Conceptual Development of an Opto-electronic Perpendicular Utilizing Electro-active Glass

Christoph Gugg

Master Thesis of Christoph Gugg

supervised by

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Paul O'Leary

and

Dipl.-Ing. Richard Neumayr

Chair of Automation
University of Leoben
Austria

Eidesstattliche Erklärung

Ich erkläre hiermit eidesstattlich, dass ich diese Arbeit selbstsändig verfasst, andere als die angegebenen Quellen nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient habe.

Leoben, am 1. Dezember 2011 Christoph Gugg, B.Sc.

Danksagung

Hiermit möchte ich einer Reihe von Menschen danken, ohne deren Hilfe meine Master Thesis in dieser Art und Weise nicht möglich gewesen wäre. Der größte Dank gebührt meinen Eltern, Judith und Simon. Sie waren jederzeit für mich da und standen mir immer mit ihrer Erfahrung und Unterstützung zur Seite. Durch ihr Vertrauen in mich und meine Fähigkeiten war mir eine sorgenfreie Ausbildung in diesem Ausmaß erst möglich.

Ein herzliches Dankeschön möchte ich Doris, Paul, Richard und den anderen Mitarbeitern des Lehrstuhls für Automation aussprechen, die mich während des Verfassens dieser Thesis sowohl fachlich als auch organisatorisch betreut haben. Weiters möchte ich mich bei der Firma Geodata GmbH für die kooperative Zusammenarbeit, das zur Verfügung gestellte Material sowie für die mir entgegengebrachte Wertschätzung bedanken.

Selbstverständlich möchte ich noch meinen Freunden und Kollegen für die schöne Studienzeit in Leoben danken, die mein Leben wohl dauerhaft geprägt hat. Das erfolgreiche Masterstudium und diese Thesis wurden vor allem durch Christina ermöglicht, die ich nicht nur für ihre fachlichen, sondern auch für ihre menschlichen Qualitäten schätze.

Abstract

This master thesis investigates the conceptual development of an opto-electronic perpendicular utilizing electro-active glass. An exactly aligned laser beam serves as reference axis to enable the measurement of relative movements, caused by geological activities, within underground structures. This application is designed as a monitoring system, which automatically triggers maintenance tasks.

The paper's scope covers the documentation as well as the implementation of several image processing techniques for camera aided position determination. The primary process is composed of a logical sequence of four sub-procedures: at first, during *image preprocessing*, the contrast is normalized and the noise is suppressed through digital filtering; the *segmentation of image objects* is accomplished by binary morphology, employed ideas to calculate a suitable threshold are Otsu's method, brightness class average, Gaussian curve intersection and maximal normal distance; the *determination of an object's center* is performed through the computation of its center of gravity, its Gaussian curve extremum or fitting of a conic, to be specific the approximation of a circle and an ellipse; the last step is *coordinate mapping* through linear homography, polynomial control points or the newly introduced tensor interpolation via basis function sectioning and quad tree decomposition.

The different approaches are evaluated through experiments with a laboratory prototype. Consequently, verification and analysis is carried out on the numerical results to give a statement about the suitability of the different methods for such a kind of measurement instrument.

Index Terms

laser; perpendicular; electro-active glass; image processing; contrast normalization; noise suppression; segmentation; binarization; threshold; coordinate mapping; homography; polynomial control points; tensor interpolation; quad tree decomposition

Kurzfassung

Diese Diplomarbeit beschreibt die konzeptionelle Entwicklung eines opto-elektronischen Lots unter Verwendung von elektro-aktivem Glas. Ein exakt ausgerichteter Laserstrahl fungiert als Referenzachse um Relativbewegungen, ausgelöst durch geologische Aktivitäten, in Tiefbauanlagen zu registrieren. Die Anwendung stellt ein Überwachungssystem dar, mit dessen Hilfe Wartungsaufgaben automatisch angestoßen werden.

Die Arbeit umfasst sowohl die Dokumentation als auch die Implementierung verschiedener Bildverarbeitungstechniken zur kameragestützten Positionsbestimmung. Der Hauptprozess besteht aus folgenden vier Unterprozeduren: zuerst wird während der *Bildvorverarbeitung* der Kontrast normalisiert und das Rauschen durch digitales Filtern gedämpft; die *Segmentierung von Bildobjekten* wird durch Binarisierung erreicht, umgesetzte Ideen zur Berechnung eines geeigneten Schwellwerts sind Otsu's Methode, Helligkeitsklassendurchschnitt, Gauß Kurvenschnittpunkt und maximaler Normalabstand; die *Mittelpunktbestimmung der Objekte* erfolgt durch Schwerpunktberechnung, Gauß Kurvenextremum oder Kegelschnittanpassung, im Konkreten die Approximation durch einen Kreis und eine Ellipse; der letzte Schritt ist *Koordinatenabbildung* mittels linearer Homographie, polynomialen Kontrollpunkten und die in diesem Zusammenhang neue Tensorinterpolation durch Auswahl von Basisfunktionsabschnitten und hierarchischer Viertelzerlegung.

Die unterschiedlichen Ansätze werden durch Experimente mit einem Laborprototyp evaluiert sowie verifiziert. Die Analyse der numerischen Ergebnisse liefert eine Aussage über die Anwendbarkeit der untersuchten Methoden für diese Problemstellung.

Schlagwörter

Laser; Lot; elektro-aktives Glas; Bildverarbeitung; Kontrastnormalisierung; Rauschunterdrückung; Segmentierung; Binarisierung; Schwellwert; Koordinatenabbildung; Homographie; Polynomiale Kontrollpunkte; Tensorinterpolation; hierarchische Viertelzerlegung

Contents

Chapter 1

Introduction

This master thesis investigates several methods to determine a *laser*[1] spot's position on a target plate via image processing. The discussed concepts allow it to assemble an *opto-electronic perpendicular* for surveillance purposes. An exactly aligned laser beam and a set of measurement stations, whereby each one is composed of a target plate and a camera, deliver the possibility to detect relative movements of the environment, caused by geological activities, within a tunnel, a mine shaft or any other underground structure. The aim of the work is the conceptual development of such a device.

1.1 Logistic Relevance

It is in the nature of any infrastructure, that it deteriorates with time. Maintenance logistics addresses this issue by observing and tracking those objects and their changes to trigger corresponding support processes. The goal is to create systems, which last longer and require less service, thereby their overall lifetime costs are reduced and their safety is increased. The impact is measured in terms of metrics such as reliability, availability and maintainability. The robust and automated tracking of information over the entire operational life cycle provides the opportunity to optimize maintenance strategies. These ideas and concepts are normally applied for machines and their spare parts, however in the application at hand they are used for whole facilities [14].

[1]The acronym *laser* stands for *light amplification by stimulated emission of radiation.*

According to DIN 31051 [6], maintenance[2] is defined as: '*The entire effort to achieve and conserve the desired condition. Additionally, the current state must be determined and evaluated.*' The related DIN 13306 [5] specifies the primary processes, which are inspection, service, reparation and improvement. An adequate inspection strategy is vital for condition monitoring and error diagnosis. Failure prevention is achieved by enhancing the related technical and administrative processes continuously. From an economic point of view, the costs for the observation system must be smaller than the total costs of a system breakdown [18]. Generally, the maintenance processes are scheduled according to a specific plan. The scope ranges from periodical intervals to just-in-time and risk-based approaches. Therefore, the downtimes can be organized well in contrast to unexpected, time-critical errors and the unproductive time they induce to the system. Another important aspect is safety at work, because only a well maintained system is a secure one [10].

According to literature, maintenance logistics is responsible for supplying resources such as manpower, material and information. Additionally, it synchronizes these factors with the operational facilities. The last aspect is known as the coordinating and administrating process and is normally covered by a software package, which is part of the maintenance planning and controlling system. The introduced surveillance concept could be integrated via standard interfaces to automate the complete process chain. By simplifying the management of the support processes, more effort can be focused on the important value adding processes [1].

1.2 Problem Statement

The thesis presents the conceptual development of an opto-electronic perpendicular utilizing electro-active glass. The schematics in Figure 1.1, 1.2 and 1.3 show the desired functionality of the instrument. There's an exactly calibrated laser, which radiates a beam along the reference path. The laser beam impinges upon several measurement stations along its path, whereby a portion of the light is scattered and a visible spot is produced. Each station consists of a target plate and a camera. It is important, that these two components are mounted relatively and unchangeably to each other to avoid displacements. The position of the laser spot on the target is the relevant information. By logging its temporal position on the plate, the relative movements of the environment can be determined. For that purpose, several image processing methods are introduced in later chapters.

[2]The German terms are *Instandhaltung: Inspektion, Wartung, Instandsetzung* and *Verbesserung*

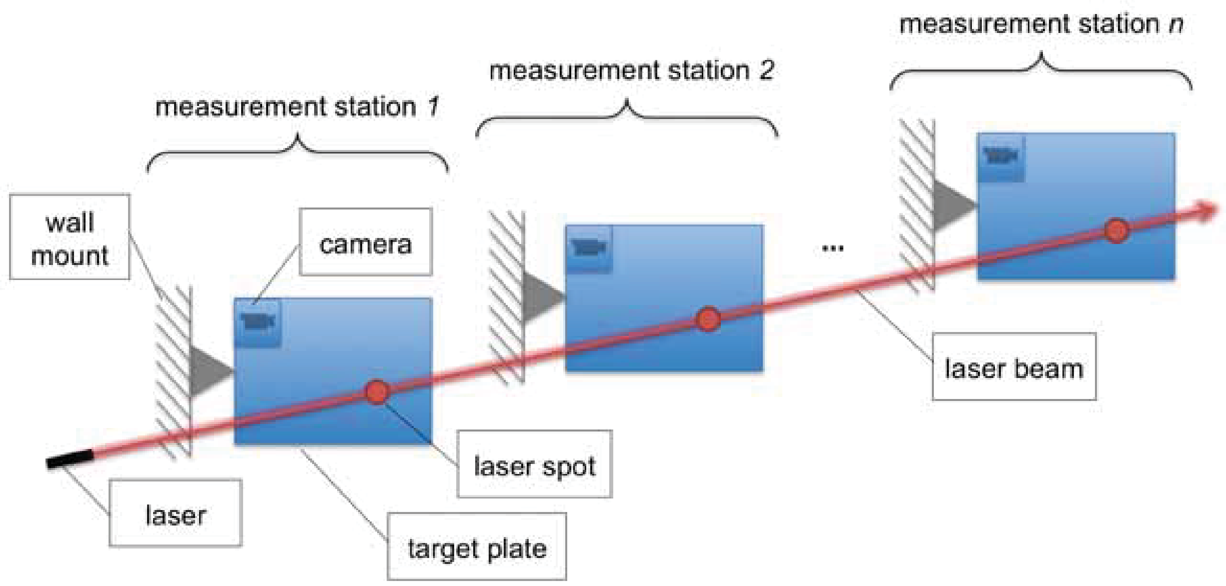


Figure 1.1: The laser radiates a laser beam along an underground structure, such as a tunnel or mine shaft, to observe the changes of the environment. The laser spot occurs when the beam meets a measurement station and impinges upon the attached target plate. The station is installed to the structure via a wall mount. The camera registers the laser spot and delivers the acquired image to a connected computer for further processing.

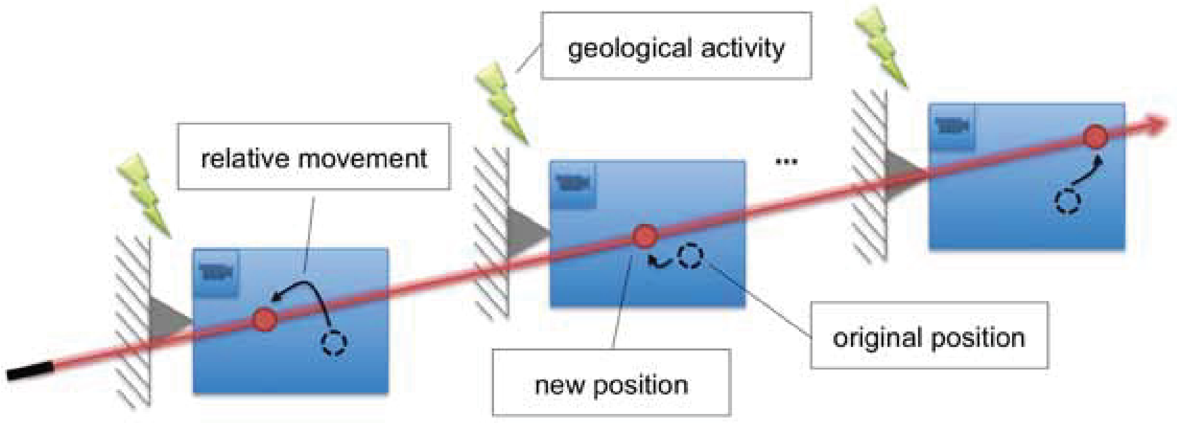


Figure 1.2: Environmental changes are caused by geological activities. The measurement stations are attached to the observed structure. Hence, they also change their position if the mountain is moving. The relative change of the position can be registered, as the laser's original orientation stays the same. The difference between the laser spot's new and original position is detected and evaluated in further process steps.

Electro-active glass is investigated as a possible means of improving the optical behavior of the target. This composite material can switch its opalescence from opaque to transparent by applying a voltage to a set of electrical contacts. The aim is to enable a larger number of targets in series.

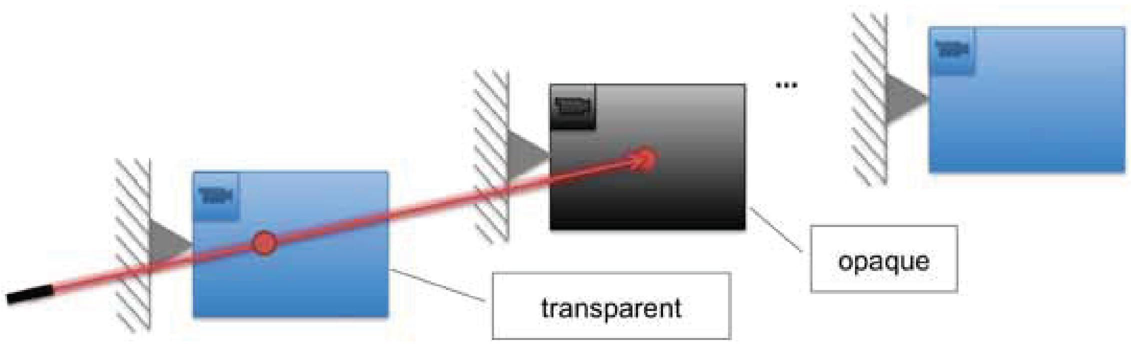


Figure 1.3: Underground structures are likely to have extensive dimensions, e.g. several kilometer long tunnels. The observation of sequentially arranged measurement stations requires transparent target plates. Nevertheless, the transparency does not deliver the best surface for laser measurements. To address this issue, electro-active glass is introduced to the application. It can change its opalescence as the needs arises: it's opaque during the measurement and transparent when the laser beam should pass through it.

Geological movements are effecting the statics of a structure. A perpendicular can act as reference axis. The displacement relative to the axis defines the changes in the system. Based on this data, operational decisions can be made and support processes can be triggered automatically when a critical value is reached.

1.3 Approach for Solution

The idea to use a calibrated laser as a perpendicular is not new, but in combination with electro-active glass a wide range of applications becomes possible. In fact, it is possible to measure long distances at any given angle, this is almost ideal for monitoring common underground structures or even bridges and dams. Of course, there exist other solutions such as the *telependulum* from *Huggenberger*[3]. The device quantifies automatically, contactlessly and continuously coordinates of a wire, which is attached to an anchor point at the top and a great mass at the bottom. Light sources cast a shadow of the plumb wire, the effect is detected by bright-dark sensors. The exact position of the wire is determined from the location of the light sources and the shadow focal points. This idea works very well and delivers an accuracy of $\pm 0.05mm$ to $\pm 0.10mm$, but only in vertical dimension and for this reason the system is very limited.

[3]©Huggenberger AG, Horgen, Switzerland, www.huggenberger.com

A laser offers the greatest flexibility for the discussed problem. It enables many possible setups in contrast to conventional devices such as the described telependulum. Of course, this advantage comes with the price of higher computational complexity, i.e. a sequence of operations must be applied to an image to extract the wanted information. The following steps must be performed to the dataset:

1. The image containing the current position of the laser spot is taken by a digital camera. The raw and unprocessed image is the initial data for all further computations;

2. The image is prepared by improving its contrast and applying noise suppression;

3. A threshold determines the borders of image objects and enables their segmentation from the background. The surrounding area is also relevant for the calculations and it must be considered by adding a confidence range;

4. The image objects are classified to acquire the regions of interest and to reject unwanted optical information;

5. Each region of interest is defined by its center. This information is the dominating factor for all following computations, as its accuracy defines the quality of the results;

6. The camera-target system must be calibrated, so that the image coordinates can be mapped to real world coordinates. Thus, suitable mathematical models are required;

7. After the calibration is performed, it is possible to convert any given point from its camera coordinates to its real world coordinates.

The *real space* is modeled as a Cartesian coordinate system with Euclidean distances, see Figure 2.12. The *camera space* is biased, because of optical effects of the camera's objective. Both spaces are related via a projection. The ultimate goal of the explained procedure is the possibility to map a point $\boldsymbol{p} = [x_c, y_c]^T$ from the camera space to its corresponding real space equivalent $\boldsymbol{q} = [x_r, y_r]^T$:

$$\boldsymbol{p} \xrightarrow{\text{mapping}} \boldsymbol{q}. \tag{1.1}$$

The complete process together with the used hard- and software will be described in Chapter 2. Figure 1.4 shows the camera space and Figure 1.5 shows its real world coordinates. Like in Equation 1.1, the mapping process is also true for the deployed units:

$$[pixel] \xrightarrow{\text{mapping}} [mm]. \tag{1.2}$$

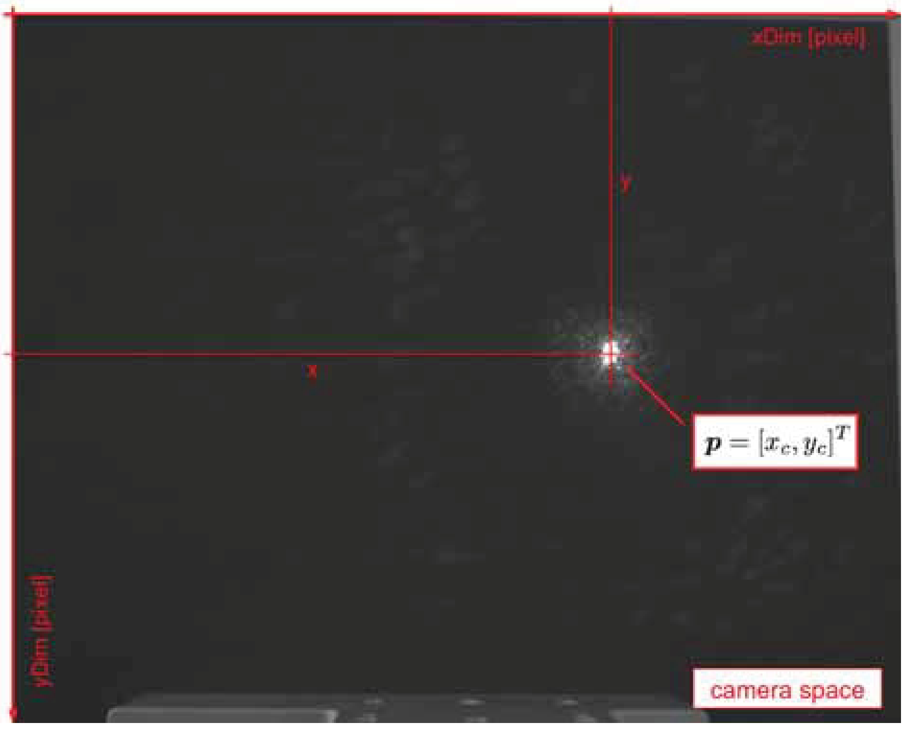


Figure 1.4: The camera space is biased because of optical effects. A mathematical relation is established through coordinate mapping methods, which are able to transform the coordinates of $\boldsymbol{p}$ to its real world pendant $\boldsymbol{q}$. Distances are measured in pixels.

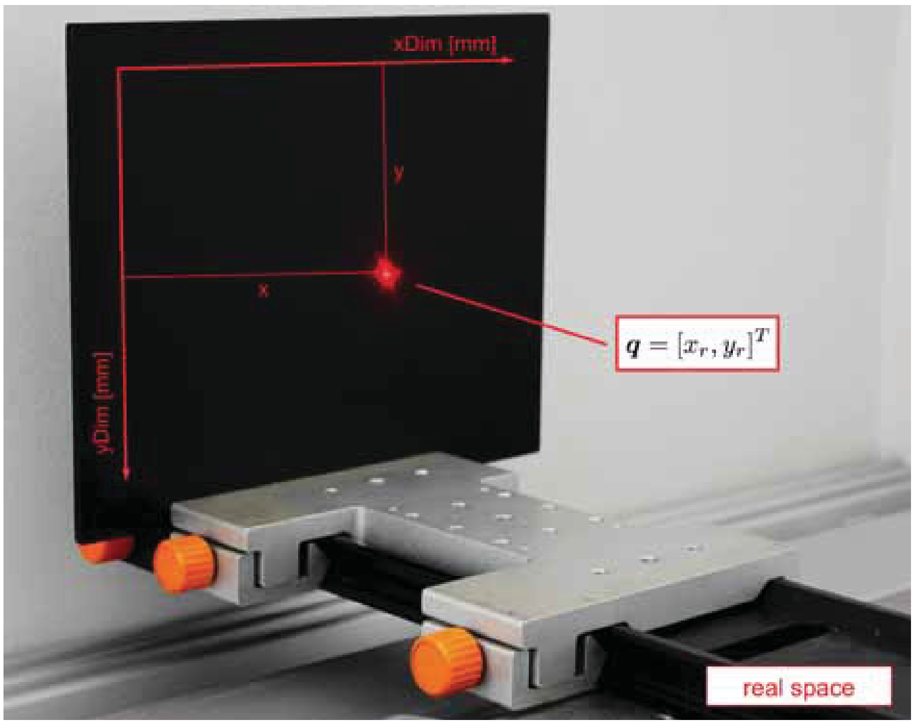


Figure 1.5: The real space is modeled by a Cartesian coordinate system with Euclidean distances. Changes of the laser spot's position can be measured in millimeters.

1.4 Research Questions

The primary goal of the master thesis is the conceptual development, i.e. a feasibility study, of an instrument, which is capable of measuring relative movements of structures caused by external forces. Image processing techniques enable the registration of these environmental changes. A set of procedures is necessary to acquire the desired information. For every process step, there exist several possible alternative methods. A further secondary objective is the evaluation of the object segmentation, center determination and coordinate mapping functions. The thesis' scope also covers the question, how precisely the overall system is working. A laboratory prototype was constructed and tested to verify the complete concept.

Chapter 2

Principle of Operation

The methodology of the introduced application is based on the requirements of the hardware setup. Figure 2.1 shows the laboratory prototype, Figure 2.2 focuses on the arrangement of a single measurement station. The acquired image is processed via Matlab.

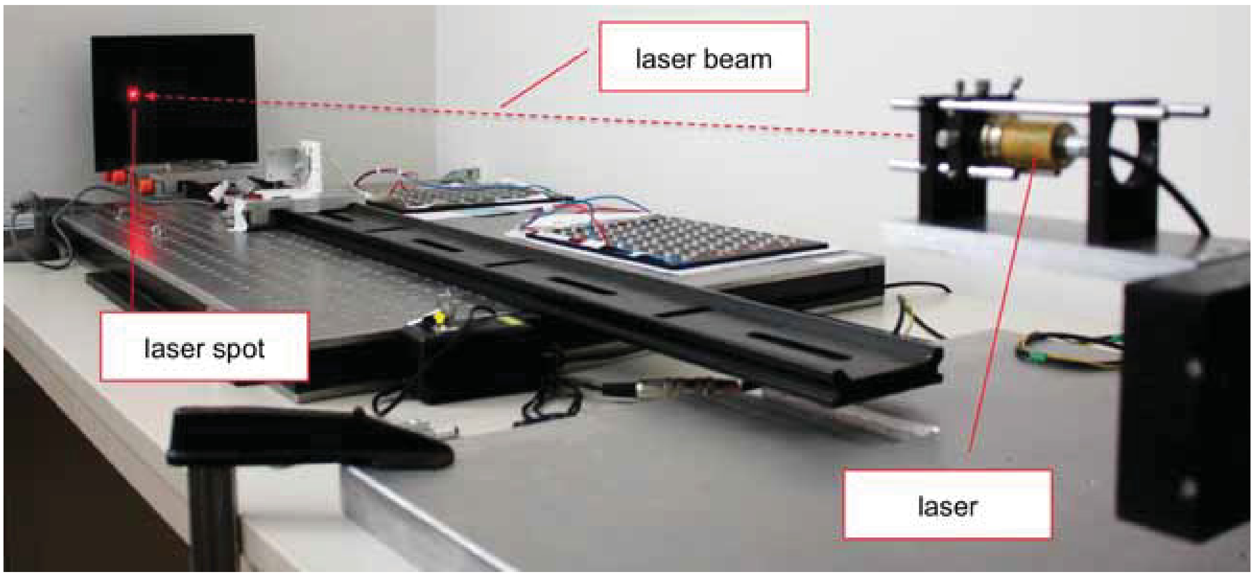


Figure 2.1: The conceptual laboratory prototype is a working miniature version of the actual instrument with all needed components: the laser, the target plate and the camera. The laser generates a laser beam. When it impinges the target plate, the laser spot is produced. See Figure 2.10 and the according text for a detailed description.

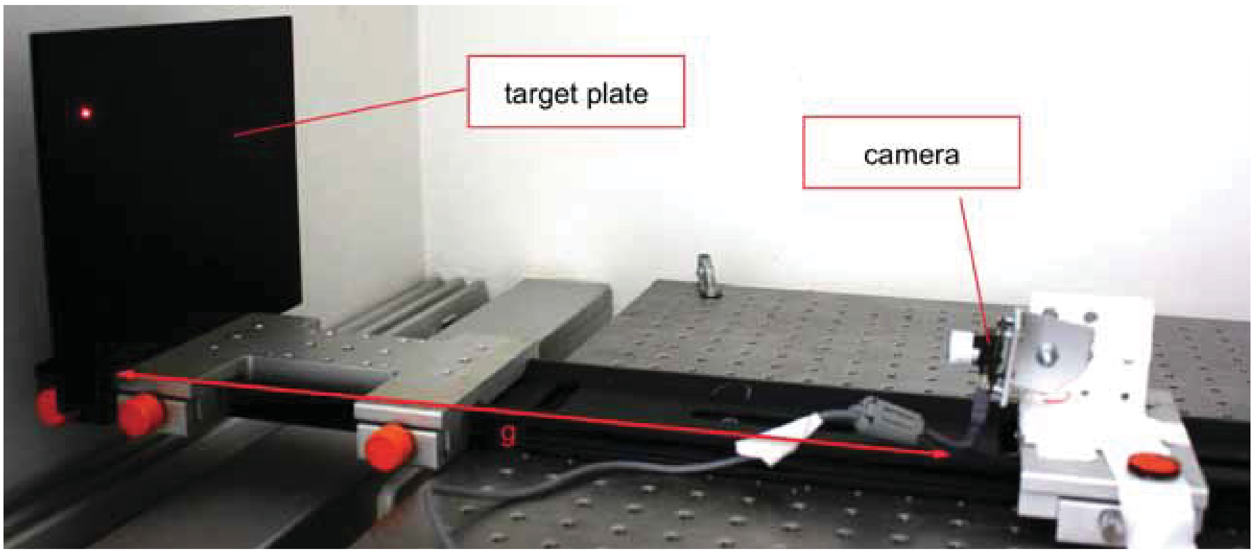


Figure 2.2: The photograph shows the optical arrangement of the application. The target plate is left and the camera is right. Both components are mounted on a static carrier cantilever to ensure a fixed object distance g. The schematic in Figure 2.13 illustrates the geometric relations.

2.1 Methodology and Processes

The measurement procedure consists of several processes, whereby each routine fulfills a specific task. The image is acquired on the so called *iconic level*, which implies the raw data. By performing the steps explained in this chapter, the data is reduced to its *symbol level* and only contains the relevant information for the following computations, i.e. the camera space coordinates in pixels. Figure 2.3 illustrates how the information density is increased.

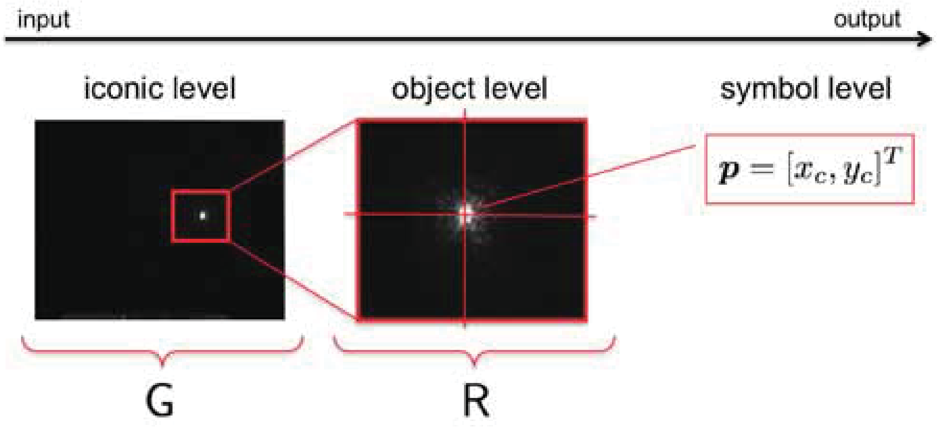


Figure 2.3: The acquired image G is the input for the overall process and resides on the so called iconic level. By extracting the image's characteristics, i.e. the regions of interest $\{\mathsf{R}_1, \ldots, \mathsf{R}_m\}$, the *object level* is reached. Each object's defining attribute is the center point $\boldsymbol{p}$, its coordinates are the desired information and represent the symbol level.

According to literature, the proposed approach is the most common workflow and best-practice in industries [4, 7, 28]. Note, that accuracy is lost over the whole process, because every step is based on assumptions and simplifications. The entire methodology is composed of following steps, which will be introduced in the next sections and explained in their according chapters:

1. structure image:
   (a) image acquisition,
   (b) image preprocessing,
   (c) object segmentation,
   (d) center determination;
2. system calibration;
3. coordinate mapping.

The processes yield a number of variables, which will not change their designated name throughout in this thesis. Table 2.1 presents an overview of the used terminology.

variable name:	**description:**
d	degree
f	focal length of a lens
t	threshold level
G	originally acquired grayscale image
D	normalized grayscale image containing $\{\mathsf{R}_1, \ldots, \mathsf{R}_m\}$
B	binarized monochrome image using t
R	image object respectively region of interest, $\mathsf{R} \in \mathsf{D}$
A^*	set of unclassified image objects, $A^* = \{\mathsf{R}_1, \ldots, \mathsf{R}_m\}$
A	set of classified image objects, $A = \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$ and $A \in A^*$
$\boldsymbol{p} = [x_c, y_c]^T$	camera space center coordinates, $\boldsymbol{u}$ in homogeneous form
$\boldsymbol{q} = [x_r, y_r]^T$	real space center coordinates, $\boldsymbol{v}$ in homogeneous form
$\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$	array of camera space points
$\mathsf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]$	array of real space points
m	number of a vector's elements or number of a matrix' rows
n	number of a vector's elements, a matrix' columns or iterations
i	indexing variable for rows, equivalent to y dimension
j	indexing variable for columns, equivalent to x dimension
$c_{ij} = c_{yx}$	an element's or pixel's value at i^{th} row and j^{th} column
k	various constants

Table 2.1: Used variable names and terminology in the thesis.

Structure Image

The following methods are used to structure an image, that includes preparing it and extracting the image object's attributes. The logical sequence of the implemented procedures is illustrated in Figure 2.4. The structured image is the origin for the *system calibration* and *coordinate mapping* steps.

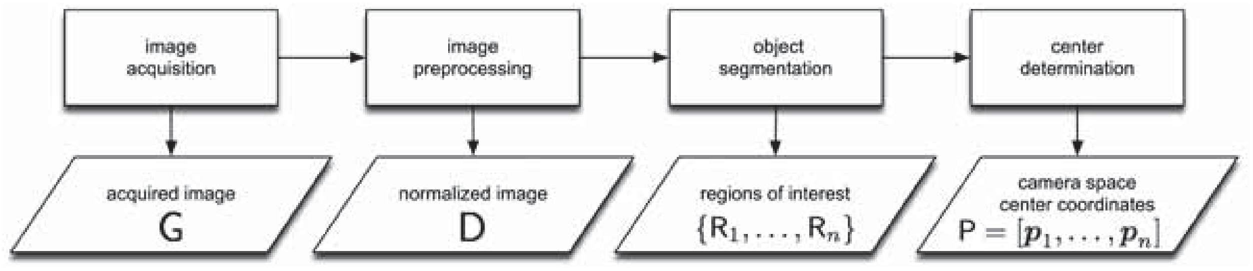


Figure 2.4: The *structure image* process is composed of several subroutines, which are necessary for the system calibration and the laser spot measurement. At first, the image is taken and prepared. Afterwards, the image objects are segmented and classified. The center coordinates of the camera space are the main attributes.

Image Acquisition

The camera is employed as an optical sensor, which delivers the image of the target. The image is in fact a raster of pixels with quantized values. The result is a matrix, whereby each discrete value represents the pixel's intensity, i.e. the brightness. Common encodings are 8 or 10 bit[1]. In this application, a grayscale camera with a resolution of 1280×1024 usable pixels and an 8 bit encoding is used. The device delivers the *original grayscale image* G.

Image Preprocessing

In Chapter 3 there're explanations of two methods for manipulating the image's properties to condition the signal for the impending operations:

- *contrast normalization* enables the usage of the whole encoding spectrum and
- *noise suppression* smooths the image through digital filtering.

The first method is used on the whole original image G. The *normalized image* is then referenced as D. The second method is only performed on portions of the image R.

[1]A 8 bit code has $2^8 = 256$ values, a 10 bit code has $2^{10} = 1024$ values.

Object Segmentation

Generally, an image holds parts that are more important than the rest, the so called *image objects* contained in the set $A^* = \{\mathsf{R}_1, \dots, \mathsf{R}_m\}$. The identification of these objects can be implemented by analyzing the whole image and extracting integral features. A common approach for the named problem is *binary segmentation*, which is in fact a bright-dark detection. If a pixel's intensity is higher than a certain *threshold* t, it is considered as relevant. Vice versa, if it is below this threshold, it is irrelevant and belongs to the background. The introduced procedure delivers a monochromatic or *binary image* B.

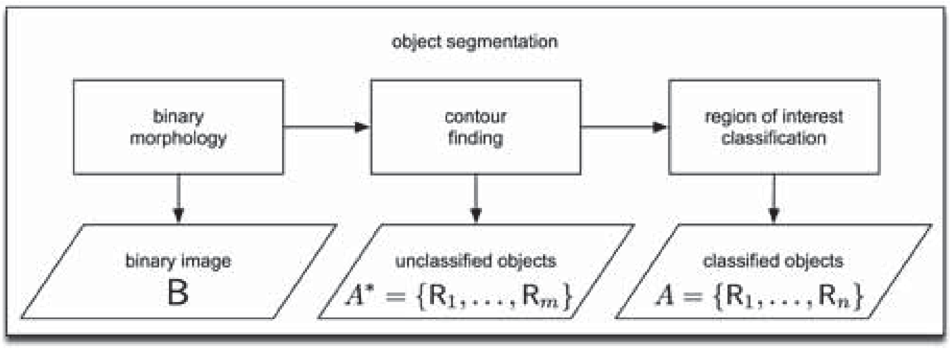

object segmentation
binary morphology
contour finding
region of interest classification
binary image
B
unclassified objects
$A^* = \{\mathsf{R}_1, \dots, \mathsf{R}_m\}$
classified objects
$A = \{\mathsf{R}_1, \dots, \mathsf{R}_n\}$

Figure 2.5: The segmentation process contains three subprocedures. The binary image B is the origin for locating the sought objects. Their contours deliver vital information about their usability, i.e. if the found objects in A^* are relevant or not. The identification of the ROIs in A is necessary for the following process steps.

Finding a meaningful threshold t is not trivial, as it depends on the image's quality and the treated problem. In Chapter 4 four different methods will be explained with histogram analysis:

- *Otsu's method*, which is based on the separation of two brightness classes,
- the *brightness class average*, the arithmetic mean value of the classes' peaks,
- *Gaussian curve intersection*, which utilizes two bell curves and
- the *maximal normal distance* of two brightness classes.

A good threshold is necessary to detect the image objects A^*, which are basically defined by their contours. Contour algorithms are based on the idea, that there are connected binary borders. By tracing these closed lines, the sought objects can be isolated. A confidence range is added to ensure that no relevant information is lost. Normally, it's implemented by utilizing a surrounding rectangle.

Another concept for object segmentation would be *Haar-like features*, they are based on arithmetic differences in parts of the image.

The image objects delivered by the segmentation process must be classified, because not every found object in A^* is relevant for the next computations. The routine generates defined areas known as *regions of interest* (ROI), which are contained in the set $A = \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$.

$$A^* \xrightarrow{\text{classification}} A, \tag{2.1}$$

whereby $A \in A^*$ and the number of their elements is $n \leq m$. The classifier decides with the help of quantitative characteristics if a feature R_i is usable or not. This issue will be explained in Chapter 4. The remaining relevant image objects in a A represent the regions of interest of the acquired image.

Center Determination

The following center determination methods are implemented and tested for their suitability to find the center coordinates $\boldsymbol{p} = [x_c, y_c]^T$ of a region of interest R:

- computation of the *centroid* or *center of gravity* of an area,
- *Gaussian curve extremum*, which uses two bell curves' peak values and
- conic matching by the fitting a *circle* and an *ellipse* based on contours.

When the described task has been finished, the regions of interest can be sorted according to their center coordinates yielding the matrix of camera points $\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$. Further explanations are found in Chapter 5.

System Calibration

Calibration is the act of applying a known input, i.e. an array of a-priori known real points $\mathsf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]$ and camera points $\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$, to the measurement system for the purpose of forming the logic how to interpret unknown input values. The described procedure must be performed before any actual measurement can take place. Chapter 6 describes how this linkage is created to employ *coordinate mapping*, Figure 2.6 shows the flowchart.

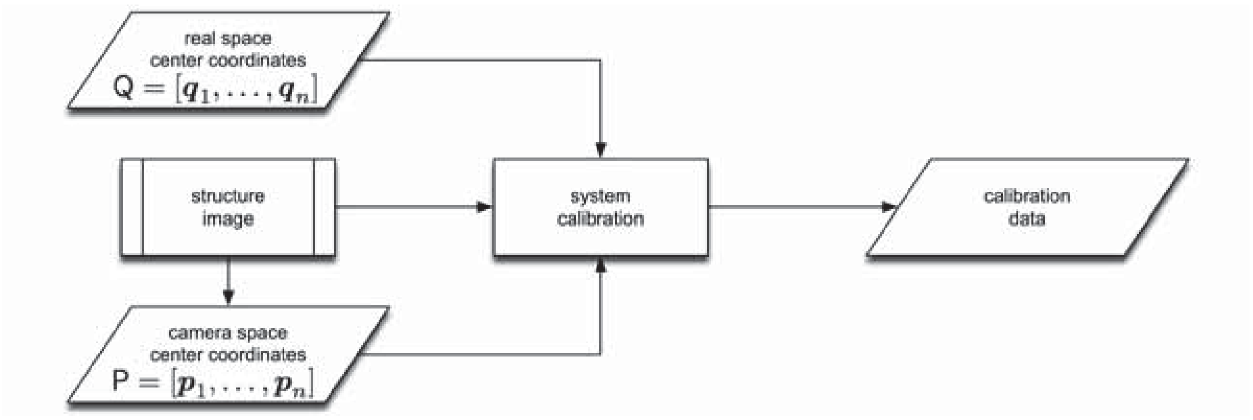

real space
center coordinates
$\mathsf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]$
structure
image
system
calibration
calibration
data
camera space
center coordinates
$\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$

Figure 2.6: At first the acquired image is structured and all ROI center coordinates are determined. During the calibration, a relation between the registered camera space coordinates P and the known real space coordinates Q is established. The linkage allows mapping of points between those spaces through a mathematical framework.

It is of utmost importance, that the relative distance g between the target plate and the camera is constant. Furthermore, the function settings must stay the same. If one of those factors changes, then the adjustment is not transferable to the actual measurement situation anymore.

Two ideas are introduced during the experiments in Chapter 8:

- a dedicated *calibration plate* utilizing an array of light emitting diodes (LED) and
- a movable *xy-table*, which generates the calibration pattern on the go.

Figur 2.7 shows how a plate with mounted LEDs is used to calibrate the system. Each LED's position in the real space is known a-priori. The calibration plate is manufactured precisely, but it must be replaced by the actual target plate after the adjustment. This causes changes to the installation and is a possible source of errors.

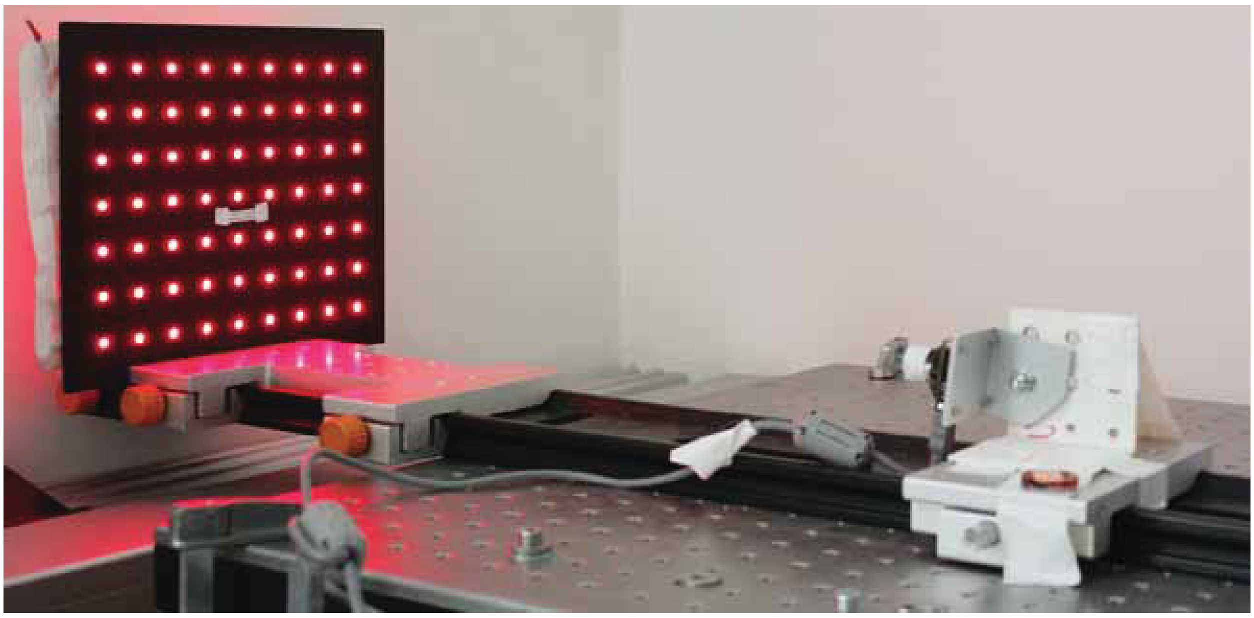


Figure 2.7: The distinct calibration plate is mounted with an array of exactly positioned LEDs. It must be replaced by the target plate for the following measurements.

The calibration utilizing the xy-table is more flexible and has the same assembly as the actual measurement environment, including the target plate. Nevertheless, the xy-table is victim to mechanical inexactness. Figure 2.8 gives an impression of how it works.

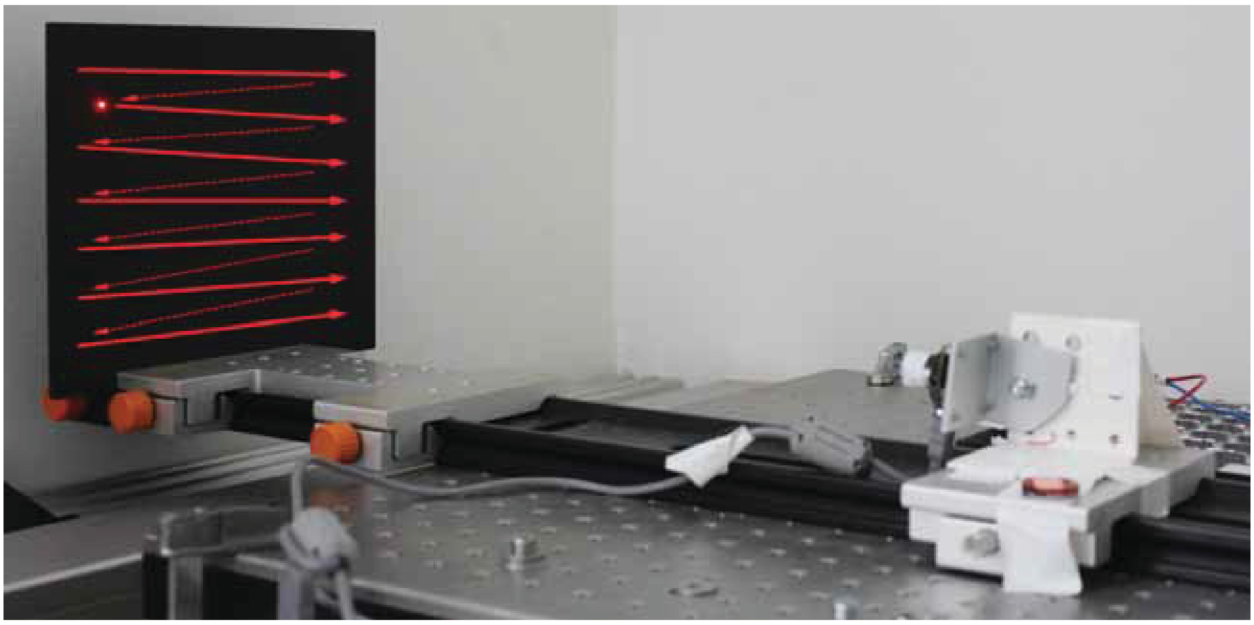


Figure 2.8: The calibration via the xy-table utilizes the target plate, i.e. there are no error-prone changes of the installation. The laser spot visits several exactly defined positions on its way from the left upper to the right lower corner. This procedure yields a laser spot calibration matrix. The recalibration of the system is also unproblematic.

Coordinate Mapping

The final step delivers the real space coordinates $\boldsymbol{q} = [x_r, y_r]^T$ of one laser spot. It assigns a value to the physical image acquired by the digital camera $\boldsymbol{p} = [x_c, y_c]^T$. This works through coordinate mapping methods, which will be explained in Chapter 6 in detail:

- *linear homography*,
- *polynomial control points* and
- *tensor interpolation* via *basis function sectioning* as well as *quad tree decomposition.*

The first two ideas are known approaches in image processing, the other one utilizes a set of bases functions and is new for such a kind of problem. Note, that measuring the same object with different methods delivers varying results. Figure 2.9 shows the flowchart of the process.

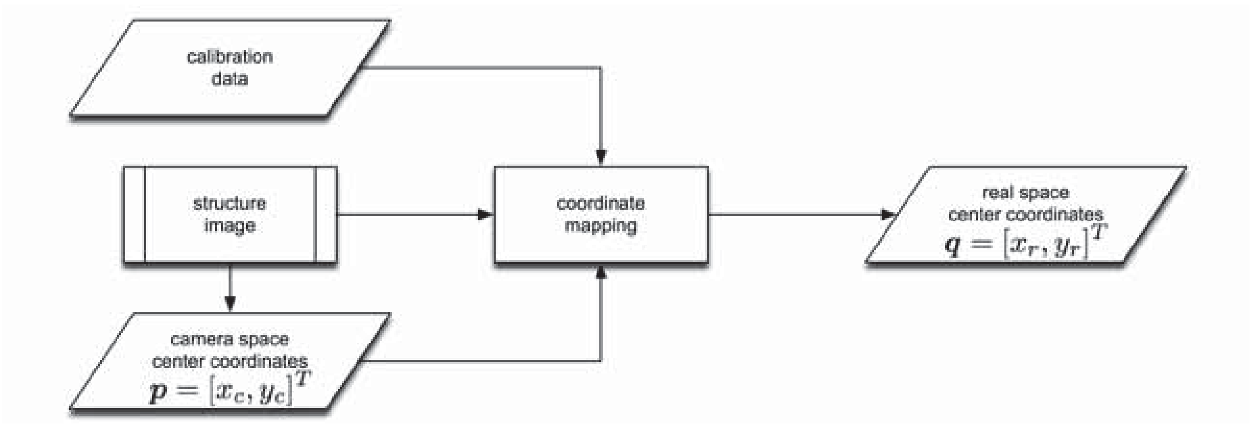

calibration data
structure image
coordinate mapping
real space center coordinates $\boldsymbol{q} = [x_r, y_r]^T$
camera space center coordinates $\boldsymbol{p} = [x_c, y_c]^T$

Figure 2.9: At first, the acquired image is structured and all ROI center coordinates are determined. In fact, there should only be one ROI: the wanted laser spot. The center coordinates of the camera space $\boldsymbol{p}$ are then combined with the calibration data to map the laser spot to its real space coordinates $\boldsymbol{q}$.

The repeatability of measurements is influenced by certain factors. When the unchanged target object is acquired more often, then the results are slightly varying. Chapter 7 will deal with this random error effects. Other sources of inexactness are systematic errors like the finite resolution of the camera, the limited calibration accuracy and the restricted mathematical models itself. Nevertheless, sub-pixel exactness can be achieved, because the procedures take numerous discrete values into account and their combination is more accurate than their single treatment.

2.2 Hardware Setup

The prototype's hardware consists of three main components:

- a laser,
- a target plate and
- a camera,

whereby the target plate and the camera are mounted on a static carrier cantilever, these parts compose a *measurement station*. The intended application can consist of several stations as illustrated in Figure 1.1. For the discussed experiments, only one was used due to economic reasons. Figure 2.10 shows the assembled components.

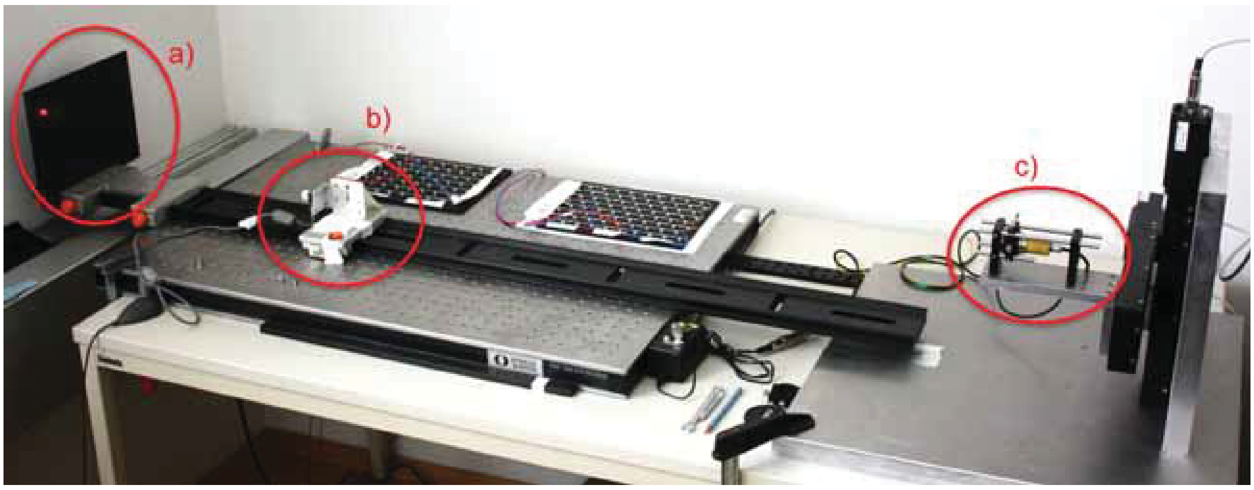


Figure 2.10: The prototype consists of a target plate a), a camera b) and a laser c). A measurement station is composed of a) and b), whereby both parts are unchangeable attached to a carrier cantilever to keep the relative distance static. The laser c) is mounted on a xy-table (there's also a setup with a linear drive) to enable relative movements between the laser beam and the measurement station.

The actual application will have a fixed calibrated laser, while the individual measurement stations are moving according to the mountain's geological activities, because they are mounted on the tunnel's or shaft's wall. Under laboratory conditions, it is irrelevant which one - the laser or the station - is moving, as it is only necessary to generate relative movements between those parts of the device. In Figure 2.10, a xy-table holds the laser, in another setup it is replaced by a linear drive. A xy-table is capable of moving its mounting platform in horizontal and vertical dimensions, a linear drive can only change the position in one dimension. Nevertheless, both systems are used, because the more complex xy-table is more sensible to errors and sometimes a two-dimensional movement is not required.

Optical Arrangement

The measurement station consists of the target plate and the camera. Digital measurements have high demands on the deployed sensor devices, therefore an industry level camera is chosen. The prototype uses the *VRmC-9+ COB (b/w)* from *VRmagic*[2]. It has a 1/2" optical format and an usable resolution of 1280×1024 pixels. To acquire a sharp image with high contrast, the lighting conditions must be acceptable. The camera's exposure time is a parameter, which can be adjusted to fit the environmental conditions, but only within certain limits. A grayscale camera was selected, because no color information is needed. Moreover, this design is more accurate than comparable color cameras with their relatively rough Bayer patterns. This avoids other spheric errors resulting from that design and weakens chromatic aberration, because only monochromatic light is detected. In addition, there's no need for another white-balancing step. Figure 2.11 shows the device.

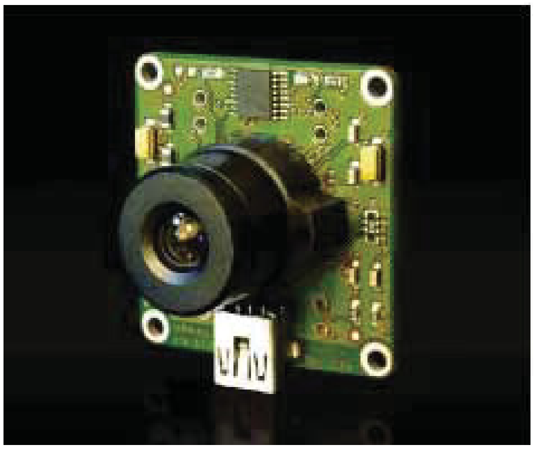


Figure 2.11: The VRmagic VRmC-9+ COB (b/w) is a 1/2" grayscale industry approved camera with a resolution of 1280×1024 pixel. The resolution must be high enough to catch all the details, as the whole target object is pictured at once. Tested objectives are a normal-angle lens with $f = 12mm$ and a wide-angle lens with $f = 3.6mm$.

The standard normal-angle lens has a focal length of $f = 12mm$, the calculations are also carried out for a wide-angle lens with $f = 3.6mm$. A 1/2" chip has an image hight $B_h = 4.8mm$ and an image width $B_w = 6.4mm$, the target plate has a dimension of $150mm \times 200mm$. An additional safety margin of $k = 20mm$ is added, therefore the object hight is

[2]©VRmagic GmbH, Mannheim, Germany, www.vrmagic.com

$G_h = (150 + 20)mm$ and object width is $G_w = (200 + 20)mm$. By putting the values into Equation 2.2, the value for the object distance g is computed using commonly known optical coherency [4]:

$$G = B\left(\frac{g-f}{f}\right) \quad \text{and consequently} \quad g = g(f) = f\left(\frac{G+B}{B}\right). \tag{2.2}$$

The computation must be done for the *hight* and for the *width* values, retrieving the object distance in respect to the height g_h and the width g_w. Thus, the chosen value for g is

$$g = \max(g_h, g_w). \tag{2.3}$$

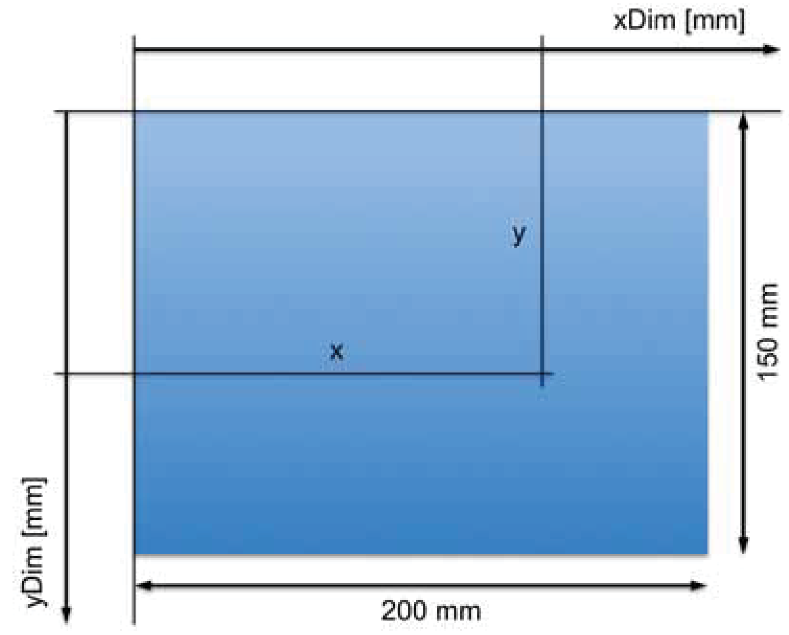


Figure 2.12: The dimensions of the target plate are equivalent to the object size parameters $G_h = 150mm$ and $G_w = 200mm$ with an additional safety margin $k = 20mm$. The origin of the Cartesian coordinate system is the left upper corner.

This delivers an object distance $g(f = 12mm) = 440mm$ and $g(f = 3.6mm) = 140mm$. The detailed calculation can be found in Appendix A. Figure 2.13 illustrates the optical arrangement. The amount of illumination can only be controlled by manipulating the exposure time with the software, because the aperture size is unchangeable. The depth of field can be set through manually focusing the objective to adjust the sharpness of the acquired image.

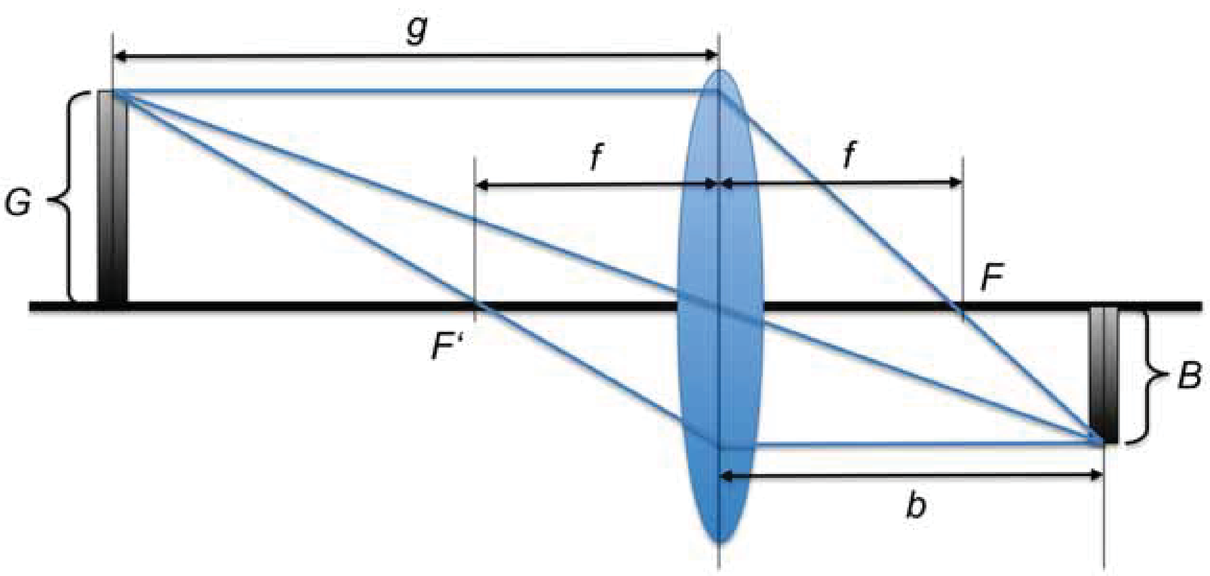


Figure 2.13: The schematic of an optical lens illustrates the relation between the object G and image B, the object distance g and image distance b as well as the focal length f. The correct distance between the camera and the target is essential for good results. It is mandatory to capture the whole object sharply with full resolution. See Figure 2.2 for the laboratory setup.

The camera's finite resolution in x and y dimension, $x_{res} = 1280pixel$ and $y_{res} = 1024pixel$, delivers a certain inexactness. As a result, each pixel covers only a quantized distance of the object's real dimension $G_h = 150mm$ and $G_w = 200mm$. Consequently, the error of the hight Δy is

$$\Delta y = \frac{G_h + k}{y_{res}} = \frac{150 + 20}{1024} = \pm 0.166 \frac{mm}{pixel} \tag{2.4}$$

and the error of the width Δx is

$$\Delta x = \frac{G_w + k}{x_{res}} = \frac{200 + 20}{1280} = \pm 0.172 \frac{mm}{pixel}. \tag{2.5}$$

In order to improve the lighting conditions, an additional interference filter is mounted onto the camera's objective. So only certain wavelengths can pass through the filter, e.g. the laser's and the LEDs' red light with a wavelength of around $\lambda = 650nm$. As a result, common daylight can widely be prevented from entering the camera.

Target Plate

There are two different kinds of target plates employed in this application:

- a black *solid eloxadized aluminium* plate and
- a plate composed of *electro-active glass*.

The metal plate is used for experimental testing, but the final instrument should utilize the glass. Electro-active glass is a composite material, which is capable of changing its opalescence instantly and with no alteration of light level. This means, the glass can switch from transparent to opaque and vice versa. Two ways are known to implement the mentioned function: electro-chromatic glass, which is based on a chemical effect, and the mentioned electro-active glass.

The experiments are carried out by utilizing the *SGG PrivaLite*[3] glass. Its primary deployments can be found in architecture. The idea to use it for a measurement instrument is new. Figure 2.14 illustrates the design of the material.

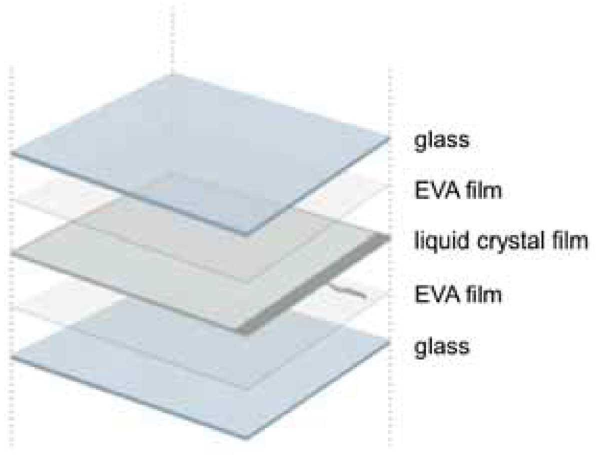


Figure 2.14: Layer design of the Saint-Gobain PrivaLite electro-active glass.

The product is composed of two sheets of extra-clear glass encapsulating a liquid crystal film sandwiched between two ethylene vinyl acetate (EVA) interlayer films. The material is constituted by two plates of polyethylene terephtalate (PET) films, coated with a transparent metallic deposit and laminated together with a very fine layer of liquid crystal gel. When an alternating voltage of $100V$ is applied to the lateral copper electrodes, the liquid crystals

[3]©Saint-Gobain Glass, Courbevoie, France, www.privalite.com

orient themselves in the same direction. The initially milky film gets instantly transparent. The power consumption is $24VA/m^2$. Its light transmission in transparent mode is 77% and 76% when it's opaque.

The application utilizes this kind of glass, because it enables the reachability of a series of consecutively installed measurement stations. Figure 2.15 shows the opaque glass and the laser spot is generated directly on its surface. When the glass is transparent, the laser beam can pass through it and reaches the solid target plate. The effect can be observed in Figure 2.16.

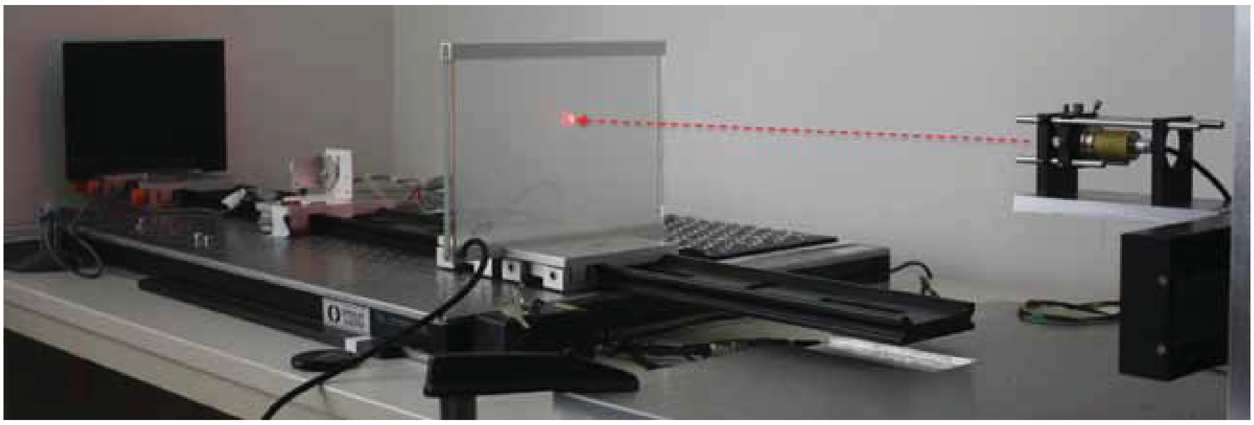


Figure 2.15: The electro-active glass is turned off and opaque. The laser spot is generated directly on the glass plate's surface by the laser beam.

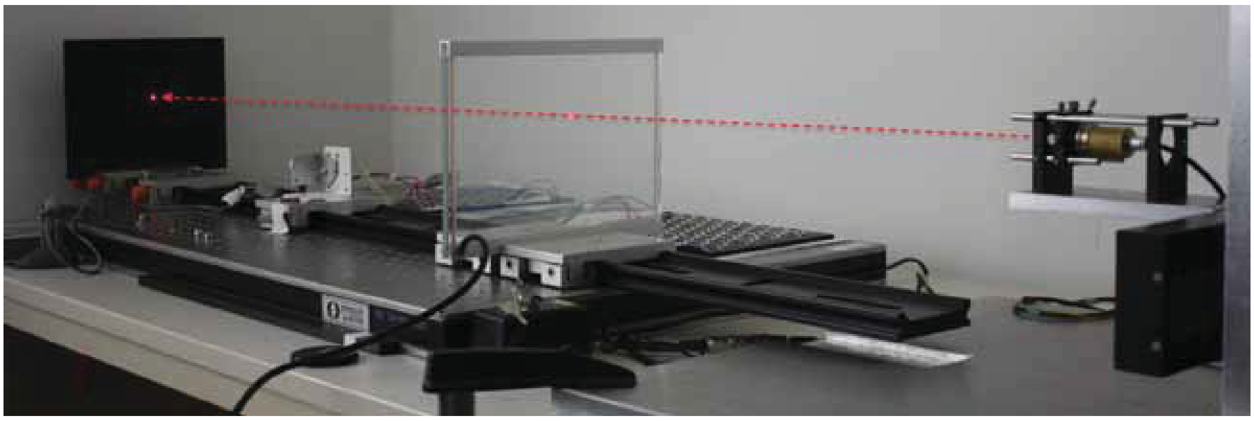


Figure 2.16: The electro-active glass is turned on and transparent. The laser beam passes through the glass and impinges the metal target plate behind it.

Calibration Plate

The calibration plate features an array of mounted LEDs, which are the *L-483 SRSGW* from *Kingbright*[4]. Those LEDs' characteristics are: a diameter of $\varnothing 5mm$, a flat head, a white diffused light and a wavelength of $\lambda = 660 \pm 20nm$. Their operating point is approximately at $2.1V$.

The LEDs are arranged in a $(m \times n) = (7 \times 9)$ matrix, i.e. there are 63 LEDs in total. Their horizontal and vertical distance to each other is $\delta_x = \delta_y = \delta = 25mm$, the manufacturing tolerance is $\pm 0.01mm$. The size of the grid defines the achievable accuracy for the coordinate mapping functions. Note, that in some experiments not all LEDs are used. The information of each LED's location in the real space is needed for the calibration process and it's saved in the matrix $\mathsf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]$.

Linear Drive and xy-Table

The experiments in Chapter 8 take advantage of two drive systems:

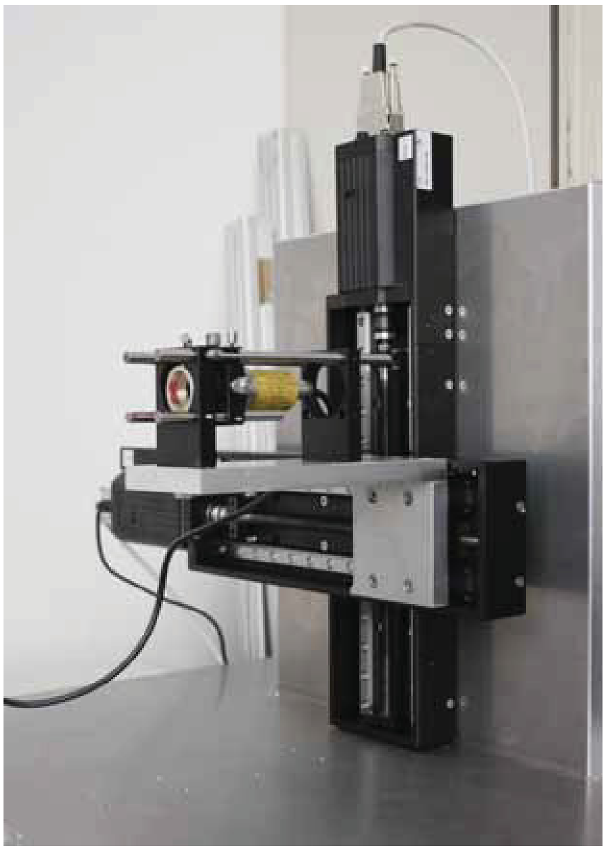
- the linear drive *THK Guide Actuator KR* for one-dimensional movement and
- the xy-table *iTK Dr. Kassen ST9* for two-dimensional, orthogonal movement.

The first device is operated via the *Bernecker & Rainer*[5] *PS465/CP360* Programmable Logic Controller (PLC) with the attached *Acopos 1045* controller. This arrangement theoretically offers a position repeatability of $\pm 0.01mm$. The PLC runs an Open Process Control (OPC) service, allowing the direct reading and writing of files via common Ethernet with TCP/IP protocol. Note, that it's not working in real time any more! The communication with Matlab is established through the standardized Process Visualization Interface (PVI).

The communication of the xy-table is set up through a common RS232 serial interface between a computer and the *Pollux 2* controller. The achievable position repeatability is $\pm 0.02mm$ according to its datasheet. Figure 2.17 shows the vertically assembled device and the mounted laser.

[4]©Kingbright, Taiwan, www.kingbright.com

[5]©Bernecker & Rainer Industrie-Elektronik GmbH, Eggelsberg, Austria, www.br-automation.com


(a) The laser at the start position.

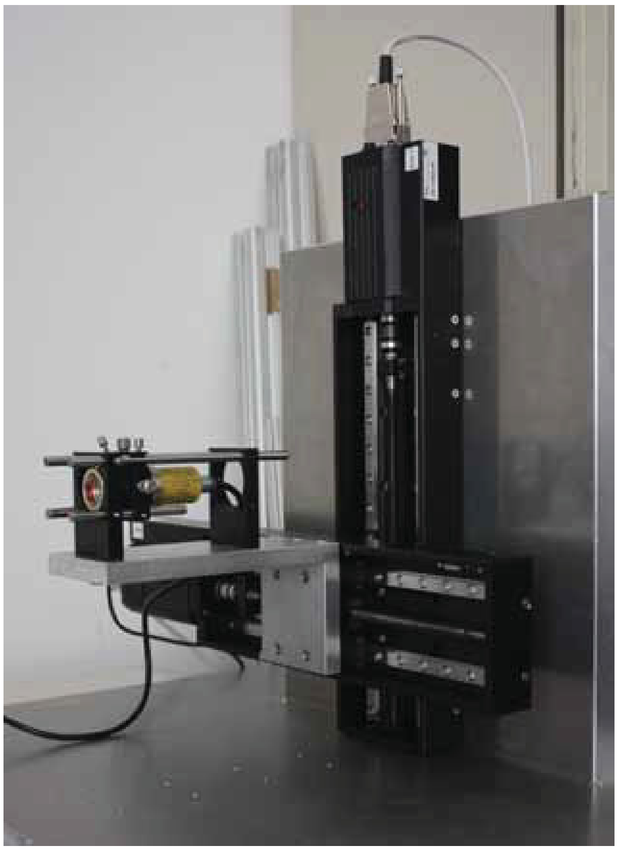

(b) The laser at the end position.

Figure 2.17: The xy-table changes the position of the mounted laser. The movement is orthogonal and two-dimensional. The vertical installation enables testing and calibration of the system.

Laser

The application utilizes a red laser from *Schäfter und Kirchhoff*[6], to be specific the *55Lpm-653x11-S01-07+LR25S1000 SN:334*. The emitted coherent light has as wavelength of $\lambda = 638nm$ with a radiation power of $11mW$, this is equivalent to a $3B$ laser class. It is energized by a direct current of $5V$.

The measured object's size b should be at least five times bigger than the light's wavelength λ to avoid certain optical distortions. This indicates, that the laser spot's position can't be located more accurate than determined by this physical barrier:

$$b \geq 5\lambda \qquad \text{resulting in} \qquad b \geq 3.2\mu m. \tag{2.6}$$

[6]©Schäfter und Kirchhoff, Hamburg, Germany, www.sukhamburg.com

2.3 Software Implementation

All calculations are done with *Matlab*[7] *version 7.8.0.347 (R2009a)*. Special advantage is taken of the *Image Acquisition Toolbox* (imaqtool). Additionally, VRmagic delivers a configuration tool: *CamLab 2.7.2.6*. It's useful for setting up the camera for different lighting conditions.

When performing image operations in Matlab, the x dimension is equivalent to the columns of a matrix and the y dimension is equivalent to its rows. Furthermore, Matlab sets the image's origin point, meaning the c_{11} pixel with the $(i, j) = (y, x) = (1, 1)$ coordinates, to the left upper corner. As seen in Figure 2.12, this circumstance leads to the decision to put the real world origin to that position. Depending on the camera's resolution, there are m rows and n columns. The acquired grayscale image is therefore a $(m \times n)$ matrix, whereby each element $c_{ij} = c_{yx}$ is equivalent to the pixel's 8 bit encoded brightness ranging from 0 (black) to 255 (white). Binary images are just represented by the values 0 (black) and 1 (white).

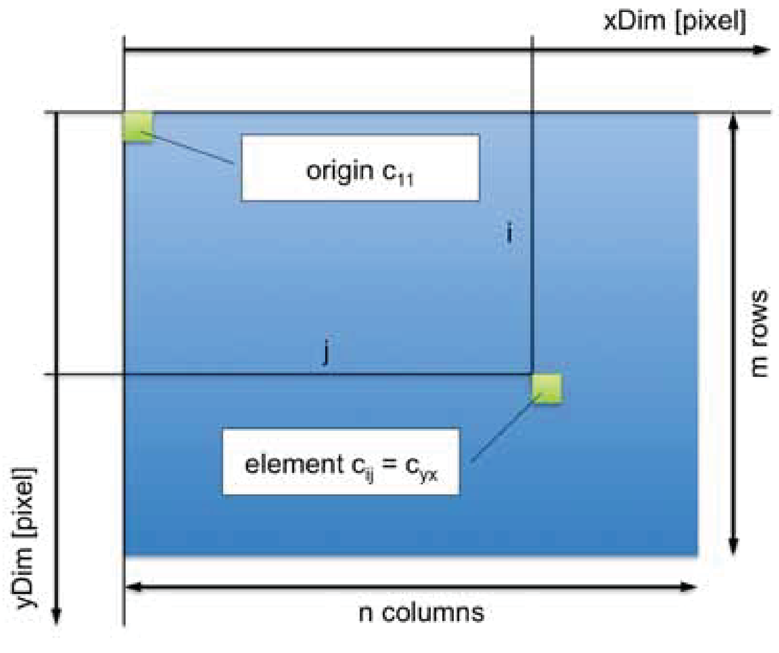


Figure 2.18: Each matrix has m rows, representing the y dimension, and n columns, representing the x dimension. The corresponding indexing variables are i and j. Each element $c_{ij} = c_{yx}$, i.e. a pixel, has a defined position at the i^{th} row (y dimension) and the j^{th} column (x dimension) as well as an 8 bit brightness value ranging from $0 \ldots 255$. The origin c_{11} is in the left upper corner of the matrix.

[7]©The MathWorks Inc., Natick, MA, United States of America, www.mathworks.com

Chapter 3

Image Preprocessing

The preprocessing yields the normalized image D from the original image G:

$$\mathsf{G} \xrightarrow{\text{normalization}} \mathsf{D}. \tag{3.1}$$

There are two dominating external factors: the lighting condition, which causes unequally illuminated images and noise, which is a random variation of the measured signal's value. It's essential to take care of these undesired trends inside the images to avoid misinterpretation of results. As a consequence, the methods described in this chapter try to handle these effects economically [4].

3.1 Contrast Normalization

This method is a simple and efficient way to distribute the brightness over the whole available range of pixel intensities. It's basically a so called *point operation*, because it only takes the currently investigated pixel value into account. To be correct, it's not completely local, as the minimal and a maximal values depend on the original image's brightness. At first, the parameters must be read out from the image. Except for this aspect, its handling is just like a common point operation.

The original grayscale image G has an 8 bit encoding, i.e. the pixels' values are theoretically ranging from $0 \dots 255$. In many cases they don't use the provided space. The smallest and biggest values become the minimal value g_{min} and maximal value g_{max} of G. The normalized image D utilizes the whole value spectrum by eliminating the unused space. Its minimal value d_{min} and maximal value d_{max} can be set to fit the problem. In this application, the original integer values are replaced by double values. Matlab provides better support for normalized data, thus

$$d_{min} = 0 \qquad \text{and} \qquad d_{max} = 1. \tag{3.2}$$

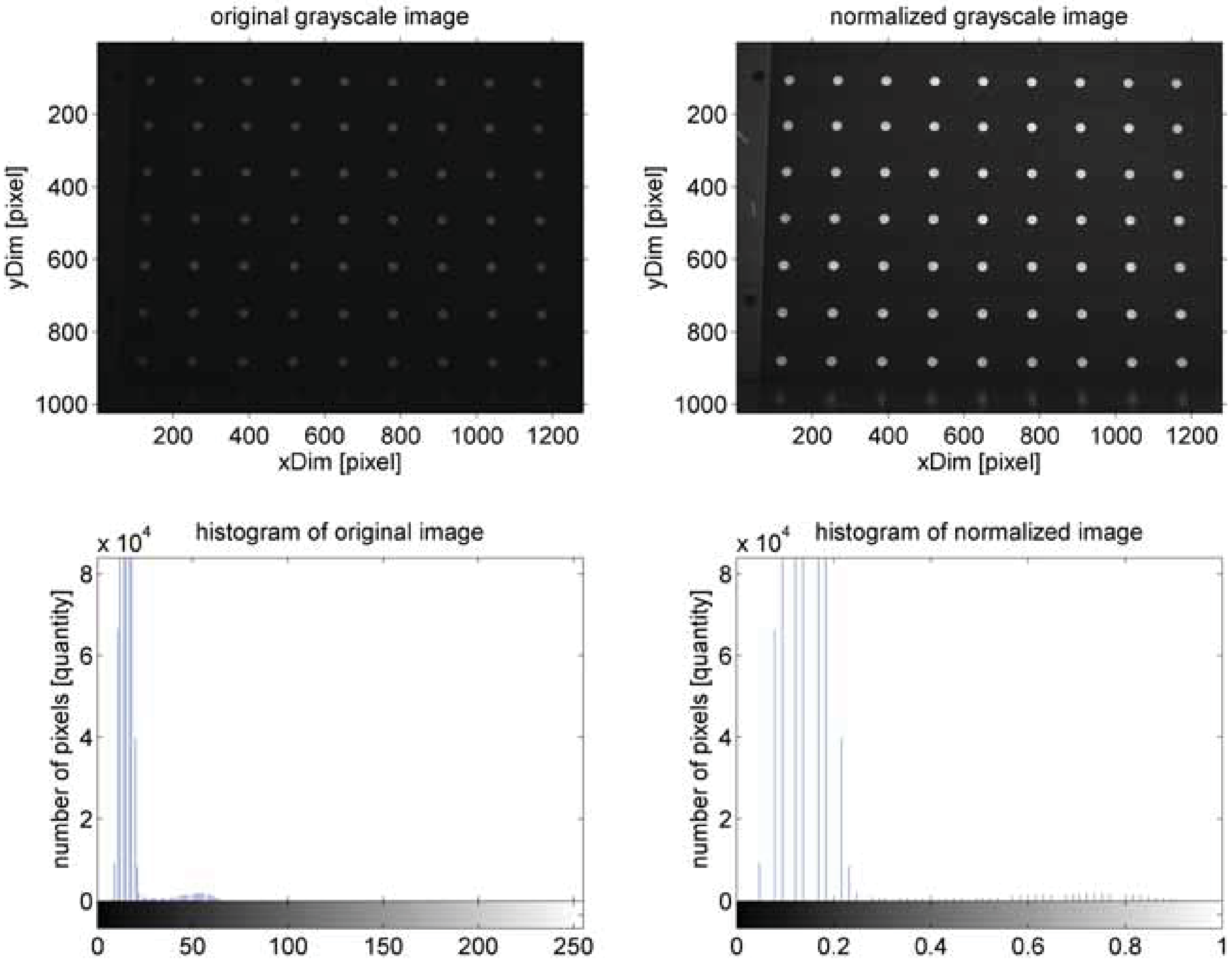


Figure 3.1: This example employs the turned-off calibration plate for demonstration. The original grayscale image G has very low contrast, the histogram shows values ranging from approximately 10 to 70 (integer values). The normalized grayscale image D uses the whole spectrum ranging from 0 to 1 (double values). The result is an image with a higher contrast.

The values for g_{min} and g_{max} are acquired by analyzing the original image G. The formula in Equation 3.3 delivers the output brightness $c_{ij}^{(out)}$ for every input pixel $c_{ij}^{(in)}$ of the i^{th} row (y dimension) and j^{th} column (x dimension):

$$c_{ij}^{(out)} = \frac{d_{max} - d_{min}}{g_{max} - g_{min}}(c_{ij}^{(in)} - g_{min}) + d_{min}. \tag{3.3}$$

The Matlab implementation is very efficient as all variables are scalars.

```
D = (dMax - dMin) / (gMax - gMin) * (G - gMin) + dMin;
```

Source Code 3.1: Matlab code to perform image normalization.

3.2 Noise Suppression

Filtering is necessary to reduce the noise of images and to smooth value peaks. The procedure is a local operation, because it adds information from neighboring points [4]. The defined neighborhood is called *tile*, the weighted sum of the pixels included in this tile delivers the filtered value. The area enclosed by the tile is called *masksize* f_m.

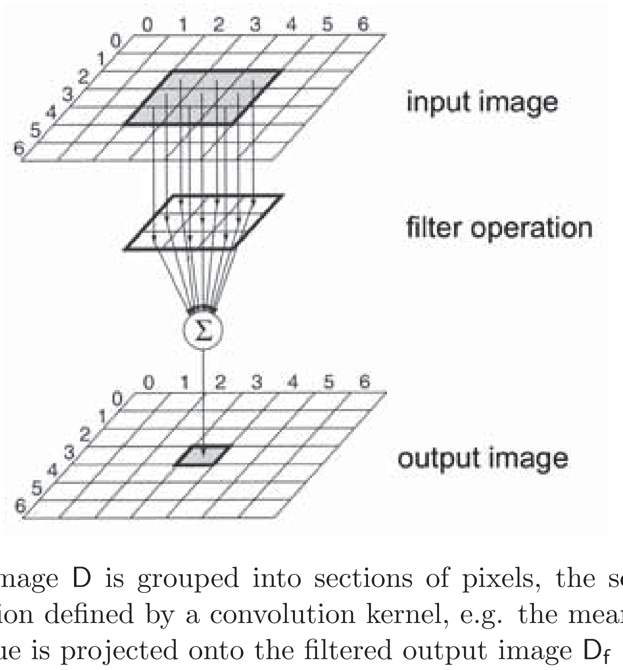


Figure 3.2: The input image D is grouped into sections of pixels, the so called *tiles*. Each tile undergoes a filter operation defined by a convolution kernel, e.g. the mean brightness of the pixel group. The resulting value is projected onto the filtered output image $\mathsf{D_f}$ [4].

In this implementation, the mask is standardized to the quadratic masksize to obtain a smooth image. See Figure 3.3 for an example of how a scattered image of a laser spot is filtered.

```
mask = ones(fm)/fm^2; % fm ... filter masksize
Df = filter2(mask, D); % linear 2D FIR filter
```

Source Code 3.2: Linear two-dimensional filtering with Matlab.

Matlab's function *filter2()* filters the input data D with the two-dimensional FIR filter mask f_m. The application uses $f_m = 5$ for the operation. It computes the result, $\mathsf{D_f}$, using two-dimensional correlation. The delivered convolution kernel is then used for the filter operation.

Matlab utilizes a *finite impulse response* (FIR) filter for digital image processing. It's impulse response is of finite duration, because it settles to zero in finite time. There are no internal feedback loops, i.e. the implementation is easier and there aren't any rounding errors compounded by summed iterations. FIR filters are stable as they do not possess any poles. As a drawback, they need more computation power than other comparable filter types.

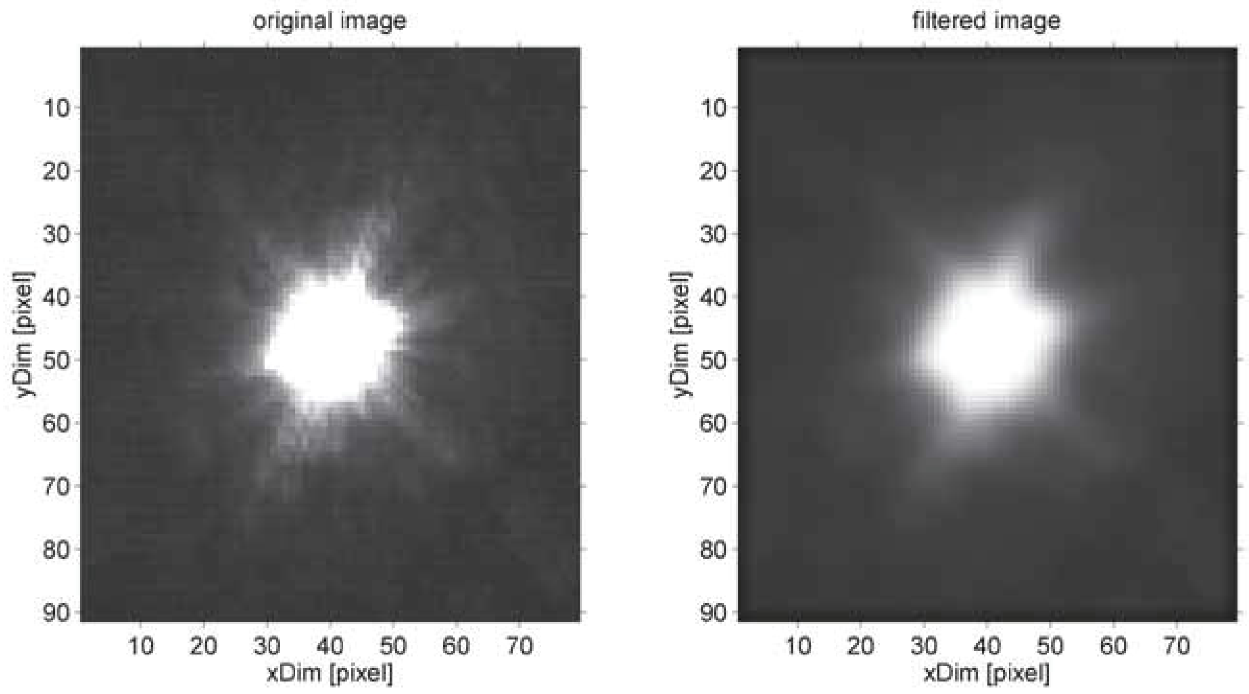


Figure 3.3: The original image D shows a scattered laser spot. It would be hard for the following operations, e.g. the contour algorithm, to handle such raw data. The example uses a masksize of $f_m = 5$ to filter the original image. The filtered image $\mathsf{D_f}$ is much smoother and the laser's gradient is well defined. Note, that the procedure is often carried out only for certain regions R_i of an image D.

Chapter 4

Object Segmentation

The *set of image objects* $A^* = \{\mathsf{R}_1, \ldots, \mathsf{R}_m\}$ is essential for image processing, because it makes a statement of what is actually measured. Segmentation is the process of isolating these image objects from their background, because each R_i is part of the complete image D:

$$\mathsf{R}_i \in \mathsf{D}. \tag{4.1}$$

Generally, this is a hierarchical approach: a workspace, which contains the image objects, is defined. After the segmentation, they are classified for their relevance, delivering the *set of regions of interest* $A = \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$, whereby

$$A \in A^* \qquad \text{and} \qquad n \leq m, \tag{4.2}$$

when n and m are the numbers of contained elements. This yields the following procedure:

$$\mathsf{D} \xrightarrow{\text{binarization}} \mathsf{B} \xrightarrow{\text{segmentation}} A^* = \{\mathsf{R}_1, \ldots, \mathsf{R}_m\} \xrightarrow{\text{classification}} A = \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}. \tag{4.3}$$

Note, that B is just used to find the ROIs, but further computations are carried out on D. The determination of the *center coordinates* $\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$ is the next process step. The sections in this chapter introduce several methods for *binary segmentation*, the most common technique to identify image objects via contour finding. Additionally, the principles of classification are described [4].

4.1 Binary Morphology

Methods, which are based on the determination of thresholds in grayscale images, are utilized for binary segmentation. This basic principle is described as

$$c_{ij}^{(out)} = \begin{cases} 1 & \text{if } c_{ij}^{(in)} > t \\ 0 & \text{else,} \end{cases} \tag{4.4}$$

when $c_{ij}^{(in)}$ and $c_{ij}^{(out)}$ are the intensities of the input and output pixels of the i^{th} row (y dimension) and j^{th} column (x dimension) of the image D. This means, if the input value is higher than a specific threshold t, then the investigated pixel's output value is set to 1 (white), if it's lower it is set to 0 (black). The result is a black and white, i.e. binary or monochrome, image B. Finding a suitable threshold for the binary morphology is one of the major difficulties in image processing. The value of t tends to change with different lighting conditions, especially when there's a notable gradient. There are two different approaches:

- *global thresholds*, which are valid for the complete image D, and
- *local thresholds*, only determined from a local feature R_i.

This means, instead of having a single global threshold, the local one smoothly adapts its value across the image. The task is to find an appropriate level for t, which suits the application. The identification and segmentation of image objects is simplified. As a downside, most threshold methods are not robust and deliver different solutions for the same image under unstable conditions. As a result, they must be determined by hand. The process additionally induces loss of information, because the original gray values, e.g. ranging from $0 \ldots 255$ for an 8 bit encoding, are reduced to just 2 values. Another major problem is, that only the intensity is considered, but not any relationships between the pixels. There is no guarantee that the pixels identified by the thresholding process are contiguous. Therefore, it's recommended to filter the image before binarizing it to avoid sparkles. This effect would cause the contour algorithm to find many unwanted objects, like in Figure 4.6.

Figure 4.1 shows an unfiltered grayscale image of a laser spot. This example will demonstrate, that even in this simple case different threshold strategies will lead to varying results for the same input. Figure 4.3 shows the outcome of binarizing the image with four types of methods, all of them are based on considering the pixels' brightness. They will be described in the following sections.

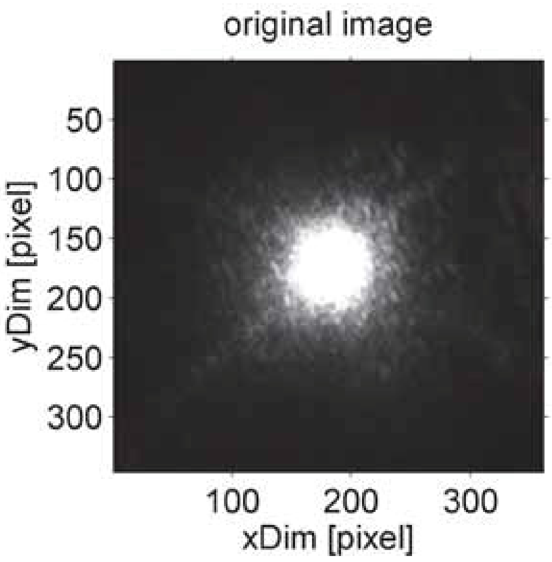


Figure 4.1: The image object R_i of the original grayscale image D contains the laser spot. In this example, it has an 8 bit encoding, i.e. 256 discrete values. In fact this is already an extracted object, therefore the resulting threshold will be valid only locally. If it is the goal to find the object's contour, then the image should be filtered.

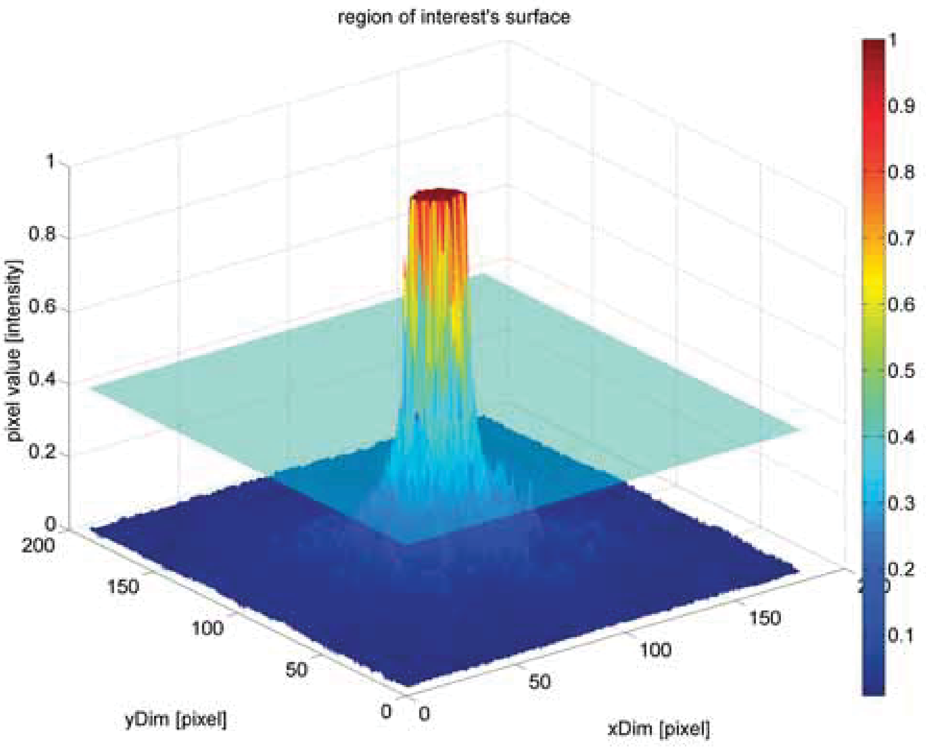


Figure 4.2: The threshold t acts as classifier: each pixel value below t is set to 0, each value above t is set to 1. The surface plot visualizes the issue, whereby the transparent plane is equivalent to the threshold level.

A local threshold is better suited for the discussed problem, because the black to white ratio is not balanced globally. Most of the original image is black, only a few parts are gray and white. Computing a global threshold under this condition would lead to biased results.

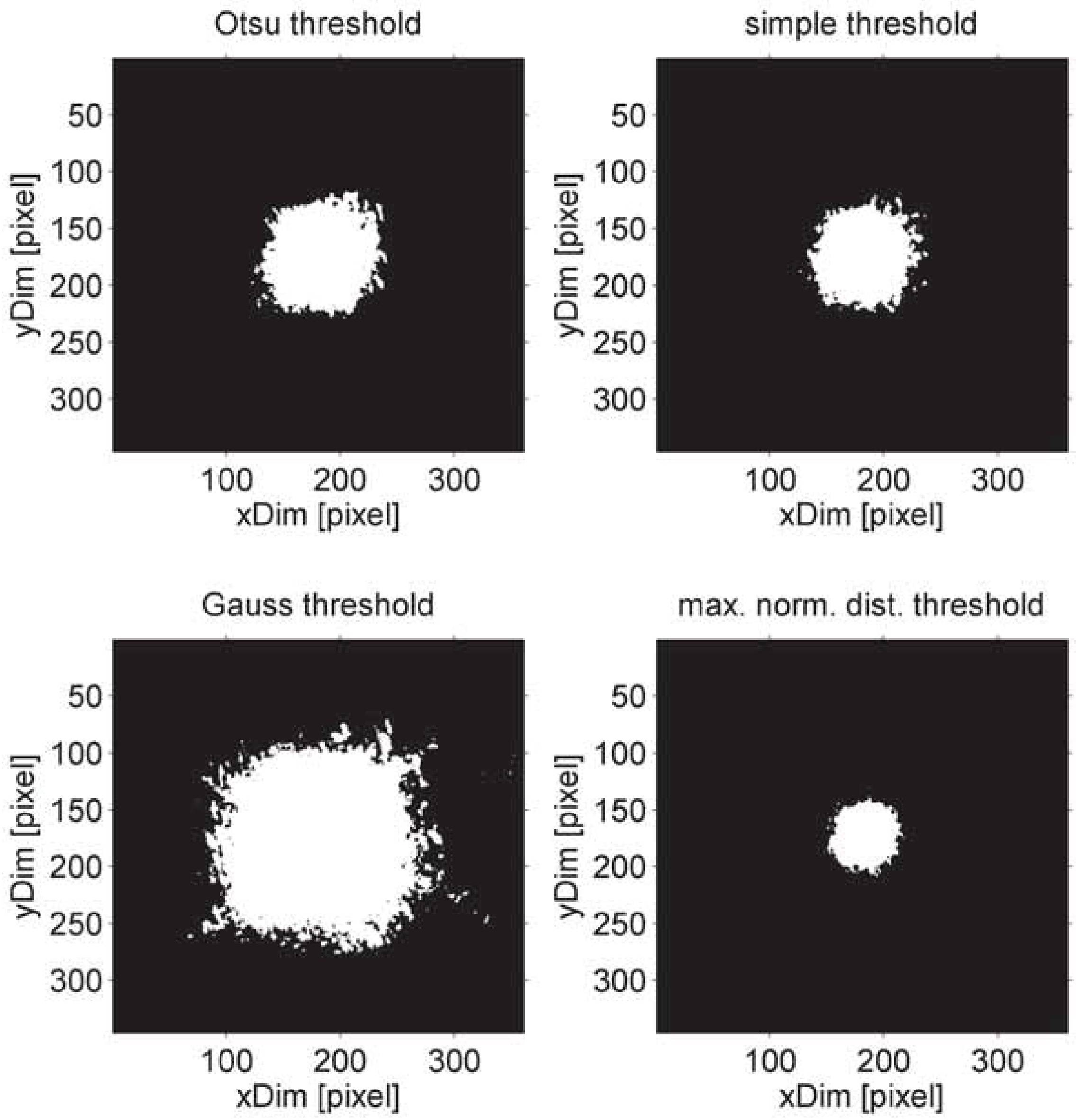


Figure 4.3: Each binarizing method delivers a different outcome. There's no general statement of which one is the best, it depends on the investigated problem. During the experiments, the local Otsu threshold is used, as it proofed to be the most reliable one.

In order to compute the different thresholds, the pixels are classified into discrete bins according to their intensities. The resulting histogram in Figure 4.4 utilizes 1024 bins. The pixel values are rescaled to the common 0 to 1 range through normalization. Figure 4.5 shows the associated levels for t.

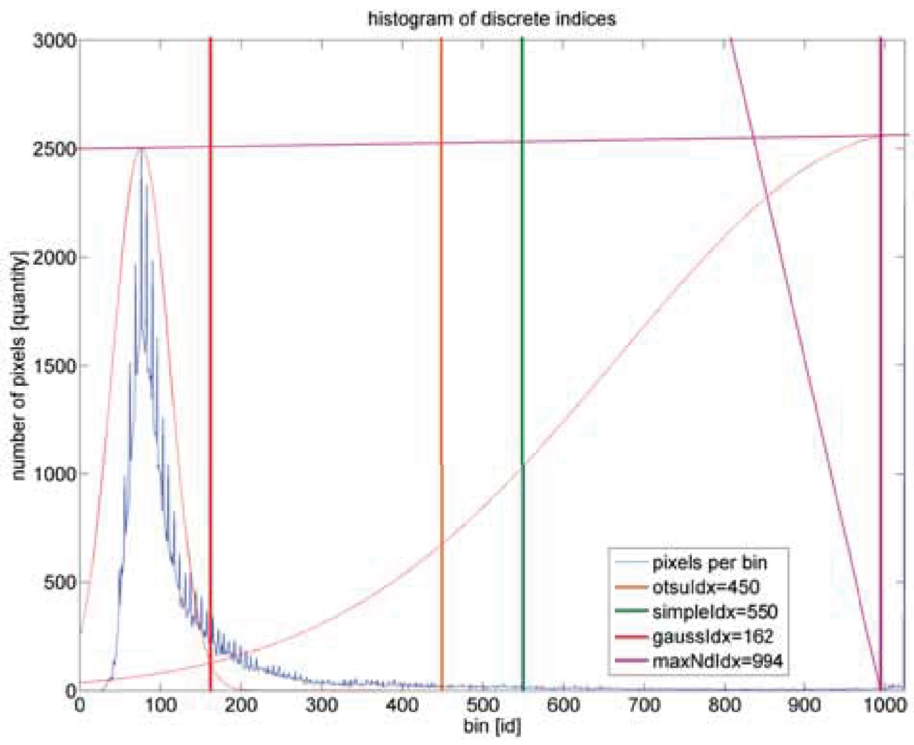


Figure 4.4: Each pixel is classified into one discrete bin by its intensity.

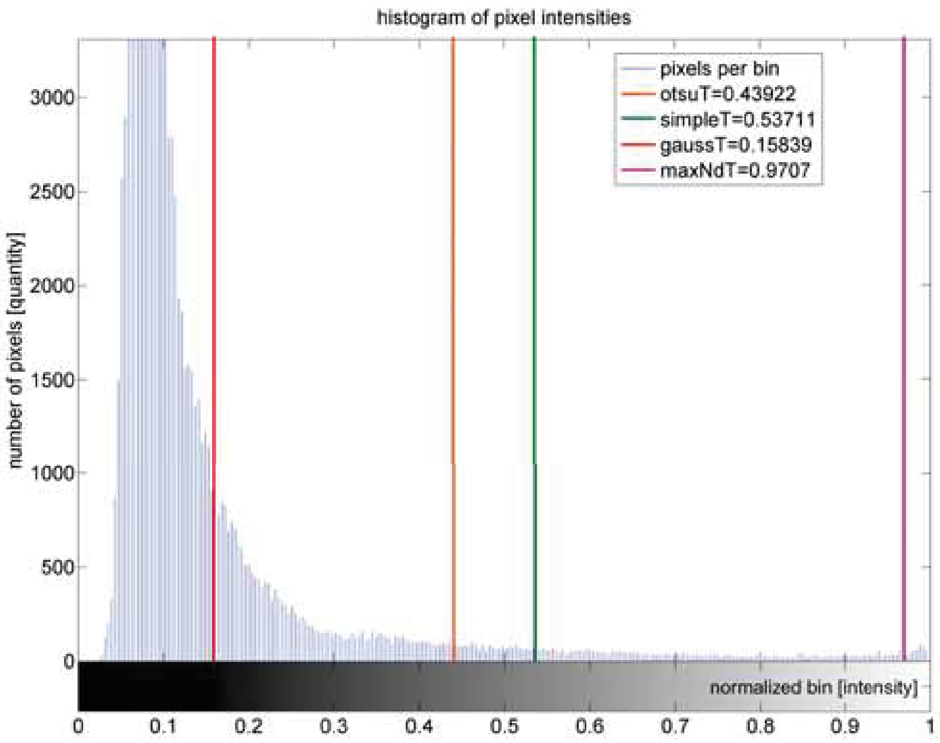


Figure 4.5: The threshold's bin index is normalized to fit a value between 0 and 1.

Note, the results of the methods differ significantly. As a consequence, the image object's edges are clearly shifted and with them their defining center coordinates. The binarizing method must fit the problem or the following processes are based on bad quality data. This is a challenging task, because it's not possible to say generally which one is the best. The Matlab implementation of the algorithms can be found in Appendix B.

4.1.1 Otsu's Method

Otsu's method is based on the idea, that there are basically two groups of pixels with different value ranges of their intensities, i.e. the result is a *bivariate histogram*. The primary difficulty is, that these ranges are overlapping and that only the histogram for the combined regions is available. The goal is to minimize the error of classifying a background pixel as a foreground one or vice versa. The method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels of each side of the threshold, i.e. the pixels that either fall in foreground or background. The threshold minimizing the combined spread should be determined [25].

The method defines two classes of pixels: the brighter (white) ones w and the darker (black) ones b. The exhaustive search for the *Otsu threshold* t_o minimizes the intra-class variance, which is defined as a weighted sum of variances of the two classes:

$$\sigma_w^2(t_o) = \omega_1(t_o)\sigma_1^2(t_o) + \omega_2(t_o)\sigma_2^2(t_o) \leftarrow \min, \tag{4.5}$$

whereby ω is the probability and σ^2 is the variance of the investigated class. In fact, it's the same as maximizing the inter-class variance between white σ_w^2 and black σ_b^2 pixels:

$$\sigma_b^2(t_o) = \sigma^2 - \sigma_w^2(t_o) \leftarrow \max\,. \tag{4.6}$$

The iterative updates of the histogram and the probabilities of each intensity level yields an effective algorithm, whereby the desired threshold corresponds to the found maximum value. Matlab implements this functionality in the *graythresh()* function.

```
t = graythresh(D); % t ... threshold
B = im2bw(D,t); % binarizing using t
```

Source Code 4.1: Binarizing an image with Matlab.

The following approaches are also based on the idea of two classes w and b in a bivariate histogram. Hence, Otsu's method is utilized as an effective way to acquire them.

4.1.2 Brightness Class Average

The simplest algorithm is to determine the intensities μ_w and μ_b associated with the two peaks of a bivariate histogram p_w and p_b, i.e. the brighter and darker pixels, to compute the *simple threshold* t_s:

$$t_s = \frac{\mu_w + \mu_b}{2}. \tag{4.7}$$

The peaks are acquired by finding the maximum values of the two classes delivered by Otsu's method. The simple threshold is suitable for high quality images with balanced bimodal histograms.

4.1.3 Gaussian Curve Intersection

Given a bivariate histogram, it's a reasonable idea to consider the intersection point of two Gaussian bell curves, modeled by the two intensity classes w and b, as a good threshold value, because the data of each class is normally distributed. The curves are described by the equations

$$y_w(a) = p_w \cdot \mathrm{e}^{-\frac{1}{2}\left(\frac{a-\mu_w}{\sigma_w}\right)^2} \qquad \text{and} \qquad y_b(a) = p_b \cdot \mathrm{e}^{-\frac{1}{2}\left(\frac{a-\mu_b}{\sigma_b}\right)^2}, \tag{4.8}$$

when a is the index of the investigated bin. The curve is evaluated by the mean value μ, which is in fact the index of the associated peak p_w or p_b, and the standard deviation σ of the particular class. Consequently, the intersection point can be found by solving the equation $y_w(a) = y_b(a)$ for a:

$$\ln p_w - \left(\frac{a-\mu_w}{\sigma_w}\right)^2 = \ln p_b - \left(\frac{a-\mu_b}{\sigma_b}\right)^2. \tag{4.9}$$

This is a simple quadratic equation, the valid solution lies between the two intensity peaks p_w and p_b. The calculated value represents the *Gauss threshold* t_g.

4.1.4 Maximal Normal Distance

The described algorithm works for images with bad illumination conditions, i.e. the bivariate nature is not clearly present. This leads to strongly asymmetric histograms. Like the other methods, it's based on two brightness classes. The peaks are connected by a line, the bin with the *maximal normal distance* is equivalent to the sought *threshold* t_n. Note, that in Figure 4.4 the two magenta lines are orthogonal to each other, the axes are not equalized.

4.2 Contour Finding

The monochromatic image B is used to find the image objects R_i, because of the higher information density in comparison to the grayscale image D. When the location of each R_i is acquired, the further computations are carried out for D. This is no problem, because B and D are the same size and also contain the same information about the objects. Therefore, the location of a R_i is the same in both images. Each separated region $\mathsf{R}_i \in \mathsf{D}$ is equivalent to its contour. The algorithm detects this *bright-dark transfers*, whereby it is triggered by a threshold t. An isolated object generates a point cloud on the edges around it. By connecting them in the right order, the defining contour is acquired.

This issue is not trivial, because the order of the points is not always clear without any doubt. A common solution is the introduction of a *neighborhood* to the investigated point. A contour point on the edge of an object is then connected to the next point on this edge, as long as it is a neighbor of the original point. This procedure must iteratively be performed clock-wise or counter-clock-wise, but the chosen direction must be kept constant. The algorithm is complete, when the first point is reached again and a contour around an object is acquired. If the first point cannot be reached, then it's an open contour and not an object of interest.

Matlab's function *contour()* delivers a vector of points $\boldsymbol{c}$. The points are settled around the image object's edge and define its contour.

```
c = contour(D,t); % t ... threshold vector, c ... contour points
```

Source Code 4.2: Finding an image object's contour with Matlab.

The threshold vector $\boldsymbol{t}$ has the form $\boldsymbol{t} = [t_1, t_2]$. If $t_1 \neq t_2$, then Matlab delivers 10 contours using 10 evenly distributed thresholds between the two values. If $t_1 = t_2$, then Matlab delivers only one contour in $\boldsymbol{c}$. Still, this effect can be exploited to generate as much contours as needed, e.g. $\boldsymbol{t}_1 = [t_1, t_1], \ldots, \boldsymbol{t}_n = [t_n, t_n]$. Figure 4.6 illustrates the outcome for a filtered and an unfiltered image.

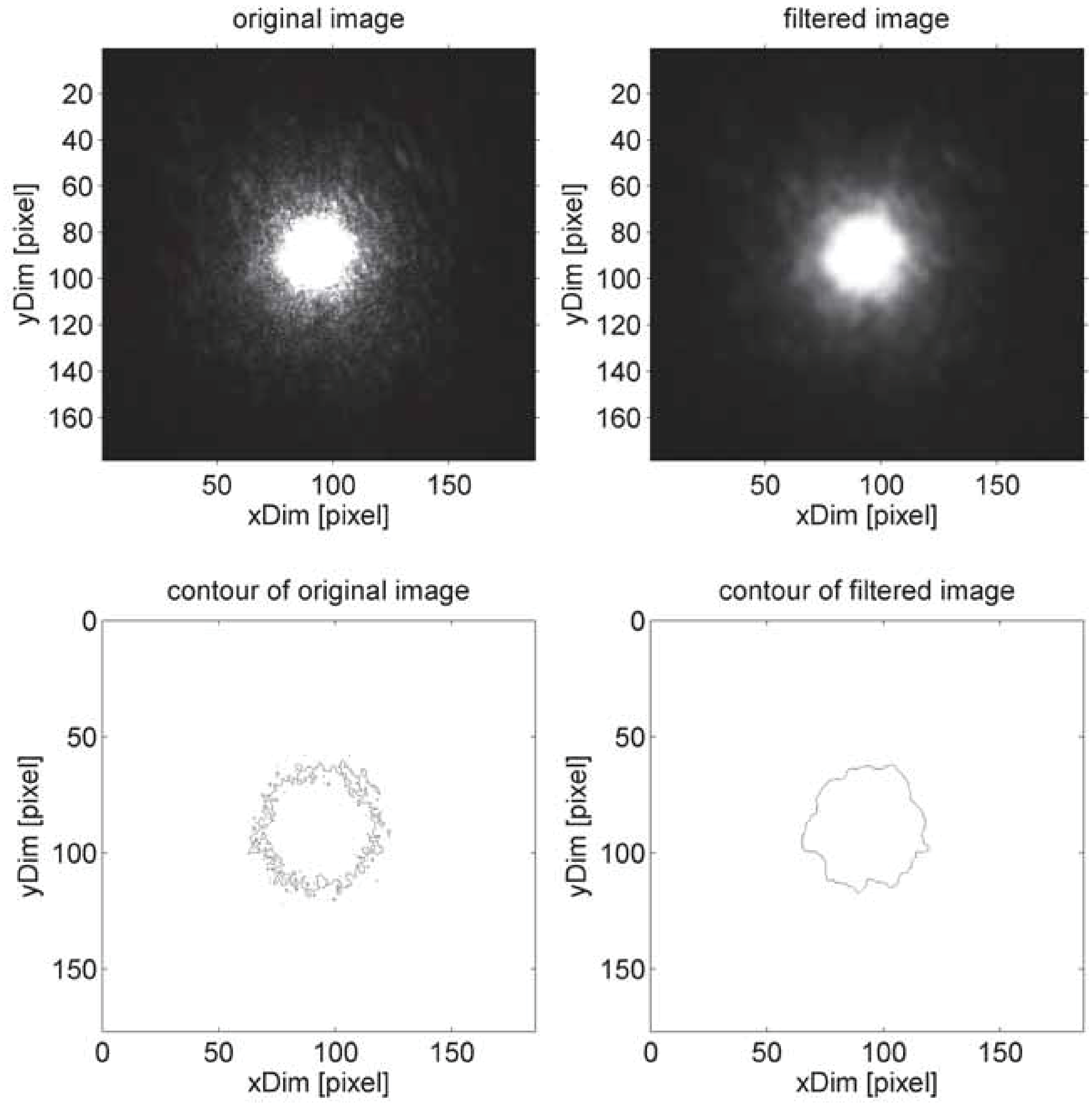


Figure 4.6: The ROI R shows one laser spot, but the contour plot reveals, that there exist numerous tiny objects around the main object. Filtering solves the problem by reducing the noise and removing the brightness peaks, which are causing the errors. The example uses a filter mask $f_m = 5$ and an Otsu threshold vector $\boldsymbol{t} = [t_o, t_o]$.

4.3 Region of Interest Classification

The last step of the segmentation is the classification of image objects $A^* = \{\mathsf{R}_1, \ldots, \mathsf{R}_m\}$ as regions of interest $A = \{\mathsf{R}_1, \ldots, \mathsf{R}_n\}$. Not every found feature is relevant for the measurement task. As a consequence, each image object R_i is checked for its geometric properties, which are:

- the minimum (x_{min}, y_{min}) and
- the maximum (x_{max}, y_{max}) object size as well as
- the length to width ratio (r_{min}, r_{max}, whereby $r = \frac{x}{y}$)

in respect of its x and y dimensions. The Matlab code is:

```
% check each criterium
xDimValid = (xDim ≥ xDimMin) && (xDim ≤ xDimMax);
yDimValid = (yDim ≥ yDimMin) && (yDim ≤ yDimMax);
xyPropValid = (xDim/yDim ≥ xyPropMin) && (xDim/yDim ≤ xyPropMax);
% check if image object is a region of interest
roiValid = (xDimValid && yDimValid && xyPropValid);
```

Source Code 4.3: Image object classification for ROI determination.

The parameters are based on a-priori knowledge. Following settings to deliver reliable results under laboratory conditions:

$$x_{min} = y_{min} \approx 10pixel,$$

$$x_{max} = y_{max} \approx 100pixel,$$

$$r_{min} \approx 0.5,$$

$$r_{max} \approx 2.0.$$

These values should only give an impression of their dimensions, they can vary by the factor 2 because of lighting conditions and camera settings. The best idea is to set them up during the calibration procedure. Figure 4.7 shows an example of an image object, which is not a region of interest. It's rejected as it does not meet the described criteria.

The classification for the example in Figure 4.7 works this way:

$$A^* = \{\mathsf{R}_1, \mathsf{R}_2\} \xrightarrow{\text{classification}} A = \{\mathsf{R}_1\}. \tag{4.10}$$

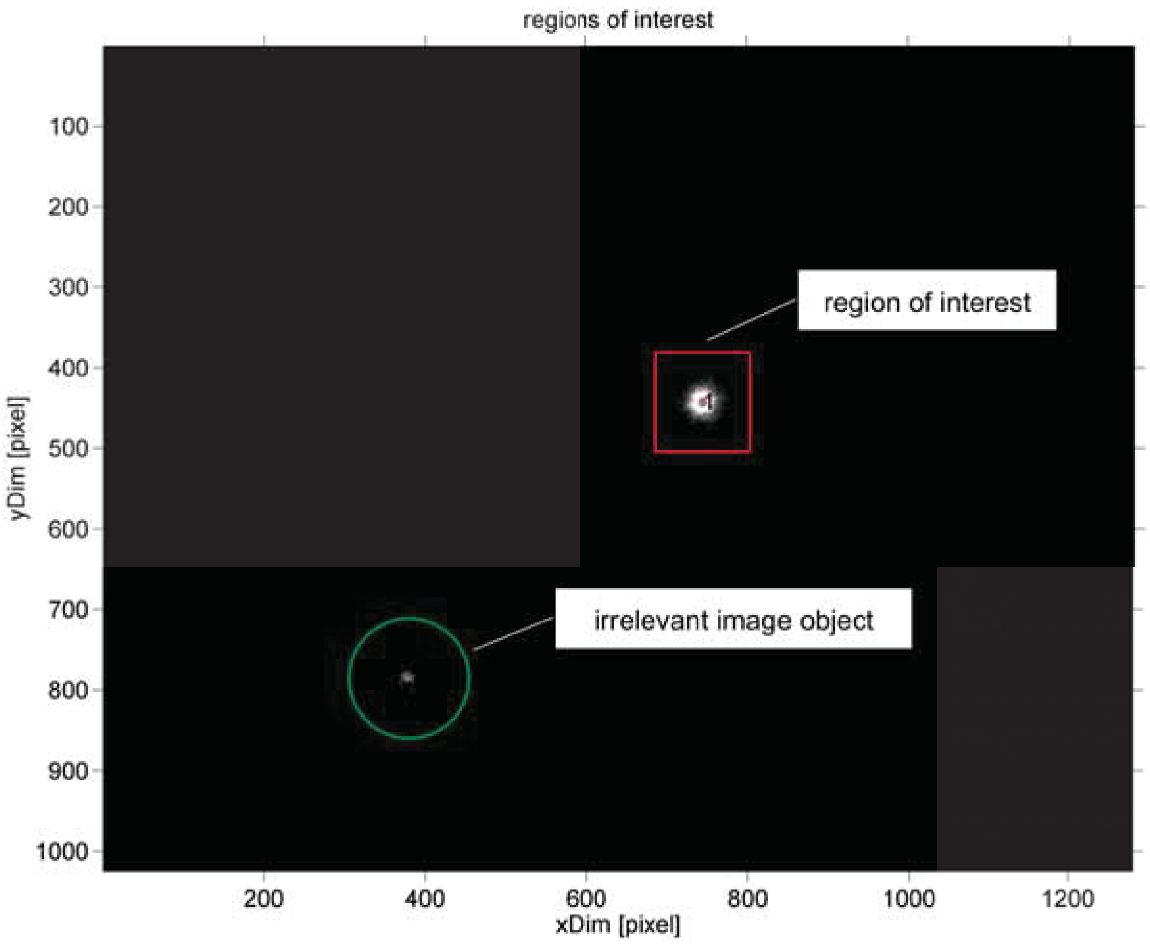


Figure 4.7: This is an image of the target plate. The laser spot R_1 is marked with the red surrounding rectangle, which also indicates the confidence range added to this image object. The object fulfills the described criteria by having valid dimensions and proportions. Therefore, it is a region of interest. However, there's also a disturbing object R_2 caused by the installed interference filter, marked with a green circle. It's recognized as an image object by the binary segmentation, but thanks to the classification this irrelevant feature is not used.

Each relevant object in A gets an additional security margin of varying size added to it's dimensions. The reasons are: to ensure to don't miss relevant data at the border of the object and it's necessary for certain center determination methods. In fact, each R_i is a cut out window of the image D, the binary image B was just used to find its location.

Chapter 5

Center Determination

This chapter investigates a collection of different techniques to determine the *center point* $\boldsymbol{p}$ of an image object R, because this feature is its defining attribute. The matrix R is used equivalently to the general image object $\mathsf{R}_i \in \mathsf{D}$ for easier reading of the following section:

$$\mathsf{R} \xrightarrow{\text{determination}} \boldsymbol{p} = [x_c, y_c]^T. \tag{5.1}$$

In the application, the region of interest R contains a laser spot, which is projected on a metal or glass target plate. Consequently, its appearance is determined by the reflection from the rough or glossy surface, the constructive and destructive interferences of coherent light and the blooming effect of the camera sensor. All these effects cause the image curve to be spread over several pixels with varying width in a direction orthogonal to the laser beam. Furthermore, the intensity values along the spread are distributed in the neighborhood around the sought center of the object. The explanation states, that the algorithms must be capable of handling these influencing factors.

5.1 Center of Gravity

The centroid, more commonly called the center of gravity (COG), is considered to be the equilibrium point at which the entire mass of an object is concentrated. In image processing, the mass of the object is replaced with the pixels' intensities. As a consequence, the center coordinates $\boldsymbol{p}$ of any given geometrical object can easily be acquired. Equation 5.2 calculates the centroid, whereby c_{yx} is the intensity of the pixel at the corresponding y and x position, i.e. i^{th} row and j^{th} column, k is a power factor to emphasize bright pixels in a grayscale image and R is the investigated area [20, 17]:

$$\boldsymbol{p} = [x_c, y_c]^T = \left[\frac{\sum_{y,x \in \mathsf{R}} x \cdot c_{yx}^k}{\sum_{y,x \in \mathsf{R}} c_{yx}^k}, \frac{\sum_{y,x \in \mathsf{R}} y \cdot c_{yx}^k}{\sum_{y,x \in \mathsf{R}} c_{yx}^k} \right]^T . \tag{5.2}$$

The calculation's precision is influenced by involving pixels, which are not part of the actual object. The binary image in Figure 5.2 clearly contains such pixels. The intensity values of all pixels inside R are evaluated by a threshold t, which is used to separate the object's pixels from the background and thereby determines which ones should be used for the computation. To conclude, a small threshold leads to a biased object center, because more pixels enter the calculation, a high value leads to lower precision. Only a proper threshold value t and an adequate method to compute it will effectively separate the object from its background and surrounding noise. Note, that for this algorithm, the values for the x dimension are stored in the rows and the values for y are stored in the columns.

```matlab
% matrix with each pixel set to its coordinate values
[rows,cols] = size(D);
x = ones(rows,1)*(1:cols);
y = (1:rows)'*ones(1,cols);
% calculating the centroid
area = sum(sum(D));
xCenter = sum(sum(D.^p).*x))/area;
yCenter = sum(sum(D.^p).*y))/area;
```

Source Code 5.1: Computation of an area's center of gravity with Matlab.

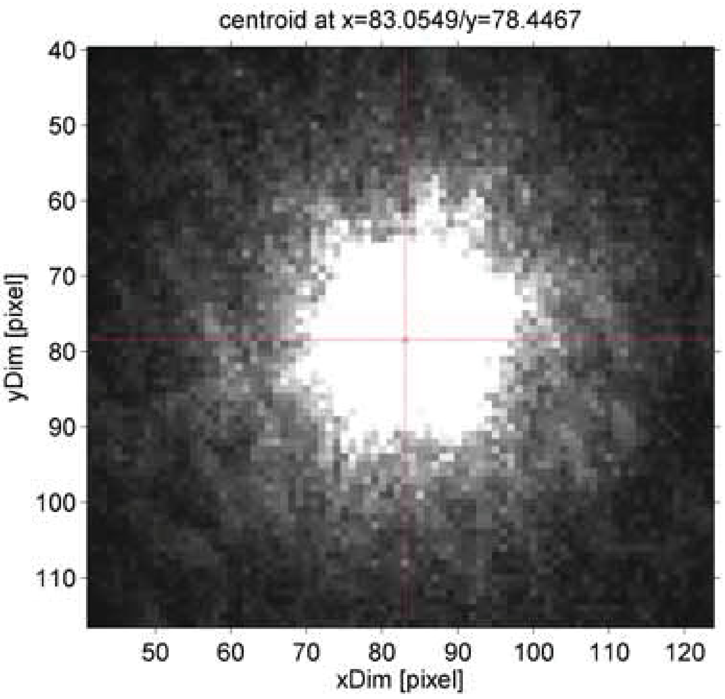


Figure 5.1: The grayscale image's pixels have double values ranging from 0 to 1. As a result, each pixel's intensity in R acts as a weighting factor for the calculation. In the example, the emphasizing factor is set to neutral, i.e. $k = 1$.

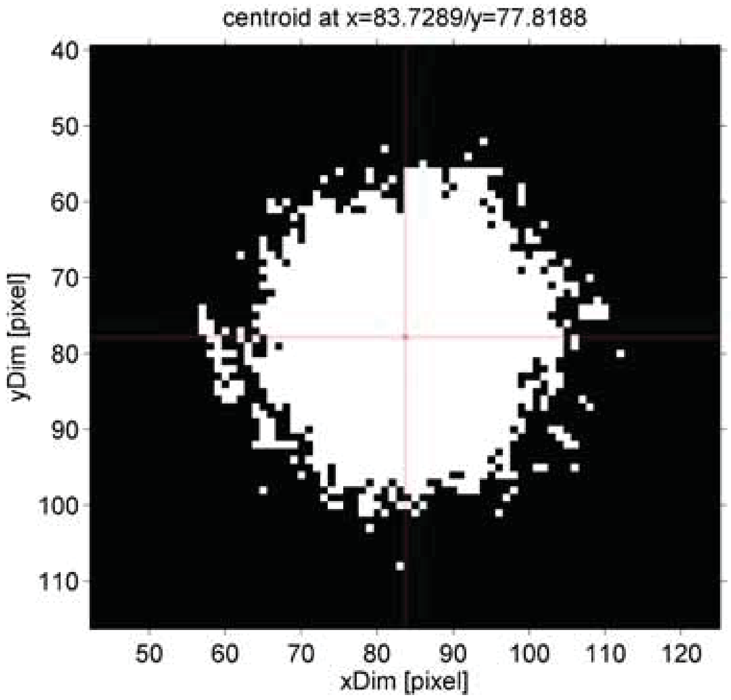


Figure 5.2: The image R has been binarized by using an Otsu threshold t_o and therefore the former gradient has been eliminated. The pixel intensities are now 0 or 1. As a consequence, the resulting center points are different.

5.2 Gaussian Curve Extremum

The method uses the idea of determining a Gaussian bell curve's extremum to obtain the center point of an uniform geometric figure along its main axes. A laser spot approximately represents a circle on the target plate in R. Therefore, it can be examined from every angle. Figure 5.3 illustrates the concept: the mean intensity of each column is plotted on the x axis and the mean intensities of each row is plotted on the y axis. The resulting curves of mean intensities are smoothed by a moving average method.

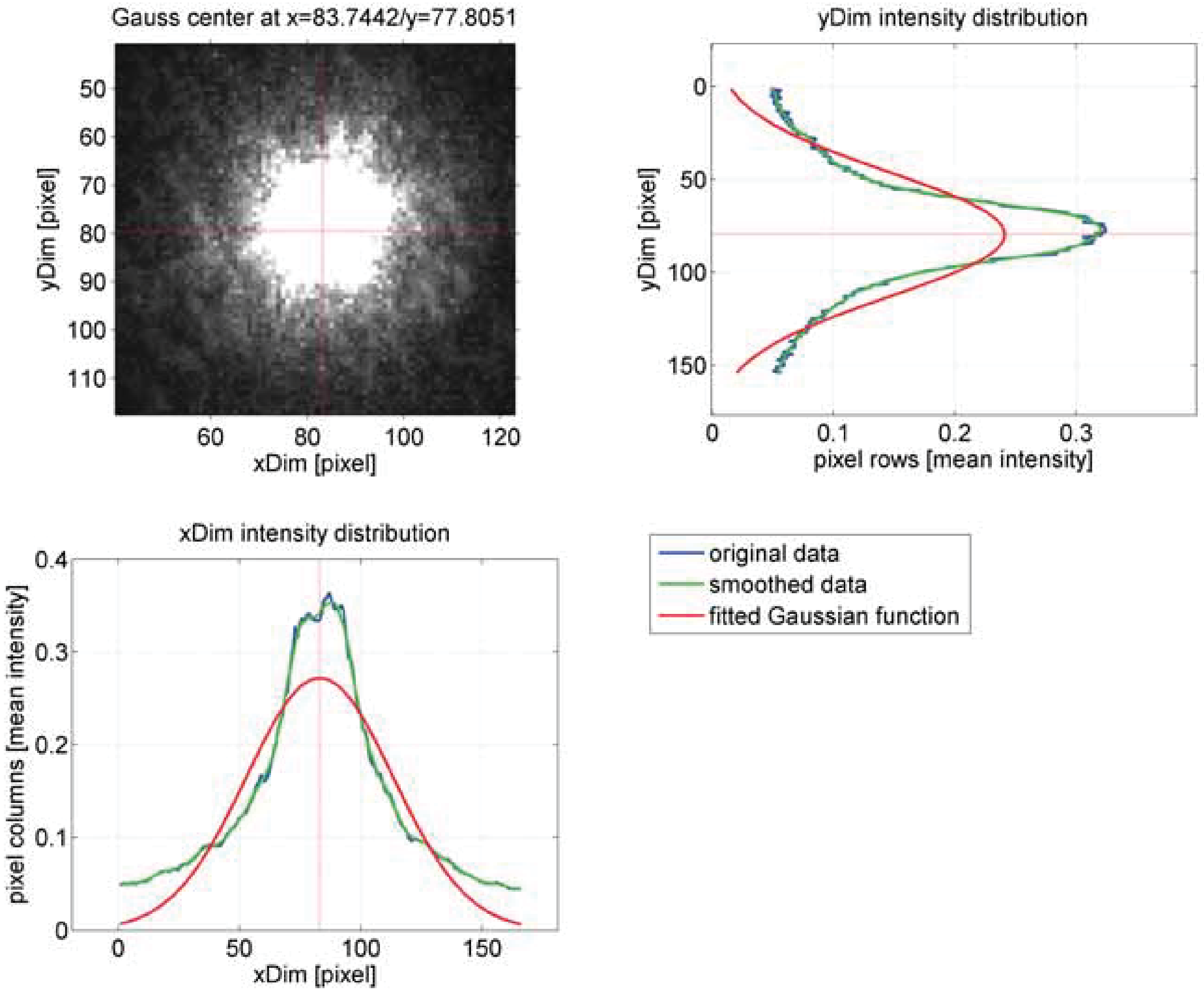


Figure 5.3: The mean intensities of R's columns and rows are plotted in respect of the x and y axis to obtain a set of data points. The data points are smoothed and fitted by a least squares polynomial. The peak values of the resulting Gaussian curves are equivalent to the laser spot's center coordinates $\boldsymbol{p}$ in R.

A least squares polynomial fit is performed onto the smoothed data points $\boldsymbol{a}$ to acquire the bell curve's characteristics, which are the mean μ, the standard deviation σ and a scale factor k:

$$f(a_i) = k \cdot \mathrm{e}^{-\frac{1}{2}\left(\frac{a_i-\mu}{\sigma}\right)^2}. \tag{5.3}$$

The peak values, i.e. the extremum of each curve, are equivalent to the center points of the corresponding axes of R. Combining them yields the coordinates of the center point $\boldsymbol{p}$. Note, like the center of gravity method, this approach uses the unfiltered image.

5.3 Algebraic Conic Fitting

A laser spot's contour can be modeled as a cloud of scattered data points. Fitting a *closed planar second order implicit curve*, i.e. a circle or an ellipse, is a good idea to determine such an object's center coordinates. However, a projected circle yields a distorted ellipse and as a result, the center point is not invariant, Figure 5.4 illustrated the effect.

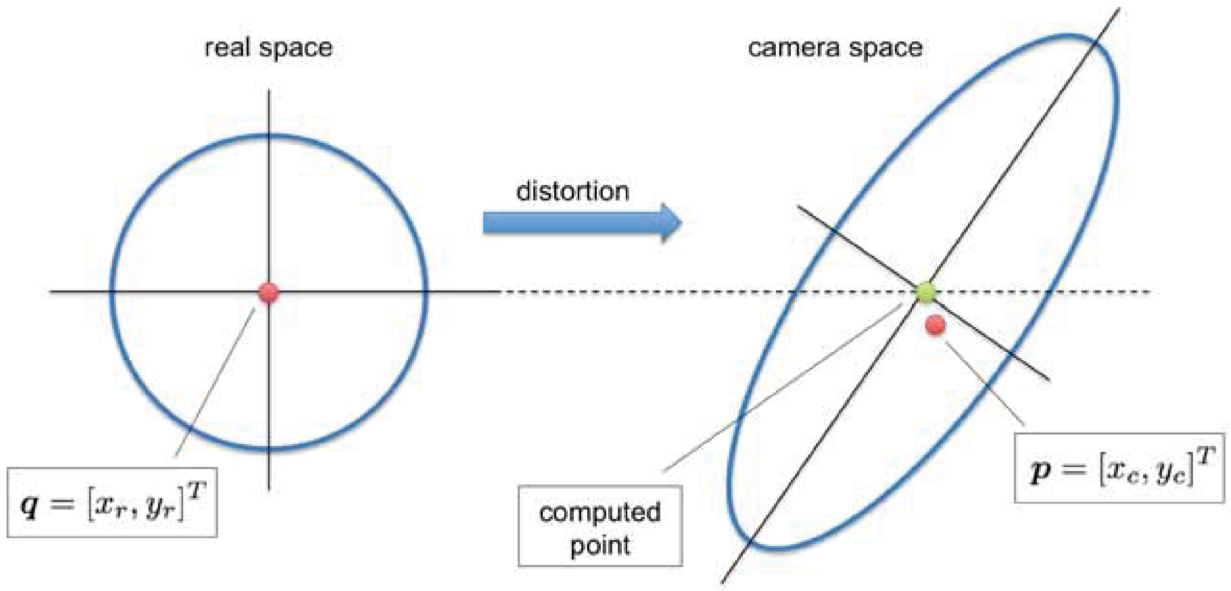

real space
camera space
distortion
$\boldsymbol{q} = [x_r, y_r]^T$
computed point
$\boldsymbol{p} = [x_c, y_c]^T$

Figure 5.4: The schematic visualizes the non-invariant center point. In real space, the coordinates of $\boldsymbol{q}$ describe exactly the circle's center. However, in camera space, the equivalent $\boldsymbol{p}$ is not located in the center of the ellipse. The computed point is slightly different than the actual point $\boldsymbol{p}$, because of the camera objective's distortion.

The algebraic method introduced in this section is based on linear least mean squares fitting, whereby the approximated conic's curve is compared to the contour points by calculating the resulting residuals. This approach can be considered as one kind of template matching, as a defined geometric figure is fitted to a given data set. A detailed introduction to the Euclidean classification of conics can be found in literature [15, 12].

A generic conic is represented by its concise matrix form:

$$\boldsymbol{u}^T \mathsf{C} \boldsymbol{u} = \begin{bmatrix} x & y & w \end{bmatrix} \begin{bmatrix} c_1 & \frac{c_2}{2} & \frac{c_4}{2} \\ \frac{c_2}{2} & c_3 & \frac{c_5}{2} \\ \frac{c_4}{2} & \frac{c_5}{2} & c_6 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix} = 0. \tag{5.4}$$

The *homogeneous coordinates* of a point on the conic are given by the vector $\boldsymbol{u} = [x, y, w]^T$, whereby the conic's center is $\boldsymbol{u}_c = [x_c, y_c, w_c]^T$. Consider, that homogeneous and affine (or inhomogeneous) coordinates of a point are related as following:

$$\underbrace{\boldsymbol{u} = \begin{bmatrix} x \\ y \\ w \end{bmatrix}}_{\text{homogeneous}} \quad \text{and} \quad \underbrace{\boldsymbol{p} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}}_{\text{affine}}. \tag{5.5}$$

Euclidean invariants, that determine the conic type, are embedded inside the symmetric coefficient matrix C. Every point $\boldsymbol{u}_i = [x_i, y_i, w_i]^T$ imposes a constraint on the coefficients of the conic. Because the multiplication of Equation 5.4 by a non-zero constant is of no consequence, five known points are enough do describe a generic conic. Consequently, expanding the equation with $w = 1$, yields an implicit polynomial in x and y, which can be written as the product of two vectors $\boldsymbol{n}$ and $\boldsymbol{c}$:

$$\boldsymbol{n}^T \boldsymbol{c} = \begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = 0, \tag{5.6}$$

where the vector $\boldsymbol{c}$ contains the conic coefficients and $\boldsymbol{n}$ carries the point information. For fitting a conic with n given points, this results in n equations for the algebraic distance from each point $\boldsymbol{u}_i$ to the approximated conic. The set of equations provides the design matrix N for the fitting procedure.

$$\mathsf{N}\boldsymbol{c} = \begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & x_ny_n & y_n^2 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} = \boldsymbol{e}, \tag{5.7}$$

whereby the e_i are the algebraic distances of each point to the conic, i.e. the resulting error vector $\boldsymbol{e}$. The last column of N contains only ones, which are statistically invariant with respect to the data points. For numerical stability, the calculation of the total least squares requires this column to be removed to avoid the unusable result of a hyperplane. The decomposition is called *orthogonal residualization* [24]. The goal is the partitioning of N into

$$\mathsf{N} = [\mathsf{N}_1\mathsf{N}_0] = \left[\begin{array}{ccccc|c} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 & x_ny_n & y_n^2 & x_n & y_n & 1 \end{array}\right], \tag{5.8}$$

from which the new design matrix $\hat{\mathsf{N}}$ is sought. It is the *orthogonal residual* of N_1 in respect of N_0:

$$\hat{\mathsf{N}} = \mathsf{N}_1 - \mathsf{N}_0\mathsf{N}_0^+\mathsf{N}_1, \tag{5.9}$$

whereby the *Moore-Penrose pseudo inverse* N_0^+ is defined by:

$$\mathsf{N}_0^+ \triangleq (\mathsf{N}_0^T\mathsf{N}_0)^{-1}\mathsf{N}_0^T, \qquad \text{for a vector containing } n \text{ ones:} \qquad \mathsf{N}_0^+ = \frac{1}{n}\mathsf{N}_0^T. \tag{5.10}$$

Back-substitution into Equation 5.9 yields

$$\hat{\mathsf{N}} = \mathsf{N}_1 - \mathsf{N}_0\frac{1}{n}\mathsf{N}_0^T\mathsf{N}_1, \tag{5.11}$$

which is equivalent to the removal of the mean value from each column of matrix N_1:

$$\hat{\mathsf{N}}\boldsymbol{c}^* = \begin{bmatrix} x_1^2 - \overline{x^2} & x_1y_1 - \overline{xy} & y_1^2 - \overline{y^2} & x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^2 - \overline{x^2} & x_ny_n - \overline{xy} & y_n^2 - \overline{y^2} & x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} = \boldsymbol{e}. \tag{5.12}$$

As a consequence, the cost function K, which should be minimized, is

$$K = \boldsymbol{e}^T \boldsymbol{e} = \sum_{i=1}^{n} e_i^2 \leftarrow \min, \tag{5.13}$$

i.e. the sum of squared normal distances e_i. Figure 5.5 illustrates the idea.

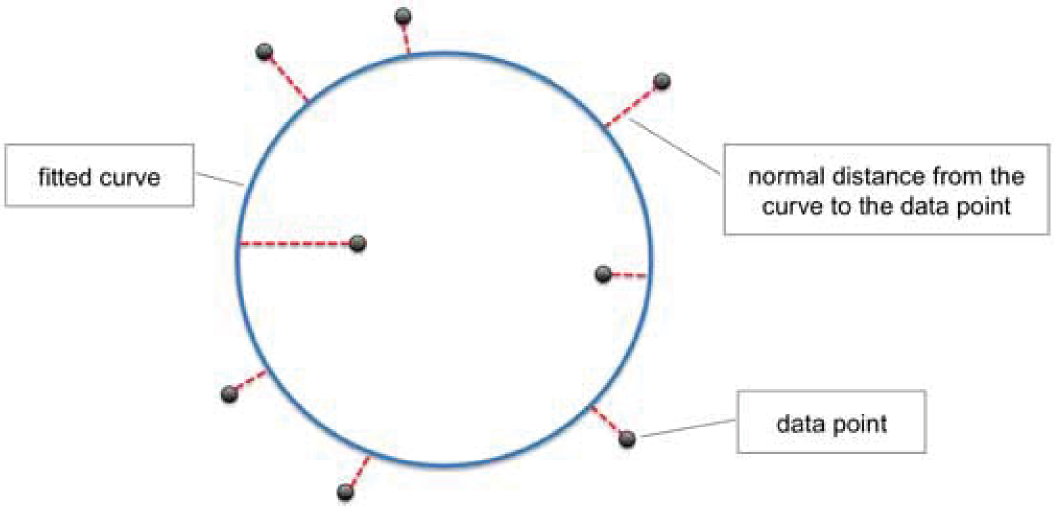


Figure 5.5: The schematic illustrates the usability of an algebraic least squares approach for conic fitting. Each data point $\boldsymbol{u}_i$ (equivalent to $\boldsymbol{p}_i$) has a normal distance e_i to the curve, which represents the error. By minimizing the cost function K, i.e. the sum of quadratic errors, the best fitting conic, e.g. a circle or an ellipse, is computed.

The minimization is performed by applying *singular value decomposition* (SVD) to the new design matrix $\hat{\mathsf{N}} \xrightarrow{\text{SVD}} [\mathsf{U}, \mathsf{S}, \mathsf{V}]$, whereby $\hat{\mathsf{N}} = \mathsf{U}\mathsf{S}\mathsf{V}^T$. The right singular column vector $\boldsymbol{v}^*$ of V corresponds to the smallest singular vector value in S. The SVD ensures the constraint $||\boldsymbol{v}^*|| = 1$ to obtain a non-trivial solution for the vector $\boldsymbol{v}^*$. If $\hat{\mathsf{N}}$ is a $(m \times n)$ matrix, then U is a $(m \times m)$ and V is a $(n \times n)$ unitary, i.e. normalized and orthogonal, matrix. Hence, the magnitude information is contained in S. Moreover, the constraint imposes that the residuals e_i are the normal distances of the data points to the fitted plane.[15]

The thesis focuses on circular and elliptical solutions, because these two geometrical figures deliver the best quality of fit for an imaged laser spot. To enable an efficient computation inside a region of interest R, the pixel coordinates are first normalized. To fit an adequate conic, noise suppression is necessary to avoid unwanted contours from sparkles.

5.3.1 Ellipse Approximation

The well known equation of a normal orientated ellipse in the plane is given by

$$\frac{(x_i - x_c)^2}{a^2} + \frac{(y_i - y_c)^2}{b^2} = 1, \tag{5.14}$$

where x_i, y_i are the Euclidean coordinates of a point lying on the ellipse with the center point $\boldsymbol{p}_c = [x_c, y_c]^T$ and the form factors a and b. Thus, by expanding Equation 5.4, the scalar point equation of a conic, which also describes an ellipse in its homogeneous form, is derived:

$$c_1 x_i^2 + c_2 x_i y_i + c_3 y_i^2 + c_4 x_i w_i + c_5 y_i w_i + c_6 w_i^2 = 0. \tag{5.15}$$

Considering, that the measurement data is scattered, the algebraic distance e_i of a point $\boldsymbol{u}_i = [x_i, y_i, w_i]^T$ to the ellipse is

$$c_1 x_i^2 + c_2 x_i y_i + c_3 y_i^2 + c_4 x_i w_i + c_5 y_i w_i + c_6 w_i^2 = e_i. \tag{5.16}$$

The elliptical parameters are embedded inside the conic coefficient matrix C, which are the homogeneous coordinates of the center point $\boldsymbol{u}_c = [x_c, y_c, w_c]^T$, the direction through the eigenvectors $\boldsymbol{i}_1$, $\boldsymbol{i}_2$ and the lengths l_1, l_2 of the major and minor principal axes. The homogeneous coordinate w_i acts as a scaling factor and it's best to set $w_i = 1$, leading to:

$$c_1 x_i^2 + c_2 x_i y_i + c_3 y_i^2 + c_4 x_i + c_5 y_i + c_6 = 0, \tag{5.17}$$

which is equivalent to Equation 5.6. Consequently, five given points fully define an ellipse. This means, if $n \geq 6$ data points, then a system of linear equations is derived like in Equation 5.7.

The formulas explained so far are completely general for conics. The Euclidean invariants describing an ellipse are depending on the coefficient matrix C, which are

$$D = \det \mathsf{C} \quad \text{and} \tag{5.18}$$

$$\Delta = \begin{vmatrix} c_1 & \frac{c_2}{2} \\ \frac{c_2}{2} & c_3 \end{vmatrix} = c_1 c_3 - \frac{c_2^2}{4}. \tag{5.19}$$

If $D \neq 0$ and $\Delta > 0$, then the coefficients of the matrix C describe an ellipse, if $D \neq 0$ and $\Delta < 0$ the shape of the conic is hyperbolic and finally, if $D \neq 0$ and $\Delta = 0$, it is parabola.

The affine center point can be derived by acquiring

$$\boldsymbol{u}_c = [x_c, y_c, w_c]^T \qquad \text{and converting it to} \qquad \boldsymbol{p}_c = \left[\frac{x_c}{w_c}, \frac{y_c}{w_c}\right]^T. \tag{5.20}$$

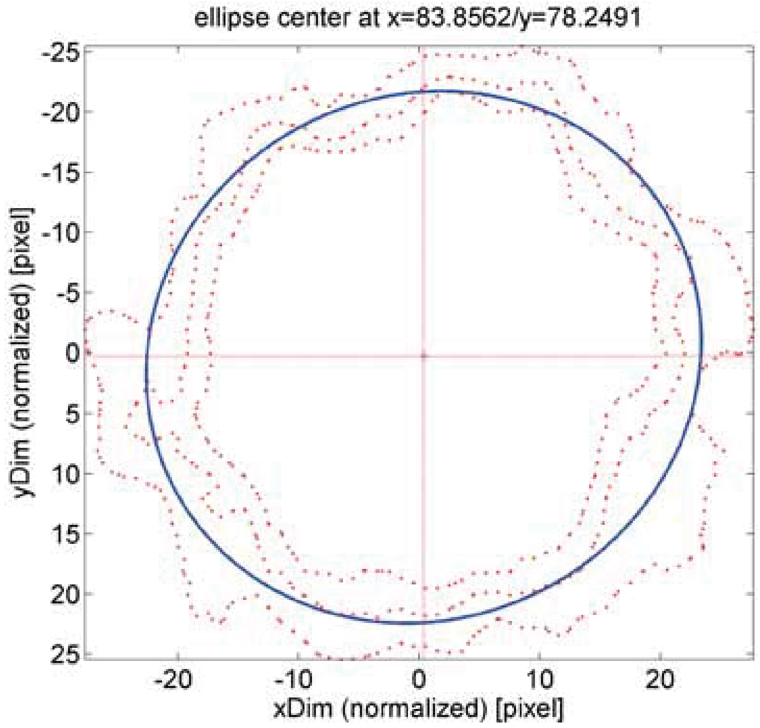


Figure 5.6: Compare the plot to the schematic in Figure 5.5. The red dots indicate the points $\boldsymbol{u}_i$, the distances to the fitted blue conic line are the errors e_i. The approximated ellipse represents the least squares solution. The example uses three contours with the Otsu threshold vectors $\boldsymbol{t}_1 = [0.9t_o, 0.9t_o]$, $\boldsymbol{t}_2 = [1.0t_o, 1.0t_o]$ and $\boldsymbol{t}_3 = [1.1t_o, 1.1t_o]$.

5.3.2 Circle Approximation

The equation of a circle in the plane is given by

$$(x_i - x_c)^2 + (y_i - y_c)^2 - r^2 = 0, \tag{5.21}$$

where x_i, y_i are the Euclidean coordinates of a point lying on the circle with the center point $\boldsymbol{p}_c = [x_c, y_c]^T$ and the radius r. A circle is a special kind of ellipse, because the axes have the same lengths $l_1 = l_2 = r$. Because of the constraints, it is possible to describe a circle with just three given points.

The explanation skips the use of homogeneous coordinates by directly utilizing *Grassmannian tetra-circular coordinates* on the above equation:

$$c_1(x_i^2 + y_i^2) + c_2 x_i + c_3 y_i + c_4 = 0. \tag{5.22}$$

As already mentioned, any point of the circle can be derived as a linear combination of three known points. Hence, the linear system of equations can be solved for the coefficients c_i by

$$\begin{vmatrix} (x^2 + y^2) & x & y & 1 \\ (x_1^2 + y_1^2) & x_1 & y_1 & 1 \\ (x_2^2 + y_2^2) & x_2 & y_2 & 1 \\ (x_3^2 + y_3^2) & x_3 & y_3 & 1 \end{vmatrix} = 0. \tag{5.23}$$

The linear system of equations is derived by Grassmannian expansion of the determinant. Taking into account, that the measurement data is scattered, the algebraic distance e_i of a point $\boldsymbol{p}_i$ to the circle is

$$c_1(x_i^2 + y_i^2) + c_2 x_i + c_3 y_i + c_4 = e_i. \tag{5.24}$$

Having $n \geq 4$ data points, to describe the circle, Equation 5.24 can be rewritten as

$$\begin{bmatrix} (x_1^2 + y_1^2) & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ (x_n^2 + y_n^2) & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}, \tag{5.25}$$

which is equal to its abbreviated form:

$$\mathsf{N}\boldsymbol{c} = \boldsymbol{e}, \tag{5.26}$$

whereby N is once again the design matrix. Analog to Equation 5.8, orthogonal residualization is performed:

$$\mathsf{N} = [\mathsf{N}_1\mathsf{N}_0] = \left[\begin{array}{ccc|c} (x_1^2 + y_1^2) & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ (x_n^2 + y_n^2) & x_n & y_n & 1 \end{array}\right], \tag{5.27}$$

hereby acquiring the mean-free form:

$$\hat{\mathsf{N}}\boldsymbol{c}^* = \begin{bmatrix} (x_1^2 + y_1^2) - \overline{(x^2 + y^2)} & x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots & \vdots \\ (x_n^2 + y_n^2) - \overline{(x^2 + y^2)} & x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} = \boldsymbol{e}. \tag{5.28}$$

The computation of the coefficients c_1, c_2 and c_3 is done through back-substitution [15]. The center of the circle $\boldsymbol{p}_c = [x_c, y_c]^T$ and the radius r are calculated the following way:

$$\boldsymbol{p}_c = \begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} -\frac{c_2}{2c_1} \\ -\frac{c_3}{2c_1} \end{bmatrix} \quad \text{and} \quad r = \sqrt{x_c^2 + y_c^2 - \frac{c_4}{c_1}}. \tag{5.29}$$

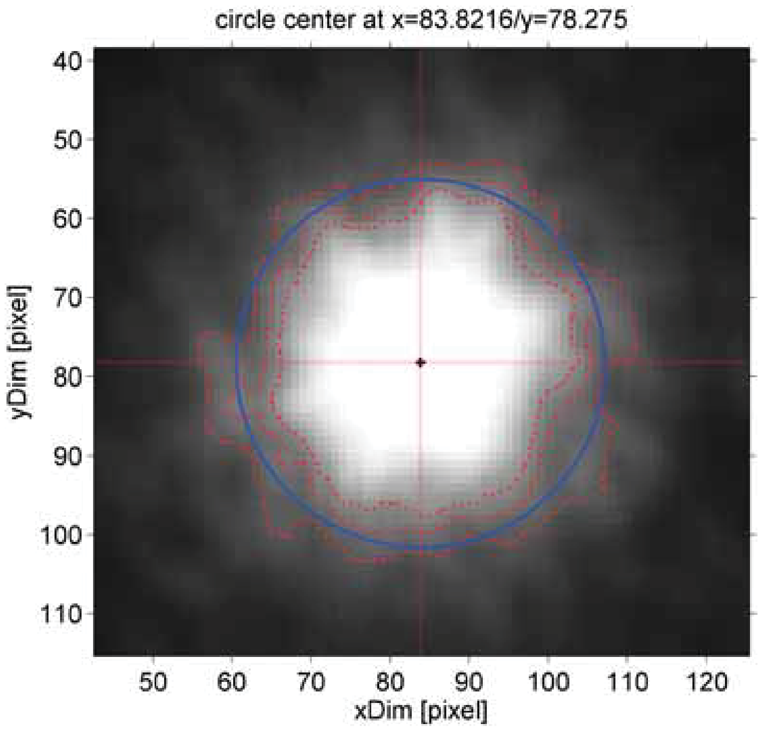


Figure 5.7: Compare the plot to the schematic in Figure 5.5. The red dots indicate the points $\boldsymbol{p}_i$, the distances to the fitted blue conic line are the errors e_i. The approximated circle represents the least squares solution. Once again, the example uses three contours with the Otsu threshold vectors $\boldsymbol{t}_1$, $\boldsymbol{t}_2$ and $\boldsymbol{t}_3$.

Chapter 6

Coordinate Mapping

Linear two-dimensional projections can be represented through a *projection matrix* M:

$$\mathsf{M} = \begin{bmatrix} a_{11} & a_{12} & d_x \\ a_{21} & a_{22} & d_y \\ g_x & g_y & 1 \end{bmatrix}, \tag{6.1}$$

whereby $\{a_{11}, a_{12}, a_{21}, a_{22}\}$ describe the translation, rotation, scaling and affine transformation by the amounts of d_x and d_y of a planar object. Additionally, g_x and g_y model the projective geometry. The last element of the matrix is a scalar and is set to 1, i.e. the matrix is normalized. This leads to a total number of 8 degrees of freedom for two-dimensional projections. The only remaining invariant is cross ratios [13]. Note, if $g_x = g_y = 0$, then it's an *affine transformation* and original parallel lines stay parallel after the projection. The schematic in Figure 6.1 gives an idea how a warped linear space, including its objects, can be transformed to the real space, which is basically an Euclidean plane with a Cartesian coordinate system. The ultimate goal of the procedure is the piecewise mapping of a camera space point $\boldsymbol{p} = [x_c, y_c]^T$ to its corresponding real space point $\boldsymbol{q} = [x_r, y_r]^T$, therefore

$$\boldsymbol{p} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} [\text{pixel}] \xrightarrow{\textit{mapping}} \boldsymbol{q} = \begin{bmatrix} x_r \\ y_r \end{bmatrix} [\text{mm}]. \tag{6.2}$$

The problem visualized in Figure 6.1 is a well explored branch of image processing, the commonly utilized method is *linear homography*. The issue gets more complex when adding non-linear effects to it, e.g. through optical distortions caused by a camera's objective. The illustration in Figure 6.2 shows such a *fish-eye effect*.

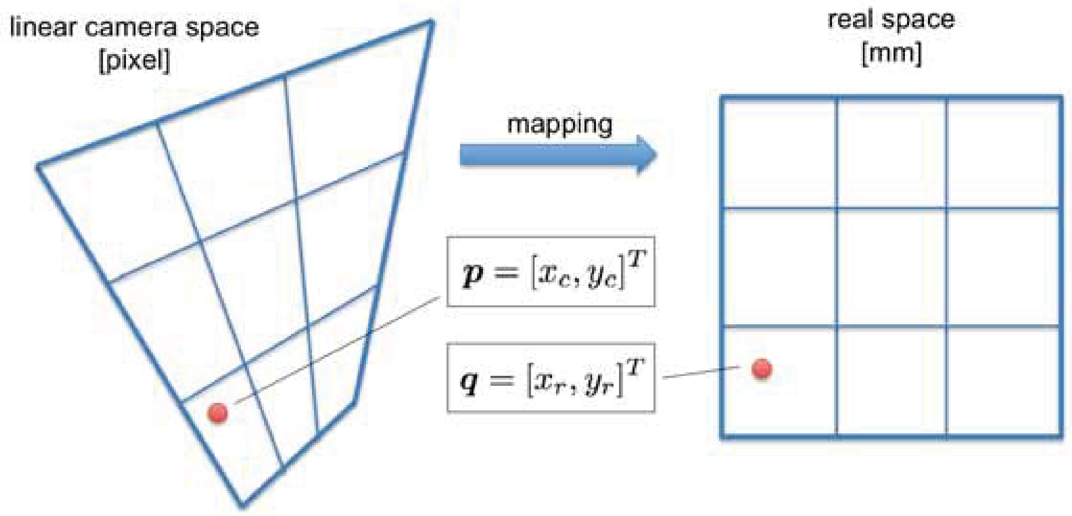


Figure 6.1: The schematic gives an idea how a point $\boldsymbol{p}$ in the camera space can be mapped to its real space pendant $\boldsymbol{q}$. The linear perspective effects are described via the projection matrix M, which yields 8 degrees of freedom. A solution can by acquired by the usage of linear homography.

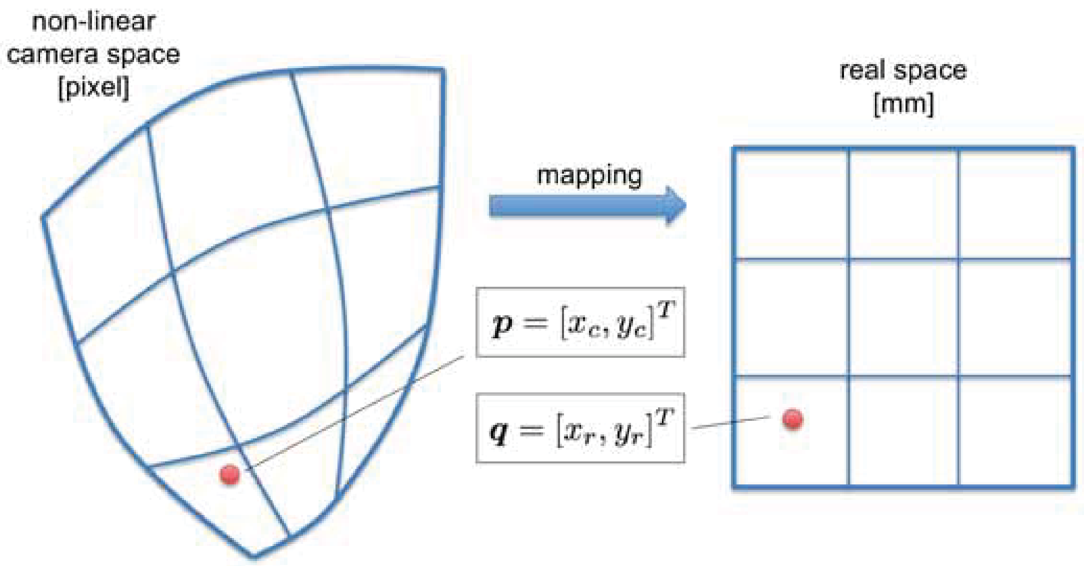


Figure 6.2: A non-linear camera space makes the mapping process more complex, as the utilized methods are also based on non-linear models. This thesis uses the idea of polynomial fitting and tensor interpolation.

The handling of such non-linear effects demands the introduction of more specialized methods. The thesis will investigate two ideas: *polynomial control points* as well as *tensor interpolation via basis function sectioning* and *quad tree decomposition.*

6.1 System Calibration

In this application, the calibration process generates a relation between the camera space and the metric real space. To determine corresponding points inside the images, a set of predefined and uniquely identifiable points, known as the *base points* or *control points*, are selected from each space. Once two sets of control points are chosen, techniques are available to match these two sets.

The first set Q is an array of vectors, i.e. a matrix, of *known real space coordinates* with n_r points:

$$\mathsf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n] \tag{6.3}$$

and the second set P is an array of vectors of *known camera space coordinates* with n_c points

$$\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]\,. \tag{6.4}$$

The points in P are acquired from the originally acquired grayscale image G. The calibration yields:

$$\mathsf{Q}\ \&\ \mathsf{P} \xrightarrow{\text{calibration}} \text{calibration data.} \tag{6.5}$$

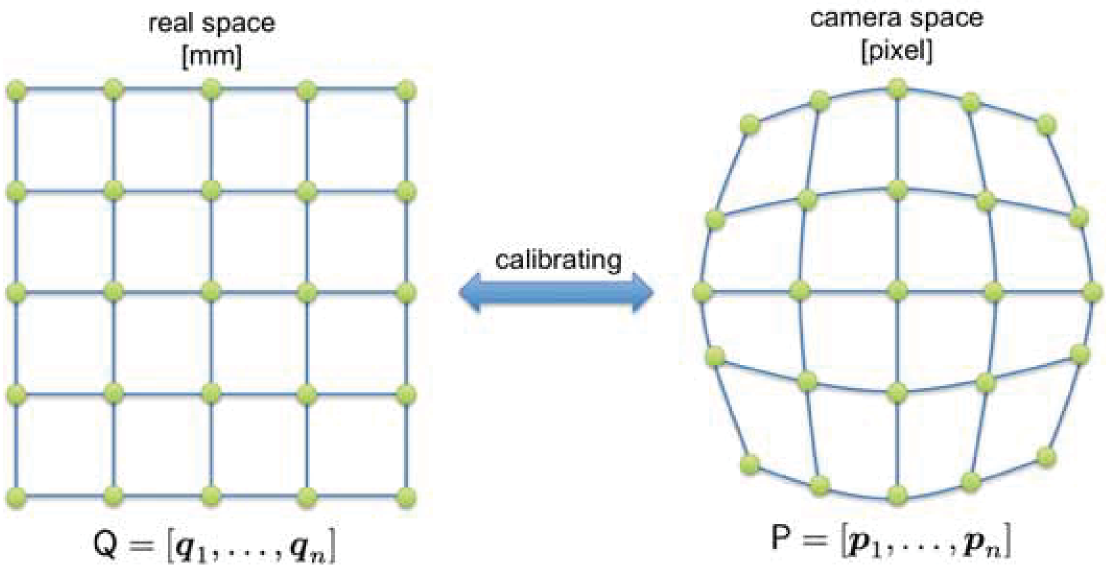

real space
[mm]
camera space
[pixel]
calibrating
$\mathsf{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n]$
$\mathsf{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n]$

Figure 6.3: The calibration process consists of two steps: at first, the image's base points P are selected and matched to their corresponding real world base points Q. Afterwards, a mapping function is determined, which is capable of transforming other points in the image using information about the matched base points. This is true for linear and non-linear spaces.

Obviously, a calibration for a *bijective mapping* function can only be carried out, if the condition $n_r = n_c$ is fulfilled. The schematic in Figure 6.3 illustrates how the generated mapping functions characterize geometric differences between the two spaces by establishing a mathematical relation between the base points P and Q. A number of factors determine the acquirable calibration quality and introduce a systematic error:

- the real world's reference points Q were produced with limited preciseness,
- the camera space's base points P can only be determined with a certain accuracy and
- the deployed mathematical models are approximation methods.

Each method requires a minimal number of n reference points to work as intended, whereby the number must be equal in both spaces, thus $n = n_r = n_c$. Additionally, the number n depends on the degree d of the function. Table 6.1 provides an overview.

method:	**required number of points:**
linear homography	$n = 4$
polynomial control points	$n = 6$ for $d = 2$ $n = 10$ for $d = 3$ $n = 15$ for $d = 4$
tensor interpolation	$n = d + 1$

Table 6.1: Required number of points for the coordinate mapping methods.

The metric coordinates in Q are a-priori known, the pixel coordinates P are extracted automatically from the image D, but in an arbitrary order. The calibration requires a sorted list of coordinates, whereby the elements in each list must match each other, i.e. $\boldsymbol{p}_i \leftrightarrow \boldsymbol{q}_i$ describe the same point in different spaces. The arranging of P is implemented via a custom hash code and *insertionsort*, the fastest sorting algorithm which is not based on a divide & conquer approach. The Source Code 6.1 shows the Matlab implementation, whereby $\boldsymbol{a}$ contains the elements and $\boldsymbol{b}$ holds the original index of each element.

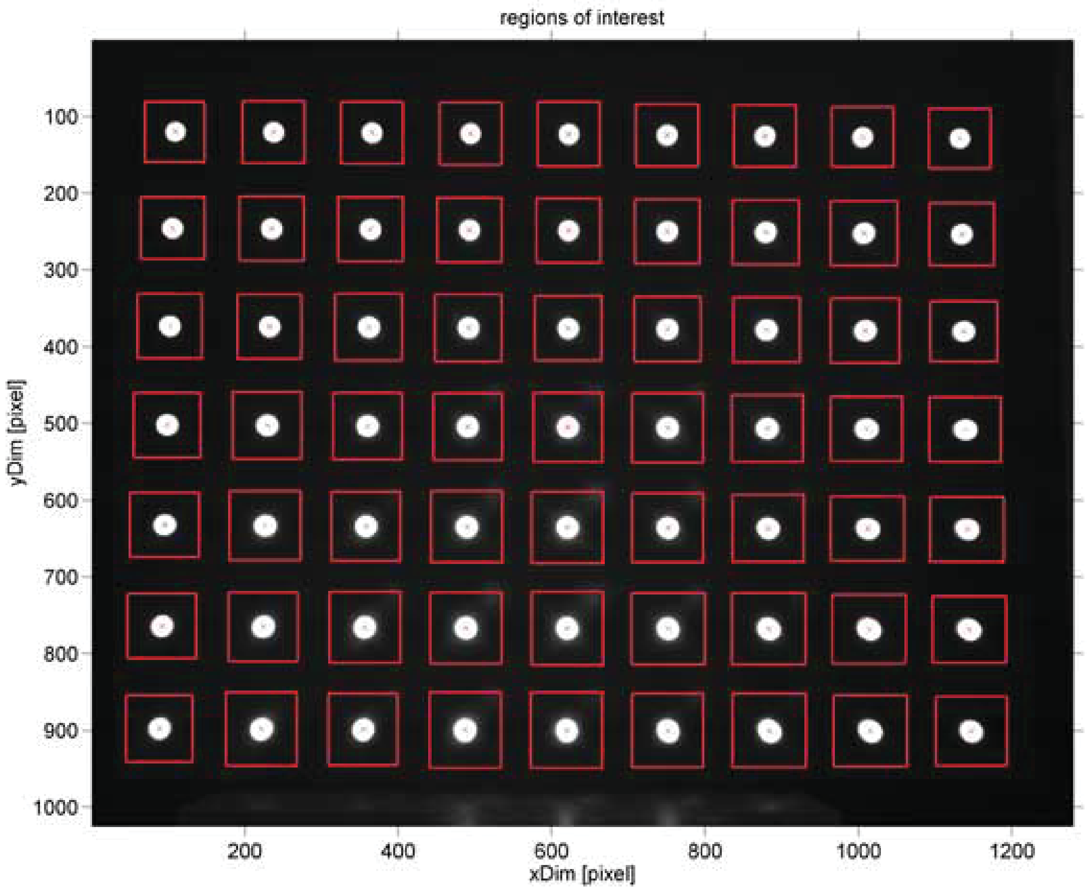
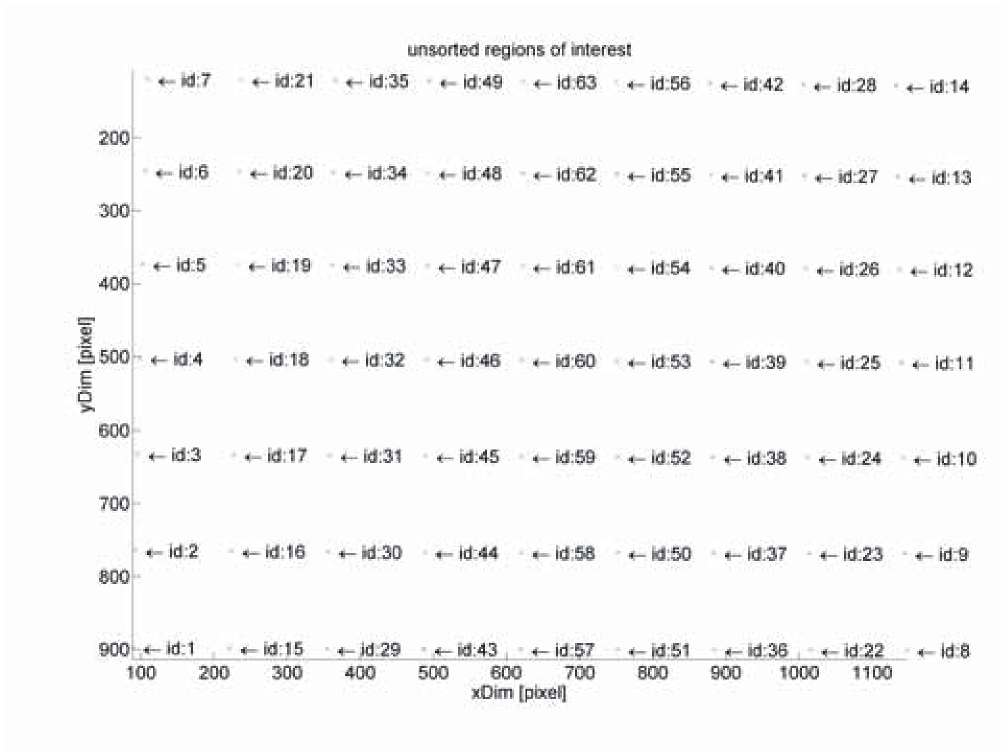


Figure 6.4: Each red rectangle indicates the existence of a region of interest R_i. The coordinates of the center points $\boldsymbol{p}_i$ are automatically acquired. Afterwards, a sorting procedure must be applied.

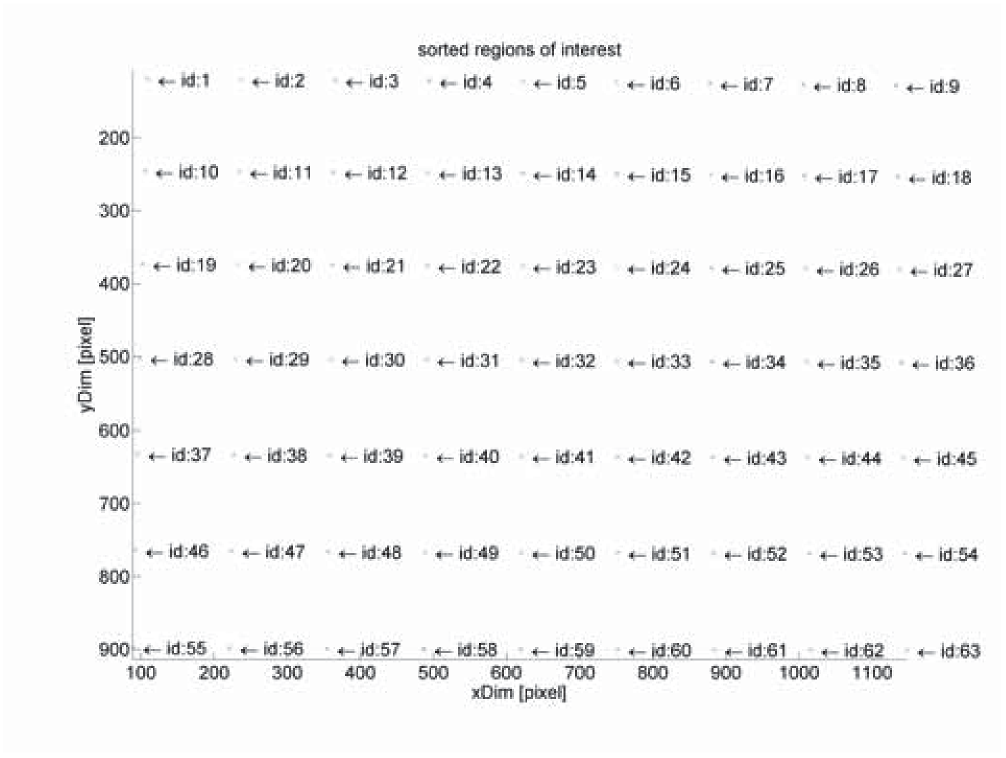
```
for i = 2:n
   cmp = a(i);
   idx = b(i);
   j = i-1;
   while (j≥ 1 && a(j)>cmp)
      a(j+1) = a(j);
      b(j+1) = b(j);
      j=j-1;
   end;
   a(j+1) = cmp;
   b(j+1) = idx;
end;
```

Source Code 6.1: The Matlab implementation of insertionsort.

The results of the automatic acquisition of the regions of interest R_i of an image D is visualized in Figure 6.4. The pre- and post sorted data points of P are shown in Figure 6.5.



(a) Unsorted regions of interest.



(b) Sorted regions of interest.

Figure 6.5: The ROIs are sorted from the upper left to the lower right corner. The result of the automatic acquisition is stored in the matrix of camera coordinates P. The correct order is essential for establishing a meaningful relation between P and Q.

The application's main problems are the geometric distortions caused by the non-perfect camera components, i.e. the objective and the sensor, and the non-affine projection. These circumstances generate optical effects, which must be handled by the mapping functions. Although the polynomial control points and tensor interpolation are more attractive than the linear homography, their price in terms of computation time is generally higher. Each method will be explaint thoughtfully in the following sections.

6.2 Linear Homography

As already mentioned in Chapter 5, homogeneous and affine coordinates of a point are related as following:

$$\boldsymbol{u} = [x_c, y_c, w_c]^T \qquad \text{and} \qquad \boldsymbol{p} = \left[\frac{x_c}{w_c}, \frac{y_c}{w_c}\right]^T, \tag{6.6}$$

which is a non-bijective mapping of

$$\mathbb{R}^3 \longrightarrow \mathbb{R}^2, \tag{6.7}$$

where w is the homogeneous component. Homogeneous coordinates allow it to deal with points at infinity (singularity) by setting $w = 0$. This issue allows the intersection of parallel lines in non-affine projections. The transformation is given by

$$\boldsymbol{v} = \mathsf{H}\boldsymbol{u}, \tag{6.8}$$

when $\boldsymbol{v}$ is the homogeneous equivalent of the affine point $\boldsymbol{q}$. The common approach to compute the homography matrix H is the *direct linear transformation* (DLT) algorithm. Equation 6.8 is expressed in homogeneous coordinates:

$$\begin{bmatrix} x_r \\ y_r \\ w_r \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix}. \tag{6.9}$$

The (3×3) *homography matrix* H may always be divided by a non-zero real number, i.e. there are only eight unknown variables. As a consequence, the location of at least four non-collinear points, i.e. they do not lie on one mutual line, must be known on each plane to yield a total of eight linear equations.

The affine coordinates can be acquired by rewriting Equation 6.9:

$$\boldsymbol{v} = \begin{bmatrix} \frac{x_r}{w_r} \\ \frac{y_r}{w_r} \\ \frac{w_r}{w_r} \end{bmatrix} = \begin{bmatrix} \frac{h_{11}x_c+h_{12}y_c+h_{13}}{h_{31}x_c+h_{32}y_c+h_{33}} \\ \frac{h_{21}x_c+h_{22}y_c+h_{23}}{h_{31}x_c+h_{32}y_c+h_{33}} \\ 1 \end{bmatrix} \qquad \text{and consequently} \qquad \boldsymbol{q} = \begin{bmatrix} \frac{x_r}{w_r} \\ \frac{y_r}{w_r} \end{bmatrix}. \tag{6.10}$$

The implicit form of Equation 6.10 is

$$-h_{11}x_c - h_{12}y_c - h_{13} + h_{31}x_c\frac{x_r}{w_r} + h_{32}y_c\frac{x_r}{w_r} + h_{33}\frac{x_r}{w_r} = 0, \tag{6.11}$$

$$-h_{21}x_c - h_{22}y_c - h_{23} + h_{31}x_c\frac{y_r}{w_r} + h_{32}y_c\frac{y_r}{w_r} + h_{33}\frac{y_r}{w_r} = 0, \tag{6.12}$$

i.e. each known point yields two equations. Redundant information can be employed by adding more than the needed four points per space to the application to make the process more robust. By applying the SVD operation, a least squares solution can be acquired for the overdetermined equation system. For $n \geq 4$ it can be written:

$$\begin{bmatrix} -x_c(1) & -y_c(1) & -1 & 0 & 0 & 0 & x_c(1)\frac{x_r(1)}{w_r(1)} & y_c(1)\frac{x_r(1)}{w_r(1)} & \frac{x_r(1)}{w_r(1)} \\ 0 & 0 & 0 & -x_c(1) & -y_c(1) & -1 & x_c(1)\frac{y_r(1)}{w_r(1)} & y_c(1)\frac{y_r(1)}{w_r(1)} & \frac{y_r(1)}{w_r(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ -x_c(n) & -y_c(n) & -1 & 0 & 0 & 0 & x_c(n)\frac{x_r(n)}{w_r(n)} & y_c(n)\frac{x_r(n)}{w_r(n)} & \frac{x_r(n)}{w_r(n)} \\ 0 & 0 & 0 & -x_c(n) & -y_c(n) & -1 & x_c(n)\frac{y_r(n)}{w_r(n)} & y_c(n)\frac{y_r(n)}{w_r(n)} & \frac{y_r(n)}{w_r(n)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_{2n} \end{bmatrix}, \tag{6.13}$$

which is in concise matrix form

$$\mathsf{C}\boldsymbol{h} = \boldsymbol{e}. \tag{6.14}$$

In fact, the total least squares approach deployed by the DLT algorithm is not the best idea, because it makes no difference between linear and quadratic parts of the error as well as the statistically invariant constants. Residualization based on orthogonal matrix projections delivers a reduced error structure for the linear system of equations. That is a new approach for the efficient non-iterative computation of H and can be found in literature [11].

6.3 Polynomial Control Points

The method locally models patches via polynomials by dividing the image into small regions. The overall mapping function is obtained by piecing together each pair of corresponding patches from camera space and real space. Like the linear homography, this technique is a total least squares approach. One drawback of the method is: when the images have local geometric differences or the base points (control points) are measured inaccurately, the accuracy averages out equally over the whole image. Orthogonal polynomials are used instead of ordinary polynomials to avoid unnecessary computational effort.

The described algorithm is directly implemented in Matlab via the *cp2tform()* function. The general procedure uses a valid pair of control points to infer a *spatial transformation* T from camera space to real space according to the defined transformation type. The method varies depending on the type. In this application, the polynomial transformation is utilized. It's advised to deploy this type when the used objective causes a curved image. Other possible options are: *non-reflective similarity, similarity, affine* and *projective.* The order of the polynomials can be specified in the function, with the degree $d = \{2, 3, 4\}$. Theoretically, a polynomial of degree $d = 2$ is enough to describe the optical distortions of a camera objective. The generated effect is based on a sphere's surface and therefore there's a quadratic relationship. However, the testing showed, that choosing a polynomial of degree $d = 4$ is better suited for the mathematical description of the discussed effect, because the camera space is obviously not a perfect sphere surface. The higher the order of the polynomial, the better the fit, but the result could contain more curves than the original image. When the minimum quantity of base points is available, the function finds the coefficients exactly. If there are more points accessible, a least squares solution is found. Ill-conditioned polynomials might result if too little pairs are used. When either the data of P or the Q has a large offset with respect to their origin, i.e. relative to the range of values that the data spans, then *cp2tform()* normalizes the data prior to fitting to enhance the numerical stability.

The deployed polynomial transformation uses polynomial functions of x and y to determine the mapping. The coordinates in the camera space are described by x_c and y_c, the coordinates in the real space are described by x_r and y_r. The transformation is performed through the spatial transformation matrix T. The second order $d = 2$ polynomial for each pair is described by 6 terms:

$$\begin{bmatrix} x_r & y_r \end{bmatrix} \mathsf{T} = \begin{bmatrix} 1 & x_c & y_c & x_c y_c & x_c^2 & y_c^2 \end{bmatrix}. \tag{6.15}$$

At least 6 control point pairs are needed to solve for the 12 unknown coefficients, T is (2×6) in this case. The fourth order $d = 4$ polynomial is described by 15 terms:

$$\begin{bmatrix} x_r & y_r \end{bmatrix} \mathsf{T} = \begin{bmatrix} 1 & x_c & y_c & x_c y_c & x_c^2 & y_c^2 & x_c^2 y_c & x_c y_c^2 & x_c^3 & y_c^3 & x_c^3 y_c & x_c^2 y_c^2 & x_c y_c^3 & x_c^4 & y_c^4 \end{bmatrix} \tag{6.16}$$

and here are at least 15 control point pairs necessary to solve for the 30 unknown coefficients, T is (2×15) in this case. Both, x_c and y_c, represent the polynomial coefficients of the specified degree [8, 9].

The Matlab code for the transformation is:

```
% inputPoints are the known real space coordinates and
% basePoints are the corresponding camera space coordinates
T = cp2tform(inputPoints, basePoints, 'polynomial', d);
% compute the mapped point's real coordinates
[xR, yR] = tforminv(T, xC, yC); % remapping
```

Source Code 6.2: Polynomial control points with the Matlab function *cp2tform()*.

6.4 Tensor Interpolation

The method uses an approach to describe the camera space via bases functions. The idea is to deploy *discrete orthogonal Gram polynomials* to model the distortions caused by the projection and the optics. The Gram basis is suitable when the data is predominantly geometric in nature, like it is the case in this application [21, 23].

The section introduces a *bivariate Gram polynomial tensor product approximation* for *image regularization*. The utilized bases are unitary[1], i.e. $\mathsf{X}^T\mathsf{X} = \mathsf{I}$, $\mathsf{Y}^T\mathsf{Y} = \mathsf{I}$, and compete, i.e. $\mathsf{X}\mathsf{X}^T = \mathsf{I}$, $\mathsf{Y}\mathsf{Y}^T = \mathsf{I}$. A consequence is, that Gaussian noise is evenly spread onto all spectral components. The polynomial decimation and tensor regularization serve to reduce the noise power significantly, resulting in a registration, which is not susceptible to such disturbances. The use of bases functions normally comes along with the idea of filtering, i.e. abstracting data to increase the information density[2]. The proposed method follows the inverse path: based on a few known base points, the space between them is interpolated. Another idea, which is not investigated further, would be the numerically time consuming local approximation via spline fitting [3, 2, 22].

[1]A unitary matrix is normalized and orthogonal, it can contain real and complex elements.
[2]This issue is explained in my bachelor thesis.

If a set of bases functions is applied to a two-dimensional data matrix Z, there must be one for the x dimension, i.e. X, and one for the y dimension, i.e. Y. By combining the bases functions X and Y with the spectrum S, the data Z can be synthesized. Note, that the computation is carried out separately for x and y dimensions, i.e. Z one time contains the base point's x coordinates and the other time it contains the y coordinates.

At first, the process is explained for complete bases functions for better understanding. The following equation describes the relation of the mentioned components:

$$\mathsf{Z}_c = \mathsf{Y}_c \mathsf{S}_c \mathsf{X}_c^T, \tag{6.17}$$

whereby Z_c is the complete data set, X_c and Y_c are the complete unitary bases functions in x and y dimension. Furthermore, S_c is the complete spectrum. Keep in mind, that premultiplication effects the rows and postmultiplication effects the columns of a matrix. Like mentioned in the previous section, an even degree of $d = 2$, $d = 4$ or $d = 6$ for the Gram polynomials suits the problem best. The matrices' dimensions[3] are very important for the comprehension of the introduced mathematical model, therefore they are written down explicitly for Equation 6.17:

$$(u \times v) = (u \times u)(u \times v)(v \times v). \tag{6.18}$$

By shifting the equation, the explicit form of S_c can be acquired.

$$\mathsf{S}_c = (\mathsf{Y}_c)^{-1} \mathsf{Z}_c (\mathsf{X}_c^T)^{-1}. \tag{6.19}$$

The bases functions' orthogonality conditions yields following property:

$$\mathsf{X}_c^T = \mathsf{X}_c^{-1} \qquad \text{and} \qquad \mathsf{Y}_c^T = \mathsf{Y}_c^{-1}. \tag{6.20}$$

Hence,

$$\mathsf{S}_c = (\mathsf{Y}_c)^T \mathsf{Z}_c (\mathsf{X}_c^T)^T. \tag{6.21}$$

Because of the calculation rule $(\mathsf{A}^T)^T = \mathsf{A}$, when A is a matrix, it can be written

$$\mathsf{S}_c = \mathsf{Y}_c^T \mathsf{Z}_c \mathsf{X}_c. \tag{6.22}$$

[3]The used notation is $(rows \times columns)$.

By subsidizing Equation 6.22 in Equation 6.17, the projections onto the complete bases functions $\mathsf{X}_c\mathsf{X}_c^T = \mathsf{I}$ and $\mathsf{Y}_c\mathsf{Y}_c^T = \mathsf{I}$ are acquired. It's only true for complete unitary matrices.

$$\hat{\mathsf{Z}}_c = \mathsf{Y}_c\mathsf{Y}_c^T\mathsf{Z}_c\mathsf{X}_c\mathsf{X}_c^T, \tag{6.23}$$

which is following in respect of the matrices' dimensions:

$$(u \times v) = (u \times u)(u \times u)(u \times v)(v \times v)(v \times v). \tag{6.24}$$

Obviously, the complete data set Z_c is the same size as the reprojected data set $\hat{\mathsf{Z}}_c$, i.e. $\mathsf{Z}_c = \hat{\mathsf{Z}}_c$ in this case. Incomplete sets of bases functions X_r and Y_r are used to describe the reduced spectrum S_r with the help of Z_r, which is the set of available base points:

$$\mathsf{Z}_r \approx \mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T. \tag{6.25}$$

The equation's dimensions are

$$(m \times n) = (m \times u)(u \times v)(v \times n). \tag{6.26}$$

In fact, the incomplete sets of bases functions X_r and Y_r are the known portions of the complete sets X_c and Y_c, i.e.

$$\mathsf{X}_r \in \mathsf{X}_c \qquad \text{and} \qquad \mathsf{Y}_r \in \mathsf{Y}_c. \tag{6.27}$$

The residuum E is the difference between the actual data Z_r and its approximation:

$$\mathsf{E} = \mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T - \mathsf{Z}_r. \tag{6.28}$$

This yields the cost function K, which should be minimized to get the least squares solution:

$$K = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^2 = ||\mathsf{E}||_F^2 = ||\mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T - \mathsf{Z}_r||_F^2 = \text{trace}\left\{\mathsf{E}\mathsf{E}^T\right\} \leftarrow \min, \tag{6.29}$$

whereby $||\mathsf{E}||_F^2$ is the squared *Frobenius norm* of E, i.e. the sum of the squares of all elements c_{ij} of the matrix. Evaluation of the term yields:

$$\begin{aligned} K &= \text{trace}\left\{(\mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T - \mathsf{Z}_r)(\mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T - \mathsf{Z}_r)^T\right\} & (6.30)\\ &= \text{trace}\left\{\mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T\mathsf{X}_r\mathsf{S}_r^T\mathsf{Y}_r^T\right\} - \text{trace}\left\{\mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T\mathsf{Z}_r^T\right\} & (6.31)\\ &\quad -\text{trace}\left\{\mathsf{Z}_r\mathsf{X}_r\mathsf{S}_r^T\mathsf{Y}_r^T\right\} + \text{trace}\left\{\mathsf{Z}_r\mathsf{Z}_r^T\right\}. & (6.32) \end{aligned}$$

By differentiating the cost function K with respect to the matrix S_r and setting it equal to zero, the matrix equation for the minimization is delivered:

$$\frac{\partial K}{\partial \mathsf{S}_r} = 2\left(\mathsf{Y}_r^T\mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T\mathsf{X}_r - \mathsf{Y}_r^T\mathsf{Z}_r\mathsf{X}_r\right) = 0. \tag{6.33}$$

Consequently,

$$\mathsf{Y}_r^T\mathsf{Y}_r\mathsf{S}_r\mathsf{X}_r^T\mathsf{X}_r = \mathsf{Y}_r^T\mathsf{Z}_r\mathsf{X}_r. \tag{6.34}$$

It follows, that

$$\mathsf{S}_r = (\mathsf{Y}_r^T\mathsf{Y}_r)^{-1}\mathsf{Y}_r^T\mathsf{Z}_r\mathsf{X}_r(\mathsf{X}_r^T\mathsf{X}_r)^{-1}, \tag{6.35}$$

whereby $\mathsf{A}^+ \triangleq (\mathsf{A}^T\mathsf{A})^{-1}\mathsf{A}^T$ is the *Moore-Penrose Pseudo inverse* of a rectangular, i.e. singular, matrix, which is in fact the desired least squares approach. Hence,

$$\mathsf{S}_r = \mathsf{Y}_r^+\mathsf{Z}_r(\mathsf{X}_r^+)^T \tag{6.36}$$

with the dimensions

$$(u \times v) = (u \times m)(m \times n)(n \times v). \tag{6.37}$$

The *interpolated grid* Z_g is obtained by synthesizing the reduced spectrum S_r with the complete bases functions X_c and Y_c. The process yields the least squares tensor product approximation:

$$\mathsf{Z}_g = \mathsf{Y}_c\mathsf{S}_r\mathsf{X}_c^T. \tag{6.38}$$

The matrices' dimensions are

$$(u \times v) = (u \times u)(u \times v)(v \times v). \tag{6.39}$$

By putting Equation 6.38 into Equation 6.36 the interpolation over the whole grid is acquired:

$$\mathsf{Z}_g = \mathsf{Y}_c\mathsf{Y}_r^+\mathsf{Z}_r(\mathsf{X}_r^+)^T\mathsf{X}_c^T, \tag{6.40}$$

$$(u \times v) = (u \times u)(u \times m)(m \times n)(n \times v)(v \times v). \tag{6.41}$$

The described process accomplishes following:

$$\underbrace{\mathsf{Z}_r}_{(m\times n)} \xrightarrow{\text{interpolate}} \underbrace{\mathsf{Z}_g}_{(u\times v)}, \tag{6.42}$$

whereby $u > m$ and $v > n$, i.e. the interpolated grid in Z_g has a higher resolution k than the original points stored in Z_r. Figure 6.6 illustrates the process.

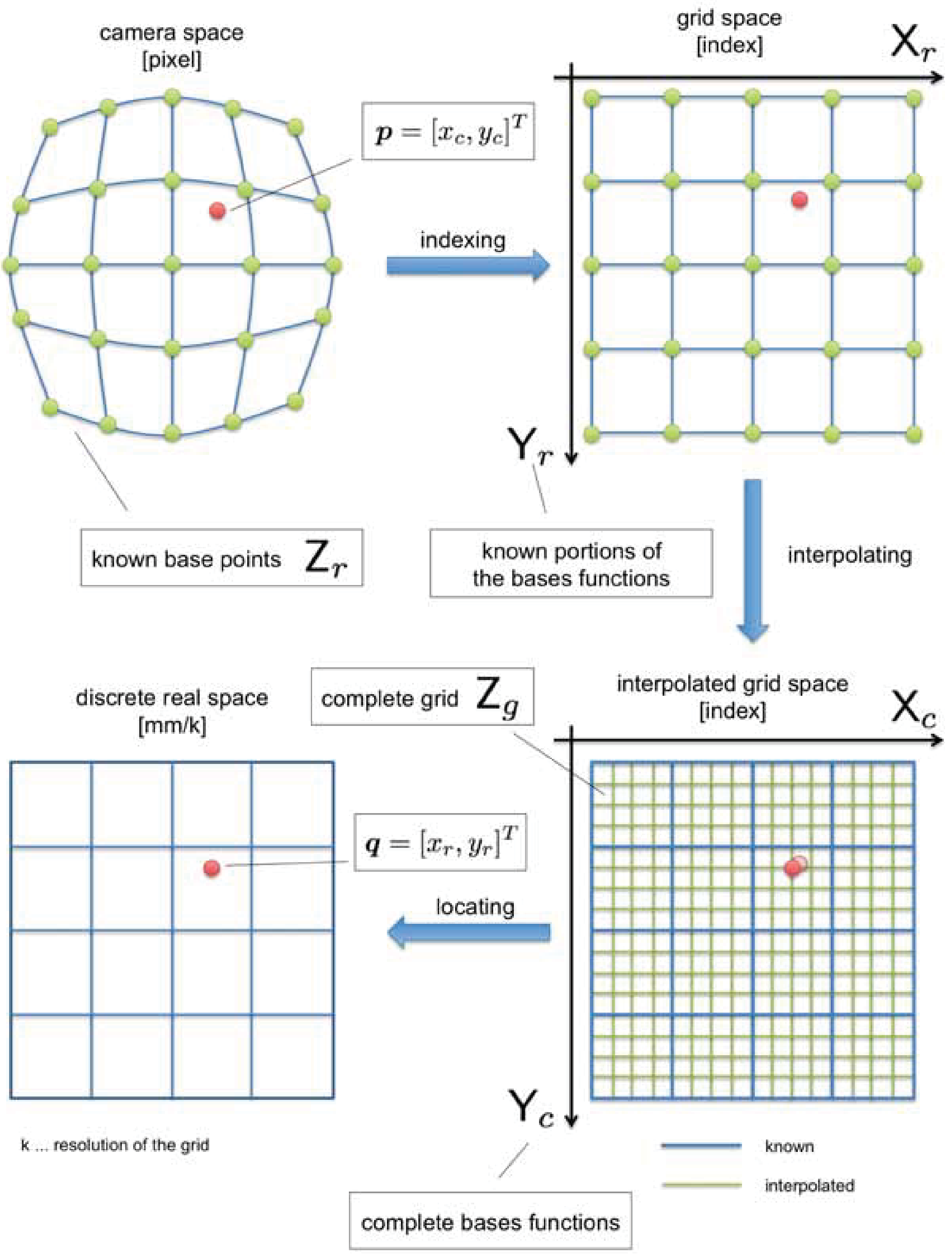


Figure 6.6: The goal is to map the point $\boldsymbol{p} = [x_c, y_c]^T$ from the camera space to its real space pendant $\boldsymbol{q} = [x_r, y_r]^T$. The known base points coordinates are stored in the matrix Z_r. The geometry is described by deploying a set of discrete orthogonal Gram polynomials in x and y direction. The reduced functions X_r and Y_r are the known portions of the complete bases X_c and Y_c. The performed interpolation yields the complete grid Z_g. The point $\boldsymbol{q}$ can now be located in the resulting discrete real space with the resolution k.

Furthermore, the known portions X_r and Y_r are equivalent to the base points stored in Z_r. This means, the polynomials are fitted through them and therefore describe the geometry. The space between the mentioned base points is interpolated. The achievable accuracy is determined through the number of elements in X_c and Y_c. This yields a discrete Cartesian coordinate frame, i.e. a grid, of the desired resolution k, which also defines the application's measurement quality. Note, that the point $\boldsymbol{p}$ can only lie on the interpolated grid, i.e. the application's accuracy depends on the resolution k. This is also the reason for the former continuous real space to be now discrete, as it depends on the underlying grid. Figure 6.7 visualizes the selection of the nearest grid point.

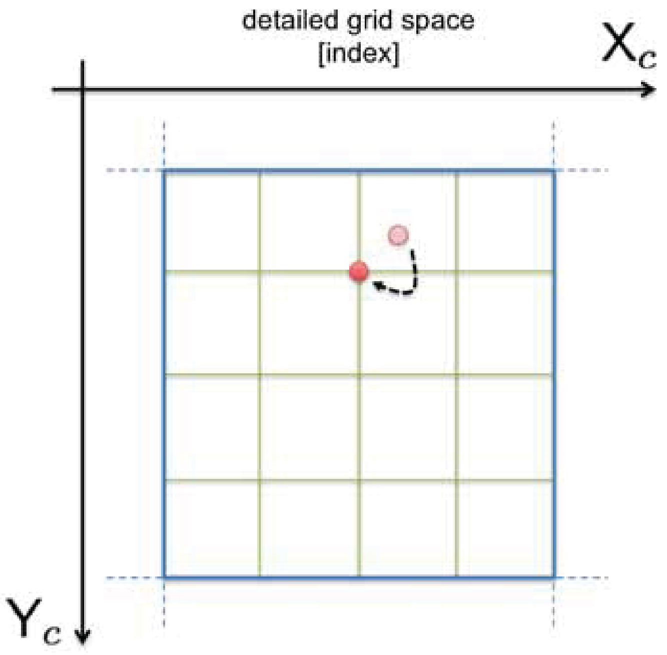


Figure 6.7: The point $\boldsymbol{p}$ can only lie on the grid, therefore the grid's resolution k determines the overall measurement accuracy. During locating, the nearest grid point next to $\boldsymbol{p}$ is searched and $\boldsymbol{p}$ is placed there. Afterwards, the corresponding location $\boldsymbol{q}$ in the discrete real space can be found.

Imagine, that Z_g had a $1mm$ grid. Thus, the resulting accuracy would be $\pm 0.5mm$. Of course this is insufficient accurate. The proposed method allows it to deploy a resolution of $0.01mm$, i.e. $k = 100$, when the base points are arranged in a $\delta_x = \delta_y = 25mm$ raster. Still, it's too less accurate. Additionally the computation is very ineffective and the numerical effort is enormous.

The procedure is implemented in Matlab as following:

```
% real space coordinates of known points, multiplied by resolution k
xs = xReal * k;
ys = yReal * k;
% determine number of elements (i.e. size of interpolated grid)
noXs = max( xs(:) );
noYs = max( ys(:) );
% generate the required sets of Gram basis functions of degree d
Xc = generateGramBasis( noXs, d);
Yc = generateGramBasis( noYs, d);
% extract the known portions of the bases functions
Xr = Xc( xs, :);
Yr = Yc( ys, :);
% compute the reduced spectra for x and y dimension utilizing the camera coordinates
Sx = pinv( Yr ) * xCam * pinv( Xr )';
Sy = pinv( Yr ) * yCam * pinv( Xr )';
% the interpolated grid is computed seperately for x and y dimension
Zx = Yc * Sx * Xc';
Zy = Yc * Sy * Xc';
```

Source Code 6.3: Tensor interpolation with Matlab.

Consider the following example: The investigated area in the real space is $(y \times x) = (150mm \times 200mm)$. If the resolution $k = 100$, then a grid of $(15\,000mm/k \times 20\,000mm/k)$ is acquired. According to Equation 6.36, the resulting matrix dimensions are

$$(15\,000 \times 20\,000) = (15\,000 \times 7)(7 \times 9)(9 \times 20\,000), \tag{6.43}$$

when Z_r has (7×9) base points. If the resolution goes even higher, the required memory demands are unacceptable. As a consequence, the following two subsections will propose ideas to handle the problem in a more economic manner. Note, that interpolating the whole image leads to complete image registration. It's is not necessary in this application, because only one point is of interest. Additionally, the implementation is not capable of extrapolating over the borders of the support.

6.4.1 Basis Function Sectioning

The idea of this method is to perform only a partial registration of the synthesis function. The rough location of the laser spot is known, i.e. the four nearest base points. By selecting only those portions X_a and Y_a of the complete bases functions X_c and Y_c, which are really needed, the *partly interpolated grid* Z_a can be computed:

$$\mathsf{Z}_a = \mathsf{Y}_a \mathsf{S}_r \mathsf{X}_a^T = \mathsf{Y}_a \mathsf{Y}_r^+ \mathsf{Z}_r (\mathsf{X}_r^+)^T \mathsf{X}_a^T, \tag{6.44}$$

which is in terms of matrix dimensions:

$$(a \times b) = (a \times u)(u \times m)(m \times n)(n \times v)(v \times b). \tag{6.45}$$

The bases functions X_a and Y_a are sections of X_c and Y_c

$$\mathsf{X}_c^T = \begin{bmatrix} \dots & \mathsf{X}_a^T & \dots \end{bmatrix} \qquad \text{and} \qquad \mathsf{Y}_c = \begin{bmatrix} \vdots \\ \mathsf{Y}_a \\ \vdots \end{bmatrix}. \tag{6.46}$$

Accordingly,

$$\mathsf{X}_a \in \mathsf{X}_c \qquad \text{and} \qquad \mathsf{Y}_a \in \mathsf{Y}_c. \tag{6.47}$$

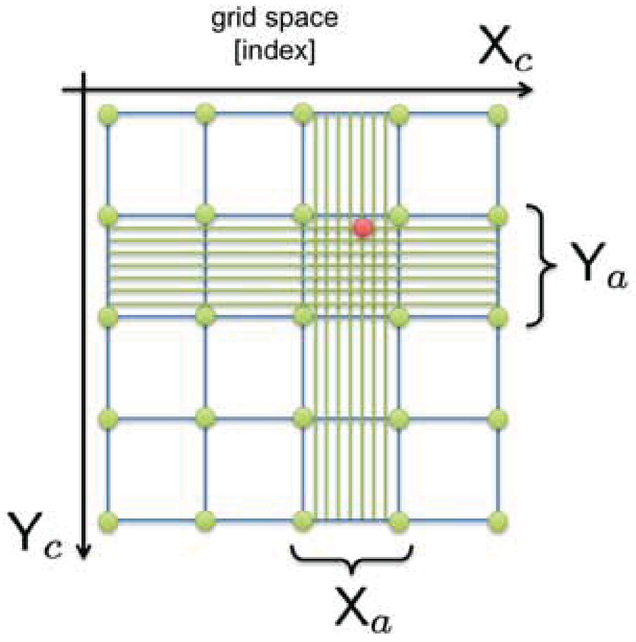


Figure 6.8: The partial bases functions X_a and Y_a are sections of X_c and Y_c. The partly interpolated grid Z_a can provide a much higher resolution than the complete grid Z_g.

Figure 6.8 shows the effect of the approach. The partly interpolated grid Z_a contains the laser spot. This matrix is considerable smaller than the complete grid Z_g, i.e. the resolution can be much higher than original. Note, that the resulting offset must be considered. Nevertheless, there's space for further improvements. The matrices are still very large, especially when the resolution is high. Thus, many operations are necessary, because the object of interest is just one single point. The next subsection introduces an approach through hierarchical subdivision.

6.4.2 Quad Tree Decomposition

This variation of the introduced method applies a hierarchical subdivision to the algorithm. The major improvement is achieved by performing a suitable decimation at each level. It enables a numerically more efficient implementation than interpolating the complete grid Z_g or the partial grid Z_a. A local registration is performed at each level while ascending the hierarchy, whereby each iteration computes just *one point* z_p via the following formula:

$$z_p = \boldsymbol{y}_p \mathsf{S}_r \boldsymbol{x}_p^T = \boldsymbol{y}_p \mathsf{Y}_r^+ \mathsf{Z}_r (\mathsf{X}_r^+)^T \boldsymbol{x}_p^T, \tag{6.48}$$

which is in terms of matrix dimensions:

$$(1 \times 1) = (1 \times u)(u \times m)(m \times n)(n \times v)(v \times 1). \tag{6.49}$$

The vectors $\boldsymbol{x}_p$ and $\boldsymbol{y}_p$ are sections of X_c and Y_c

$$\mathsf{X}_c^T = \begin{bmatrix} \dots & \boldsymbol{x}_p^T & \dots \end{bmatrix} \qquad \text{and} \qquad \mathsf{Y}_c = \begin{bmatrix} \vdots \\ \boldsymbol{y}_p \\ \vdots \end{bmatrix}. \tag{6.50}$$

This means

$$\boldsymbol{x}_p \in \mathsf{X}_c \qquad \text{and} \qquad \boldsymbol{y}_p \in \mathsf{Y}_c. \tag{6.51}$$

The process pins down the pixel coordinates of the camera point $\boldsymbol{p}$ incrementally. The idea can be described as a *'binary search in a two-dimensional space'*, which is in fact a *quaternary search*. Each iteration step increases the accuracy until one of the following termination conditions triggers:

- the defined tolerance or
- the maximal number of iterations is reached.

After the procedure, the real coordinates of $\boldsymbol{q}$ can easily be determined by finding the corresponding location in the discrete real space. Figure 6.9 illustrates, that theoretically, it would be enough to use just two vectors to describe the point's position. It's an immense saving of computation effort and runtime in comparison to the other methods. Consequently, the algorithm could run on an embedded system. Practically, the achievable resolution k has no relevant limits in respect to the mathematical model behind it. The limiting factors are based on the physical boundaries and the used equipment.

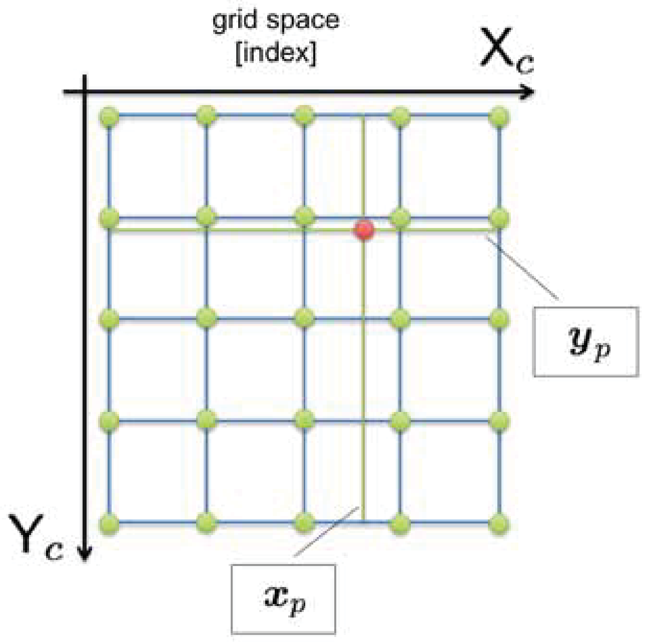


Figure 6.9: The basis function vectors $\boldsymbol{x}_p$ and $\boldsymbol{y}_p$ are sections of the complete bases functions X_c and Y_c. The point's position can be described by just these two vectors. The accuracy has no relevant limits caused by the mathematical model.

The process starts with the a-priori knowledge of the base points' location saved in the matrix Z_r. The enclosed area is quartered and the quarter containing the sought point is selected. The four corners' locations are computed utilizing the basis function vectors $\boldsymbol{x}_p$ and $\boldsymbol{y}_p$. Once again, the resulting rectangle is quartered and the procedure starts anew. The recursion is stopped when one of the mentioned termination conditions is fulfilled.

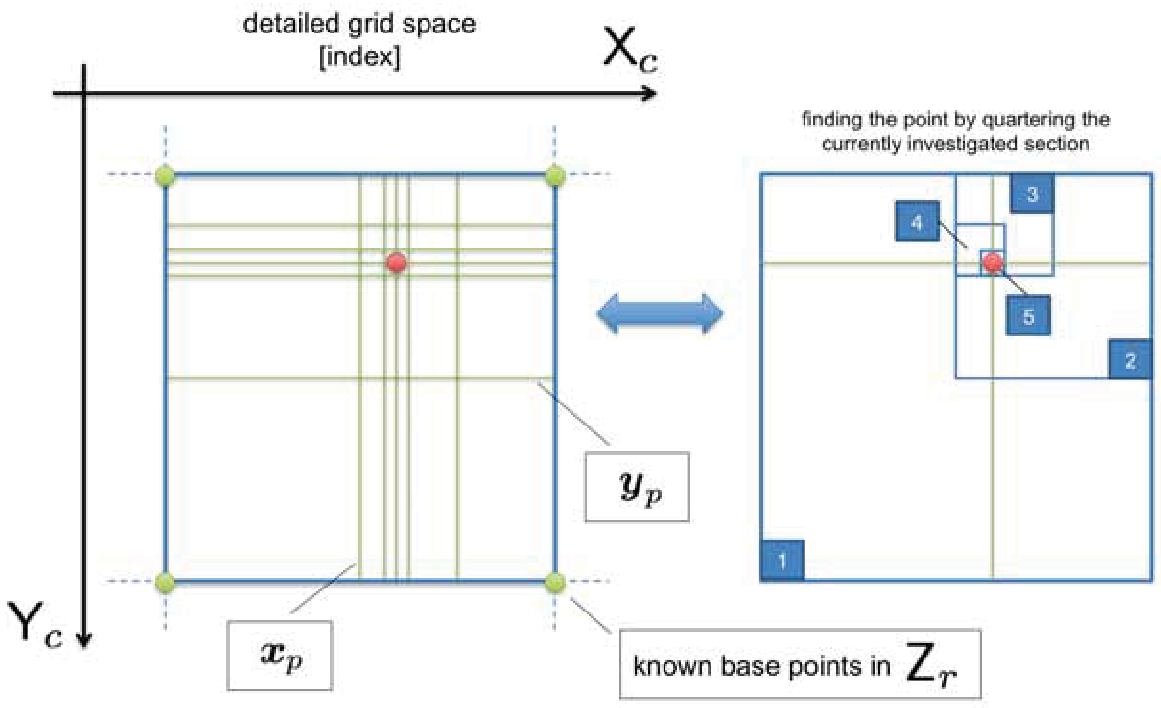

detailed grid space [index]
X_c
Y_c
finding the point by quartering the currently investigated section
1
2
3
4
5
$\boldsymbol{y}_p$
$\boldsymbol{x}_p$
known base points in Z_r

Figure 6.10: The section of Z_r containing the searched point is selected. The investigated area is quartered and the quarter's corners are computed via the basis function vectors $\boldsymbol{x}_p$ and $\boldsymbol{y}_p$. Afterwards, the procedure is executed again. The process incrementally approximates the point's location until the desired accuracy is reached.

The proposed hierarchical approach utilizing only portions of unitary bases functions is new for such kind of image processing routine. The *quaternary search tree algorithm* can be implemented as a highly efficient data structure. The resulting depth of the quad tree τ is equivalent to the number of elements n, i.e. the possible locations inside the interpolated grid:

$$\tau = \log_4 n. \tag{6.52}$$

The runtime t_r of the algorithm is proportional:

$$t_r = \mathcal{O}(\log_4 n) \tag{6.53}$$

Consider the following example for a rough runtime estimation: the real coordinates of the base points in Z_r are positioned in a $\delta = 25mm$ grid and the utilized resolution $k = 1\,000$, i.e. the measurement accuracy is $\pm 0.5\mu m$. The possible locations inside the grid are

$$n = 25 \cdot k \cdot 25 \cdot k = 25\,000 \cdot 25\,000 = 625\,000\,000. \tag{6.54}$$

It's a huge number and such matrix operations cannot be computed economically. As a result, the standard tensor interpolation reaches it's limits when the resolution is too high. Even by utilizing bases function sectioning, the computational effort is extreme. By deploying quad tree decomposition, the depth of the search structure for this example is

$$\tau = \log_4 n = \frac{\ln n}{\ln 4} = \frac{\ln 625\,000\,000}{\ln 4} = 14.6096 \approx 15. \tag{6.55}$$

Thus, maximal $\tau = 15$ search levels are needed. Still, it's the worst case scenario, because the point could be found earlier during the search and the procedure terminates prematurely! For a resolution of $k = 100\,000$, which actually makes no sense in a practical manner, the number of steps is $\tau = 22$. It's obvious, that the savings in runtime are immense and this approach provides much potential for other fields of applications.

Chapter 7

Statistical Analysis

In this application, there are two different types of data depending on their origin: the calibration LEDs and the laser spot, which have their own characteristics and therefore the measured data must be investigated thoroughly. The acquired empirical sample values describe how the overall system behaves in terms of preciseness and robustness. Statistical analysis helps to understand the data's nature [26].

Two basic test-cases are defined to cover both types of data, which can be embedded inside the investigated feature R:

- measurement of a *single LED* and
- measurement of a *single laser spot*

over n repetitions. Each case delivers sets of the original camera coordinates $\boldsymbol{p} = [x_c, y_c]^T$ and in addition the mapped real coordinates $\boldsymbol{q} = [x_r, y_r]^T$. The goal of this procedure is to show, that the measured sample values have Gaussian distribution. Only in this case, a meaningful error estimation is possible. Furthermore, the proposed methods will deliver maximum likelihood estimations.

7.1 Descriptive Statistics

Consider a discrete data stream, i.e. a set of samples $\{a_1, \dots, a_i, \dots, a_n\}$, which is saved in a vector $\boldsymbol{a}$, whereby a_i indicates the current element and n is the number of elements. For statistical analysis, the concept of *central moments* is utilized, which allows it to meaningfully characterize the properties of a data set's probability distribution. They are better suited for the task than ordinary moments, because the values' higher order quantities relate only to the spread and shape of the distribution, rather than to its location. Central moments are described via the k^{th} moment about the mean[1]. For a real-value data set $\boldsymbol{a}$ it is

$$\mu_k = \mathbb{E}\left[(\boldsymbol{a} - \mathbb{E}[\boldsymbol{a}])^k\right], \tag{7.1}$$

where $\mathbb{E}$ is the expectation operator. The central moments of degree 0 and 1 are $\mu_0 \triangleq 1$ and $\mu_1 \triangleq 0$. The second degree moment is known as the variance $\mu_2 \triangleq \sigma^2$, the third and fourth central moment are only used in their standardized form in respect of the σ of degree k. This leads to *standardized moments*:

$$\frac{\mu_k}{\sigma^k}. \tag{7.2}$$

The third and fourth order standardized moments are known as skewness γ_1 and kurtosis β_2:

$$\gamma_1 \triangleq \frac{\mu_3}{\sigma^3} \qquad \text{and} \qquad \beta_2 \triangleq \frac{\mu_4}{\sigma^4}. \tag{7.3}$$

When investigating measured empirical data, which is identically distributed, but the population distribution itself is unknown, following simplifications can be assumed:

$$\mu = \mathbb{E}[\boldsymbol{a}] = \bar{a} \qquad \text{and} \qquad \sigma^2 = \mathrm{Var}[\boldsymbol{a}] = s^2. \tag{7.4}$$

This knowledge enables the computation of the following data attributes:

1. The *mode* $\hat{a}$ is the most common value in a set of samples $\boldsymbol{a}$;

2. The *median* $\tilde{a}$ delivers the element, which is in the middle of $\boldsymbol{a}$;

$$\tilde{a} = \begin{cases} a_{\frac{n+1}{2}} & \text{if } n \text{ is odd} \\ \frac{1}{2}(a_{\frac{n}{2}} + a_{\frac{n}{2}+1}) & \text{if } n \text{ is even.} \end{cases} \tag{7.5}$$

[1] It is also called the k^{th} central moment.

3. The *mean* $\bar{a}$ is the arithmetic average value of all elements in $\boldsymbol{a}$;

$$\bar{a} = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{7.6}$$

4. The *variance* s^2 and *standard deviation* s for a set of sample $\boldsymbol{a}$ are computed as[2]

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(a_i - \bar{a})^2 \qquad \text{and therefore} \qquad s = \sqrt{s^2} \tag{7.7}$$

   and describe the natural scatter of the data;

5. The *skewness* for empirical data $\hat{\gamma_1}$ is computed as

$$\hat{\gamma_1} = \frac{\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^3}{(\sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^2})^3} \tag{7.8}$$

   and can be interpreted as following: if $\hat{\gamma_1} > 0$, then most samples are left of the median $\tilde{a}$; if $\hat{\gamma_1} < 0$, then most samples are right of the median $\tilde{a}$; and only if $\hat{\gamma_1} \approx 0$, then the data is symmetric as in a Gaussian distribution;

6. The *kurtosis* for empirical data $\hat{\beta_2}$ is computed as

$$\hat{\beta_2} = \frac{\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^4}{(\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^2)^2} \tag{7.9}$$

   and can be interpreted as following: if $\hat{\beta_2} \approx 3$, then the data looks like a Gaussian bell-shaped curve; if $\hat{\beta_2} < 3$, then the data is more flat; if $\hat{\beta_2} > 3$, then the data is more peaked.

These statistical properties will be utilized during the evaluation of the measured data.

[2]This is the formula for the corrected variance, which is utilized for empirical data.

7.2 Standard Error

The *standard error* estimates how well a measured value is known [27]. In this case, the standard error of the mean e is computed by standardizing the standard deviation s in respect to the root of the sample size n:

$$e = \frac{s}{\sqrt{n}}.$$

If the data samples are assumed to be normally distributed, quantiles of the normal distribution, the sample mean and standard error can be used to calculate approximate confidence intervals for the mean. The following expressions are used to compute the upper and lower confidence limits for 95% and 99% respectively:

$$e_{95\%} = Z_{95\%}\frac{s}{\sqrt{n}} \qquad \text{and} \qquad Z_{95\%} = 1.96, \tag{7.10}$$

$$e_{99\%} = Z_{99\%}\frac{s}{\sqrt{n}} \qquad \text{and} \qquad Z_{99\%} = 2.58. \tag{7.11}$$

The Z value describes in how many cases the expected value lies within the desired boundaries of a Gaussian curve. The standard error estimates the real value of the mean with a given uncertainty:

$$\bar{a}_{95\%} = \bar{a} \pm e_{95\%} \qquad \text{and} \qquad \bar{a}_{99\%} = \bar{a} \pm e_{99\%}. \tag{7.12}$$

According to most literature [26, 27, 19] the number of empirical samples, such as measurement data, n must be at least 30, i.e.

$$n \geq 30. \tag{7.13}$$

There also exists a rule of thumb: $n \approx 5^d$, when d is the degree of the moment to be computed, e.g. if the variance should be acquired, then $d = 2$. Some statistical tests also provide an operation characteristics (OC) curve to determine a suitable n. Be aware, that there's an important difference between the standard deviation s and the standard error e. The natural scatter of data is described by s. It doesn't matter how big the number of samples n is, s stays almost constant. On the other hand, the standard error e describes how precise the mean $\bar{a}$ can be determined, in fact e gets lower the more samples exist.

7.3 Kolmogorov-Smirnov Test

This statistical non-parametric test introduced by Kolmogorov and Smirnov (*K-S test*) is used to check the goodness of fit between one-dimensional empirical data and its expected reference continuous parent distribution. The test quantifies the distance between both cumulative distribution functions [16].

During the experiments, the measured data $\boldsymbol{a}$ is tested, if it is normally distributed according to the Gaussian probability density function $f(t)$, where t is the continuous equivalent of the discrete a_i values:

$$f(t) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}\left(\frac{t-\mu}{\sigma}\right)^2}. \tag{7.14}$$

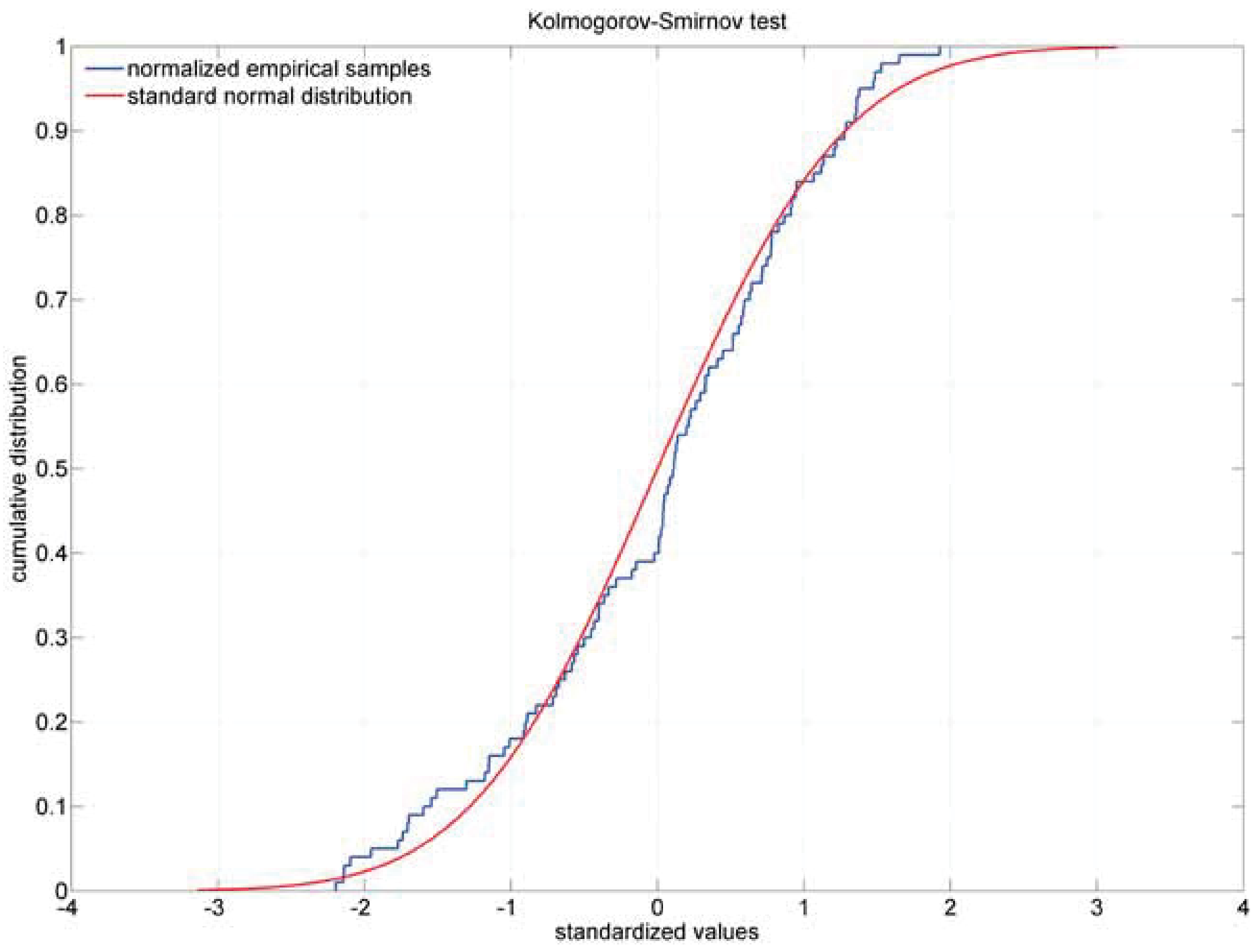


Figure 7.1: The K-S test quantifies the difference between the cumulative distribution functions of the normalized empirical data $F(\boldsymbol{a})$ and the standard normal distribution $F(t)$. If the difference is not too high, both functions are considered to be equivalent in terms of their distribution densities $f(\boldsymbol{a})$ and $f(t)$. Note, that $\boldsymbol{a}$ are discrete and t are continuous values.

By integrating the probability density function $f(t)$, the cumulative distribution function $F(t)$ with respect to t is computed by

$$F(t) = \int_{-\infty}^{t} f(t)\,\mathrm{d}t = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{t} \mathrm{e}^{-\frac{1}{2}(\frac{t-\mu}{\sigma})^2}\,\mathrm{d}t. \tag{7.15}$$

The standard normal distribution function $\Phi(t)$ is acquired with the parameter values $\mu = 0$ and $\sigma = 1$, hence

$$\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} \mathrm{e}^{-\frac{1}{2}t^2}\,\mathrm{d}t. \tag{7.16}$$

To enable the comparison between the discrete empirical samples a_i, their normalized values are computed and evaluated at the function steps:

$$\Phi\left(\frac{a_i - \bar{a}}{s}\right). \tag{7.17}$$

If the difference between the measured data and the expected distribution is not too larg, let's say a 95% confidence interval, which is approximately equivalent to 2σ, then the data is assumed to be normally distributed and the null-hypothesis H_0 is accepted. See Figure 7.1 for an example.

7.4 Measurements and Results

To ensure significant results, the number of iterations n is set to 100 to meet the condition in Equation 7.13 with an additional safety factor[3]. As already mentioned, the measurements are carried out for two different types of data sources for R: a single LED and a single laser spot. The numerical results of the statistical testing can be found in Appendix C.

The tests show, that the camera coordinates $\boldsymbol{p} = [x_c, y_c]^T$ are always normally distributed. Depending on the used mapping method, this is not entirely true for the real coordinates $\boldsymbol{q} = [x_r, y_r]^T$: on the one hand the data computed via linear homography and polynomial control points do have Gaussian distribution, on the other hand the tensor interpolation via basis function sectioning and tensor quad tree decomposition are not normally distributed in general, because of the discrete way they are computed. Nevertheless, it can be assumed, that the measured data samples are normally distributed and the operations performed on them do not change their statistical nature.

[3]Note, that in further chapters, ρ is the number of repetitions and ε is the number of investigated objects.

Chapter 8

Experimental Verification

In this chapter, the performance of the introduced methods is investigated under different conditions. It's demonstrated, that the procedures work as intended and the predicted behavior is verified. The identification of relevant process parameters is of utmost importance for the experimental strategy and for later applications. The following three setups are deployed to test certain aspects of the system:

1. The *base points*, which were originally be used to calibrate the system, are *remapped* from the camera space to the real space. The technique for determining an image object's center as well as the degrees of the non-linear coordinate mapping methods are varied. Additionally, two types of optics are employed: a normal-angle lens with $f = 12mm$ and a wide-angle lens with $f = 3.6mm$. The goal is to find out, if the mapping methods work for non-linear spaces and if the center coordinates of the image objects can be extracted reliably. See Section 8.1;

2. The system can either be adjusted by a designated calibration plate with LEDs or a laser mounted on a xy-table. An interlaced cross validation matrix is produced to verify the quality of each *calibration mode*. Once again, different center determination techniques are tested. The experiment's aim is to give a statement, which one of the calibration modes is better suited for the application. See Section 8.2;

3. The selection of the target is important when it comes to real operating conditions. Therefore, measurements are carried out on following hardware setups:

   - a metal target, with no obstacle between the target and the laser,
   - a metal target, with transparent glass between the target and the laser and
   - an opaque glass target, with no obstacle between the target and the laser.

   The goal is to verify, if *measuring the laser spot* on different targets influences the output and if the solution is valid in terms of accuracy. See Section 8.3.

The experiments take advantage of the center determination techniques introduced in Chapter 5. Furthermore, each coordinate mapping method is evaluated, whereby these synonyms are used in the next sections:

synonym:	**coordinate mapping method:**
homography	linear homography
polyCtrlPts	polynomial control points
tensorBFS	tensor interpolation via basis function sectioning
tensorQTD	tensor interpolation via quad tree decomposition

Table 8.1: Synonyms for the coordinate mapping methods.

The number of repetitions for the measurement of one point is ρ. The number of investigated objects is ε, whereby this can either be one point or the distance between a pair of points. This data acquisition under fixed operating conditions provides the opportunity to differentiate systematic errors, which remain constant during a given series of repeated measurements, and random errors. The standard error $e_{95\%}$ for a 95% confidence interval is computed either for ρ, if one point is measured often, or for ε, if many objects are measured once, i.e.

$$e_{95\%} = Z_{95\%}\frac{s}{\sqrt{\rho}} \qquad \text{or} \qquad e_{95\%} = Z_{95\%}\frac{s}{\sqrt{\varepsilon}}, \tag{8.1}$$

whereby s is the standard deviation of the test data. The segmentation process uses Otsu's local thresholding method for the binary morphology; the tensor interpolation uses a resolution of $k_{BFS} = 100$ for basis function sectioning and $k_{QDT} = 10\,000$ for quad tree decomposition. The interpretation of the results delivered by the three experiments can be found in Section 8.4.

8.1 Base Points Remapping

During this experiment, the base points, which were originally used to calibrate the system, are remapped from the camera space to the real space utilizing the introduced methods for center determination and coordinate mapping. It is known a-priori, where the remapped points should be located, as they should have the same positions as the corresponding base points. The difference between an individual base point $\boldsymbol{q}_c$ and the according remapped point $\boldsymbol{q}_m$ is the resulting error $\boldsymbol{e}$, i.e.

$$\boldsymbol{q}_c - \boldsymbol{q}_m = \boldsymbol{e}, \qquad \text{which is} \qquad \begin{bmatrix} x_c \\ y_c \end{bmatrix} - \begin{bmatrix} x_m \\ y_m \end{bmatrix} = \begin{bmatrix} e_x \\ e_y \end{bmatrix}. \tag{8.2}$$

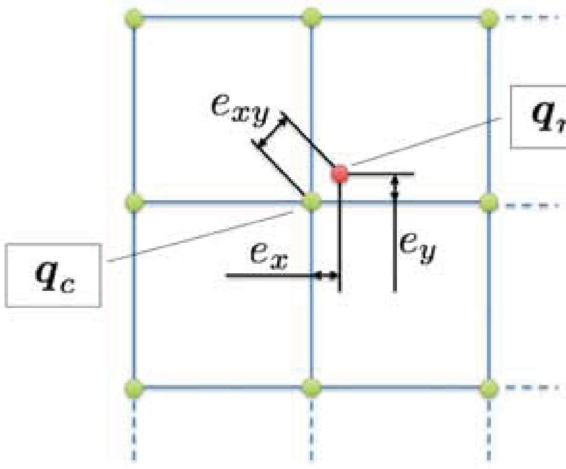


Figure 8.1: The green dots indicate the known base point $\boldsymbol{q}_c$, the red dot is the measured point $\boldsymbol{q}_m$. The errors e_x and e_y are the differences in x and y dimensions, e_{xy} is the distance between both points.

The error calculations are carried out for $(m \times n)$ base points, i.e. the number of elements is $\varepsilon = (m \cdot n)$. The characteristics of the outcome can be summarized by the means of errors in x and y dimensions, $\bar{e}_x$ and $\bar{e}_y$:

$$\bar{e}_x = \frac{1}{\varepsilon}\sum_{i=1}^{\varepsilon} |e_{xi}| \qquad \text{and} \qquad \bar{e}_y = \frac{1}{\varepsilon}\sum_{i=1}^{\varepsilon} |e_{yi}| \tag{8.3}$$

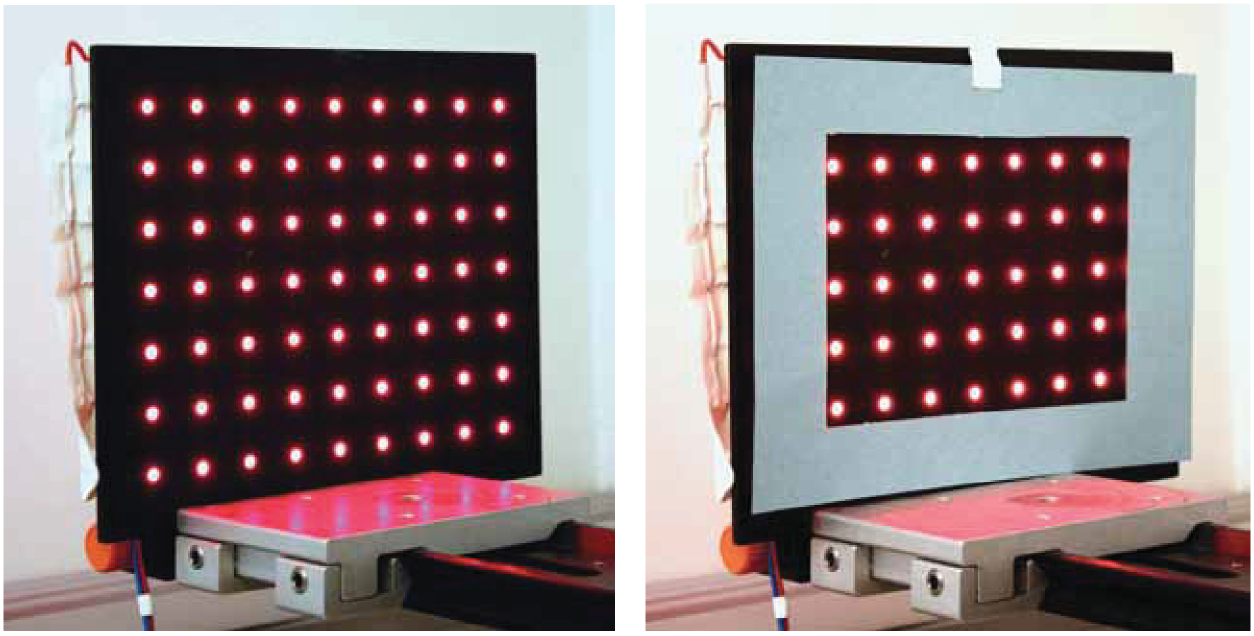
as well as their combined error e_{xy} and its mean $\bar{e}_{xy}$:

$$e_{xy_i} = \sqrt{e_{xi}^2 + e_{yi}^2}, \qquad \text{therefore} \qquad \bar{e}_{xy} = \frac{1}{\varepsilon}\sum_{i=1}^{\varepsilon} e_{xy_i}. \tag{8.4}$$

The standard error $e_{95\%}$ is computed for the examined e_{xy_i} values, i.e. the expected error for one point is $\bar{e}_{xy_{95\%i}} = \bar{e}_{xy_i} \pm e_{95\%_i}$.

The calibration utilizes $(7 \times 9) = 63$ LEDs, this can be seen in Figure 8.3 and 8.9. The validation image is just using the inner data points, meaning the first and the last row as well as the first and the last column are ignored. This leads to $(5 \times 7) = 35$ LEDs, which are used for the evaluation of the remapping quality, see figures 8.4 and 8.10. Hence, the number of data points to be evaluated is $\varepsilon = 35$. The outcome for two optical arrangements is tested: for a normal-angle lens with $f = 12mm$ as well as for a wide-angle lens of $f = 3.6mm$. As stated in Chapter 2, the object distance g must be adjusted depending on the installed lens, i.e. $g(f = 12mm) = 440mm$ and $g(f = 3.6mm) = 140mm$.

The figures in this section the show best solutions for each coordinate mapping method. The detailed results for the base points remapping can be found in Appendix D.



(a) The calibration plate features 63 LEDs.

(b) The validation plate features 35 LEDs.

Figure 8.2: The system is calibrated with 63 LEDs. By remapping the inner 35 LEDs, the quality of the mathematical models is evaluated. The position of each LED in real space is known a-priori, this allows the direct evaluation of the remapping error.

8.1.1 Focal Length of $f = 12mm$

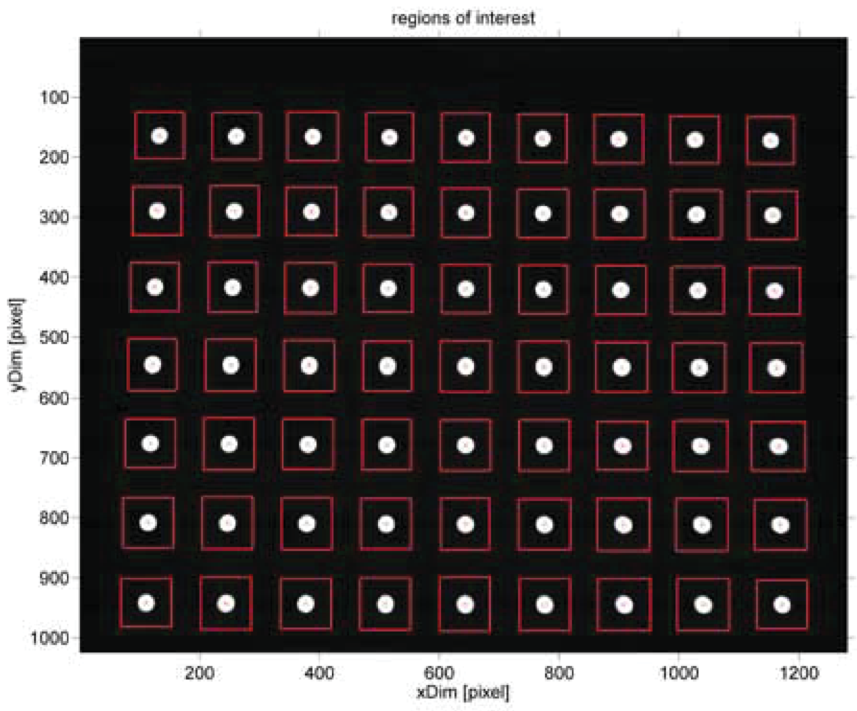


Figure 8.3: Calibration image utilizing a lens of $f = 12mm$.

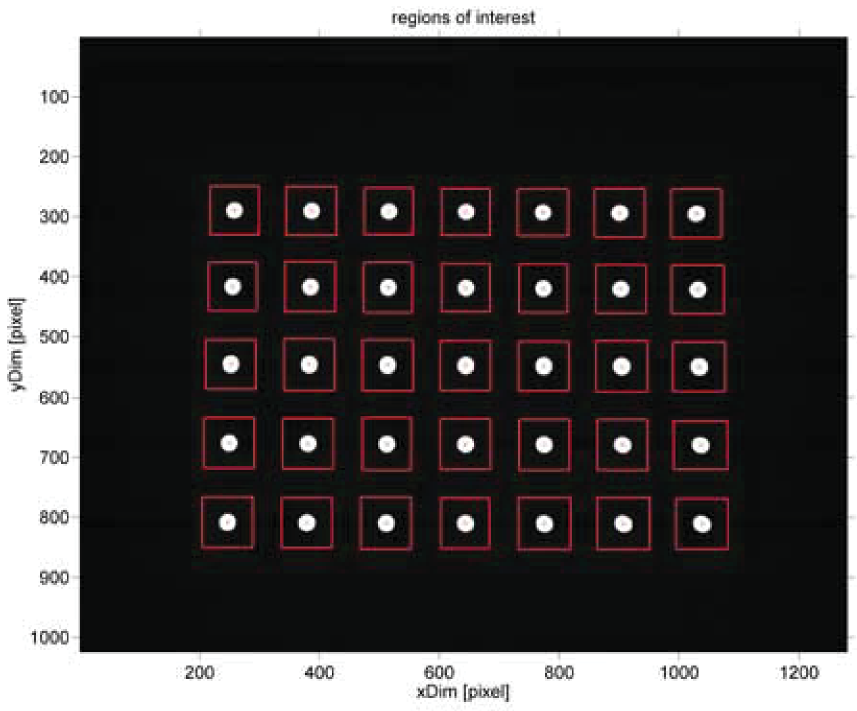


Figure 8.4: Validation image utilizing a lens of $f = 12mm$.

Center Determination via Center of Gravity

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d = 1$	0.0425	0.0339	0.0603	0.00130
polyCtrlPts	$d = 2$	0.0466	0.0419	0.0695	0.00120
	$d = 4$	0.0069	0.0084	0.0122	0.00035
tensorBFS	$d = 2$	0.0303	0.0011	0.0308	0.00100
	$d = 4$	0.0043	0.0003	0.0046	0.00034
	$d = 6$	0.0006	0.0026	0.0031	0.00030
tensorQDT	$d = 2$	0.0299	0.0013	0.0304	0.00099
	$d = 4$	0.0036	0.0004	0.0039	0.00027
	$d = 6$	0.0014	0.0029	0.0040	0.00025

Table 8.2: Remapping results utilizing $f = 12mm$ and center of gravity.

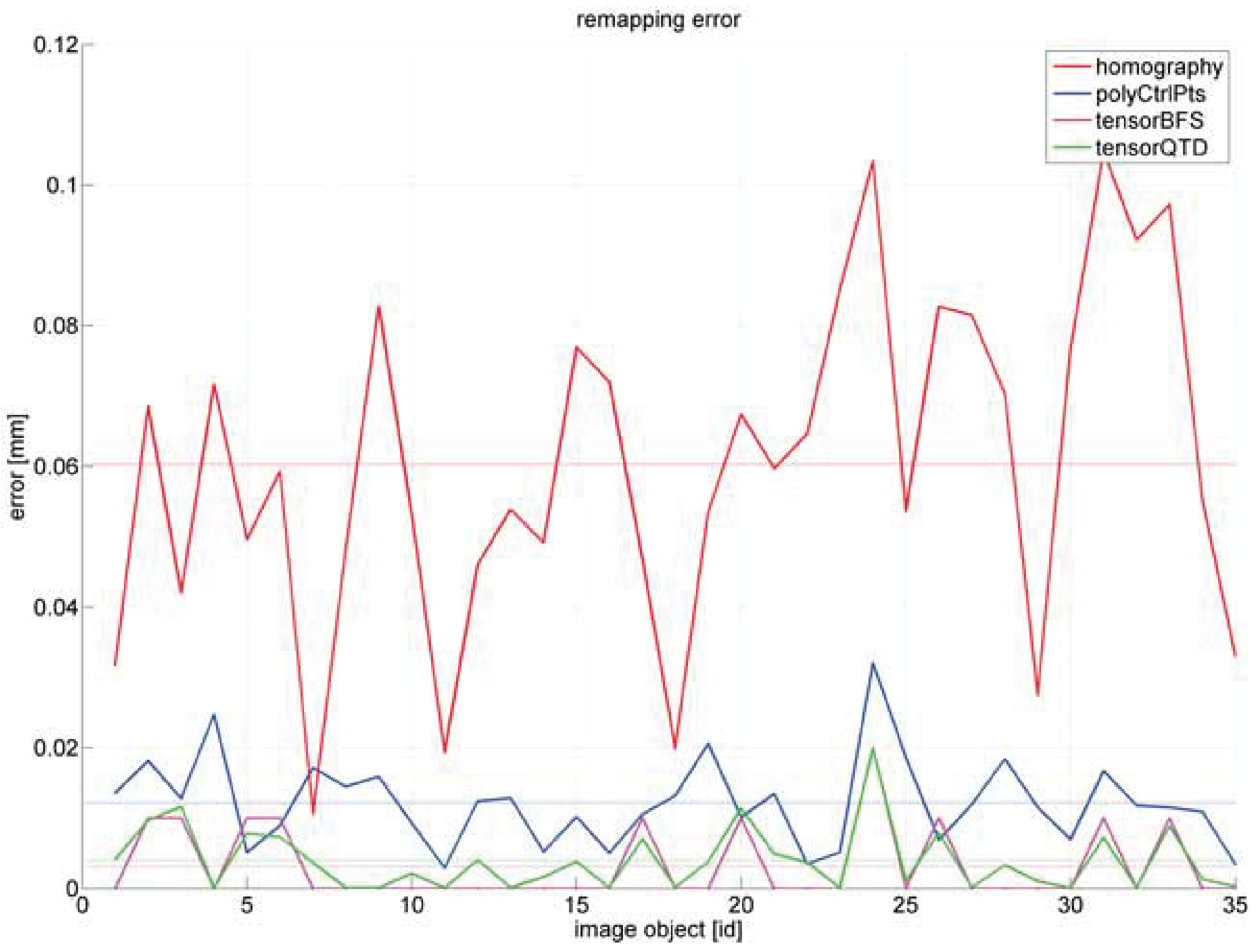


Figure 8.5: Remapping error for $f = 12mm$ and center of gravity.

Center Determination via Gaussian Curve Extremum

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d=1$	0.0442	0.0376	0.0650	0.00151
polyCtrlPts	$d=2$	0.0484	0.0393	0.0698	0.00147
	$d=4$	0.0106	0.0118	0.0176	0.00055
tensorBFS	$d=2$	0.0317	0.0109	0.0364	0.00129
	$d=4$	0.0051	0.0063	0.0102	0.00061
	$d=6$	0.0031	0.0014	0.0046	0.00039
tensorQDT	$d=2$	0.0320	0.0102	0.0361	0.00125
	$d=4$	0.0046	0.0065	0.0099	0.00055
	$d=6$	0.0036	0.0016	0.0050	0.00039

Table 8.3: Remapping results utilizing $f = 12mm$ and Gaussian curve extremum.

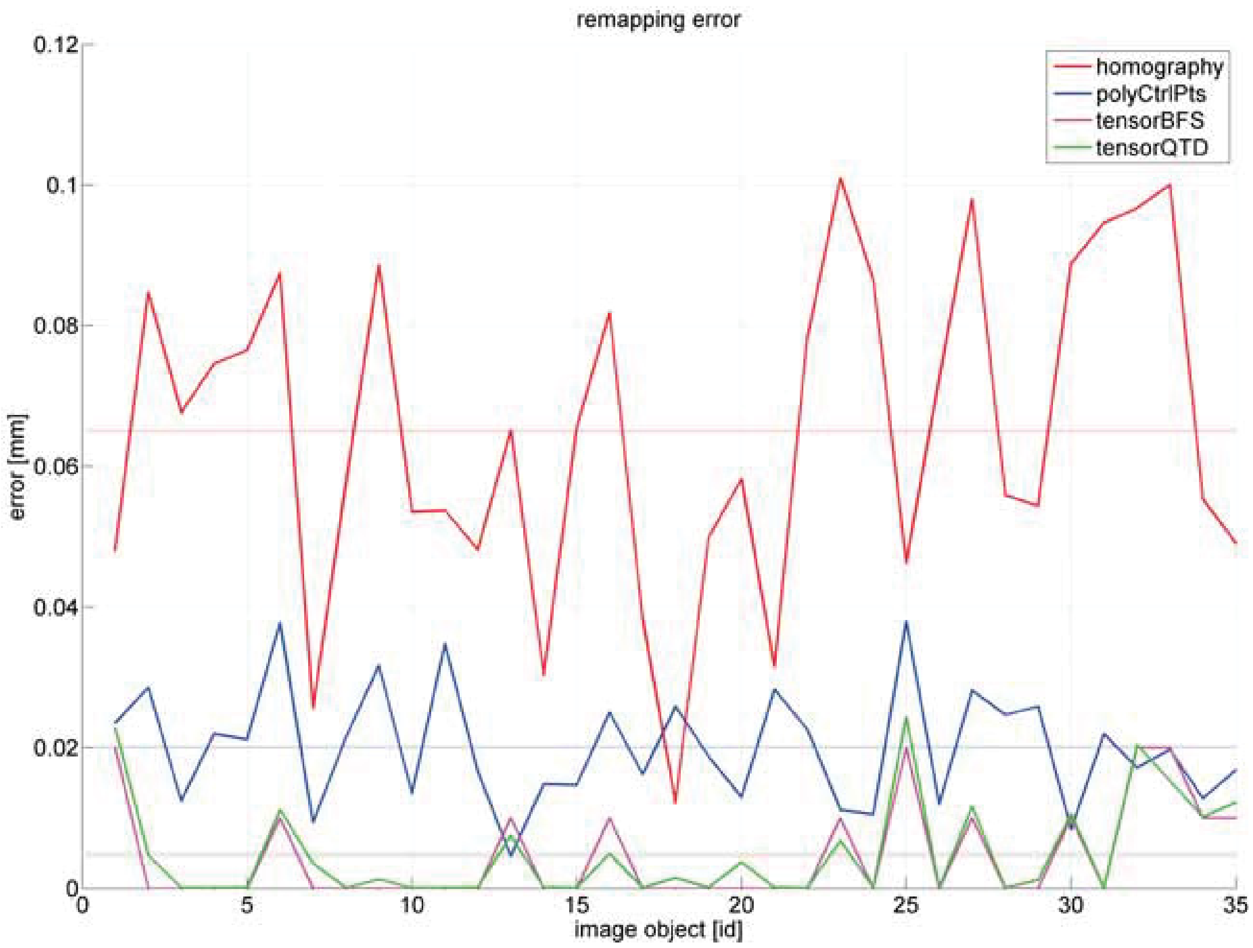


Figure 8.6: Remapping error for $f = 12mm$ and Gaussian curve extremum.

Center Determination via Ellipse Approximation

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d = 1$	0.0447	0.0386	0.0653	0.00117
polyCtrlPts	$d = 2$	0.0469	0.0378	0.0677	0.00136
	$d = 4$	0.0069	0.0157	0.0182	0.00043
tensorBFS	$d = 2$	0.0300	0.0071	0.0334	0.00106
	$d = 4$	0.0046	0.0009	0.0053	0.00035
	$d = 6$	0.0020	0.0000	0.0020	0.00023
tensorQDT	$d = 2$	0.0301	0.0071	0.0333	0.00107
	$d = 4$	0.0041	0.0010	0.0048	0.00030
	$d = 6$	0.0026	0.0003	0.0028	0.00019

Table 8.4: Remapping results utilizing $f = 12mm$ and ellipse approximation.

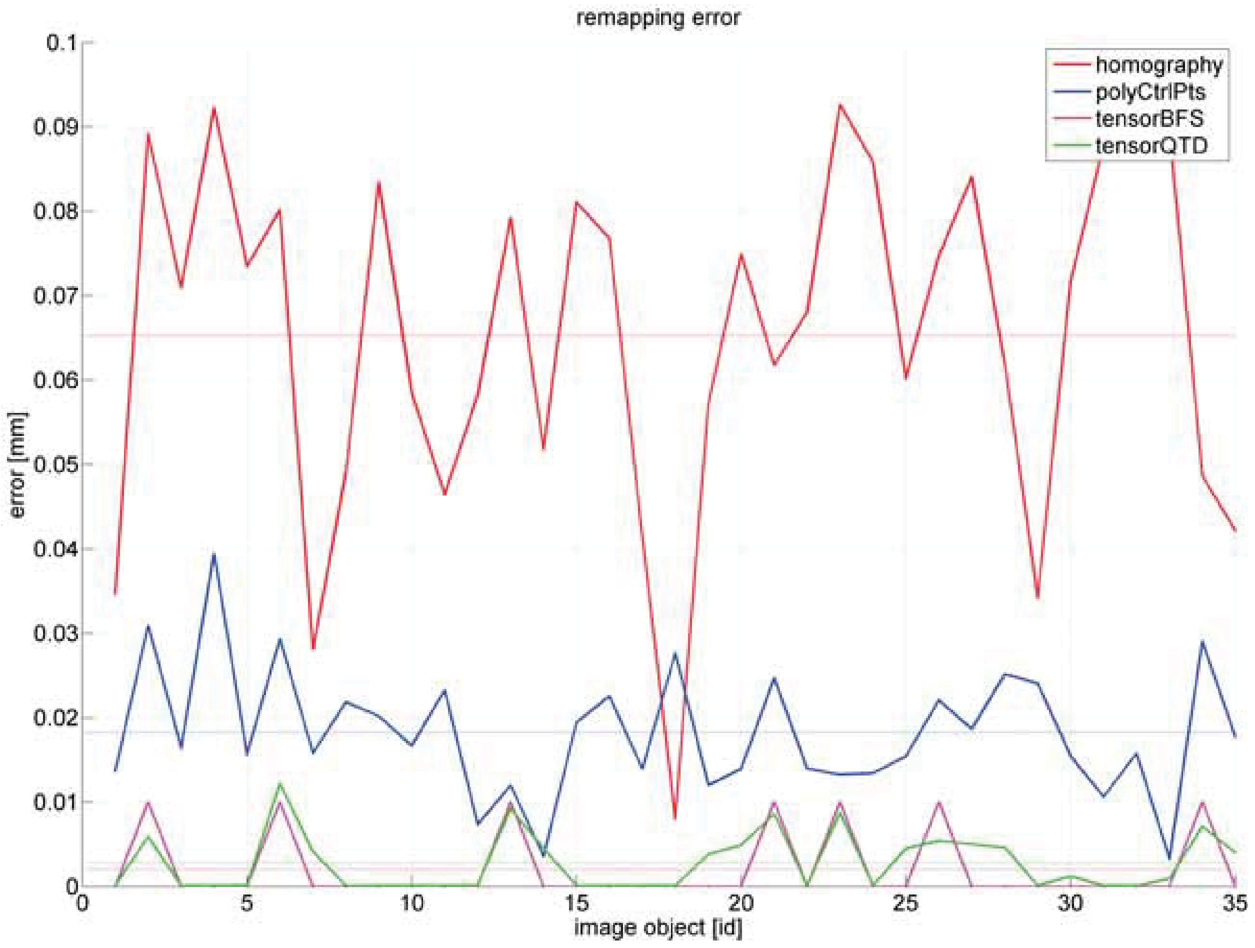


Figure 8.7: Remapping error for $f = 12mm$ and ellipse approximation.

Center Determination via Circle Approximation

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d=1$	0.0446	0.0402	0.0661	0.00110
polyCtrlPts	$d=2$	0.0468	0.0379	0.0678	0.00136
	$d=4$	0.0082	0.0204	0.0232	0.00045
tensorBFS	$d=2$	0.0303	0.0060	0.0329	0.00106
	$d=4$	0.0037	0.0006	0.0043	0.00031
	$d=6$	0.0034	0.0000	0.0034	0.00030
tensorQDT	$d=2$	0.0301	0.0068	0.0329	0.00107
	$d=4$	0.0039	0.0007	0.0043	0.00028
	$d=6$	0.0034	0.0001	0.0034	0.00024

Table 8.5: Remapping results utilizing $f = 12mm$ and circle approximation.

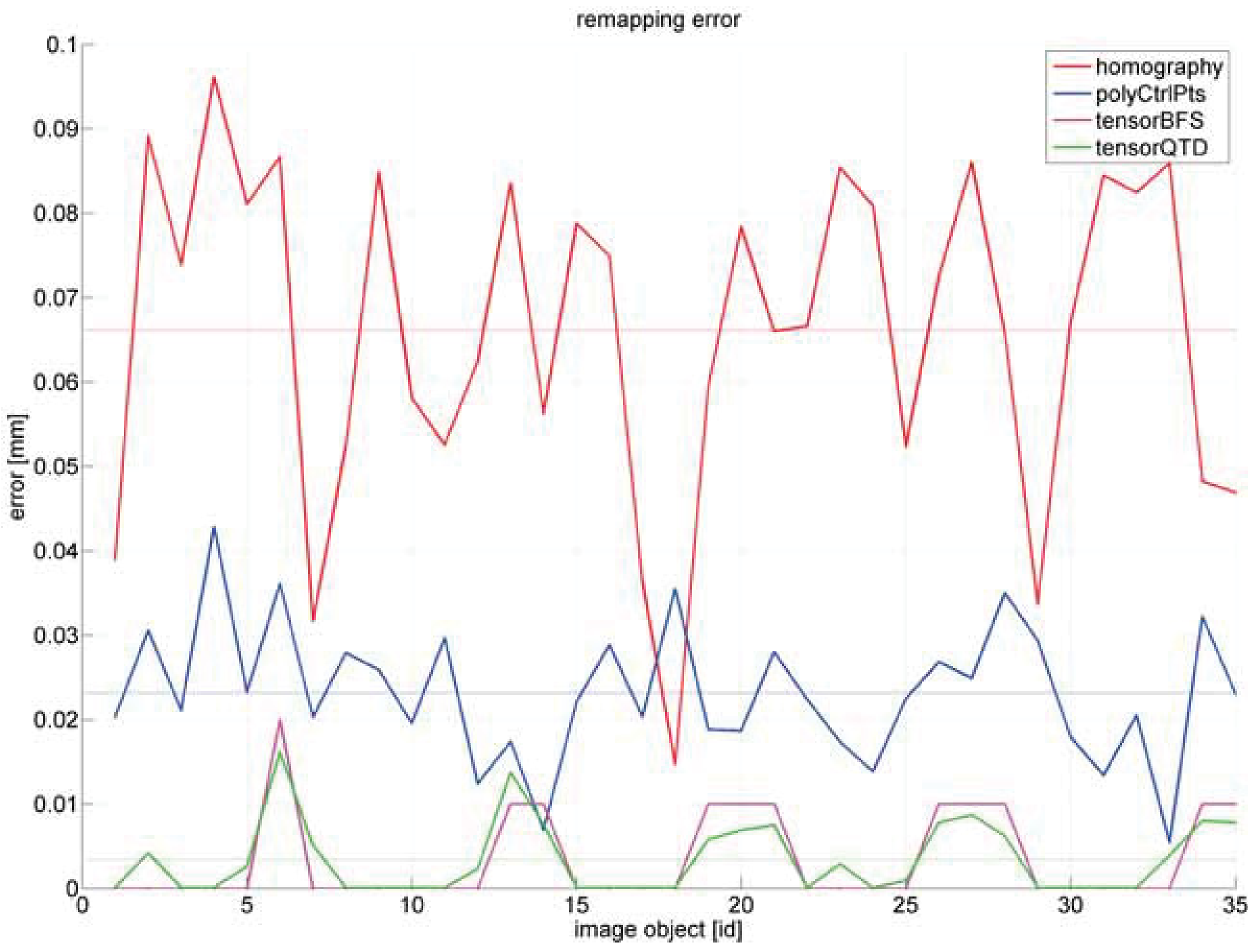


Figure 8.8: Remapping error for $f = 12mm$ and circle approximation.

8.1.2 Focal Length of $f = 3.6mm$

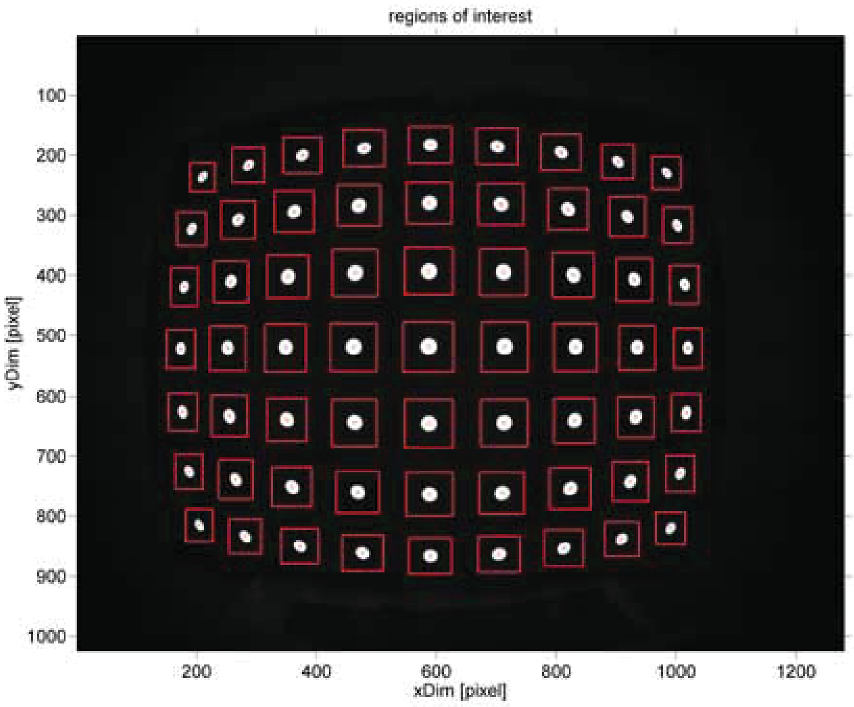


Figure 8.9: Calibration image utilizing $f = 3.6mm$.

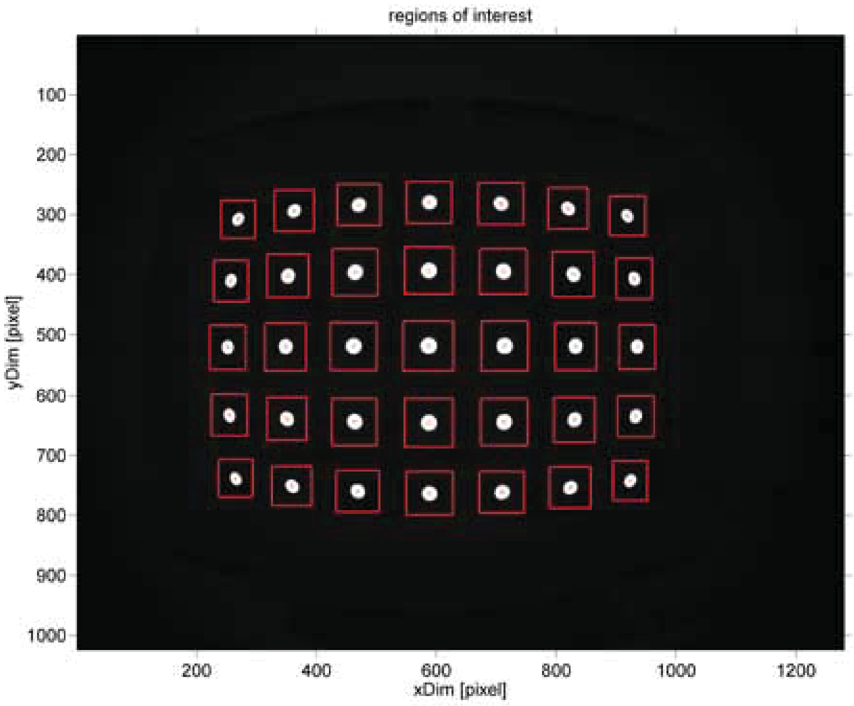


Figure 8.10: Validation image utilizing $f = 3.6mm$.

Center Determination via Center of Gravity

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d = 1$	3.4908	2.1590	4.5597	0.08848
polyCtrlPts	$d = 2$	3.4588	2.1228	4.5473	0.08982
	$d = 4$	0.3092	0.2428	0.4391	0.01075
tensorBFS	$d = 2$	2.7960	1.4891	3.3613	0.06585
	$d = 4$	0.1160	0.0366	0.1375	0.00588
	$d = 6$	0.0063	0.0026	0.0083	0.00044
tensorQDT	$d = 2$	2.7956	1.4888	3.3607	0.06587
	$d = 4$	0.1154	0.0316	0.1314	0.00611
	$d = 6$	0.0061	0.0026	0.0080	0.00041

Table 8.6: Remapping results utilizing $f = 3.6mm$ and center of gravity.

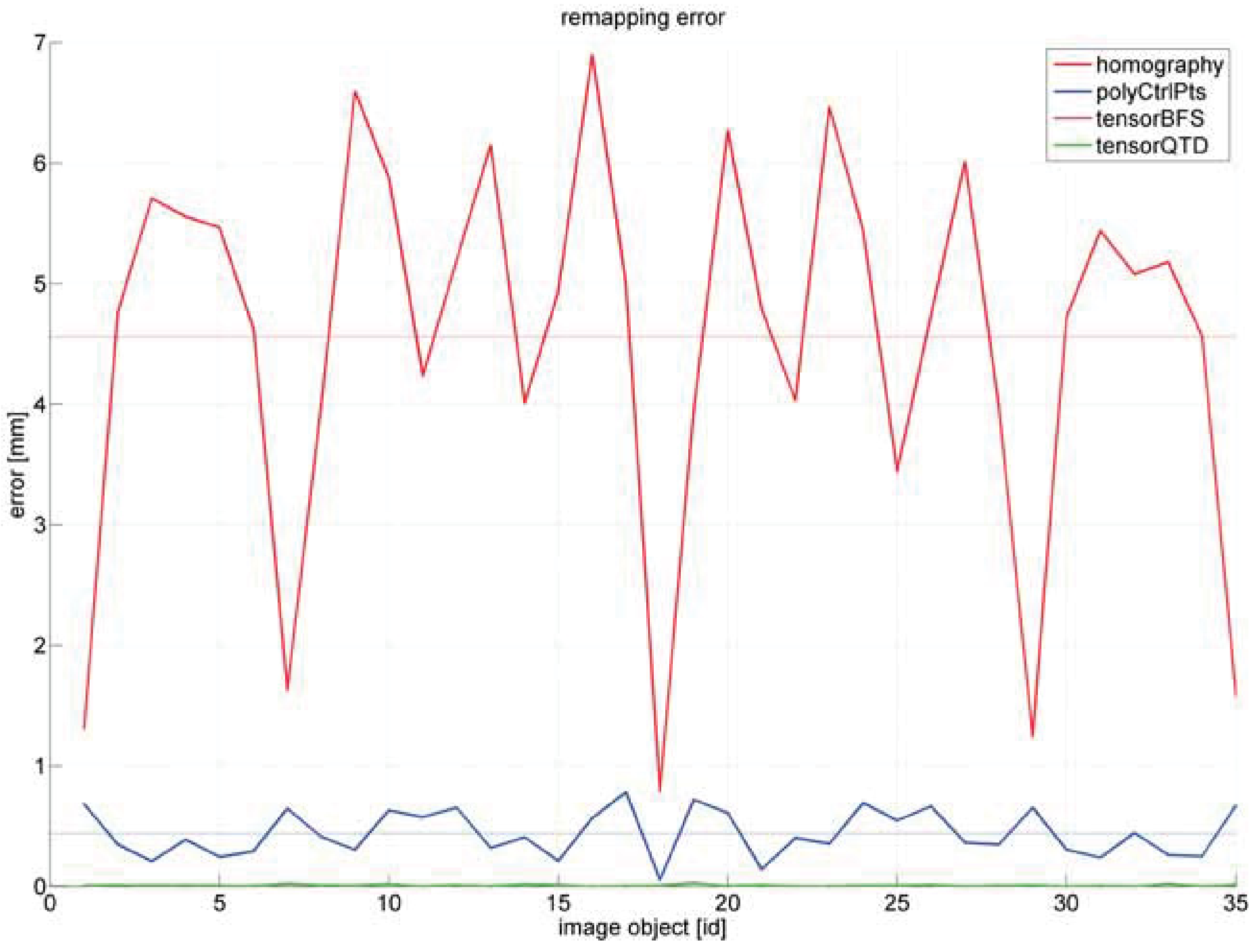


Figure 8.11: Remapping error for $f = 3.6mm$ and center of gravity.

Center Determination via Gaussian Curve Extremum

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d = 1$	3.5989	2.1635	4.6714	0.08389
polyCtrlPts	$d = 2$	3.4767	2.1226	4.5684	0.09017
	$d = 4$	0.4399	0.2471	0.5565	0.02084
tensorBFS	$d = 2$	2.8126	1.4880	3.3792	0.06506
	$d = 4$	0.1203	0.0380	0.1432	0.00584
	$d = 6$	0.1383	0.0206	0.1444	0.01793
tensorQDT	$d = 2$	2.8128	1.4876	3.3792	0.06509
	$d = 4$	0.1192	0.0316	0.1356	0.00613
	$d = 6$	0.1385	0.0204	0.1441	0.01800

Table 8.7: Remapping results utilizing $f = 3.6mm$ and Gaussian curve extremum.

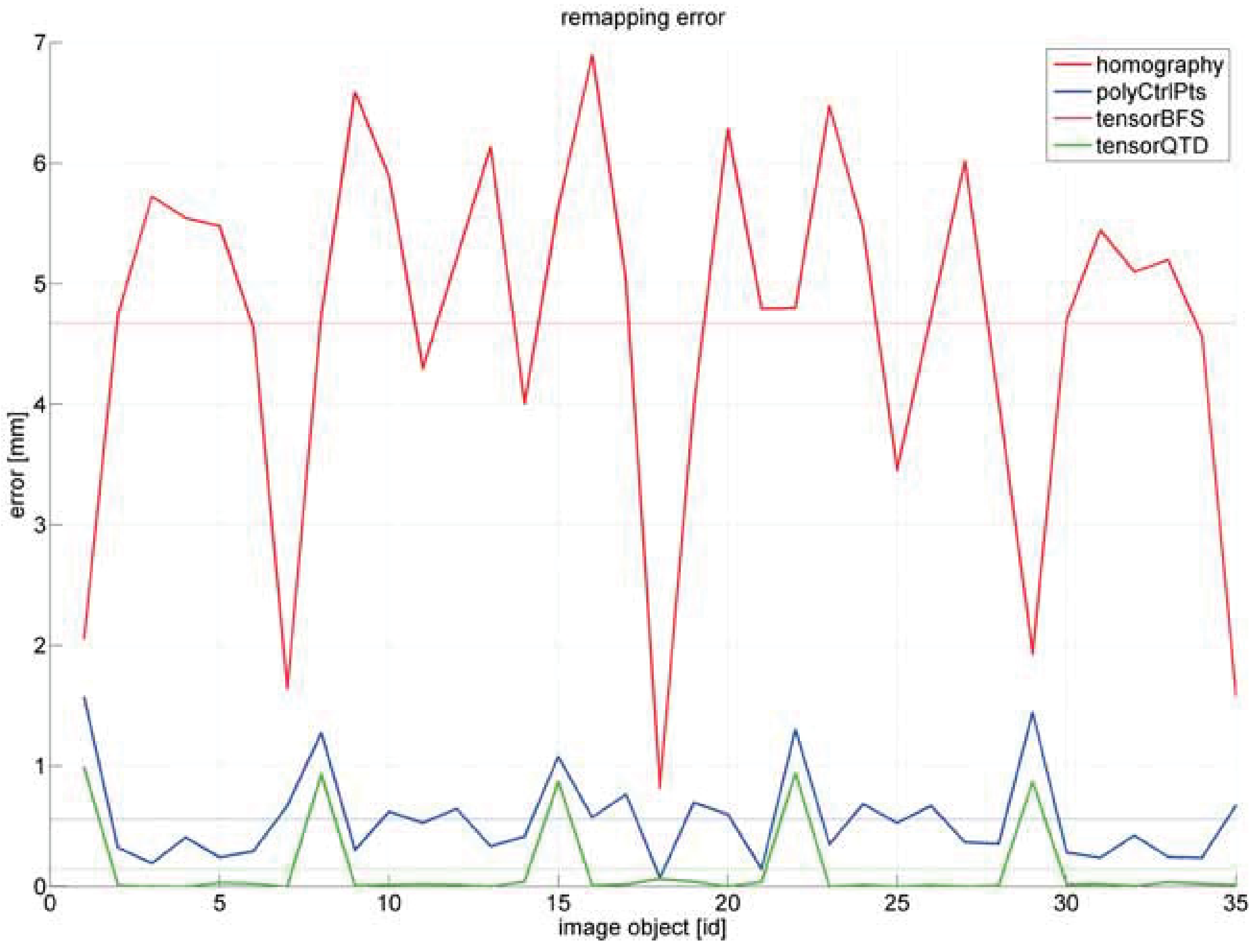


Figure 8.12: Remapping error for $f = 3.6mm$ and Gaussian curve extremum.

Center Determination via Ellipse Approximation

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d=1$	3.4920	2.1580	4.5597	0.08847
polyCtrlPts	$d=2$	3.4600	2.1214	4.5473	0.08980
	$d=4$	0.3085	0.2423	0.4393	0.01081
tensorBFS	$d=2$	2.7969	1.4894	3.3621	0.06585
	$d=4$	0.1160	0.0374	0.1379	0.00600
	$d=6$	0.0063	0.0014	0.0074	0.00036
tensorQDT	$d=2$	2.7966	1.4888	3.3614	0.06589
	$d=4$	0.1154	0.0316	0.1316	0.00626
	$d=6$	0.0067	0.0020	0.0079	0.00034

Table 8.8: Remapping results utilizing $f = 3.6mm$ and ellipse approximation.

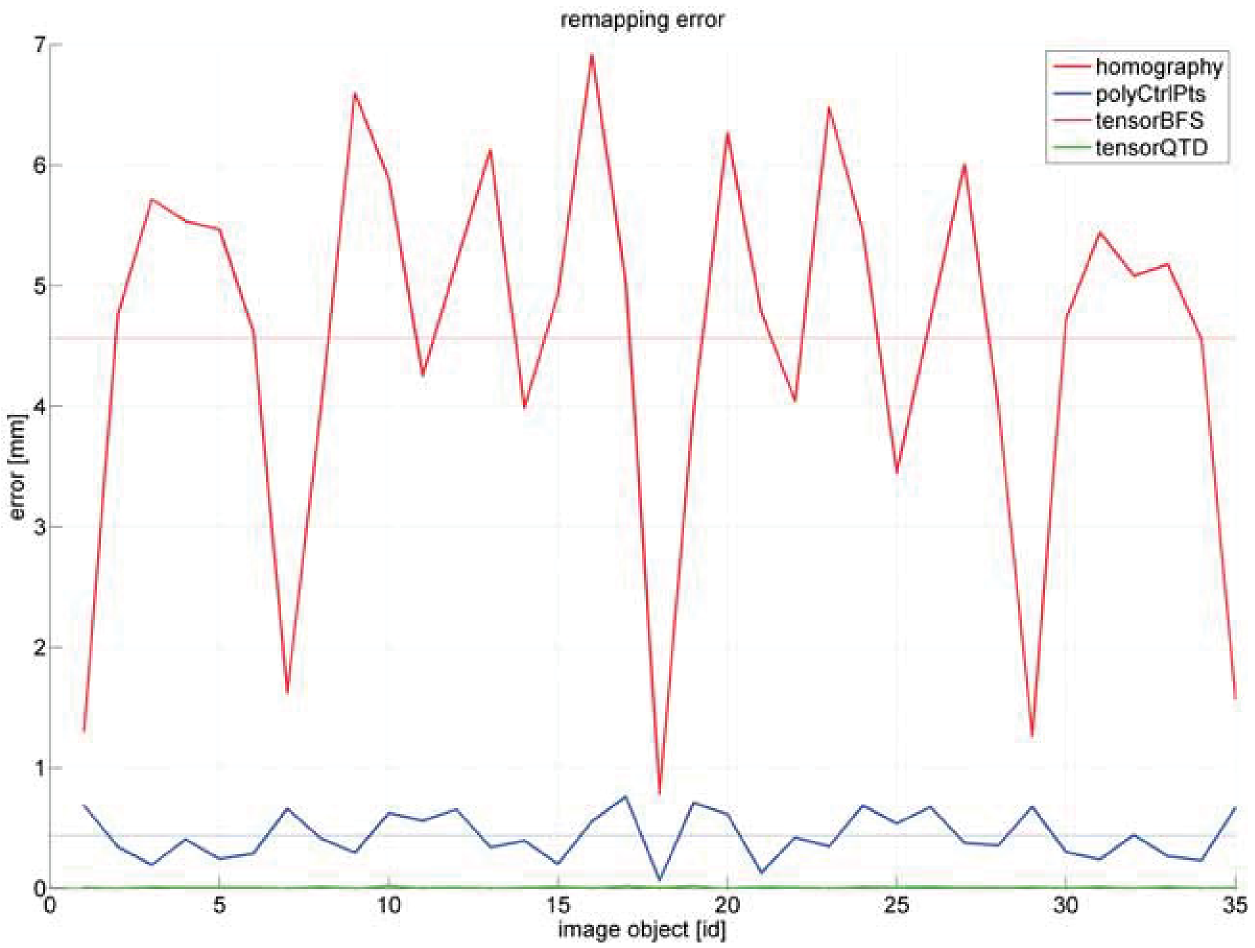


Figure 8.13: Remapping error for $f = 3.6mm$ and ellipse approximation.

Center Determination via Circle Approximation

method:	degree	$\bar{e}_x/[mm]$	$\bar{e}_y/[mm]$	$\bar{e}_{xy}/[mm]$	$\pm e_{95\%}/[mm]$
homography	$d=1$	3.4907	2.1580	4.5593	0.08843
polyCtrlPts	$d=2$	3.4587	2.1211	4.5470	0.08980
	$d=4$	0.3101	0.2418	0.4400	0.01079
tensorBFS	$d=2$	2.7960	1.4891	3.3620	0.06586
	$d=4$	0.1157	0.0374	0.1382	0.00609
	$d=6$	0.0020	0.0020	0.0094	0.00064
tensorQDT	$d=2$	2.7960	1.4883	3.3615	0.06593
	$d=4$	0.1144	0.0316	0.1310	0.00607
	$d=6$	0.0078	0.0022	0.0093	0.00059

Table 8.9: Remapping results utilizing $f = 3.6mm$ and circle approximation.

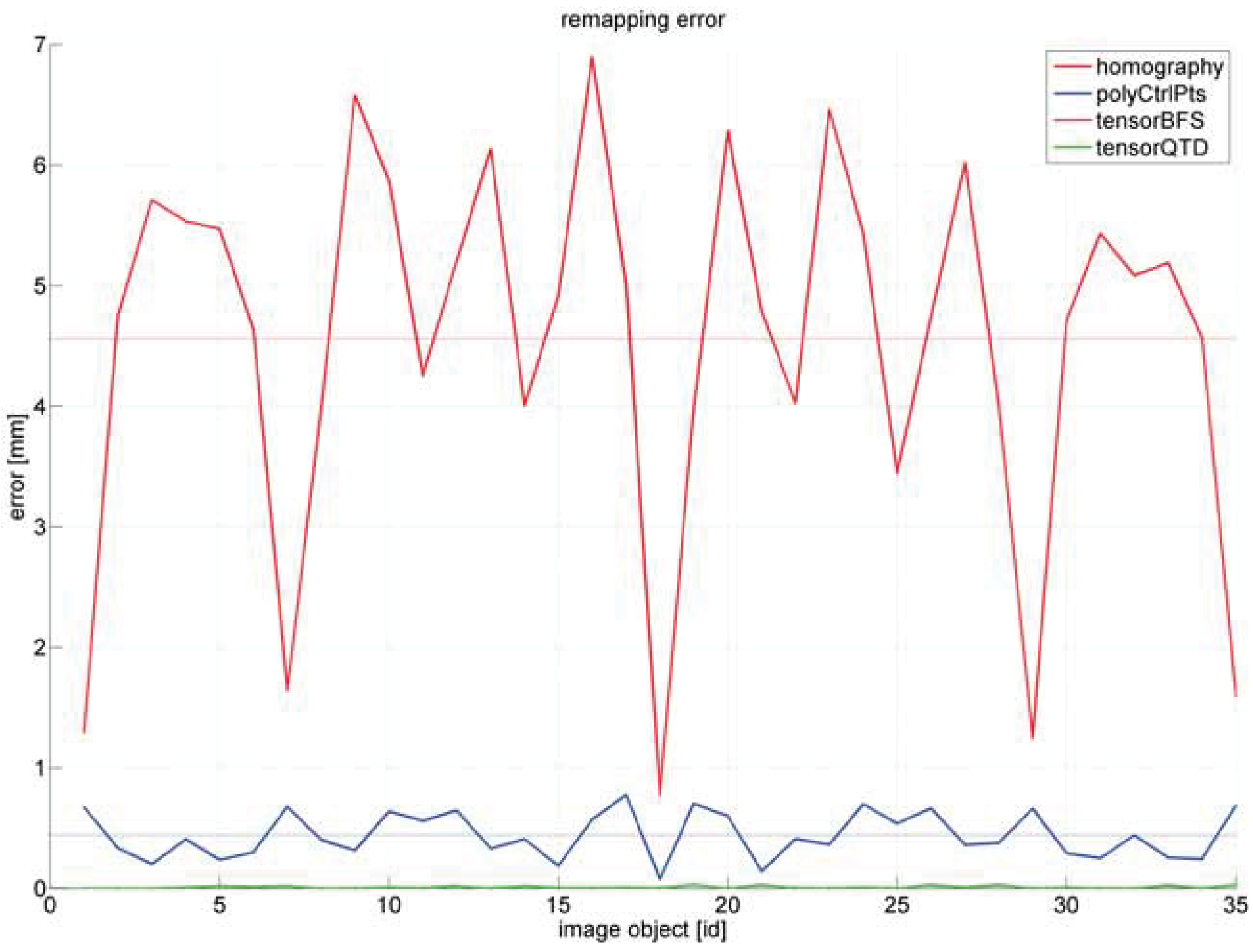


Figure 8.14: Remapping error for $f = 3.6mm$ and circle approximation.

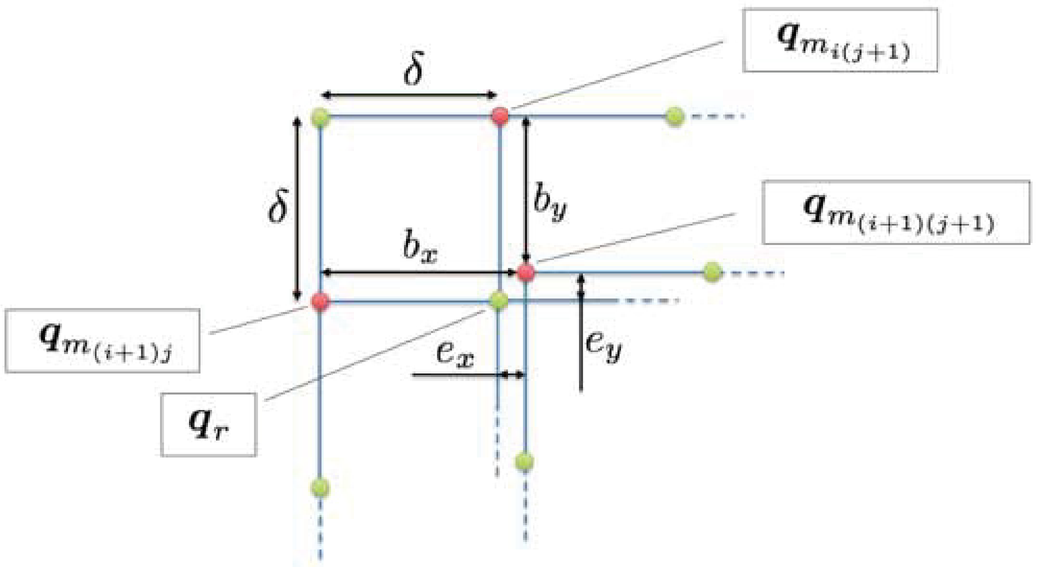
8.2 Calibration Modes

The system can either be adjusted with the help of a calibration plate with LEDs or by utilizing a xy-table with a mounted laser, which generates a laser spot calibration matrix. Figure 8.16 visualizes the resulting matrices. The experiment examines how the source of calibration data influences the system's output. Therefore, an interlaced cross validation $(m \times n)$ matrix is produced for evaluation, but unlike in the previous section, not each point's location is investigated. Instead, the differences b_x and b_y between two measured points $\boldsymbol{q}_m$ are acquired for every row and for every column, thus

$$b_x = x_{j+1} - x_j \qquad \text{and} \qquad b_y = y_{i+1} - y_i, \tag{8.5}$$

when i is the indexing variable for rows and j is the indexing variable for columns. The cross validation matrix features a $\delta = \delta_x = \delta_y = 25mm$ raster in x and y dimensions, yielding a raster point $\boldsymbol{q}_r$ relatively to every $\boldsymbol{q}_m$. The errors e_{x_j} and e_{y_i} are the differences between b and δ for every pair of neighboring points, i.e.

$$e_{x_j} = \delta - b_{x_j} \qquad \text{and} \qquad e_{y_i} = \delta - b_{y_i}. \tag{8.6}$$



δ
$\boldsymbol{q}_{m_{i(j+1)}}$
δ
b_y
$\boldsymbol{q}_{m_{(i+1)(j+1)}}$
b_x
$\boldsymbol{q}_{m_{(i+1)j}}$
e_x
e_y
$\boldsymbol{q}_r$

Figure 8.15: The green dots indicate the resulting raster points $\boldsymbol{q}_r$ relatively to every measured point $\boldsymbol{q}_m$, which are both separated by $\delta = \delta_x = \delta_y = 25mm$. The errors e_x and e_y are the differences between δ and b_x respectively b_y.

The errors e_x and e_y of the cross validation are examined separately for rows and columns. The number of inspected image object distances ε is

$$\varepsilon_x = m \cdot (n-1) \qquad \text{and} \qquad \varepsilon_y = (m-1) \cdot n. \tag{8.7}$$

Thus, the computation of the mean error $\bar{e}_x$, is carried out for all rows m, but only $(n-1)$ columns:

$$\bar{e}_x = \frac{1}{\varepsilon_x} \sum_{i=1}^{m} \sum_{j=1}^{n-1} \left| \delta - (x_{i(j+1)} - x_{ij}) \right| . \tag{8.8}$$

Consequently, $\bar{e}_y$ is calculated for $(m-1)$ rows and n columns:

$$\bar{e}_y = \frac{1}{\varepsilon_y} \sum_{i=1}^{m-1} \sum_{j=1}^{n} \left| \delta - (y_{(i+1)j} - y_{ij}) \right| . \tag{8.9}$$

Hence, for a $(m \times n) = (4 \times 8)$ cross validation matrix, there are $\varepsilon_x = 4 \cdot 7 = 28$ values for b_x and $\varepsilon_y = 3 \cdot 8 = 24$ values for b_y. The reason for the separate testing of the x and y errors is that $\varepsilon_x \neq \varepsilon_y$. The standard error $e_{95\%}$ is computed for the investigated mean values $\bar{e}_x$ and $\bar{e}_y$ for ε_x respectively ε_y elements.

The non-linear coordinate mapping functions use a degree of $d = 4$, because for a lens with $f = 12mm$, this delivers the best solutions. Once again, the computations are carried out for all center determination methods. The detailed results for the cross validation can be found in Appendix E.

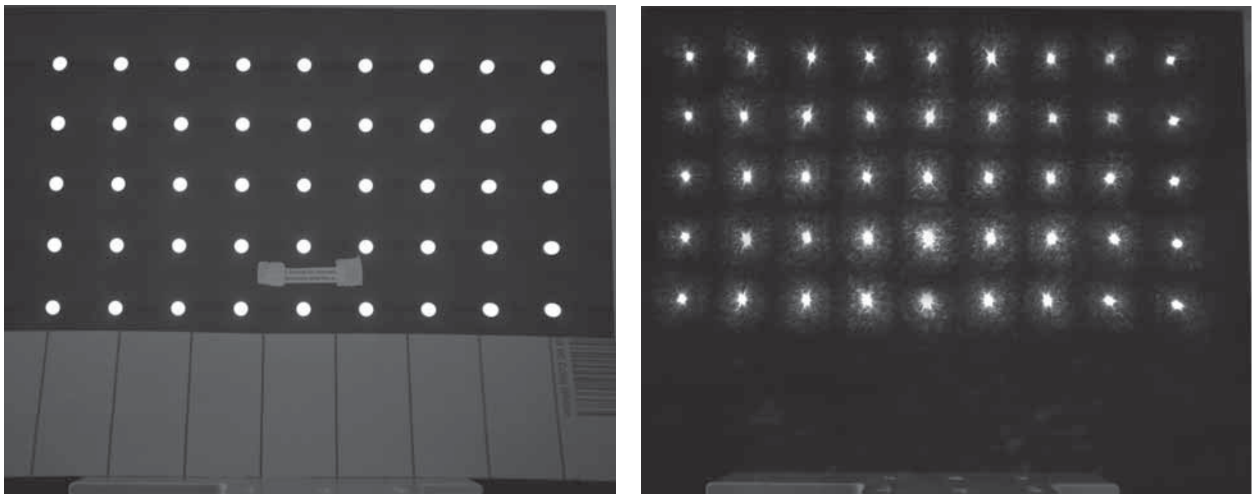

(a) Calibration with LEDs. (b) Calibration with laser spots.

Figure 8.16: The system can either be calibrated by utilizing a calibration plate with mounted LEDs or by using a laser, which is mounted on a xy-table, to generate a laser spot matrix. Only 45 calibration points are used during this experiment because of hardware limitations.

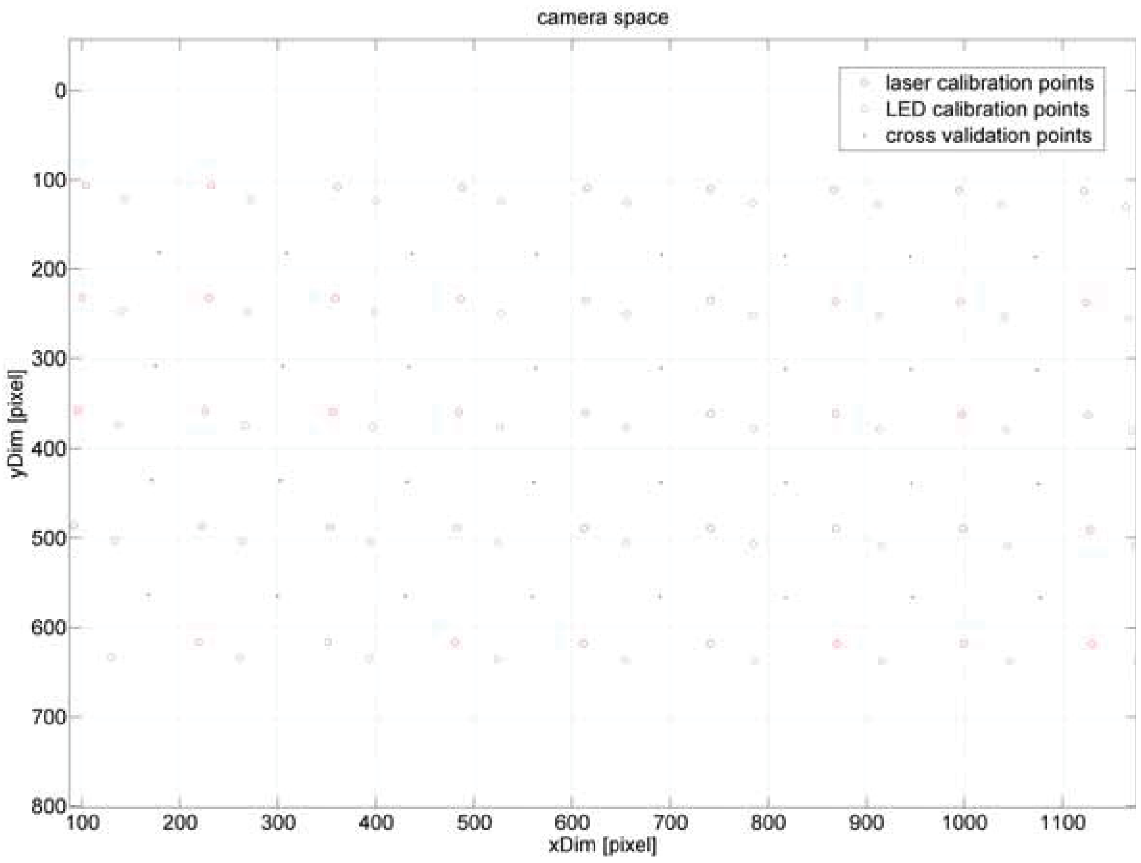


Figure 8.17: The plot shows the camera space generated using both calibration modes. The matrix has $(5 \times 9) = 45$ calibration points. To enable the evaluation of the reached quality, a (4×8) interlaced cross validation matrix is generated with the xy-table.

Center Determination via Center of Gravity

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d = 1$	0.1748	0.0124	0.1416	0.0102
polyCtrlPts	$d = 4$	0.2016	0.0145	0.1466	0.0103
tensorBFS	$d = 4$	0.2000	0.0143	0.1438	0.0101
tensorQDT	$d = 4$	0.2009	0.0144	0.1448	0.0101

Table 8.10: Calibration evaluation utilizing LEDs and center of gravity.

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d = 1$	0.1321	0.0122	0.0956	0.0094
polyCtrlPts	$d = 4$	0.0858	0.0069	0.0441	0.0045
tensorBFS	$d = 4$	0.0864	0.0070	0.0458	0.0048
tensorQDT	$d = 4$	0.0856	0.0069	0.0445	0.0047

Table 8.11: Calibration evaluation utilizing laser spots and center of gravity.

Center Determination via Gaussian Curve Extremum

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d = 1$	0.1540	0.0133	0.1083	0.0096
polyCtrlPts	$d = 4$	0.1714	0.0146	0.1156	0.0096
tensorBFS	$d = 4$	0.1725	0.0147	0.1150	0.0097
tensorQDT	$d = 4$	0.1673	0.0142	0.1138	0.0095

Table 8.12: Calibration evaluation utilizing LEDs and Gaussian curve extremum.

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d = 1$	0.1434	0.0133	0.1057	0.0104
polyCtrlPts	$d = 4$	0.1130	0.0104	0.0927	0.0097
tensorBFS	$d = 4$	0.1154	0.0111	0.1258	0.0124
tensorQDT	$d = 4$	0.1156	0.0111	0.1261	0.0123

Table 8.13: Calibration evaluation utilizing laser spots and Gaussian curve extremum.

Center Determination via Ellipse Approximation

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d=1$	0.1731	0.0122	0.1478	0.0107
polyCtrlPts	$d=4$	0.1998	0.0143	0.1569	0.0111
tensorBFS	$d=4$	0.2000	0.0143	0.1567	0.0110
tensorQDT	$d=4$	0.2010	0.0143	0.1553	0.0109

Table 8.14: Calibration evaluation utilizing LEDs and ellipse approximation.

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d=1$	0.1299	0.0121	0.1032	0.0100
polyCtrlPts	$d=4$	0.0816	0.0066	0.0542	0.0053
tensorBFS	$d=4$	0.0821	0.0067	0.0504	0.0052
tensorQDT	$d=4$	0.0820	0.0067	0.0510	0.0052

Table 8.15: Calibration evaluation utilizing laser spots and ellipse approximation.

Center Determination via Circle Approximation

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d=1$	0.1738	0.0124	0.1493	0.0110
polyCtrlPts	$d=4$	0.2004	0.0145	0.1584	0.0113
tensorBFS	$d=4$	0.2000	0.0145	0.1592	0.0113
tensorQDT	$d=4$	0.2005	0.0145	0.1574	0.0112

Table 8.16: Calibration evaluation utilizing LEDs and circle approximation.

method:	degree	$\bar{e}_x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$\bar{e}_y/[mm]$	$\pm e_{y_{95\%}}/[mm]$
homography	$d=1$	0.1337	0.0123	0.1066	0.0102
polyCtrlPts	$d=4$	0.0848	0.0070	0.0564	0.0055
tensorBFS	$d=4$	0.0861	0.0072	0.0550	0.0055
tensorQDT	$d=4$	0.0853	0.0071	0.0540	0.0054

Table 8.17: Calibration evaluation utilizing laser spots and circle approximation.

8.3 Laser Spot Measurements

The goal of this scenario is to determine how electro-active glass influences the outcome of the measurements. Three different setups are examined, which are visualized in Figure 8.21, 8.24 and 8.25. A linear drive, like in Figure 8.20, enables one-dimensional movement to produce an interlaced cross validation vector, i.e. a $(1 \times m) = (1 \times 8)$ matrix of 8 measured points $\boldsymbol{q}_m$. Each point is acquired $\rho = 30$ times, thus

$$x = \frac{1}{\rho}\sum_{i=1}^{\rho} x_i \qquad \text{as well as} \qquad y = \frac{1}{\rho}\sum_{i=1}^{\rho} y_i. \tag{8.10}$$

Their standard errors $e_{x_{95\%}}$ and $e_{y_{95\%}}$ are also computed in respect of the number of repetitions ρ. Like in the previous experiment, the distances b_i between a pair of points are evaluated. Consequently, the number of investigated image object distances is $\varepsilon = 1 \cdot (8-1) = 7$. Again, the relative distance of a measured point $\boldsymbol{q}_m$ and the next raster point $\boldsymbol{q}_r$ is $\delta = 25mm$. The distance b_i between two points $\boldsymbol{q}_{m_i}$ and $\boldsymbol{q}_{m_{i+1}}$ is computed by

$$b_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{8.11}$$

to compensate inexact alignment of the linear drive. The error e_i of each tested distance is the difference between the measured points $\boldsymbol{q}_m$ and the corresponding raster point $\boldsymbol{q}_r$, hence

$$e_i = \delta - b_i \tag{8.12}$$

and the overall mean error $\bar{e}$ is

$$\bar{e} = \frac{1}{\varepsilon}\sum_{i=1}^{\varepsilon} |e_i|. \tag{8.13}$$

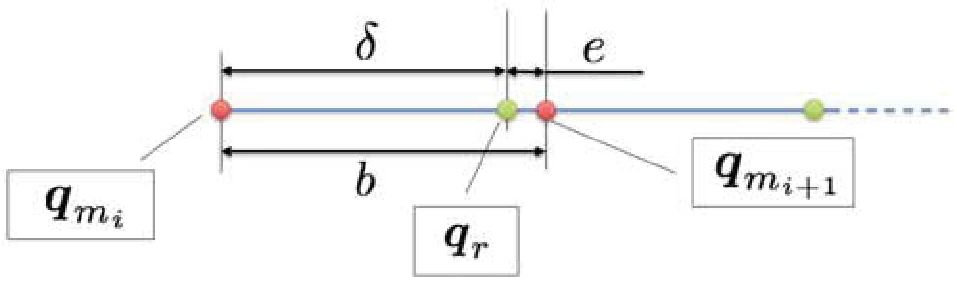
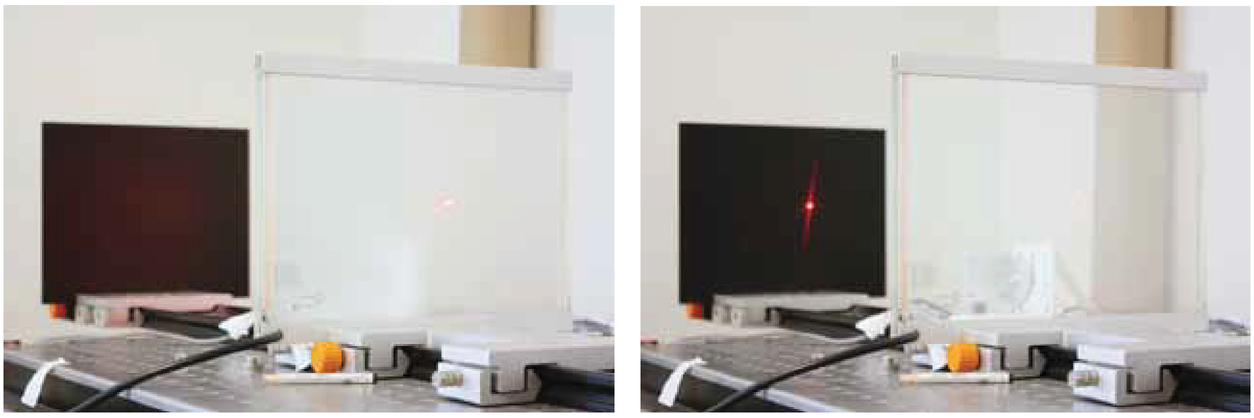


Figure 8.18: The green dots indicate the raster points $\boldsymbol{q}_r$, which are separated by $\delta = 25mm$ from their corresponding measured point $\boldsymbol{q}_m$. The error e is the difference between δ and b.



(a) Metal target with opaque glass. (b) Metal target with transparent glass.

Figure 8.19: It's tested, if a transparent glass between the laser and the target influences the outcome. The issue is important, because this feature allows the installation of a series of measurement stations in a consecutive order.

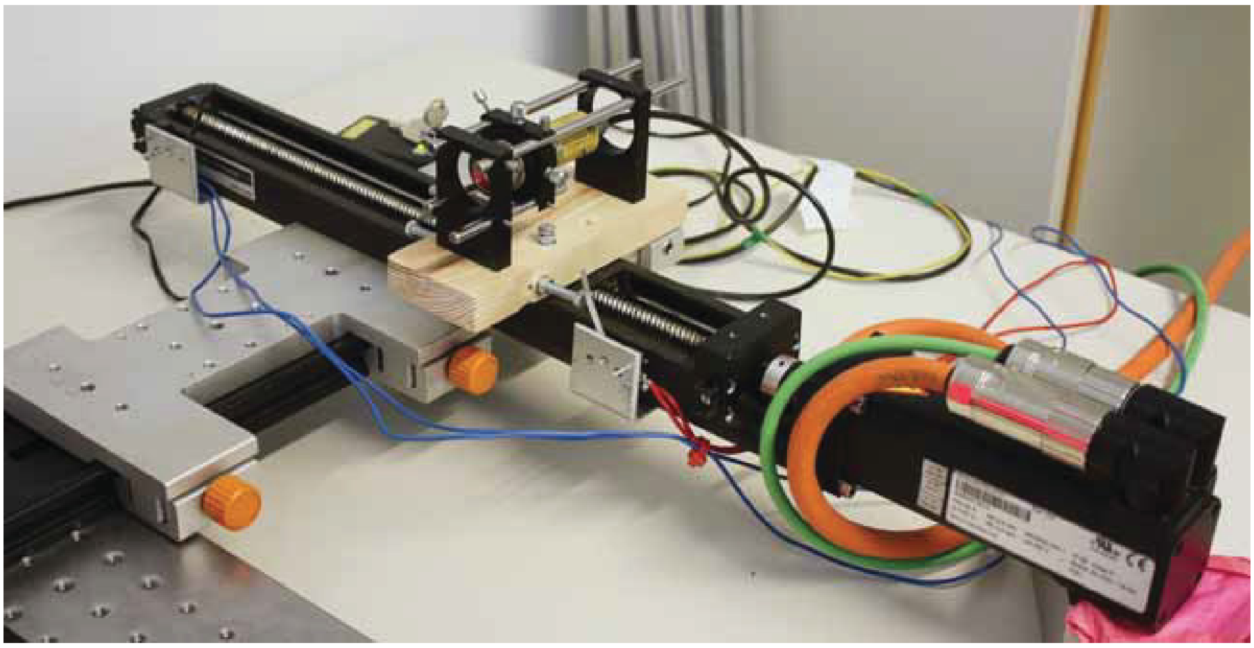


Figure 8.20: The laser is mounted on the linear drive's slide, which allows it to perform one-dimensional movements. It's controlled via a PLC.

The deployed degree of non-linear coordinate mapping functions is $d = 4$ for a lens of $f = 12mm$. The used center determination method is ellipse approximation, because this technique delivers the best results as proofed in the previous experiments.

8.3.1 Metal Target without Glass

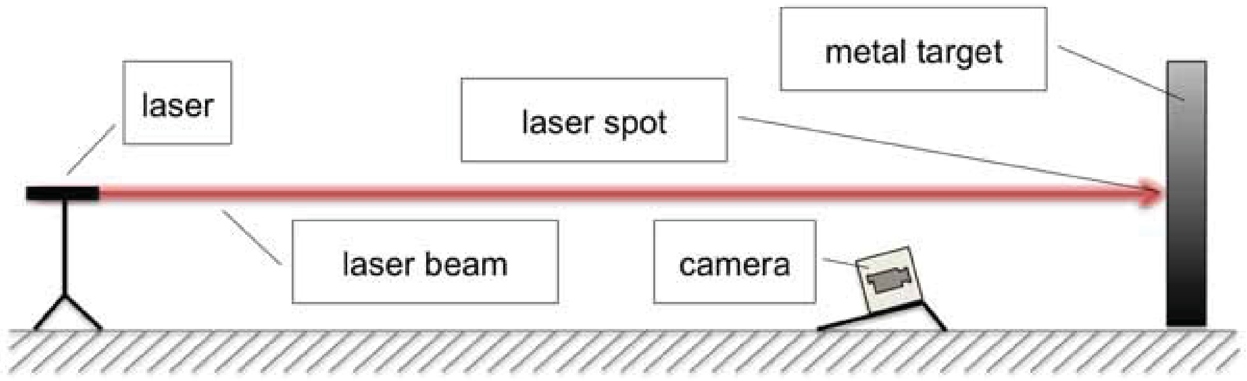


Figure 8.21: The standard setup for the measurements features a metal target.

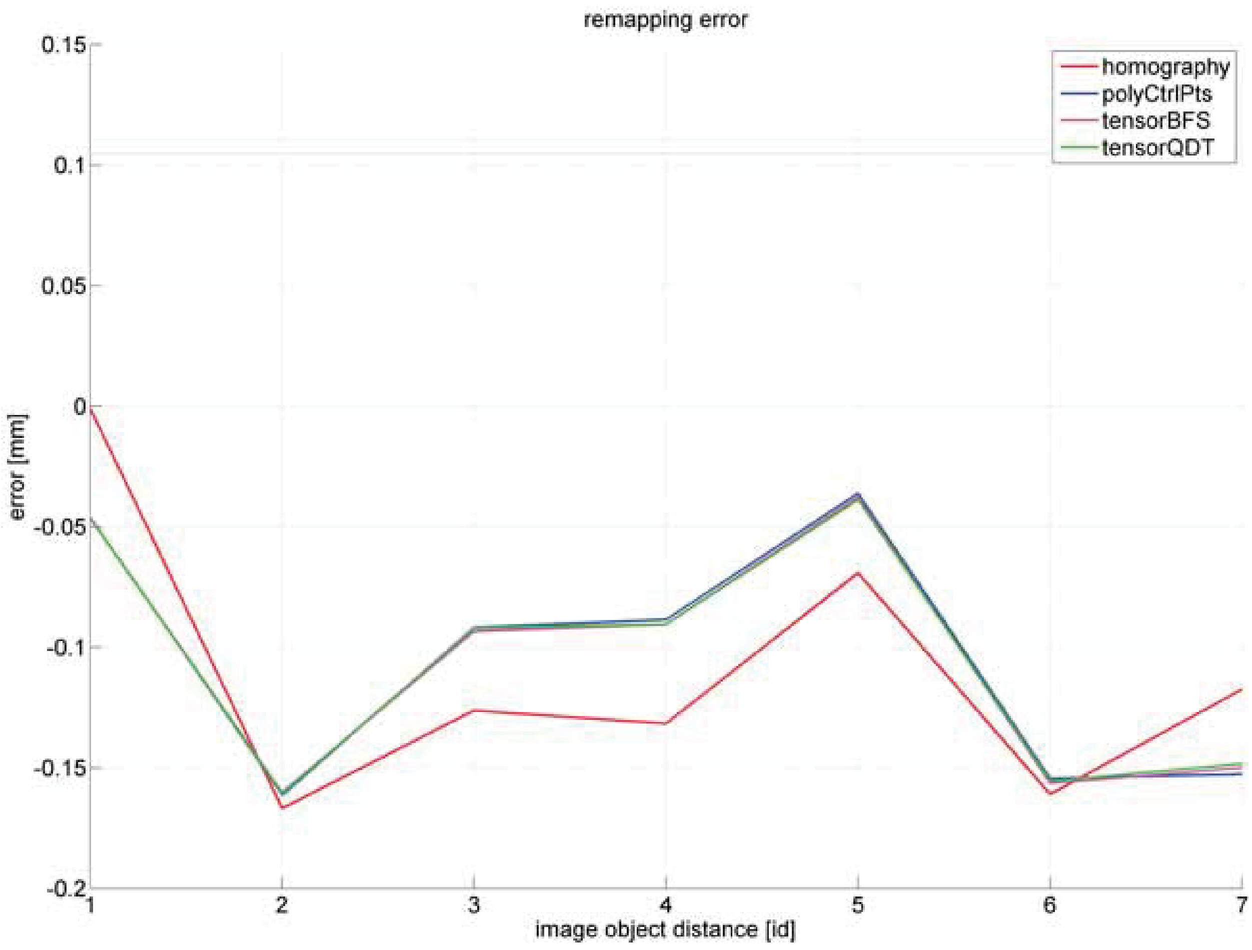


Figure 8.22: Resulting error on a metal target without glass.

Linear Homography

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.9404	0.0022	66.4875	0.0005	25.0010	-0.00098
$\boldsymbol{q}_2$	33.9393	0.0011	66.1650	0.0002	25.1668	-0.16679
$\boldsymbol{q}_3$	59.1023	0.0022	65.7260	0.0004	25.1262	-0.12617
$\boldsymbol{q}_4$	84.2265	0.0025	65.4109	0.0002	25.1316	-0.13159
$\boldsymbol{q}_5$	109.3553	0.0017	65.0365	0.0005	25.0692	-0.06923
$\boldsymbol{q}_6$	134.4231	0.0017	64.7731	0.0005	25.1610	-0.16099
$\boldsymbol{q}_7$	159.5806	0.0035	64.3538	0.0005	25.1174	-0.11743
$\boldsymbol{q}_8$	184.6943	0.0004	63.9166	0.0007		

Table 8.18: Homography error on metal target without glass, $\bar{e} = 0.1105mm$.

Polynomial Control Points

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.9600	0.0022	66.5005	0.0005	25.0462	-0.04622
$\boldsymbol{q}_2$	34.0040	0.0011	66.1705	0.0002	25.1613	-0.16133
$\boldsymbol{q}_3$	59.1614	0.0022	65.7242	0.0004	25.0918	-0.09179
$\boldsymbol{q}_4$	84.2512	0.0024	65.4068	0.0002	25.0886	-0.08856
$\boldsymbol{q}_5$	109.3370	0.0016	65.0367	0.0005	25.0362	-0.03621
$\boldsymbol{q}_6$	134.3719	0.0016	64.7825	0.0005	25.1544	-0.15441
$\boldsymbol{q}_7$	159.5230	0.0035	64.3731	0.0005	25.1526	-0.15260
$\boldsymbol{q}_8$	184.6718	0.0004	63.9382	0.0007		

Table 8.19: PolyCtrlPts error on metal target without glass, $\bar{e} = 0.1045mm$.

Tensor Interpolation via Basis Function Sectioning

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.9587	0.0022	66.4943	0.0005	25.0464	-0.04640
$\boldsymbol{q}_2$	34.0030	0.0012	66.1723	0.0003	25.1602	-0.16016
$\boldsymbol{q}_3$	59.1593	0.0022	65.7333	0.0004	25.0934	-0.09337
$\boldsymbol{q}_4$	84.2507	0.0024	65.4137	0.0004	25.0905	-0.09047
$\boldsymbol{q}_5$	109.3383	0.0017	65.0383	0.0005	25.0377	-0.03770
$\boldsymbol{q}_6$	134.3747	0.0017	64.7763	0.0005	25.1564	-0.15644
$\boldsymbol{q}_7$	159.5277	0.0035	64.3603	0.0006	25.1500	-0.14996
$\boldsymbol{q}_8$	184.6740	0.0004	63.9330	0.0007		

Table 8.20: TensorBFS error on metal target without glass, $\bar{e} = 0.1049mm$.

Tensor Interpolation via Quad Tree Decomposition

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.9593	0.0022	66.4945	0.0005	25.0464	-0.04643
$\boldsymbol{q}_2$	34.0037	0.0011	66.1726	0.0002	25.1606	-0.16064
$\boldsymbol{q}_3$	59.1605	0.0022	65.7322	0.0004	25.0920	-0.09201
$\boldsymbol{q}_4$	84.2505	0.0024	65.4148	0.0002	25.0905	-0.09049
$\boldsymbol{q}_5$	109.3382	0.0016	65.0387	0.0005	25.0389	-0.03887
$\boldsymbol{q}_6$	134.3756	0.0016	64.7753	0.0005	25.1555	-0.15551
$\boldsymbol{q}_7$	159.5277	0.0035	64.3605	0.0005	25.1484	-0.14838
$\boldsymbol{q}_8$	184.6725	0.4285	63.9338	0.6793		

Table 8.21: TensorQDT error on metal target without glass, $\bar{e} = 0.1046mm$.

8.3.2 Metal Target with Glass

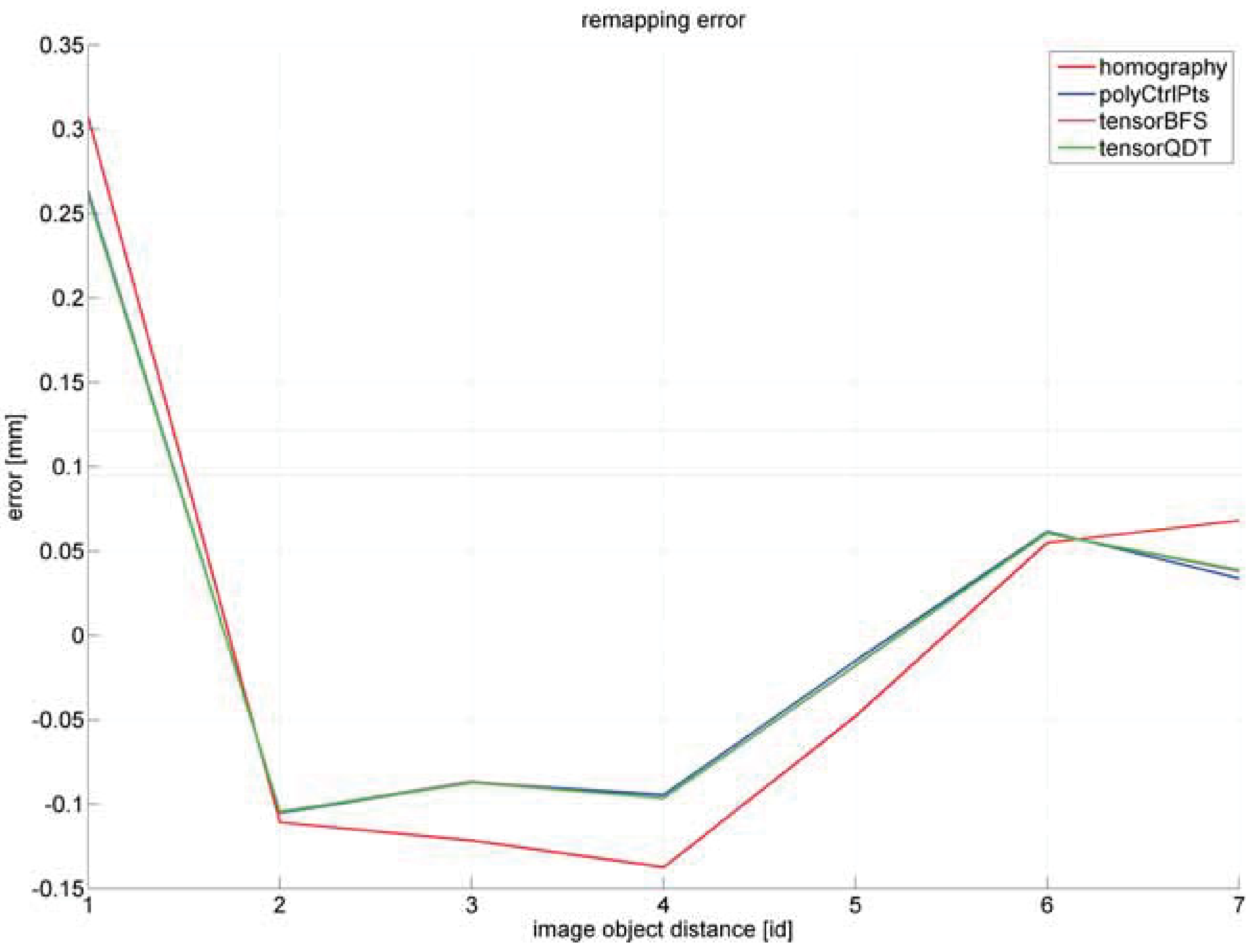


Figure 8.23: Resulting error on a metal target with glass.

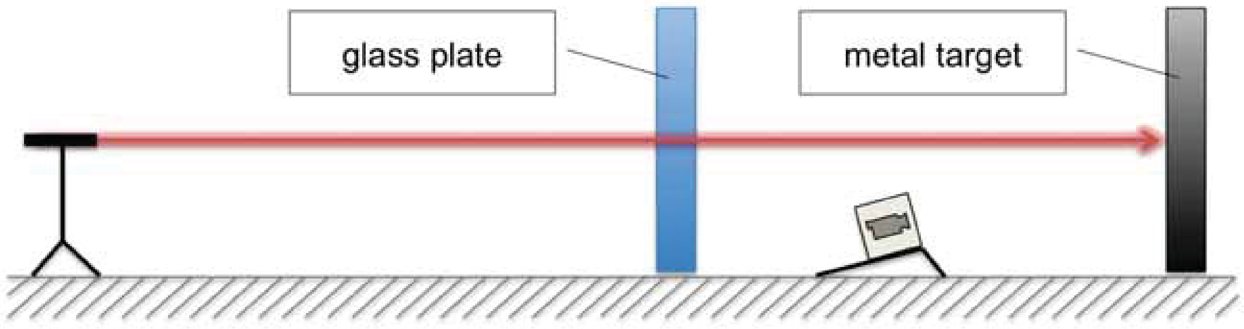


Figure 8.24: This setup features a transparent glass between the laser and the metal target.

Linear Homography

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	9.3305	0.0014	66.5048	0.0006	24.6926	0.30743
$\boldsymbol{q}_2$	34.0214	0.0019	66.2199	0.0007	25.1108	-0.11080
$\boldsymbol{q}_3$	59.1287	0.0013	65.8025	0.0003	25.1216	-0.12161
$\boldsymbol{q}_4$	84.2480	0.0011	65.4629	0.0011	25.1376	-0.13761
$\boldsymbol{q}_5$	109.3820	0.0017	65.0359	0.0006	25.0481	-0.04810
$\boldsymbol{q}_6$	134.4280	0.0026	64.7091	0.0005	24.9450	0.05495
$\boldsymbol{q}_7$	159.3716	0.0021	64.4404	0.0006	24.9320	0.06805
$\boldsymbol{q}_8$	184.2987	0.0041	63.9516	0.0005		

Table 8.22: Homography error on metal target with transparent glass, $\bar{e} = 0.1212mm$.

Polynomial Control Points

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	9.3512	0.0014	66.5177	0.0006	24.7367	0.26335
$\boldsymbol{q}_2$	34.0861	0.0019	66.2253	0.0007	25.1053	-0.10527
$\boldsymbol{q}_3$	59.1878	0.0013	65.8005	0.0003	25.0872	-0.08721
$\boldsymbol{q}_4$	84.2727	0.0011	65.4585	0.0011	25.0946	-0.09455
$\boldsymbol{q}_5$	109.3637	0.0017	65.0361	0.0006	25.0151	-0.01508
$\boldsymbol{q}_6$	134.3768	0.0026	64.7186	0.0005	24.9384	0.06156
$\boldsymbol{q}_7$	159.3139	0.0021	64.4595	0.0006	24.9663	0.03374
$\boldsymbol{q}_8$	184.2754	0.0041	63.9733	0.0005		

Table 8.23: PolyCtrlPts error on metal target with transparent glass, $\bar{e} = 0.0944mm$.

Tensor Interpolation via Basis Function Sectioning

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	9.3490	0.0014	66.5123	0.0006	24.7383	0.26169
$\boldsymbol{q}_2$	34.0857	0.0019	66.2270	0.0007	25.1048	-0.10483
$\boldsymbol{q}_3$	59.1870	0.0013	65.8080	0.0004	25.0867	-0.08665
$\boldsymbol{q}_4$	84.2713	0.0011	65.4670	0.0011	25.0967	-0.09668
$\boldsymbol{q}_5$	109.3643	0.0017	65.0373	0.0006	25.0181	-0.01812
$\boldsymbol{q}_6$	134.3803	0.0025	64.7117	0.0005	24.9394	0.06061
$\boldsymbol{q}_7$	159.3183	0.0022	64.4487	0.0007	24.9619	0.03805
$\boldsymbol{q}_8$	184.2757	0.0041	63.9687	0.0005		

Table 8.24: TensorBFS error on metal target with transparent glass, $\bar{e} = 0.0952mm$.

Tensor Interpolation via Quad Tree Decomposition

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	9.3485	0.0013	66.5118	0.0006	24.7390	0.26098
$\boldsymbol{q}_2$	34.0859	0.0019	66.2274	0.0007	25.1044	-0.10445
$\boldsymbol{q}_3$	59.1873	0.0014	65.8083	0.0005	25.0874	-0.08742
$\boldsymbol{q}_4$	84.2720	0.0011	65.4665	0.0011	25.0965	-0.09654
$\boldsymbol{q}_5$	109.3649	0.0017	65.0383	0.0006	25.0177	-0.01767
$\boldsymbol{q}_6$	134.3804	0.0026	64.7115	0.0005	24.9396	0.06037
$\boldsymbol{q}_7$	159.3186	0.0021	64.4470	0.0006	24.9613	0.03872
$\boldsymbol{q}_8$	184.2753	0.0040	63.9686	0.0005		

Table 8.25: TensorQDT error on metal target with transparent glass, $\bar{e} = 0.0952mm$.

8.3.3 Glass Target

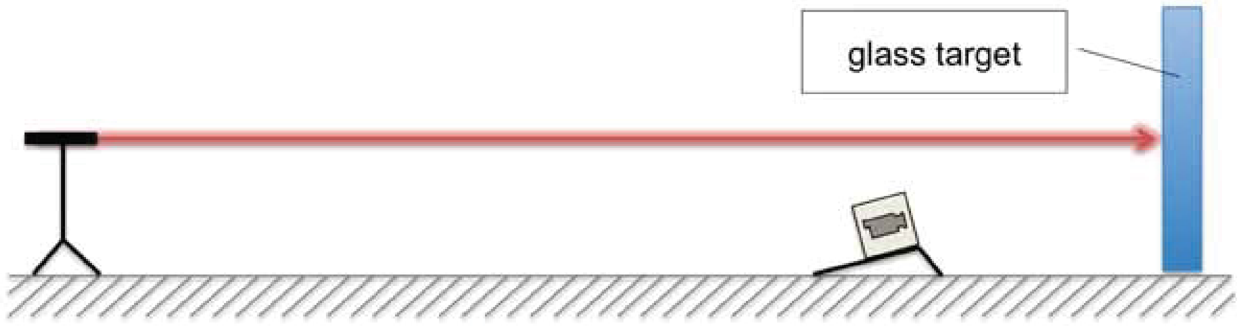


Figure 8.25: This setup features an opaque glass target.

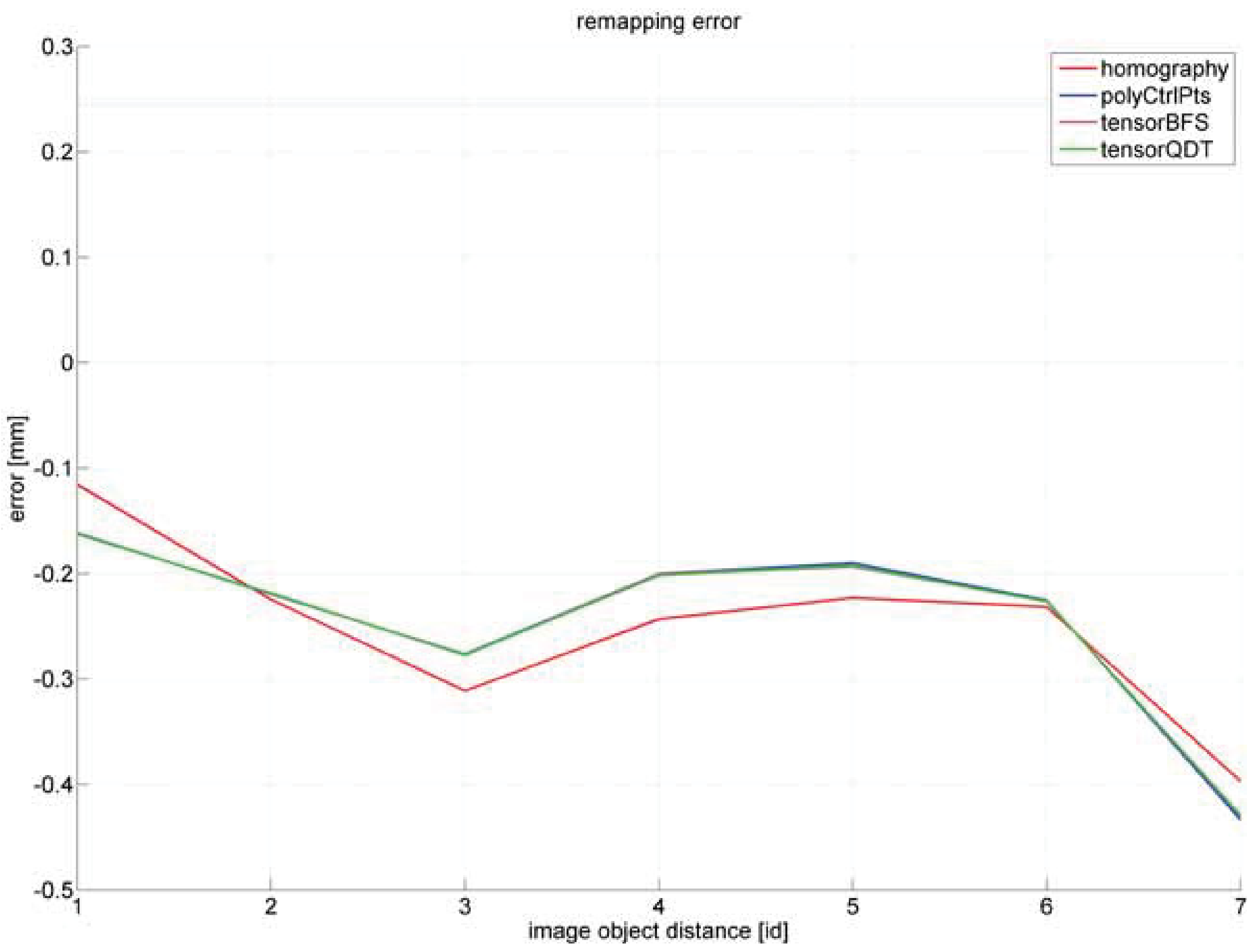


Figure 8.26: Resulting error on a glass target.

Linear Homography

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.6335	0.0006	66.3595	0.0004	25.1156	-0.11557
$\boldsymbol{q}_2$	33.7456	0.0012	65.9409	0.0004	25.2247	-0.22469
$\boldsymbol{q}_3$	58.9671	0.0018	65.5402	0.0005	25.3112	-0.31118
$\boldsymbol{q}_4$	84.2627	0.0017	64.6495	0.0016	25.2432	-0.24319
$\boldsymbol{q}_5$	109.5056	0.0020	64.7587	0.0003	25.2230	-0.22301
$\boldsymbol{q}_6$	134.7264	0.0006	64.4280	0.0004	25.2316	-0.23158
$\boldsymbol{q}_7$	159.9551	0.0018	64.0416	0.0004	25.3969	-0.39689
$\boldsymbol{q}_8$	185.3458	0.0017	63.4809	0.0003		

Table 8.26: Homography error on opaque glass target, $\bar{e} = 0.2494mm$.

Polynomial Control Points

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.6520	0.0006	66.3725	0.0004	25.1617	-0.16166
$\boldsymbol{q}_2$	33.8101	0.0012	65.9467	0.0004	25.2195	-0.21950
$\boldsymbol{q}_3$	59.0263	0.0018	65.5388	0.0005	25.2765	-0.27648
$\boldsymbol{q}_4$	84.2871	0.0017	64.6471	0.0016	25.2002	-0.20021
$\boldsymbol{q}_5$	109.4870	0.0020	64.7596	0.0003	25.1900	-0.18996
$\boldsymbol{q}_6$	134.6749	0.0006	64.4382	0.0004	25.2255	-0.22554
$\boldsymbol{q}_7$	159.8976	0.0018	64.0614	0.0004	25.4336	-0.43360
$\boldsymbol{q}_8$	185.3251	0.0017	63.5024	0.0003		

Table 8.27: PolyCtrlPts error on opaque glass target, $\bar{e} = 0.2439mm$.

Tensor Interpolation via Basis Function Sectioning

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.6510	0.0006	66.3657	0.0004	25.1621	-0.16211
$\boldsymbol{q}_2$	33.8097	0.0012	65.9493	0.0004	25.2189	-0.21889
$\boldsymbol{q}_3$	59.0253	0.0018	65.5460	0.0006	25.2774	-0.27736
$\boldsymbol{q}_4$	84.2870	0.0016	64.6553	0.0015	25.2002	-0.20023
$\boldsymbol{q}_5$	109.4870	0.0020	64.7623	0.0003	25.1935	-0.19351
$\boldsymbol{q}_6$	134.6783	0.0006	64.4313	0.0004	25.2262	-0.22624
$\boldsymbol{q}_7$	159.9017	0.0019	64.0483	0.0005	25.4296	-0.42962
$\boldsymbol{q}_8$	185.3253	0.0017	63.4980	0.0003		

Table 8.28: TensorBFS error on opaque glass target, $\bar{e} = 0.2440mm$.

Tensor Interpolation via Quad Tree Decomposition

point:	$x/[mm]$	$\pm e_{x_{95\%}}/[mm]$	$y/[mm]$	$\pm e_{y_{95\%}}/[mm]$	$b/[mm]$	$e/[mm]$
$\boldsymbol{q}_1$	8.6508	0.0006	66.3664	0.0004	25.1625	-0.16253
$\boldsymbol{q}_2$	33.8099	0.0012	65.9488	0.0004	25.2187	-0.21872
$\boldsymbol{q}_3$	59.0254	0.0018	65.5468	0.0005	25.2770	-0.27704
$\boldsymbol{q}_4$	84.2867	0.0017	64.6553	0.0016	25.2018	-0.20179
$\boldsymbol{q}_5$	109.4883	0.0020	64.7615	0.0003	25.1926	-0.19256
$\boldsymbol{q}_6$	134.6787	0.0006	64.4310	0.0004	25.2265	-0.22654
$\boldsymbol{q}_7$	159.9023	0.0018	64.0488	0.0004	25.4290	-0.42899
$\boldsymbol{q}_8$	185.3254	0.0017	63.4985	0.0003		

Table 8.29: TensorQDT error on opaque glass target, $\bar{e} = 0.2440mm$.

8.4 Result Interpretation

The research questions demand a proposal for the methodology to be deployed to achieve the best results for the actual measurement instrument. The experiments with the prototype showed following facts:

1. The *image acquisition* works for a normal-angle lens with $f = 12mm$ as well as for a lens with $f = 3.6mm$, the object distance g must be adjusted accordingly. It's recommended to use an interference filter to avoid errors from external light sources. The resolution of the digital camera is a limiting factor for the achievable accuracy. Still, a better device is no guarantee for increased quality, because of subpixel accuracy effects;

2. *Image preprocessing* is vital for the efficient computation of certain methods, as the full encoding spectrum can be used and therefore the contrast is increased. Furthermore, normalized data is required for many functions to work correctly. Noise suppression is essential for center determination through conic fitting, also the contour finding of image objects requires this filtering step;

3. Local Otsu's method delivers the best and most robust threshold for binary morphology. Therefore, *object segmentation* and contour finding can be efficiently carried out. The classification of image objects as regions of interest can be performed by deploying a-priori knowledge of the geometry;

4. The experiments in Section 8.2 and 8.1 showed, that for *center determination*, the conic fitting via ellipse approximation delivers the best results. If computation time is critical or the object has an arbitrary geometrical form, the center of gravity is also a good option;

5. The *system* should be *calibrated* directly with a xy-table. This brings a number of advantages: the system is directly adjusted using the actual target plate, i.e. there are no changes of the hardware setup; the mechanical errors even out, because they occur during the calibration and the measurement process; the LEDs and laser spots have different geometrical nature. This effect was shown in Section 8.2;

6. The quality of the *coordinate mapping* methods strongly depends on the objectives being used. For a normal-angle lens, homography works quite good, polynomial control points of $d = 4$ are okay and for tensor interpolation a degree $d = 4$ is sufficient. For a wide-angle lens, the homography doesn't work at all, polynomial control points could be used for other less accurate applications and tensor interpolation of $d = 6$ delivers almost a perfect mapping.

Chapter 7 proofed, that the acquired data is normally distributed, therefore the reachable accuracy can be estimated by employing the concept of the standard error with a 95% confidence interval. By recalling the experiments from chapter 8, it can be assumed, that the measurement error e for this instrument is about

$$e \leq \pm 0.05mm. \tag{8.14}$$

This is about ten times higher than the physical barrier of $3.2\mu m$ determined by the laser's wavelength as stated in Equation 2.6, which is in fact a reasonable solution. The experiments from Section 8.3 showed errors of $e \approx \pm 0.10mm$, but as this error is stable and appears every time, it can be assumed to be a systematic error caused by the linear drive. Therefore, electro-active glass has much potential for such an application.

Keep in mind, that there are fundamental challenges for this application when it's used under real conditions in an underground structure: dust and moisture on the target plates, vibrations through heavy machinery or other external influences. It might be necessary to install more than one system, since curved structures may need to be monitored.

Chapter 9

Discussions

The experiments covered by this feasibility study showed, that the implemented methods work under laboratory conditions and it's possible to detect relative movements caused by external forces.

9.1 Conclusion

It turned out, that the system can be calibrated directly with a xy-table without the need of a designated calibration plate. Therefore, it could be tested, how a *theodolite* would perform, as this device allows a highly accurate adjustment. Still, LEDs could be used to validate the correct position of a target plate. Other possible options would be the use of a camera with a higher resolution or the deployment of a laser with a shorter wavelength.

Additionally, this thesis introduced the idea of tensor interpolation via quad tree decomposition. It was shown, that this technique is highly efficient for remapping one point from the camera space to its real space equivalent. This brings three main advantages: it works for wide-angle objectives, therefor a compact design is possible; the optics can be cheap in comparison to common industry equipment; and the low computational effort allows the algorithm to run on an embedded system.

The software implementation could be translated from Matlab to a native *C*-code or the process could be modeled in *Simulink*[1] to profit from higher computational performance.

[1]©The MathWorks Inc., Natick, MA, United States of America, www.mathworks.com

Consequently, the next step of the concept's development would be the deployment of a prototype in an actual working environment for further testing and improvement. Especially, the installation of a series of electro-active glass plates and how this effects the outcome must be investigated next.

9.2 Potential

The fields of application for electro-active glass are numerous, e.g. when a barrier for light is required. A possibility would be its deployment as a replacement for mechanical shutters in cameras or even in theodolites.

The tensor interpolation via quad tree decomposition has much potential, especially when a complete image registration is not necessary and only certain points are of interest. Still, there are possible expansions to improve the method further, e.g. the extrapolation over the borders of the support or to use only the even portions of the bases functions, because the camera space has axis symmetry.

Bases functions are commonly used for filtering purposes. This thesis has proofed, that the inverse way, i.e. interpolation of data, is also working. This means, based on a few known points, assumptions about the whole data set can be made. The idea is not restricted for two-dimensional images, it can also be adapted to work for spoken language or three-dimensional objects.

Appendices

Appendix A

Optical Arrangement

This appendix features the calculations done for the optical arrangement of chapter 2.

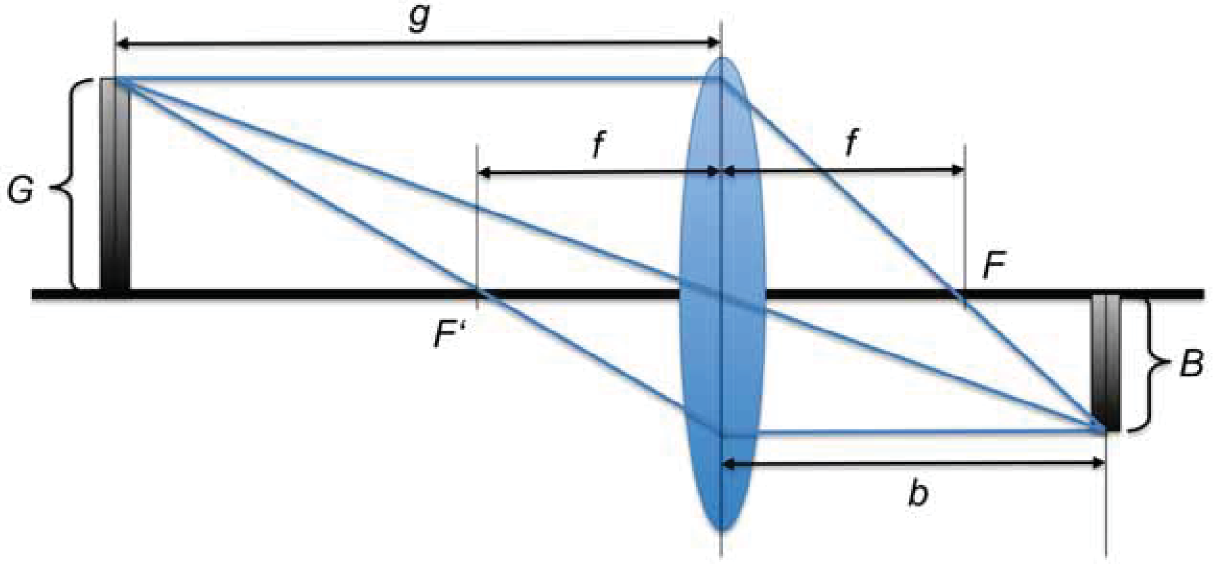


Figure A.1: This schematic of an optical lens illustrates the relation between the object G and image B, the object distance g and image distance b as well as the focal length f.

Focal Length of $f = 12mm$

```
% 1/2" CCD chip (image size, Bildhoehe), fixed values
Bh = 4.80; % higth [mm]
Bw = 6.40; % width [mm]
% target (object size, Gegenstandshoehe)
Gh = 150.00 + 20.00; % higth [mm]
Gw = 200.00 + 20.00; % width [mm]
% focal length (Brennweite des Objektivs)
f = 12; % [mm]
% object distance (Gegenstandsweite)
% G = (g-f)*B/f >> g = f * (G+B)/B
gh = f * (Gh+Bh)/Bh; % gh = 437.00 mm
gw = f * (Gw+Bw)/Bw; % gw = 424.50 mm
% take the maximal object distance and round it to the next [cm]
g = ceil(max(gh,gw)/10)*10; % g = 440.00 mm
% the target size with the computed distance
Gh1 = (g-f)*Bh/f; % Gh1 = 171.20 mm
Gw1 = (g-f)*Bw/f; % Gw1 = 228.27 mm
```

Source Code A.1: Computation of the optical arrangement for $f = 12mm$.

Therefore the object distance $g = 44cm$ for the focal length $f = 12mm$.

Focal Length of $f = 3.6mm$

```
% 1/2" CCD chip (image size, Bildhoehe), fixed values
Bh = 4.80; % higth [mm]
Bw = 6.40; % width [mm]
% target (object size, Gegenstandshoehe)
Gh = 150.00 + 20.00; % higth [mm]
Gw = 200.00 + 20.00; % width [mm]
% focal length (Brennweite des Objektivs)
f = 3.6; % [mm]
% object distance (Gegenstandsweite)
% G = (g-f)*B/f >> g = f * (G+B)/B
gh = f * (Gh+Bh)/Bh; % gh = 131.10 mm
gw = f * (Gw+Bw)/Bw; % gw = 127.35 mm
% take the maximal object distance and round it to the next [cm]
g = ceil(max(gh,gw)/10)*10; % g = 140.00 mm
% the target size with the computed distance
Gh1 = (g-f)*Bh/f; % Gh1 = 181.87 mm
Gw1 = (g-f)*Bw/f; % Gw1 = 242.49 mm
```

Source Code A.2: Computation of the optical arrangement for $f = 3.6mm$.

Therefore the object distance $g = 14cm$ for the focal length $f = 3.6mm$.

Appendix B

Threshold Computation

The computation of a suitable threshold is necessary for a successful image object segmentation. The implementations presented here allow to derive the Otsu threshold t_o, the simple threshold t_s, the Gauss threshold t_g and the maximal normal distance threshold t_n.

```
function [t, tIdx] = getThreshold (D, bins, method, debug)
x = 1:bins;
% Otsu threshold and corresponding index:
% ----------------------------------------
otsuT = graythresh(D);
otsuIdx = round(otsuT*bins);
% get histogram values
h = imhist(D,bins);
% find classes:
% ----------------------------------------
idxT = round(bins*otsuT);
% class1
class1 = h(1:idxT);
max1 = max(class1); % peak value
meanIdx1 = find(class1 == max(class1));
mean1 = x(meanIdx1(1)); % index of peak value
sigmaIdx1 = findnearest(std(class1),class1);
sigma1 = mean1-x(sigmaIdx1(1)); % difference of indices
y1 = max1*exp(-((x-mean1)/sigma1).^2); % Gauss curve
% class2
class2 = h(idxT+1:end);
max2 = max(class2);
meanIdx2 = find(class2 == max(class2))+idxT;
mean2 = x(meanIdx2(1));
sigmaIdx2 = findnearest(std(class2),class2)+idxT;
sigma2 = mean2-x(sigmaIdx2(1));
y2 = max2*exp(-((x-mean2)/sigma2).^2);
% simple threshold:
% ----------------------------------------
simpleIdx = round(((meanIdx1+meanIdx2)/2));
simpleT  = ((meanIdx1+meanIdx2)/2)/bins;
%
```

```
% Gauss method:
% ----------------------------------------
mean1_2  =  mean1^2;
sigma1_2 =  sigma1^2;
mean2_2  =  mean2^2;
sigma2_2 =  sigma2^2;
a = - 1/sigma1_2 + 1/sigma2_2;
b = + 2*( -mean2/sigma2_2 + mean1/sigma1_2 );
c = - mean1_2/sigma1_2 + mean2_2/sigma2_2 + log(max1/max2);
temp = sqrt( b^2 - 4*a*c);
% solutions
sol1 = (-b +temp)/(2*a);
sol2 = (-b -temp)/(2*a);
% only one solution is valid
if (sol1 > mean1) && (sol1 < mean2)
   solG = sol1;
elseif (sol2 > mean1) && (sol2 < mean2);
   solG = sol2;
else
   solG = [];
end;
gaussIdx = round(solG);
gaussT = solG/bins;
% maximal normal distance:
% ----------------------------------------
lineP = [ max1-max2, mean2-mean1,(max2*mean1)-(max1*mean2)];
scale = sqrt(lineP(1)^2 + lineP(2)^2);
lineP = lineP/scale;
yP = -lineP(1)/lineP(2)*x-lineP(3)/lineP(2); % plotting lineP
% points = [bins counts 1];
points = [x(mean1:mean2); h(mean1:mean2)'; ones(size((mean1:mean2)))];
% find the index with the maximal normal distance
ds = lineP * points;
[maxD maxDidx] = max (abs(ds));
maxDidx = maxDidx + mean1;
% determine the normal line;
lineT = [ -lineP(2), lineP(1), 0];
lineT(3) = -lineT * [x(maxDidx);h(maxDidx);1];
yT = -lineT(1)/lineT(2)*x-lineT(3)/lineT(2); % plotting lineT
% compute threshold and corresponding index
maxNdIdx = maxDidx;
maxNdT = maxDidx/bins;
```

Source Code B.1: Threshold computation with Matlab.

Appendix C

Statistical Analysis

The measurements are carried out for two different types of data sources to determine their statistical behavior, which are

1. TC#1 (Test-Case 1): *single LED*;
2. TC#2 (Test-Case 2): *single laser spot*.

The data samples are acquired for the camera coordinates of $\boldsymbol{p} = [x_c, y_c]^T$, additionally the corresponding real coordinates $\boldsymbol{q} = [x_r, y_r]^T$ for every coordinate mapping method are computed. All values are examined with the K-S test: if H_0 is accepted, then the data has normal distribution, if H_0 is rejected, then the data has no normal distribution . The number of repetitions is $\rho = 100$.

TC#1: Single LED / Camera Samples / x Dimension

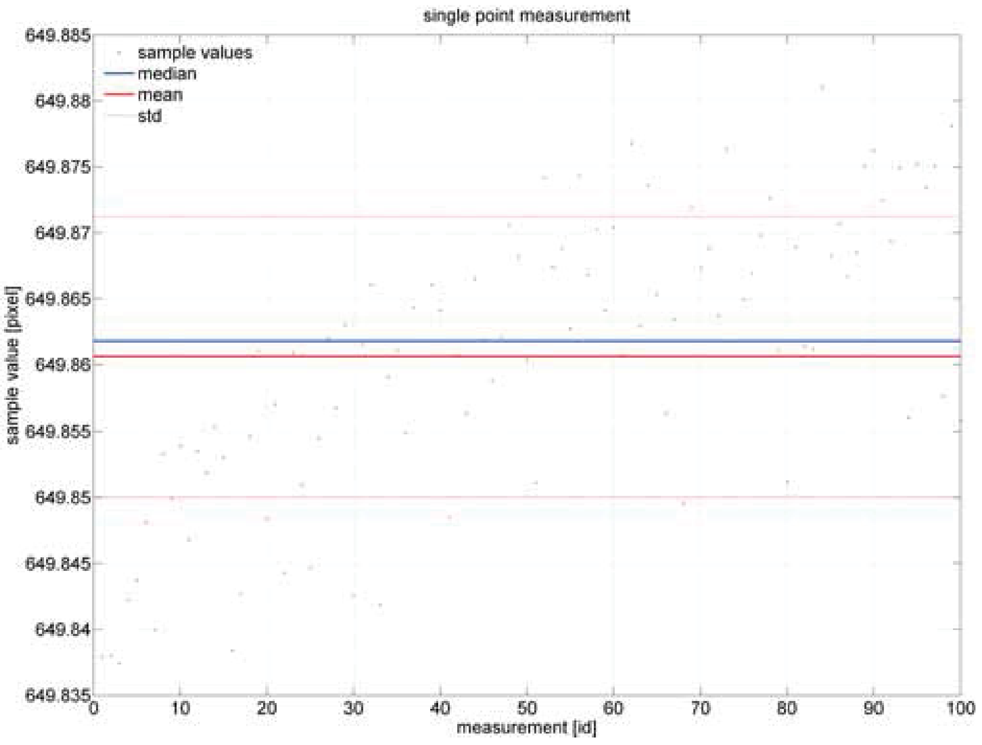


Figure C.1: Measured LED camera coordinates in x dimension.

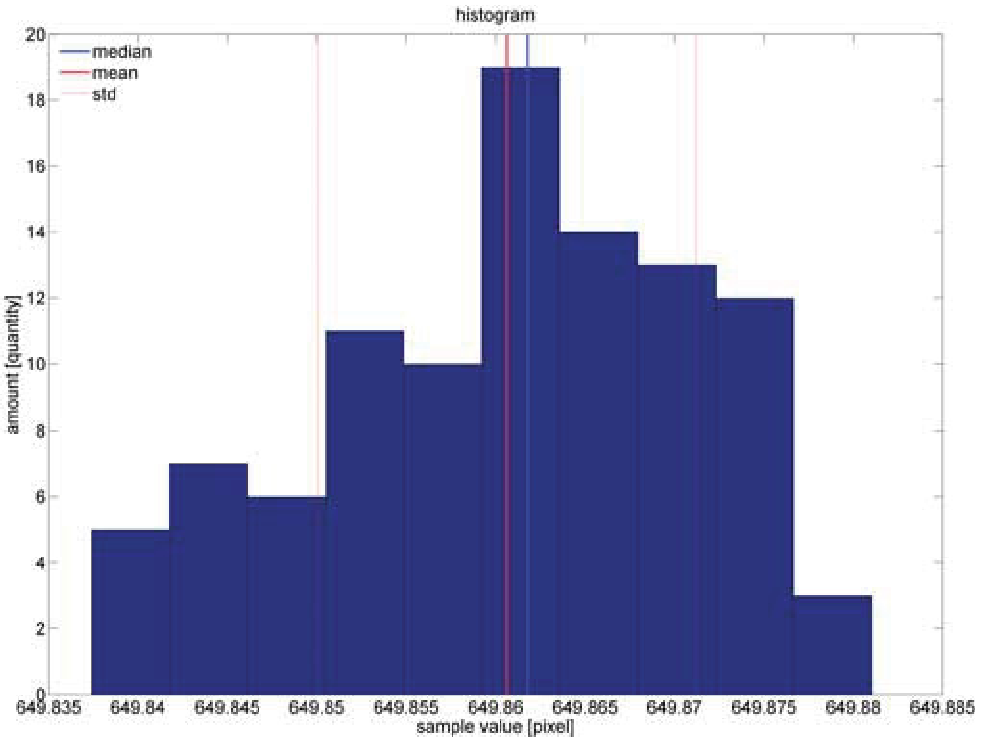


Figure C.2: Histogram of measured LED camera coordinates in x dimension.

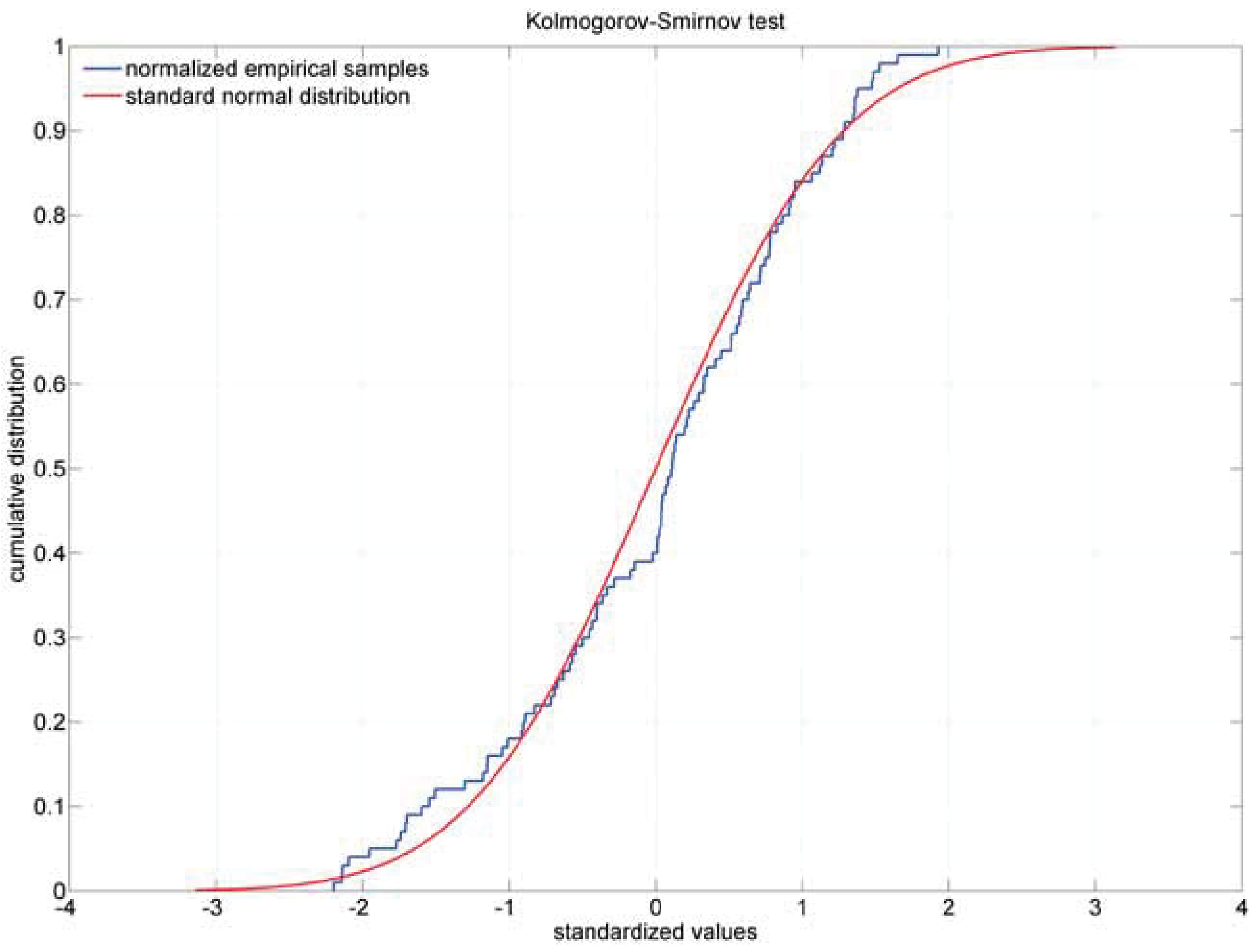


Figure C.3: K-S test of measured LED camera coordinates in x dimension.

Statistical analysis:		
$\hat{a}$	649.8374	$pixel$
$\tilde{a}$	649.8618	$pixel$
$\bar{a}$	649.8606	$pixel$
s^2	0.00011213	$pixel^2$
s	0.010589	$pixel$
e	0.0010589	$pixel$
$e_{95\%}$	0.0020755	$pixel$
$e_{99\%}$	0.002679	$pixel$
$\hat{\gamma}_1$	-0.40949	
$\hat{\beta}_2$	2.427	
K-S test	H_0 accepted	

Table C.1: Stat. analysis of measured LED camera coordinates in x dimension.

TC#1: Single LED / Camera Samples / y Dimension

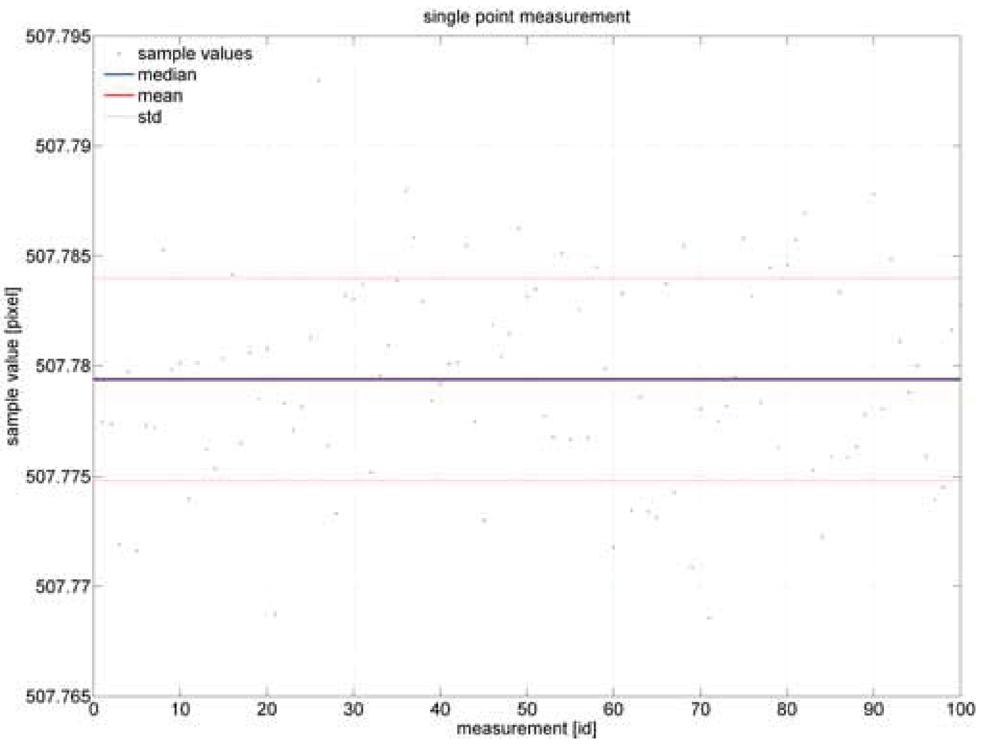


Figure C.4: Measured LED camera coordinates in y dimension.

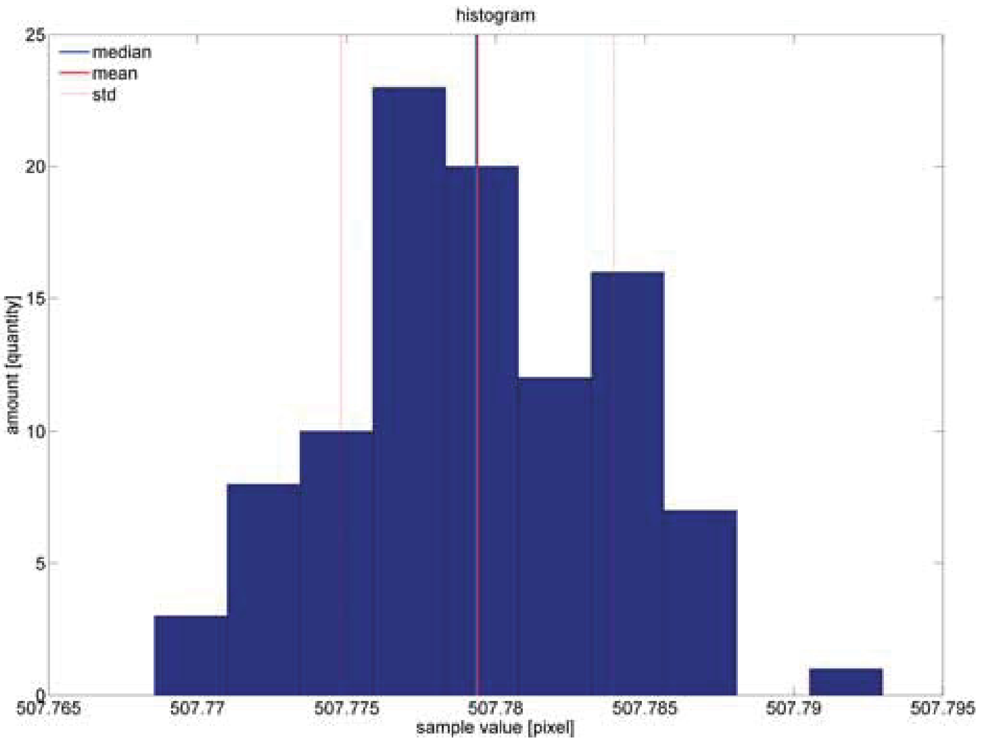


Figure C.5: Histogram of measured LED camera coordinates in y dimension.

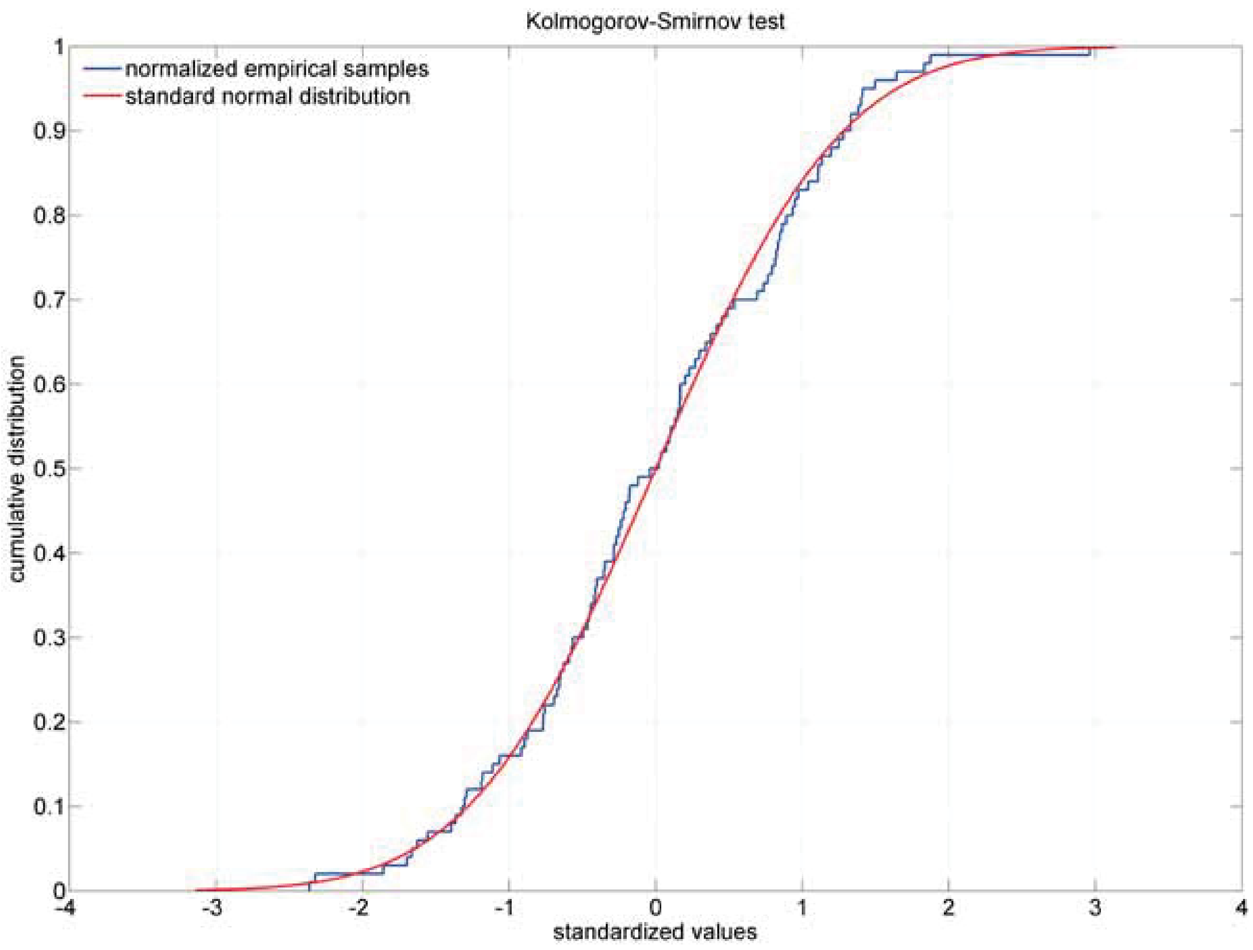


Figure C.6: K-S test of measured LED camera coordinates in y dimension.

Statistical analysis:		
$\hat{a}$	507.7686	$pixel$
$\tilde{a}$	507.7793	$pixel$
$\bar{a}$	507.7794	$pixel$
s^2	0.000021018	$pixel^2$
s	0.0045845	$pixel$
e	0.00045845	$pixel$
$e_{95\%}$	0.00089856	$pixel$
$e_{99\%}$	0.0011599	$pixel$
$\hat{\gamma}_1$	0.050525	
$\hat{\beta}_2$	2.8473	
K-S test	H_0 accepted	

Table C.2: Stat. analysis of measured LED camera coordinates in y dimension.

TC#1: Single LED / Homography / x Dimension

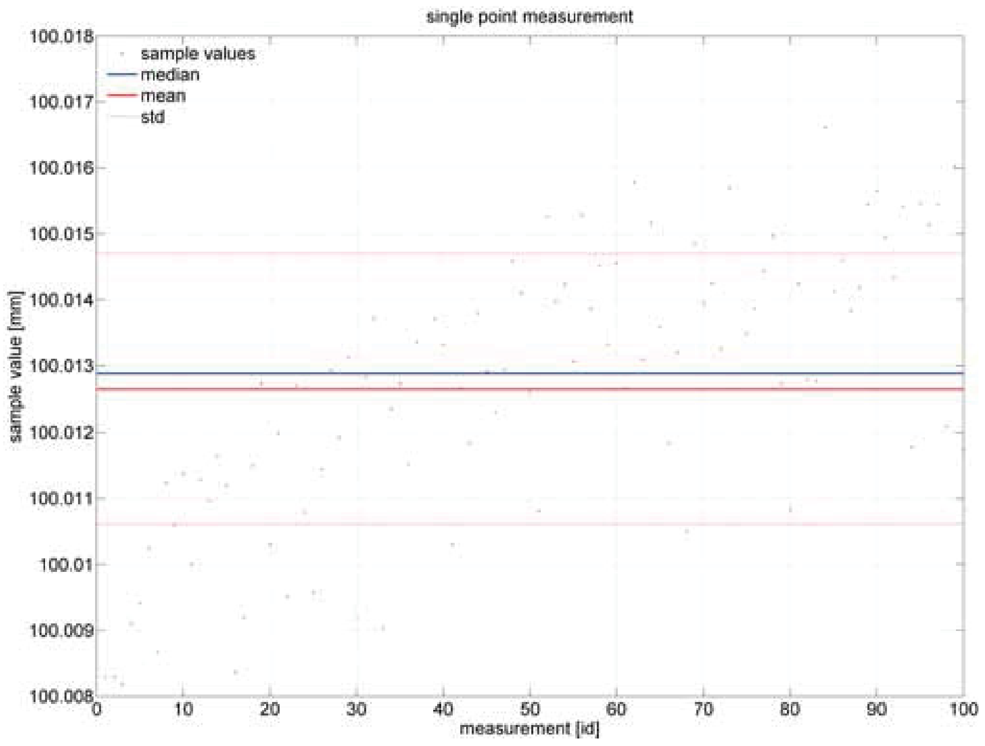


Figure C.7: LED real coordinates in x dimension via homography.

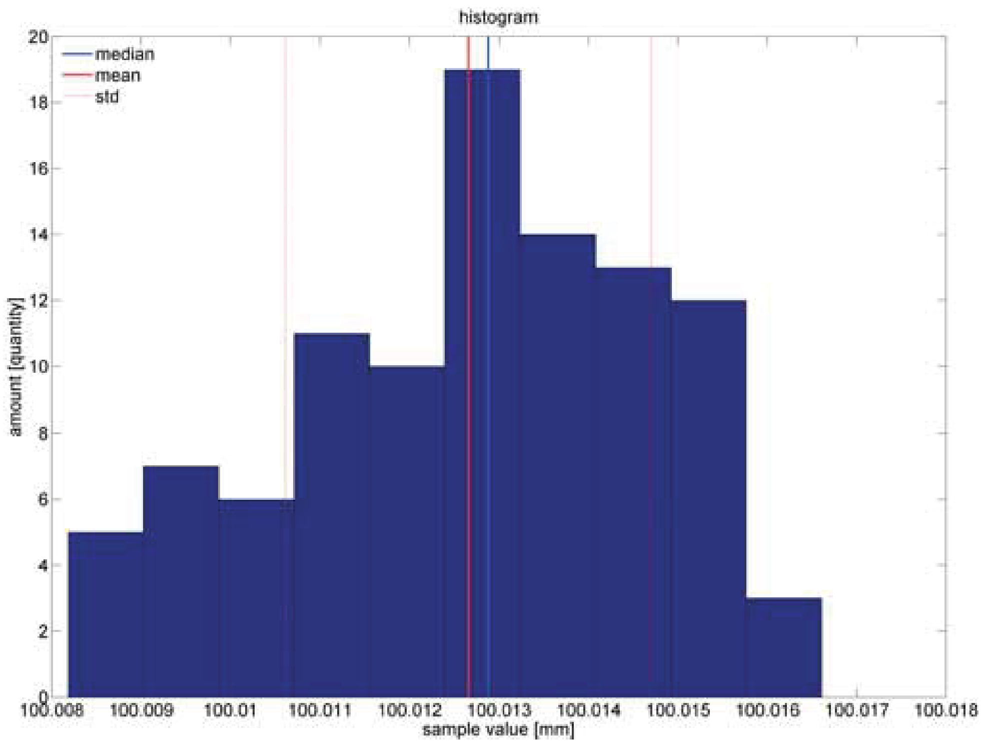


Figure C.8: Histogram of LED real coordinates in x dimension via homography.

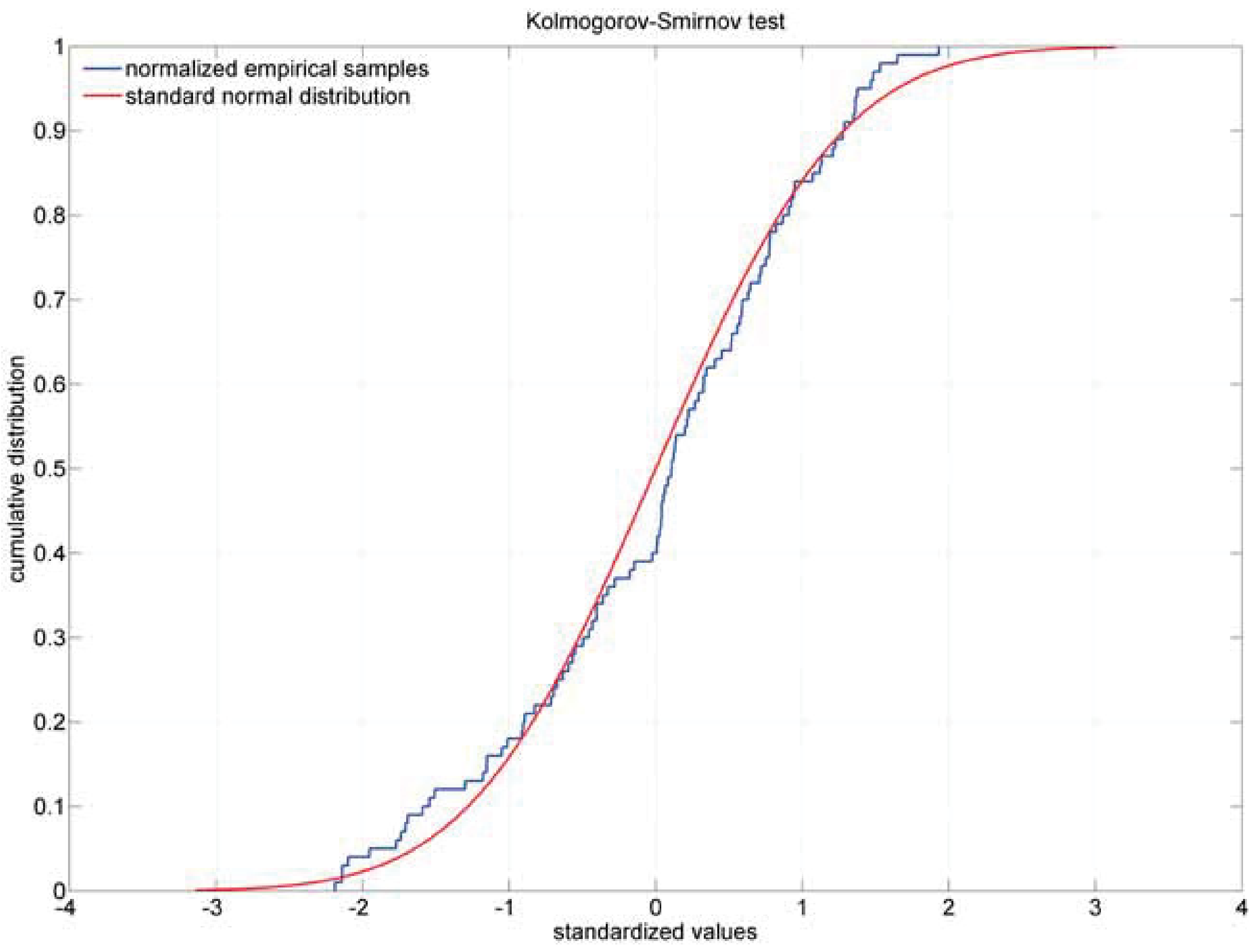


Figure C.9: K-S test of LED real coordinates in x dimension via homography.

Statistical analysis:		
$\hat{a}$	100.0082	mm
$\tilde{a}$	100.0129	mm
$\bar{a}$	100.0127	mm
s^2	0.00000418	mm^2
s	0.0020445	mm
e	0.00020445	mm
$e_{95\%}$	0.00040072	mm
$e_{99\%}$	0.00051726	mm
$\hat{\gamma}_1$	-0.40753	
$\hat{\beta}_2$	2.4245	
K-S test	H_0 accepted	

Table C.3: Stat. analysis of LED real coordinates in x dimension via homography.

TC#1: Single LED / Homography / y Dimension

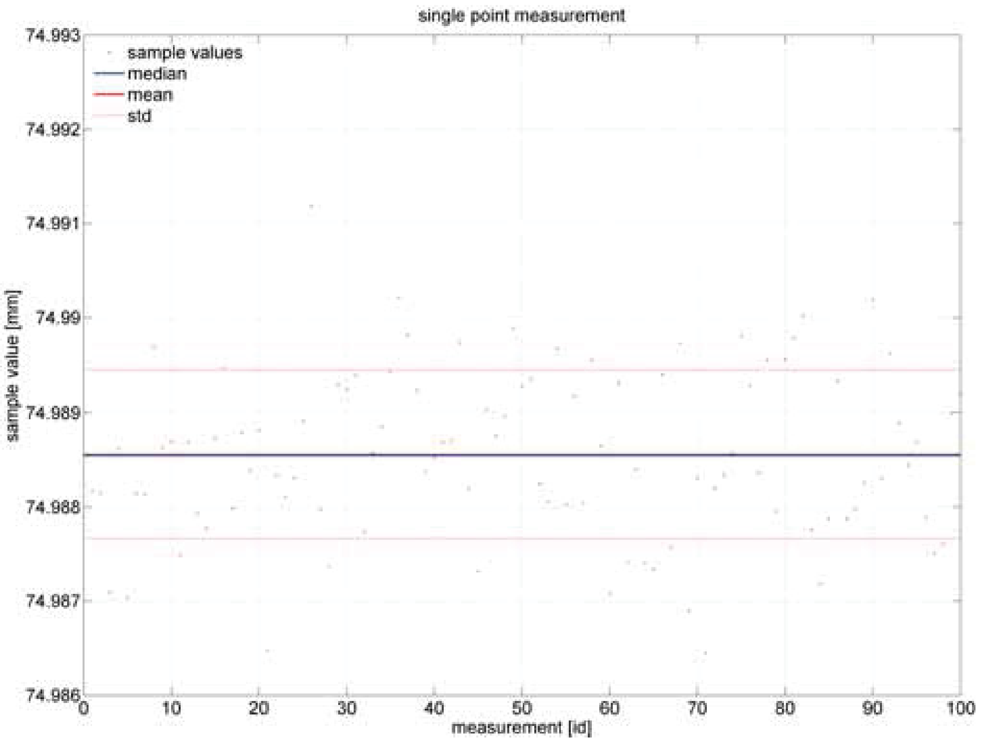


Figure C.10: LED real coordinates in y dimension via homography.

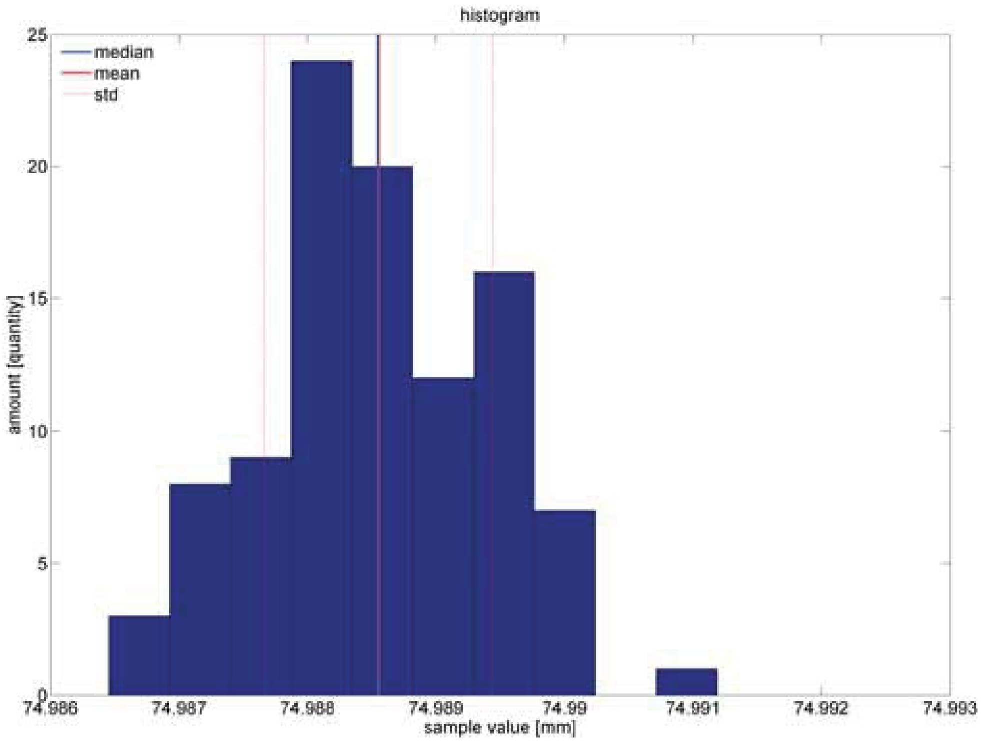


Figure C.11: Histogram of LED real coordinates in y dimension via homography.

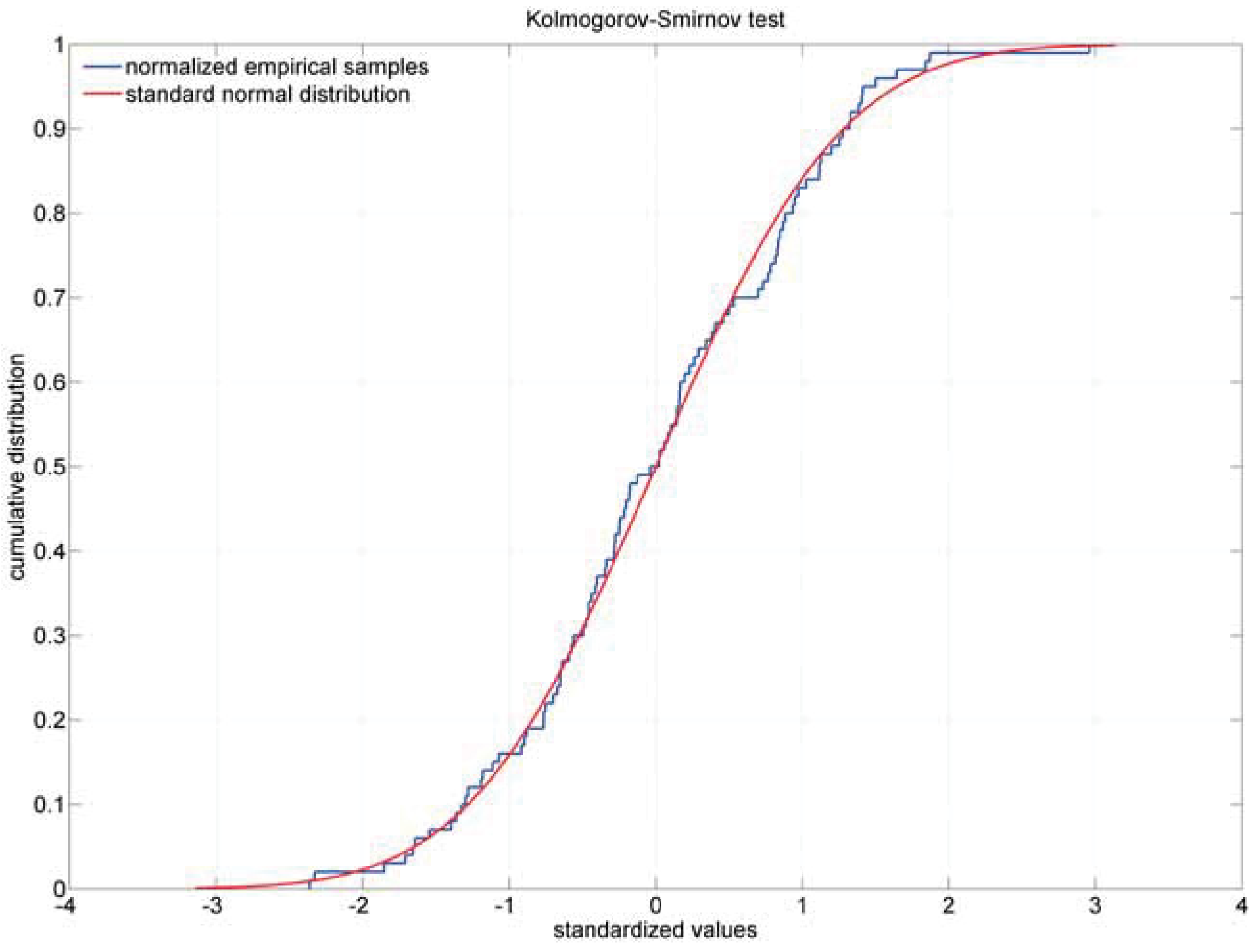


Figure C.12: K-S test of LED real coordinates in y dimension via homography.

Statistical analysis:		
$\hat{a}$	74.9865	mm
$\tilde{a}$	74.9885	mm
$\bar{a}$	74.9886	mm
s^2	0.00000079207	mm^2
s	0.00088998	mm
e	0.000088998	mm
$e_{95\%}$	0.00017444	mm
$e_{99\%}$	0.00022517	mm
$\hat{\gamma}_1$	0.052232	
$\hat{\beta}_2$	2.8453	
K-S test	H_0 accepted	

Table C.4: Stat. analysis of LED real coordinates in y dimension via homography.

TC#1: Single LED / PolyCtrlPts / x Dimension

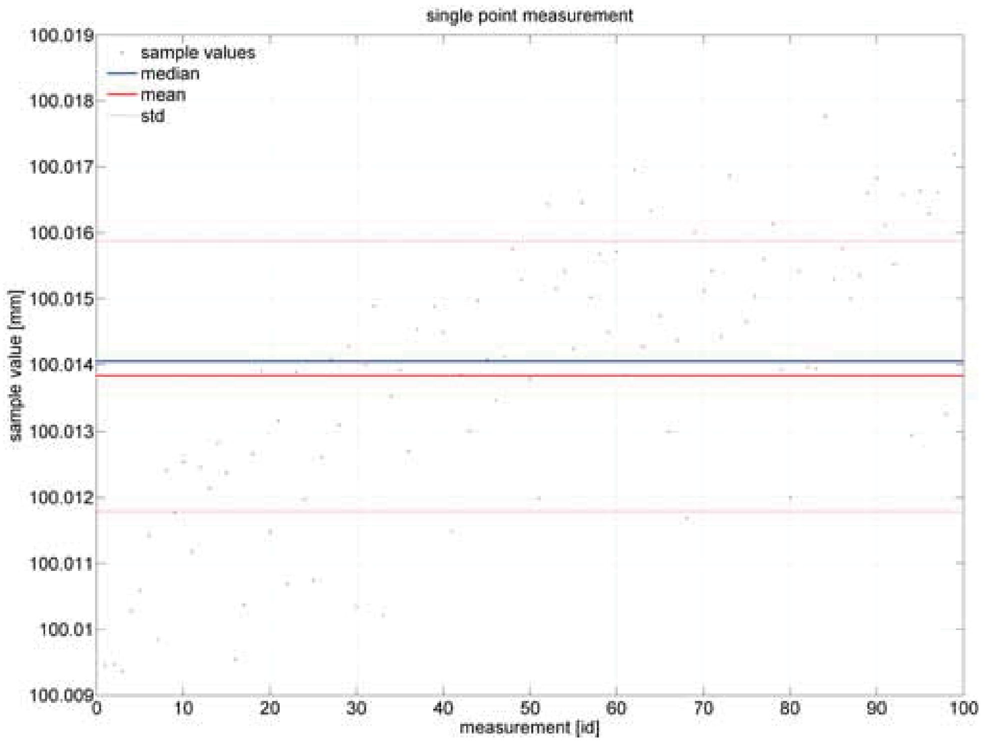


Figure C.13: LED real coordinates in x dimension via polyCtrlPts.

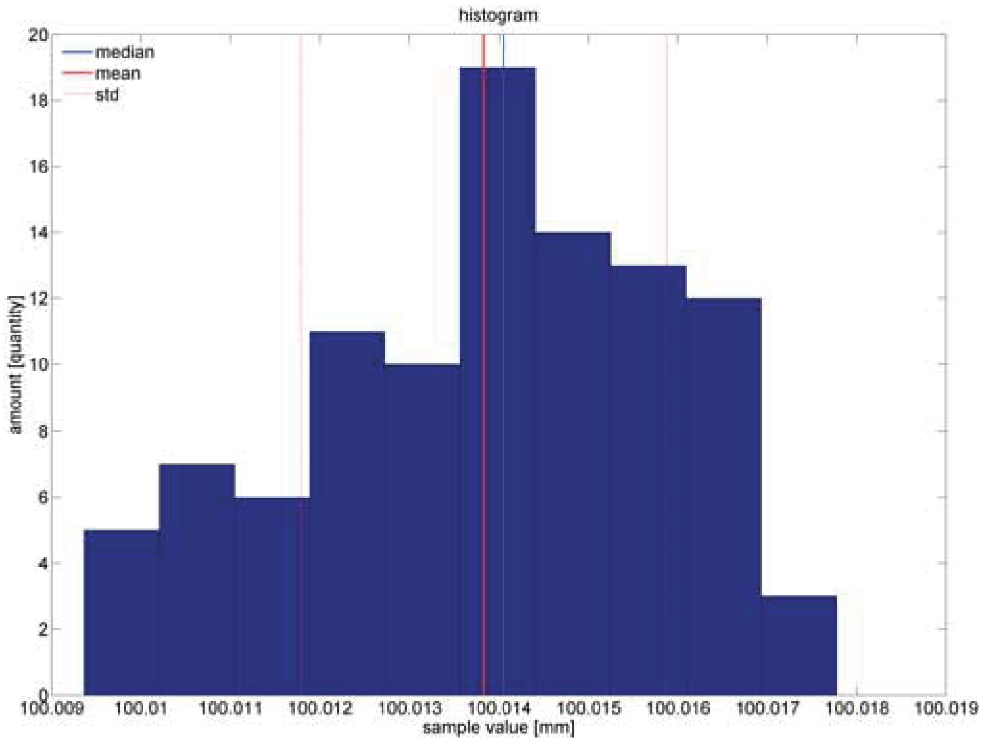


Figure C.14: Histogram of LED real coordinates in x dimension via polyCtrlPts.

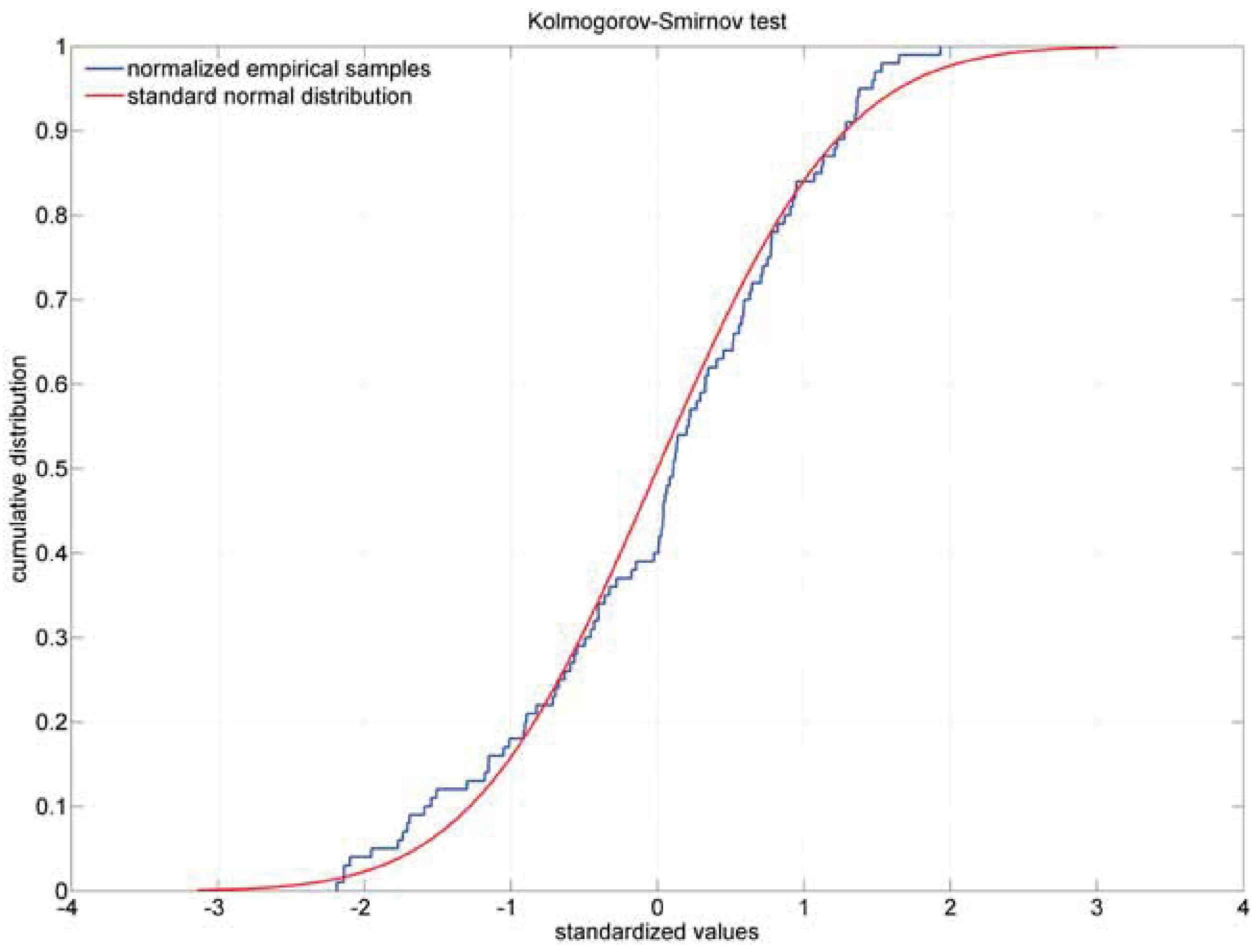


Figure C.15: K-S test of LED real coordinates in x dimension via polyCtrlPts.

Statistical analysis:		
$\hat{a}$	100.0094	mm
$\tilde{a}$	100.0141	mm
$\bar{a}$	100.0138	mm
s^2	0.0000041659	mm^2
s	0.0020411	mm
e	0.00020411	mm
$e_{95\%}$	0.00040005	mm
$e_{99\%}$	0.00051639	mm
$\hat{\gamma}_1$	-0.40755	
$\hat{\beta}_2$	2.4245	
K-S test	H_0 accepted	

Table C.5: Stat. analysis of LED real coordinates in x dimension via polyCtrlPts.

TC#1: Single LED / PolyCtrlPts / y Dimension

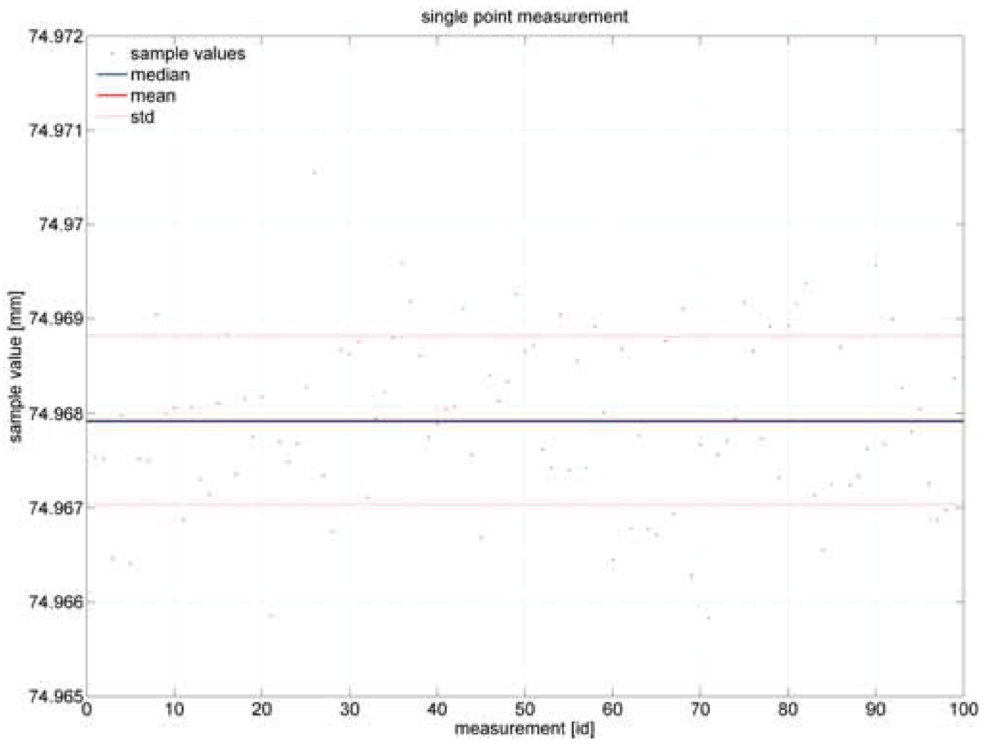


Figure C.16: LED real coordinates in y dimension via polyCtrlPts.

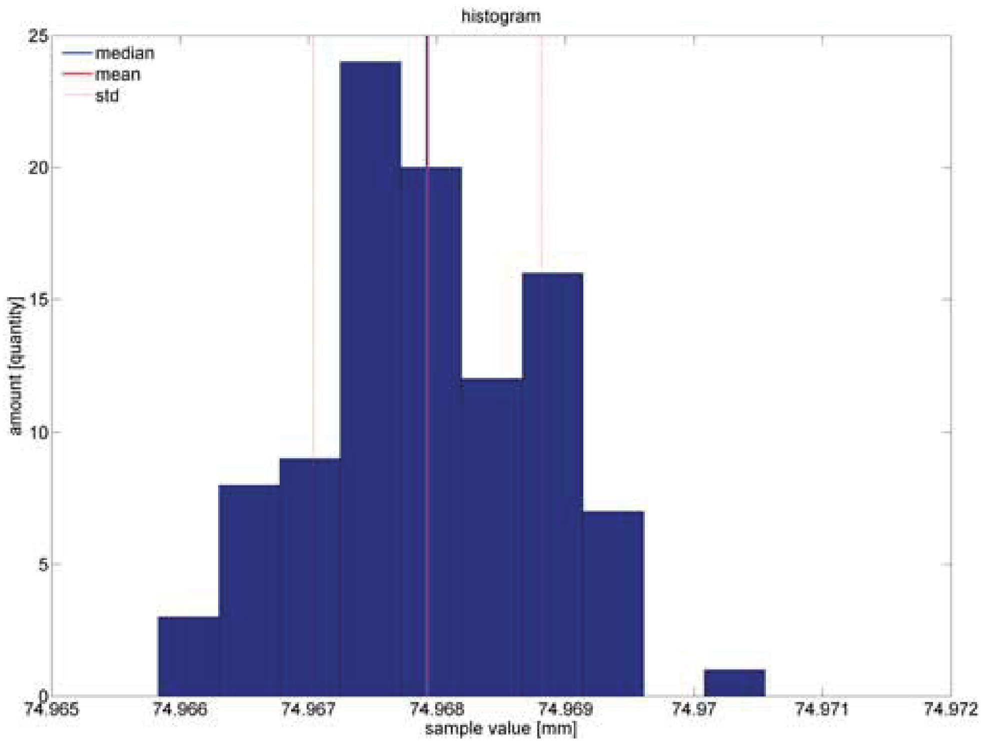


Figure C.17: Histogram of LED real coordinates in y dimension via polyCtrlPts.

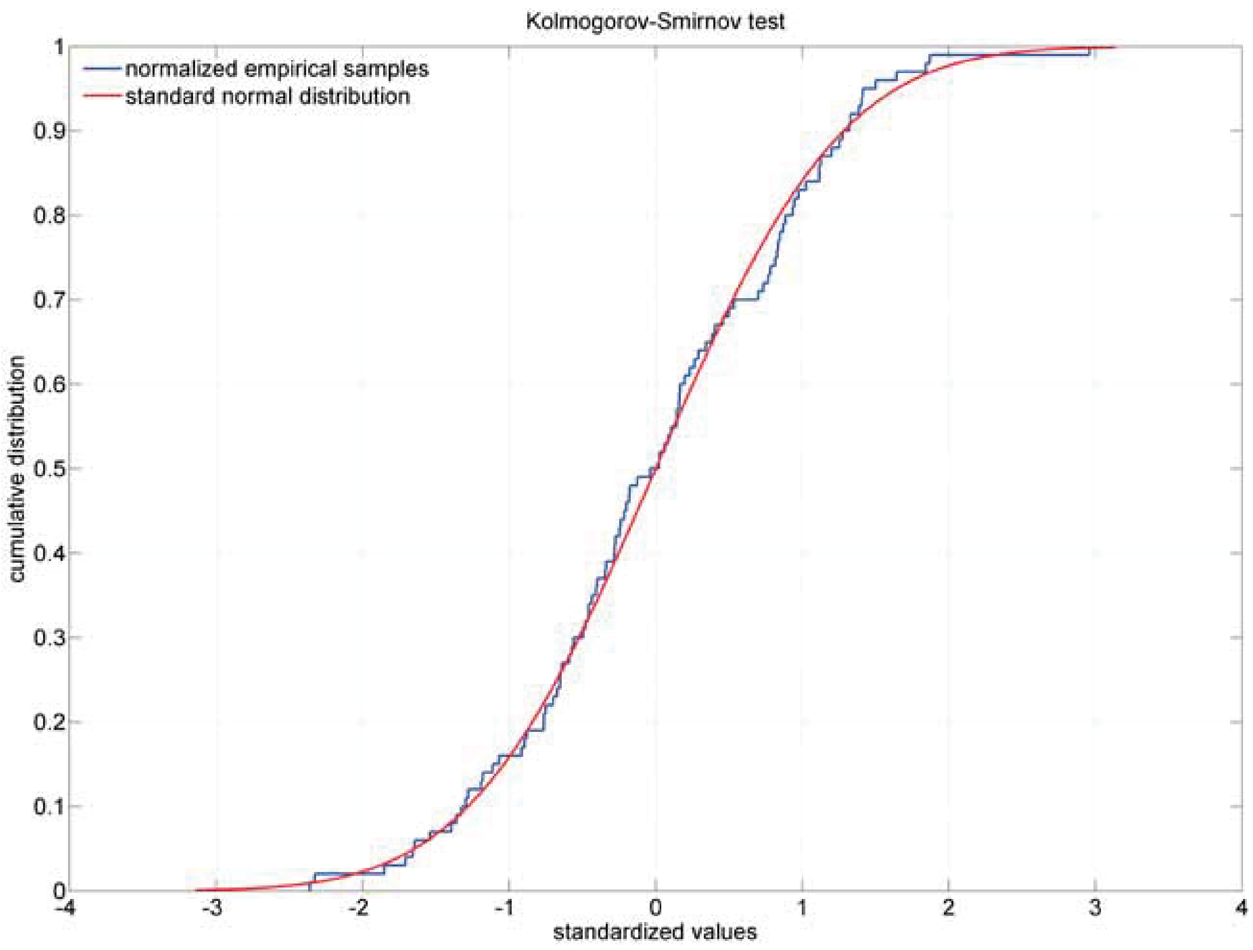


Figure C.18: K-S test of LED real coordinates in y dimension via polyCtrlPts.

Statistical analysis:		
$\hat{a}$	74.9658	mm
$\tilde{a}$	74.9679	mm
$\bar{a}$	74.9679	mm
s^2	0.00000078846	mm^2
s	0.00088795	mm
e	0.000088795	mm
$e_{95\%}$	0.00017404	mm
$e_{99\%}$	0.00022465	mm
$\hat{\gamma}_1$	0.052372	
$\hat{\beta}_2$	2.8451	
K-S test	H_0 accepted	

Table C.6: Stat. analysis of LED real coordinates in y dimension via polyCtrlPts.

TC#1: Single LED / TensorBFS / x Dimension

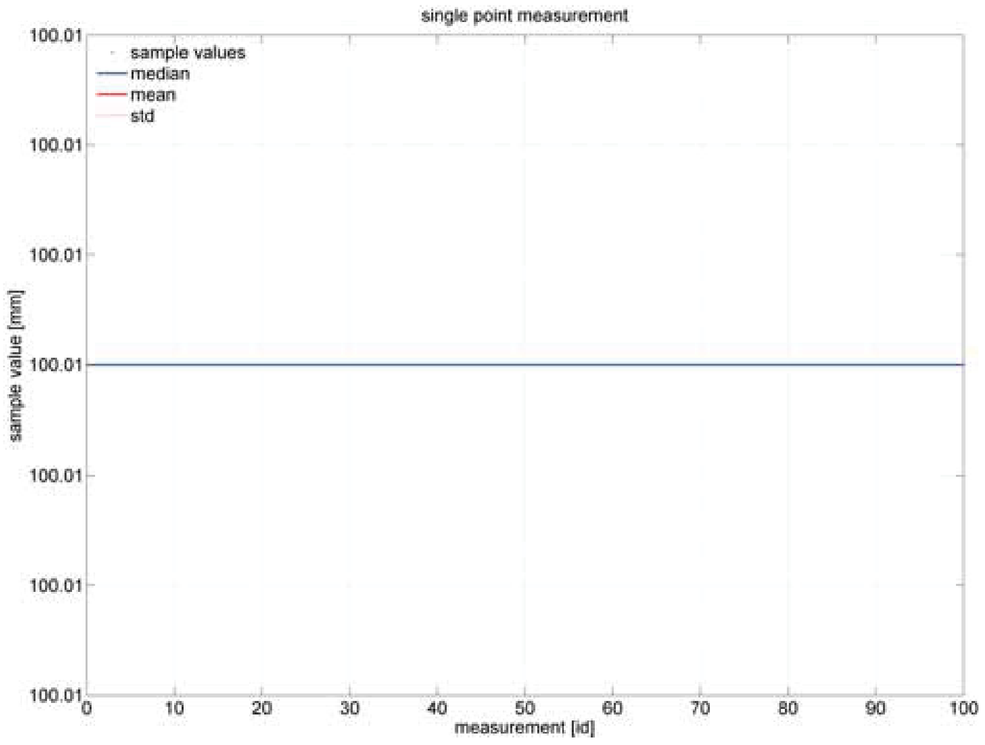


Figure C.19: LED real coordinates in x dimension via tensorBFS.

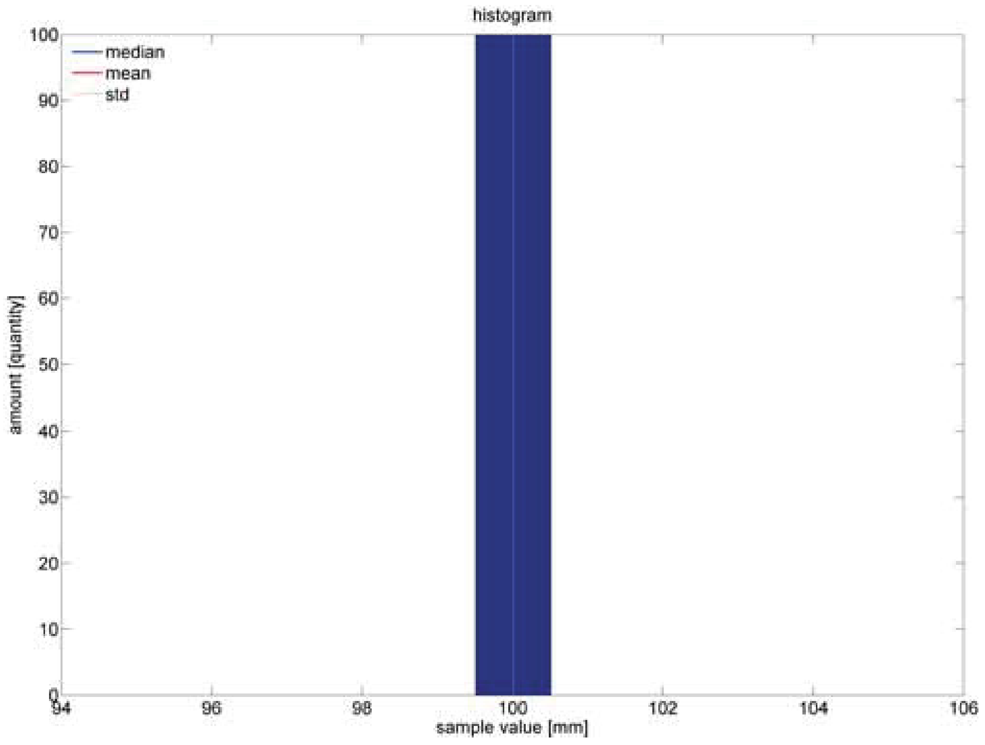


Figure C.20: Histogram of LED real coordinates in x dimension via tensorBFS.

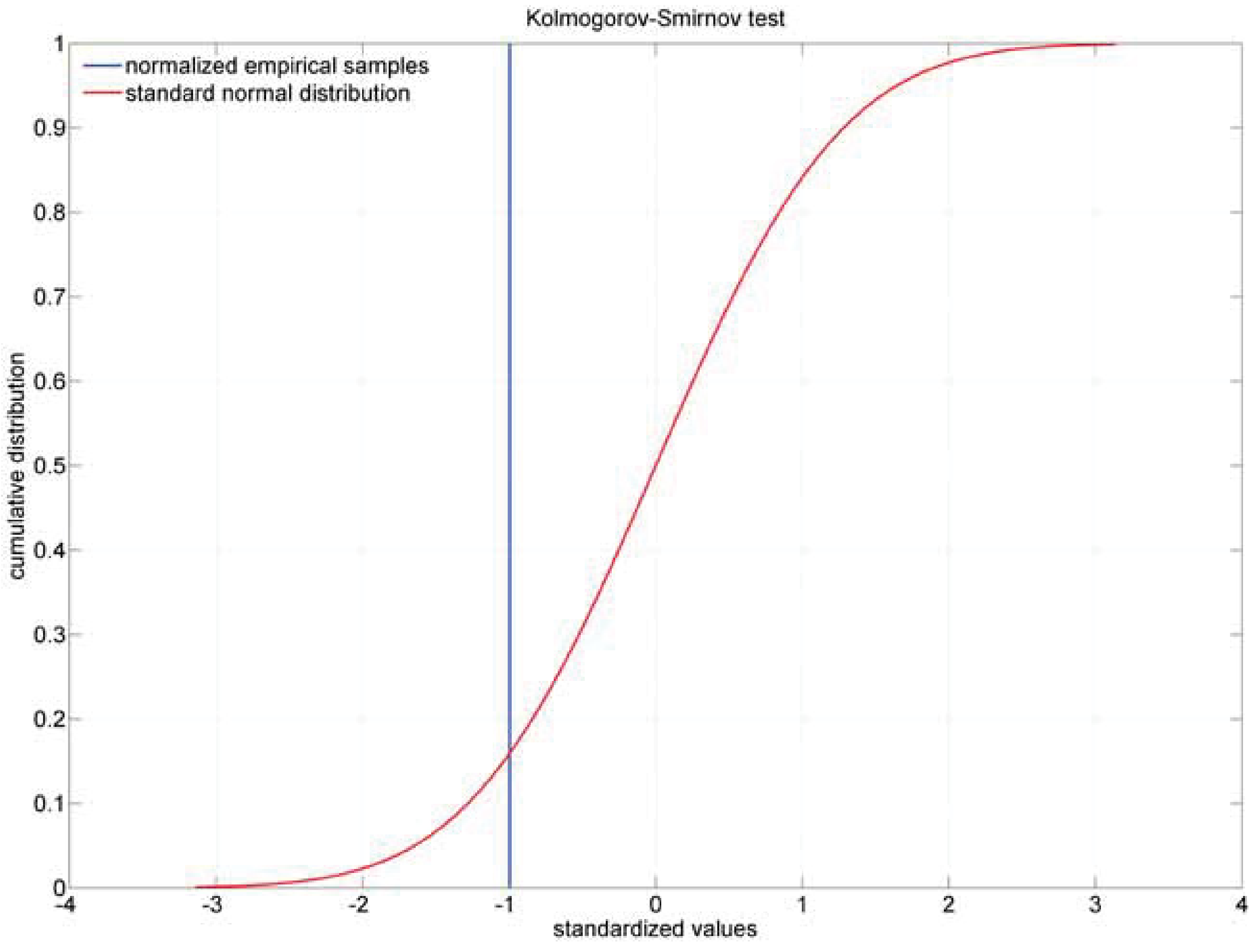


Figure C.21: K-S test of LED real coordinates in x dimension via tensorBFS.

Statistical analysis:		
$\hat{a}$	100.01	mm
$\tilde{a}$	100.01	mm
$\bar{a}$	100.01	mm
s^2	$2.4683\ 10^{26}$	mm^2
s	$1.5711\ 10^{13}$	mm
e	$1.5711\ 10^{14}$	mm
$e_{95\%}$	$3.0793\ 10^{14}$	mm
$e_{99\%}$	$3.9748\ 10^{14}$	mm
$\hat{\gamma}_1$	-1	
$\hat{\beta}_2$	1.0	
K-S test	H_0 rejected	

Table C.7: Stat. analysis of LED real coordinates in x dimension via tensorBFS.

TC#1: Single LED / TensorBFS / y Dimension

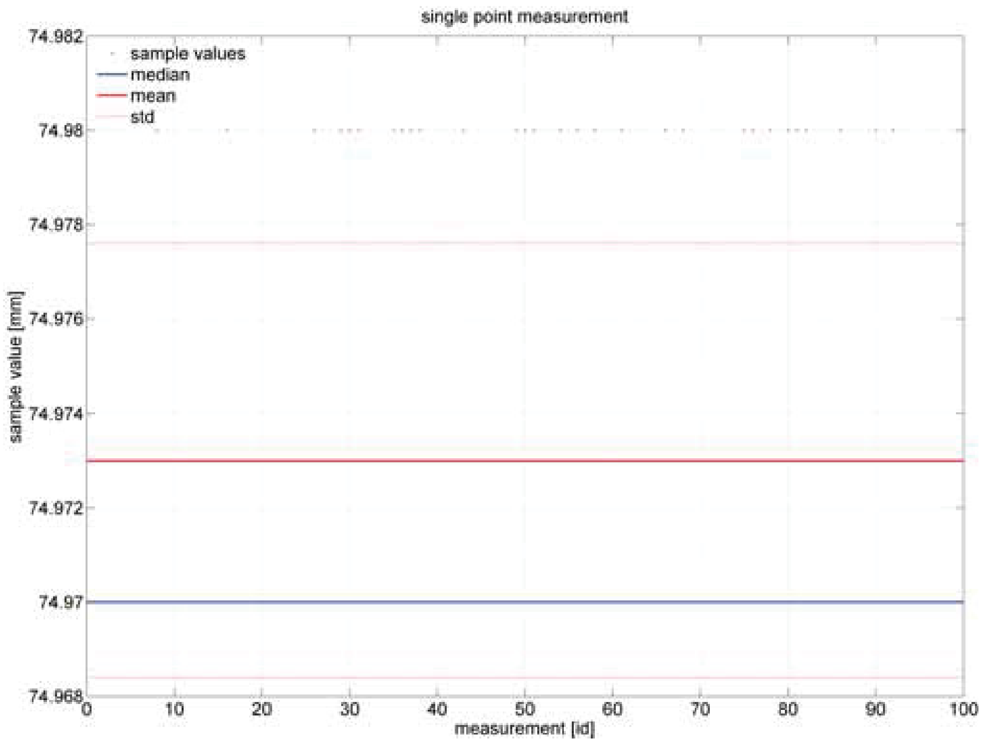


Figure C.22: LED real coordinates in y dimension via tensorBFS.

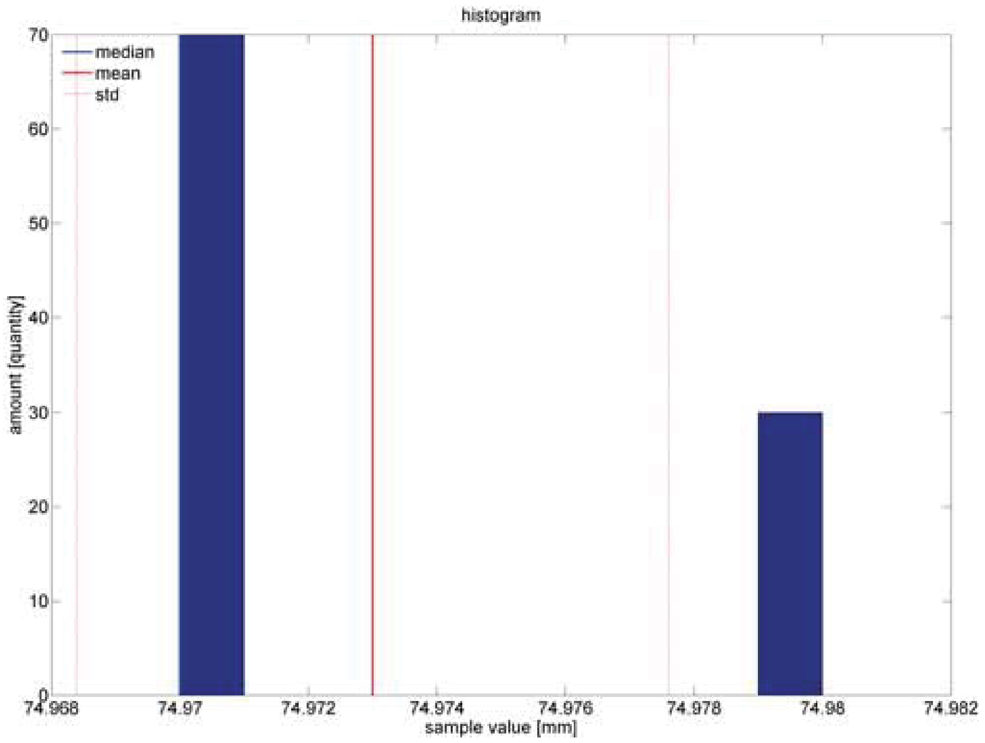


Figure C.23: Histogram of LED real coordinates in y dimension via tensorBFS.

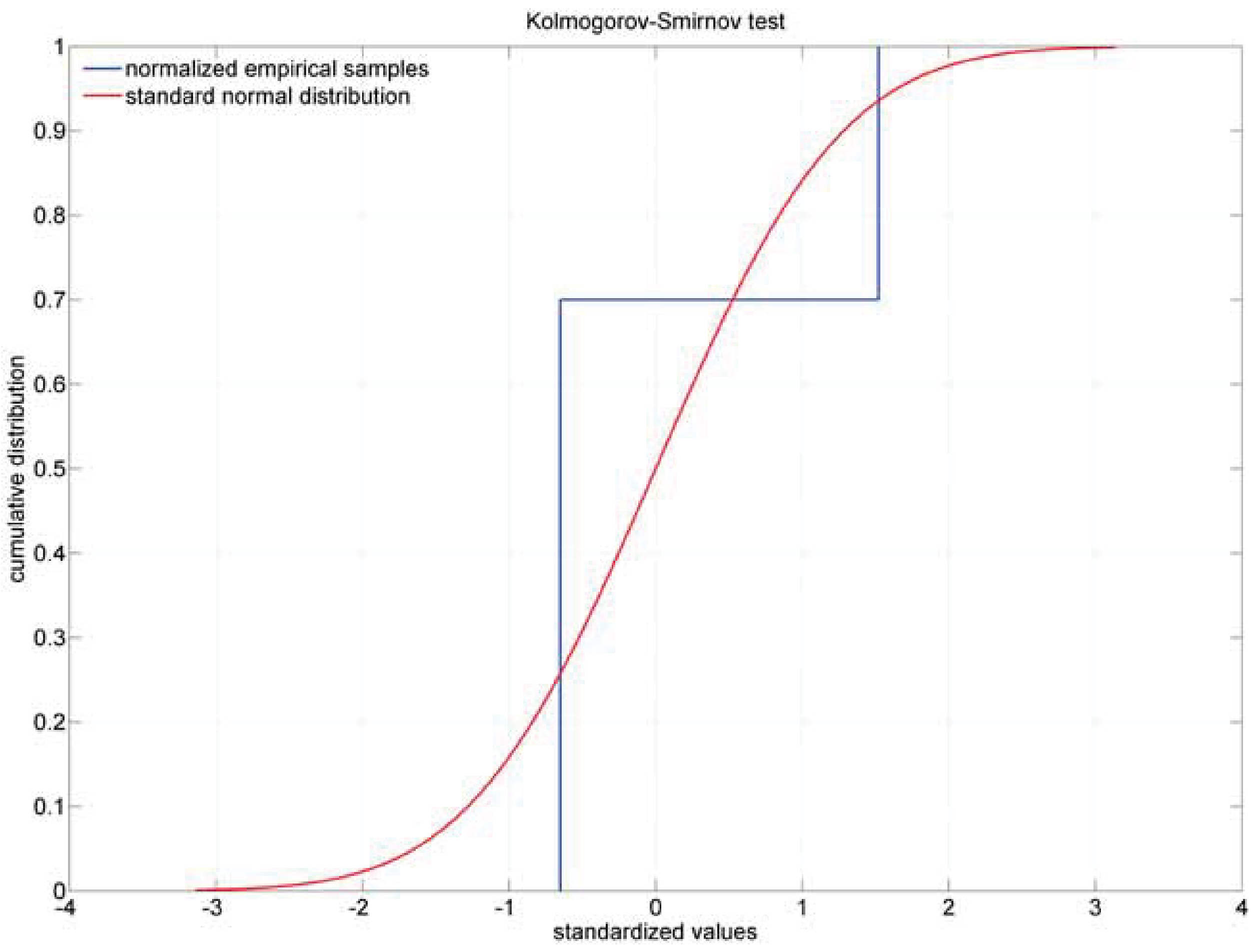


Figure C.24: K-S test of LED real coordinates in y dimension via tensorBFS.

Statistical analysis:		
$\hat{a}$	74.97	mm
$\tilde{a}$	74.97	mm
$\bar{a}$	74.973	mm
s^2	0.000021212	mm^2
s	0.0046057	mm
e	0.00046057	mm
$e_{95\%}$	0.00090271	mm
$e_{99\%}$	0.0011652	mm
$\hat{\gamma}_1$	0.87287	
$\hat{\beta}_2$	1.7619	
K-S test	H_0 rejected	

Table C.8: Stat. analysis of LED real coordinates in y dimension via tensorBFS.

TC#1: Single LED / TensorQTD / x Dimension

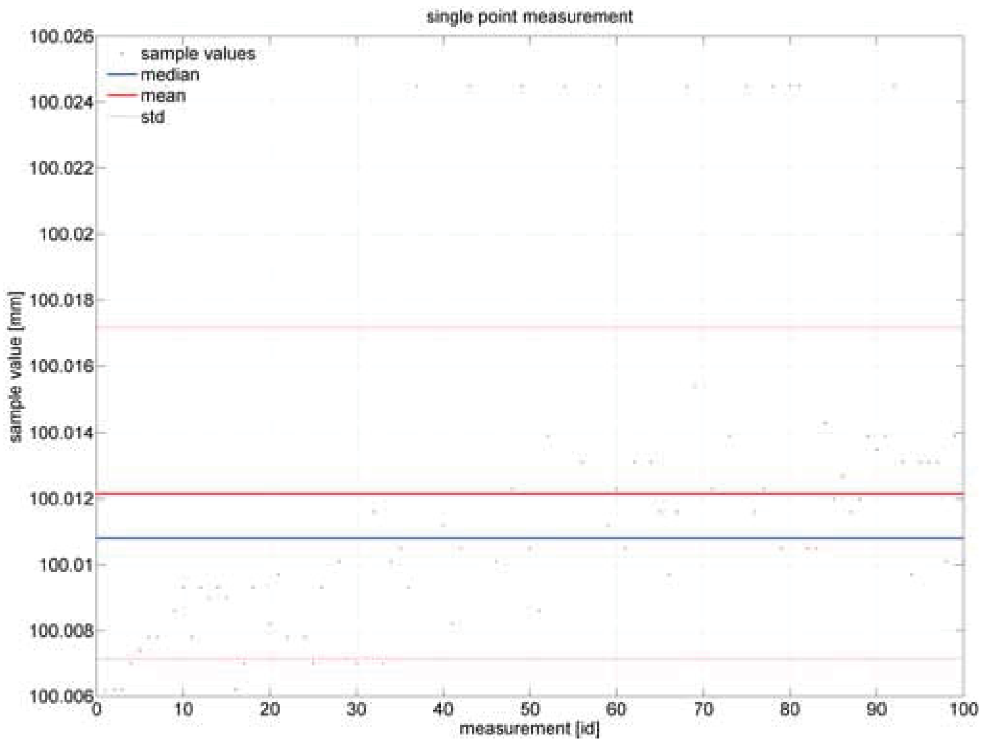


Figure C.25: LED real coordinates in x dimension via tensorQTD.

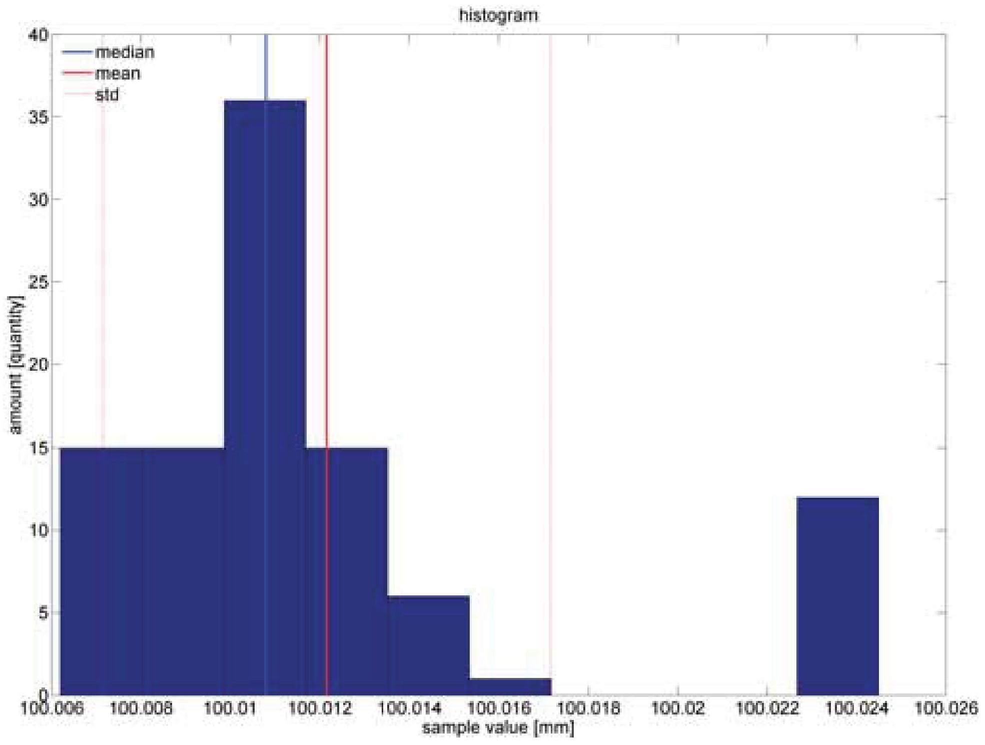


Figure C.26: Histogram of LED real coordinates in x dimension via tensorQTD.

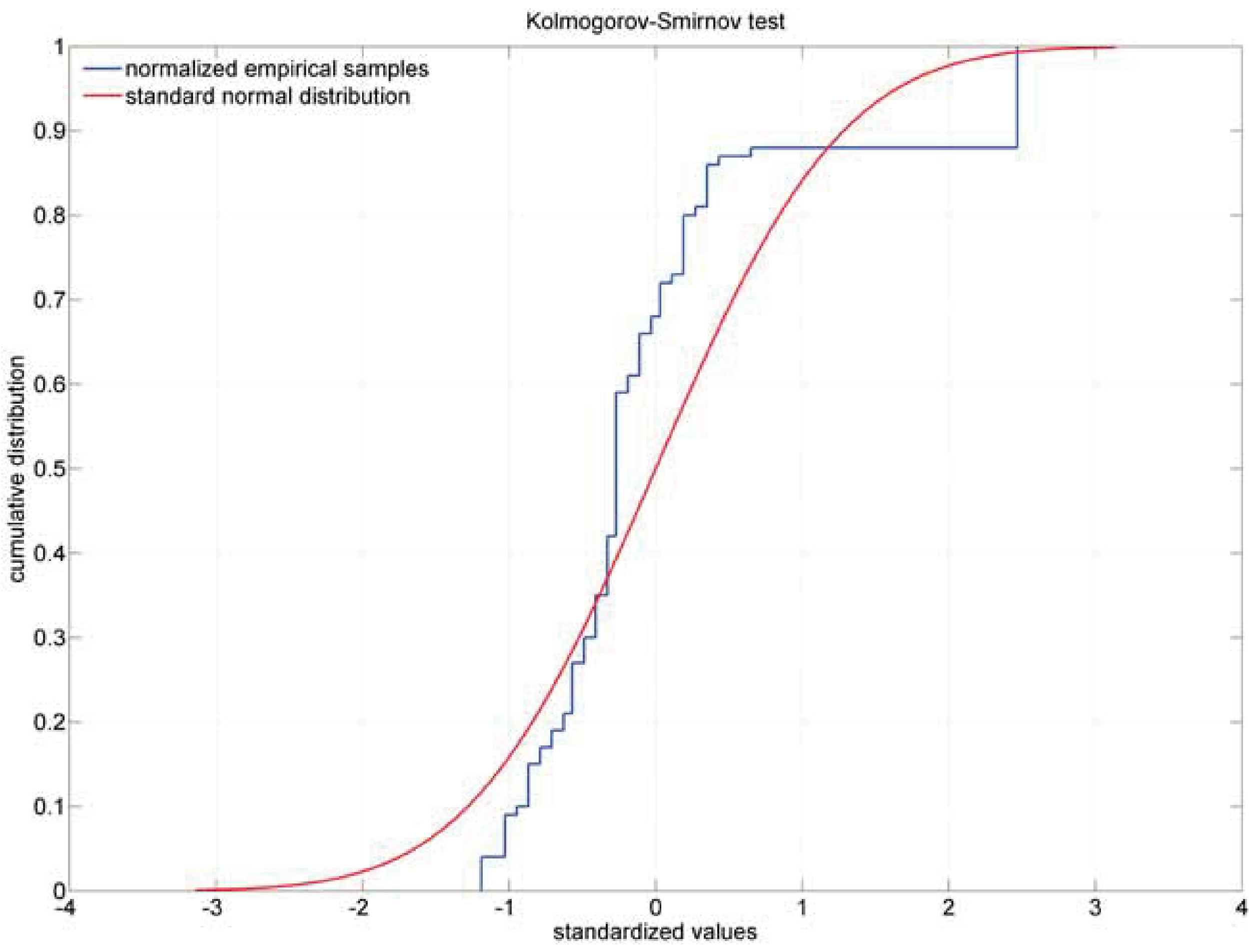


Figure C.27: K-S test of LED real coordinates in x dimension via tensorQTD.

Statistical analysis:		
$\hat{a}$	100.0108	mm
$\tilde{a}$	100.0108	mm
$\bar{a}$	100.0122	mm
s^2	0.000025031	mm^2
s	0.0050031	mm
e	0.00050031	mm
$e_{95\%}$	0.00098062	mm
$e_{99\%}$	0.0012658	mm
$\hat{\gamma}_1$	1.6263	
$\hat{\beta}_2$	4.753	
K-S test	H_0 rejected	

Table C.9: Stat. analysis of LED real coordinates in x dimension via tensorQTD.

TC#1: Single LED / TensorQTD / y Dimension

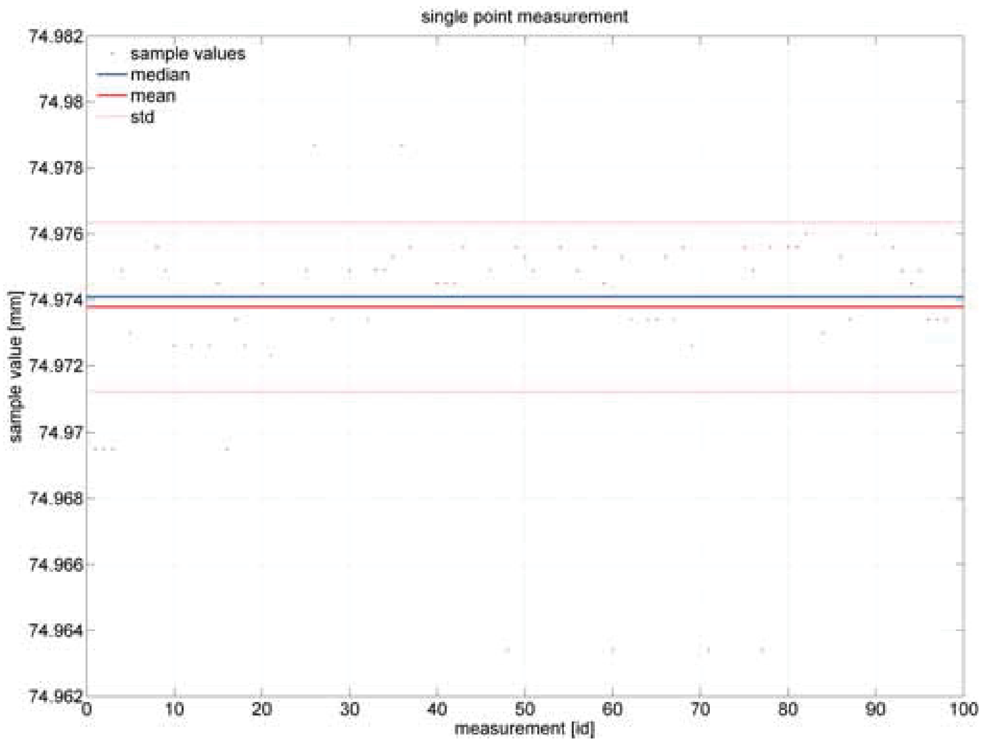


Figure C.28: LED real coordinates in y dimension via tensorQTD.

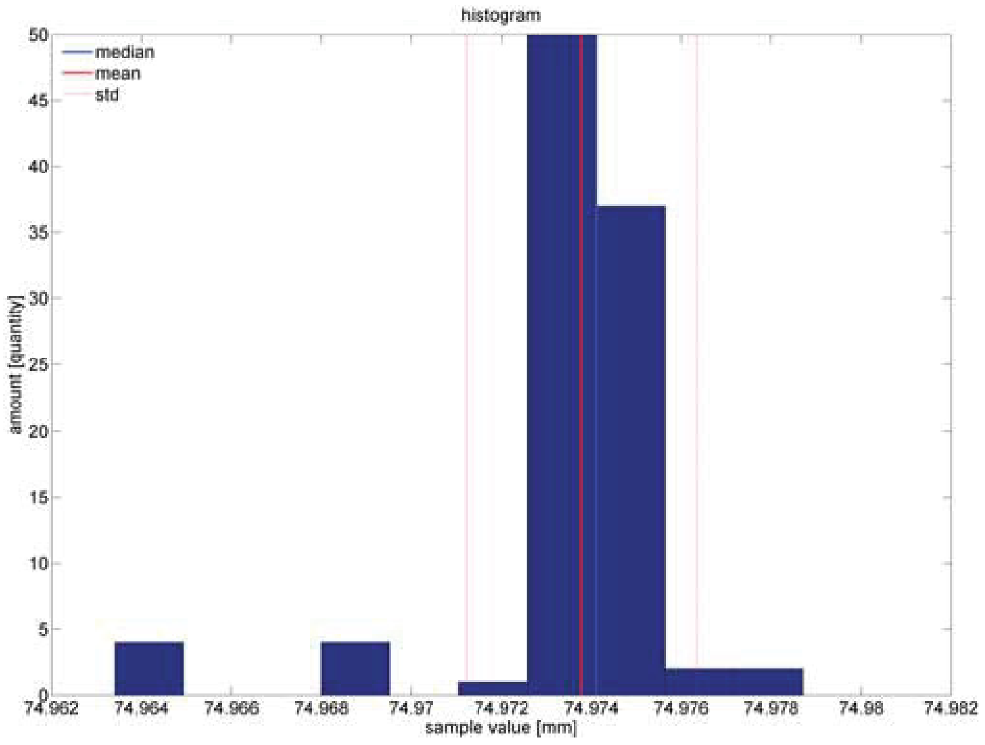


Figure C.29: Histogram of LED real coordinates in y dimension via tensorQTD.

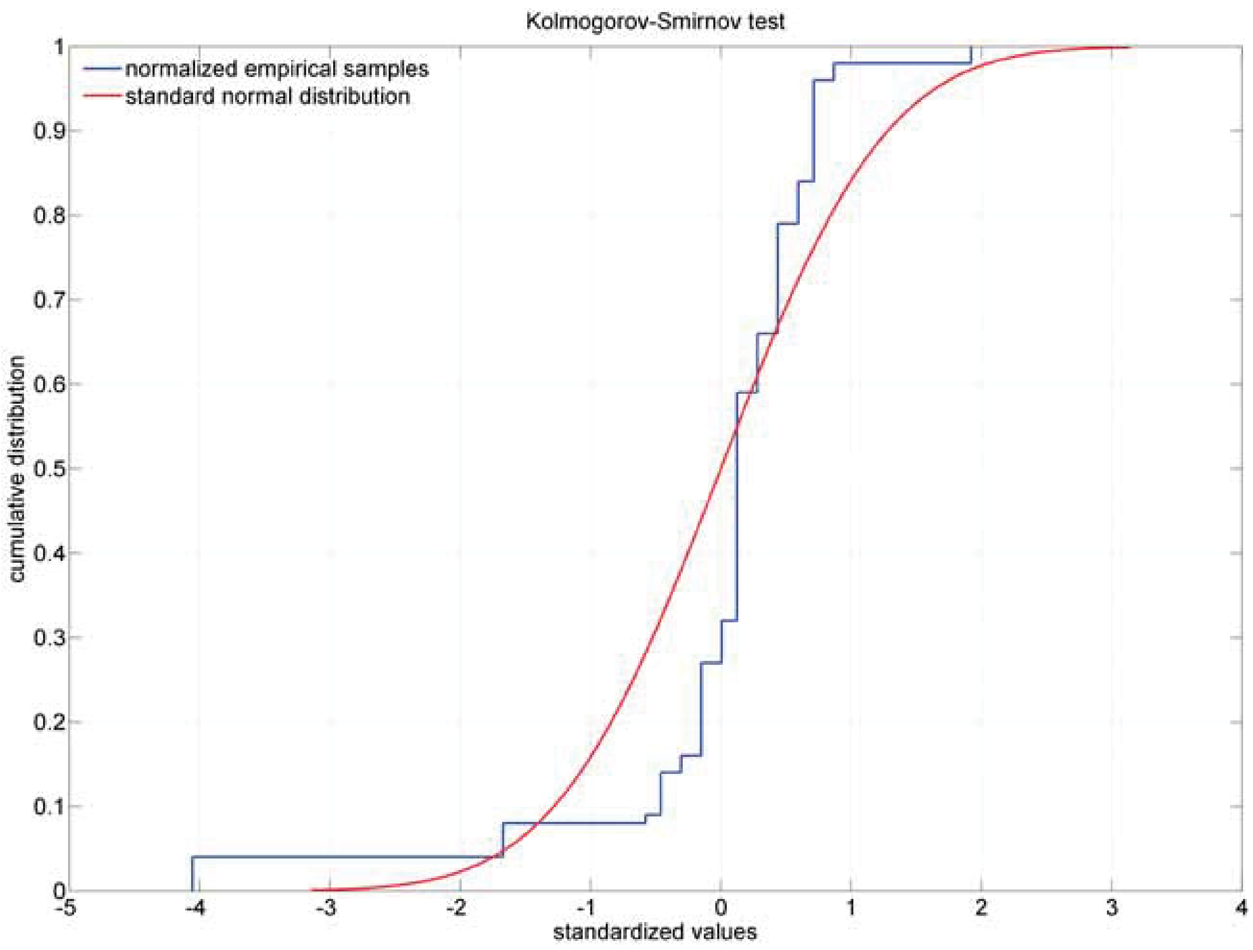


Figure C.30: K-S test of LED real coordinates in y dimension via tensorQTD.

Statistical analysis:		
$\hat{a}$	74.9741	mm
$\tilde{a}$	74.9741	mm
$\bar{a}$	100.0122	mm
s^2	0.0000065503	mm^2
s	0.0025594	mm
e	0.00025594	mm
$e_{95\%}$	0.00050163	mm
$e_{99\%}$	0.00064752	mm
$\hat{\gamma}_1$	-2.6852	
$\hat{\beta}_2$	11.7109	
K-S test	H_0 rejected	

Table C.10: Stat. analysis of LED real coordinates in y dimension via tensorQTD.

TC#2: Single Laser Spot / Camera Samples / x Dimension

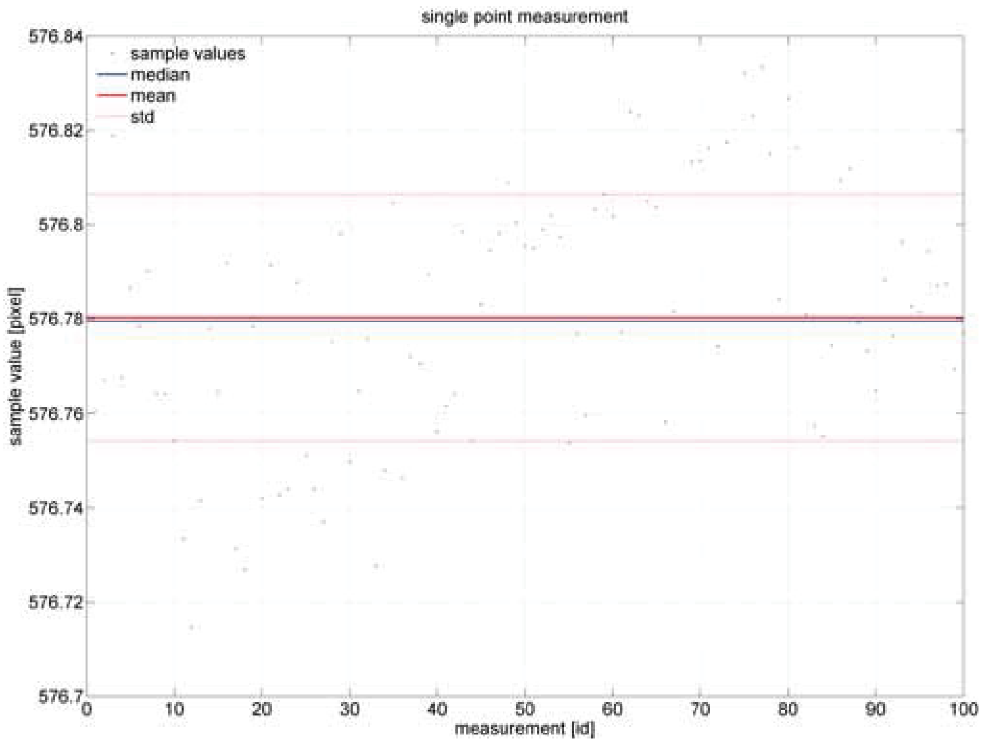


Figure C.31: Measured laser spot camera coordinates in x dimension.

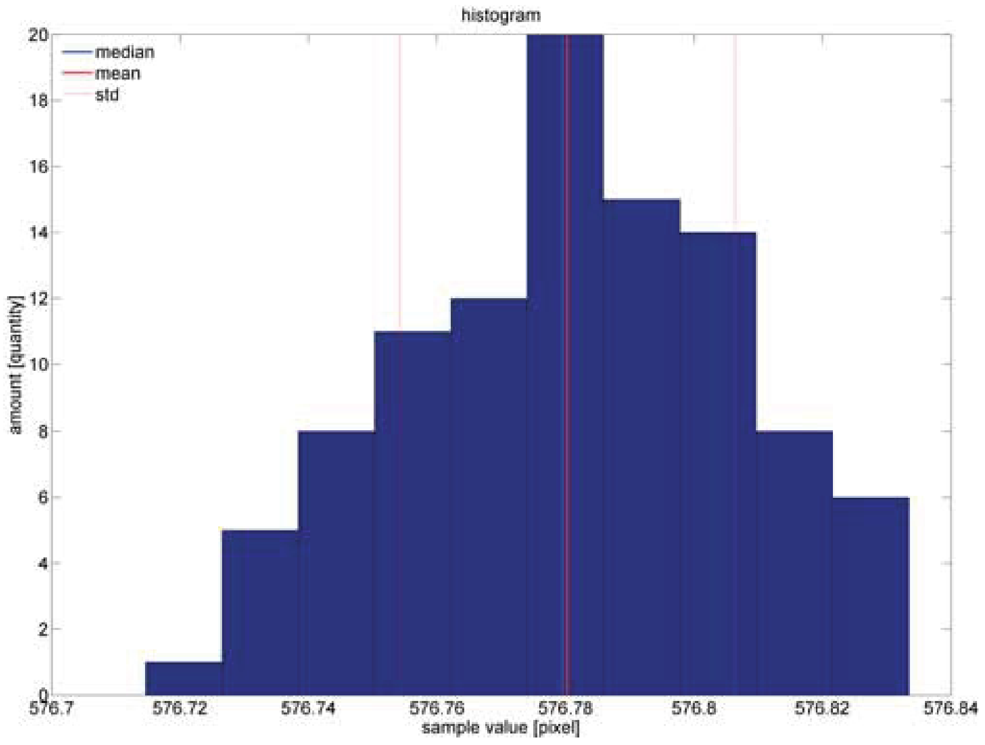


Figure C.32: Histogram of measured laser spot camera coordinates in x dimension.

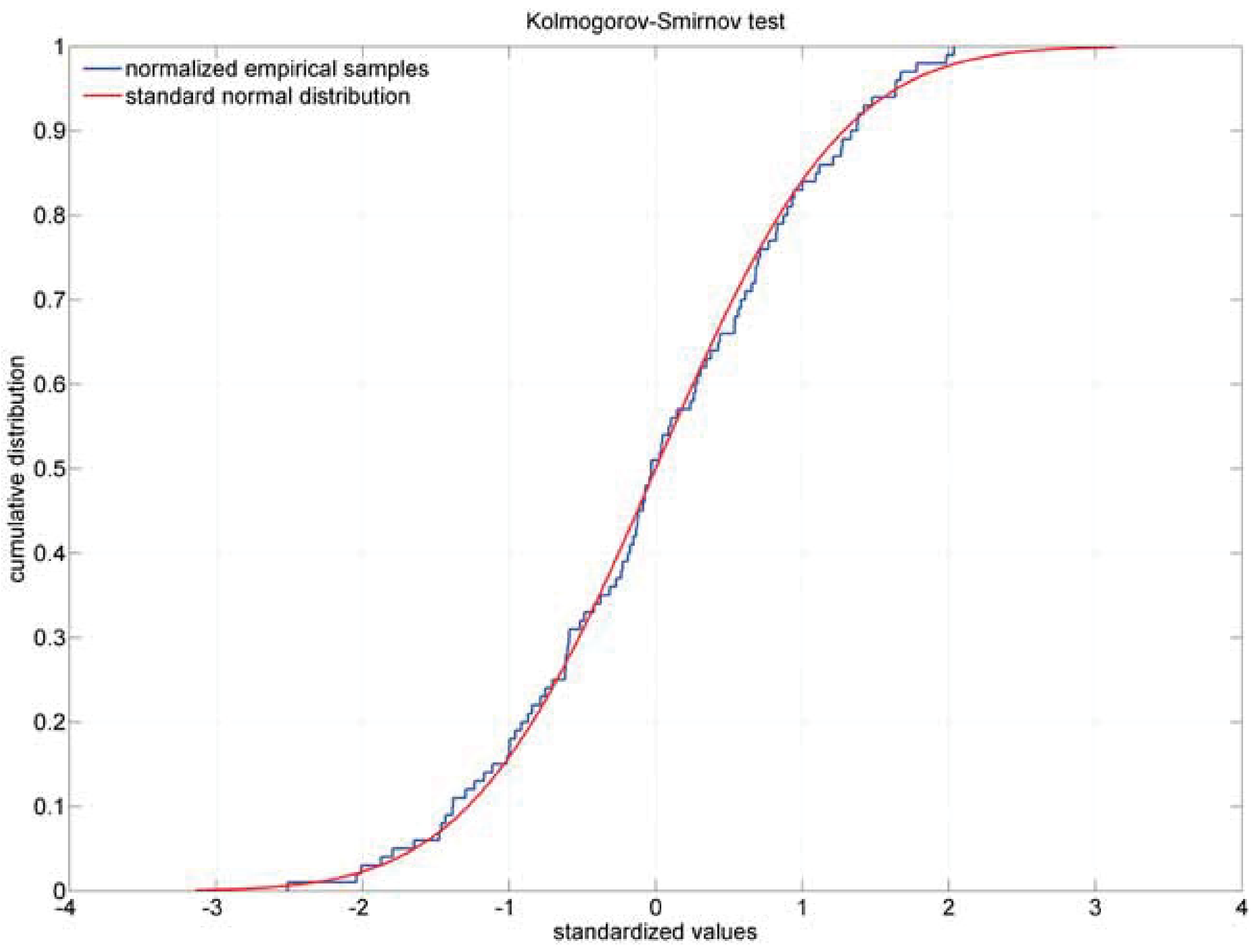


Figure C.33: K-S test of measured laser spot camera coordinates in x dimension.

Statistical analysis:		
$\hat{a}$	576.7147	*pixel*
$\tilde{a}$	576.7795	*pixel*
$\bar{a}$	576.7803	*pixel*
s^2	0.00068303	$pixel^2$
s	0.026135	*pixel*
e	0.0026135	*pixel*
$e_{95\%}$	0.0051224	*pixel*
$e_{99\%}$	0.0066121	*pixel*
$\hat{\gamma}_1$	-0.14242	
$\hat{\beta}_2$	2.469	
K-S test	H_0 accepted	

Table C.11: Stat. analysis of measured laser spot camera coordinates in x dimension.

TC#2: Single Laser Spot / Camera Samples / y Dimension

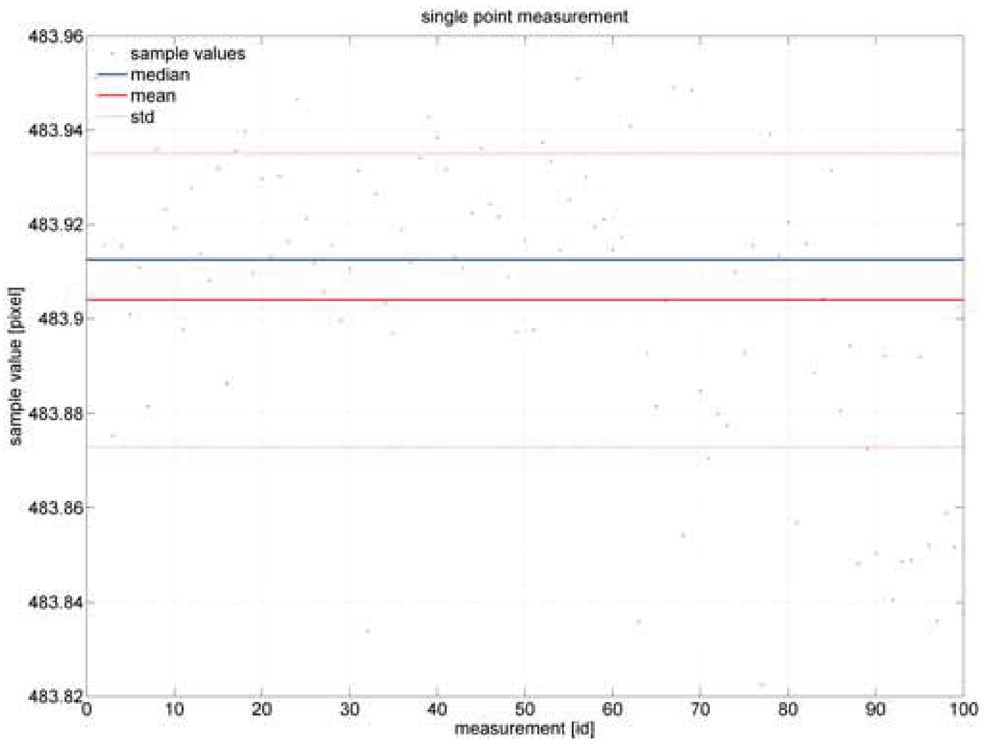


Figure C.34: Measured laser spot camera coordinates in y dimension.

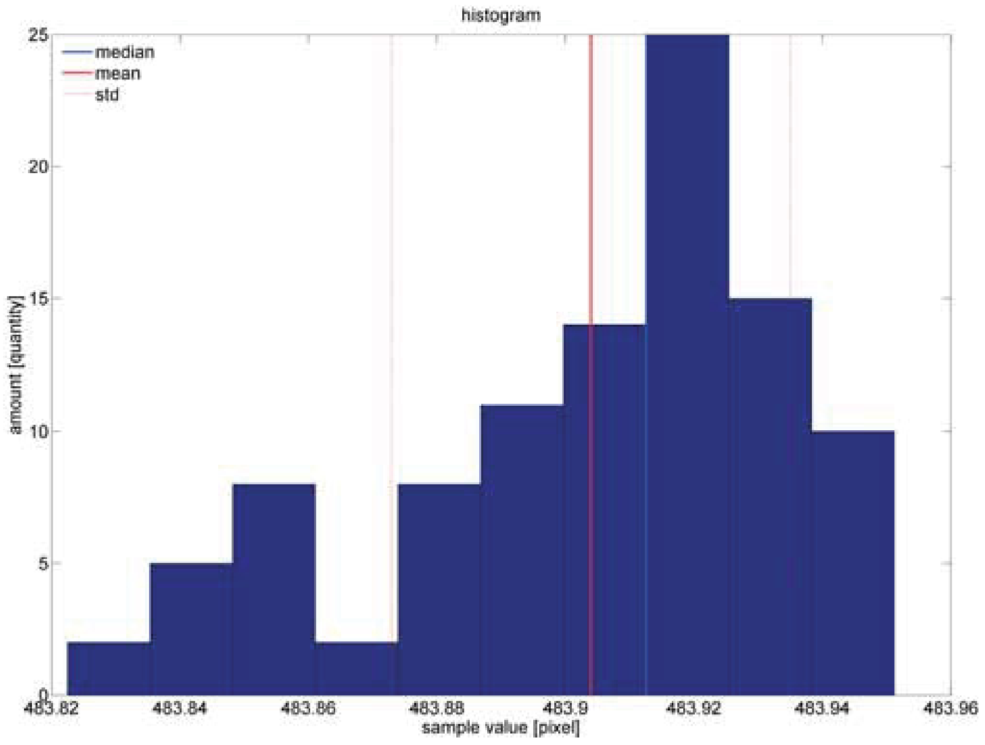


Figure C.35: Histogram of measured laser spot camera coordinates in y dimension.

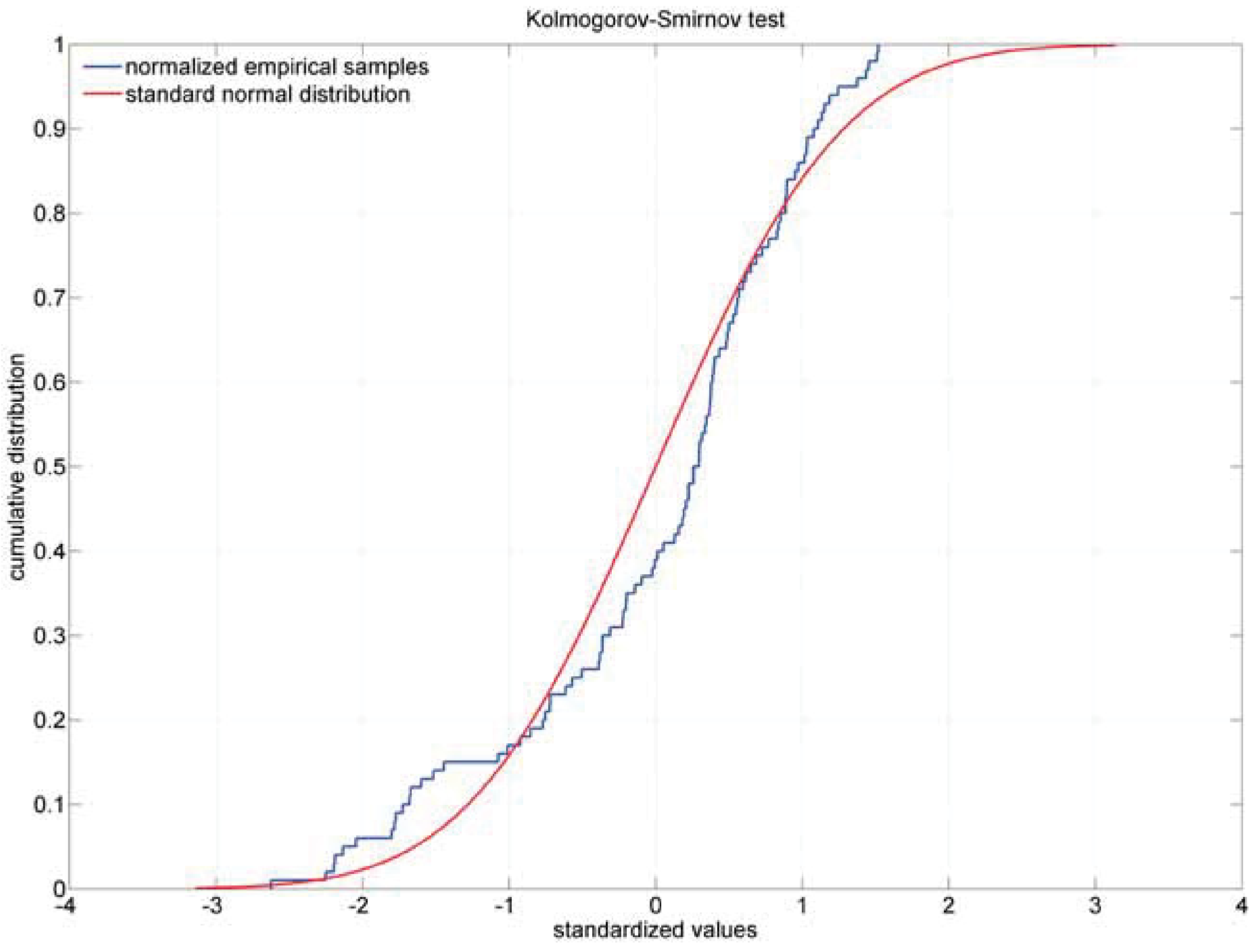


Figure C.36: K-S test of measured laser spot camera coordinates in y dimension.

Statistical analysis:		
$\hat{a}$	483.8225	$pixel$
$\tilde{a}$	483.9126	$pixel$
$\bar{a}$	483.904	$pixel$
s^2	0.00096485	$pixel^2$
s	0.031062	$pixel$
e	0.0031062	$pixel$
$e_{95\%}$	0.0060882	$pixel$
$e_{99\%}$	0.0078587	$pixel$
$\hat{\gamma}_1$	-0.80396	
$\hat{\beta}_2$	2.8444	
K-S test	H_0 accepted	

Table C.12: Stat. analysis of measured laser spot camera coordinates in y dimension.

TC#2: Single Laser Spot / Homography / x Dimension

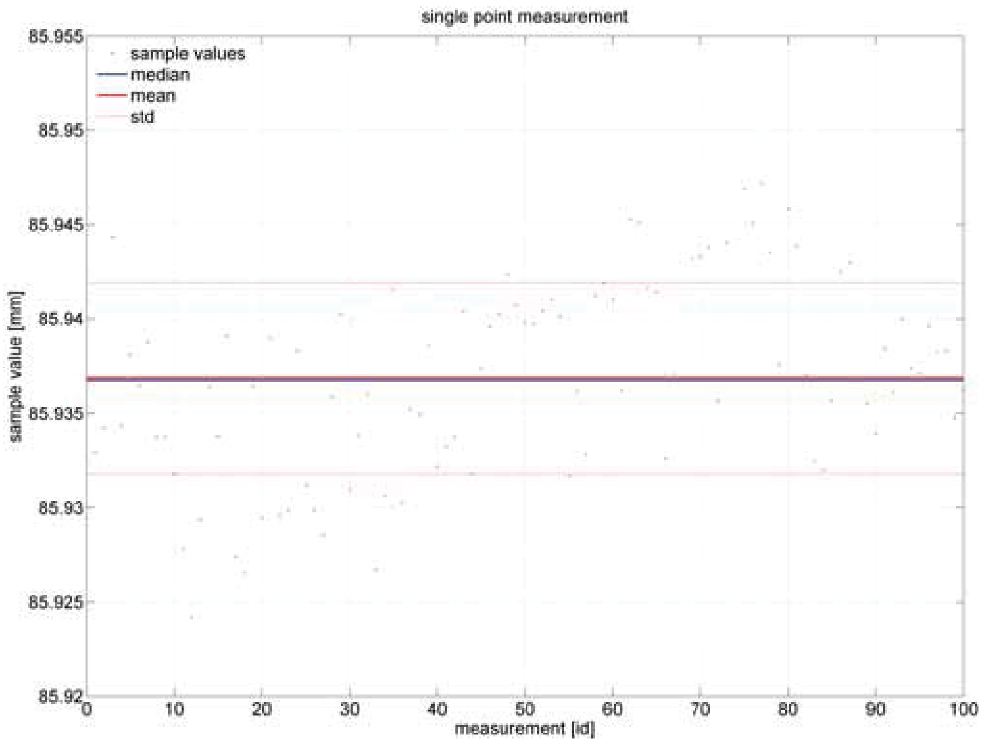


Figure C.37: Laser spot real coordinates in x dimension via homography.

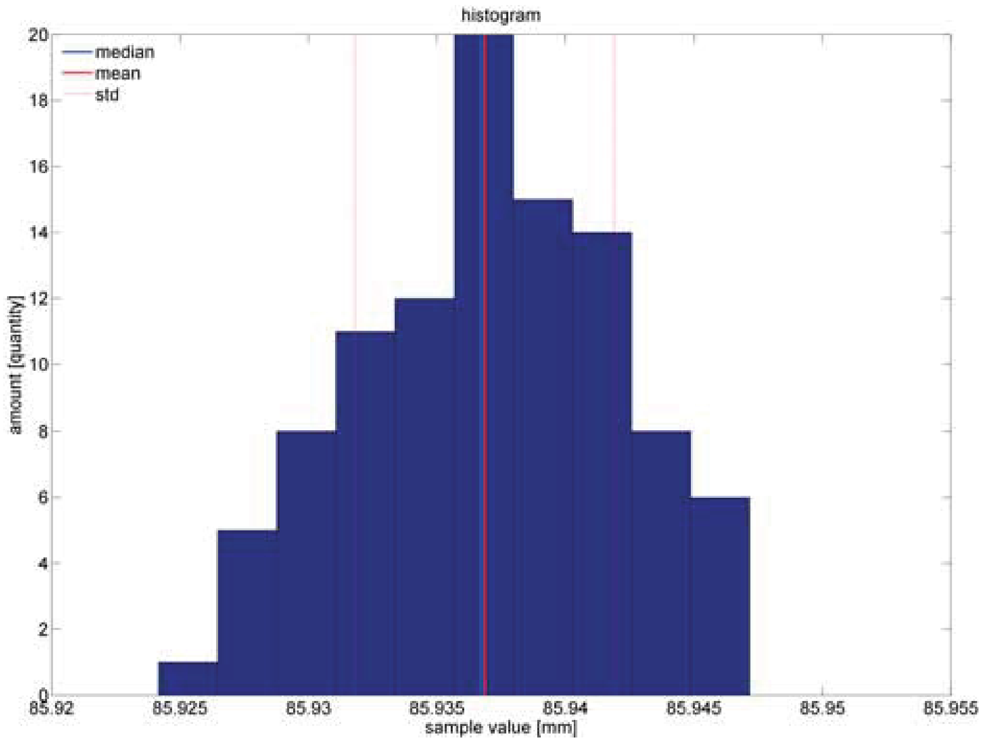


Figure C.38: Histogram of laser spot real coordinates in x dimension via homography.

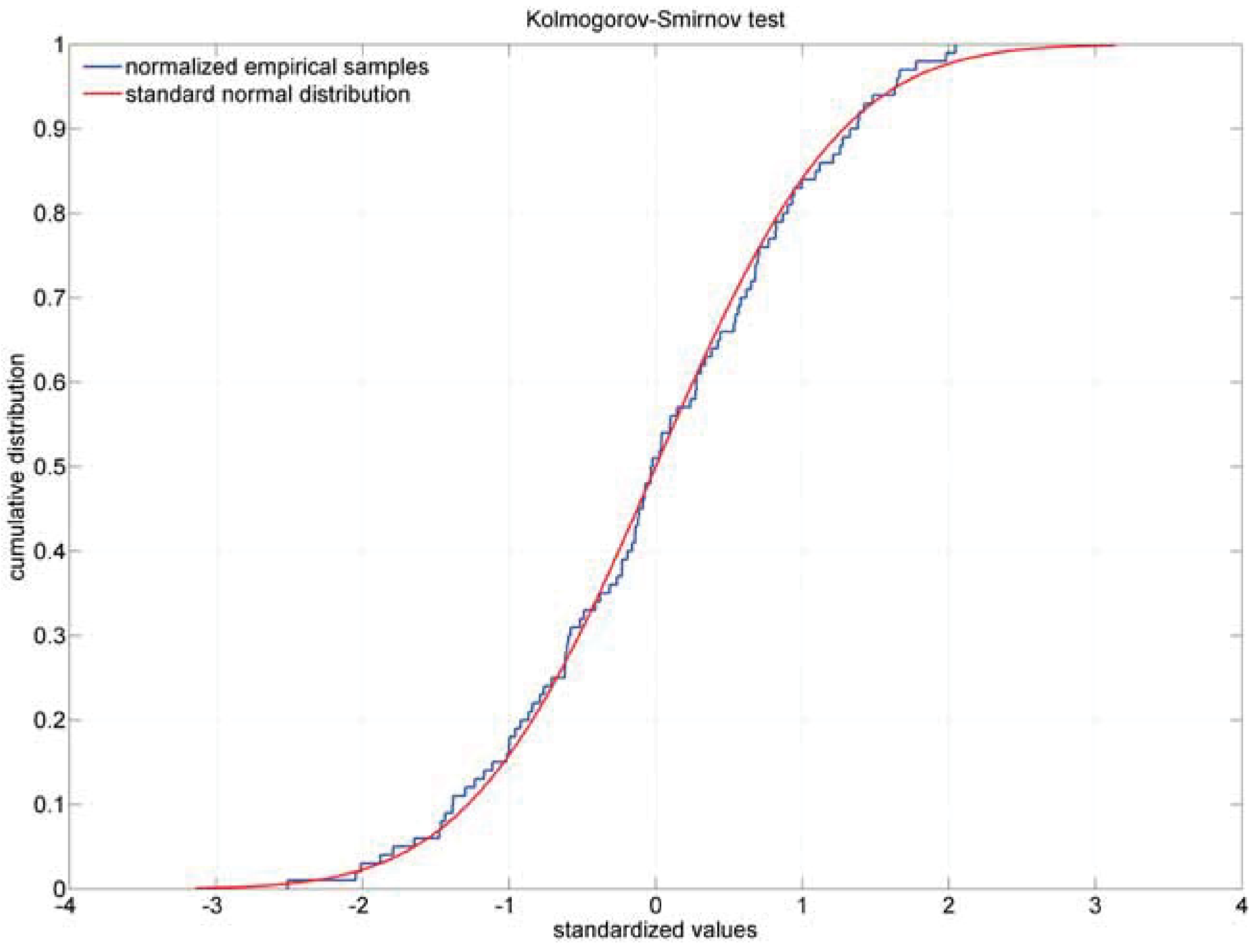


Figure C.39: K-S test of laser spot real coordinates in x dimension via homography.

Statistical analysis:		
$\hat{a}$	85.9242	mm
$\tilde{a}$	85.9367	mm
$\bar{a}$	85.9369	mm
s^2	0.000025491	mm^2
s	0.0050488	mm
e	0.00050488	mm
$e_{95\%}$	0.00098957	mm
$e_{99\%}$	0.0012774	mm
$\hat{\gamma}_1$	-0.14322	
$\hat{\beta}_2$	2.4712	
K-S test	H_0 accepted	

Table C.13: Stat. analysis of laser spot real coordinates in x dimension via homography.

TC#2: Single Laser Spot / Homography / y Dimension

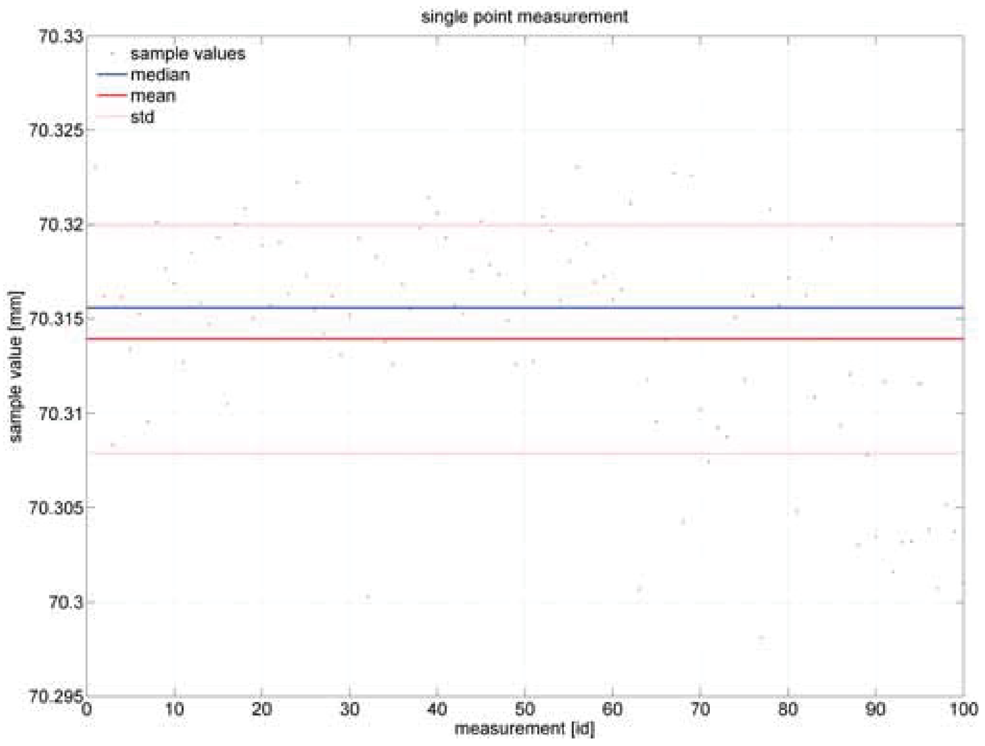


Figure C.40: Laser spot real coordinates in y dimension via homography.

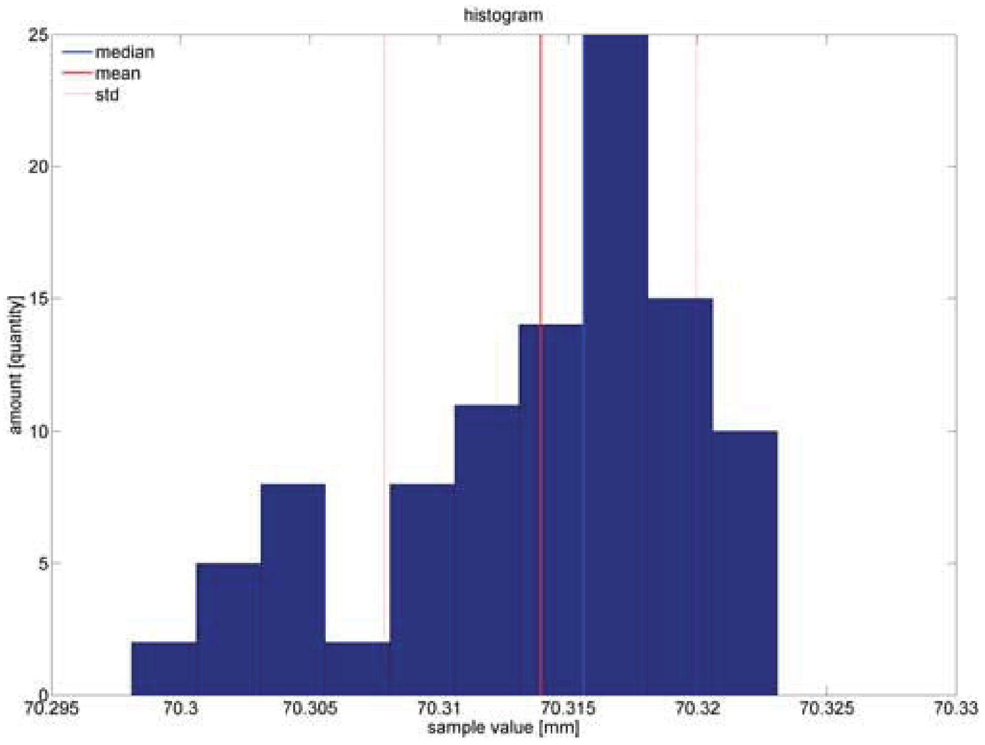


Figure C.41: Histogram of laser spot real coordinates in y dimension via homography.

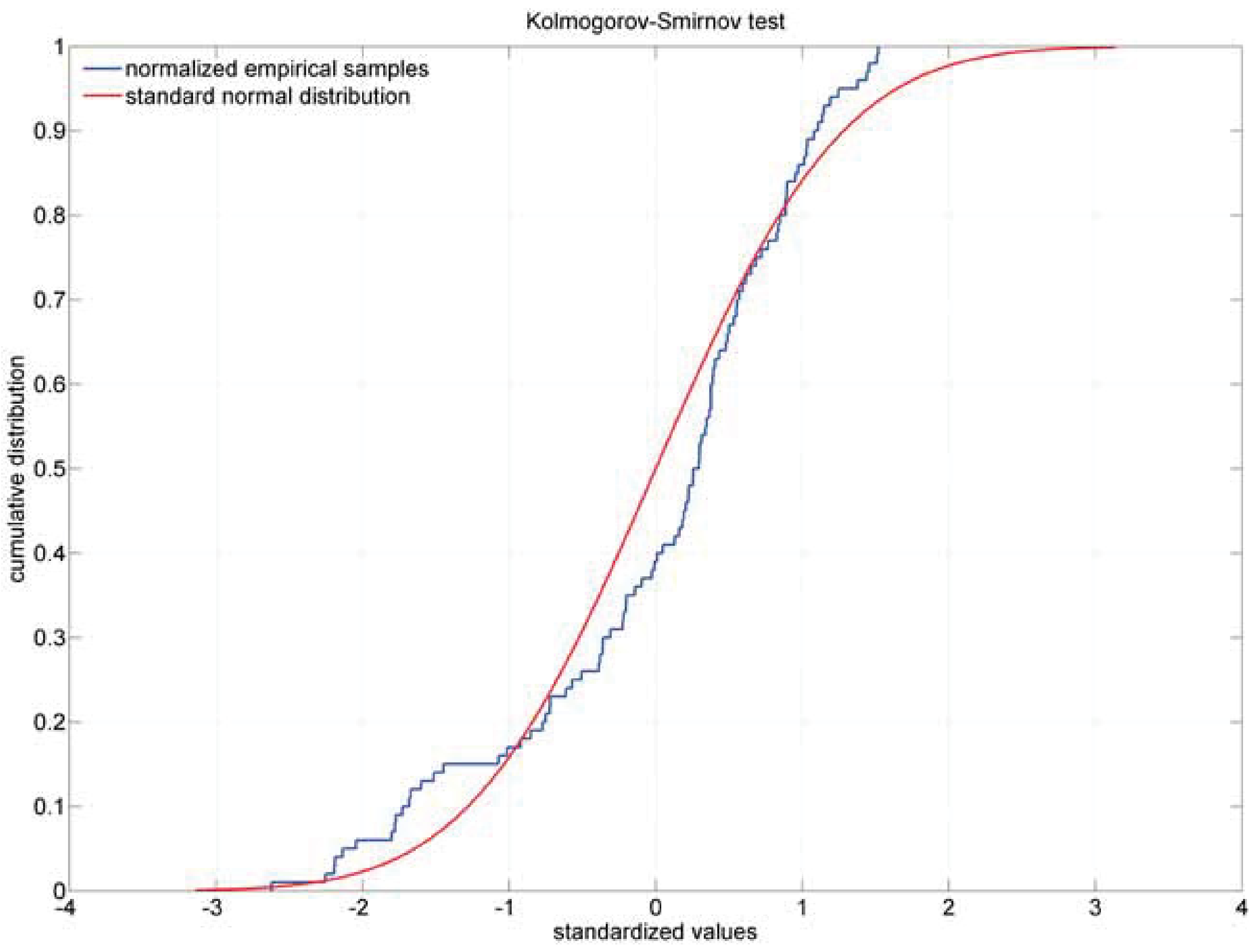


Figure C.42: K-S test of laser spot real coordinates in y dimension via homography.

Statistical analysis:		
$\hat{a}$	70.2981	mm
$\tilde{a}$	70.3156	mm
$\bar{a}$	70.3139	mm
s^2	0.000036425	mm^2
s	0.0060353	mm
e	0.0011829	mm
$e_{95\%}$	0.0011829	mm
$e_{99\%}$	0.0015269	mm
$\hat{\gamma}_1$	-0.80386	
$\hat{\beta}_2$	2.8454	
K-S test	H_0 accepted	

Table C.14: Stat. analysis of laser spot real coordinates in y dimension via homography.

TC#2: Single Laser Spot / PolyCtrlPts / x Dimension

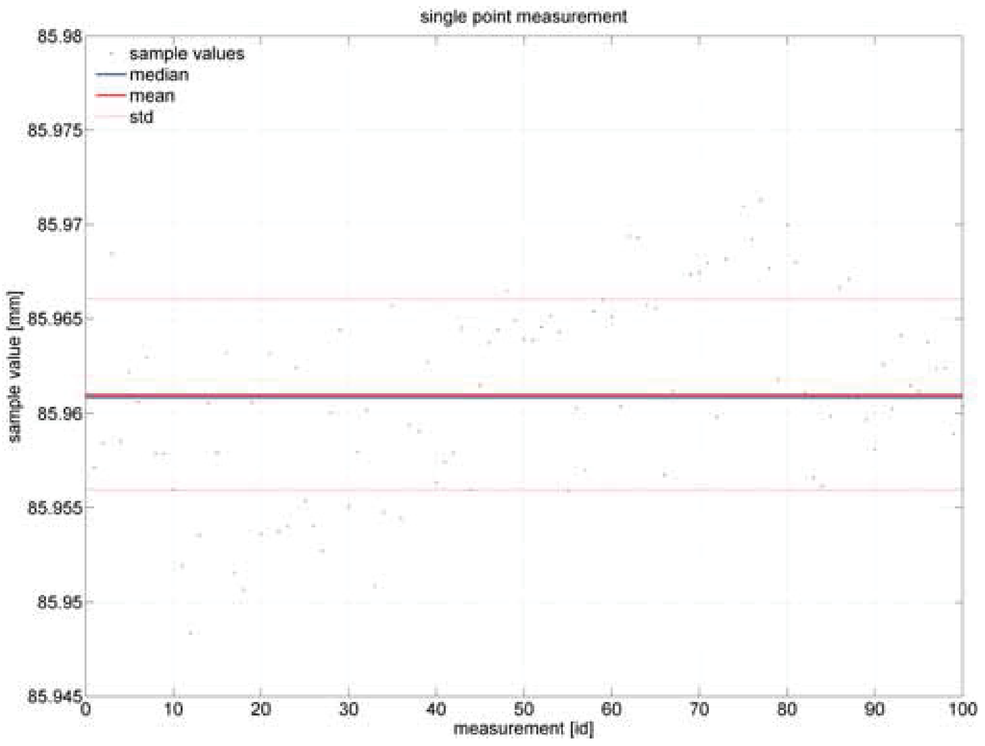


Figure C.43: Laser spot real coordinates in x dimension via polyCtrlPts.

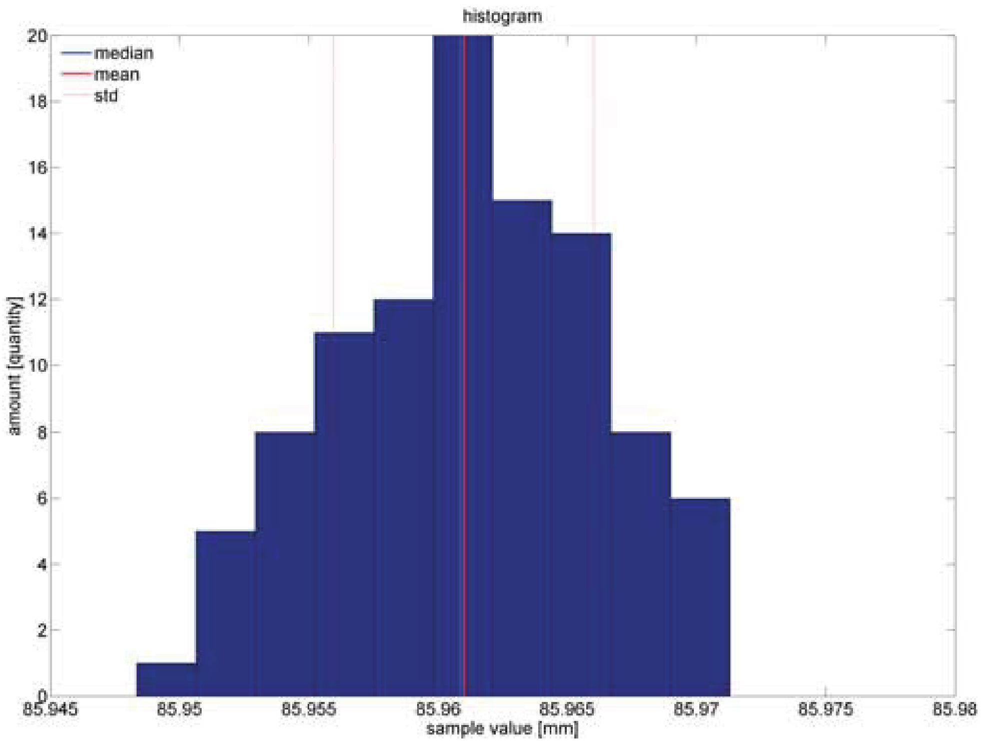


Figure C.44: Histogram of laser spot real coordinates in x dimension via polyCtrlPts.

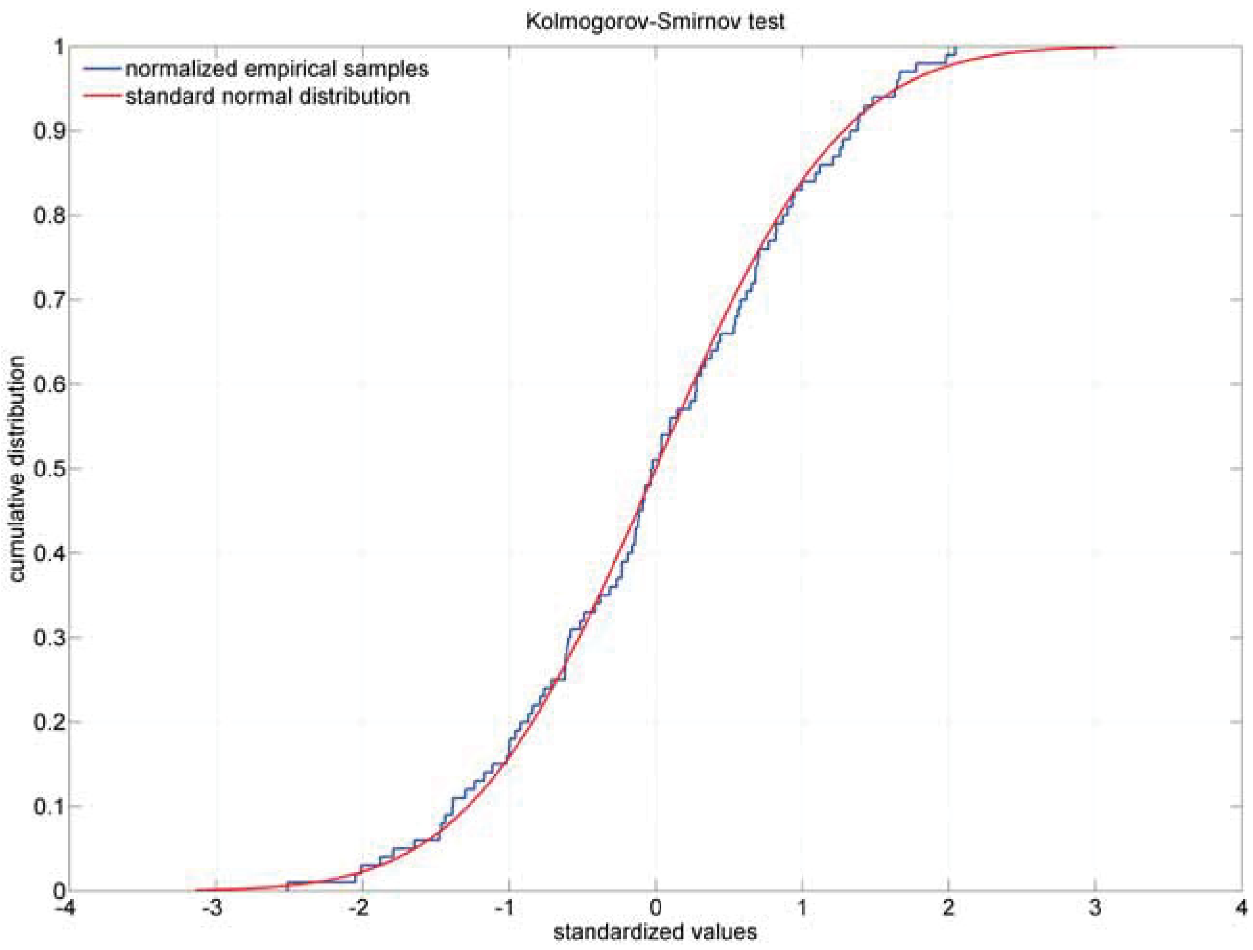


Figure C.45: K-S test of laser spot real coordinates in x dimension via polyCtrlPts.

Statistical analysis:		
$\hat{a}$	85.9483	mm
$\tilde{a}$	85.9609	mm
$\bar{a}$	85.961	mm
s^2	0.000025406	mm^2
s	0.0050404	mm
e	0.00050404	mm
$e_{95\%}$	0.00098793	mm
$e_{99\%}$	0.0012752	mm
$\hat{\gamma}_1$	-0.14319	
$\hat{\beta}_2$	2.4711	
K-S test	H_0 rejected	

Table C.15: Stat. analysis of laser spot real coordinates in x dimension via polyCtrlPts.

TC#2: Single Laser Spot / PolyCtrlPts / y Dimension

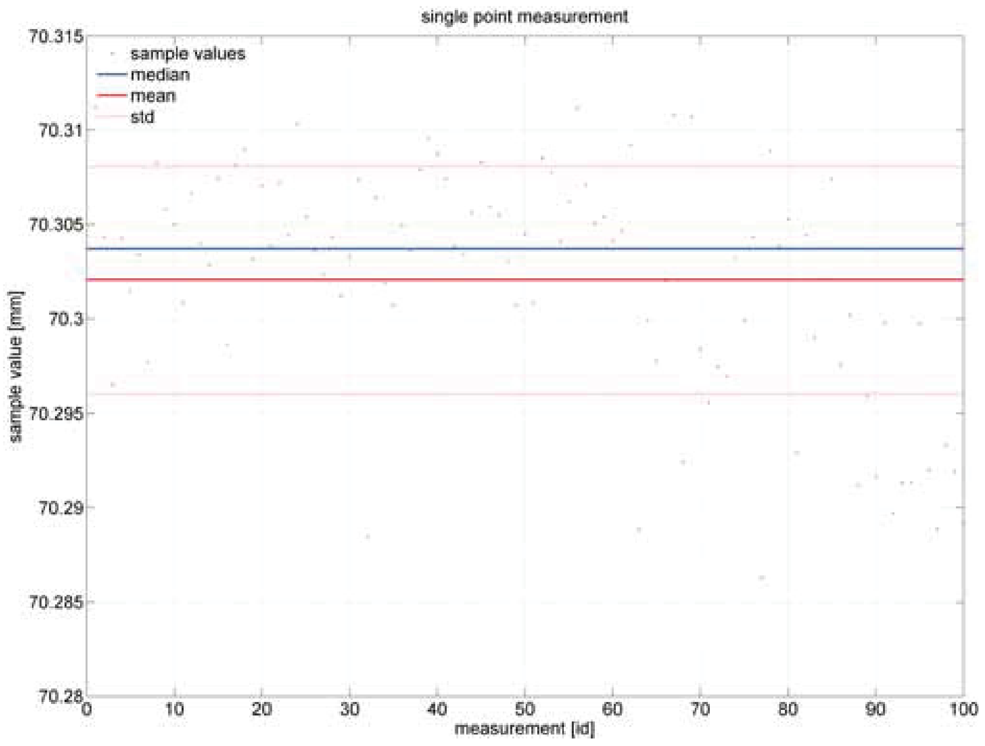


Figure C.46: Laser spot real coordinates in y dimension via polyCtrlPts.

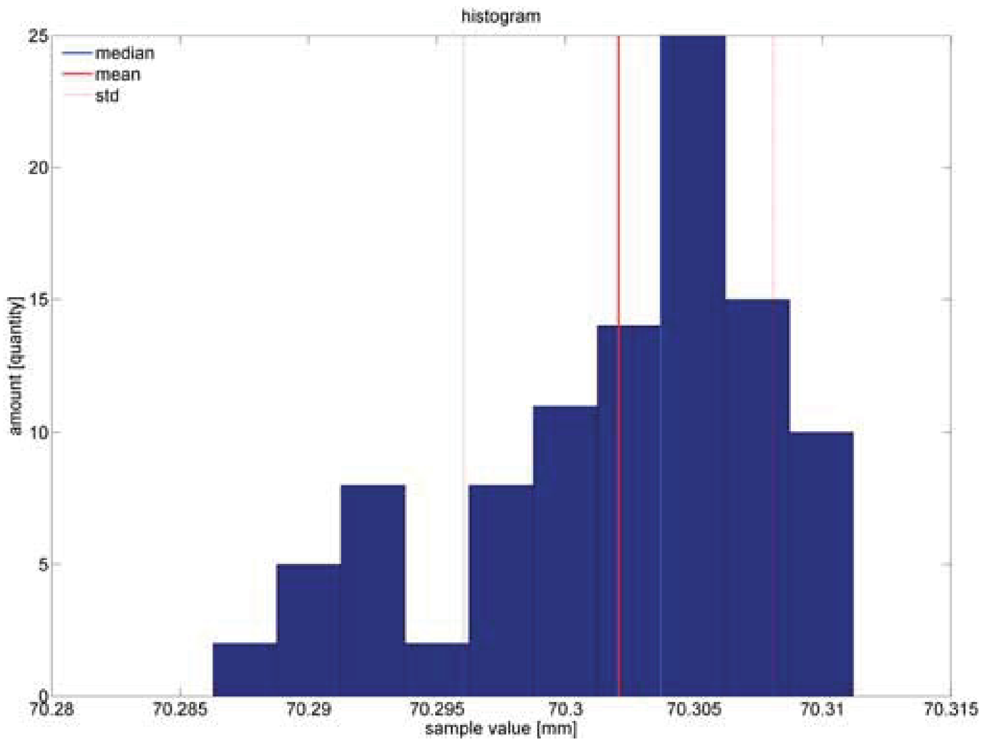


Figure C.47: Histogram of laser spot real coordinates in y dimension via polyCtrlPts.

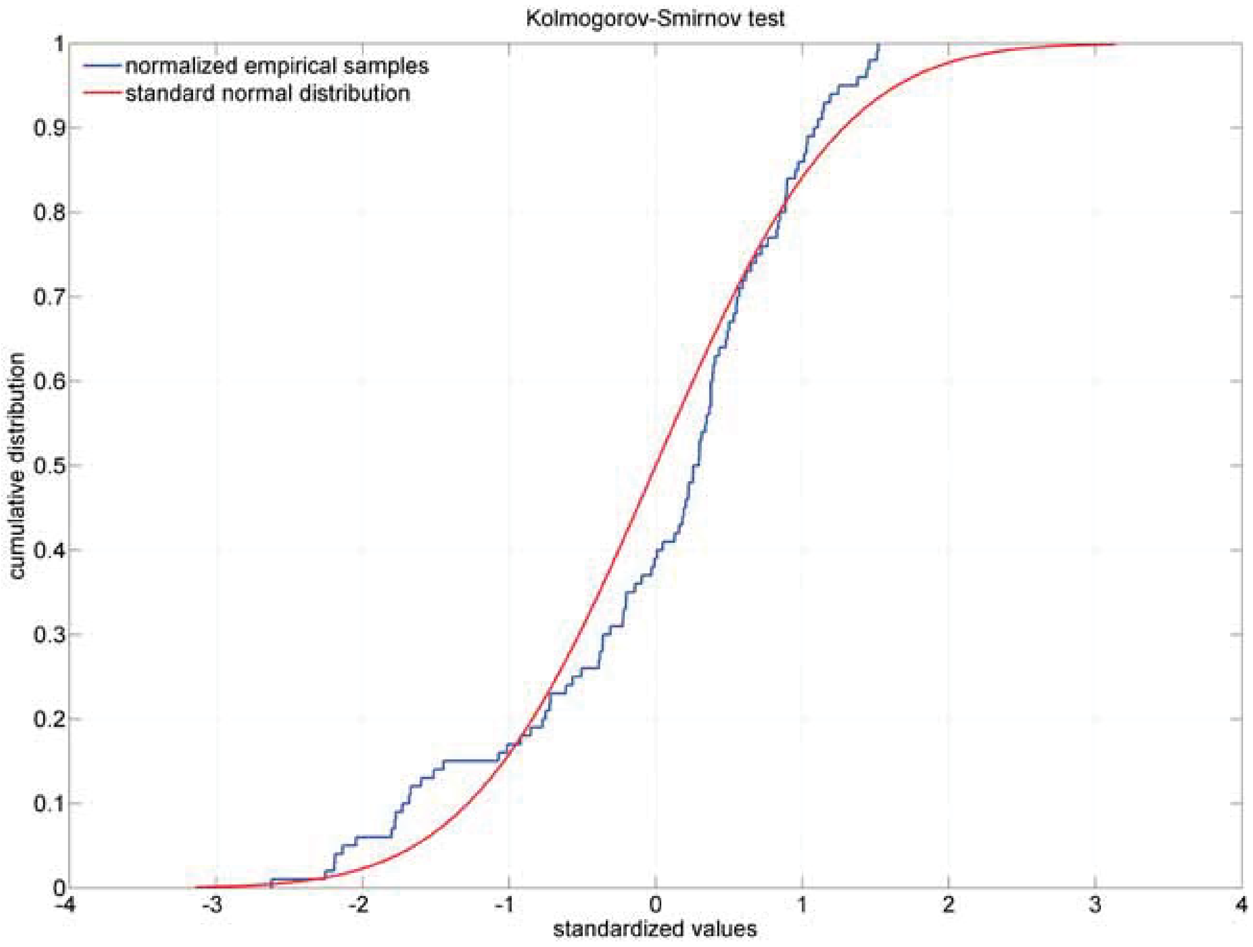


Figure C.48: K-S test of laser spot real coordinates in y dimension via polyCtrlPts.

Statistical analysis:		
$\hat{a}$	70.2863	mm
$\tilde{a}$	70.3037	mm
$\bar{a}$	70.3021	mm
s^2	0.000036253	mm^2
s	0.006021	mm
e	0.0006021	mm
$e_{95\%}$	0.0011801	mm
$e_{99\%}$	0.0015233	mm
$\hat{\gamma}_1$	-0.80386	
$\hat{\beta}_2$	2.8454	
K-S test	H_0 accepted	

Table C.16: Stat. analysis of laser spot real coordinates in y dimension via polyCtrlPts.

TC#2: Single Laser Spot / TensorBFS / x Dimension

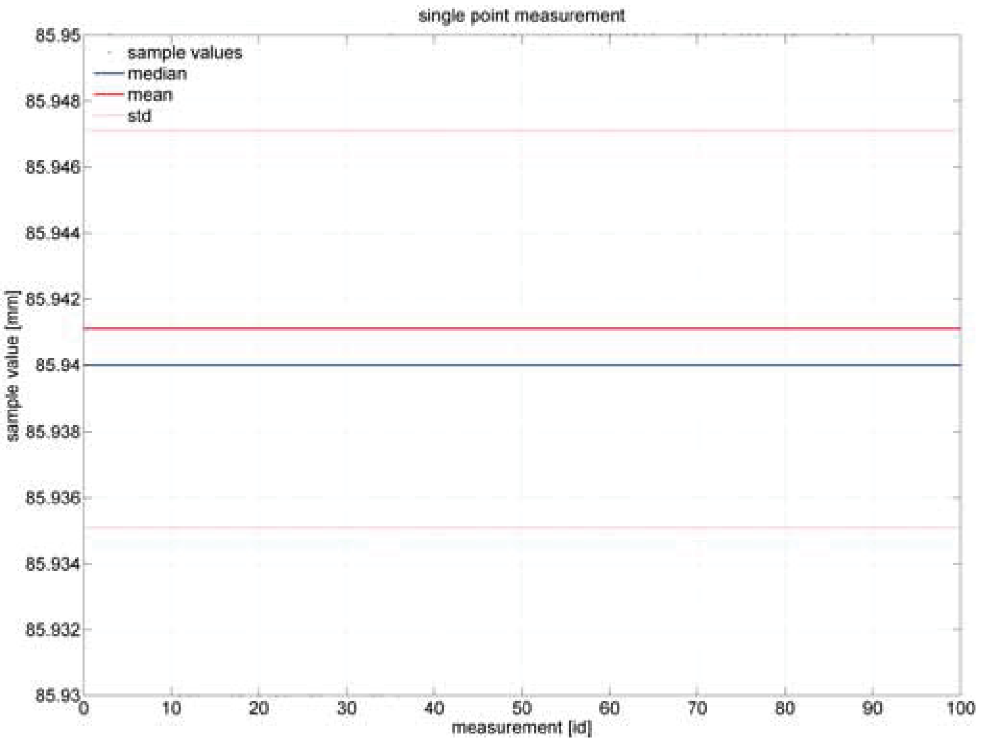


Figure C.49: Laser spot real coordinates in x dimension via tensorBFS.

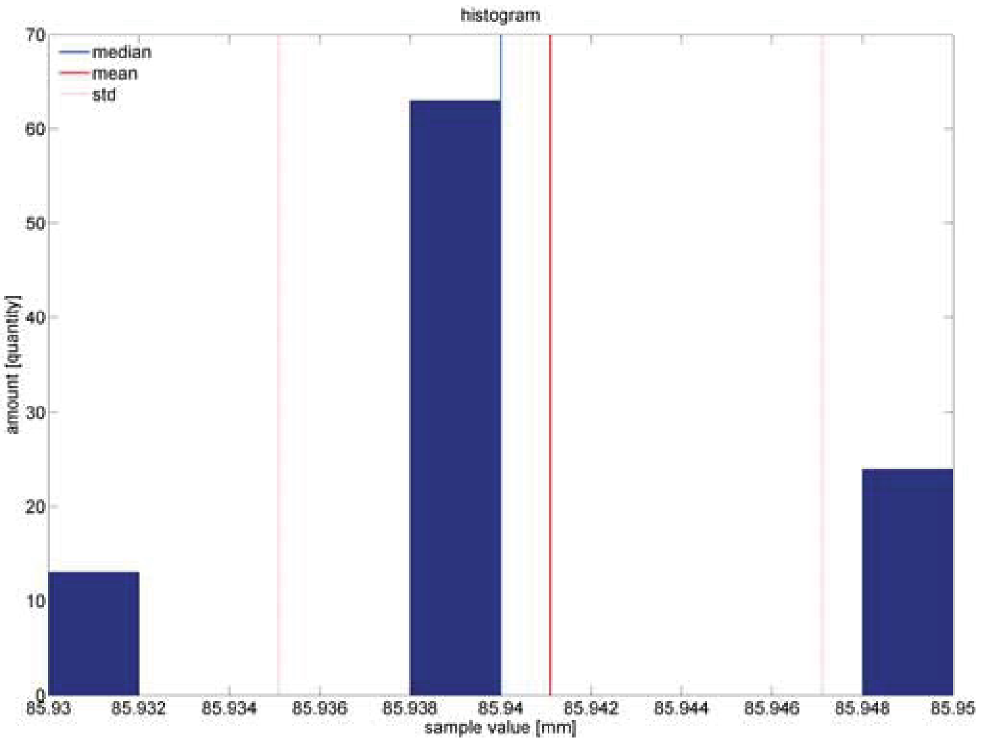


Figure C.50: Histogram of laser spot real coordinates in x dimension via tensorBFS.

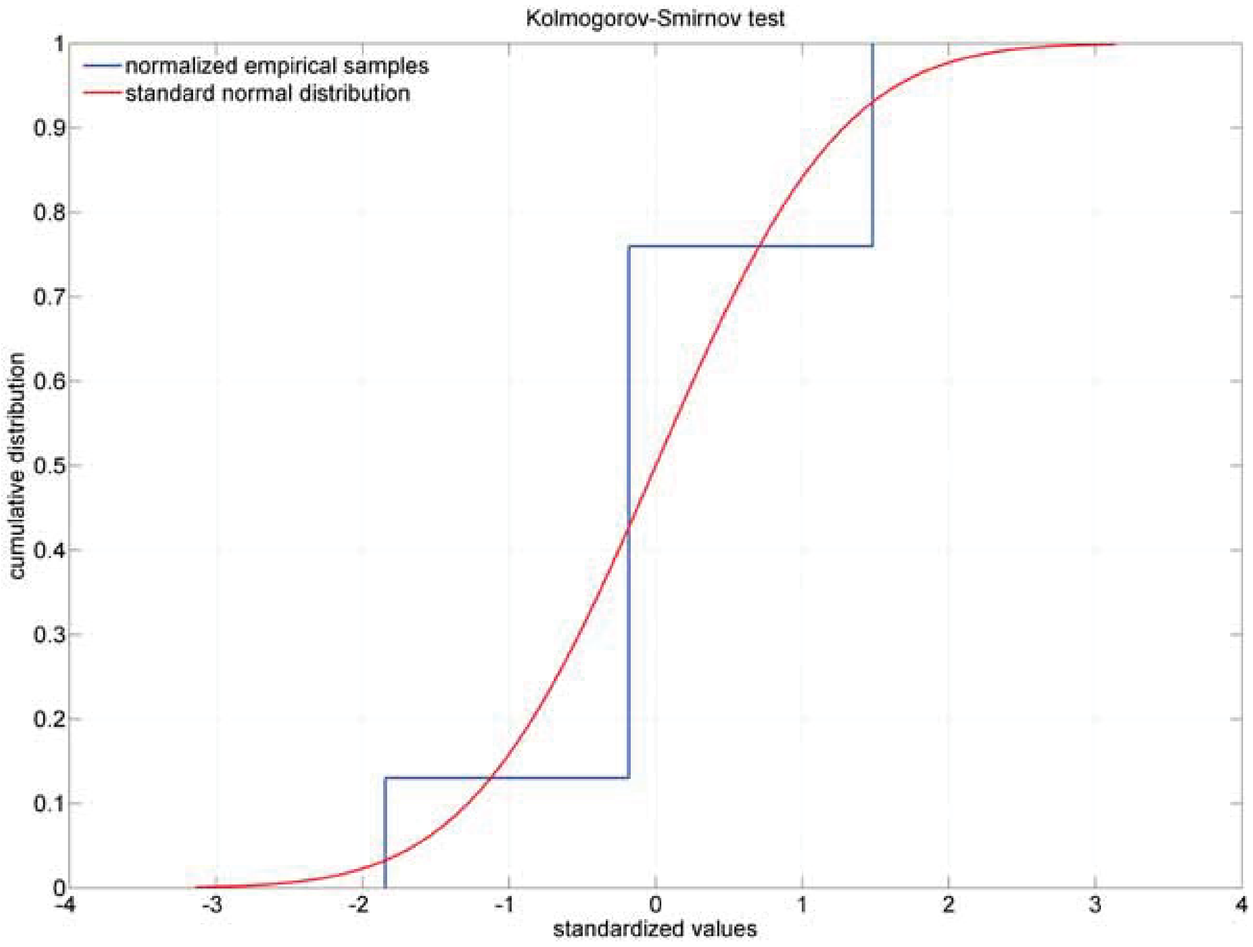


Figure C.51: K-S test of laser spot real coordinates in x dimension via tensorBFS.

Statistical analysis:		
$\hat{a}$	85.94	mm
$\tilde{a}$	85.94	mm
$\bar{a}$	85.9411	mm
s^2	0.000036152	mm^2
s	0.0060126	mm
e	0.00060126	mm
$e_{95\%}$	0.0011785	mm
$e_{99\%}$	0.0015212	mm
$\hat{\gamma}_1$	-0.04408	
$\hat{\beta}_2$	2.717	
K-S test	H_0 rejected	

Table C.17: Stat. analysis of laser spot real coordinates in x dimension via tensorBFS.

TC#2: Single Laser Spot / TensorBFS / y Dimension

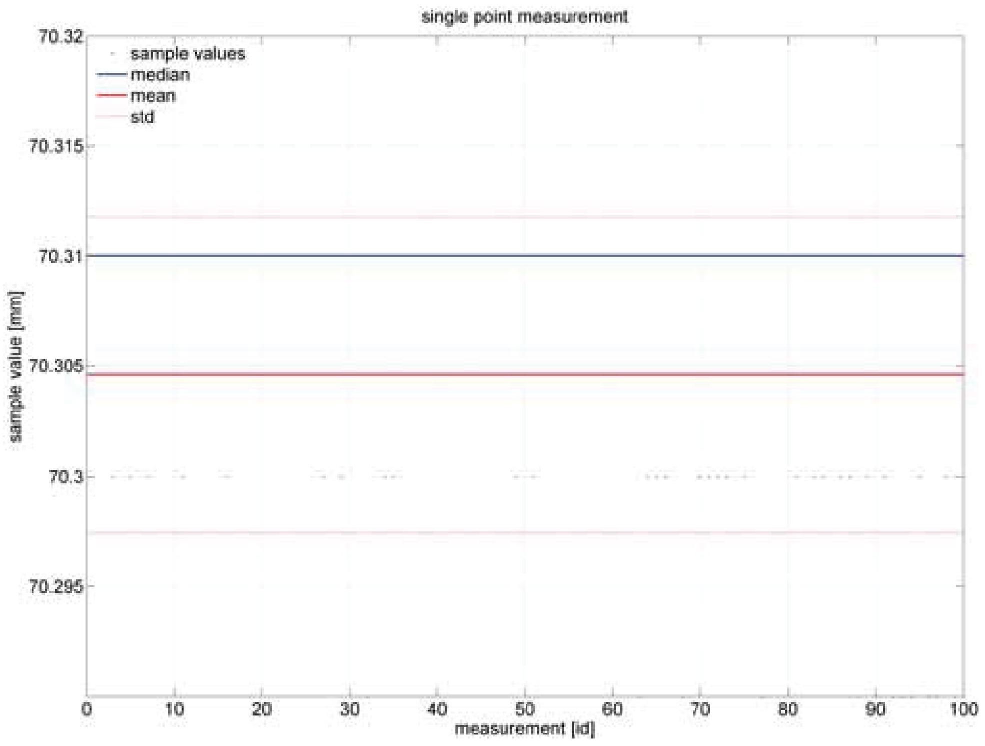


Figure C.52: Laser spot real coordinates in y dimension via tensorBFS.

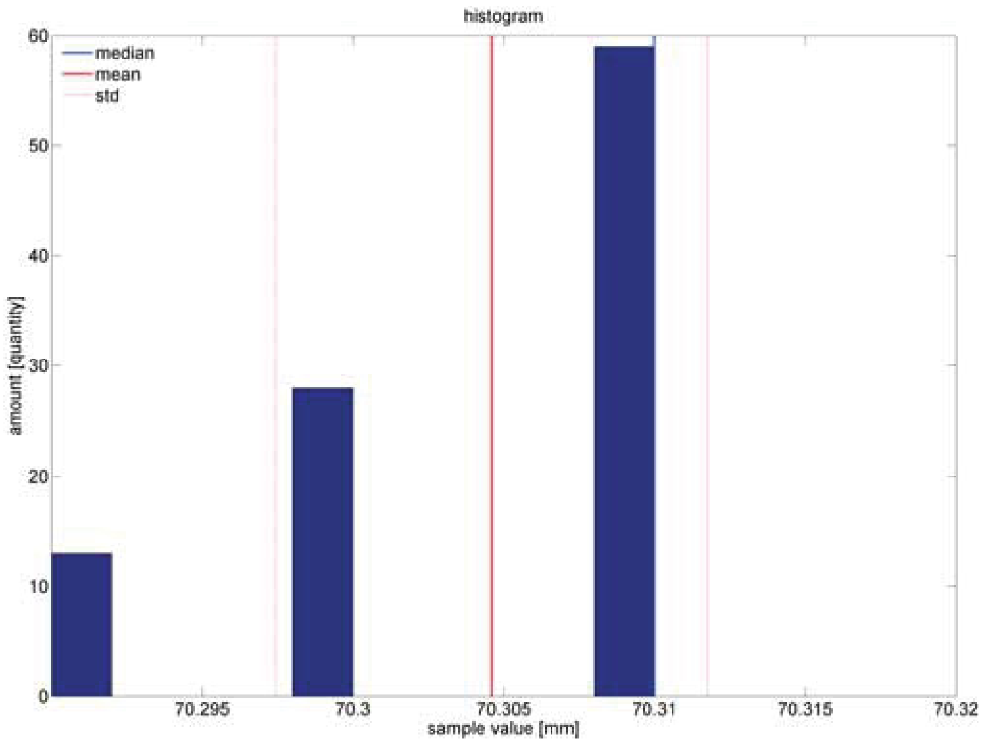


Figure C.53: Histogram of laser spot real coordinates in y dimension via tensorBFS.

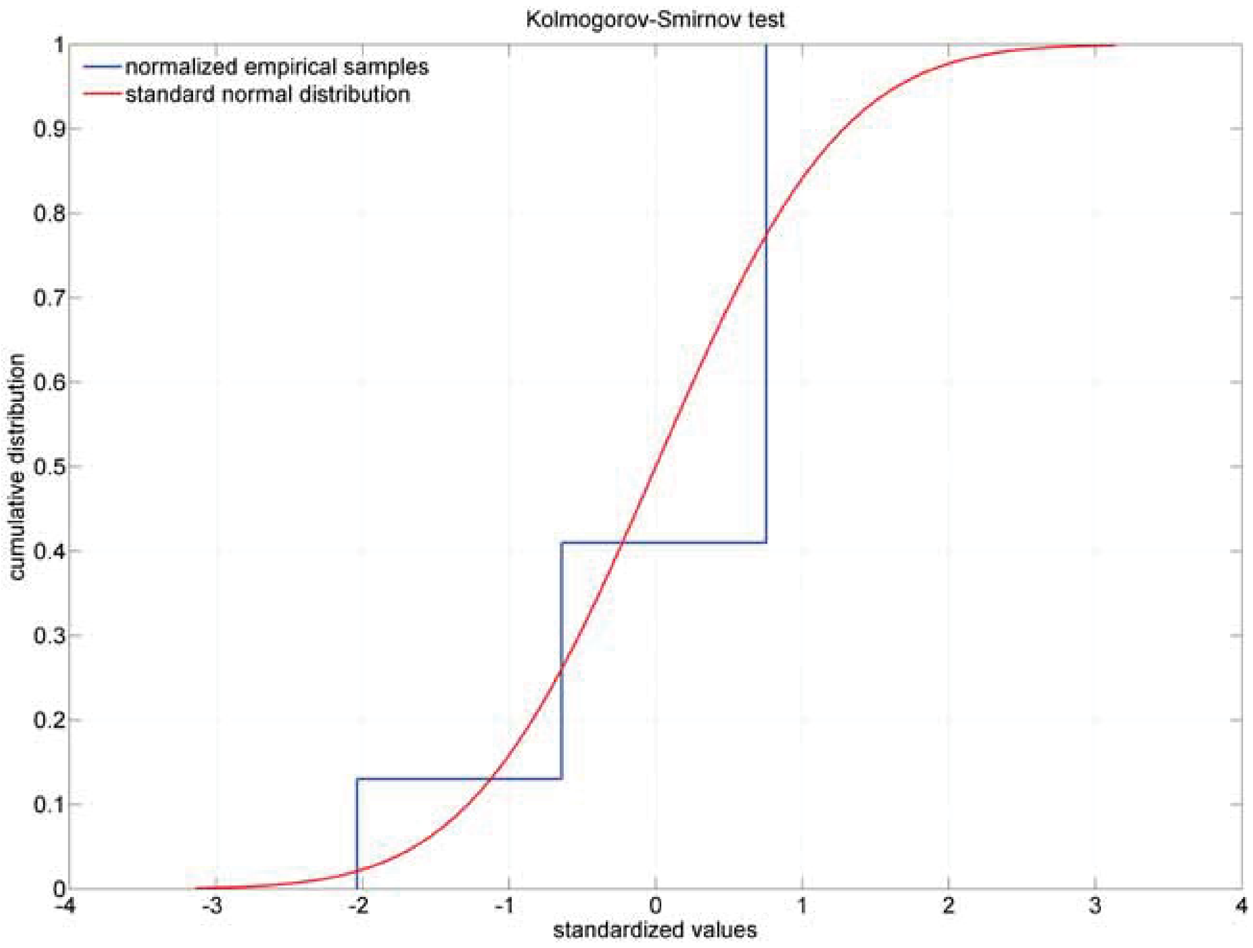


Figure C.54: K-S test of laser spot real coordinates in y dimension via tensorBFS.

Statistical analysis:		
$\hat{a}$	70.31	mm
$\tilde{a}$	70.31	mm
$\bar{a}$	70.3046	mm
s^2	0.000051354	mm^2
s	0.0071661	mm
e	0.00071661	mm
$e_{95\%}$	0.0014046	mm
$e_{99\%}$	0.001813	mm
$\hat{\gamma}_1$	-0.93497	
$\hat{\beta}_2$	2.5279	
K-S test	H_0 rejected	

Table C.18: Stat. analysis of laser spot real coordinates in y dimension via tensorBFS.

TC#2: Single Laser Spot / TensorQTD / x Dimension

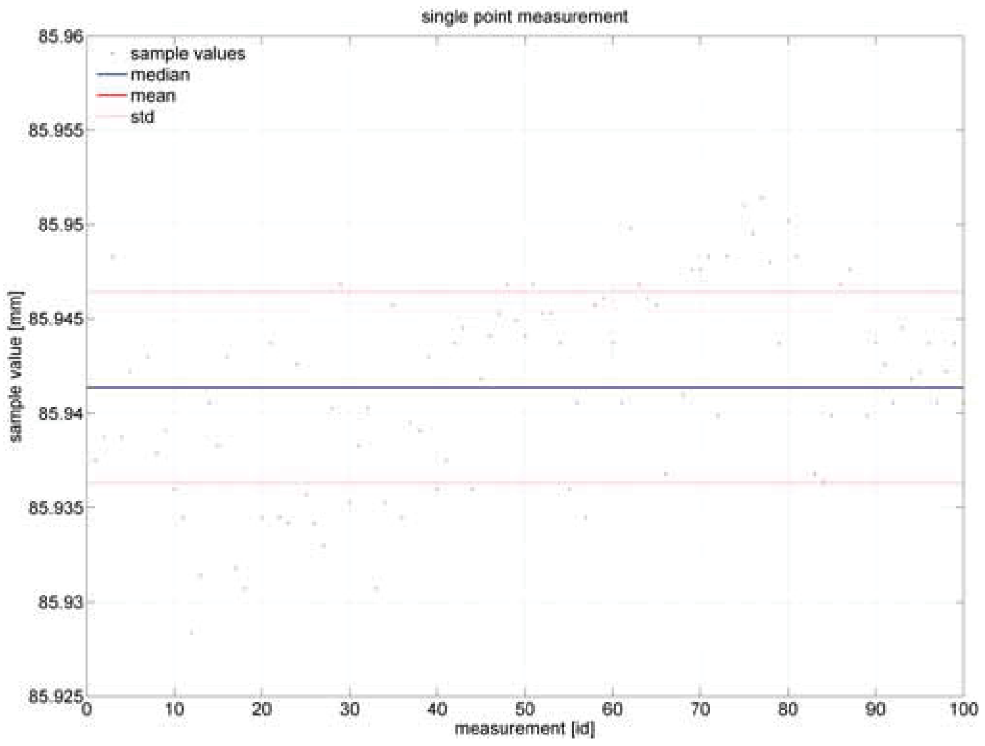


Figure C.55: Laser spot real coordinates in x dimension via tensorQTD.

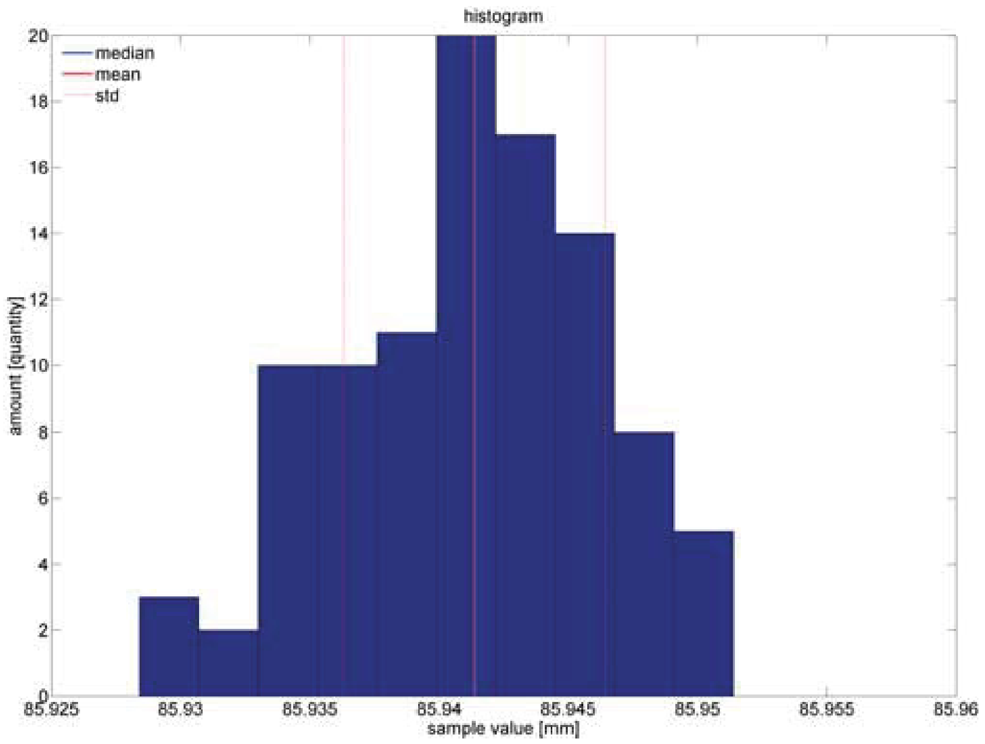


Figure C.56: Histogram of laser spot real coordinates in x dimension via tensorQTD.

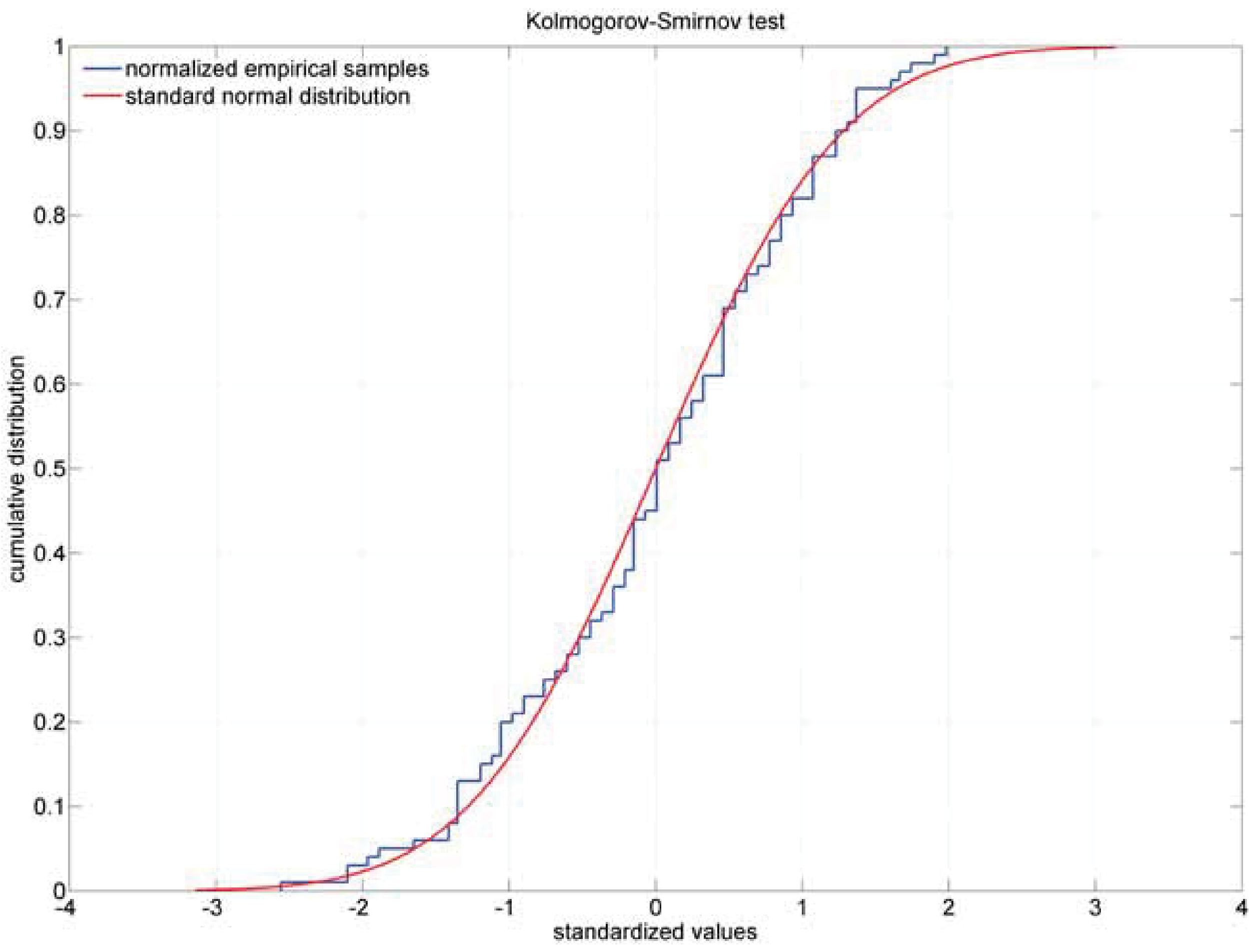


Figure C.57: K-S test of laser spot real coordinates in x dimension via tensorQTD.

Statistical analysis:		
$\hat{a}$	85.9437	mm
$\tilde{a}$	85.9414	mm
$\bar{a}$	85.9414	mm
s^2	0.0000257	mm^2
s	0.0050695	mm
e	0.00050695	mm
$e_{95\%}$	0.00099363	mm
$e_{99\%}$	0.0012826	mm
$\hat{\gamma}_1$	-0.26901	
$\hat{\beta}_2$	2.4866	
K-S test	H_0 accepted	

Table C.19: Stat. analysis of laser spot real coordinates in x dimension via tensorQTD.

TC#2: Single Laser Spot / TensorQTD / y Dimension

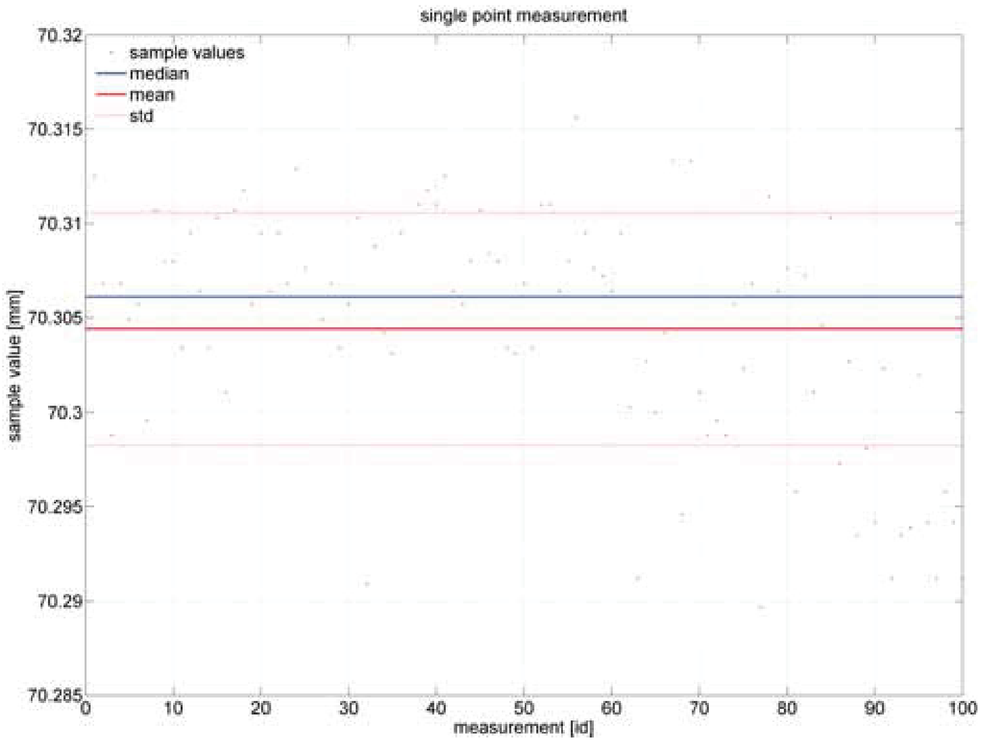


Figure C.58: Laser spot real coordinates in y dimension via tensorQTD.

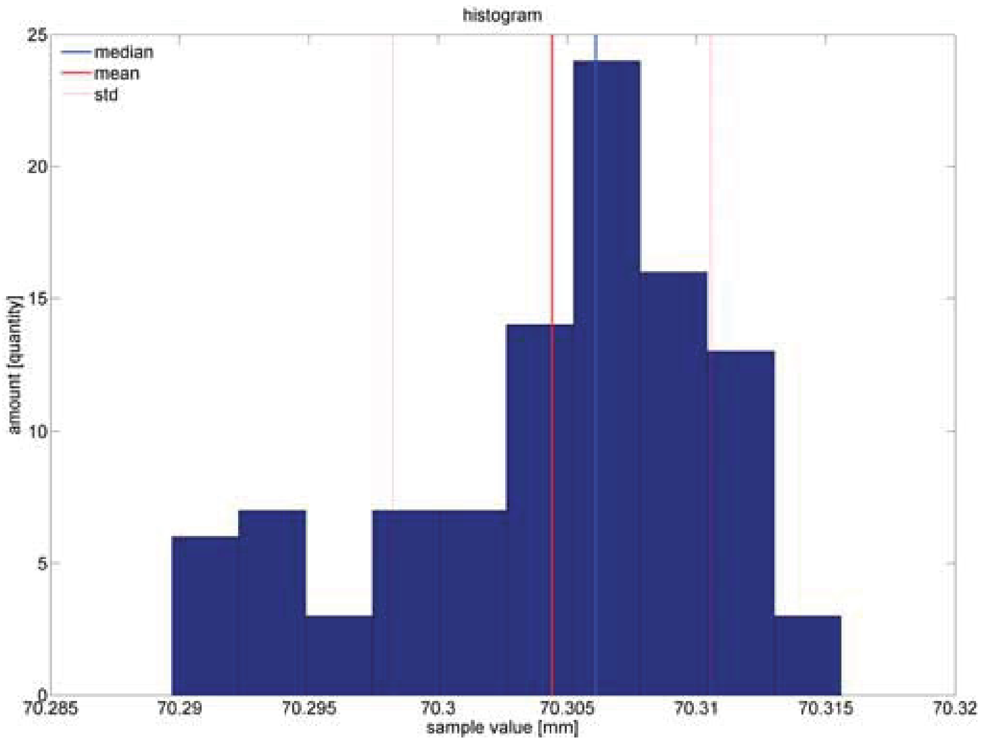


Figure C.59: Histogram of laser spot real coordinates in y dimension via tensorQTD.

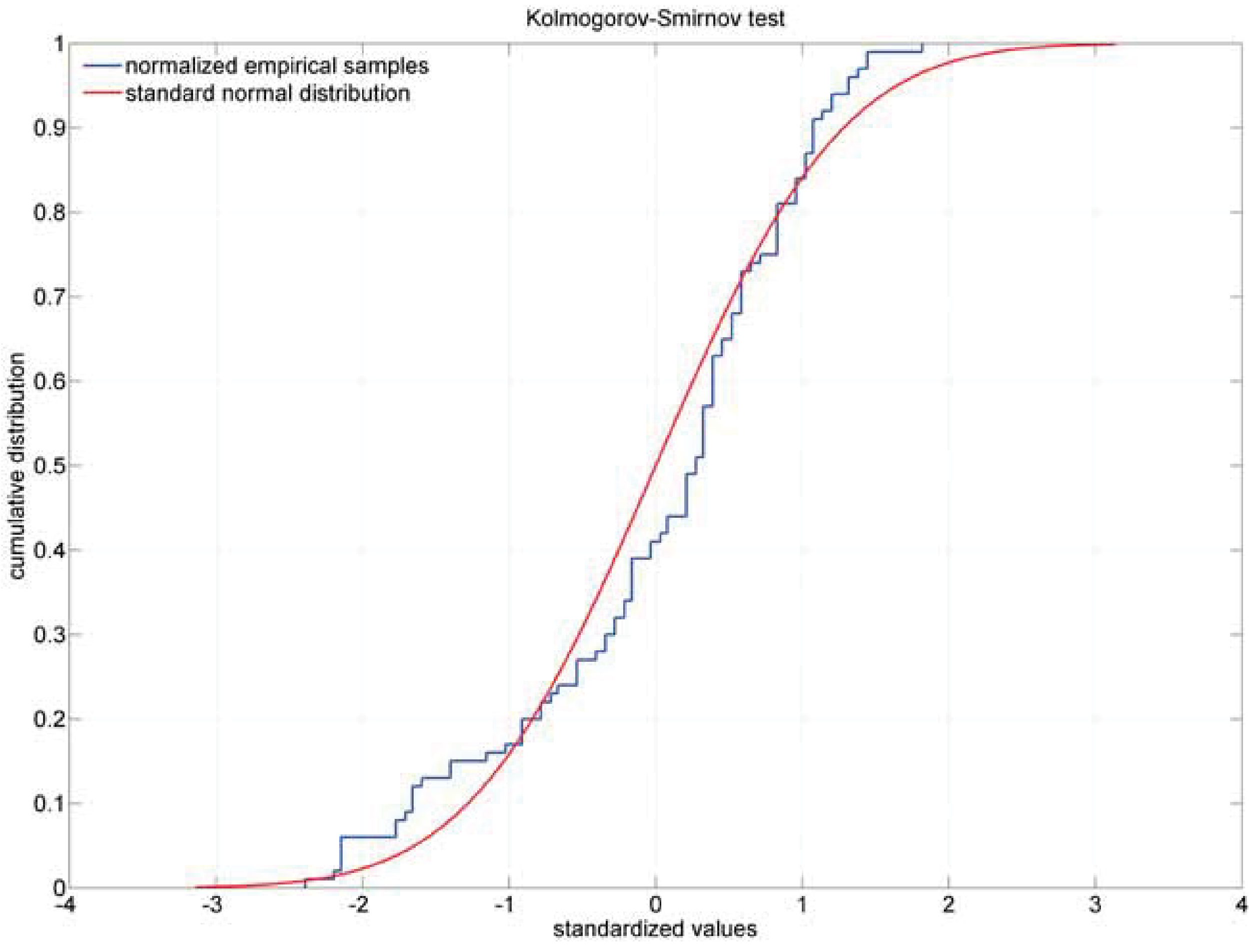


Figure C.60: K-S test of laser spot real coordinates in y dimension via tensorQTD.

Statistical analysis:		
$\hat{a}$	70.3064	mm
$\tilde{a}$	70.3061	mm
$\bar{a}$	70.3044	mm
s^2	0.00003785	mm^2
s	0.0061523	mm
e	0.00061523	mm
$e_{95\%}$	0.0012058	mm
$e_{99\%}$	0.0015565	mm
$\hat{\gamma}_1$	-0.70323	
$\hat{\beta}_2$	2.685	
K-S test	H_0 accepted	

Table C.20: Stat. analysis of laser spot real coordinates in y dimension via tensorQTD.

Appendix D

Base Points Remapping

The measurements are carried out for two different types of lenses:

1. TC#1 (Test-Case 1): a *normal-angle objective* with focal length of $f = 12mm$;
2. TC#2 (Test-Case 2): a *wide-angle objective* with focal length of $f = 3.6mm$.

The combined error e_{xy} and its mean $\bar{e}_{xy}$ are acquired by

$$e_{xy_i} = \sqrt{{e_x}_i^2 + {e_y}_i^2}, \qquad \text{and} \qquad \bar{e}_{xy} = \frac{1}{\varepsilon}\sum_{i=1}^{\varepsilon} e_{xy_i}, \tag{D.1}$$

when the number of investigated image objects $\varepsilon = 35$ LEDs. The object distance g is adjusted depending on the utilized objective, i.e. $g(f = 12mm) = 440mm$ and $g(f = 3.6mm) = 140mm$. The deployed degree d and the used mapping function is stated for each figure.

TC#1: $f = 12mm$ / Center of Gravity / Homography

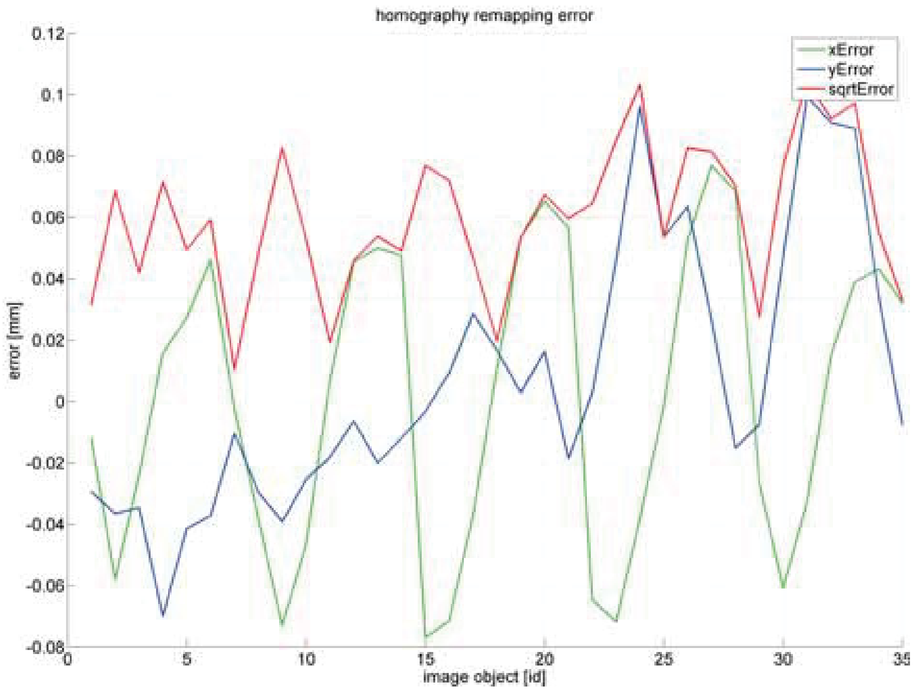


Figure D.1: Homography error utilizing $f = 12mm$ and center of gravity.

TC#1: $f = 12mm$ / Center of Gravity / PolyCtrlPts

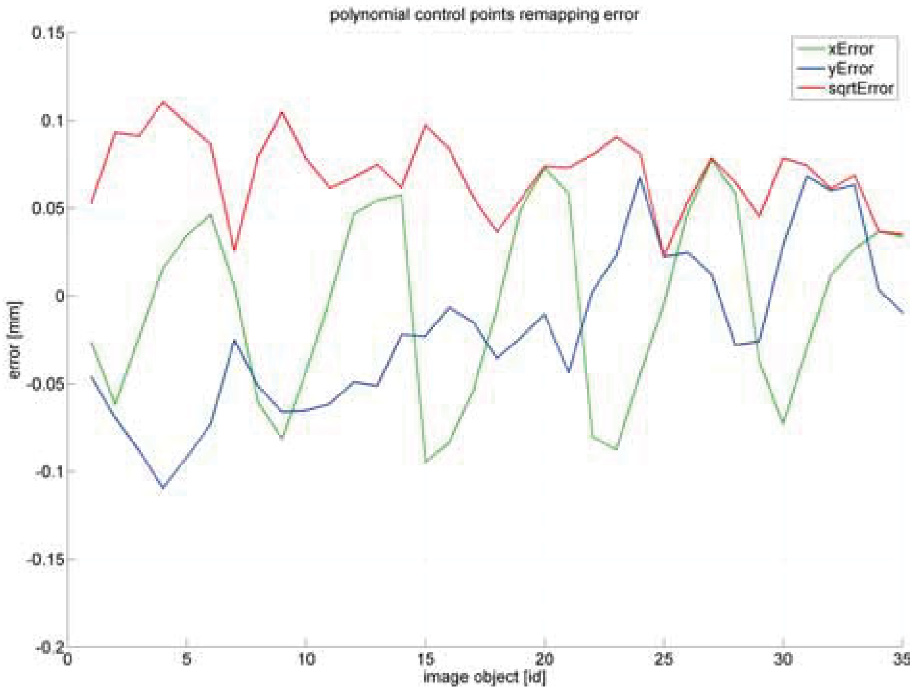


Figure D.2: PolyCtrlPts error utilizing $d = 2$, $f = 12mm$ and center of gravity.

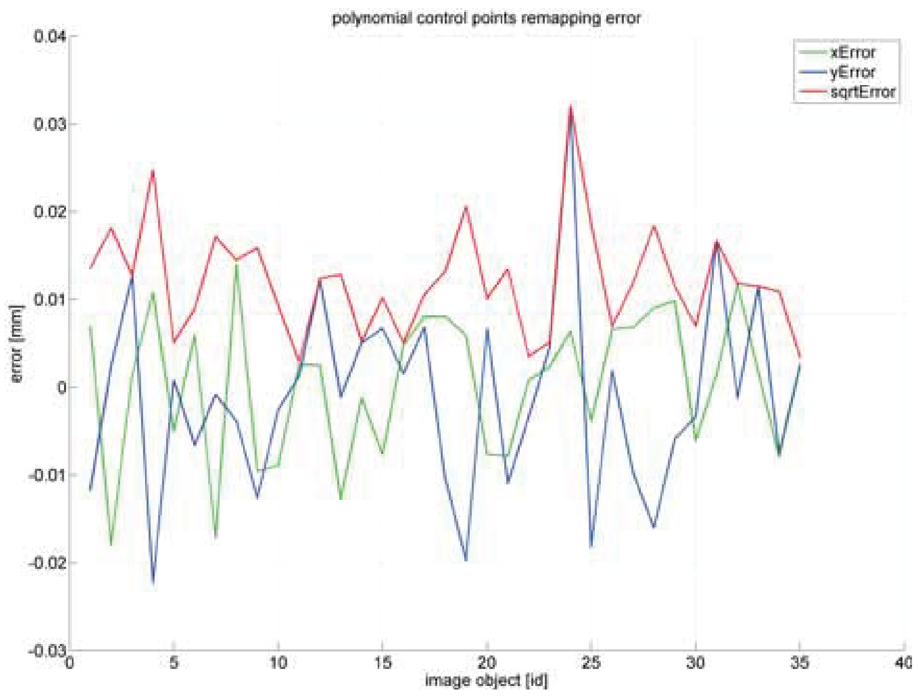


Figure D.3: PolyCtrlPts error utilizing $d = 4$, $f = 12mm$ and center of gravity.

TC#1: $f = 12mm$ / Center of Gravity / TensorBFS

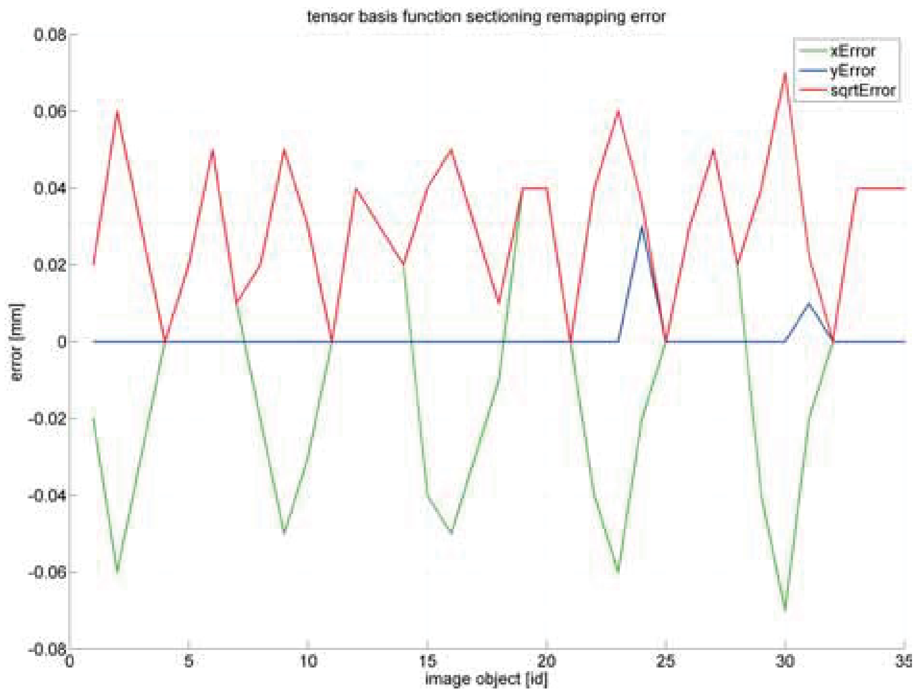


Figure D.4: TensorBFS error utilizing $d = 2$, $f = 12mm$ and center of gravity.

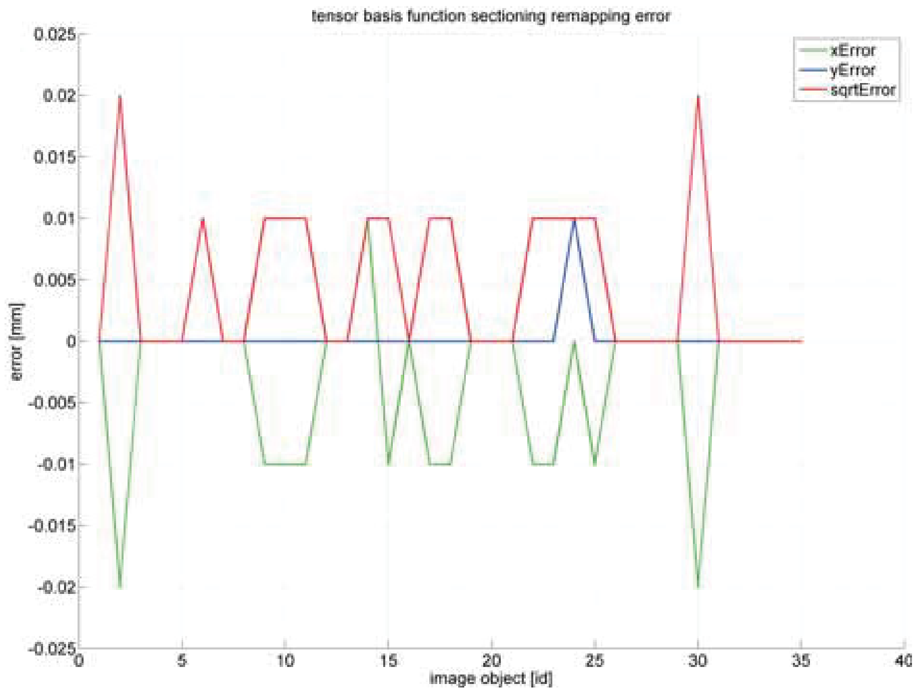


Figure D.5: TensorBFS error utilizing $d = 4$, $f = 12mm$ and center of gravity.

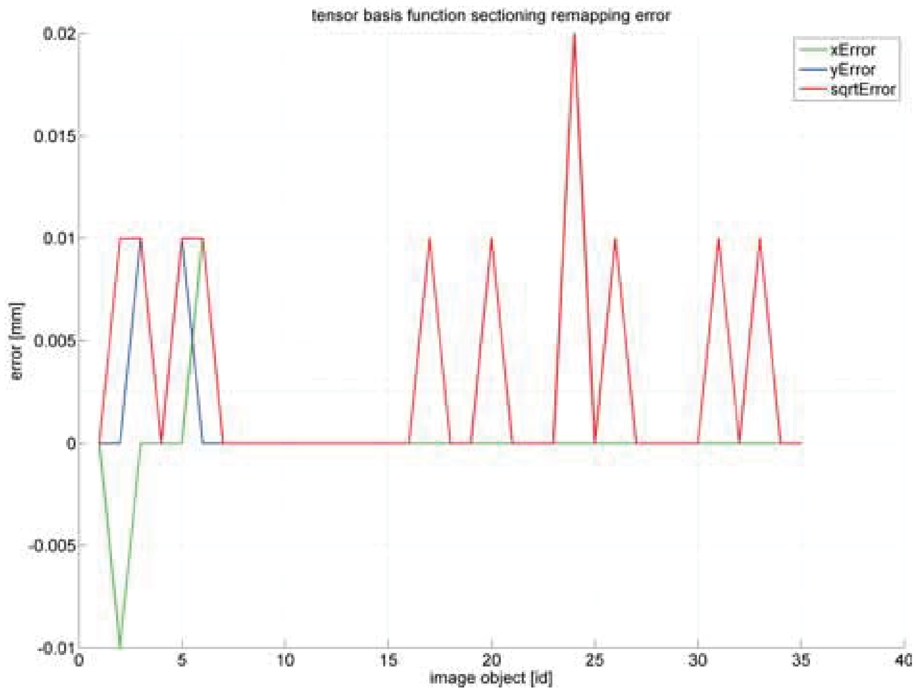


Figure D.6: TensorBFS error utilizing $d = 6$, $f = 12mm$ and center of gravity.

TC#1: $f = 12mm$ / Center of Gravity / TensorQDT

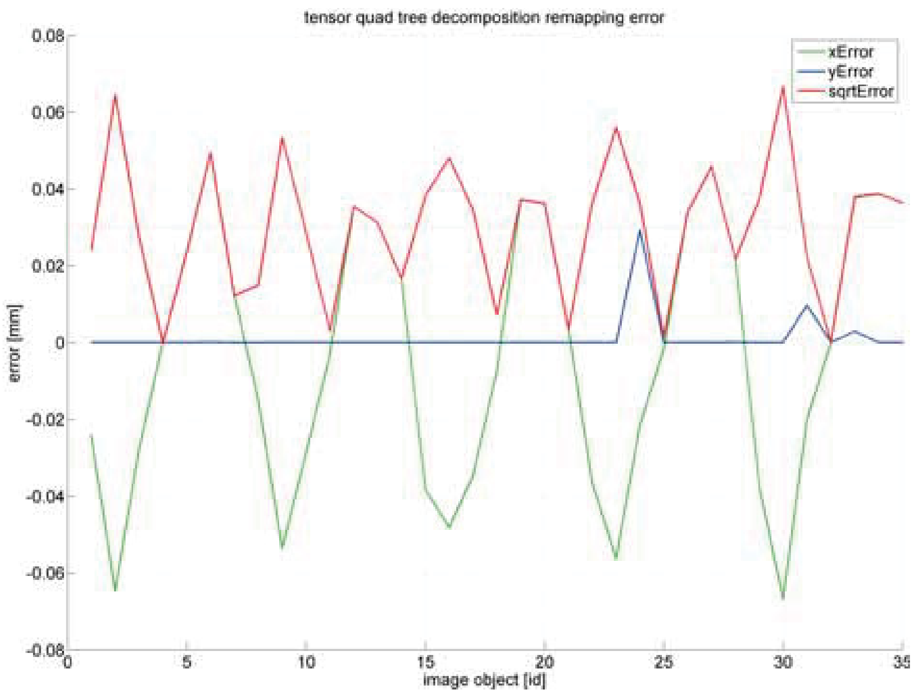


Figure D.7: TensorQDT error utilizing $d = 2$, $f = 12mm$ and center of gravity.

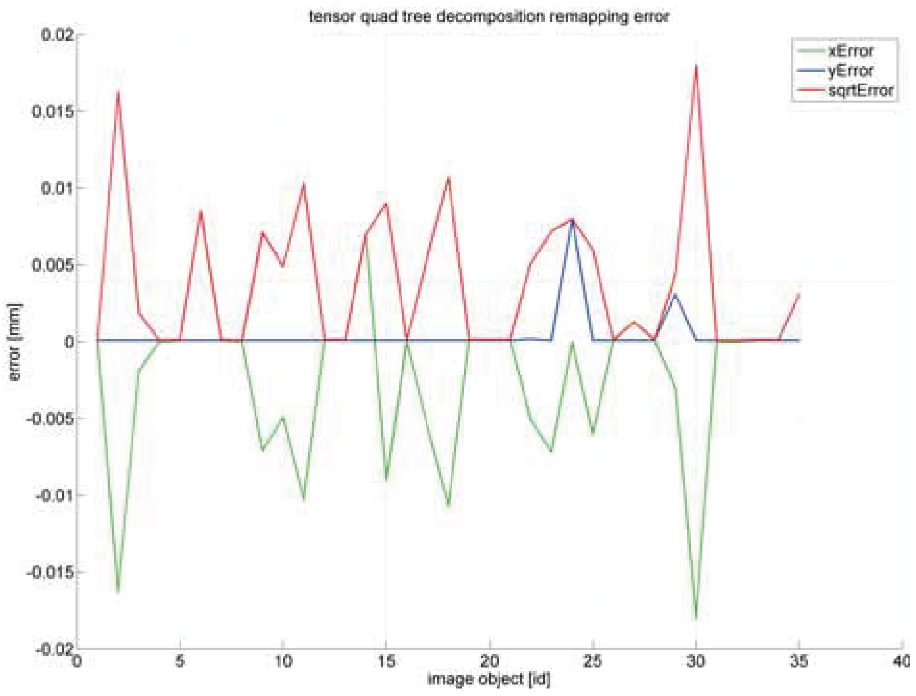


Figure D.8: TensorQDT error utilizing $d = 4$, $f = 12mm$ and center of gravity.

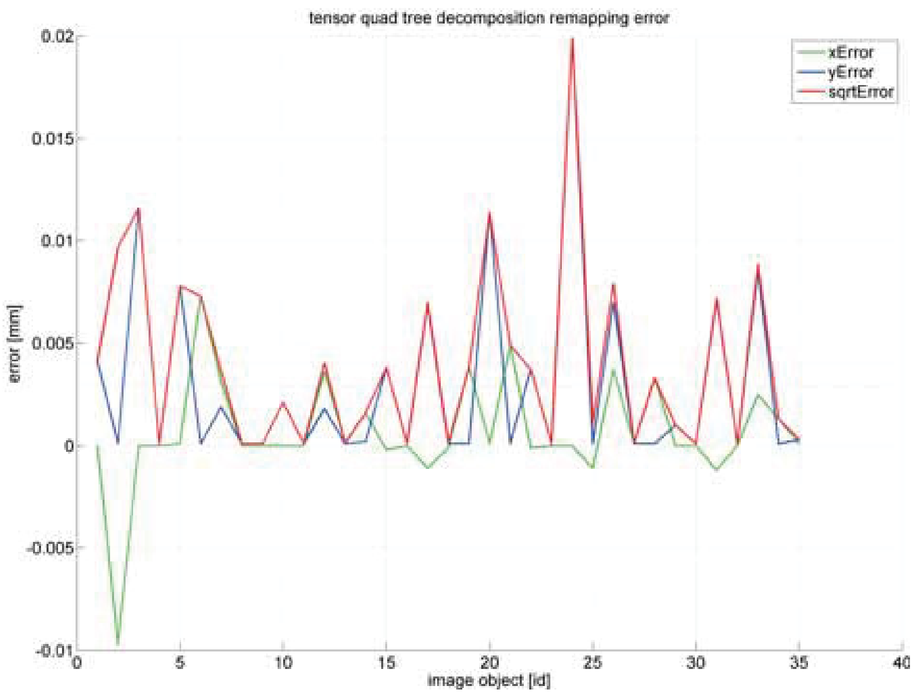


Figure D.9: TensorQDT error utilizing $d = 6$, $f = 12mm$ and center of gravity.

TC#1: $f = 12mm$ / Gaussian Curve Extremum / Homography

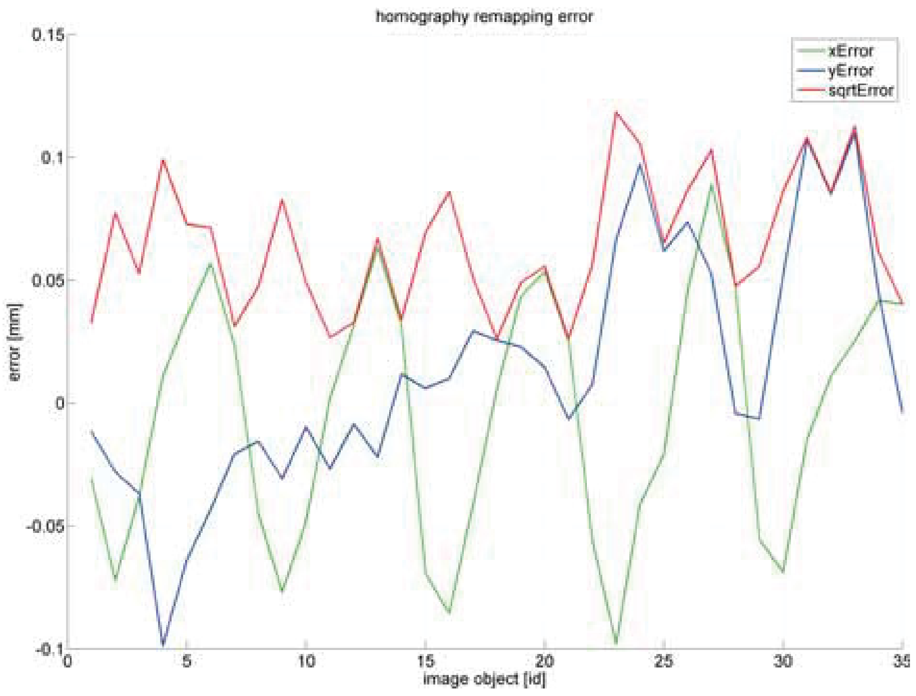


Figure D.10: Homography error utilizing $f = 12mm$ and Gaussian curve extremum.

TC#1: $f = 12mm$ / Gaussian Curve Extremum / PolyCtrlPts

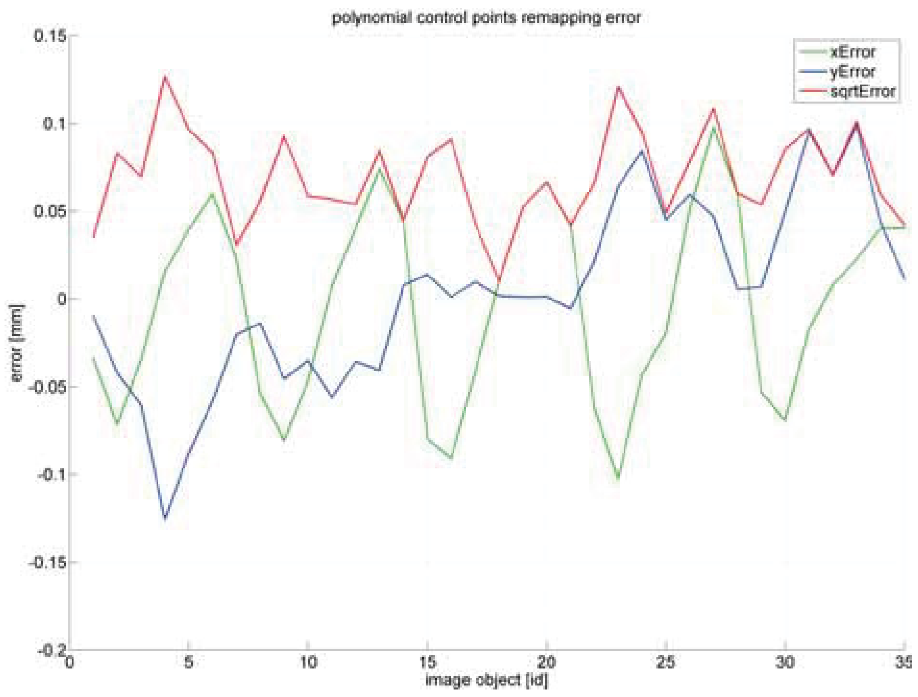


Figure D.11: PolyCtrlPts error utilizing $d = 2$, $f = 12mm$ and Gaussian curve extremum.

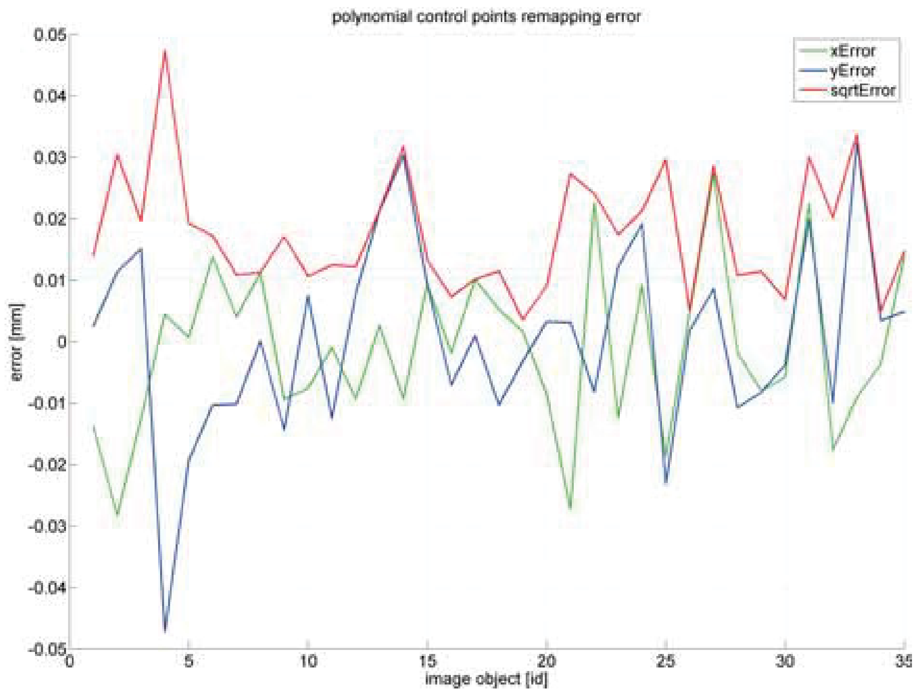


Figure D.12: PolyCtrlPts error utilizing $d = 4$, $f = 12mm$ and Gaussian curve extremum.

TC#1: $f = 12mm$ / Gaussian Curve Extremum / TensorBFS

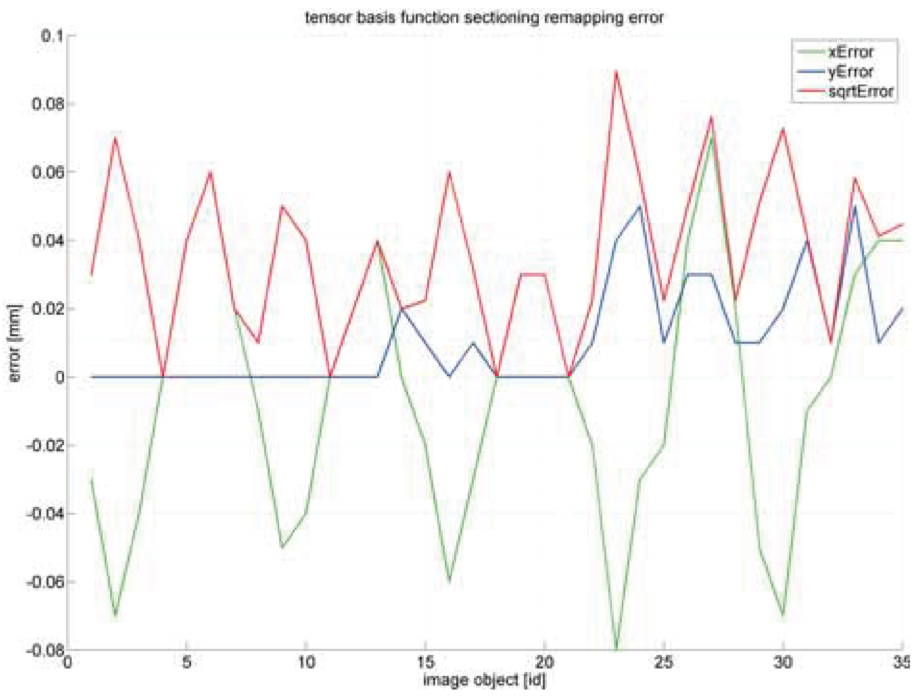


Figure D.13: TensorBFS error utilizing $d = 2$, $f = 12mm$ and Gaussian curve extremum.

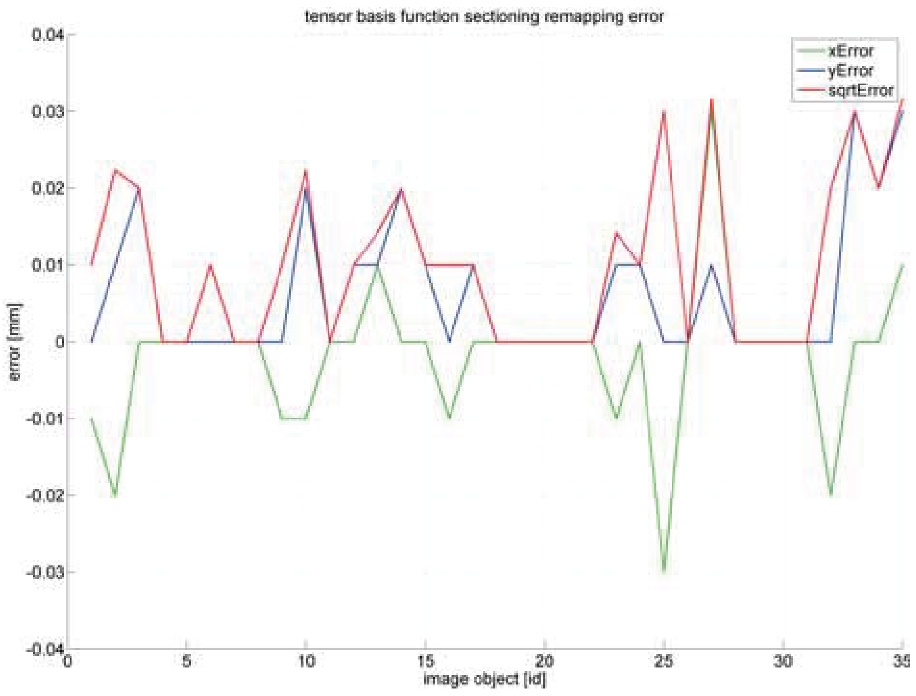


Figure D.14: TensorBFS error utilizing $d = 4$, $f = 12mm$ and Gaussian curve extremum.

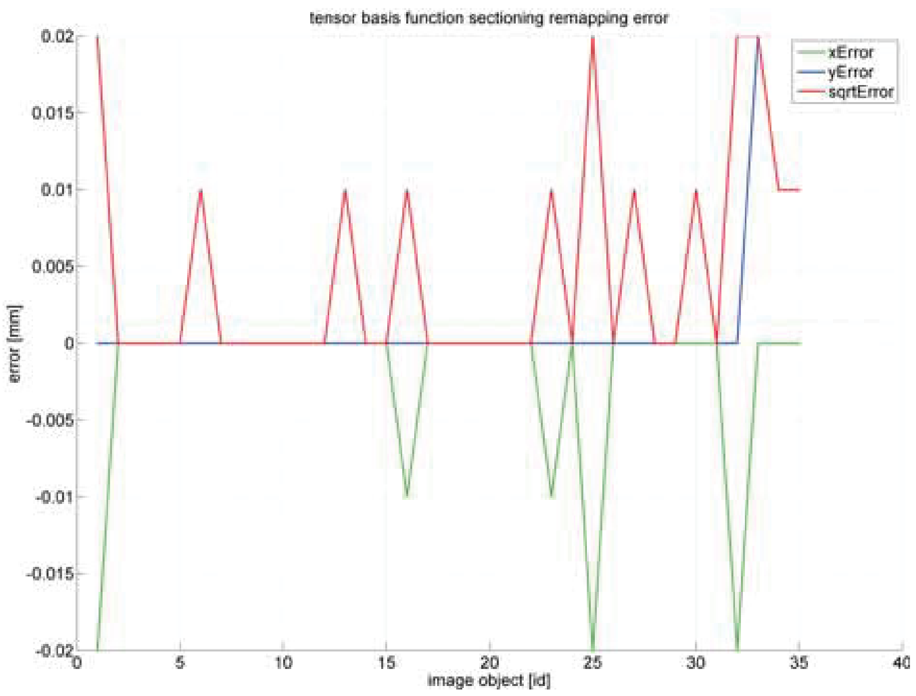


Figure D.15: TensorBFS error utilizing $d = 6$, $f = 12mm$ and Gaussian curve extremum.

TC#1: $f = 12mm$ / Gaussian Curve Extremum / TensorQDT

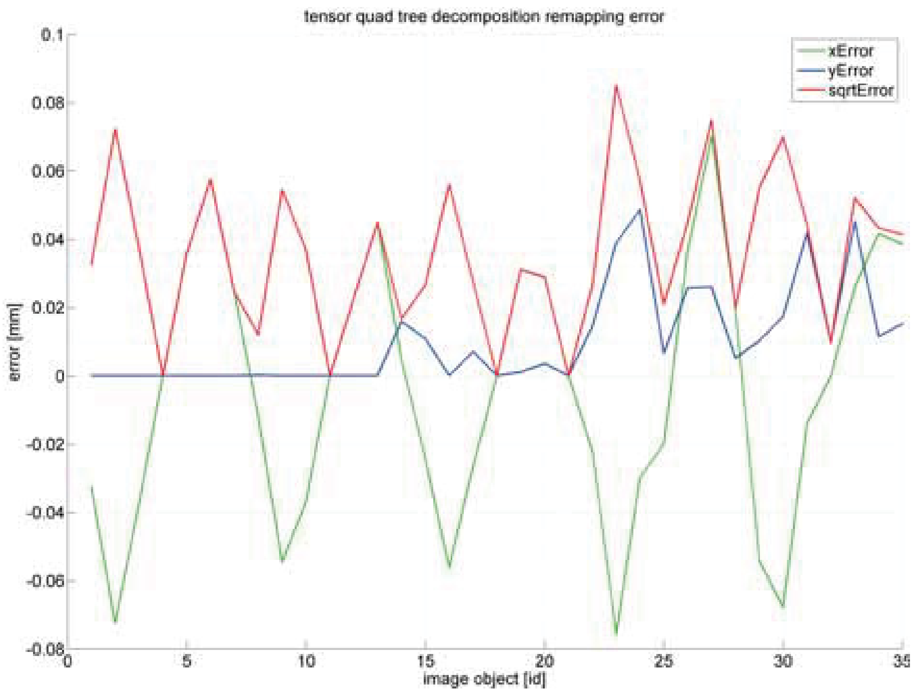


Figure D.16: TensorQDT error utilizing $d = 2$, $f = 12mm$ and Gaussian curve extremum.

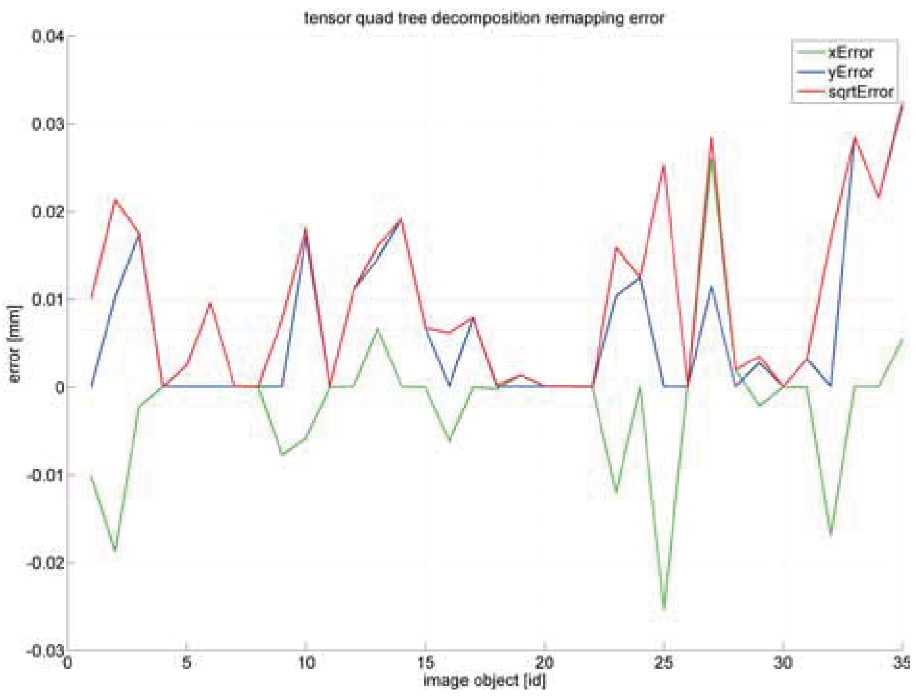


Figure D.17: TensorQDT error utilizing $d = 4$, $f = 12mm$ and Gaussian curve extremum.

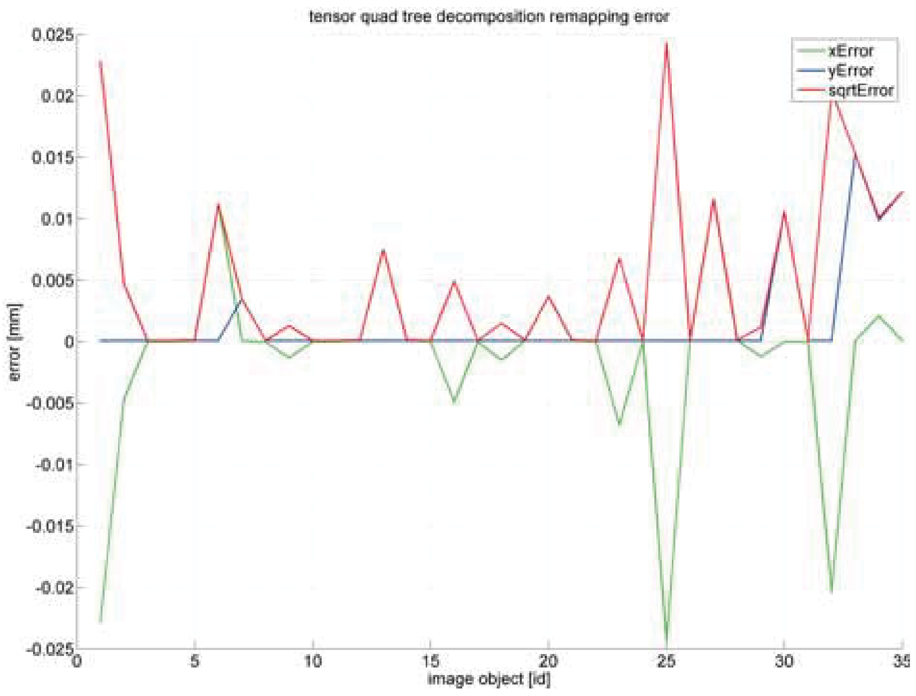


Figure D.18: TensorQDT error utilizing $d = 6$, $f = 12mm$ and Gaussian curve extremum.

TC#1: $f = 12mm$ / Ellipse Approximation / Homography

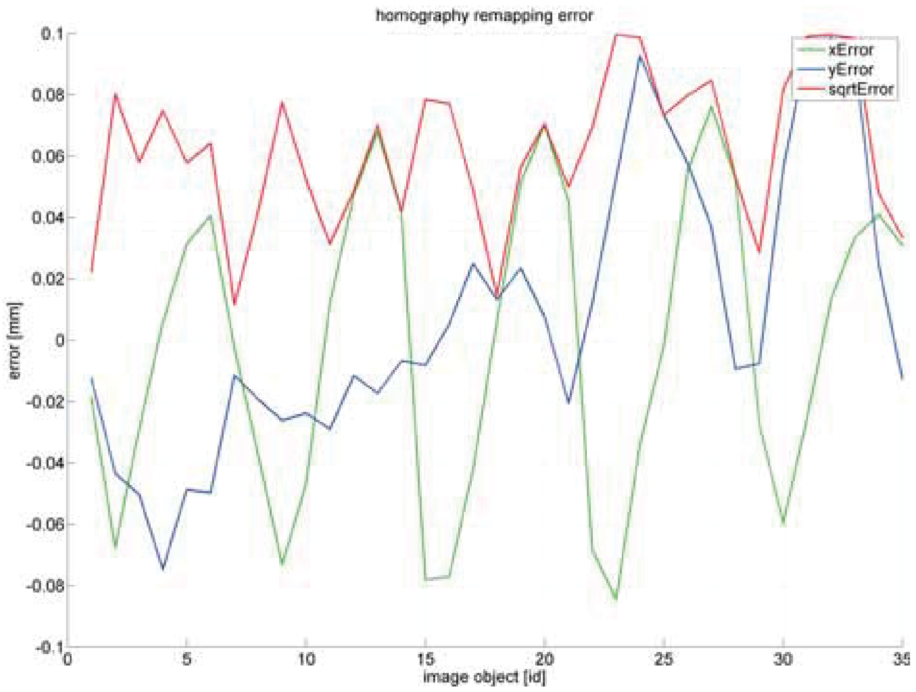


Figure D.19: Homography error utilizing $f = 12mm$ and ellipse approximation.

TC#1: $f = 12mm$ / Ellipse Approximation / PolyCtrlPts

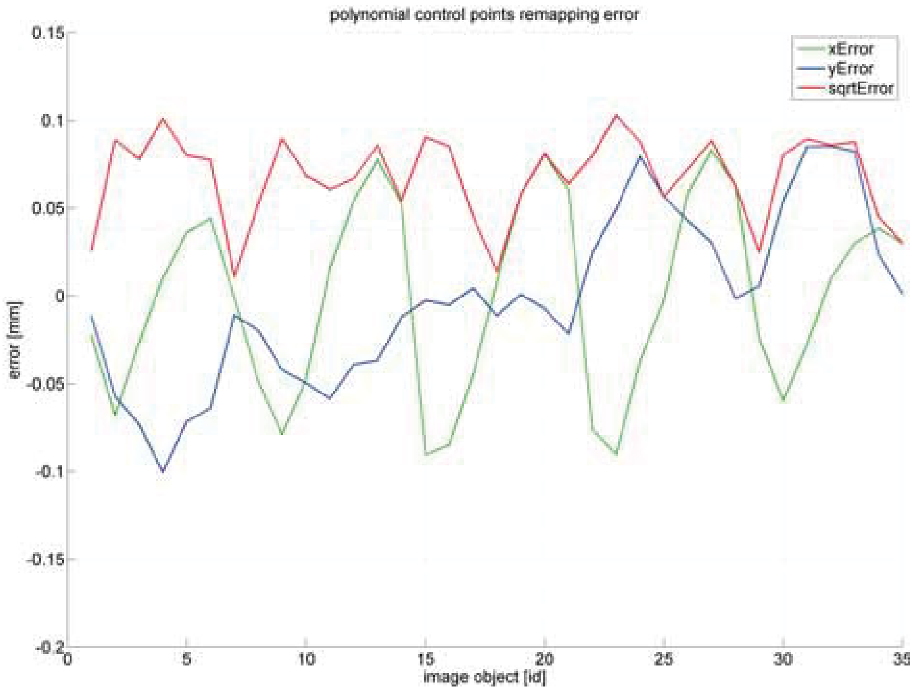


Figure D.20: PolyCtrlPts error utilizing $d = 2$, $f = 12mm$ and ellipse approximation.

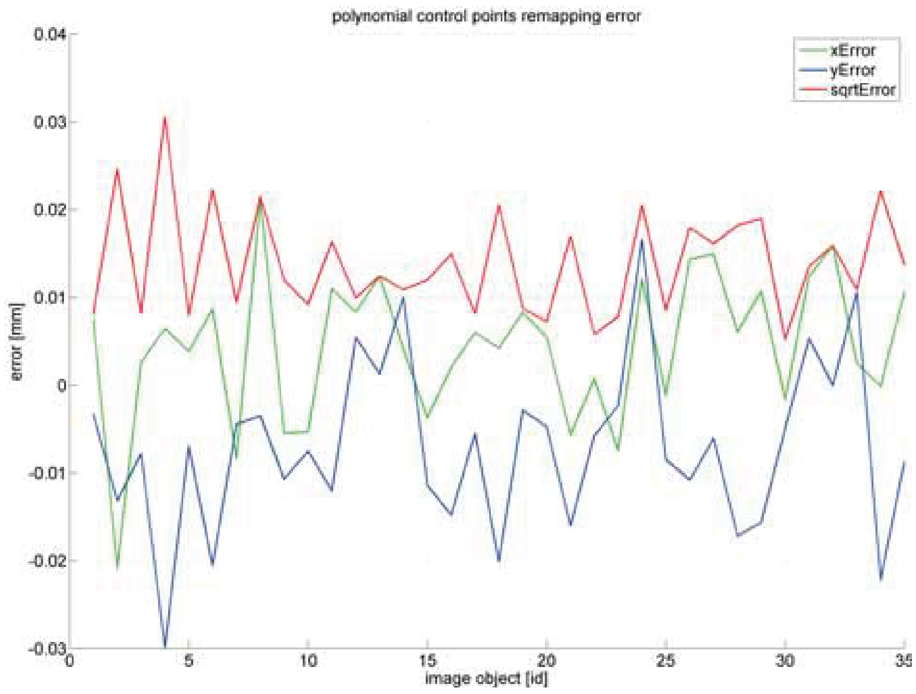


Figure D.21: PolyCtrlPts error utilizing $d = 4$, $f = 12mm$ and ellipse approximation.

TC#1: $f = 12mm$ / Ellipse Approximation / TensorBFS

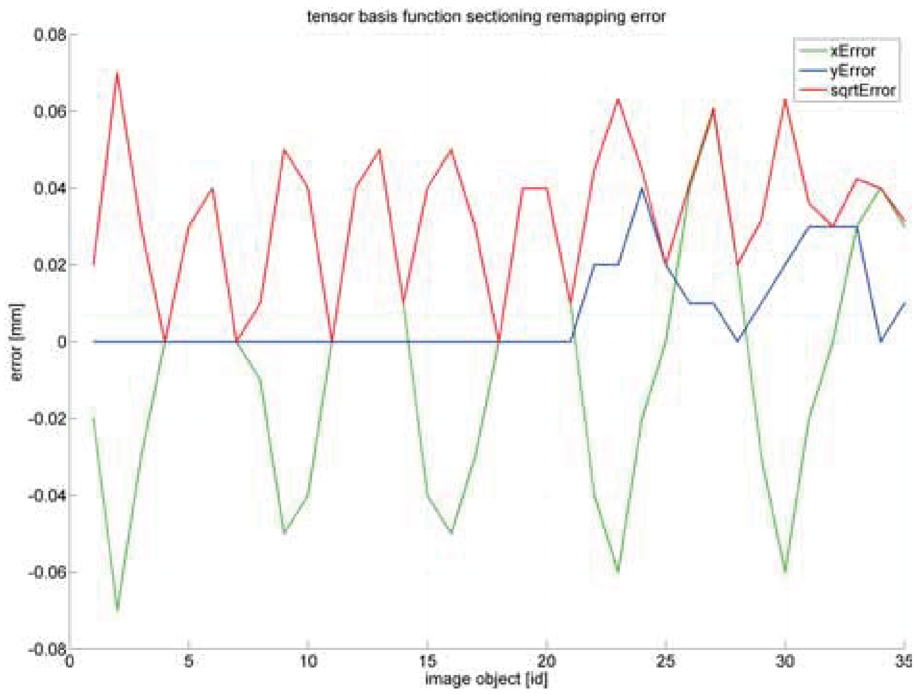


Figure D.22: TensorBFS error utilizing $d = 2$, $f = 12mm$ and ellipse approximation.

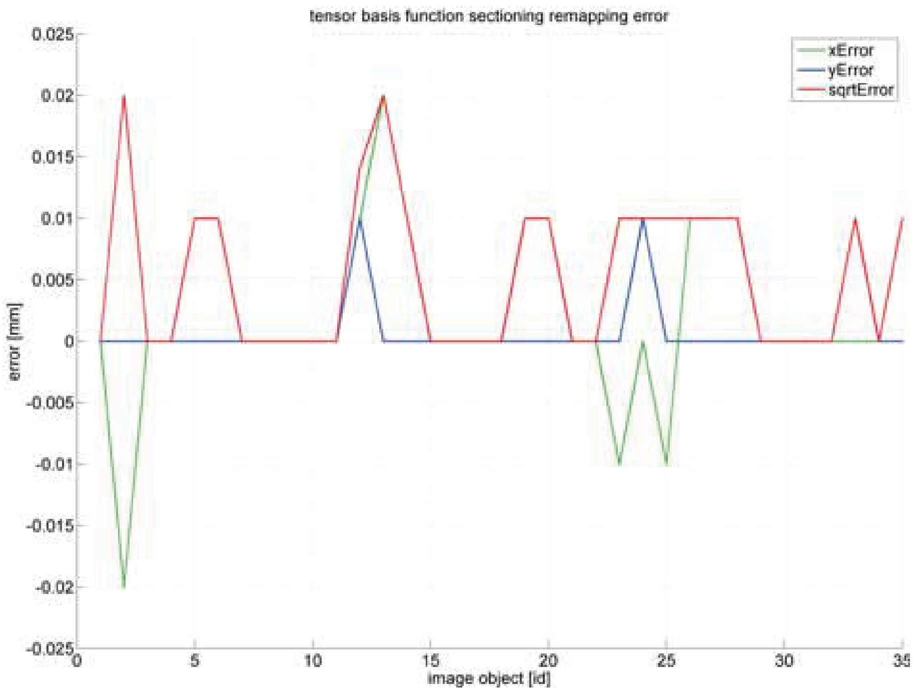


Figure D.23: TensorBFS error utilizing $d = 4$, $f = 12mm$ and ellipse approximation.

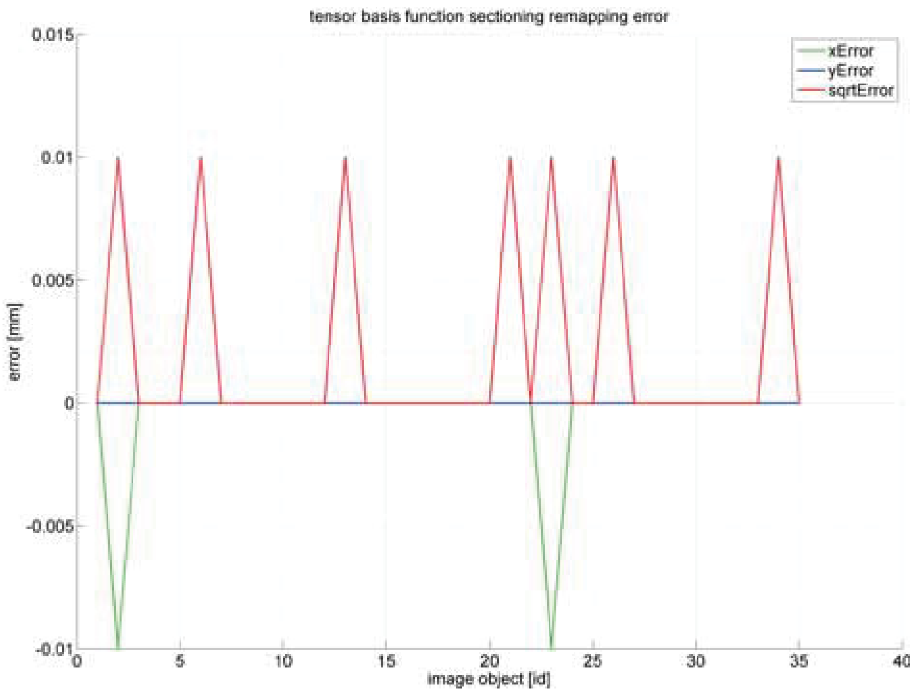


Figure D.24: TensorBFS error utilizing $d = 6$, $f = 12mm$ and ellipse approximation.

TC#1: $f = 12mm$ / Ellipse Approximation / TensorQDT

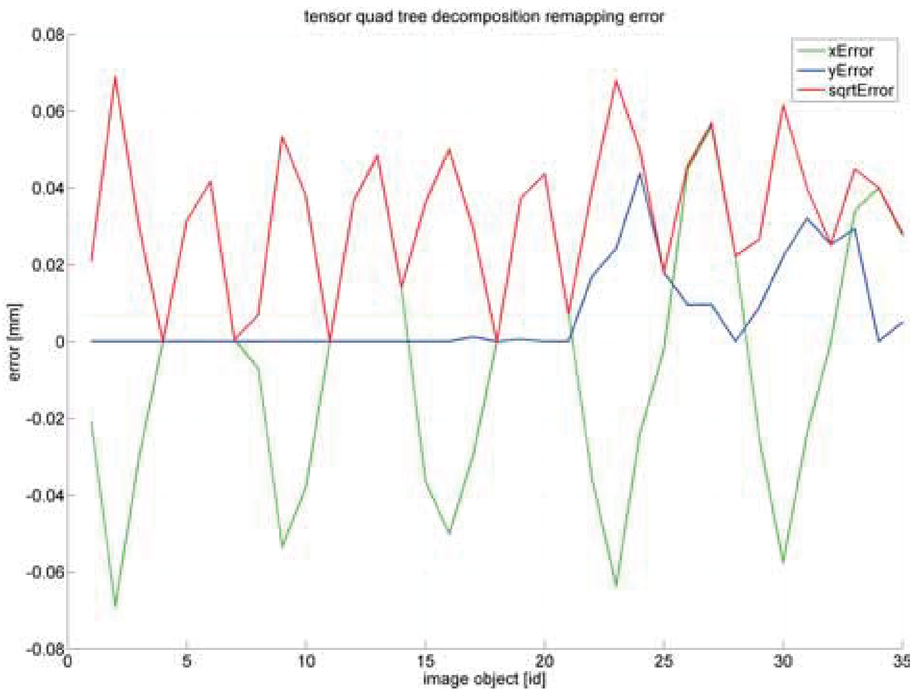


Figure D.25: TensorQDT error utilizing $d = 2$, $f = 12mm$ and ellipse approximation.

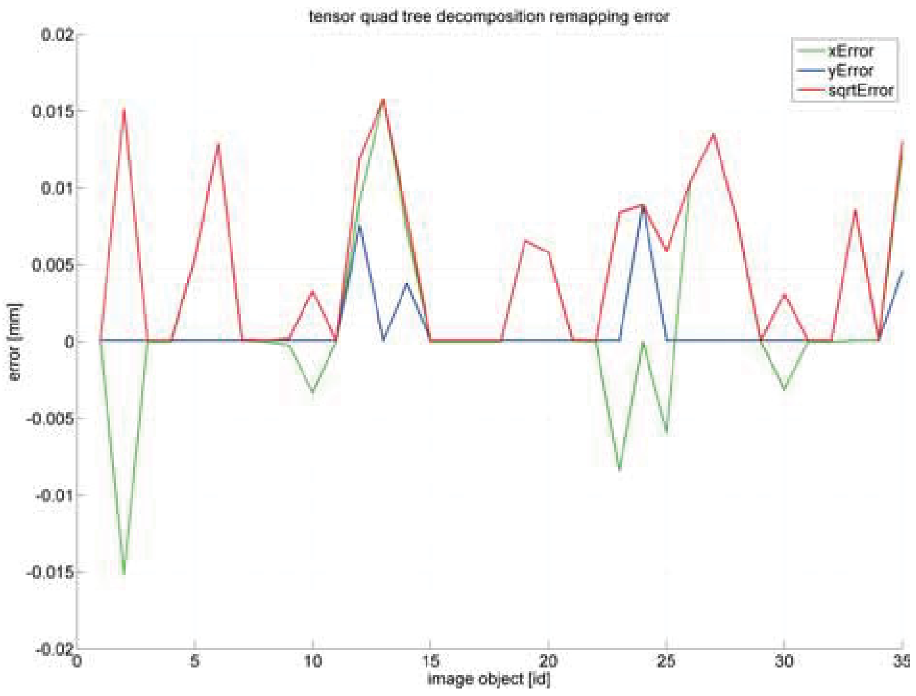


Figure D.26: TensorQDT error utilizing $d = 4$, $f = 12mm$ and ellipse approximation.

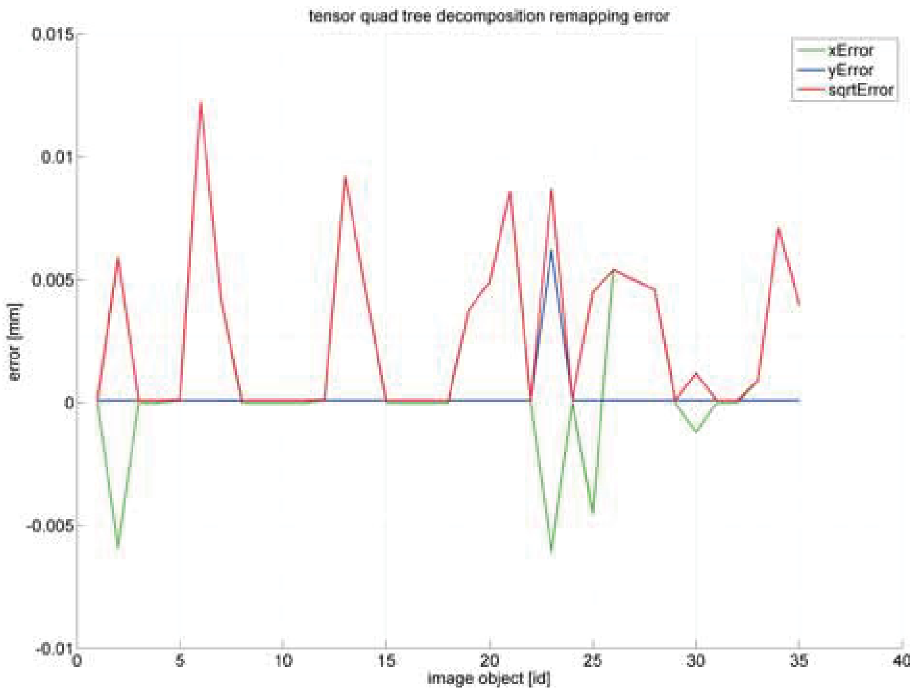


Figure D.27: TensorQDT error utilizing $d = 6$, $f = 12mm$ and ellipse approximation.

TC#1: $f = 12mm$ / Circle Approximation / Homography

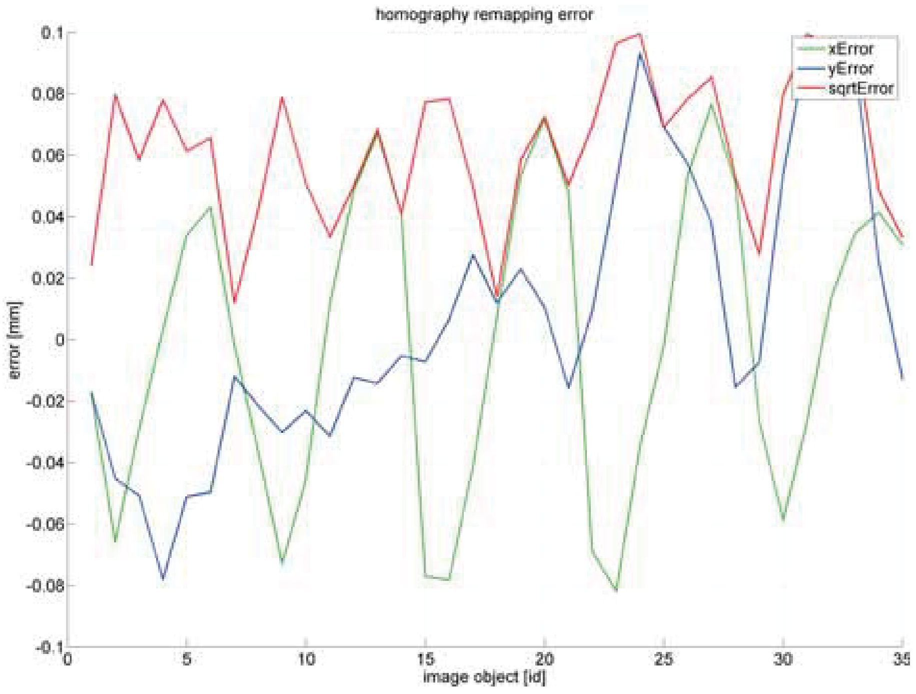


Figure D.28: Homography error utilizing $f = 12mm$ and circle approximation.

TC#1: $f = 12mm$ / Circle Approximation / PolyCtrlPts

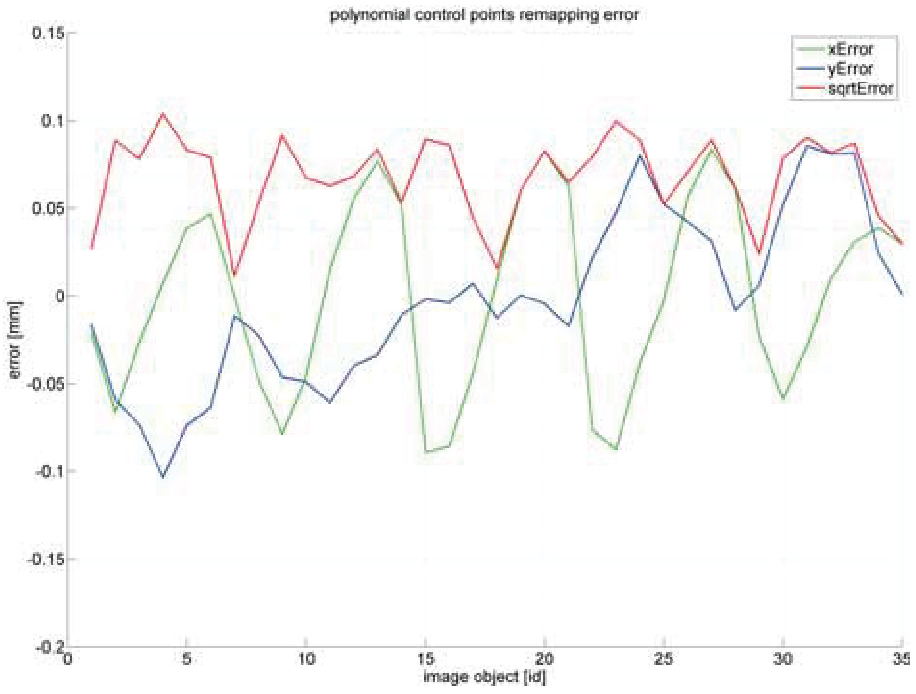


Figure D.29: PolyCtrlPts error utilizing $d = 2$, $f = 12mm$ and circle approximation.

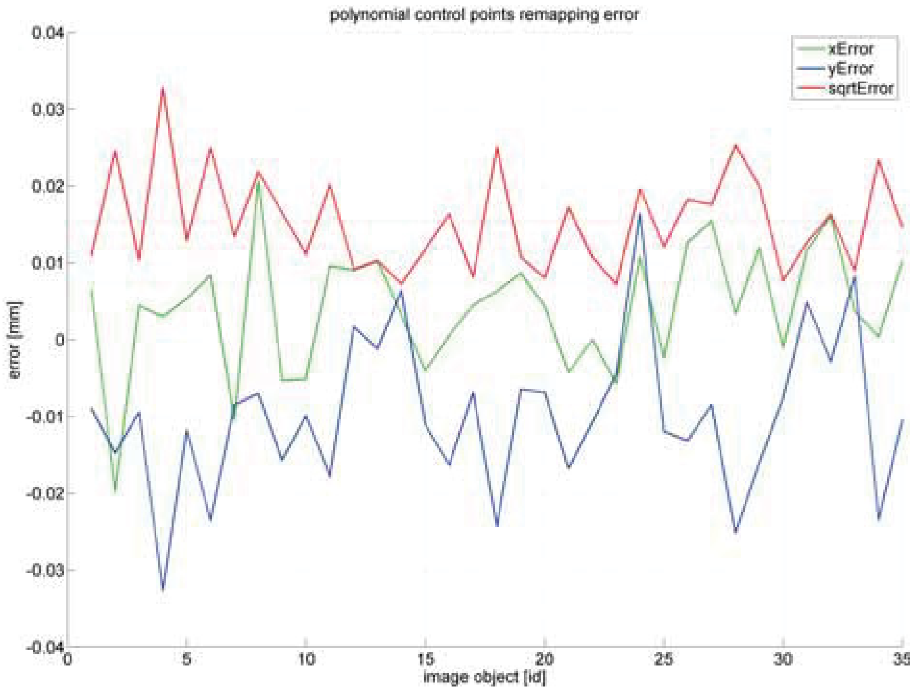


Figure D.30: PolyCtrlPts error utilizing $d = 4$, $f = 12mm$ and circle approximation.

TC#1: $f = 12mm$ / Circle Approximation / TensorBFS

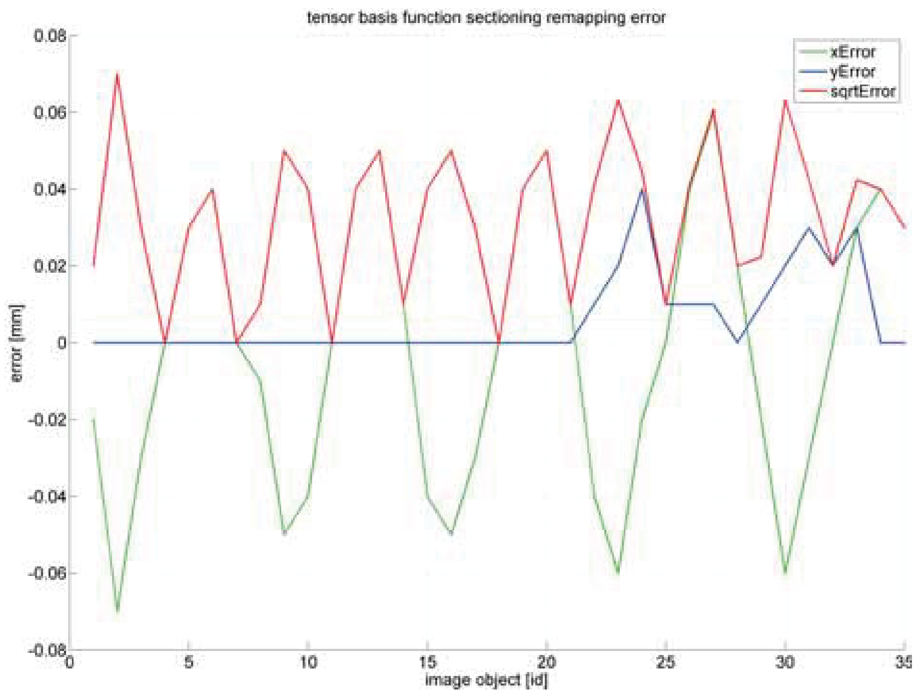


Figure D.31: TensorBFS error utilizing $d = 2$, $f = 12mm$ and circle approximation.

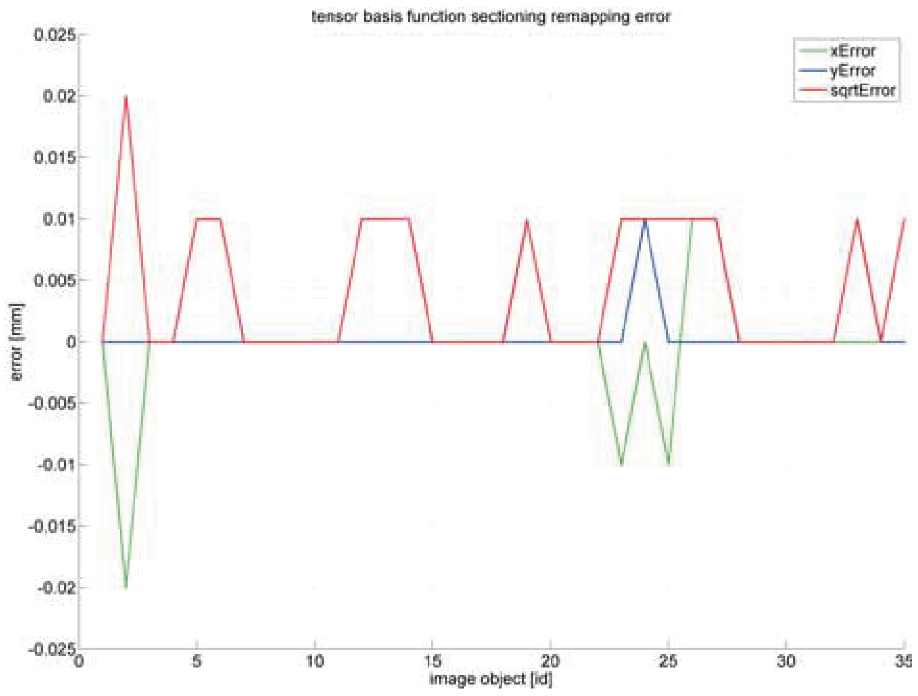


Figure D.32: TensorBFS error utilizing $d = 4$, $f = 12mm$ and circle approximation.

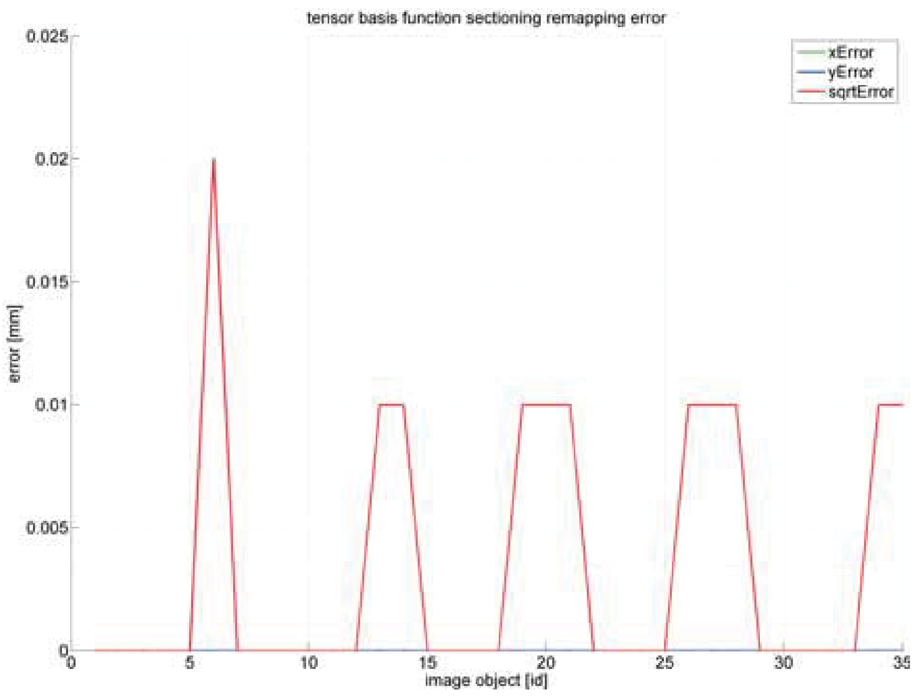


Figure D.33: TensorBFS error utilizing $d = 6$, $f = 12mm$ and circle approximation.

TC#1: $f = 12mm$ / Circle Approximation / TensorQDT

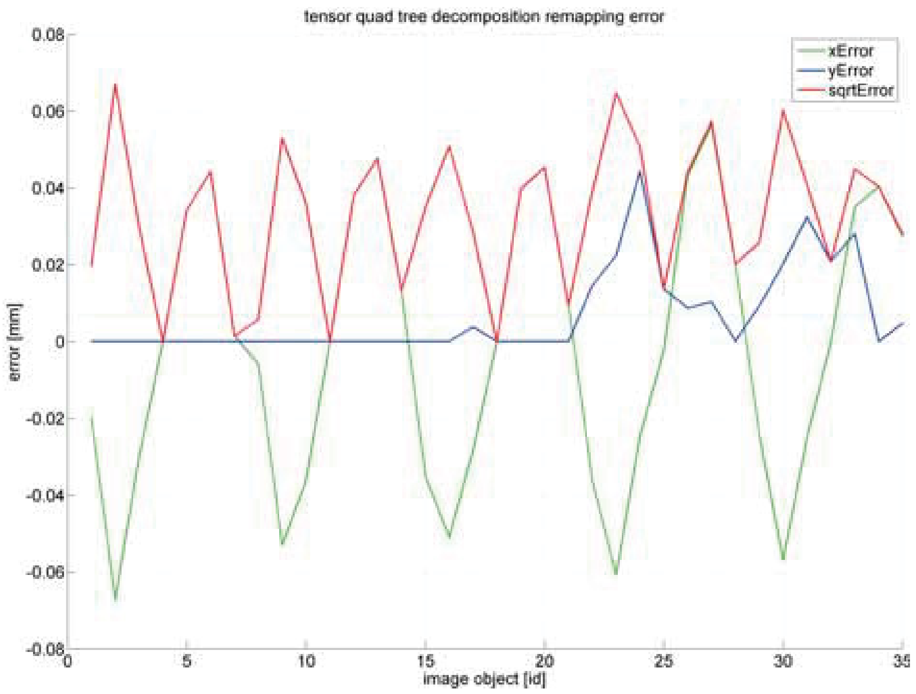


Figure D.34: TensorQDT error utilizing $d = 2$, $f = 12mm$ and circle approximation.

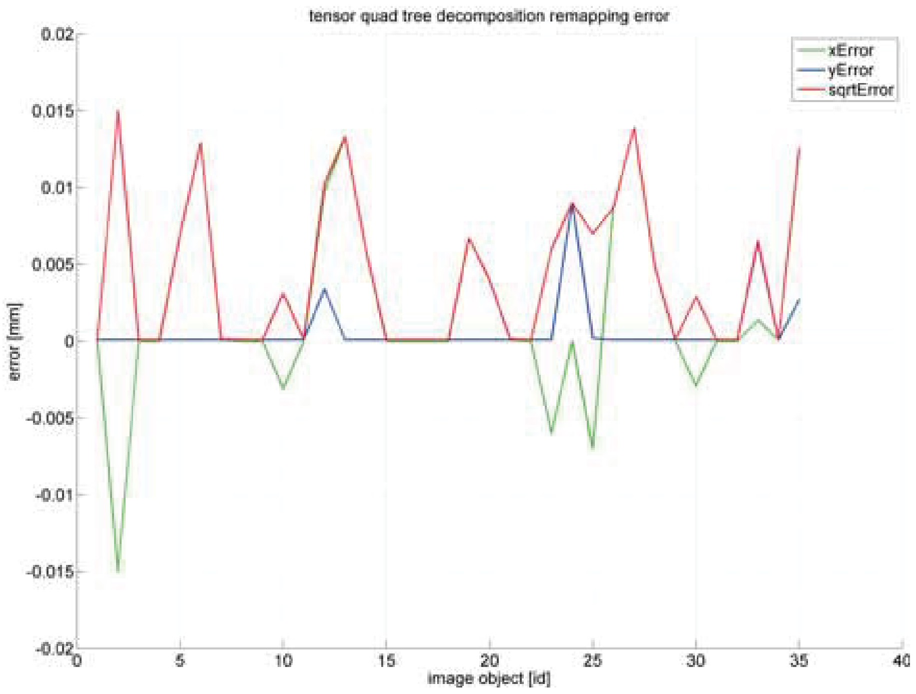


Figure D.35: TensorQDT error utilizing $d = 4$, $f = 12mm$ and circle approximation.

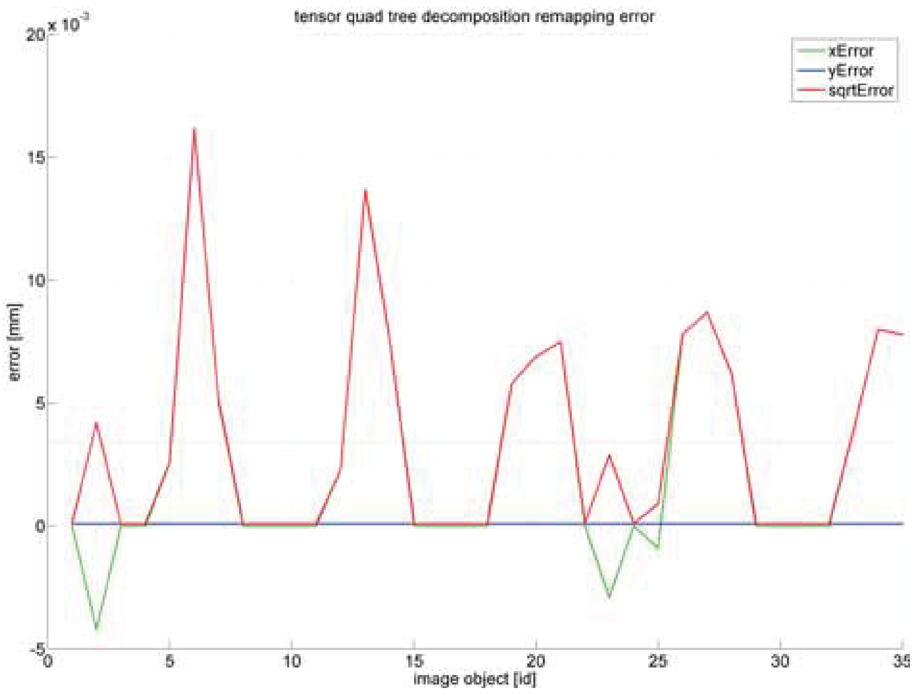


Figure D.36: TensorQDT error utilizing $d = 6$, $f = 12mm$ and circle approximation.

TC#2: $f = 3.6mm$ / Center of Gravity / Homography

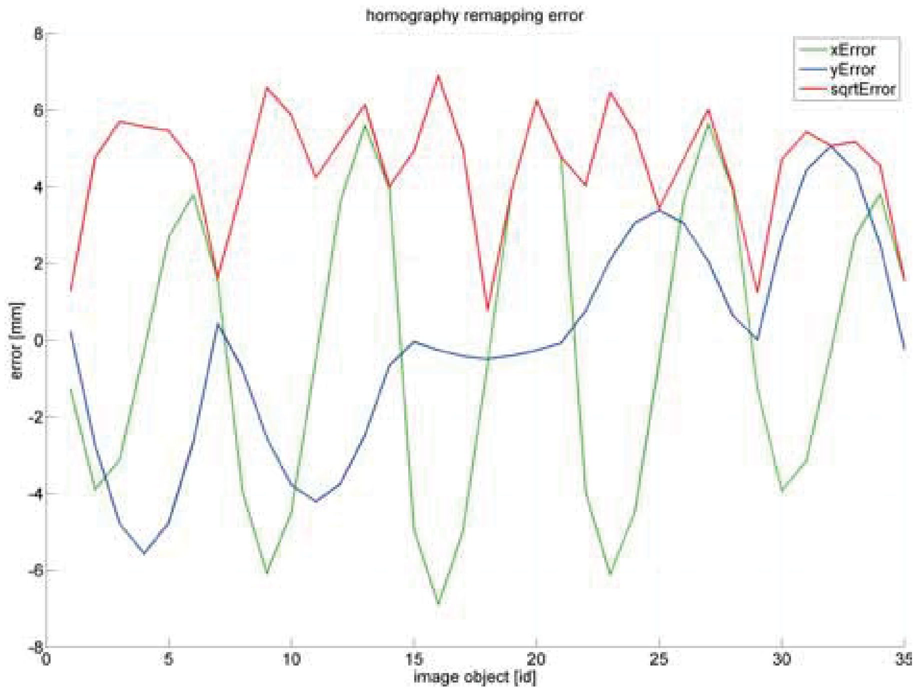


Figure D.37: Homography error utilizing $f = 3.6mm$ and center of gravity.

TC#2: $f = 3.6mm$ / Center of Gravity / PolyCtrlPts

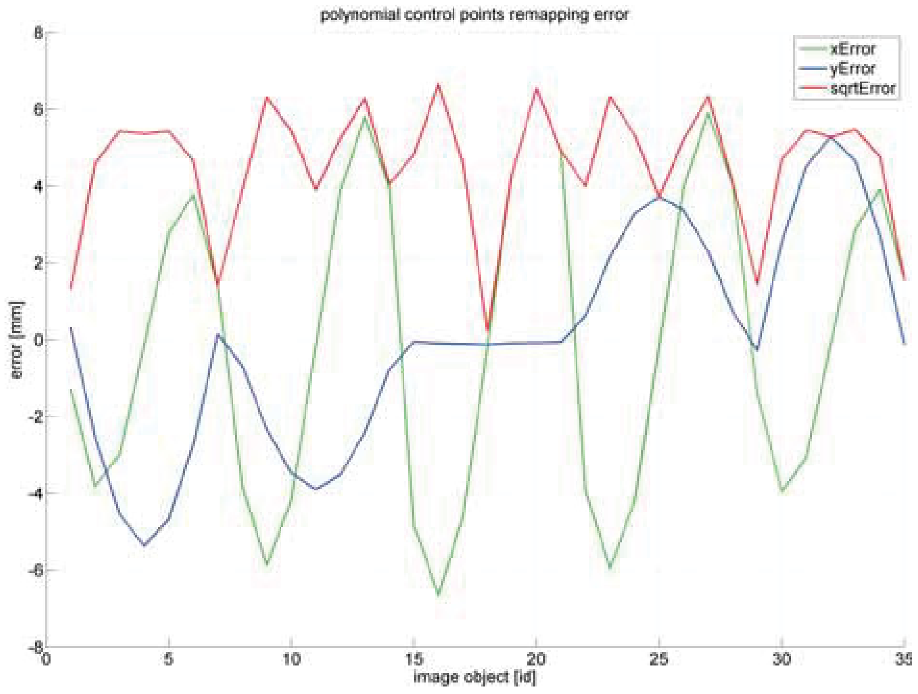


Figure D.38: PolyCtrlPts error utilizing $d = 2$, $f = 3.6mm$ and center of gravity.

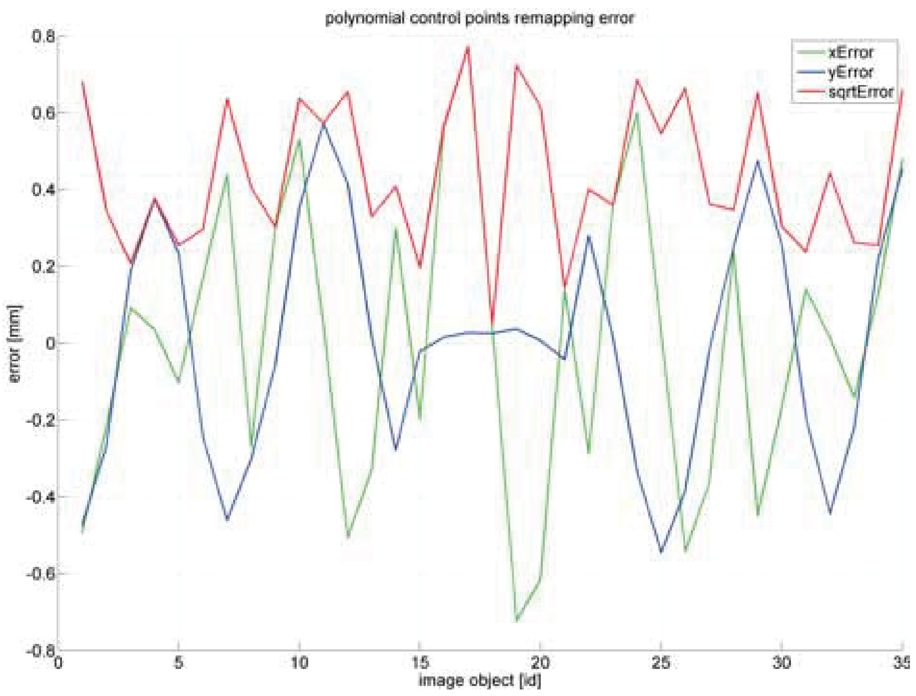


Figure D.39: PolyCtrlPts error utilizing $d = 4$, $f = 3.6mm$ and center of gravity.

TC#2: $f = 3.6mm$ / Center of Gravity / TensorBFS

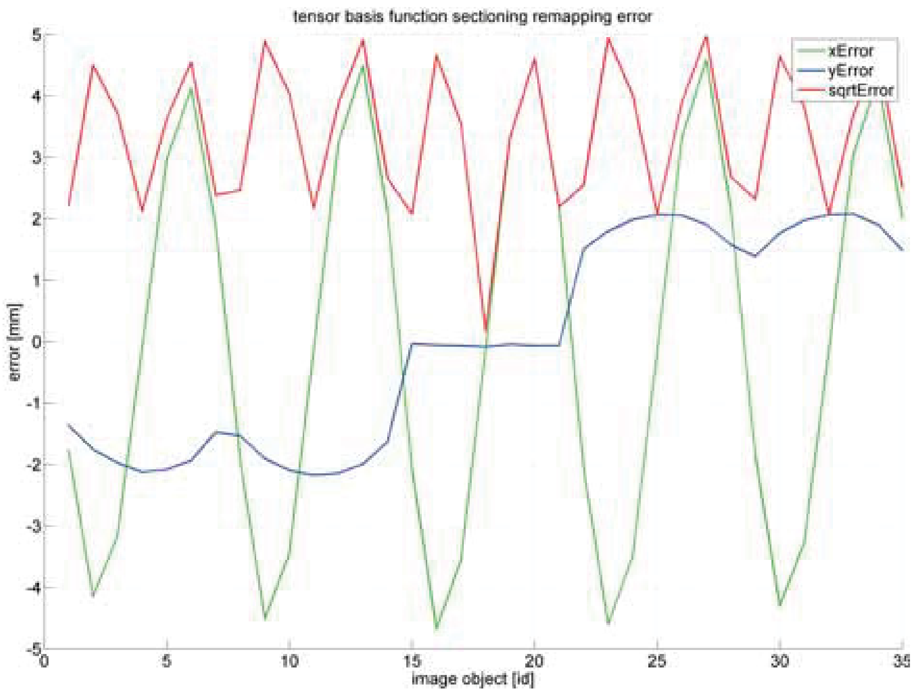


Figure D.40: TensorBFS error utilizing $d = 2$, $f = 3.6mm$ and center of gravity.

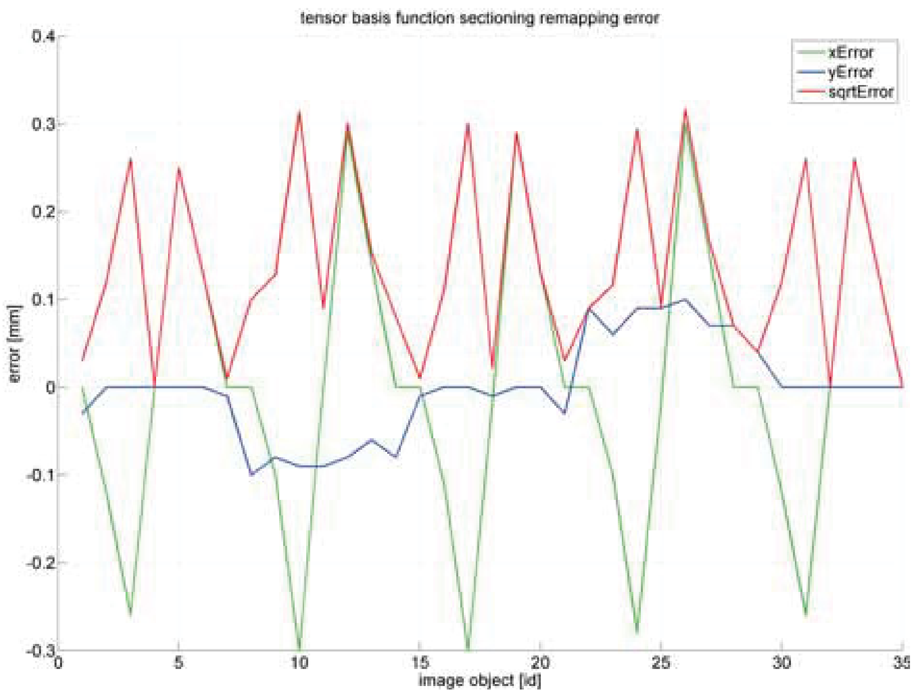


Figure D.41: TensorBFS error utilizing $d = 4,\ f = 3.6mm$ and center of gravity.

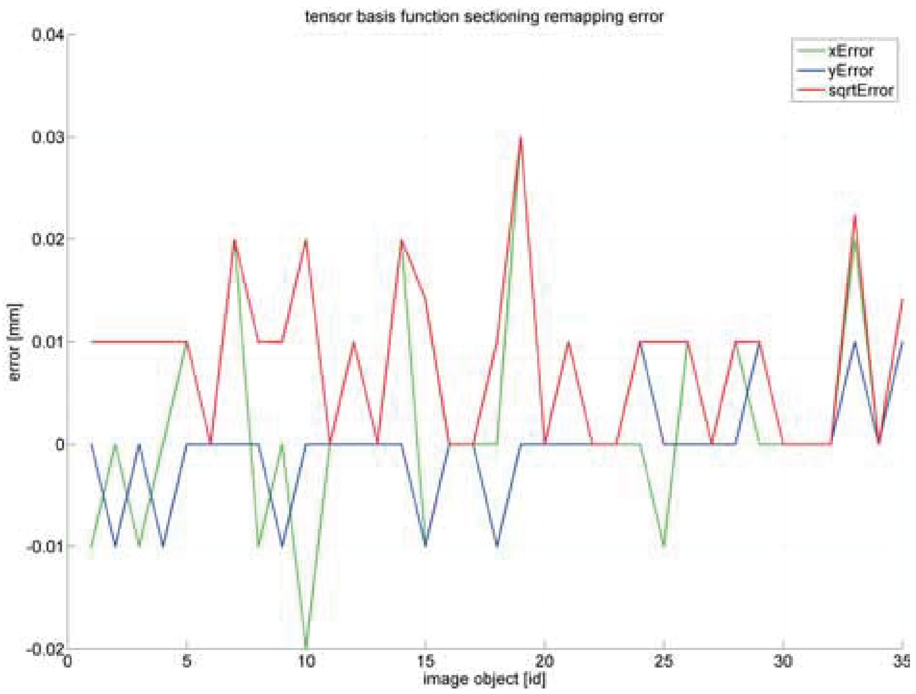


Figure D.42: TensorBFS error utilizing $d = 6,\ f = 3.6mm$ and center of gravity.

TC#2: $f = 3.6mm$ **/ Center of Gravity / TensorQDT**

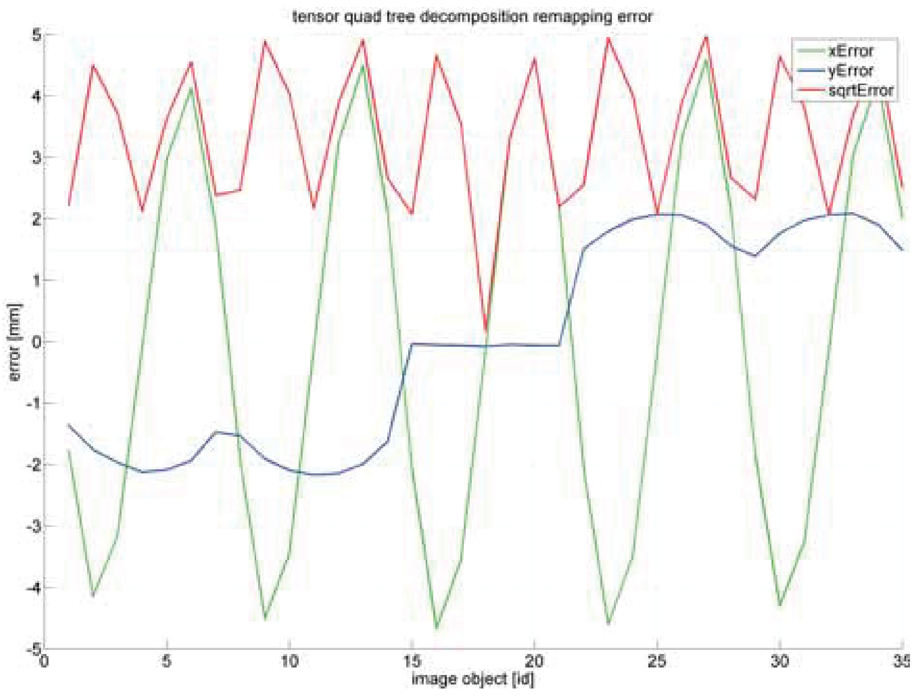


Figure D.43: TensorQDT error utilizing $d = 2$, $f = 3.6mm$ and center of gravity.

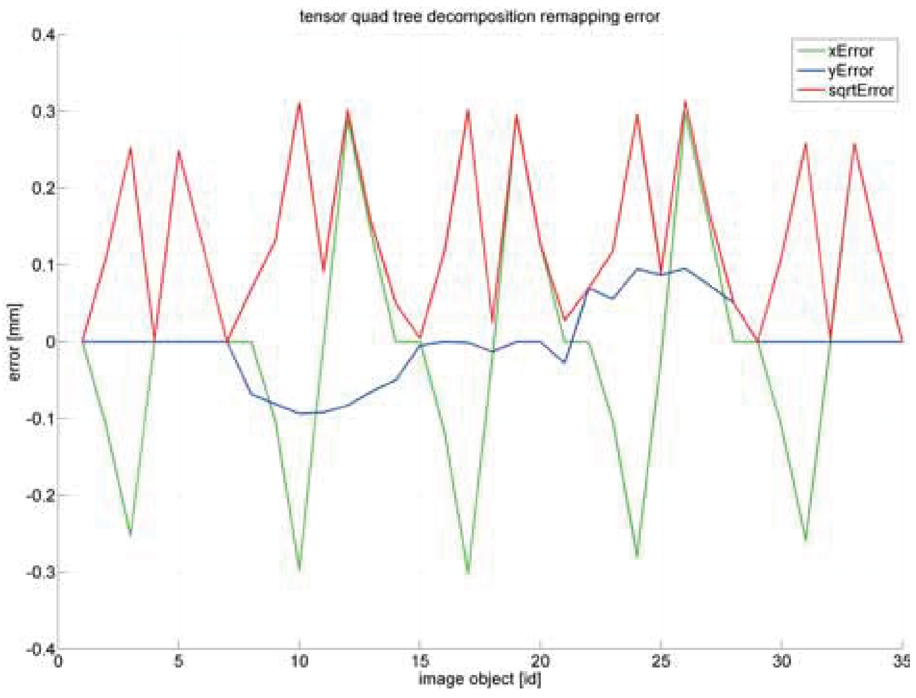


Figure D.44: TensorQDT error utilizing $d = 4$, $f = 3.6mm$ and center of gravity.

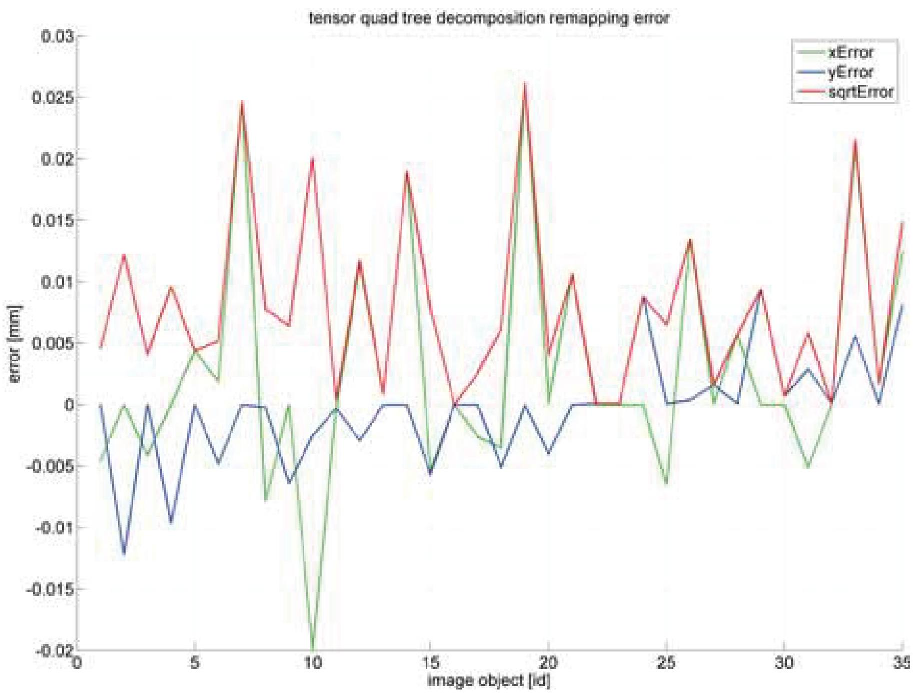


Figure D.45: TensorQDT error utilizing $d = 6$, $f = 3.6mm$ and center of gravity.

TC#2: $f = 3.6mm$ / Gaussian Curve Extremum / Homography

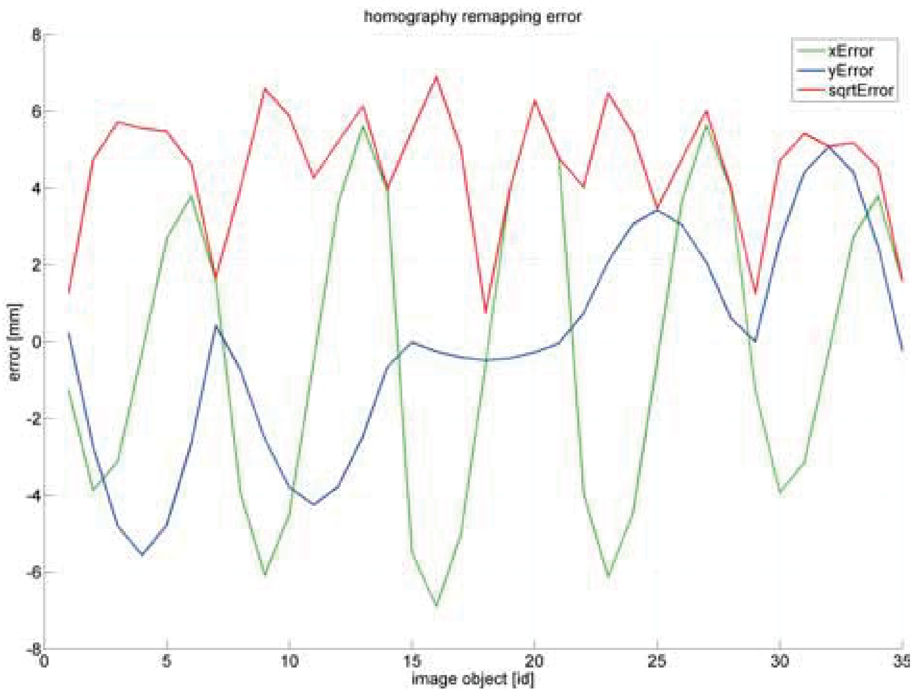


Figure D.46: Homography error utilizing $f = 3.6mm$ and Gaussian curve extremum.

TC#2: $f = 3.6mm$ / Gaussian Curve Extremum / PolyCtrlPts

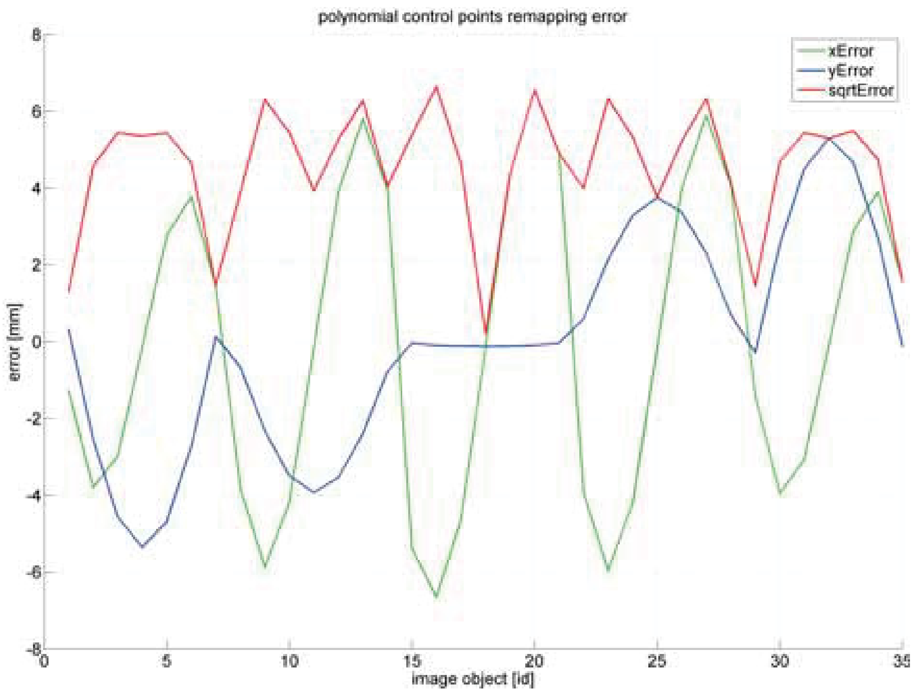


Figure D.47: PolyCtrlPts error utilizing $d = 2$, $f = 3.6mm$ and Gaussian curve extremum.

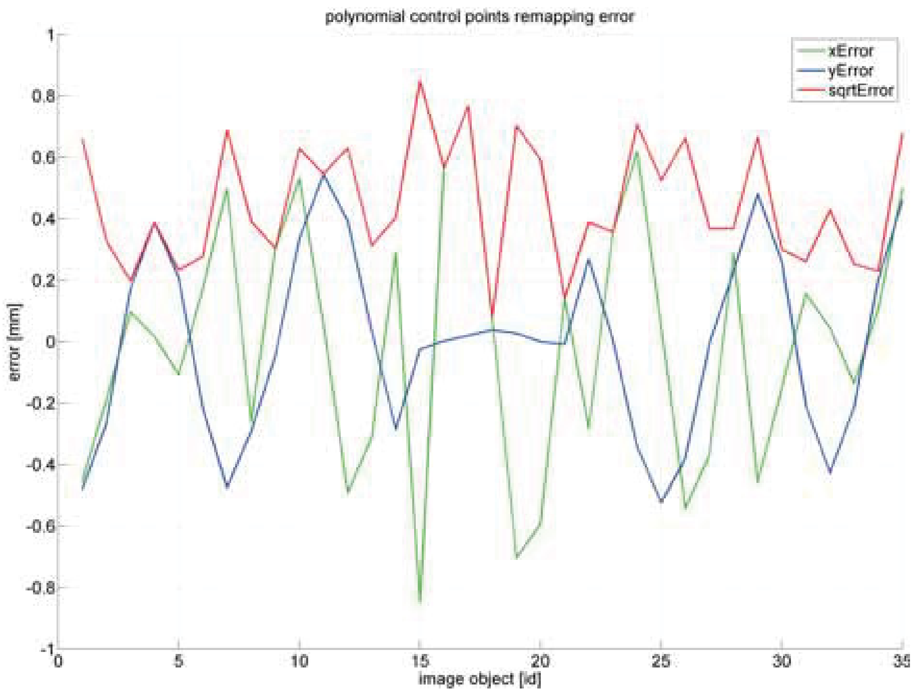


Figure D.48: PolyCtrlPts error utilizing $d = 4$, $f = 3.6mm$ and Gaussian curve extremum.

TC#2: $f = 3.6mm$ / **Gaussian Curve Extremum / TensorBFS**

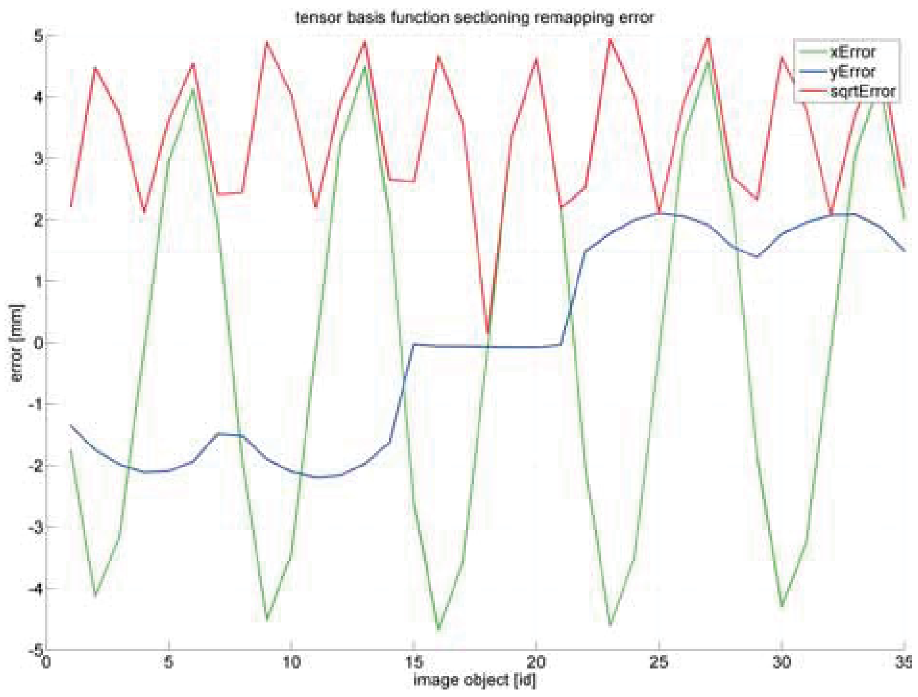


Figure D.49: TensorBFS error utilizing $d = 2$, $f = 3.6mm$ and Gaussian curve extremum.

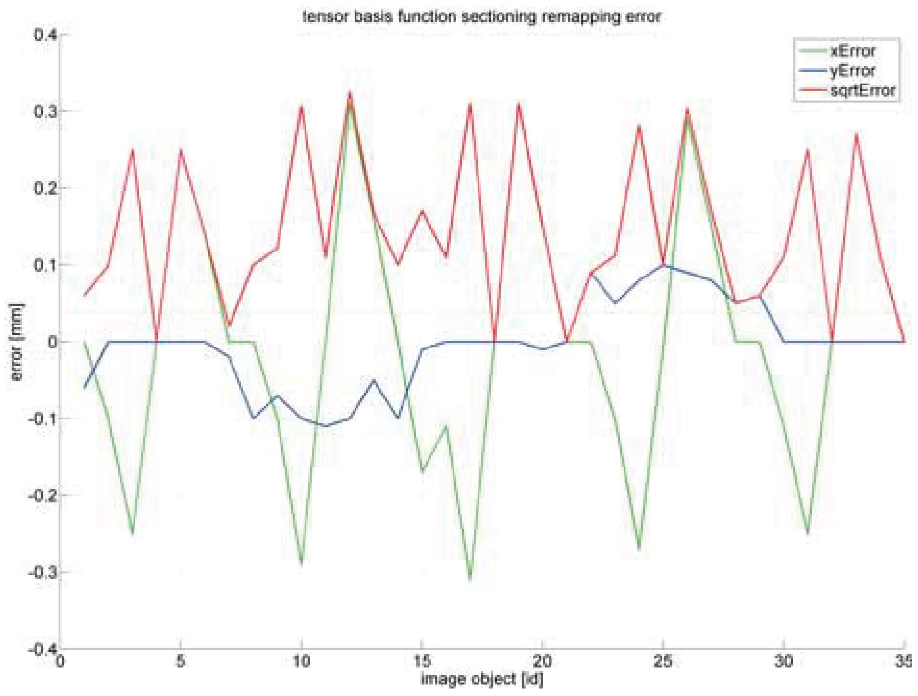


Figure D.50: TensorBFS error utilizing $d = 4$, $f = 3.6mm$ and Gaussian curve extremum.

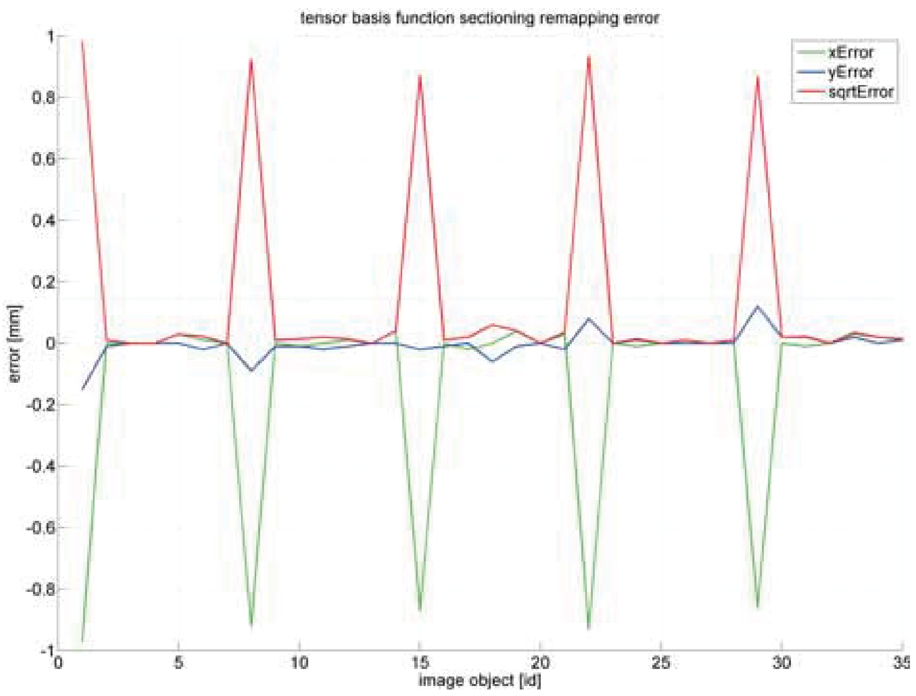


Figure D.51: TensorBFS error utilizing $d = 6$, $f = 3.6mm$ and Gaussian curve extremum.

TC#2: $f = 3.6mm$ / Gaussian Curve Extremum / TensorQDT

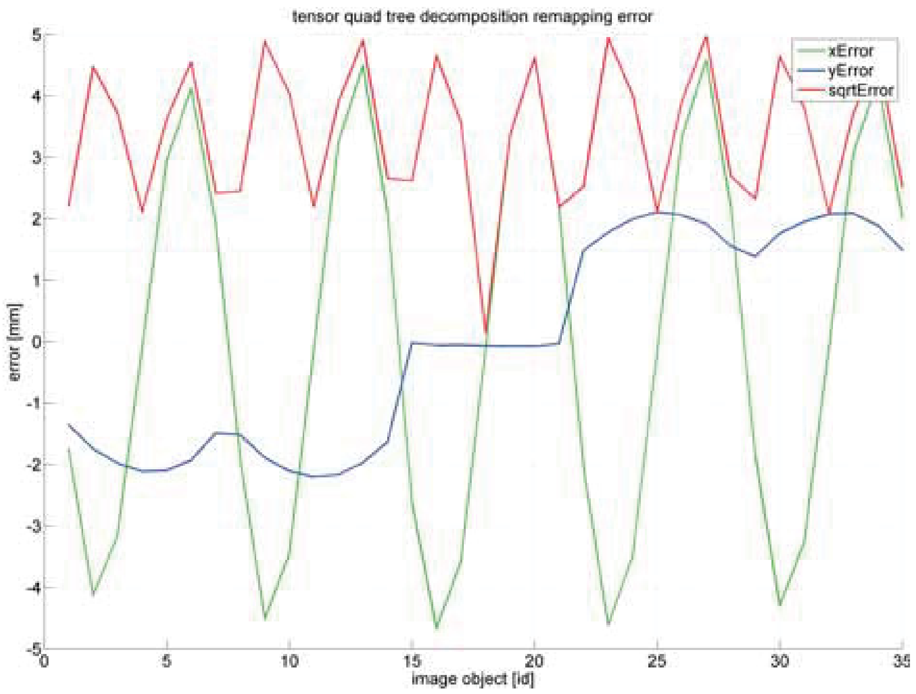


Figure D.52: TensorQDT error utilizing $d = 2$, $f = 3.6mm$ and Gaussian curve extremum.

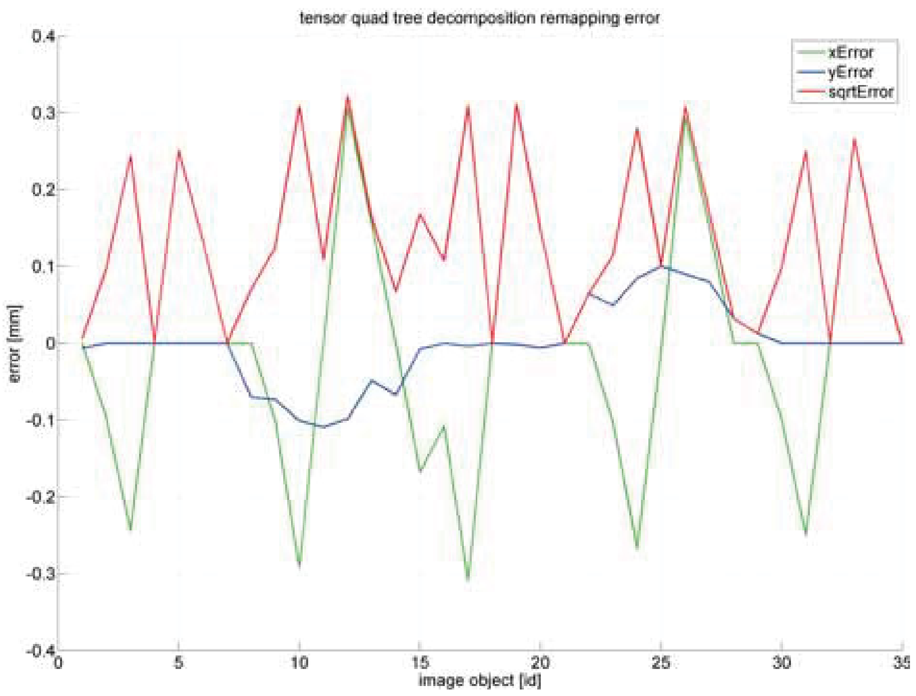


Figure D.53: TensorQDT error utilizing $d = 4$, $f = 3.6mm$ and Gaussian curve extremum.

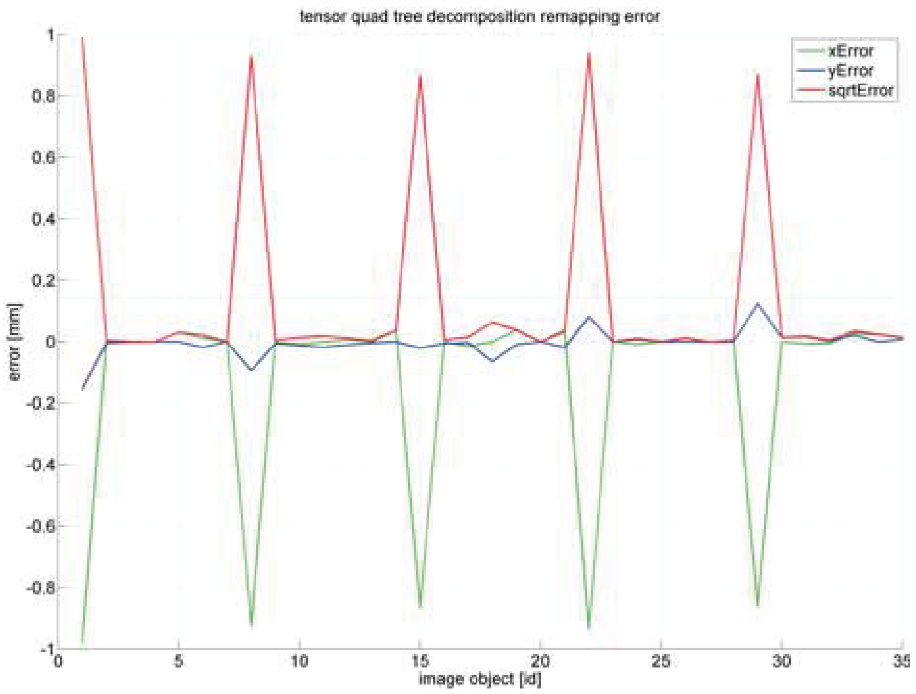


Figure D.54: TensorQDT error utilizing $d = 6$, $f = 3.6mm$ and Gaussian curve extremum.

TC#2: $f = 3.6mm$ / Ellipse Approximation / Homography

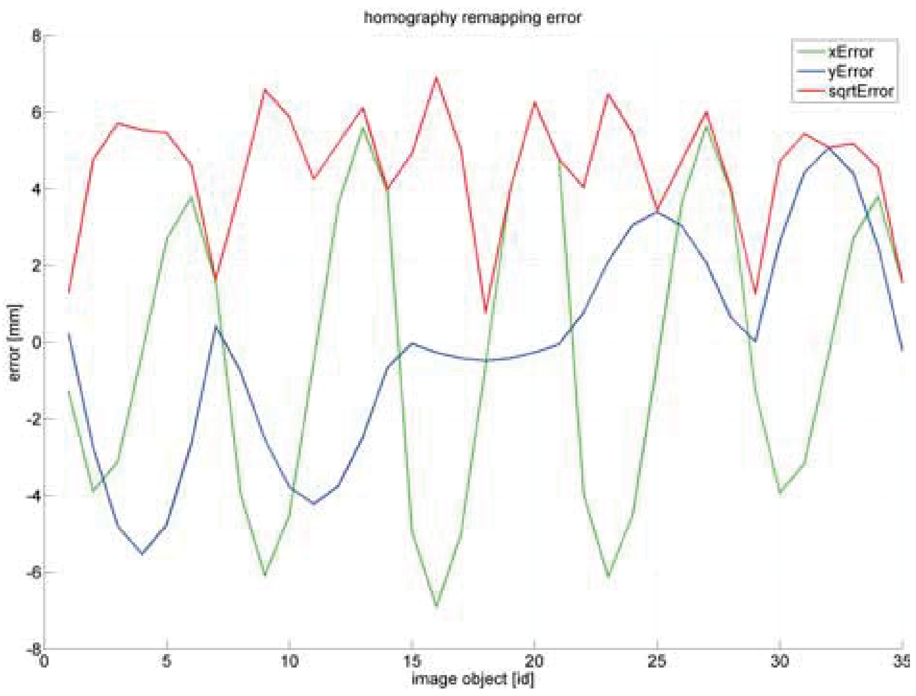


Figure D.55: Homography error utilizing $f = 3.6mm$ and ellipse approximation.

TC#2: $f = 3.6mm$ / Ellipse Approximation / PolyCtrlPts

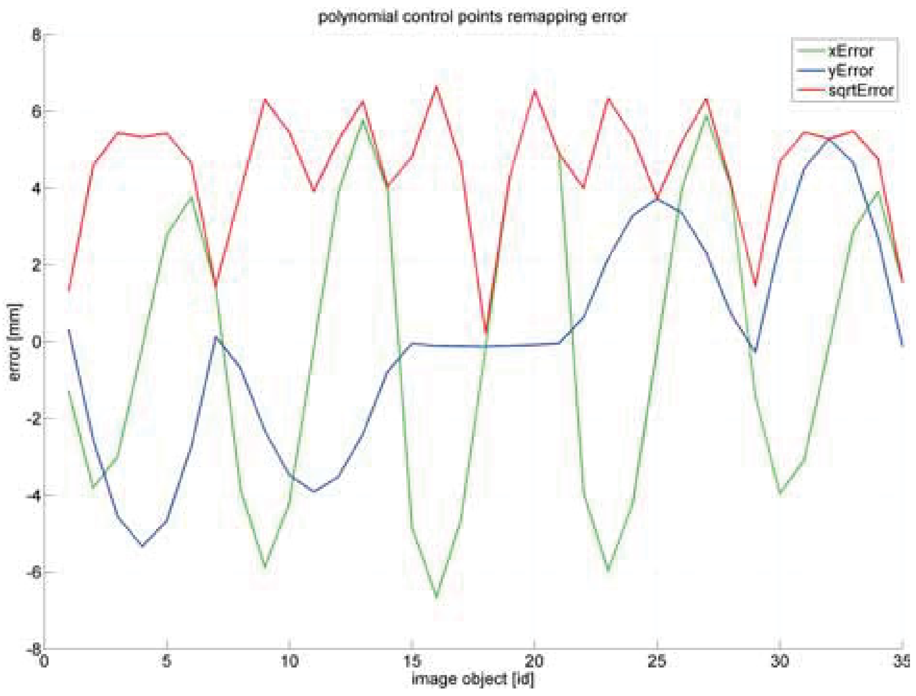


Figure D.56: PolyCtrlPts error utilizing $d = 2$, $f = 3.6mm$ and ellipse approximation.

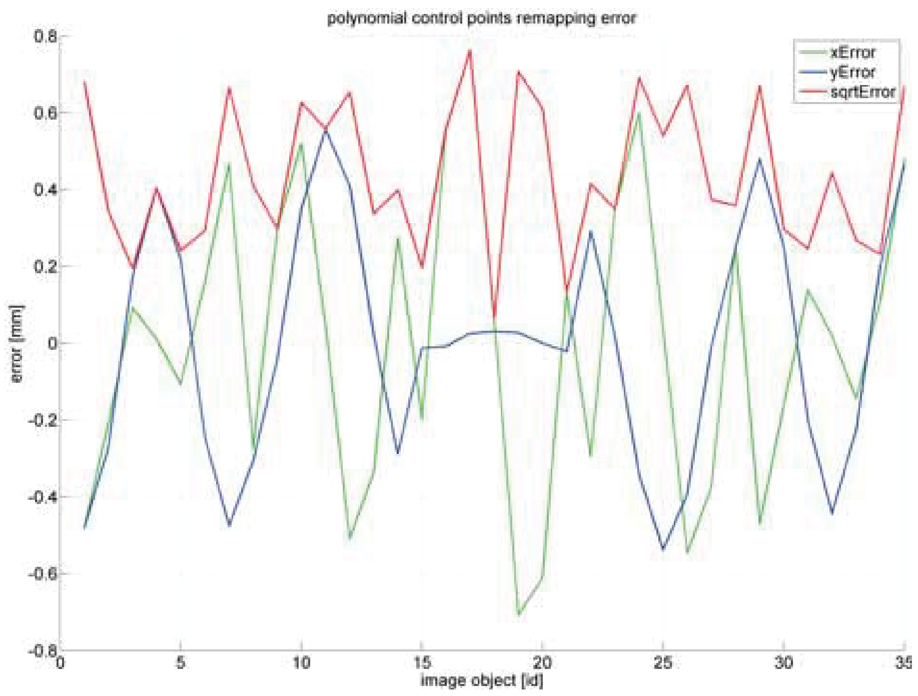


Figure D.57: PolyCtrlPts error utilizing $d = 4$, $f = 3.6mm$ and ellipse approximation.

TC#2: $f = 3.6mm$ / Ellipse Approximation / TensorBFS

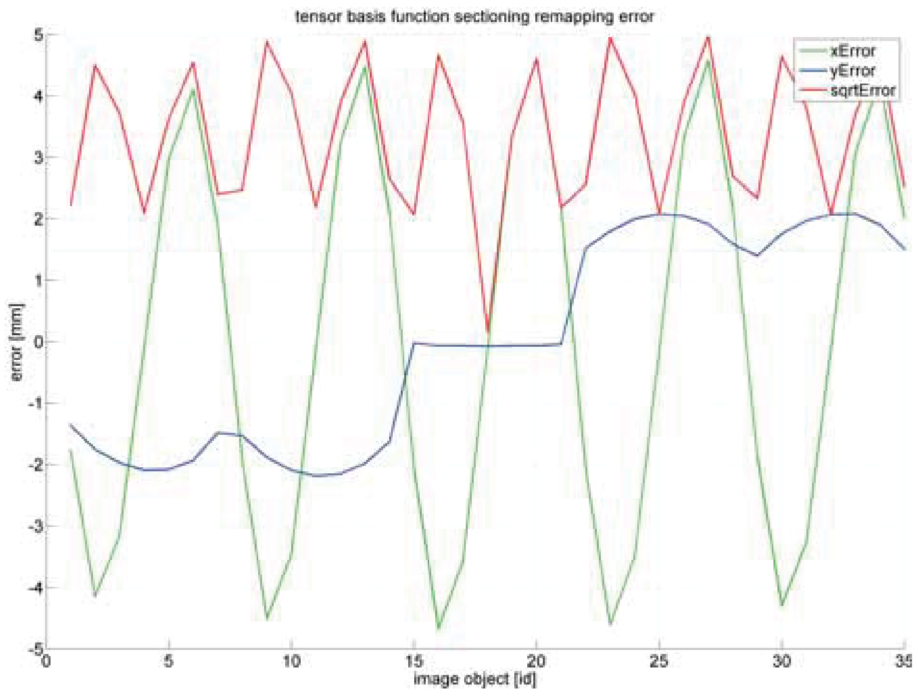


Figure D.58: TensorBFS error utilizing $d = 2$, $f = 3.6mm$ and ellipse approximation.

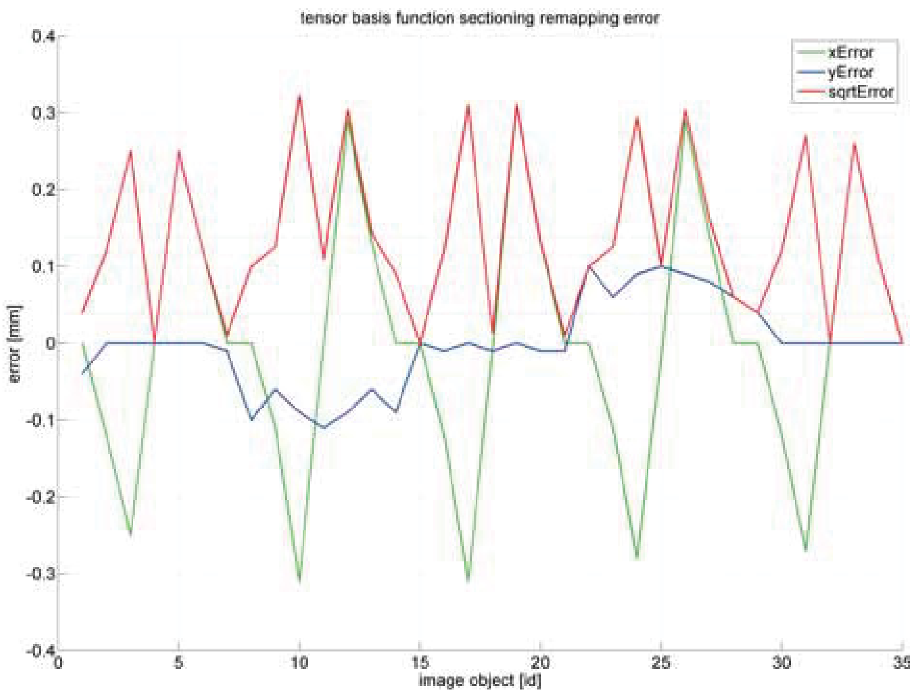


Figure D.59: TensorBFS error utilizing $d = 4$, $f = 3.6mm$ and ellipse approximation.

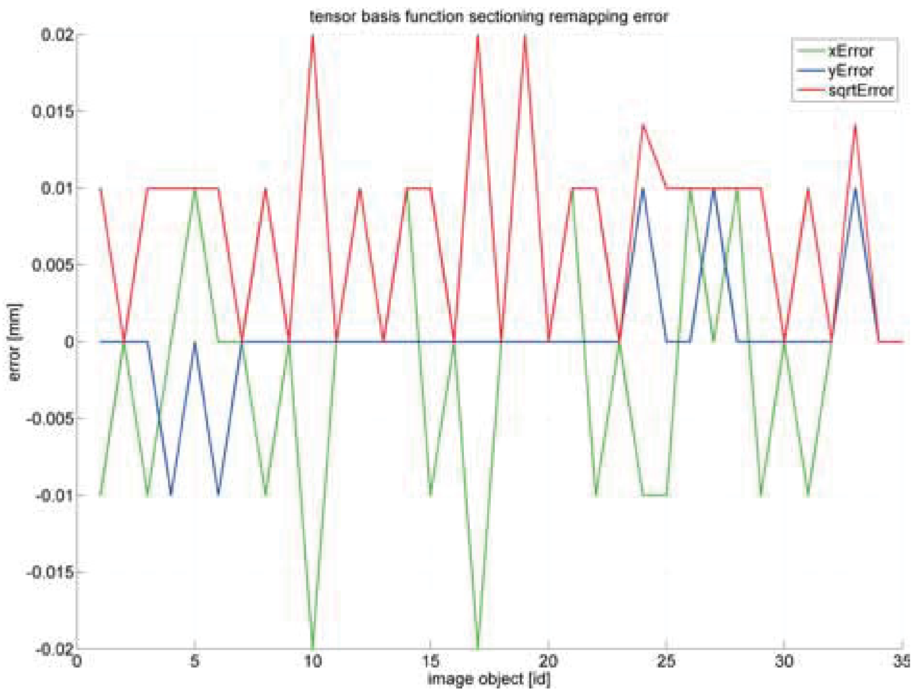


Figure D.60: TensorBFS error utilizing $d = 6$, $f = 3.6mm$ and ellipse approximation.

TC#2: $f = 3.6mm$ / Ellipse Approximation / TensorQDT

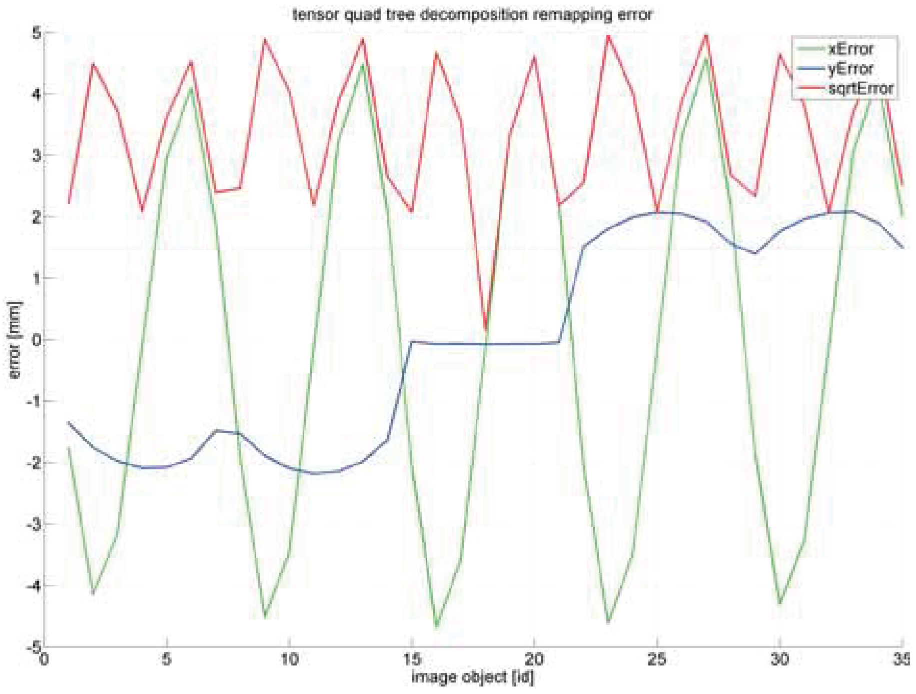


Figure D.61: TensorQDT error utilizing $d = 2$, $f = 3.6mm$ and ellipse approximation.

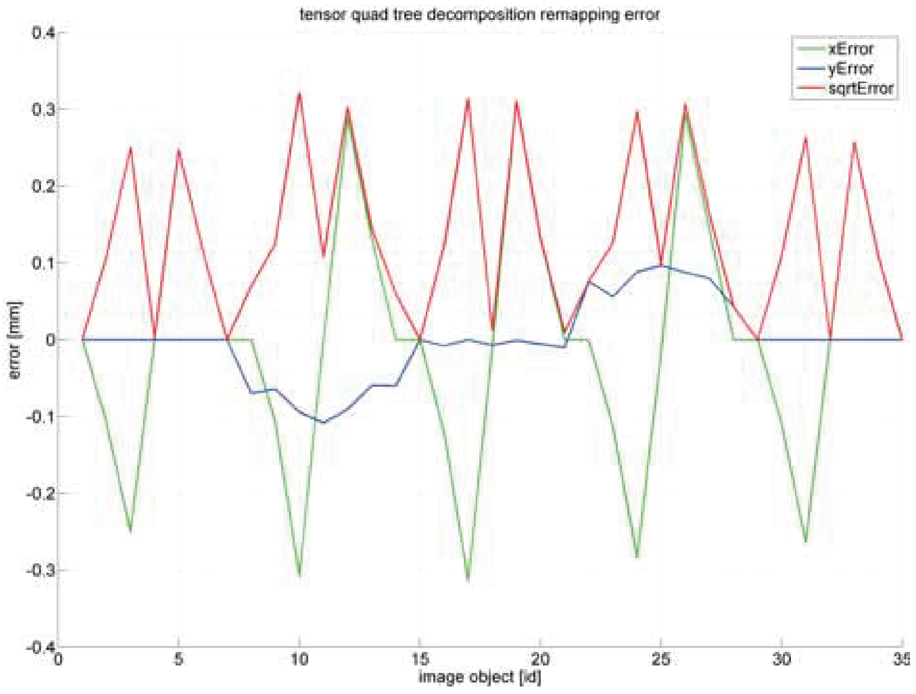


Figure D.62: TensorQDT error utilizing $d = 4$, $f = 3.6mm$ and ellipse approximation.

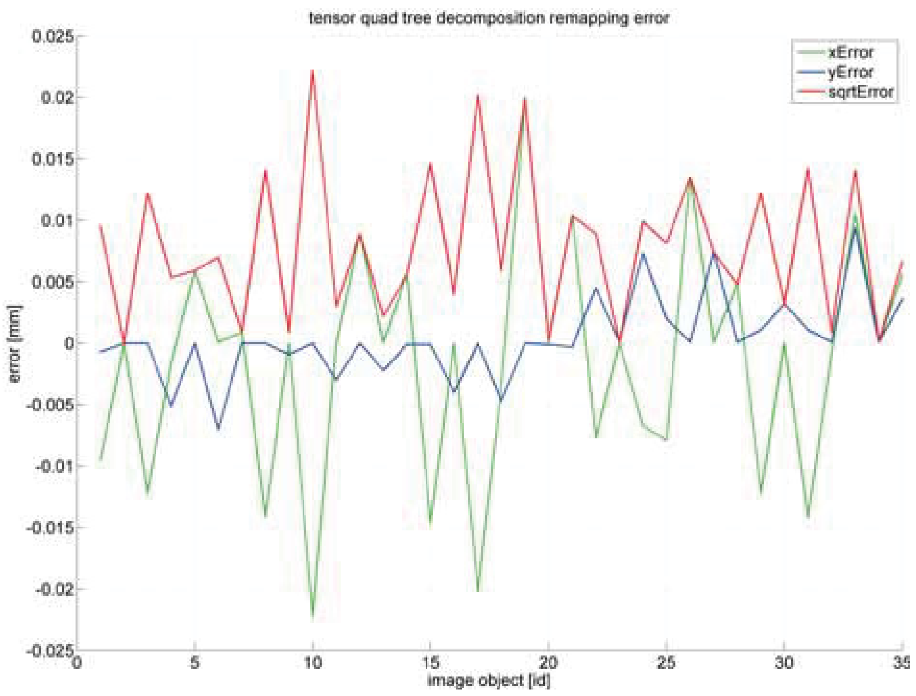


Figure D.63: TensorQDT error utilizing $d = 6,\ f = 3.6mm$ and ellipse approximation.

TC#2: $f = 3.6mm$ **/ Circle Approximation / Homography**

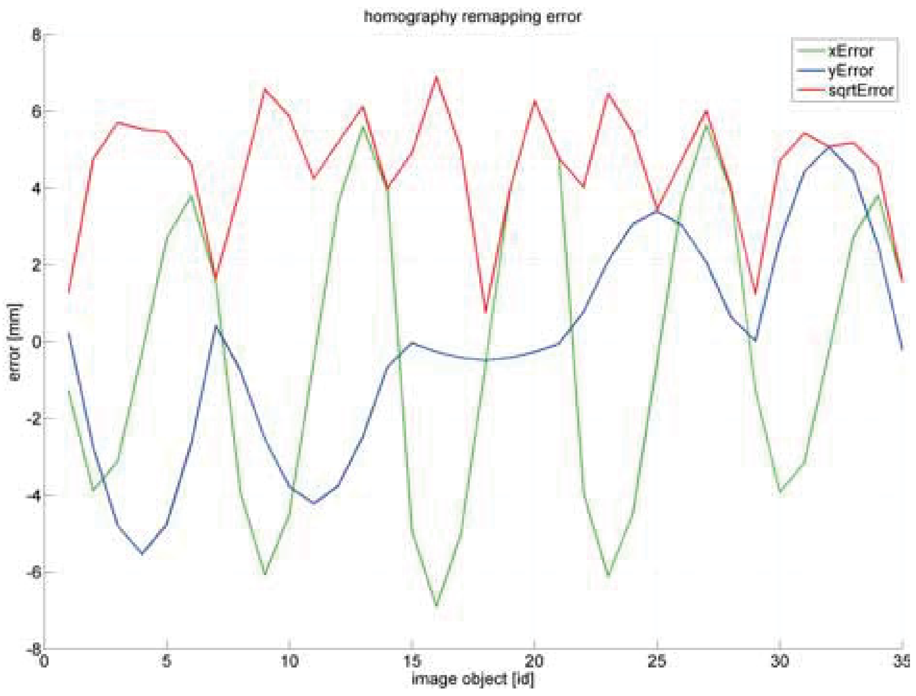


Figure D.64: Homography error utilizing $f = 3.6mm$ and circle approximation.

TC#2: $f = 3.6mm$ **/ Circle Approximation / PolyCtrlPts**

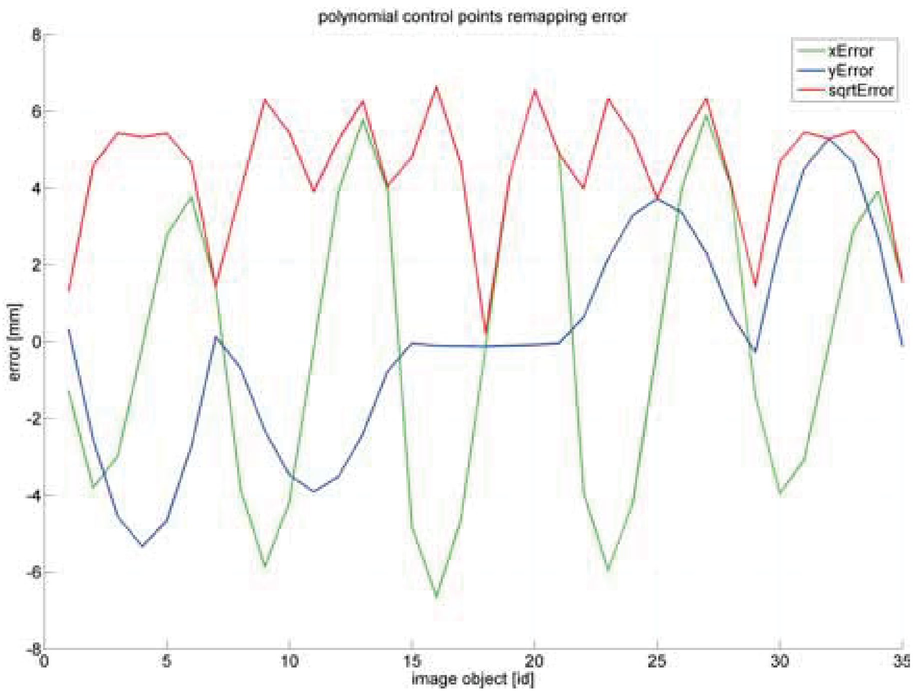


Figure D.65: PolyCtrlPts error utilizing $d = 2$, $f = 3.6mm$ and circle approximation.

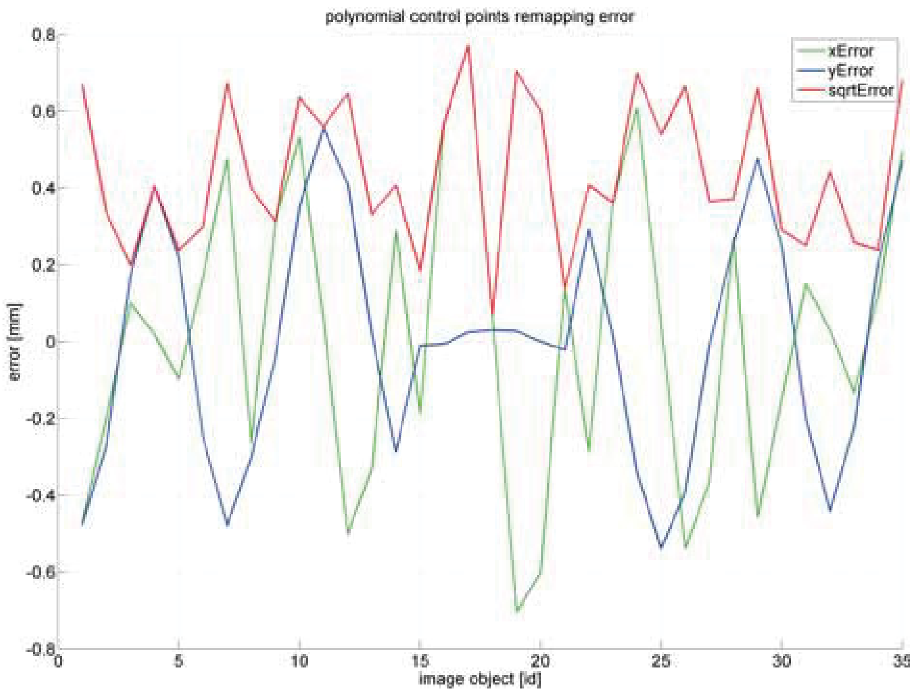


Figure D.66: PolyCtrlPts error utilizing $d = 4$, $f = 3.6mm$ and circle approximation.

TC#2: $f = 3.6mm$ **/ Circle Approximation / TensorBFS**

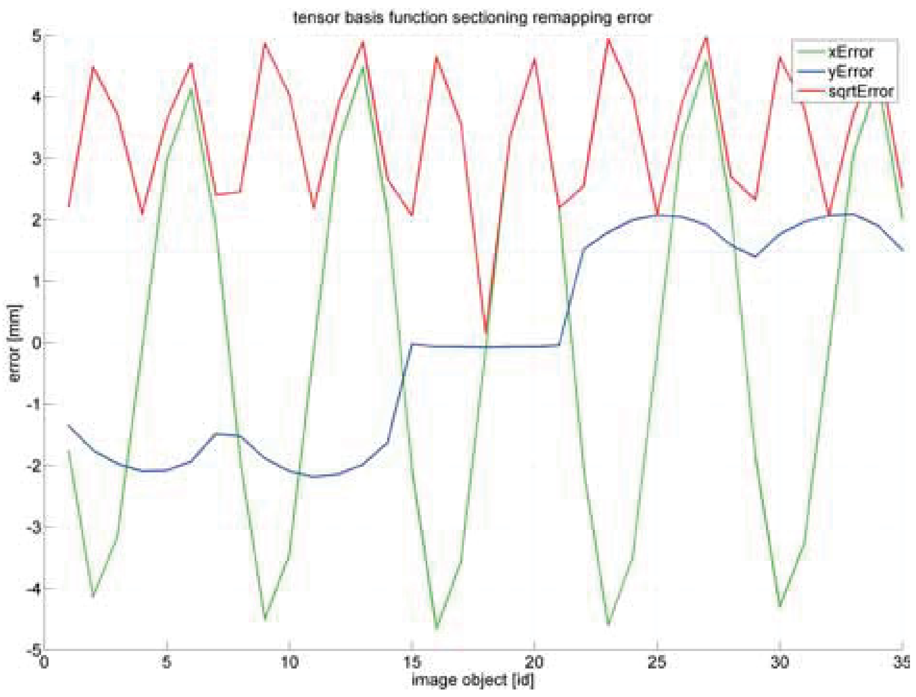


Figure D.67: TensorBFS error utilizing $d = 2$, $f = 3.6mm$ and circle approximation.

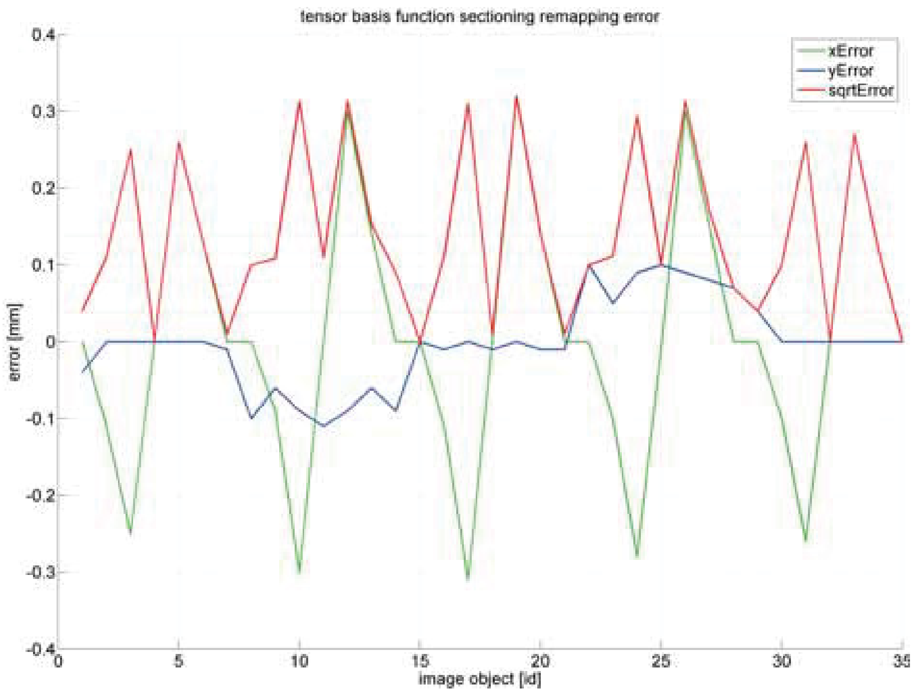


Figure D.68: TensorBFS error utilizing $d = 4$, $f = 3.6mm$ and circle approximation.

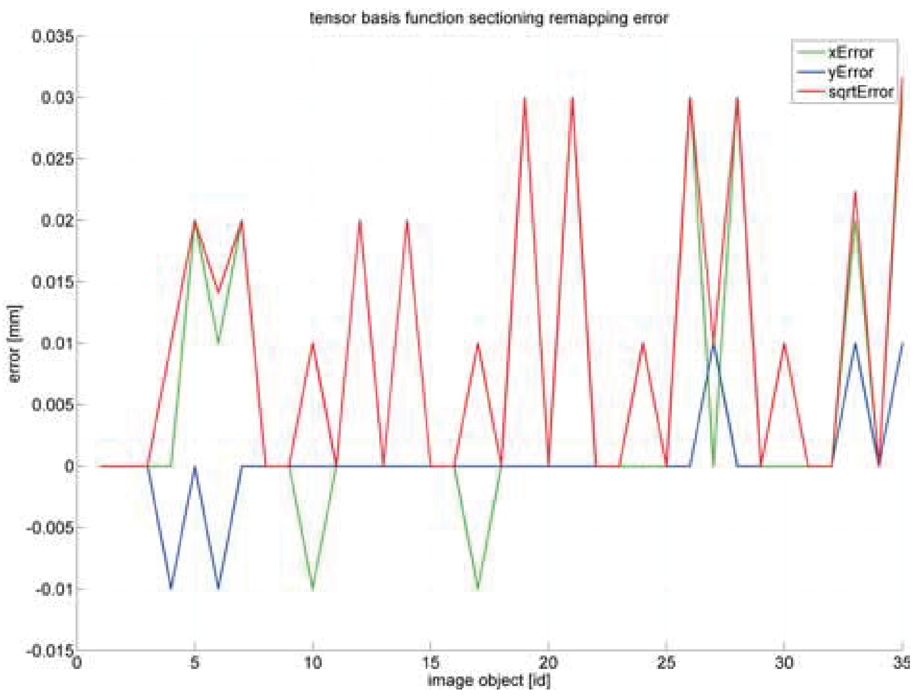


Figure D.69: TensorBFS error utilizing $d = 6,\ f = 3.6mm$ and circle approximation.

TC#2: $f = 3.6mm$ / **Circle Approximation / TensorQDT**

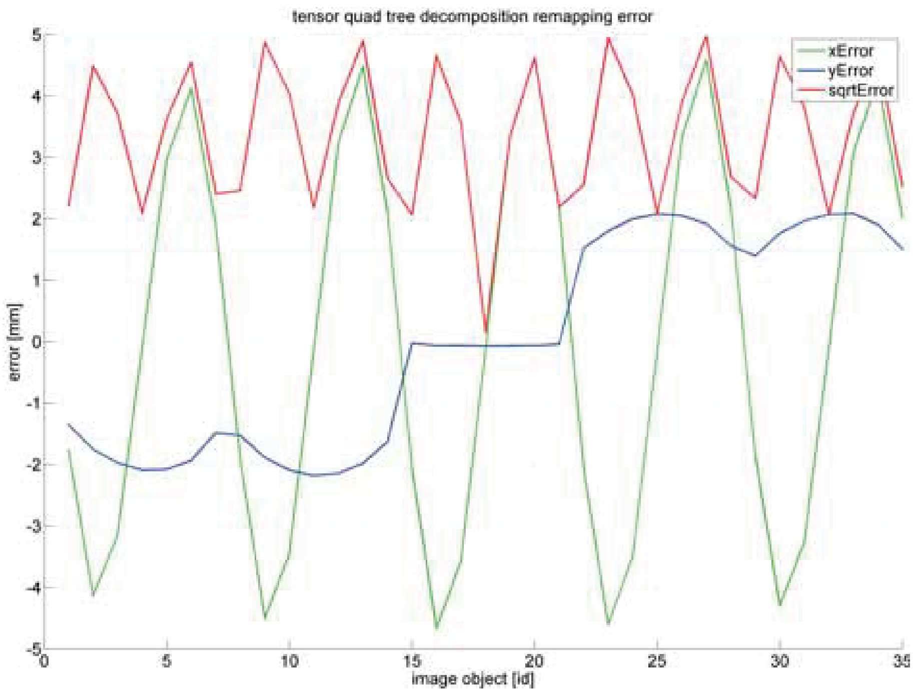


Figure D.70: TensorQDT error utilizing $d = 2,\ f = 3.6mm$ and circle approximation.

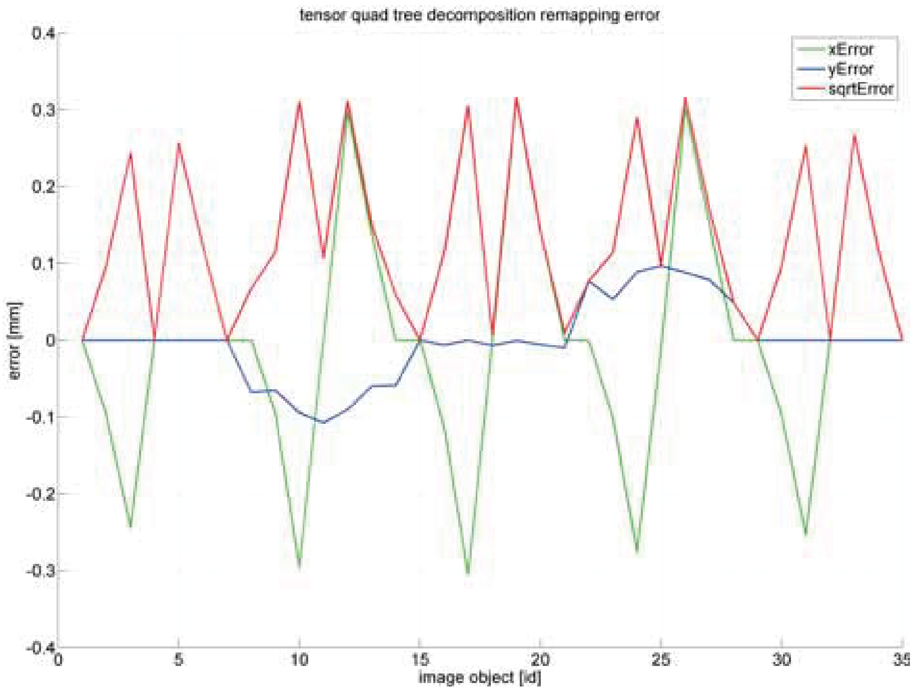


Figure D.71: TensorQDT error utilizing $d = 4$, $f = 3.6mm$ and circle approximation.

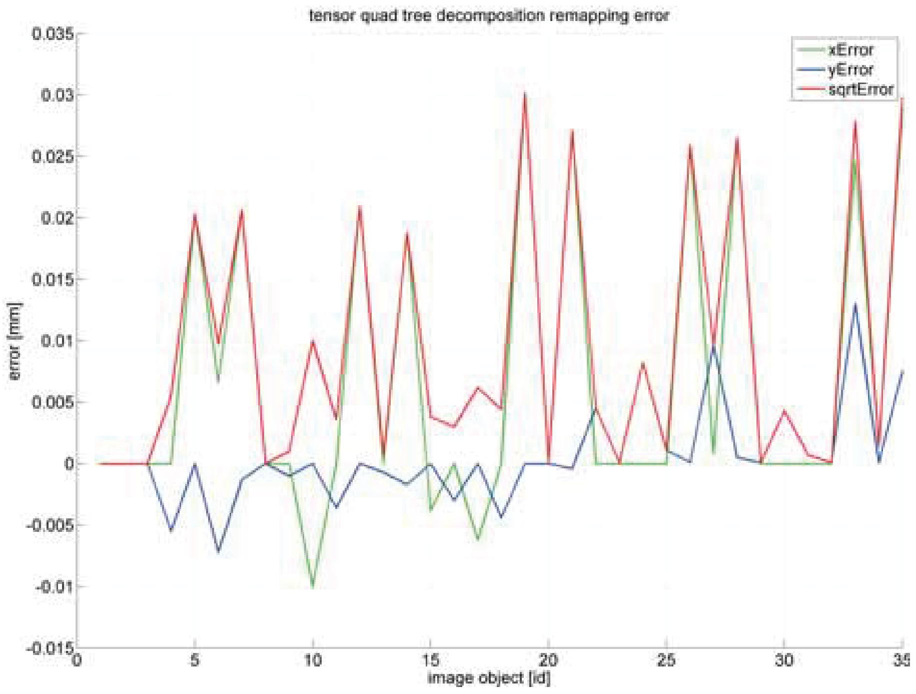


Figure D.72: TensorQDT error utilizing $d = 6$, $f = 3.6mm$ and circle approximation.

Appendix E

Calibration Modes

The measurements are carried out for every introduced center determination technique, furthermore each test-case features calibration via a LED matrix and a laser spot matrix.

1. TC#1 (Test-Case 1): center determination via *center of gravity*;
2. TC#2 (Test-Case 2): center determination via *Gaussian curve extremum*;
3. TC#3 (Test-Case 3): center determination via *ellipse approximation*;
4. TC#4 (Test-Case 4): center determination via *circle approximation.*

The differences b_x and b_y between two measured points are acquired for every row and for every column by

$$b_x = x_{j+1} - x_j \qquad \text{and} \qquad b_y = y_{i+1} - y_i, \tag{E.1}$$

when i is the indexing variable for rows and j is the indexing variable for columns. The cross validation matrix features a $\delta = 25mm$ raster in x and y dimension. The errors e_{x_j} and e_{y_i} are the differences between b and δ for every pair of neighboring points, i.e.

$$e_{x_j} = \delta - b_{x_j} \qquad \text{and} \qquad e_{y_i} = \delta - b_{y_i}, \tag{E.2}$$

when the numbers of investigated objects distances in the interlaced cross validation matrix are $\varepsilon_x = 4 \cdot 7 = 28$ values for b_x and $\varepsilon_y = 3 \cdot 8 = 24$ values for b_y. A lens with a focal length of $f = 12mm$ is employed. The degree $d = 4$ is used for the non-linear mapping functions. At first, every test-case presents the outcome of the different coordinate mapping functions, afterwards the solutions depending on the data source, i.e. LEDs or laser spots, are investigated.

TC#1: Center of Gravity / Coordinate Mapping Error

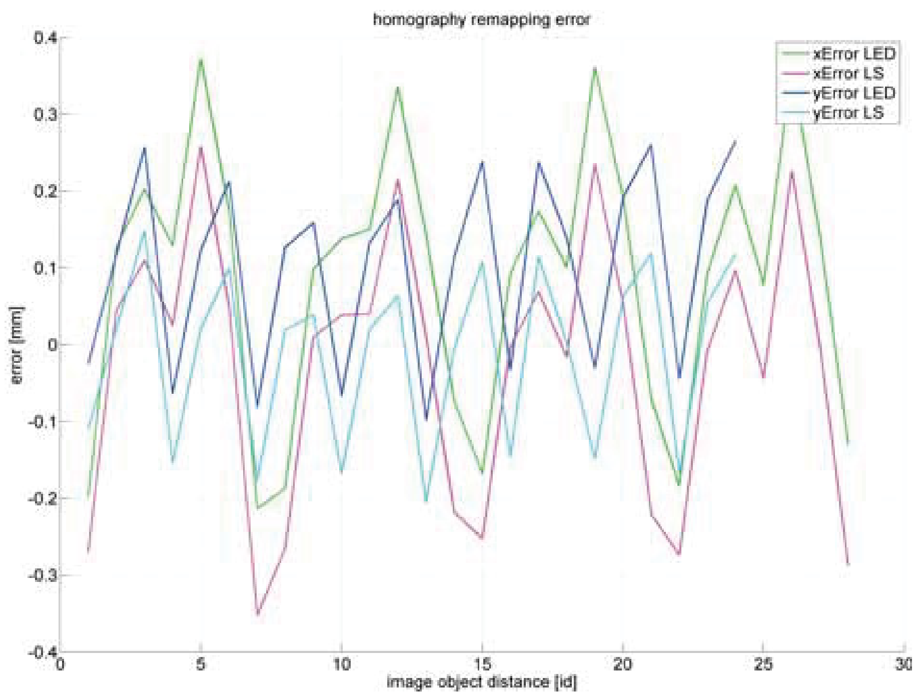


Figure E.1: Homography error utilizing center of gravity.

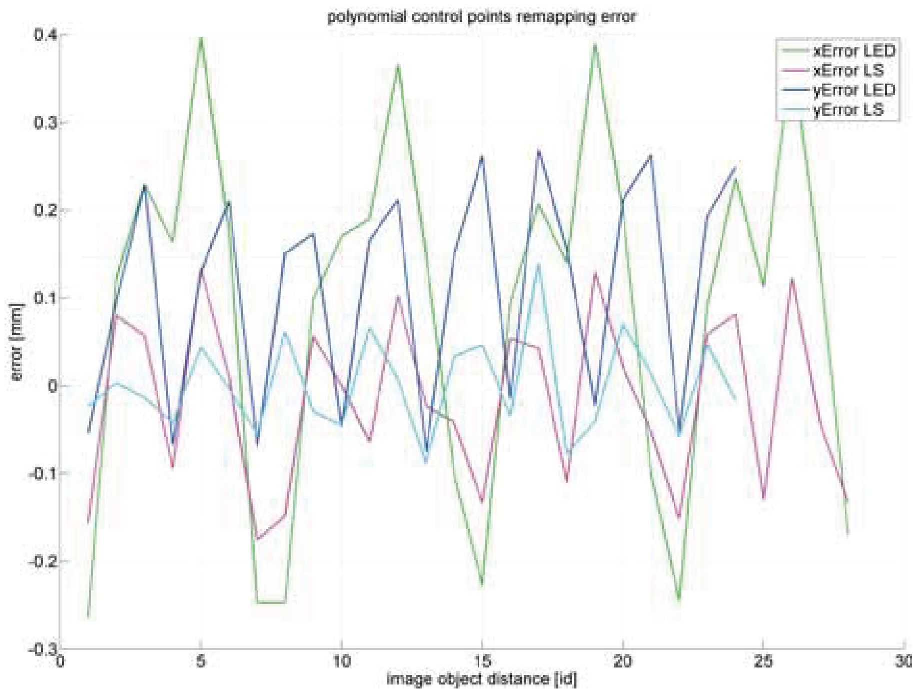


Figure E.2: PolyCtrlPts error utilizing center of gravity.

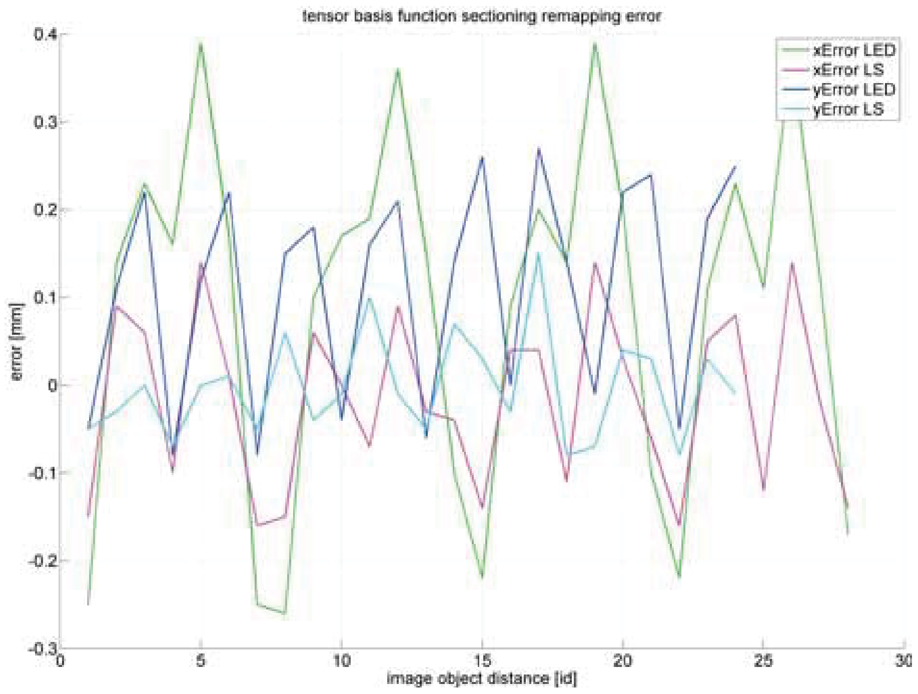


Figure E.3: TensorBFS error utilizing center of gravity.

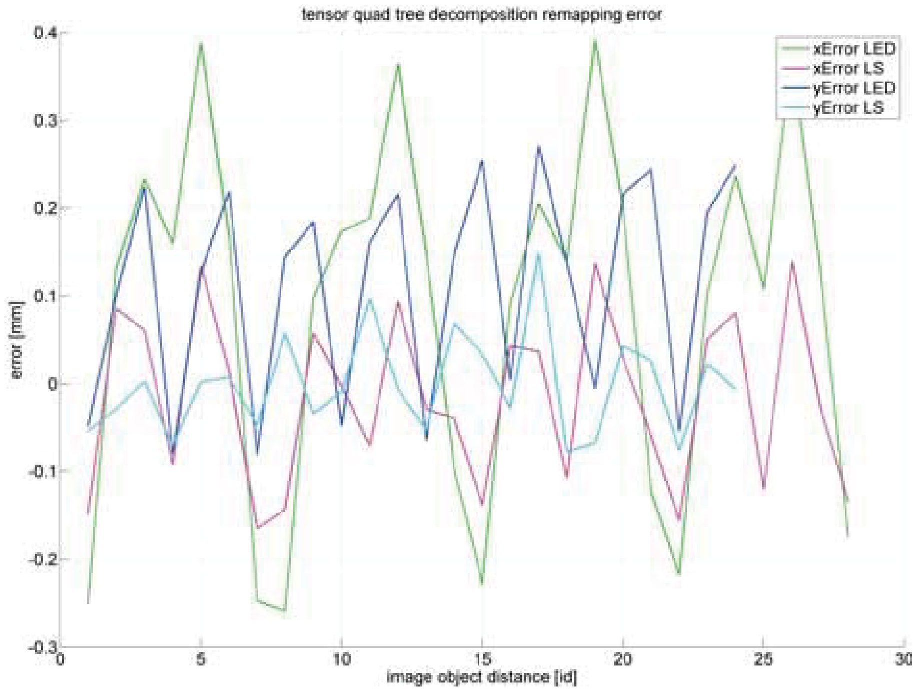


Figure E.4: TensorQDT error utilizing center of gravity.

TC#1: Center of Gravity / LEDs vs. Laser Spots Error

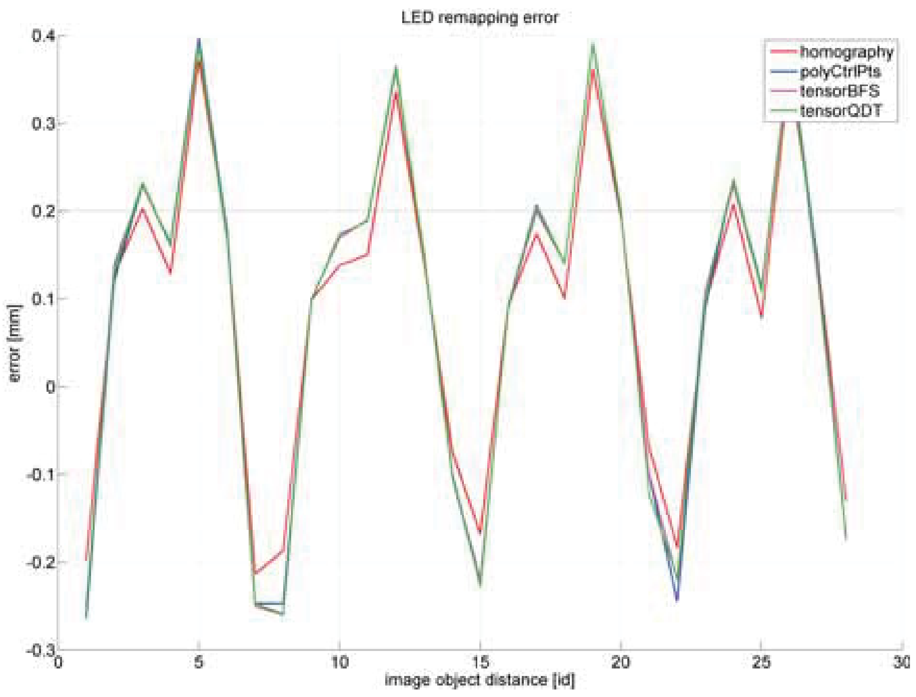


Figure E.5: Error in x dimension using LEDs and center of gravity.

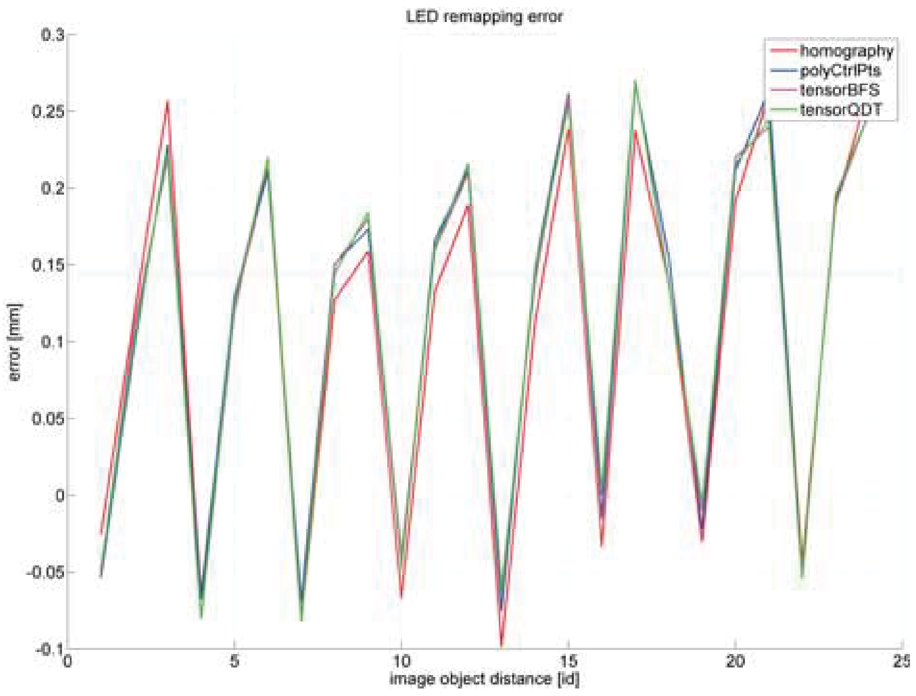


Figure E.6: Error in y dimension using LEDs and center of gravity.

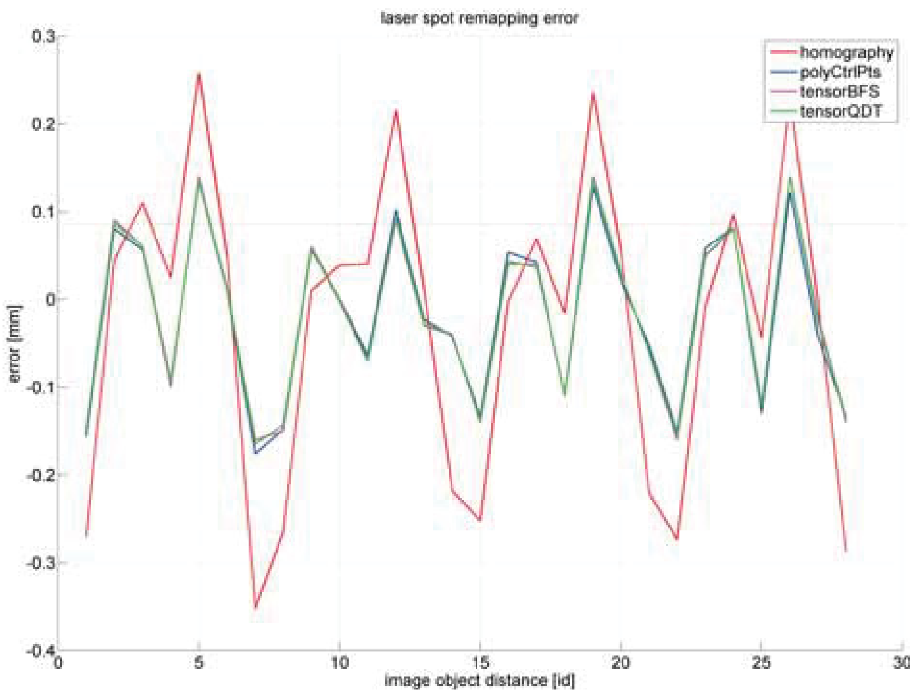


Figure E.7: Error in x dimension using laser spots and center of gravity.

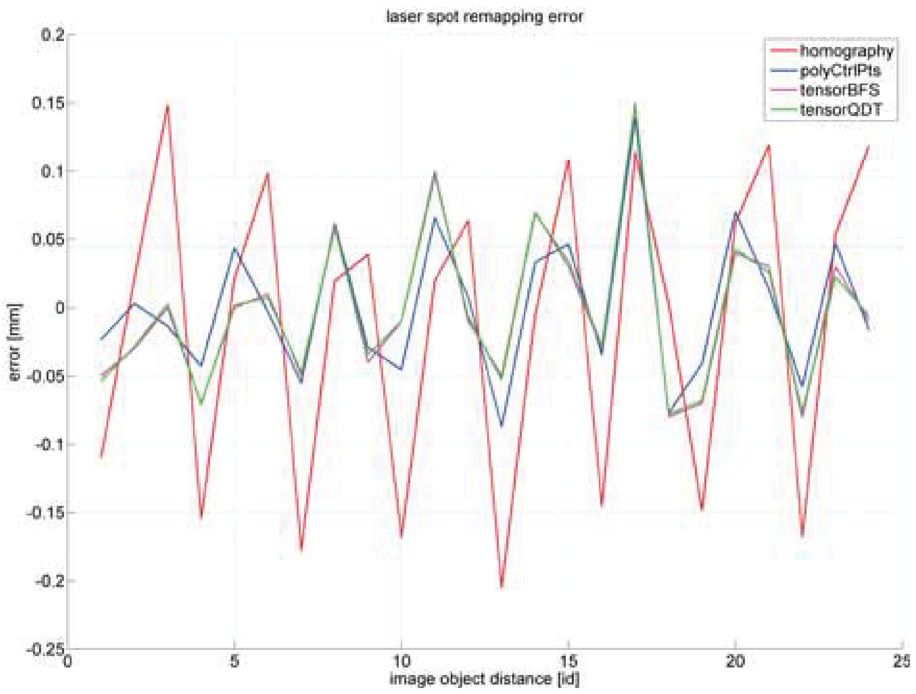


Figure E.8: Error in y dimension using laser spots and center of gravity.

TC#2: Gaussian Curve Extremum / Coordinate Mapping Error

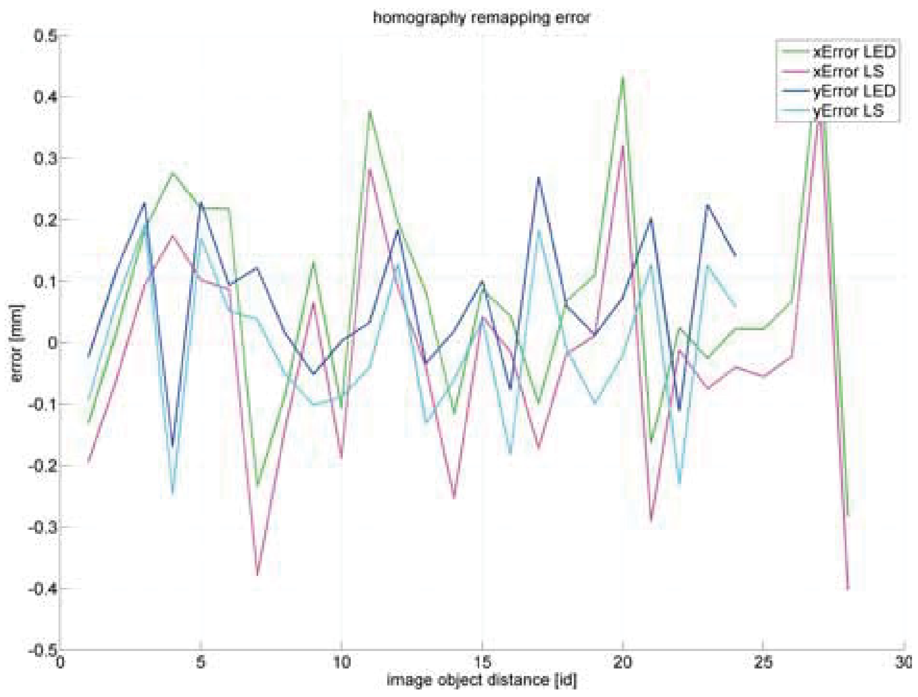


Figure E.9: Homography error utilizing Gaussian curve extremum.

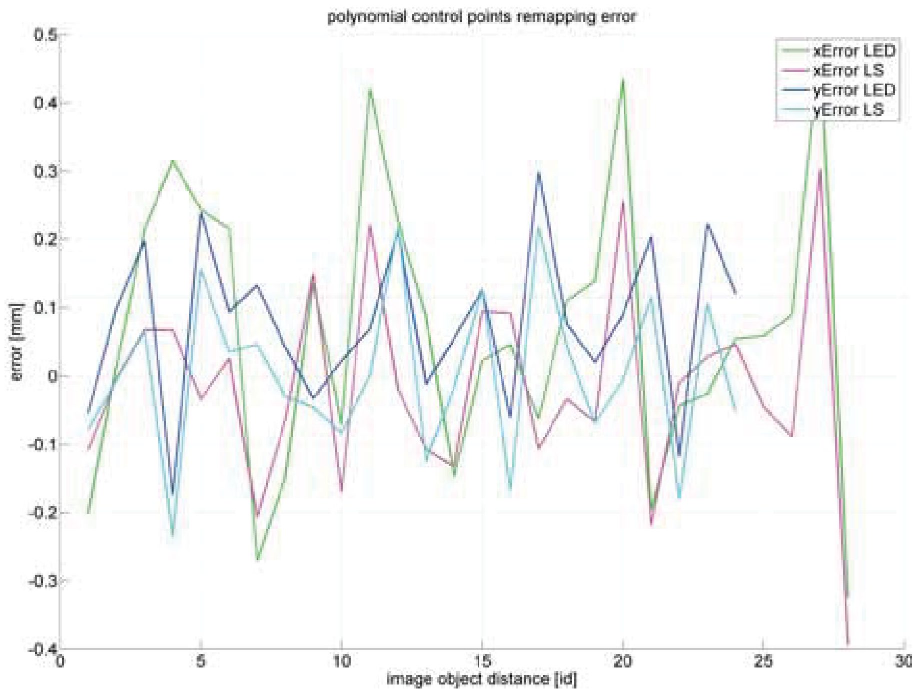


Figure E.10: PolyCtrlPts error utilizing Gaussian curve extremum.

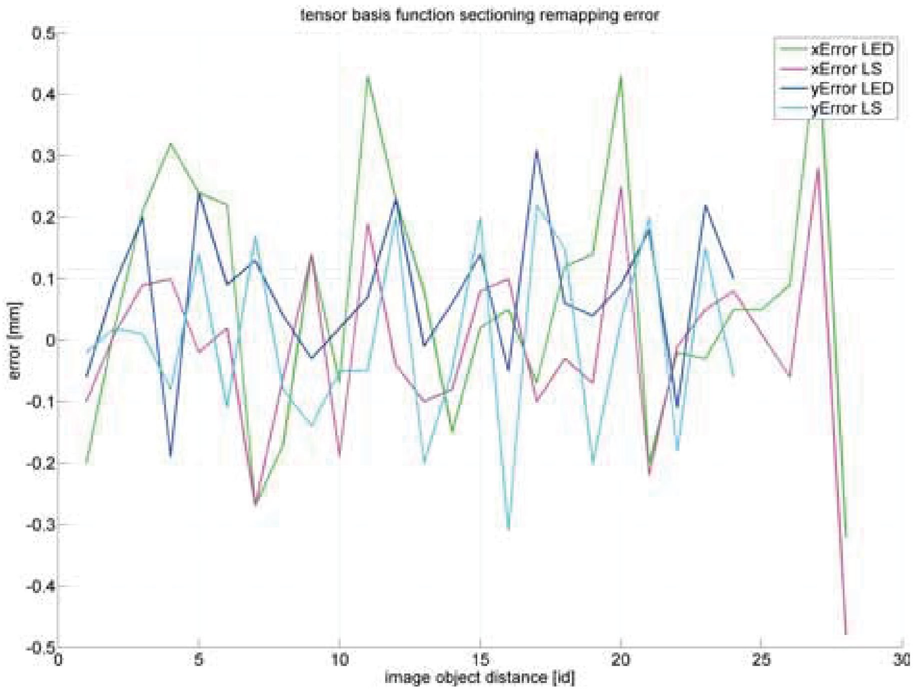


Figure E.11: TensorBFS error utilizing Gaussian curve extremum.

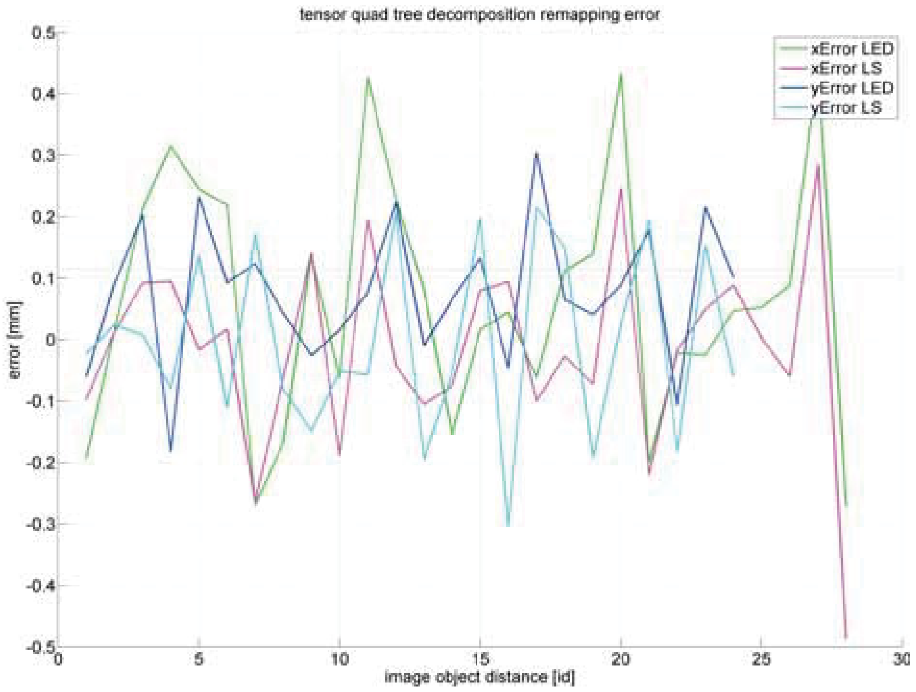


Figure E.12: TensorQDT error utilizing Gaussian curve extremum.

TC#2: Gaussian Curve Extremum / LEDs vs. Laser Spots Error

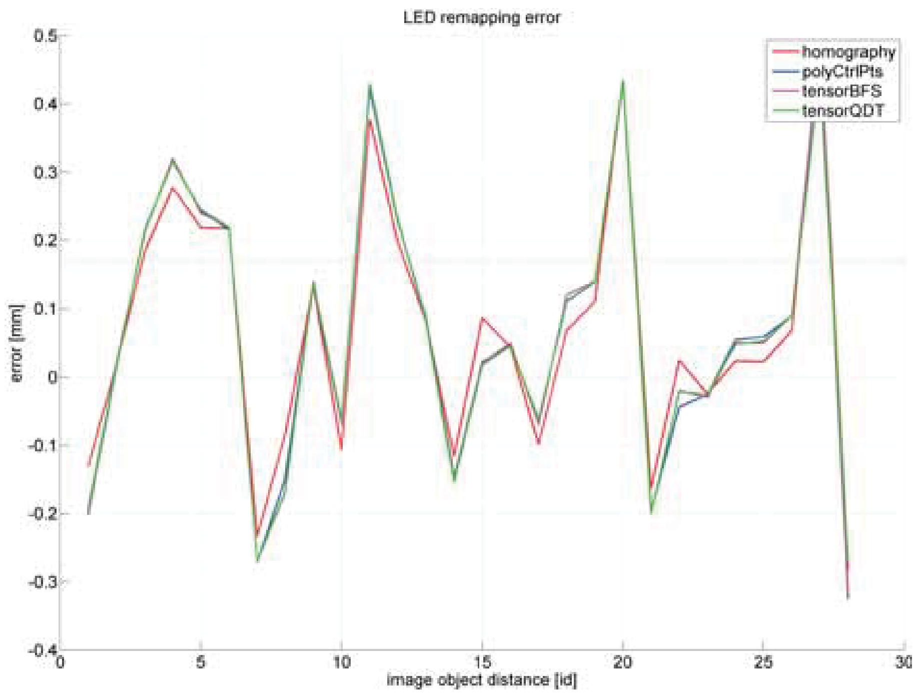


Figure E.13: Error in x dimension using LEDs and Gaussian curve extremum.

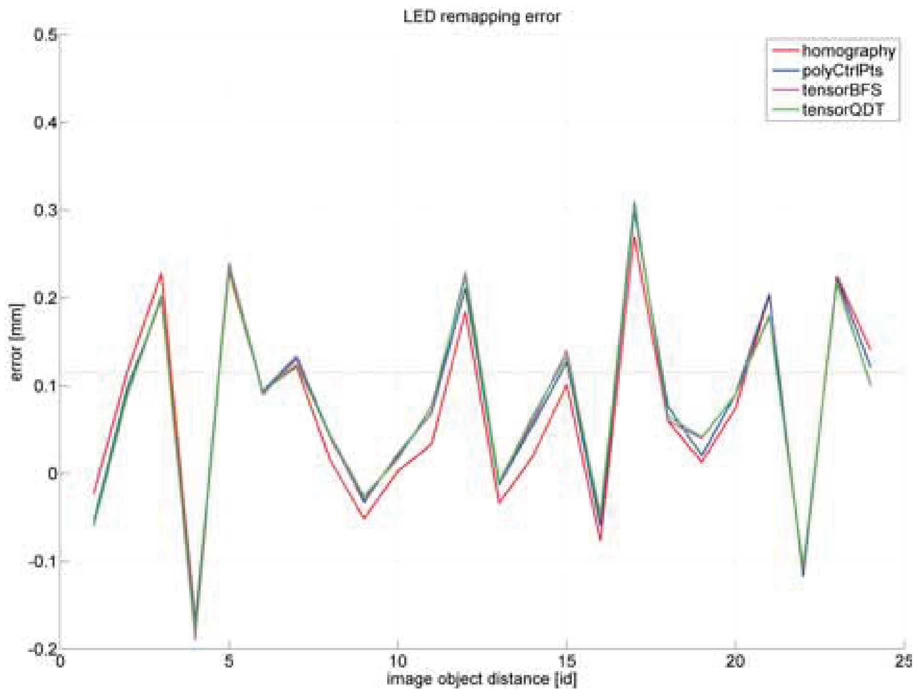


Figure E.14: Error in y dimension using LEDs and Gaussian curve extremum.

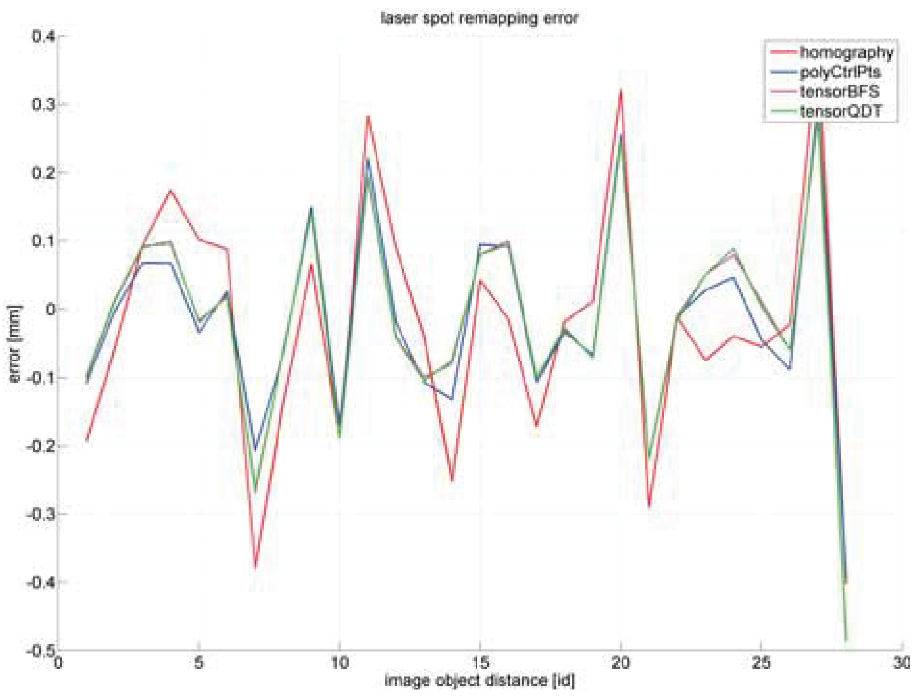


Figure E.15: Error in x dimension using laser spots and Gaussian curve extremum.

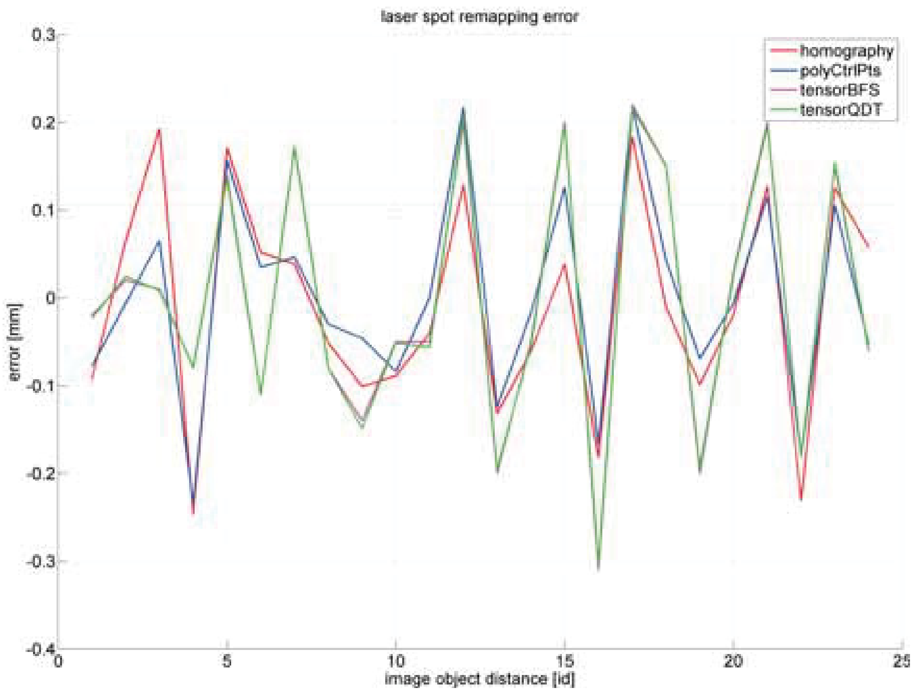


Figure E.16: Error in y dimension using laser spots and Gaussian curve extremum.

TC#3: Ellipse Approximation / Coordinate Mapping Error

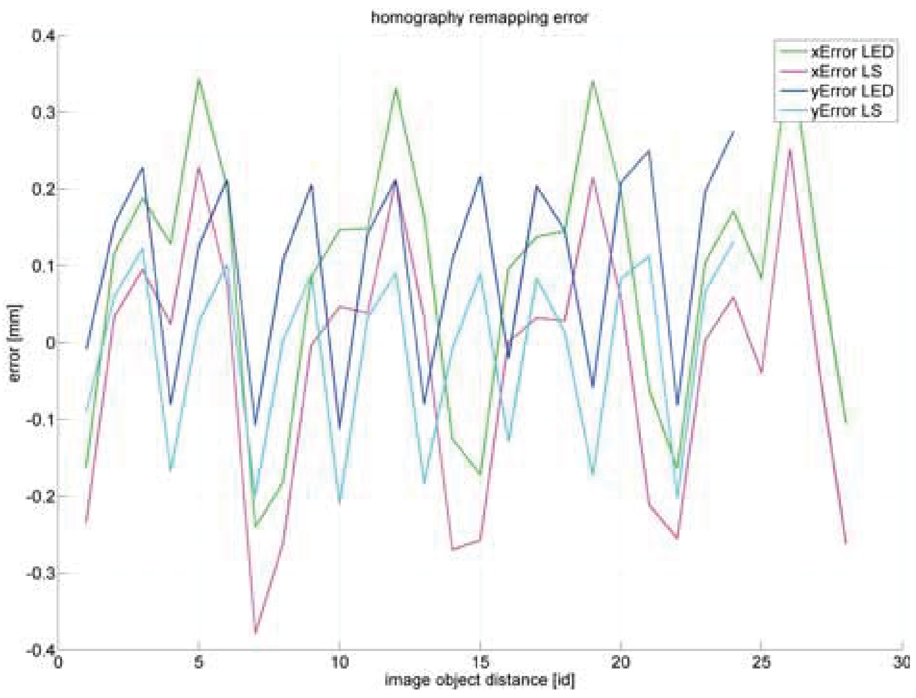


Figure E.17: Homography error utilizing ellipse approximation.

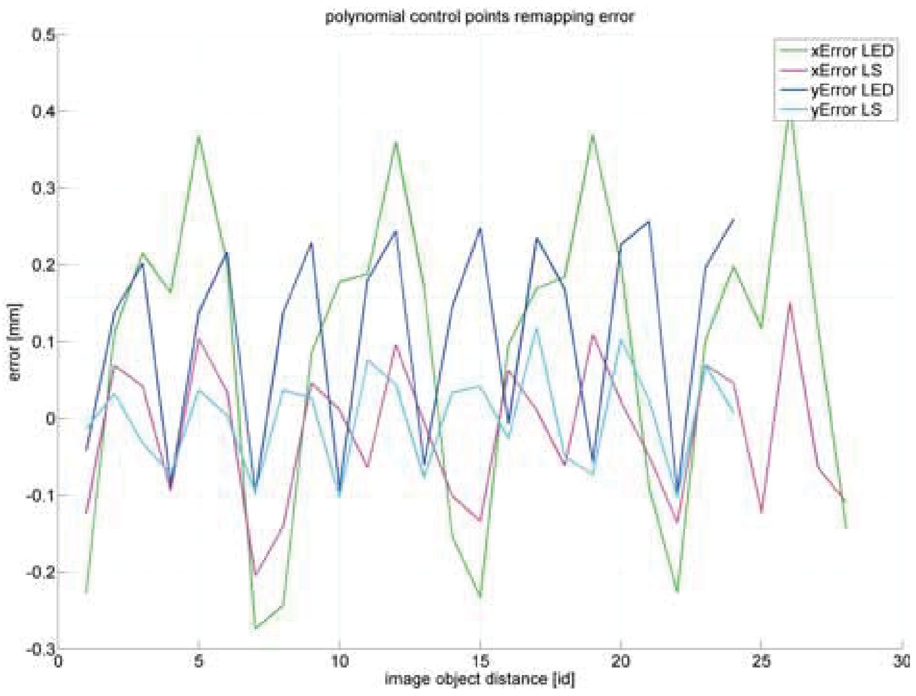


Figure E.18: PolyCtrlPts error utilizing ellipse approximation.

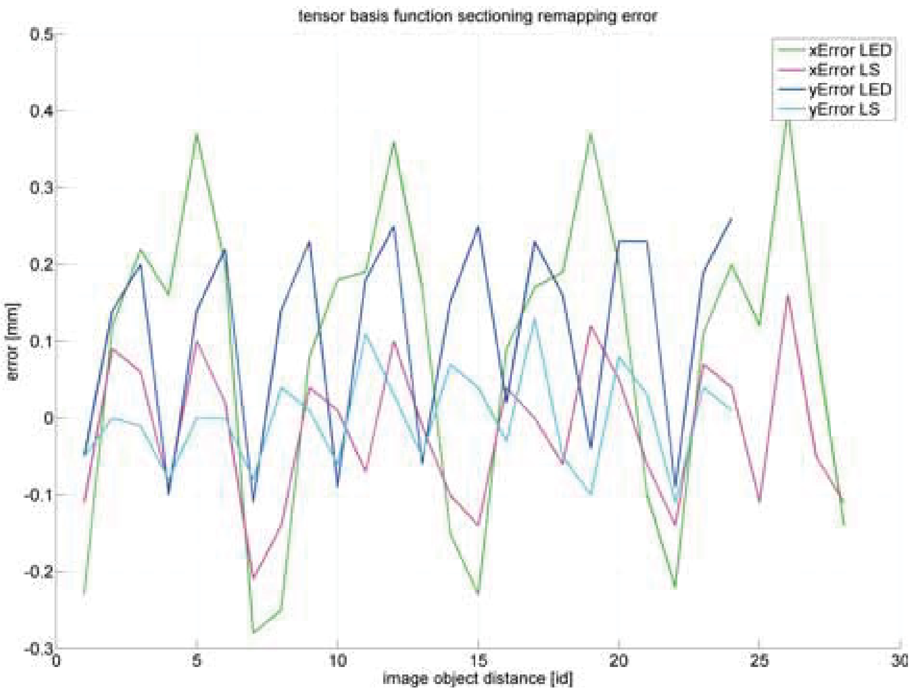


Figure E.19: TensorBFS error utilizing ellipse approximation.

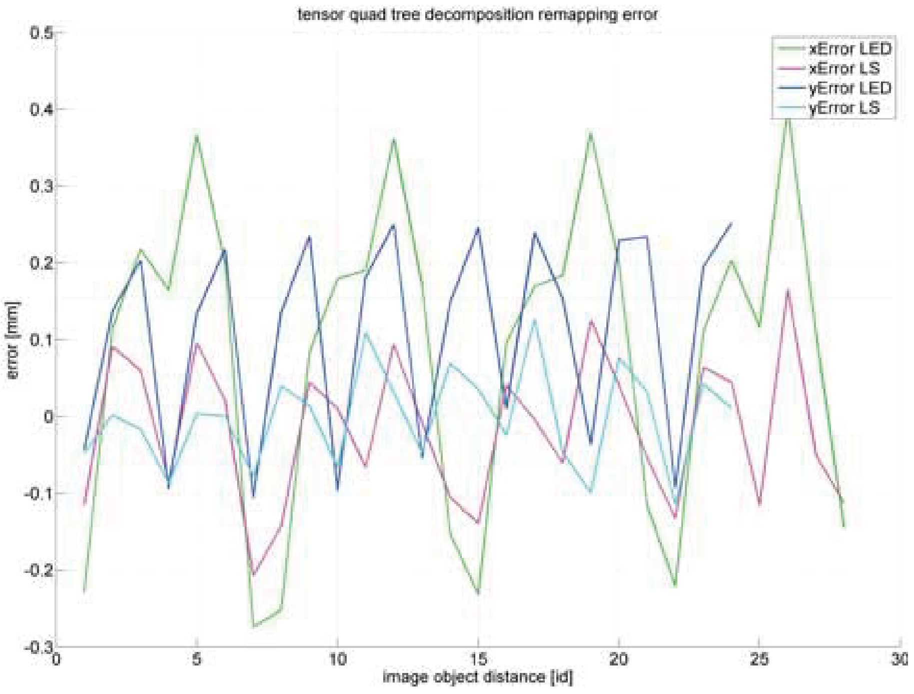


Figure E.20: TensorQDT error utilizing ellipse approximation.

TC#3: Ellipse Approximation / LEDs vs. Laser Spots Error

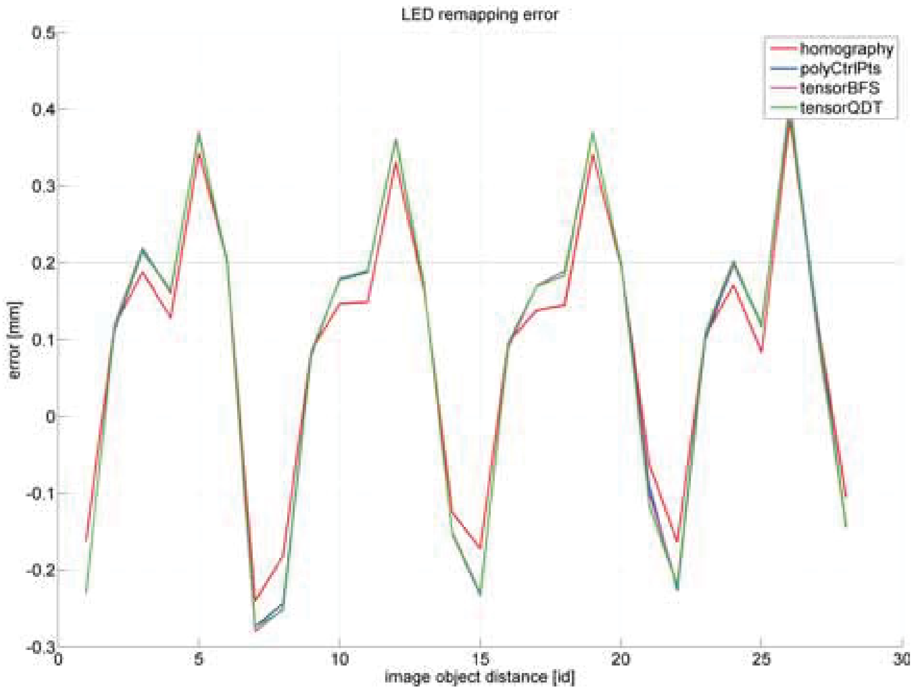


Figure E.21: Error in x dimension using LEDs and ellipse approximation.

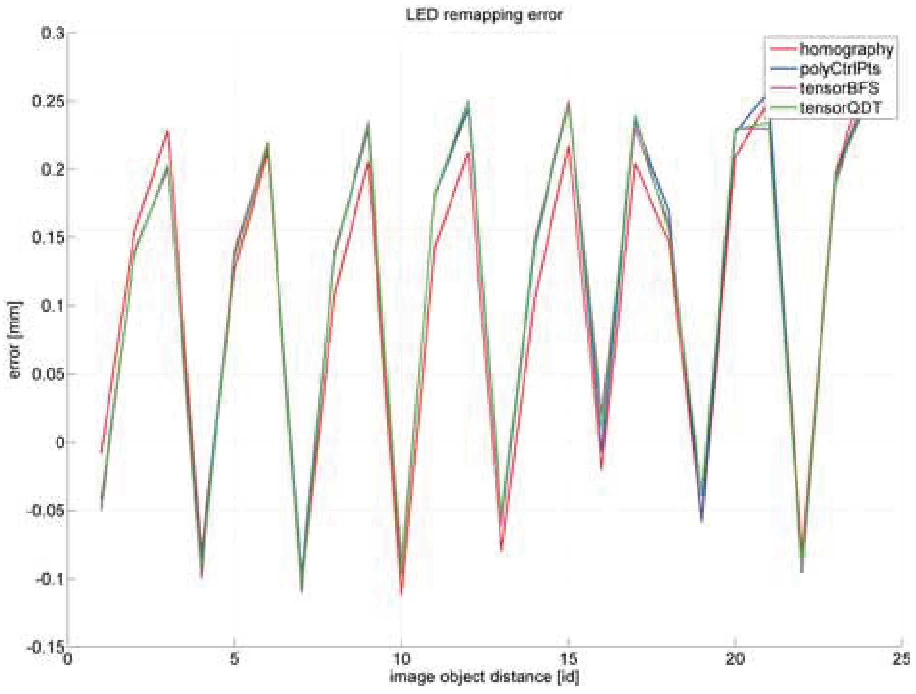


Figure E.22: Error in y dimension using LEDs and ellipse approximation.

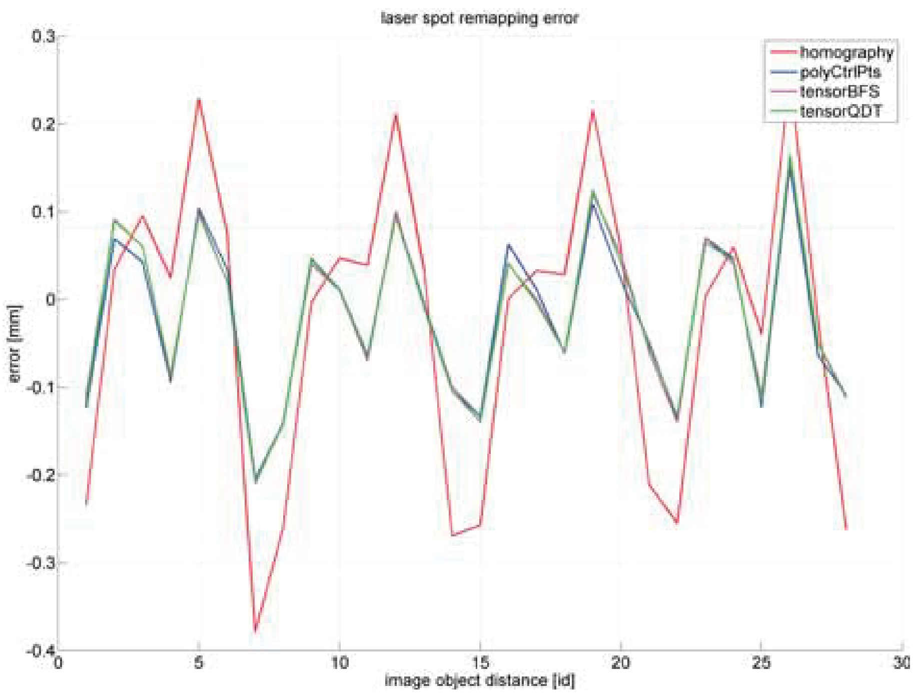


Figure E.23: Error in x dimension using laser spots and ellipse approximation.

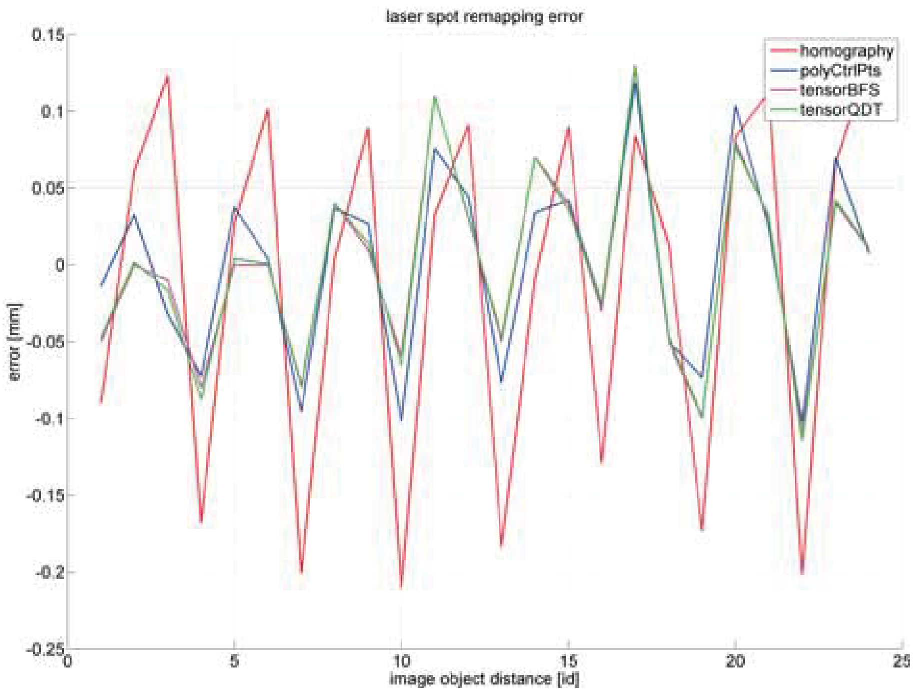


Figure E.24: Error in y dimension using laser spots and ellipse approximation.

TC#4: Circle Approximation / Coordinate Mapping Error

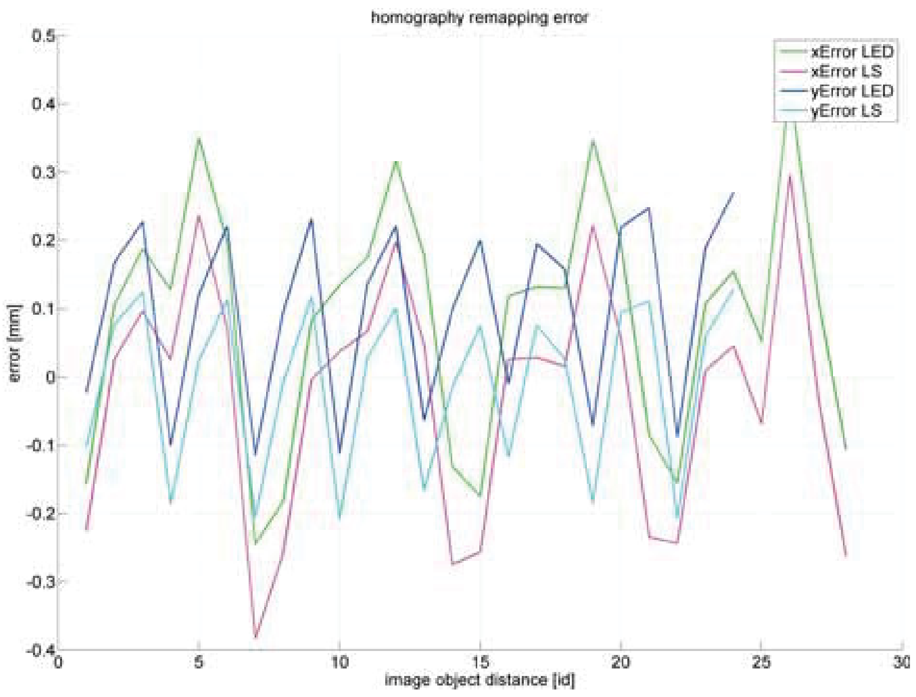


Figure E.25: Homography error utilizing circle approximation.

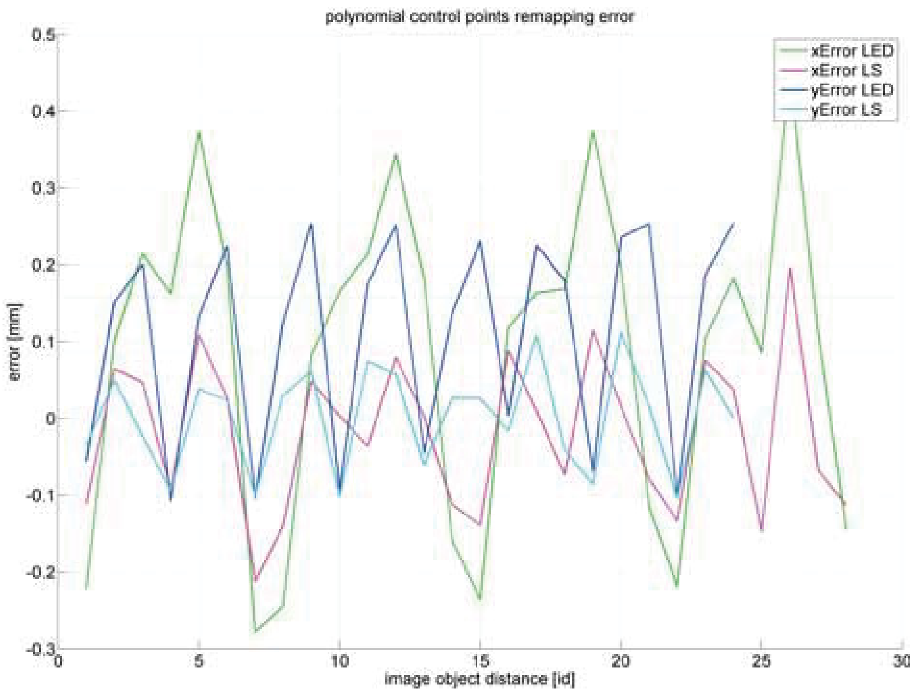


Figure E.26: PolyCtrlPts error utilizing circle approximation.

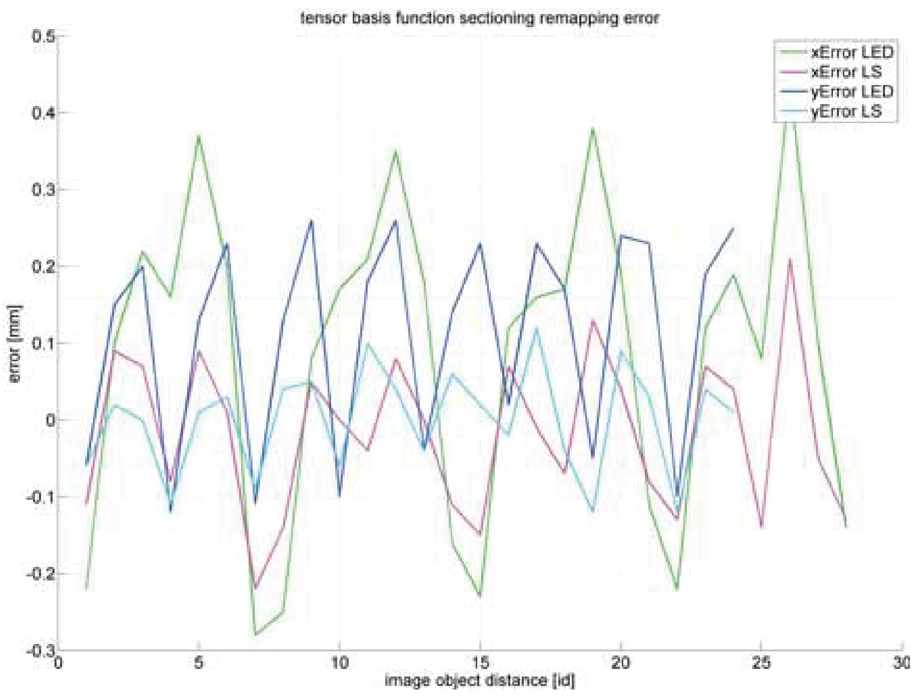


Figure E.27: TensorBFS error utilizing circle approximation.

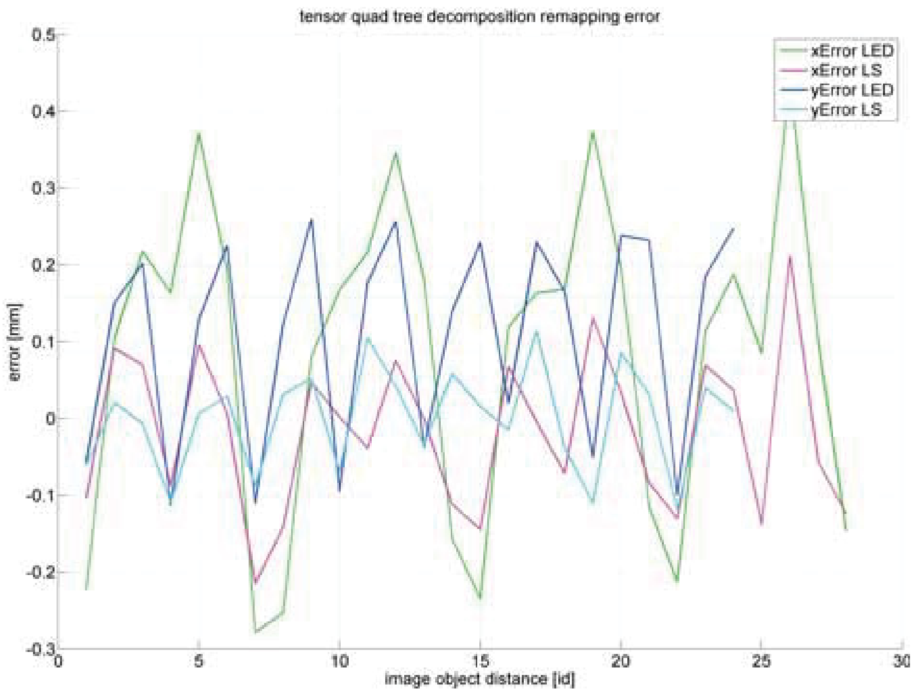


Figure E.28: TensorQDT error utilizing circle approximation.

TC#4: Circle Approximation / LEDs vs. Laser Spots Error

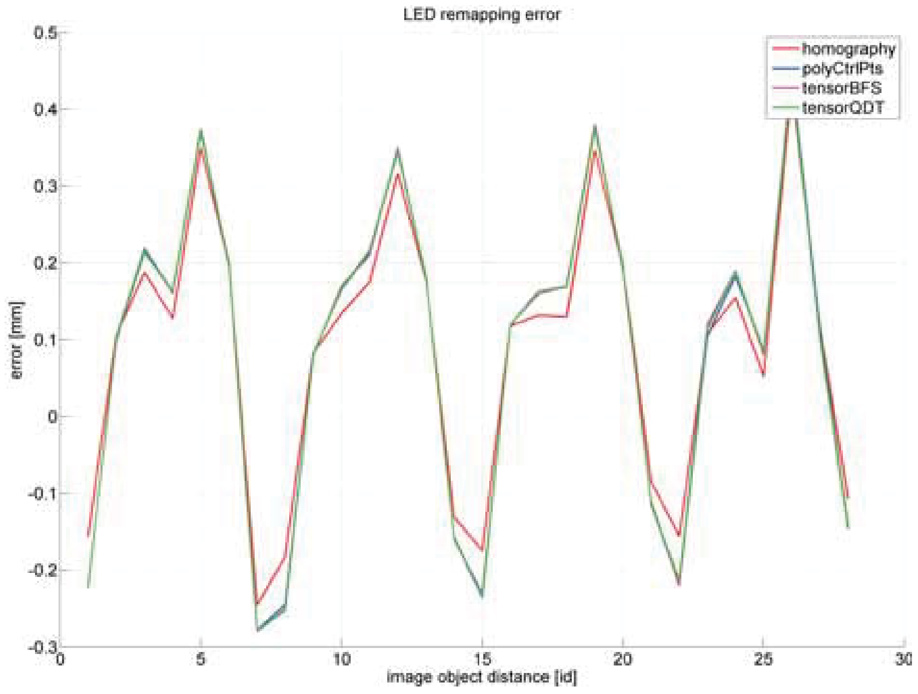


Figure E.29: Error in x dimension using LEDs and circle approximation.

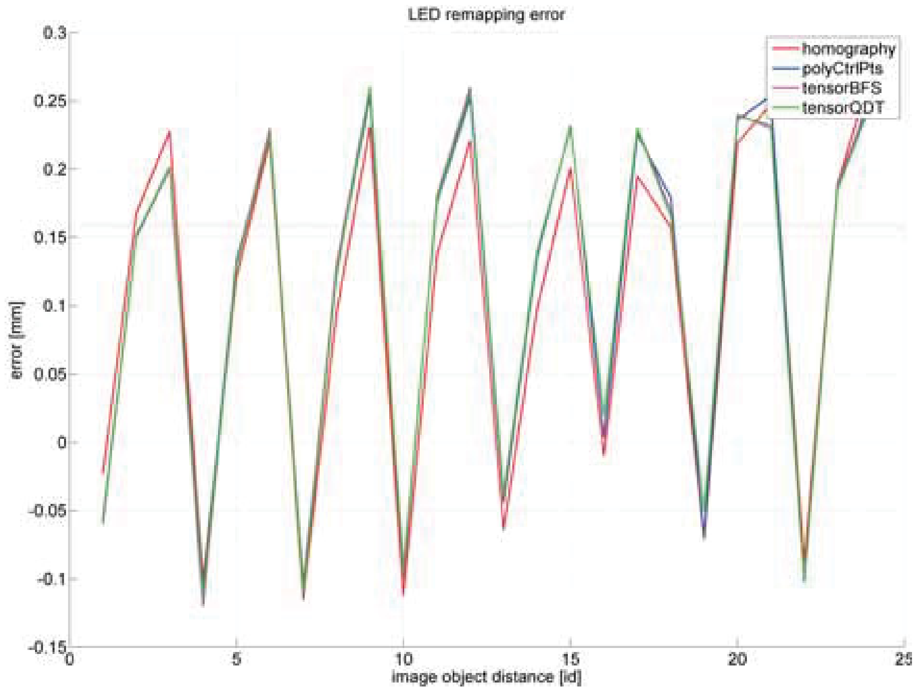


Figure E.30: Error in y dimension using LEDs and circle approximation.

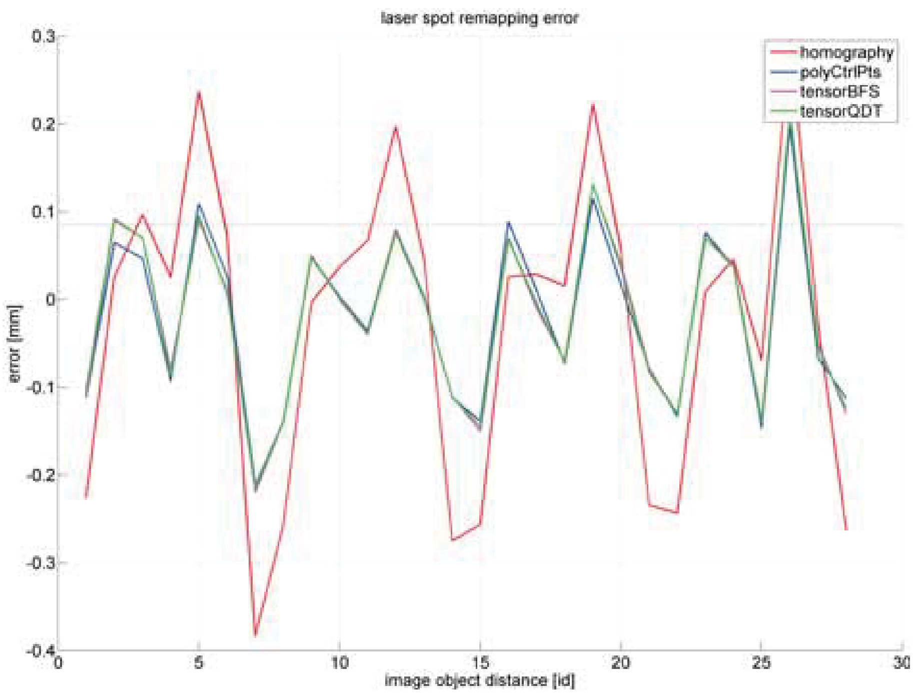


Figure E.31: Error in x dimension using laser spots and circle approximation.

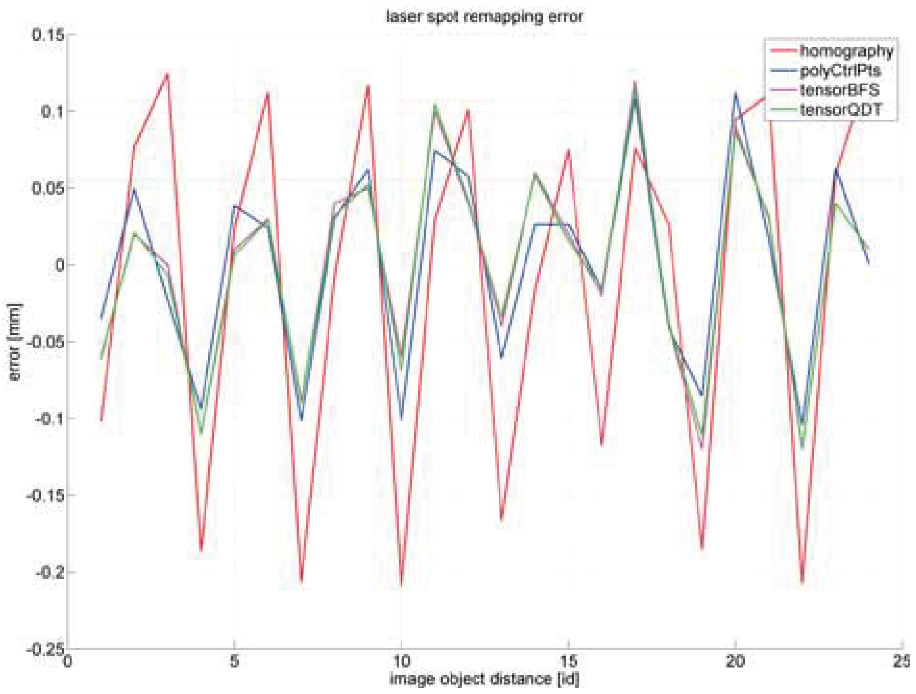


Figure E.32: Error in y dimension using laser spots and circle approximation.

List of Figures

List of Tables

List of Source Codes

Bibliography

[1] D. Arnold, A. Kuhn, K. Furmans, H. Isermann, and H. Tempelmeier. *VDI Handbuch Logistik*. Springer Verlag, Berlin, Heidelberg, 2008.

[2] A. Badshah, P. O'Leary, and M. Harker. Gram polynomial image decimation and its application to non-rigid registration. *Image Processing: Machine Vision Applications IV, SPIE*, 2011.

[3] A. Badshah, P. O'Leary, M. Harker, and C. Sallinger. Non-rigid registration for quality control of printed materials. *10th SPIE International Conference on Quality Control by Artificial Vision (QCAV)*, 2011.

[4] C. Demant, B. Streicher-Abel, and P. Waszkewitz. *Industrielle Bildverarbeitung*. Springer Verlag, Berlin, Heidelberg, 1998.

[5] DIN EN 13306:2001-09. *Begriffe der Instandhaltung*.

[6] DIN EN 31051:2003-06. *Grundlagen der Instandhaltung*.

[7] R. Figliola and D. Beasly. *Theory and Design for Mechanical Measruements*. John Wiley and Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1995.

[8] A. Goshtasby. Piecewise linear mapping functions for image registration. *Pattern Recognition*, 19, 1986.

[9] A. Goshtasby. Image registration by local approximation methods. *Image and Vision Computing*, 6, 1988.

[10] T. Gudehus. *Logistik: Grundlagen, Strategien, Anwendungen*. Springer Verlag, Heidelberg, Dordrecht, London, New York, 2010.

[11] M. Harker and P. O'Leary. Computation of homographies. *British Machine Vision Conference*, 2005.

[12] M. Harker, P. O'Leary, and P. Zsombor-Murray. Incremental matrix orthogonalization with an application to curve fitting. *Symposium on Electronic Imaging: Computational Imaging III*, 2005.

[13] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[14] R. Koether. *Taschenbuch der Logistik*. Carl Hanser Verlag, München, Wien, 2004.

[15] N. Koller. *Fully Automated Repair of Surface Flaws using an Artificial Vision Guided Robotic Grinder*. PhD thesis, University of Leoben, 2007.

[16] E. Kreyszig. *Statistische Methoden und ihr Anwendungen*. Vandenhoeck und Ruprecht, Göttingen, 1979.

[17] A. LaCruz. Accuracy evaluation of different centerline approximations of blood vessels. *Data Visualization*, 2004.

[18] K. Matyas. *Taschenbuch Instandhaltungslogistik*. Hanser Fachbuchverlag, München, Wien, 1999.

[19] D. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons, 2000.

[20] R. Ofner, P. O'Leary, and M. Leitern. A collection of algorithms for the determination of construction points in the measurement of 3d geometrics via light-sectioning. *WESIC, 2nd Workshop on European Scientific and Industrial Collaboration promoting: Advanced Technologies in Manufacturing*, 1999.

[21] P. O'Leary and M. Harker. An algebraic framework for discrete basis functions in computer vision. *IEEE Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[22] P. O'Leary and M. Harker. Discrete polynomial moments for real-time geometric surface inspection. *Journal of Electronic Imaging, SPIE*, 2009.

[23] P. O'Leary, M. Harker, and B. Mörtl. Discrete polynomial moments and the extraction of 3d embossed digits for recognition. *Journal of Electronic Imaging*, 2009.

[24] P. O'Leary, M. Harker, and P. Zsombor-Murray. Direct and least square fitting of coupled geometric objects for metric vision. *Vision, Image and Signal Processing, IEEE Proceedings-Publications*, 2005.

[25] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 1979.

[26] B. Rönz and H. Strohe. *Lexikon Statistik*. Gabler Verlag, 1994.

[27] D. Rumsey. *Statistik für Dummies*. Wiley-VCH Verlag GmbH, 2010.

[28] A. Weckenmann and B. Gawande. *Koordinatenmesstechnik*. Carl Hanser Verlag, München, Wien, 1999.