, this does not guarantee that information about the relationship to other possibly known points in the other orientation is also stored in the respective point automatically: “The neighbor of a point may be discovered by someone else without the point realizing it.” Therefore case 2a and case 2b are included to check if more information since the last time a point was modified has become available.

- Case 2a is basically a ‘safety net’: If the point that one tries to find is already known, this case ensures that the exact search data is returned and thus helps to
prevent that another, different point could be found and considered to be the point that is already known (which could happen if imprecise search data were returned and the candidate points to consider in the neighborhood were very similar to each other, such as in off-center regions).

- A possible refinement would be to add cases analogue to case 2a and case 2b between case 4 and case 5a. Since search data values obtained by a 90° rotation were found to be not precise and not helpful and therefore are, if possible at all, not used in the search anyway (section 4.3.3), this was not done.

### 4.3.3 Measuring the ‘Fitness’ of Neighbor Points

Having obtained the angle and distance search data and calculated the angle and distance for all points in the local neighborhood of the current point, a decision needs to be made about which of the neighboring points (if any) is the true next point that one tries to find. As neither distance nor angle information alone were found to sufficiently answer this question, both the reference angle and distance must be compared to those values for the neighbor points.

Trying to find the point which suits the best by looking at distance and angle information means that there is more than one criterion to consider. A way to handle both criteria at once needs to be found as situations such as “point A is better with respect to angle similarity than point B, but point B is better with respect to distance similarity than point A” are by themselves not decidable. One way to resolve this issue is to try to aggregate the criteria into just one which involves weighting the one criterion against the other. This is the approach chosen in the development of the algorithm.

Several such aggregations of angle and distance information were tried, which included operations ranging from simple addition, subtraction, multiplication or division to trigonometric functions and both absolute (absolute distance, absolute angle) and relative operands (angle and distance differences between reference value and current value). However, no single aggregation was found to sufficiently handle all possible special situations that could occur.

Generally, the most promising fitness function (to be maximized) was:

$$f_{\alpha,d}(\alpha, \alpha_{ref}, d, d_{ref}) = - [\text{absAngleDiff}(\alpha, \alpha_{ref}) + w \cdot \text{abs}(d - d_{ref})]$$  \hspace{1cm} (4.1)

$\text{absAngleDiff}(\alpha, \alpha_{ref})$ is calculated with a special function which returns the absolute angle difference between the two angle arguments, but normalizes each argument to the range $-180^\circ \ldots +180^\circ$ first and calculates the smallest angle difference between both (e.g. $\text{absAngleDiff}(-170^\circ, +170^\circ) = 20^\circ$). The weight factor $w$ was set to 1 since angular deviations (in °) and distance deviations (in pixels) are of the same order of magnitude and were found to be about equal for ‘real’ neighboring points.
4.3 Mapping the Image Coordinates to Grid Coordinates

That particular point of the neighboring points with the highest fitness is then chosen and accepted to be the next found point along the current search orientation and direction – if it also has a fitness of at least \( \text{minFitness} \). This is one of the only two ‘real’ parameters that need to be specified by the user for the algorithm to work\footnote{The other one is \( \text{absAngleTol} \) which is needed for the second fitness function in use, see later.} and makes sure that a certain absolute similarity between points is required for a point to be accepted.

Obviously, this fitness function cannot be used in the beginning of the search when no reference distance \( d_{ref} \) is available since no other points are yet known and only the orientation of the coordinate system axes are used as the reference angles, \( \alpha_{ref} \).

In addition, it was found that the search data obtained by a 90° rotation, i.e. by using case 4 - case 5c, was generally very imprecise:

- Horizontal and vertical grid lines do not meet at right angles, so the angle obtained can be a very rough estimation;
- distance between points in horizontal and vertical orientation can also be very different, depending on perspective and distortion (figure 4.4).

It was found that taking especially the imprecise reference distance \( d_{ref} \) into account in these cases led to wrong detections. E.g., sometimes only every second point in the vertical orientation was matched in a vertical search when a horizontal reference distance was used because the distance between points in the horizontal was about twice as large as the distance between points in the vertical.

Ignoring the reference distance in these cases when the search data was obtained by a 90° flip was found to be better. Therefore, since it cannot be avoided totally to use such 90° flips in the search, a second more appropriate fitness function needed to be devised for that case which does not use the reference distance \( d_{ref} \) (and for the case when no reference distance is available):

\[
 f_\alpha(\alpha, \alpha_{ref}, d) = \begin{cases} 
 -d & \text{if } \text{absAngleDiff}(\alpha, \alpha_{ref}) \leq \text{absAngleTol} \\
 \text{undefined} & \text{else} 
\end{cases} 
\]  

(4.2)

This function will result in accepting the closest neighbor point whose angle is within a certain angle tolerance \( \text{absAngleTol} \) from the reference angle (again using the function mentioned above to compute the absolute angle difference). This fitness measure is therefore (and must be) much more tolerant toward ‘dissimilar’ neighbors to cope with the poor available search data (reference values).

Note that the angle tolerance must on the one hand be large enough to compensate for the fact that the reference angle is imprecise; on the other it should not be too large to avoid being too tolerant in the search and always accepting the closest other point in the neighborhood, regardless of its angle. A value of 17° proved to be a good value for most of the available test images.
Chapter 4. Calibration Data Extraction Software

Obviously the false detection risk is much higher for this second fitness function, so it should be used in as few cases as possible.

### 4.3.4 Strategy to Traverse the Grid

Therefore a good strategy is needed to traverse the grid in a way that allows to obtain the search data without using case 4-case 5c whenever possible; this will prevent the use of the fitness function \( f_\alpha \) and allow using \( f_\alpha,d \). Such a strategy is presented in figure 4.6.

As can be seen, the use of case 4-case 5c can be avoided in all but the first two steps (figures 4.6a, 4.6b) in the shown example to find all points in the entire region.

Since this is generally the case for almost all images (usually the retries to find the missing points, as seen in figure 4.6a, are successful and can discover the missing points based on other known points), a parameter was included in the software which allows the user to prohibit the use of the risky fitness function \( f_\alpha \) in all but the first two steps. The introduction of this option has proven to be a very successful step and has highly reduced the number of misdetections. It is therefore strongly suggested that the user set the parameter `allowTolerantFitness` to 0.

Only in very few cases setting `allowTolerantFitness` to 0 prevents the software from finding certain points which can only be reached by obtaining the search data by case 4 or worse. An example is shown in figure 4.7a on p. 31. While setting `allowTolerantFitness` to 1 in such a case may allow to find the missing points (figure 4.7b), it also increases the risk of mistakes in the search. Usually it is better to set `allowTolerantFitness` to 0 and possibly increase `maxRetries` so that more rounds of retries are gone through in which remaining points can be discovered step-by-step based on other known points (usually `maxRetries` = 2 is sufficient). The result will then possibly include about two data points less, but all data points are very likely to be correct. Having several wrong data points for calibration due to mismatched points would diminish the quality of the results strongly.

Please also note that the procedure shown in figure 4.6 is only the basic guiding strategy. If certain points that the strategy is based on are missing on the image, the search is still continued by other means. E.g. if \((3dX, -dY)\) for the step shown in figure 4.6b is missing, continuing from \((2dX, -dY)\) is attempted, and even if this point is not available, the search goes on. The only prerequisite to obtain a reasonable number of data points is that the point \((0, -dY)\) must be visible in the image and that the horizontal row of points through it should reach as far from the large bar as possible. It is not necessary that precisely the same region with the same points is visible in all images, points may be present in some images and missing in others (the overlap should nonetheless be as large as possible for good results).
4.3 Mapping the Image Coordinates to Grid Coordinates

Figure 4.6: Traversing the grid. A simple example with $dX = dY = 2$ mm and few grid points is shown. The centroid of the big bar defines the coordinates of the starting point of the search, $(0, 0)$, its orientation (shown by the $X$ axis) is stored for the horizontal angle. (a) The orientation of the small bar (shown by the $Y$ axis) is used as an initial guess for the search in vertical orientation in negative direction to find $(0, -dY)$. (b) The point $(dX, -dY)$ is found by searching horizontally in positive direction from $(0, 0)$ on. Since $(0, 0)$ stores a horizontal search angle, the search angle to find $(dX, -dY)$ horizontally is obtained according to case 3b (figure 4.5f) and no angle flip (case 4, figure 4.5h, or case 5b, figure 4.5j) is necessary. Note that in these first two steps the use of the fitness function $f_\alpha(\cdot)$ is necessary: The search data is obtained using $(0, 0)$ or the initial guess as the data source which only contain angle, but no distance information.

\[ a \]

Since each point only stores information on how it can be reached from a predecessor, $(0, 0)$ stores the orientation of the big bar as the horizontal angle (defined by the connection between the (fictitious) point $(-dX, 0)$ and $(0, 0)$), but uses the orientation of the small bar as an initial guess that is passed as the optional argument $\text{initSearchAngle}$ to the function $\text{findAllGridPoints}(\cdot)$. As soon as $(0, -dY)$, the vertical predecessor of $(0, 0)$, is found, the appropriate vertical angle is stored in $(0, 0)$.

\[ b \]

If the use of the fitness function $f_\alpha, d(\cdot)$ instead of $f_\alpha(\cdot)$ in these cases were desired, additional distance information would have to be provided. For the vertical orientation, this would have to be an estimation given as an initial guess (e.g. by calculating the average distance between all points, but this would probably be rather imprecise). For the horizontal orientation, either a similar distance estimation would have to be stored in $(0, 0)$ in addition to the horizontal angle to obtain complete angle and distance search data using case 3b, or neither angle nor distance would have to be stored so that the search data would be obtained using case 5b. All in all, these measures to force the use of $f_\alpha, d(\cdot)$ are not likely to improve the results.
Figure 4.6: Traversing the grid (continued). Note that the way the grid is traversed from here on means that in all remaining example images shown, the use of the fitness function $f_{\alpha,d}(\cdot)$ is possible because complete search data, angle and distance, can be obtained without having to apply an angle flip (cases 4-5c, figures 4.5h-4.5k). Since $(0, 0)$ is known, the search data to find $(0, -2dY)$ vertically from $(0, -dY)$ on can be obtained using case 2b (figure 4.5d). Based on the initial knowledge about the points $(0, -dY)$ and $(dX, -dY)$, case 2b can again always be used to successively complete a ("top") horizontal line from there on, in positive and negative direction. . . .
4.3 Mapping the Image Coordinates to Grid Coordinates

(f) From \((0, -2dY)\) on, complete “bottom” horizontal line in both directions

(g) Complete lower part of vertical line through \((0, -2dY)\) in negative direction

(h) Complete upper part of vertical line through \((3dX, -dY)\) in positive direction

Figure 4.6: Traversing the grid (continued).

(f) Complete the “bottom” horizontal line. (g)(h) Complete the vertical line in both directions. The search data to begin the vertical line is obtained with case 2b (two horizontal lines present). The lower part of the line runs through \((0, -2dY)\) near the origin to be close to the center. The upper part must be continued elsewhere as the black bars block the way near \(X = 0\). The line is not started at \((2dX, -dY)\) but at \((3dX, -dY)\), however, because only then there are always points to both sides of the vertical line \((2dX, 0)\) would not have a neighbor to both sides if the vertical line went through here). This may be better for the next step.

\[\text{a...}\]

-- It could be searched horizontally from each point in the vertical line next. Then case 3c could be used for all these points. If the line went through \(X = 2dX\), case 3c would not be applicable for \((2dX, 0)\), case 4 would have to be used then. – In the shown example, this effect does not play a role since the next step is different.

37
(i) Search perpendicular to longer line in positive direction

Figure 4.6: Traversing the grid (continued). Next determine whether the “top” horizontal line or the vertical line is longer, pick the longer one first and search perpendicular to it (to span as much room on the image as possible in this step). In this case, the horizontal line is longer than the vertical line, so search perpendicular to the “top” horizontal line in positive direction first. Note that the vertical search data can always be obtained by applying case 2b since there are the two horizontal lines: The relationship between two points with identical $X$ coordinates is known and the vertical search data can therefore be calculated from them. . . .
4.3 Mapping the Image Coordinates to Grid Coordinates

Figure 4.6: Traversing the grid (continued). Search perpendicular to the “bottom” horizontal line in negative direction, again applying case 2b to get the search data. . . .
(k) Search perpendicular to shorter line (upper and lower part, in both directions)

(l) Try to find missing points

Figure 4.6: Traversing the grid (continued). Having searched perpendicular to the longer line in both directions, do the same for the shorter line (in this case: the vertical line). Since in this example the previous step has already unveiled all points that could have been found in this step, no change is visible the image. In general, additional points at the edges may be discovered in this step. Finally try to find the missing points: From each found point that is close to points that have not been found yet, search in all directions (example step shown: search in all directions from (4, 2) on, which is close to some of the remaining points, and find (2, 2)). \ldots
4.3 Mapping the Image Coordinates to Grid Coordinates

Figure 4.6: Traversing the grid (continued). (m) Usually all missing points can be found after just one iteration. If necessary, repeat last step, (l) since new possible starting points may have become available from which other points could be discovered.

(a) allowTolerantFitness=0  
(b) allowTolerantFitness=1

Figure 4.7: Results for different settings of the parameter allowTolerantFitness. Usually all points can be found even if allowTolerantFitness is set to 0. (a) Only in some rare cases this prevents the software from finding certain points which can only be reached by obtaining the search data by case 4 or worse. (b) Setting allowTolerantFitness to 1 in such a case may allow to find the missing points as in this example, but it also increases the risk of mistakes (assigning incorrect grid coordinates to points) in the search.
5 Results

The presented setup and software have allowed to fully achieve all goals that were initially set for this work – with the exception of one, regarding the precision of the obtained calibration results, which will be discussed later (sections 5.3 [3]).

5.1 Mechanical Setup

All requirements concerning the mechanical setup (cf. section 1.4.1) could be met:

1. The setup allows to present the pattern printed on the plate to the camera in different positions. The plate positions cover enough space in 3D within the range of the endoscope for the calibration. The plate size was chosen appropriately: For all desired plate positions, the major part of the endoscope field of view covers the grid and not the background.

2. The plate with the pattern is exchangeable. This could be taken advantage of already at the time when the first test images were taken at Universitätsspital Zürich: The design of the shaft allowed to quickly change the plates with the three test patterns which was important as the time available to take the images was limited. The plate holding unit could furthermore carry the simple test plates, made by gluing paper print-outs of the patterns onto steel, during these experiments and will also be able to eventually hold the final calibration plate made of ceramics (due to the high costs, the production of this plate was postponed until the availability of test results).

3. The device can be used in hospitals, disinfection or even sterilization is possible. All chosen materials and parts can withstand the temperature and pressure present in an autoclave. As required for the cleaning or sterilization process, the machine can be disassembled so that all openings and blind regions can be reached by the disinfectant or the sterilization steam. The machine consists of very few parts so disassembly is especially easy.

4. It is possible to take calibration images with the plate and the endoscope immersed in water. The water container is large and open enough to freely position the endoscope in it. Although underwater test images have not been taken yet (not possible with paper test pattern), a test for leak tightness could be carried out
5.2 Software and Calibration Pattern

All qualitative requirements regarding the pattern and the software could fully be met as well (cf. section \[1.4.2\]). As far as the quantitative objectives are concerned, e.g. the precision with which the data points can be localized, please consult sections \[5.3\] and \[6\].

1. The chosen pattern design for the use with the mechanical setup provides the necessary information for the automated data extraction. Orientation and location of the grid coordinate system can be identified safely in all situations except for rare exceptions (figure \[4.2.3\], section \[4.2.3\]). The locations of the grid points can be determined as well.

2. The software extracts the needed data for the calibration algorithm and makes it available in the format required for the use with the toolbox \[1\].

successfully. Due to the design of the machine, only one opening needs to be watertight which minimizes the number of possible spots for leaks. This single sealing is also resistant to the conditions in an autoclave so that it does not need to be taken out for separate disinfection or be replaced after use.

5. image acquisition with the setup is easier and more precise than with the previously used method. By tilting the plate up to 45° and rotating it up to 360°, all desired plate positions can be reached easily and quickly without any parts blocking others (as before). The way the plate and the endoscope are fixed also allows the plate center to be positioned in different positions relative to the endoscope if necessary. Positioning plate and endoscope relative to each other in such a way that as much as possible of the calibration pattern is visible on the images can be done quickly and precisely; it can be avoided more easily than before that the inclined field of view misses larger parts of the plate. The endoscope is fixed well enough to avoid camera shakes. The plate can be illuminated with an external light source appropriately. The resulting endoscope images are of good quality. All in all, a calibration image sequence can be acquired in a shorter time and more easily than before; alternatively more images can be taken in the same amount of time.

6. The prototype design is kept simple and is well-suited for manual operation. Future motorization of the moving parts without fundamental design changes is possible.

7. The device is suitable for the endoscope used in the research group, but it is also possible to use other endoscopes with the same setup because the endoscope holding is not custom-made for a particular model but universal and simple. Should a more specialized endoscope holding unit become necessary, it can be integrated easily the current design.
3. The software requires very little user interaction\textsuperscript{1} It also requires practically no a-priori knowledge about the model parameters of the optical system.

In more detail, the advantages of the chosen mapping algorithm are:

- The algorithm is fully independent of camera or distortion models. The calibration data collected is not affected by the model assumptions about the distortion.

- It makes good use of available information, processes the images fast and efficiently and usually needs only one overall-iteration (i.e. $\text{maxRetries} = 1$) to systematically find all points in the images.

- The basic idea behind the algorithm is simple and easy to understand, yet it is effective\textsuperscript{2} (It does not rely on hard-to-follow recursive procedures or the like.)

- The algorithm requires practically no a-priori knowledge about image distortions and needs only two ‘real’ parameters to be set by the user, $\text{minFitness}$ and $\text{absAngleTol}$. Both of these are not abstract, hard-to-imagine quantities, so appropriate values can easily be estimated by the user. Moreover, test-running the algorithm on most of the available images suggests that the majority of them (which cover a broad range of different situations) can be handled with identical or very similar parameter settings. Using these as default values means that the user does not need to adjust anything at all in most cases.

- The software warns the user about possible error sources as they occur (such as when the settings make the algorithm too strict or too tolerant) to help judge the result quality and gives detailed hints for improving the results\textsuperscript{1}

- Most important: After a long period of development, the numbers of errors (the number of incorrectly assigned grid coordinates per image; the risk of false detections) could be reduced strongly to practically 0. Methods such as refining the way the grid is traversed and introducing the switch $\text{allowTolerantFitness}$ to prevent the use of the more tolerant fitness function $f_\alpha$ except if definitely necessary in the first two steps (section \textsuperscript{1.3.4}). This is crucial to rely on for automated processing and necessary for good results.

\textsuperscript{1}If desired. Note that several modes of operation are possible which allow almost anything from silent, fully-automatic processing to a step-by-step advancement with detailed text and graphics output, see section \textsuperscript{A.2} for details.

\textsuperscript{2}The source code files are not as short as one might expect at first, however. This is mostly because many more cases need to be handled than one generally thinks of in the beginning, because the program can be run in several modes ranging from “fully automatic” to “debugging mode with detailed output” (see section \textsuperscript{A.2}) and because extensive comments and help texts are included in the code to ease the use for future users or developers.
• The algorithm can handle many difficult situations and is applicable to a wide range of problems (images). Some examples are shown in figure 5.1.

• If images are really ill-conditioned (e.g. when the bars identifying the coordinate system are not completely visible, as in figure 4.3c) or if certain points expected to be found cannot be discovered, the software does not crash. If appropriate and possible, continuing the search is attempted. Images with results not satisfying the user can selectively be disabled, without affecting the results for the other images.

• All in all, in practice most images should (and seem to) be processable truly automatically without user interaction after a one-time setup. The time needed for data extraction and calibration is thus reduced from more than ten minutes per calibration sequence to a few seconds.

5.3 Calibration

To demonstrate and analyze the quantitative results that can be obtained with the new setup and the new software and to show the potential for further improvements, two sample calibration runs are presented in the following: One run for the old mechanical setup, the old calibration pattern and the old software (figure 5.2a) and one run for the new mechanical setup, the new calibration pattern and the new software (figure 5.2b).

Note that the numerical results presented for both runs are not to be considered true (statistical) averages for a wide range of images or situations (such averages would be hard to obtain as there are countless possibilities to consider), but are merely supposed to be more or less typical examples. Also note that it is not very easy to compare the results from the two methods as they are based on different circumstances: Even though eight images were used for both cases, the new software led to much more data points being included in the calibration. These data points are from a larger image region and also lie in very off-center regions where distortion is very strong (see also figure 5.3a).

As can be seen from figure 5.2, the pixel errors for the completely manual calibration are considerably smaller than those for the automatic mode with the current settings. While the time gain is enormous, the pixel error is not yet small enough. Possible reasons for these outcomes and suggestions to improve them are given in section 6.

3Numerical results for both of the cases “images acquired with new setup, data extraction with old software” and “images acquired with old setup, data extraction with old software” are not shown here for the following reason: The main advantage of the new mechanical setup is that images can be taken much more easily and faster than before in good quality; this advantage has become clear and can be measured in terms of the time gain, but not well not in terms of the calibration results. This is because the calibration results very much depend on the ‘performance’ of the user, i.e. how precisely the image locations are clicked and selected, and which points on the pattern are chosen. Trying to shown large differences between the results for these two cases would be a little arbitrary; the differences that arise from how precisely the manual data selecting is done (how accurately the user clicks) are larger.
Figure 5.1: Examples of detection results. On purpose, difficult circumstances were chosen for these pictures to demonstrate the operation of the software: The plate is tilted by the largest possible angle, 45°, so perspective and distortion make the conditions (angle, distance in horizontal and vertical orientation) very different from image to image in the sequence and from one region to another within the same image; . . .
Figure 5.1: Examples of detection results (continued). . . the images are from a sequence with rather poor image quality compared to other sequences; and a small value for minArea led to many points to be detected. In practice, different settings would be used. The obtained detection results are very good: The points are mapped to correct grid coordinates, even at the ‘curved’ boundaries of the point clouds (more difficult to handle because of less neighbors) as in (c) and (d). . . .
Figure 5.1: Examples of detection results (continued). Images with very different conditions (large differences from region to region or orientation to orientation) can also be handled well as can be seen in (c), (d), (e), (f). Outlier points that arise from problems with binarization such as those in the bottom right corners of (c) and (e) are not accepted as points (no false detections).
5.3 Calibration

Calibration results (with uncertainties):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal Length</td>
<td>[389.33385 404.78074]</td>
<td>+/- [3.98477 4.43296]</td>
</tr>
<tr>
<td>Principal point</td>
<td>[329.89359 399.91877]</td>
<td>+/- [5.91746 6.11466]</td>
</tr>
<tr>
<td>Skew</td>
<td>alpha_c = 0.00000</td>
<td>+/- [0.00000]</td>
</tr>
<tr>
<td></td>
<td>=&gt; angle of pixel axes = 90.00000 +/- 0.00000 degrees</td>
<td></td>
</tr>
<tr>
<td>Distortion</td>
<td>kc = [-0.49765 0.19359 -0.00909 0.00136 0.00000]</td>
<td>+/- [0.01789 0.01904 0.00379 0.00255 0.00000]</td>
</tr>
<tr>
<td>Pixel error</td>
<td>err = [0.68533 0.64858]</td>
<td></td>
</tr>
</tbody>
</table>

Note: The numerical errors are approximately three times the standard deviations (for reference).

(a) Old setup, old pattern, old software

Calibration results (with uncertainties):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Uncertainty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focal Length</td>
<td>[381.73702 395.02785]</td>
<td>+/- [1.87263 1.87464]</td>
</tr>
<tr>
<td>Principal point</td>
<td>[226.79004 334.68252]</td>
<td>+/- [2.92482 3.21406]</td>
</tr>
<tr>
<td>Skew</td>
<td>alpha_c = 0.00000</td>
<td>+/- [0.00000]</td>
</tr>
<tr>
<td></td>
<td>=&gt; angle of pixel axes = 90.00000 +/- 0.00000 degrees</td>
<td></td>
</tr>
<tr>
<td>Distortion</td>
<td>kc = [-0.39390 0.09734 0.00198 -0.00259 0.00000]</td>
<td>+/- [0.00710 0.00434 0.00181 0.00130 0.00000]</td>
</tr>
<tr>
<td>Pixel error</td>
<td>err = [1.52314 1.51312]</td>
<td></td>
</tr>
</tbody>
</table>

Note: The numerical errors are approximately three times the standard deviations (for reference).

(b) New setup, new pattern, new software

Figure 5.2: Sample calibration results for old and new setup and software. Both results are for a sequence of eight images. Note that the actual parameters obtained are not equal because the two sequences were taken with different endoscope settings; most interesting here is the comparison of the pixel error. [a] With the old settings, i.e. manual data extraction from images acquired using the old mechanical setup, pixel errors of about 0.7 or even below can be achieved, depending on the image quality, how precisely the user selects the chessboard corners and which and how many of them are used. [b] The pixel error of about 1.5 for the new settings, i.e. fully automatic data extraction from images acquired with the developed mechanical setup, is rather large compared to that obtained with the old settings. Note, however, that the data extraction and calibration only took a few seconds in this case as opposed to more than ten minutes.
Chapter 5. Results

(a) Reprojection

Figure 5.3: Reprojection example. Having computed the distortion model parameters, the positions of the data points as found on the image on the one hand and as computed with the obtained distortion model parameters (reprojection) on the other hand can be compared. The arrows show the deviations between both. Especially for the big bar the deviation is large because its centroid on the image is not equal to the true position of the grid point \((0, 0)\). This can be seen on the reprojection error plot: While the error is about evenly distributed for most points (one mark for every data point), there are a few outliers such as the big bar.
6 Conclusion and Outlook

All initially set goals except for one could be achieved: The numerical calibration results using the automatically extracted data are not yet fully satisfying. Analyzing this effect is necessary and constitutes the main potential for future improvement. The following are suspected to be chiefly responsible for the calibration outcome as encountered at the moment:

- The locations of the grid points that are used as the calibration data are possibly not precise enough. The preprocessing may have altered the locations of the centroids so much that the results are not as good as in the manual case. Therefore, improving the preprocessing or additional refinement for the locations of the centroids should be tried. Possibly the location of the big bar – the starting point of the search with the grid coordinates \((0,0)\) – which is currently also used as a data point should not be considered since it cannot be determined as precisely as the locations of the other grid points.

- Compared to the manual extraction case, much more data points are used when the new software is utilized. These data points are also much more spread over the off-center regions in the images. Firstly, the locations of the data points cannot be determined as precisely in these regions as in the center: The points here are usually rather small which means that the pixel size is large relative to the point size, and since operations such as binarization may change several pixels, the effect on the centroid can be large. Secondly, the true lens distortion is non-linear and is maybe not described well enough with the used distortion model [2] in the off-center regions anymore (for the center region, the model is appropriate). It could therefore be tried out to exclude points that are very far from the image center (or the origin of the grid), e.g. by discarding found points that have extreme image (or grid) coordinates. However, while excluding data points from the calibration may give a smaller pixel error on paper, this does not necessarily mean that the results are truly better. By omitting points one runs the risk of using too few data which do not describe the distortion faithfully enough and allow to fit the model parameters to values other than the ‘true’ ones yielding a smaller pixel error.

- With the new setup and the new software it is possible that points are included in the calibration that are not visible on every image; data for different images may contain different sets of grid points. With the old software, on the other hand,
the same grid points are used for every image. Better results could be given by
the algorithm [2] if only points common to all images were included. The data
generated by the presented software can be modified in a very easy way to ensure
that this is done: All that needs to be done is to compare the found points for
all images and then take the largest common set. Only this is then fed into the
calibration algorithm.

Additional tests could also be done to find out if better results are obtained for more
or less different combinations of plate tilt angle and plate rotation angle: “Is it better to
use eight images all taken at a 35° plate tilt angle (for various rotational positions), or
is it better to use eight images four of which are for a plate tilt angle of 35° and the
other four for a plate tilt angle of 30°?”

Note that all suggestions for refinements or further analysis listed above do not concern
the actual mapping algorithm. Only the preprocessing of the images and the filtering of
the obtained data which were both not the main focus of this work are affected.
Bibliography

http://www.vision.caltech.edu/bouguetj/calib_doc/index.html

A MATLAB Source Code

A.1 Overview

The complete source code can be found in the accompanying CD-ROM and is not printed out here. An overview of the functions and scripts is given in the following table. For more detailed descriptions and a complete list of function arguments and return values, refer to the help documentation of the files, e.g. type `help findGridPointLine` in MATLAB.

Table A.1: Overview of the source code files

<table>
<thead>
<tr>
<th>MATLAB function/script</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridDetector(·)</td>
<td>Main function. Preprocesses image (section 4.2) and calls <code>findAllGridPoints(·)</code>.</td>
</tr>
<tr>
<td>findAllGridPoints(·)</td>
<td>Implementation of grid traversal strategy (section 4.3.4). Calls <code>findGridPointLine(·)</code> repeatedly with appropriate arguments. If use of <code>fitnessMinDist(·)</code> ($f_{alpha}$) is generally disabled (allowTolerantFitness=0) and search cannot start, temporarily enables it for first two steps.</td>
</tr>
<tr>
<td>findGridPointLine(·)</td>
<td>Core function to find a single point or a complete line of points in the grid in a given orientation and direction from a known starting point on (e.g. in figure 4.6d). Calls <code>gridSearchData(·)</code> for the start point to obtain search data, handles special cases such as initial conditions, calculates fitness by calling <code>fitnessAngleDistDiff(·)</code> or <code>fitnessMinDist(·)</code> (whichever appropriate and allowed), decides whether or not to accept a point, updates list of found and remaining points, warns user about possible error sources (section A.2).</td>
</tr>
<tr>
<td>gridSearchData(·)</td>
<td>Core function that tries to calculate/estimate the angle and distance search data for a grid point for a given search orientation and direction: Implementation of case 1-case 5c (figure 4.5).</td>
</tr>
</tbody>
</table>
A.2 Operation Modes

gridDetectorToolboxWrapper

Wrapper script to integrate the software into the calibration toolbox [1]. Asks user for parameters and operation mode (section A.2), stores them in workspace, calls gridDetector() for each image passing the parameters, asks whether or not to accept the results for each image.

gridDetectorIxConstants

Defines common index constants for functions and scripts. The help text contains basic definitions and explanations about the data structures used.

fitnessAngleDistDiff()

Implementation of $f_{\alpha,d}$ (section 4.3.3, eq. 4.1).

fitnessMinDist()

Implementation of $f_{\alpha}$ (section 4.3.3, eq. 4.2).

absAngleDiff()

Calculates smallest absolute angle difference after truncating arguments to the range $-180^\circ \ldots 180^\circ$ (cf. example in section 4.3.3).

angleDistAvg()

Calculates averages of two distances and angles using absAngleDiff(), ignoring undefined/non-existing arguments. Used in gridSearchData() so case 3a-case 3c and case 5a-case 5c can be handled by the same code.

noDuplictesInGrid()

‘Sanity check’ for final results: Ensures that no dataset is used in which a grid coordinate pair $(X,Y)$ is assigned to more than one point.

A.2 Operation Modes

To control the output, settings for the parameter verbosity (table A.2) and for the display or suppression of several warning messages (table A.3) can modified independently.

Table A.2: Settings for verbosity to control program output

<table>
<thead>
<tr>
<th>verbosity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No output (silent mode)</td>
</tr>
<tr>
<td>1</td>
<td>Detailed text output is sent to stdout.</td>
</tr>
<tr>
<td>2</td>
<td>Results are shown graphically in a MATLAB figure.</td>
</tr>
<tr>
<td>3</td>
<td>Both text and graphical output is produced.</td>
</tr>
</tbody>
</table>
Chapter A. MATLAB Source Code

-1, -2, -3  Output as for 1, 2, 3, but an additional pause to be ended by a keystroke is included after each step. In addition, the extent of the local neighborhood (scope) is visualized in the graphical output. These options are primarily intended for debugging as they allow to process an image with a step-by-step confirmation.

Table A.3: Warning message identifiers and corresponding messages. Sample messages which may appear in different wording, depending on the context, are marked with †.

<table>
<thead>
<tr>
<th>Warning identifier</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridDetector:</td>
<td>Coordinate system (location, orientation) may be wrong (big &amp; small bar too far away or angle between coordinate axes not within tolerance absAngleTol). (If the location and orientation of the coordinate system are correct [see image, visible for verbosity==2,-2], absAngleTol may be too small since the coordinate axes are outside of this tolerance.)</td>
</tr>
<tr>
<td>coordSystPlausibility</td>
<td></td>
</tr>
<tr>
<td>gridDetector:</td>
<td>Less than 10 points were detected. Maybe skip this image or retry with different parameter settings.</td>
</tr>
<tr>
<td>fewPointsFound</td>
<td></td>
</tr>
<tr>
<td>gridDetector:</td>
<td>Aborting search prematurely, point (0, −dY) not found.</td>
</tr>
<tr>
<td>findAllGridPoints:</td>
<td></td>
</tr>
<tr>
<td>prematureSearchEnd</td>
<td></td>
</tr>
<tr>
<td>gridDetector:</td>
<td>Attempted to continue search from point (+6.00, −2.00), but it is not in the list of found points.†</td>
</tr>
<tr>
<td>findAllGridPoints:</td>
<td></td>
</tr>
<tr>
<td>neededPointNotFound</td>
<td></td>
</tr>
<tr>
<td>gridDetector:</td>
<td>Using (2dX, −dY) instead of (3dX, −dY) to continue vertically (worse search data quality).</td>
</tr>
<tr>
<td>findAllGridPoints:</td>
<td></td>
</tr>
<tr>
<td>badStartingPoint</td>
<td></td>
</tr>
<tr>
<td>gridDetector:</td>
<td>Could calculate/estimate search angle (+80.23 deg) for grid point (+0.00, +0.00) WITHOUT 90 deg rotation, but initial guess also specified (+83.00 deg) - using guess.†</td>
</tr>
<tr>
<td>findGridPointLine:</td>
<td></td>
</tr>
<tr>
<td>ignoringAngle</td>
<td></td>
</tr>
<tr>
<td>gridDetector:</td>
<td>Could calculate/estimate search distance (34.76 pix) for grid point (+0.00, +0.00) WITHOUT 90 deg rotation, but initial guess also specified (50.00 pix) - using guess.†</td>
</tr>
<tr>
<td>findGridPointLine:</td>
<td></td>
</tr>
<tr>
<td>ignoringDist</td>
<td></td>
</tr>
</tbody>
</table>
### A.2 Operation Modes

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridDetector: findGridPointLine: noRefDistFitness</td>
<td>Applying fitness function that does not use reference distance. (Message may appear once or twice per image [depending on which information is stored in the search start point and if an initial distance guess was supplied] even if <code>allowTolerantFitness</code> is 0 [else search impossible]. If it appears too often, carefully check detection results on image (visible for <code>verbosity=-2</code>) for mistakes.)</td>
<td></td>
</tr>
<tr>
<td>gridDetector: findGridPointLine: fitnessCalcImpossible</td>
<td>Fitness calculation impossible, stopping (no reference distance available/angle rotated and use of fitness function that does not need reference distance prohibited [<code>allowTolerantFitness==0</code>]). If this message appears often, it merely indicates that detection for this image is difficult. Only if not enough points are detected in the end, make the search more tolerant by decreasing <code>minFitness</code> or increasing <code>absAngleTol</code> (or, as last resort, setting <code>allowTolerantFitness</code> to 1), but carefully check the detection results on the image (set <code>verbosity</code> to 2 or -2) for mistakes. Otherwise (generally) it is safe to ignore this warning.</td>
<td></td>
</tr>
<tr>
<td>gridDetector: findGridPointLine: tryingToAddDuplicate</td>
<td>A point was detected to be grid point (+18.00, -20.00) (horizontal search in positive direction, using fitness function that does not take reference distance into account), but this point is already in the list of found points - keeping previously found point, ending this search. If this messages appears too often, the algorithm is too tolerant and may produce false results, then try using <code>allowTolerantFitness==0</code>, increasing <code>minFitness</code> or decreasing <code>absAngleTol</code>.†</td>
<td></td>
</tr>
</tbody>
</table>

The display of each warning message can be enabled or suppressed individually (by issuing a command such as `warning off gridDetector:findGridPointLine:noRefDist`). The program also allows to turn on or off all warnings at the same time or to enable only the most important ones.