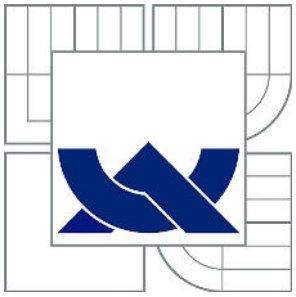
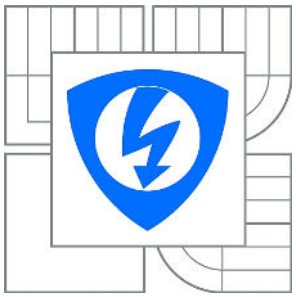


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

3D SCANNING WITH PROXIMITY PLANAR SCANNER

3D SKENOVÁNÍ POMOCÍ PROXIMITNÍHO PLANÁRNÍHO SKENERU

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

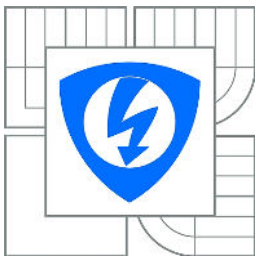
AUTOR PRÁCE — Bc. ADAM CHROMÝ

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. LUDĚK ŽALUD, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Adam Chromý	***ID:***	115187
Ročník:	2	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

3D skenování pomocí proximitního planárního skeneru

POKYNY PRO VYPRACOVÁNÍ:

- Seznamte se s předloženým planárním skenerem a vytvořte pro něj ovladač v prosředí .NET.
- Seznamte se s manipulátorem EPSON-C3 a vytvořte pro něj základní rozhraní sledování požadované trajektorie.
- Vytvořte algoritmy pro zpracování souboru naměřených bodů do jednotného souřadného systému.
- Imlementujte algoritmus pro generování povrchu z naměřeného souboru dat.
- Vytvořte vizualizační systém pro tato měření na platformě XNA.

DOPORUČENÁ LITERATURA:

H.R. Everett, Sensors for Mobile Robots, A K Peters/CRC Press, 1995, ISBN 978-1568810485

Termín zadání:	11.2.2013	***Termín odevzdání:***	20.5.2013

Vedoucí práce: doc. Ing. Luděk Žalud, Ph.D.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRACT

This thesis describes construction of scanning system providing three-dimensional models. Use of laser scanner mounted on robotic manipulator provides very flexible device capable building models of both tiny detailed structures and large object. It could be used in many various applications, especially in health care, where brings lot of advantages comparing to present scanning systems. Mechanical constitution, interfacing approaches, operating principles and calibration problems are described. This thesis also deals with visualizing of measured point clouds and its processing to form of shaded surface. Result of this work is ready-to-use device with software.

KEYWORDS

laser scanner, robotic manipulator, homogeneous transformations, calibration, surface generation from point cloud, trajectory planning, driver, 3D visualization, 3D scanner, robot control

ABSTRAKT

Tato práce popisuje konstrukci skenovacího systému pro tvorbu trojrozměrných modelů. Kombinace laserového scanneru a robotického manipulátoru tvoří velice flexibilní zařízení schopné snímat jak velké, tak malé a detailní objekty. Zařízení nachází uplatnění v mnoha aplikacích, zejména v lékařství, kde přináší řadu nesporných výhod proti stávajícím systémům. Práce popisuje mechanickou konstrukci zařízení, funkční principy a jeho kalibrační proceduru. Součástí práce je i software pro vizualizaci naměřených dat a jejich zpracování do podoby modelů se stínovaným povrchem. Výsledkem práce je funkční zařízení a rozsáhlý obslužný software.

KLÍČOVÁ SLOVA

laserový scanner, robotický manipulátor, homogenní transformace, kalibrace, generování povrchu z mraku bodů, plánování trajektorie, ovladač, vizualizace 3D dat, 3D scanner, ovládání robotu

CHROMÝ, Adam *3D Scanning with Proximity Planar Scanner*: master's thesis. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Control and Instrumentation, 2013. 81 p. Supervised by doc. Ing. Luděk Žalud, Ph.D.

DECLARATION

I declare that I have elaborated my master's thesis on the theme of "3D Scanning with Proximity Planar Scanner" independently, under the supervision of the master's thesis supervisor and with the use of technical literature and other sources of information which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the master's thesis I furthermore declare that, concerning the creation of this master's thesis, master's thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal copyright and I am fully aware of the consequences in the case of breaking Regulation § 11 and the following of the Copyright Act No 121/2000 Vol., including the possible consequences of criminal law resulted from Regulation § 152 of Criminal Act No 140/1961 Vol.

Brno ....

(author's signature)

ACKNOWLEDGEMENT

Special acknowledgement belongs to doc. Ing. Luděk Žalud, Ph.D. for the sake of granting access to the domain of robotics, for granting all resources wanted to achieve aims of this thesis and especially for cheerful approach, what I consider very valuable.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Capturing three-dimensional models has become more and more important during last years, due to several reasons. One is rapid development of object reproducing using 3D printers, which requires devices and algorithms for creating models of objects to be cloned. Other reason is increasing necessity of storing visual information in resistant form, what empowers detection of tiny changes during the time.

This thesis describes constitution and basic principles of three-dimensional scanning system based on precise robotic manipulator and laser scanner. Programmable scanning trajectory in six degrees of freedom, together with high precision of manipulator's movement, makes this system very flexible. It can be used for creating models of various objects with various complexity and size. Replaceability of laser scanner provides possibility of scanning both tiny and large structures.

The main purpose of this scanning system is capturing models of parts of human body. These models are used for example in rehabilitation process after serious injuries or after invasive surgeries. Presently, building such models for these purposes is made by very uneconomical way: there is a detailed model created by Magnetic Resonance Imaging (MRI), which contains lot of information about inner structures, which are useless for our purposes – just information about outer surface is extracted. This MRI device is very expensive as well as its operation, creating these models takes a long time and by building our models, we also blocks patients who need MRI images for more important reason.

Scanning system described in this thesis is much cheaper as well as its operation is cheaper and creating of three-dimensional models is faster.

First chapter introduces used devices, such as robotic manipulator and laser scanners, provides brief information about its parameters, operating principles, advantages and disadvantages.

Second chapter describes 3D scanner itself – its mechanical constitution, operating principles and interfacing approaches in order to control the system in real time. For this purpose, drivers for both used scanners were implemented and tested. Main part of this chapter fulfils homogeneous transformations, which provides essential knowledge for device work and measured data understanding.

Third part deals with calibration procedure. It is the biggest part of my work and the most difficult part of this thesis. Although it is not mentioned in thesis assignment, it is needful and makes entire 3D scanner working. Proposed calibration procedure is accurate and relatively fast and is independent on calibration object accuracy, what makes it significantly cheaper.

Fourth chapter describes realization of higher-level software, which includes scanning trajectory planning tools, surface generating module and graphic user interface.

This makes the whole system intuitive and easily usable.

Last chapter describes software for visualization of final 3D models and shows some results in form of scanned models.

The result of this thesis is ready-to-use device with wide range of usage together with large pack of software for device controlling and model visualization.

1 OVERVIEW OF USED DEVICES

This chapter provides brief general informations about robotic manipulators and laser scanners, which both are used in this thesis in order to build up a 3D scanner. It provides also more detailed specifications of particular products from this classes, which are applied in this solution.

1.1 General Informations about Manipulators

Robotic manipulators are devices primarily used for moving objects in order to avoid human contact with this object, to replace manpower, to ease man's work or to increase its accuracy. There are many advantages resulting from using manipulators such as decrease of operating costs, increase of product quality, increase of producing capacity, etc. Because of it, domain of robotic manipulators is rapidly growing and the development of new devices is very fast.

Generally, there are two main parts which manipulators consists from [1]:

- **Arm and Body** - The arm and body of a robot are used to move and position objects within a work envelope. They usually consists from three joints connected by links.
- **Wrist** - The wrist is used to orient the objects at the work location. It is usually formed by two or three compact joints.

Each joint is mechanical part empowering relative movement of links in between the joint is. There are several types of joints, from which most used are linear, rotary or twisting. The movement is initiated by motors usually based on electric or hydraulic servomechanism.

The arm-and-body section of robotic manipulators is usually built on the one of following configurations [1]. Each of these constitutions provides a different work envelope and is suited for different applications:

- **Gantry** – These robots have linear joints and are mounted overhead. They are also called Cartesian and rectilinear robots. (fig. 1.1a)
- **Cylindrical** – There are linear joints that connect to a rotary base joint. This anatomy is called cylindrical for the shape of its work envelope. (fig. 1.1b)
- **Polar** – The base joint of a polar robot is rotary or twisting and the other joints are a combination of rotary and linear types. The work space created by this configuration is spherical. (fig. 1.1c)
- **Jointed-Arm** – This is the most popular industrial robotic configuration. The arm connects with a twisting joint, and the links within it are connected with rotary joints. It is also called an articulated robot. (fig. 1.1d)

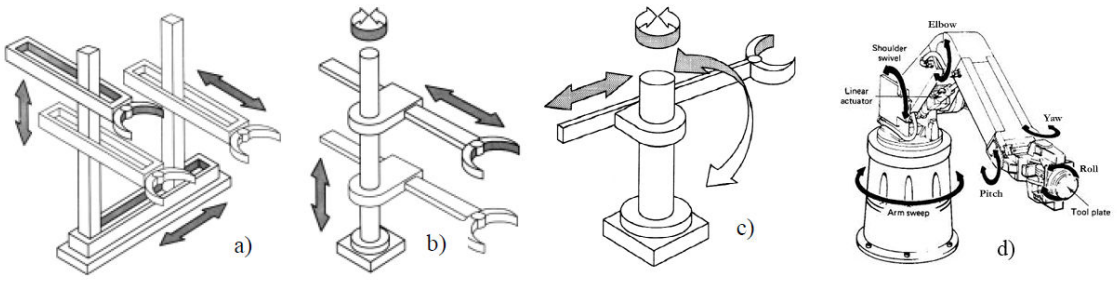


Fig. 1.1: Standard manipulator's anatomies: a) Gantry, b) Cylindrical, c) Polar, d) Jointed-Arm [1][2]

There are many types of robotic manipulators varying in several parameters. Because there is no universal manipulator which is the best in every parameter (or if it is, it must be very expensive, what is limiting factor in almost every application) it's important to get familiar with basic parameters of manipulator, and according to our purposes, choose the best fitting device. The most important parameters are [4]:

- **Total number of joints** – very important parameter of manipulator, because only 6 joints and more can provide reachability of any point inside the range of robotic manipulator. Otherwise, the operating range need not be continuous and movements are limited.
- **Operating envelope** – area, which is reachable by manipulator's end-point. Commonly defined by dimensions or volume and shape. Could be continuous or discrete. Most widely used manipulators principally have continuous work envelope with almost spherical shape.
- **Locating accuracy and repeatability** – accuracy of end-point locating. Is defined as area around desired position in which manufacturer declares that end-point is located for sure. Together with repeatability of locating determines manufacturer's precision.
- **Kinematic characteristics** – defines maximal speed and acceleration of manipulator's end-point. These parameters in fact influences global speed of entire robotic system.
- **Dynamic characteristics** – maximal forces and moments permitted inside parts of robot. Together with actual operating speed and acceleration influences maximal payload.
- **Payload** – maximal load of end-point or other parts of robot. Generally, payload value varies in full speed mode, half speed mode, 30% speed mode, etc.

- **Failure rate** – mostly defined as average time of trouble-free operation. Determines reliability of system.
- **Price and maintain costs** – purchase costs are usually most decisive factors when choosing particular model of manipulator. Generally speaking, requirement of the best values of parameters above, means higher purchasing expenses. But also maintain costs is very important parameter which should be very carefully considered, although it's commonly omitted in balance. It's important to notify, that the cheapest solutions often means high maintain expenses and low reliability, which finally stands for higher costs.

1.2 Robotic Manipulator EPSON C3

Laboratory of Telepresence and Robotics disposes with robotic manipulator EPSON C3, so as a result of this fact, this particular model was applied in my project. This section presents important parameters of this model and summarizes if it is proper device according to the aim of this project.

Fig. 1.2: Epson C3 manipulator [3]

1.2.1 Mechanical Design of Manipulator

Epson C3 is a small, compact, 6-joint robotic manipulator. Mechanical anatomy of this robot follows from Jointed-Arm kinematic concept. Arm movements are provided by 3 rotary joints, other 3 joints allows wrist movements. Robot's constitution is displayed on fig. 1.3.

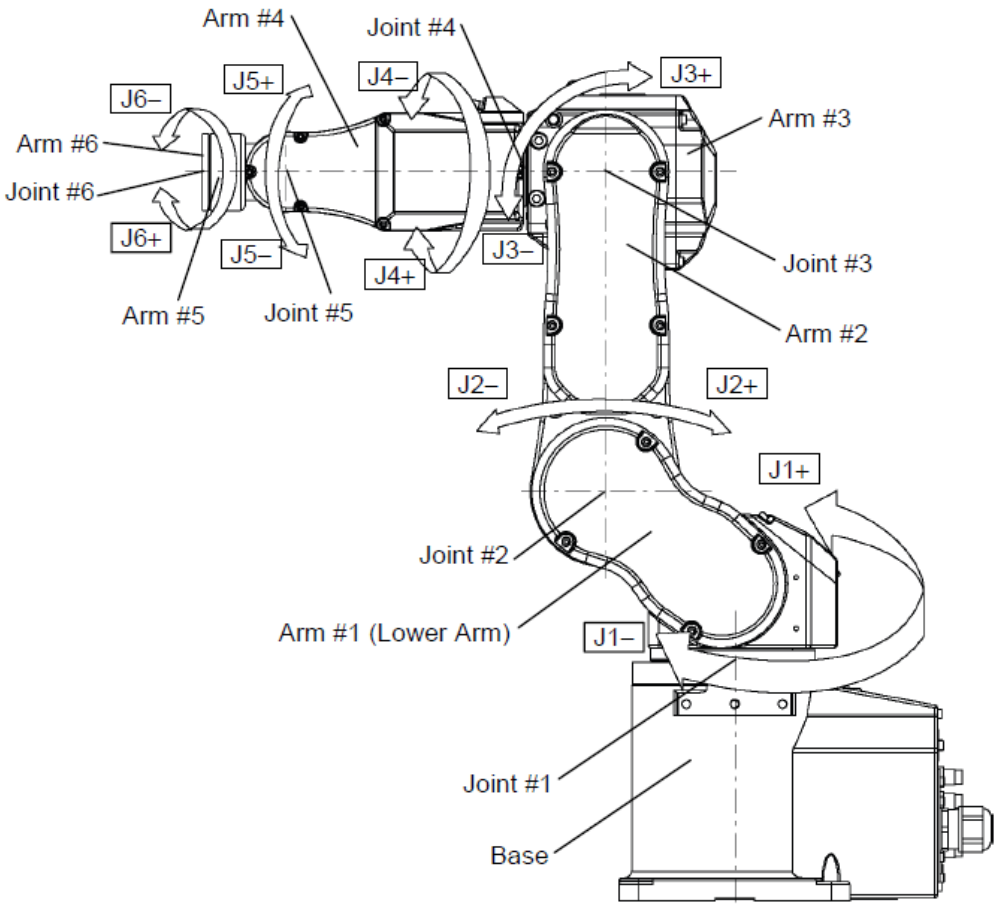


Fig. 1.3: Epson C3 manipulator's anatomy [5]

There is a signal LED lamp placed on the Arm #4, which indicates that motors are powered on. In cases, when motors are powered off or in Emergency Stop state, LED lamps is off and all joints are blocked by electromagnetic brakes which improves safety of robot's operation.

The manipulator is equipped with user wires and pneumatic tubes, which allows connecting of custom devices to the end-point. There are 4 pneumatic tubes and 9 wires, which are terminated on both ends by standard 9-pin D-sub connector. The maximal allowed voltage in user cables is AC/DC 30 V and maximal allowable current is 1 A.

1.2.2 Specifications

Locating Accuracy and Repeatability

Manufacturer declares, that repeatability of end-point locating is $\pm 0,02\ mm$ [5]. This value means, that this robot is one of the most accurate robots available on the market.

Positioning of joints is controlled by simple controller which input is actual rotation of each joint measured by incremental sensors. Accuracy of joint positioning is defined by its resolution which is displayed in tab. 1.1. It also refers to the class of today's best products.

Tab. 1.1: Resolution and maximal operating speed of each joint motor [5]

Joint number	**Positioning accuracy**	**Maximal operating speed**
Joint #1	$\pm 0,00000429$ °	450 °/s
Joint #2	$\pm 0,00000429$ °	450 °/s
Joint #3	$\pm 0,00000490$ °	514 °/s
Joint #4	$\pm 0,00000531$ °	553 °/s
Joint #5	$\pm 0,00000524$ °	553 °/s
Joint #6	$\pm 0,00000686$ °	720 °/s

Kinematic Characteristics

Actuation of joint is realized with fast AC servomotors, which power is transported to joint by gearwheels. In case of 1 kg payload, the average cycle time is 0,37 s for moving 300 mm away and back. Maximal speed depends on joints which must be moved in order to reach desired position. Particular speed for each joint actuator is summarized in tab. 1.1.

Dynamic Characteristics

Allowable moment and moment of inertia is defined for each arm in tab 1.2. Controlling unit automatically computes maximal acceleration for each joint according to these values and according to entered weight of load and its moment of inertia. It protects robot from oversize forces and optimizes movements of manipulator.

Payload

Normal operating payload is 1 kg. In this case, the maximal acceleration and maximal speed is reachable. Maximal payload is 3 kg for table-top or wall mounted manipulator, in case of ceiling mounting the maximal payload is 5 kg. When operating

Tab. 1.2: Dynamic characteristics for each arm [5]

Arm num.	**Allowable moment**	**$GD^2/4$ Allowable moment of inertia**
Arm #4	$4,41\ Nm\ (0,45\ kgfm)$	$0,15\ kg/m^2$
Arm #5	$4,41\ Nm\ (0,45\ kgfm)$	$0,15\ kg/m^2$
Arm #6	$2,29\ Nm\ (0,30\ kgfm)$	$0,10\ kg/m^2$

with payload close to this upper limit, maximal acceleration values are automatically reduced down by controlling unit in order to avoid damages caused by oversize moments and forces of the robotic manipulator.

Controlling Unit

Actuators of the manipulator are controlled by universal controlling unit RC180, which is common for all robotic products of EPSON company. It is computing device very similar to PLC, which has integrated number of abilities, which could be useful in ordinary industry processes. Unit programming could be performed via several communication interfaces such as Ethernet, RS-232 or USB Mass Storage. These interfaces could be used also for communication during manipulator's operation – for example for selecting predefined sequences, starting, terminating, etc, but also for sending user data which are processed by programme running inside controlling unit. This ability is very important for our project, because we want to control movements of manipulator in real time by external computer.

Motion Range

EPSON C3 manipulator has joints with wide motion range, as shown in tab. 1.3. Wide motion range of joints together with slim body (body volume is only $\frac{1}{44}$ of its motion range) allows flexible work in wide operation area as shown on fig. 1.4.

Tab. 1.3: Motion ranges of each joint [5]

Joint number	**Minimal rotation**	**Maximal rotation**
Joint #1	$-170°$	$+170°$
Joint #2	$-160°$	$+65°$
Joint #3	$-51°$	$+225°$
Joint #4	$-200°$	$+200°$
Joint #5	$-135°$	$+135°$
Joint #6	$-360°$	$+360°$

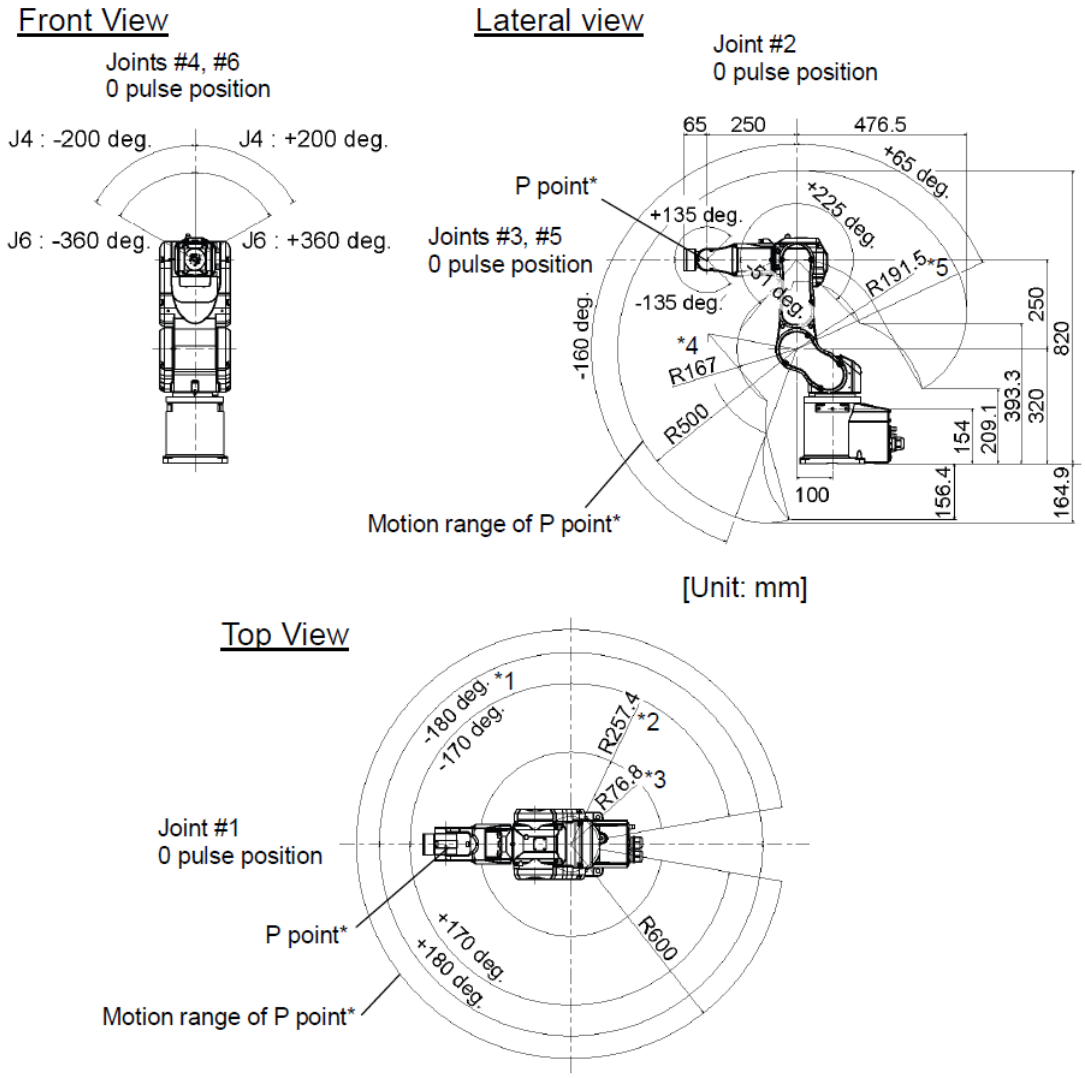


Fig. 1.4: Manipulator's motion range [5]

1.3 General Informations about Laser Scanners

Laser scanner is a contact-less measuring system, which uses controlled steering of laser beam followed by distance measurement in order to construct profile of scene in device's angle of view [6]. The output data from laser scanner is a set of distances between scanner and the nearest object in appropriate angle of view.

Basic laser scanner involves a laser range finder based on any principle described in section 1.3.1. Its laser beam is reflected by a rotating mirror gathering distance measurements at specified angle intervals.

Reason to use of laser instead of any other signal is because of its important

advantages:

- Laser provides concentrated narrow beam with very low deviation. It allows measuring distance to very tiny objects, such a depth of very narrow holes, what is not possible by ultrasonic or radio beams.
- Laser beam is monochromatic, what simplifies measurement processing.
- Laser has a high intensity which is nearly not diffusing, so it is not as much attenuated by environment as other mediums. It allows performing of long-distance measurements.

1.3.1 Measuring Principles

There are 4 most common measuring principles implemented in laser range finders and laser scanners [6]. Each of these methods has its advantages and disadvantages, so they are used in different devices for different purposes.

Time Of Flight Method

This method is based on a very simple principle: there is a generator inside the laser scanner, which is able to generate very tiny pulses of laser beam . These pulses travel through an environment to the object, where are reflected and travel back to the detector. Time difference between pulse generation and its recognition at detector is proportional to the distance of object (fig. 1.5).

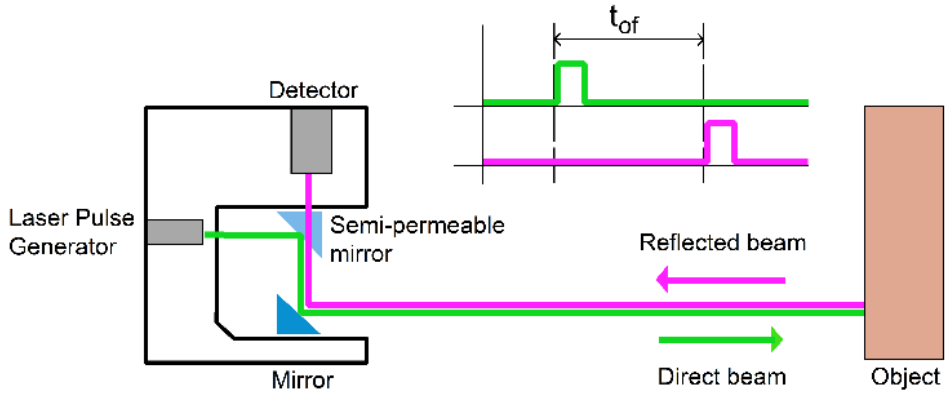


Fig. 1.5: Time-of-flight measuring principle

Although this principle is really simple, its realization is quite hard. Let's imagine, that distance to the object is given by the equation:

$$d_{obj} = \frac{1}{2} c \cdot t_{of} \tag{1.1}$$

where c is speed of light $[m.s^{-1}]$ and t_{of} is time of laser beam's flight $[s]$.

If we consider a distance of 1 mm, what is ordinary required resolution in many applications, by solving the equation 1.1, we can enumerate the required resolution of time-measuring unit of laser scanner:

$$t_{of} = \frac{2 \cdot d_{obj}}{c} = \frac{2 \cdot 1 \cdot 10^{-3}}{2,99 \cdot 10^{6}} = 6,69 \cdot 10^{-12} = 6,69 \text{ ps} \tag{1.2}$$

This implies, that we must have integrated a very precise time-measuring unit with resolution of picoseconds, if we want to reach this resolution of distance measuring.

The advantage of time-of-flight devices is that they are able to measure very long distances, on the order of kilometres, so they are suitable for scanning large objects like buildings or landscape profile. They are also independent on changes of temperature, because of compactness.

The disadvantage of time-of-flight devices is their accuracy, as was mentioned above. Due to the high speed of light, the time measuring is complicated and the accuracy of the distance measurement is relatively low, on the order of millimetres [6].

Triangulation Method

The principle of triangulation method is shown on figure 1.6. There is a laser beam generator, which produces beam which is consequently reflected by object. Reflected beam is concentrated by lens and reaches the optical position detector, for example PSD or CCD. There is a mark detected by position detector, which placing corresponds to distance of object.

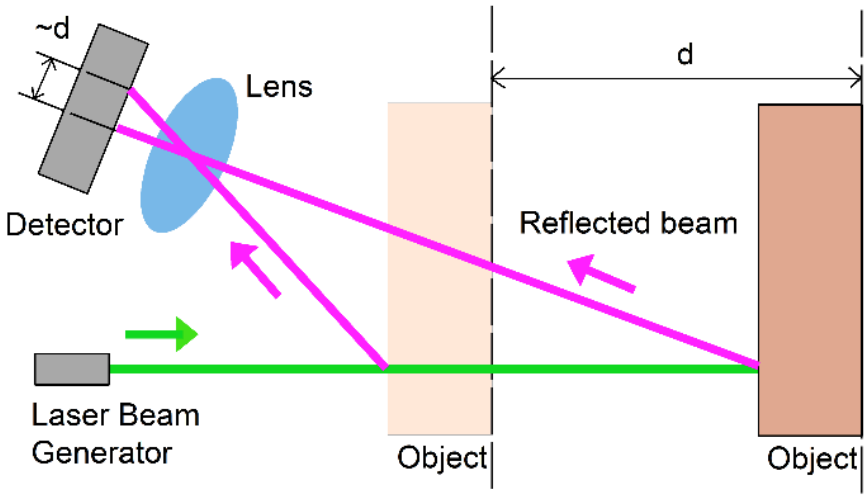


Fig. 1.6: Triangulation measuring principle

Triangulation-based devices are exactly the opposite according to time-of-flight type. They have a limited range of few meters, but their accuracy is relatively high. The accuracy of triangulation device is on the order of tens of micrometers, so they are suitable for scanning tiny structures like a bark of trees, etc.

Another disadvantage of this principle is a dependency on changes in temperature, what could be caused for example by strong sunlight on one side of the device. Link between transceiver and receiver will expand and slowly distort the scanned data. Some more developed laser scanners have level compensators to counteract any movement of the link during the scan process [6].

Interferometric Method

Principle of interferometric method is in interference of two beams which one of them passed longer way than second one. As shown on figure 1.7, generated laser beam reaches the semi-permeable mirror, where is split into two beams. First beam goes though the mirror, then is reflected by object and consequently by semi-permeable mirror and finally reaches the detector. Second beam is reflected by semi-permeable mirror, then goes through the compensator, consequently is reflected by mirror in reference distance, then goes once again through compensator, through mirror and finally reaches the detector.

The compensation block is placed into the way of second beam because the first beam goes through the glass 3 times and second one only once.

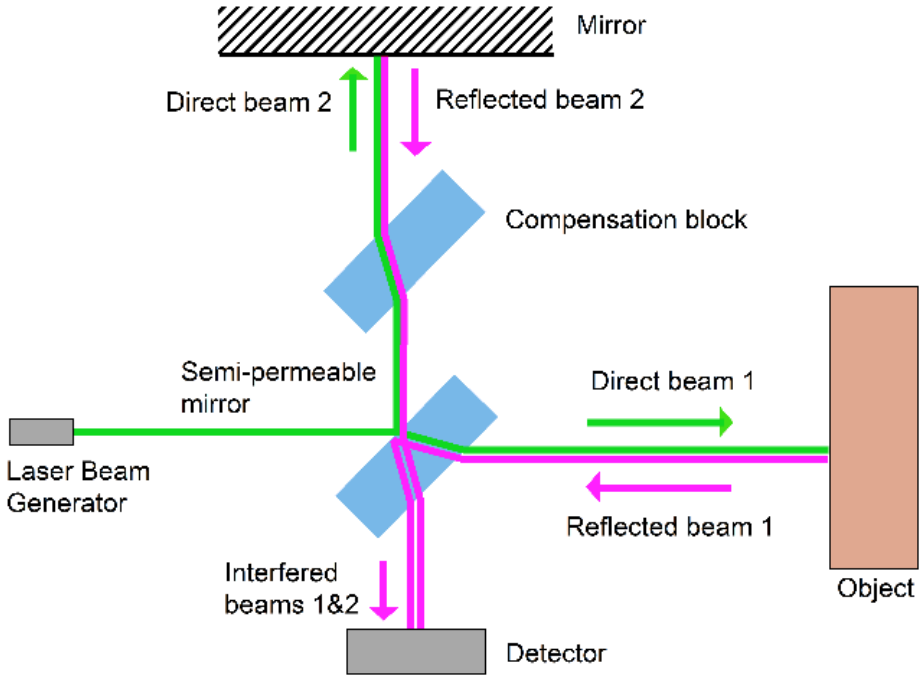


Fig. 1.7: Interferometric measuring principle

Depending on distance to the object, the time difference of two beams occurs, which is related to the difference of length of beam flight. According to this phenomenon, constructive or destructive interference appears and considering positions of minima and maxima we can define desired distance.

Interferometric principle is the method, which provides most precise measurements. Its accuracy is typically on the order of micrometres, rarely even on the order of nanometres. Use of these devices is unusual, because its accuracy is balanced by very high costs, hugeness and complexity of device.

Phase Shift Method

Phase Shift method is modified variation of Time of Flight method described above, which uses some features from Interferometric method. Compared to Time of Flight method, it differs in fact, that outputting laser beam has sinusoidally modulated optical power. Reflected light is monitored, and the phase of the power modulation is compared with that of the sent light. This phase shift is proportional to time of flight.

As for an interferometer, the phase shift method involves an ambiguity regarding the measured distance, because with increasing distance the phase will vary periodically. However, the periodicity is much larger than in an interferometer, since the modulation frequency is much lower than the optical frequency. Also, the ambiguity can be easily removed by measuring with two different modulation frequencies simultaneously [7].

Compared with interferometers, devices based on the phase shift technique are less accurate, but they allow unambiguous measurements over larger distances.

1.3.2 Important Parameters of Laser Scanners

There are many types of laser scanners, which are varying a lot from each other. As was mentioned above, for instance measuring principle significantly influences final parameters of laser scanner. The most important parameters, which should be considered when choosing the best fitting product, are:

- **Measuring range** – minimal and maximal distance from scanner to object that can be measured.
- **Field of view** – angular width of view. Together with measuring range defines working area.
- **Angular resolution** – the smallest angular difference between two measured points. Defines density of acquired points and consequently influences minimal distance between two measured points.

- **Scanning frequency** – updating frequency of point value. Defines how many measurements of distance to the one point are performed each second. Usually depends on field of view and angular resolution.
- **Resolution and accuracy** – resolution defines minimal difference between two distances which are noticeable by scanner and accuracy is defined as area around measured value in which manufacturer declares that the true value is located for sure.
- **Data interface** – important parameter which specifies how data can be transferred from measuring device to the processing device. Influences possible applications of scanner.
- **Weight** – important in cases when placed on some places where maximum load is limited, e.g. at the wrist of robotic manipulator.

1.4 Laser Scanner SICK LMS 400

Laser Scanner SICK LMS 400 is one of the most accurate laser scanners disposing with 3 m range. There are many of more accurate scanners, but no one has so wide measuring range as this model. This section presents important parameters of this model and summarizes if it is proper device according to the aim of this project.

1.4.1 Specifications

Working Area

The maximum measuring range of the SICK LMS is 3 m and the smallest permitted distance of the measurement object is 700 mm. The field of view covers an angle of 70°[8]. Entire working area is shown on figure 1.8.

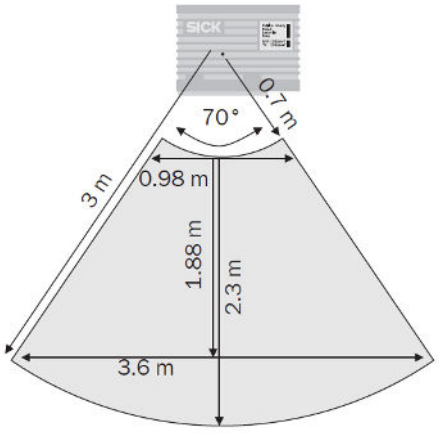


Fig. 1.8: Working area of SICK LMS 400 [8]

Angular Resolution

Angular resolution is configurable in range from 0,1333 to 1°. Maximal angular error of rotating mirror is $\pm 0,1°$ [8]. In case of measuring in perpendicular plane relative to projection axis of scanner, given angular error corresponds to 5,2 mm error of laser beam positioning in 3 m distance and 1,2 mm error in 0,7 m distance.

Scanning Frequency

Scanning frequency is configurable in range from 360 Hz to 500 Hz [8]. Configured combination of scanning frequency and angular resolution influences measurement accuracy, so not all the combinations are allowed.

Resolution and Accuracy

The typical systematic measuring error of the LMS400 is $\pm 4\ mm$. This information applies for the individual measurement point at an object remission in range from 10% to 100% at room temperature [8].

The statistical measuring error is dependent on the remission and distance of the object. Tab. 1.4 shows typical and maximal measuring errors when operating in permitted combination of scanning frequency and angular resolution, room temperature and maximum external light of 2000 Lux.

Tab. 1.4: Statistical measuring error of SICK LMS 400 [8]

Remission	**Distance**	**Statistical error**	
		Typical	**Maximal**
100%/200%	700 to 3000 mm	3 mm	–
78%	700 to 999 mm	3 mm	7 mm
	1000 to 2500 mm	3 mm	5 mm
	2501 to 3000 mm	3 mm	7 mm
40%	700 to 999 mm	4 mm	9 mm
	1000 to 2500 mm	4 mm	8 mm
	2501 to 3000 mm	4 mm	9 mm
10%	700 to 999 mm	9 mm	15 mm
	1000 to 2500 mm	9 mm	12 mm
	2501 to 3000 mm	9 mm	15 mm
6,5%	700 to 3000 mm	10 mm	–

Data Interface

The LMS400 has three different interfaces which each of them could be used for device configuring and also for receiving measured values [8]:

- **Ethernet** – TCP/IP peer to peer interface, data transmission rate of 10 MBaud (10BaseET), only half duplex supported. IP address, TCP/IP port and subnet mask is predefined by manufacturer but can be changed by configuration message. Intended for measurements output because of its baud rate.
- **Host interface** – Intended for the same use as Ethernet, but because of its baud rate doesn't allow real-time measurement output. It's physical layer can be configured as both RS-232 or RS-422. The interface parameters are freely configurable. The factory setting for the host interface is as follows: RS-232, 9600 Baud, 8 data bits, 1 stop bit, no parity.
- **Terminal interface** – this feature can operate simultaneously with the Ethernet or Host interface and is primarily intended to provide a reliable data connection for configuration. Therefore, its interface parameters can not be changed and are follows: RS-232, 9600 Baud, 8 data bits, 1 stop bit, no parity.

If real-time output of values is required, the only usable interface is Ethernet because of limited baud rate of other interfaces.

Weight

Weight of device is approximately 2,3 kg [8].

1.5 Laser Scanner MicroEpsilon scanCONTROL

MicroEpsilon scanCONTROL 2750-100 laser scanner is a very precise laser scanner based on triangulation method. Compared to the SICK Laser Scanner mentioned above, it is much more precise device, but its measuring range is much smaller. It is designed for measuring very small distances.

1.5.1 Specifications

Resolution and Accuracy

Resolution depends on material remission. Manufacturer declares, that the resolution varies in interval from 15 μm to 40 μm [13]. Linearity error on entire range is up to ± 200 μm, but repeatability of each point is within resolution interval.

Working Area

The maximum measuring range is 450 mm, when the smallest permitted distance of measured object is 350 mm. So that, operating range is 100 mm. The field of view covers an angle of 12.4°[13]. Entire working area is shown on figure 1.9.

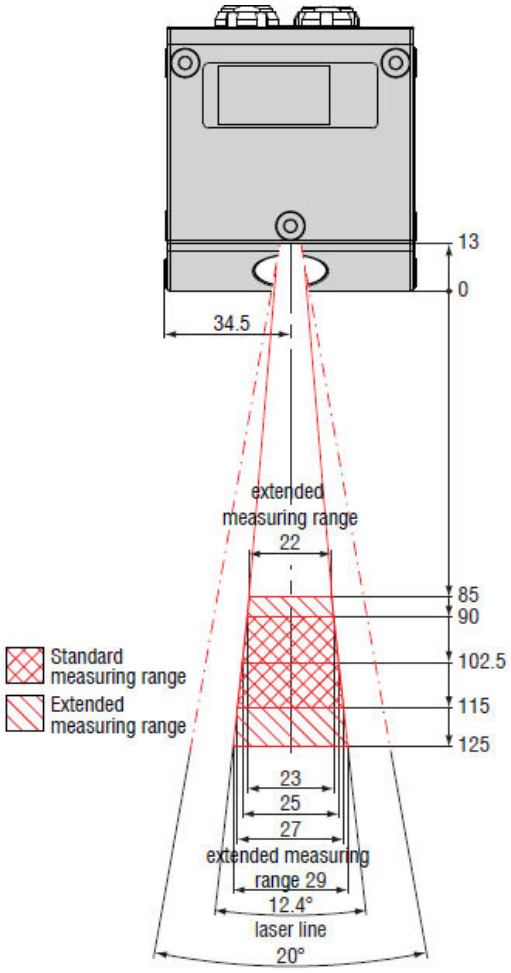


Fig. 1.9: Working area of MicroEpsilon Laser Scanner [13]

Angular Resolution

Maximal number of measured points in one profile is 640 [13], together with angular field of view, it refers to angular resolution approximately $1,94 \cdot 10^{-2}$ deg.

Scanning Frequency

This device is equipped with very high measuring frequency up to 2 000 Hz [13]. In this configuration, it produces huge amount of data, which are unprocessable by common network devices, so the speed of data transmission can be decreased by Idle time settings or by Averaging filtering.

Data Interface

Device is equipped with Ethernet, FireWire and RS422 interfaces [13]. Because of bandwidth of these interfaces, the only interface able to be operated in every configurations is Ethernet.

Weight

Weight of device is approximately 800 g [13].

1.6 Summary

In the text above, the important parameters of all three devices used in this solution are discussed.

The robotic manipulator EPSON C3 is one of the most accurate manipulators available on the market. It disposes with continuous working area and it is controllable in real-time over Ethernet interface. Thus, we can declare, that EPSON C3 manipulator is sufficient device for our purposes.

The purpose of this 3D scanner is not exactly specified. If scanned object is a large structure, the measuring range must be as wide as possible, so in this case, the SICK LMS laser scanner seems to be the best fitting product, because it provides the best accuracy in the class of scanners with as wide working range as SICK LMS has.

On the other hand, if scanned object is a tiny structure, the accuracy is decisive factor and the measuring range is not as much important. In this case, the best fitting product is the MicroEpsilon laser scanner.

Because we don't know what type of object will be scanned, our solution must be universal. Thus we use both the scanners with possibility of easy switching between these two laser scanners in dependence on size of scanned object.

2 3D SCANNER ARCHITECTURE

This chapter describes hardware constitution of 3D scanner, communication mechanisms between each devices which 3D scanner consists from and principles and functions of its kernel software. The essential equations for measurement transformations are derived. Result of this chapter is device, which inputs desired scanning trajectory and outputs cloud of points in three-dimensional space. This device is here understand under the term "3D scanner".

2.1 Mechanical Constitution

Laser scanner is mounted on the manipulator's end-point like on the figures 2.1 and 2.2. Both laser scanner and manipulators are connected via Ethernet switch to the computer, where application controlling manipulator and reading data from scanner is running.

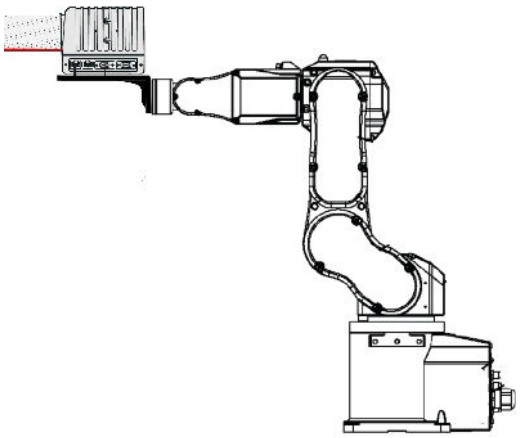


Fig. 2.1: SICK laser scanner mounted on manipulator [5][8]

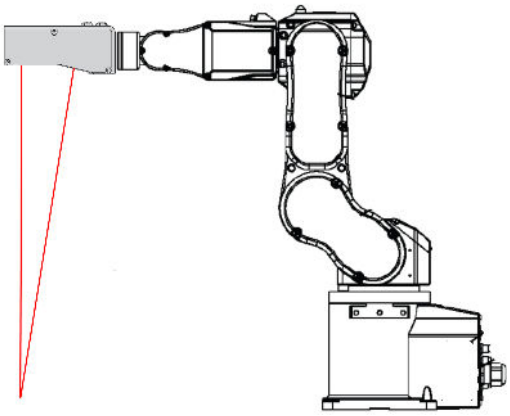


Fig. 2.2: MicroEpsilon laser scanner mounted on manipulator [5][13]

Each scanner can be easily unmounted just by unscrewing four screws and unplugging Ethernet cable. With same ease it can be mounted on again, what allows easy change of used laser scanner in dependence on object size (as mentioned in section 1.6).

2.2 Interfacing Laser Scanners

As mentioned above, there are two different laser scanners which are replaced by each other according to the measured object. Each of these scanners has different

controlling mechanism, but we want to have the unified interface, which doesn't matter on the scanner actually connected.

As a result of this, universal C# driver (with same interface) has been created for both scanners. This driver could be used also for other purposes, so application demonstrating use of this driver has been created too.

2.2.1 Driver for SICK Laser Scanner

Remote device configuration as well as measure triggering and measured data outputting is realized using messages sent over any available interface. The message is in most cases ASCII-encoded string representing transmitted request or received parameters. Message could also be binary coded, eg. in case of returning output data from measurement. We use Ethernet interface because of real-time transmission capability. Then the message is framed as follows [8]:

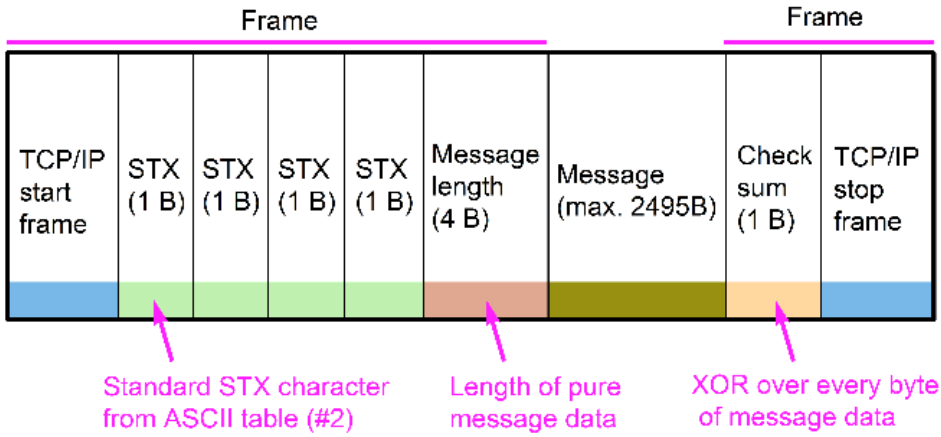


Fig. 2.3: Structure of Ethernet frame for SICK LMS400.

Message block on fig. 2.3 contains request or output data.

The main part of driver's interface consists of methods, which allows user to perform any possible scanner handling. Behind these methods, translation to message, message enclosing into the frame mentioned above, and finally its transmitting is realized.

Driver class implements its own thread, where all the inner mechanisms are performed. Because of it, driver is possible to provide asynchronous events, which occurs in cases of received output data from measurement or in cases of device configuring, measure starting, stopping, etc.

Library Interface

Detailed documentation about LMS400 Driver, describing its using and provided interface, is attached as Appendix A. The driver itself, in form of source codes, is available on enclosed DVD with electronic attachments.

The fastest way to understand all the abilities of this driver is to look at the source codes of Demonstration Application, which are also included on enclosed DVD with electronic attachments.

Demonstration Application of Driver Usage

In order to show usage of the LMS400 Driver, the Demonstration Application has been created. Using the Windows Form, it makes all the driver's abilities available for try. Screenshot of this application is shown on fig. 2.4.

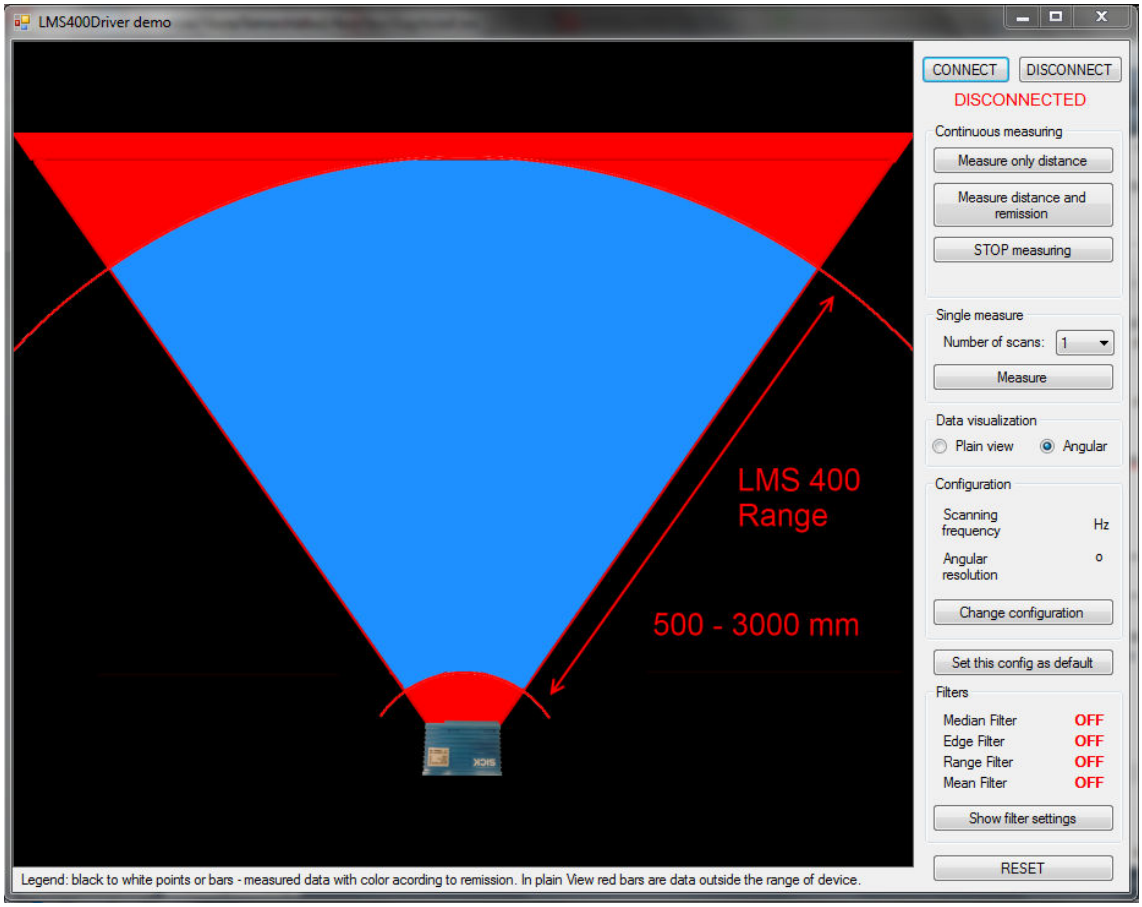


Fig. 2.4: Screenshot of LMS400 Driver Demonstration Application.

On the right side of window, the toolbar is displayed. It contains all features necessary to scanner control. Each ability of scanner can be controlled from here.

The main part of window is covered by visualization, which can be performed in two modes:

- **Angular View** – shows the position of object by dots in appropriate location within scanners range. Sequence of dots creates a curve relevant to the scanned profile. This view is easy comprehensible, but doesn't allow to show all the details because of its scale.
- **Plain View** – displays scanned profile in form of graph, where X axis is the angle of view and Y axis means measured distance. Angular transformation is performed on output data, so the flat areas are shown as flat even if their distance in not equal. This view shows more details and also can display remission of material by the color varying from black to white according to remission.

The same profile shown in both modes is presented on figure 2.5.

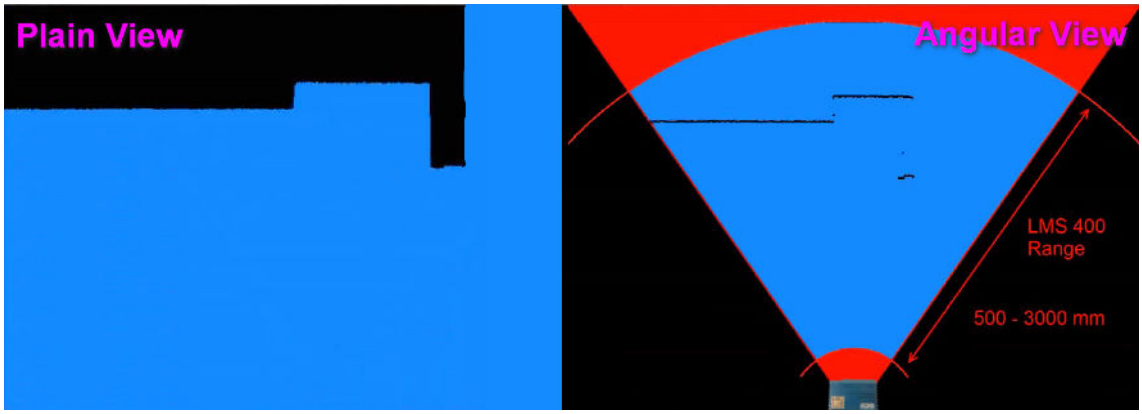


Fig. 2.5: Comparison of viewing modes of Demonstration Application.

If you don't have an access to the LMS400 laser scanner and so you aren't able to try this Demonstration Application, you can see screenshot-video of it in use, which is available on enclosed DVD with attachments.

2.2.2 Driver for MicroEpsilon Laser Scanner

The controlling mechanism of MicroEpsilon laser scanner is slightly different from the previous one. Manufacturer doesn't specify description of communication protocol, but provides DLL library with functions for scanner handling. My C# driver encapsulates these DLL functions into the one class with methods. These methods just calls appropriate DLL function.

There is also asynchronous event raising implemented and generally, controlling of laser scanner is simplified. Originally, user has to call several DLL functions and

consequently check their error status. Using this driver, just one call of method is necessary, eventual error is announced by exception.

Library Interface

Detailed documentation about MicroEpsilon Laser Scanner Driver is attached as Appendix B. The driver itself, in form of source codes, is available on enclosed DVD with electronic attachments.

The fastest way to understand all the abilities of this driver is the same as at previous driver – look at the source codes of Demonstration Application, which are also included on enclosed DVD with electronic attachments.

Demonstration Application of Driver Usage

To show usage of driver, the Demonstration Applications has been created. Using the Windows Form, it makes all the driver's abilities available for try. Screenshot of this application is shown on fig. 2.6.

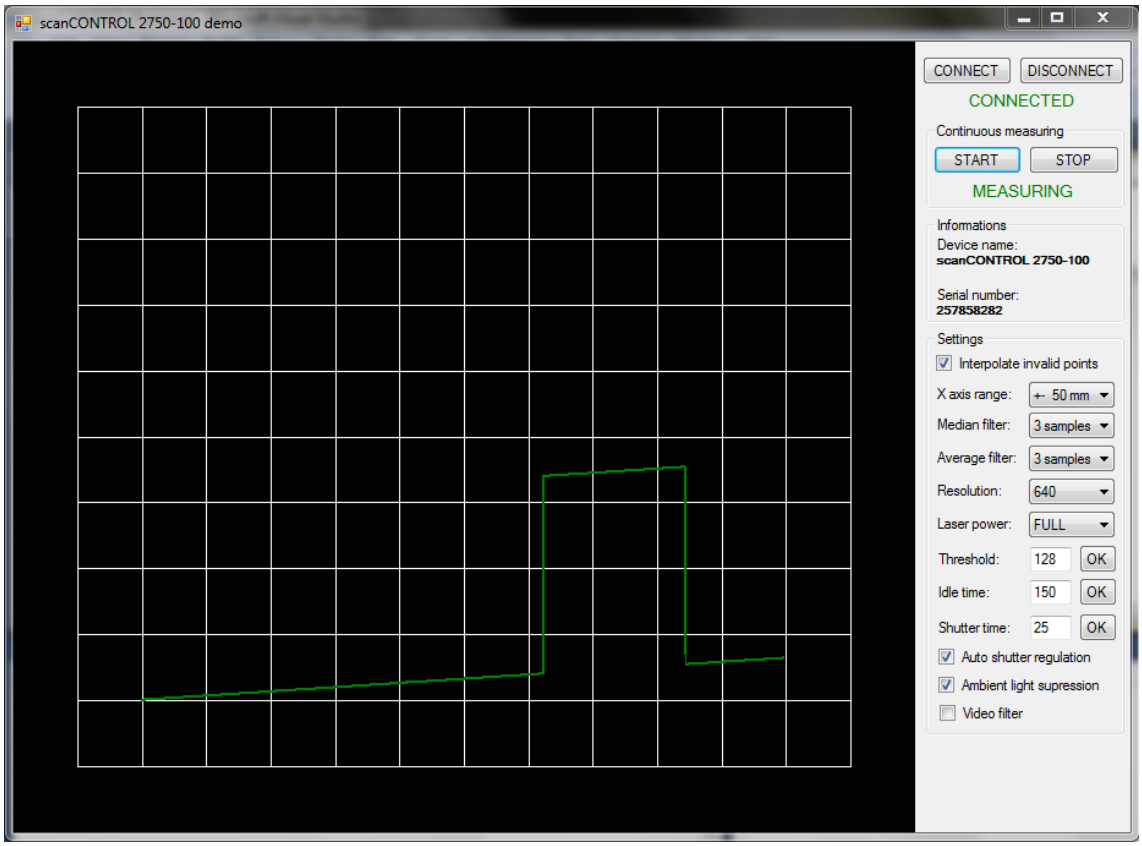


Fig. 2.6: Screenshot of MicroEpsilon Driver Demonstration Application.

On the right side of window, the toolbar is displayed. It contains all features necessary to scanner control. Each ability of scanner can be controlled from here.

2.3 Interfacing Robotic Manipulator

Robot ordinarily executes program, which has been written in manufacturer-designed scripting language (SPEL+) and uploaded into the Controlling Unit before starting the operation. It doesn't allow the flexible reaction on external events received from surrounding systems, what is important for our purposes.

However, SPEL+ includes instructions for receiving and transmitting data over Ethernet, so this could be solved by solution shown on figure 2.7.

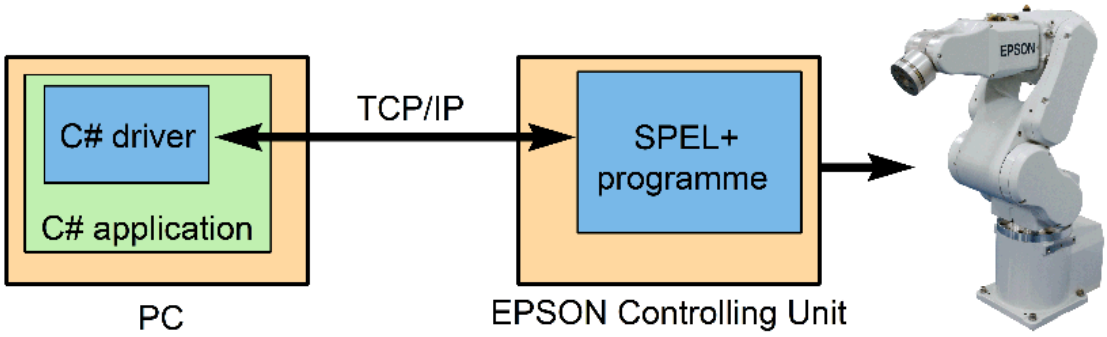


Fig. 2.7: Scheme of manipulator's driver solution.

There is a SPEL+ programme which receives commands over TCP/IP and performs desired actions. On the other side of Ethernet link, there is a C# driver which receives requests over its interface and transmits commands to the SPEL+ program. The real-time manipulator controlling (by C# application) is then empowered.

Exactly this desired driver is a result of semestral thesis of Ing. Martin Fireš ([11]), which has been published by Brno University of Technology, so we can use it for our purposes. In my project, I will use this driver in order to control manipulator in real-time. For detailed specification of methods provided by this driver see [11].

2.4 Operating Principle

As mentioned at the beginning of this chapter, 3D scanner in our meaning is device, which receives desired scanning trajectory as input and outputs cloud (set) of points in three-dimensional space. Operating principle of this device is shown and described on fig. 2.8.

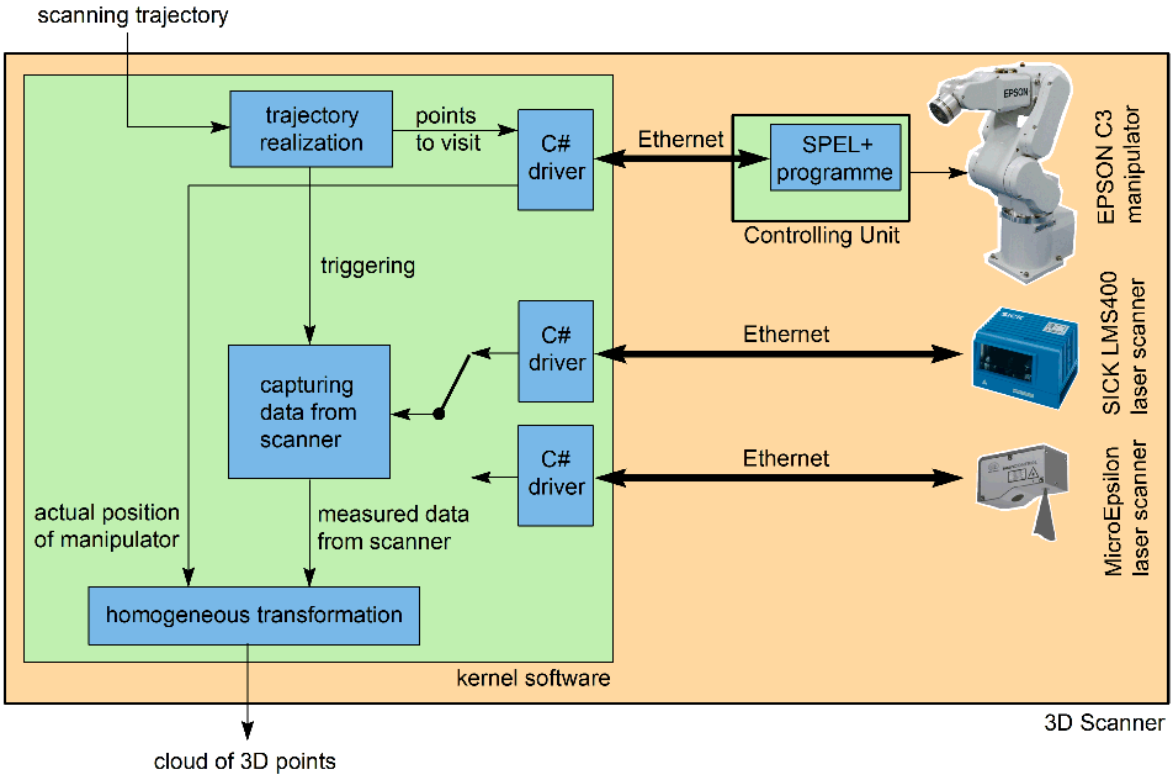


Fig. 2.8: Scheme of 3D scanner with accent on kernel software.

Desired scanning trajectory is given to the Trajectory Realization Block. This block realizes segmentation of trajectory into the list of points, which are sent into the manipulator's driver. This block simultaneously sends triggering commands to Data Capturing Block when desired point is reached.

Data Capturing Block acquires data from connected laser scanner and preprocesses them into the form of 3D vectors. There is a possibility of switching between available scanners.

Homogeneous Transformation Block provides mathematical transformations into the our coordinate systems. It receives data acquired from scanner and reads actual position of manipulator's end-point. From these values the cloud of 3D points is computed. This process is described in following section.

Other parts of 3D scanner have been explained in sections above.

2.5 Homogeneous Transformations

Homogeneous transformation computes position of measured point in default coordinate system (x_0,y_0,z_0) using data from manipulator encoders and data from laser scanner:

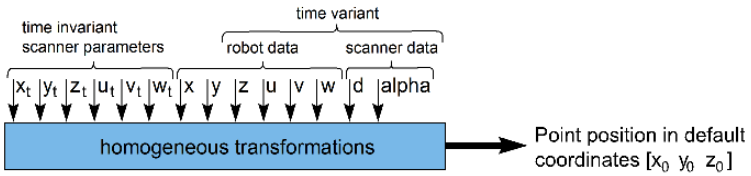


Fig. 2.9: Inference mechanism of Homogeneous transformations block

Let's divide entire system into particular systems, where each of them has its own coordinate system (fig. 2.10). System 0 is default coordinate system in which measurement results should be presented. System M is own coordinate system of manipulator. Its origin is placed in the centroid of bottom mounting plate. System E is system with origin at the manipulator's end-point.

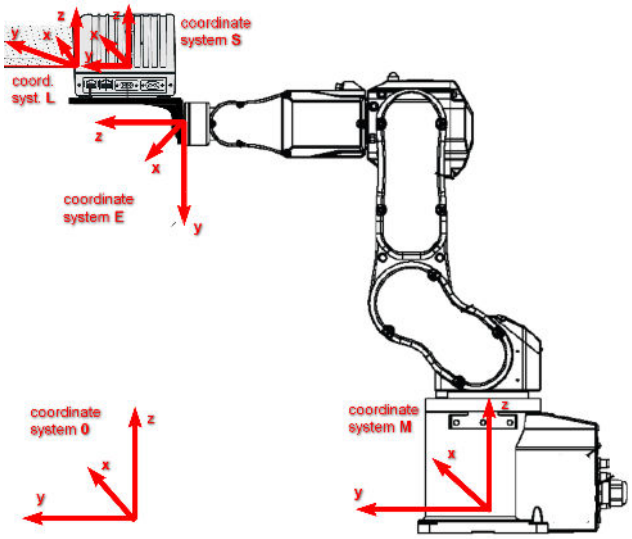


Fig. 2.10: Overview of used coordinate systems

System S is coordinate system of laser scanner and finally, system L is system of measuring laser beam emitter. Entire homogeneous transformation from measured values to point in default coordinate system could be defined as sequence of essential transformations between neighbour coordinate systems:

$$H_{0L} = H_{0M}H_{ME}H_{ES}H_{SL} \tag{2.1}$$

Then homogeneous coordinates of measured point in coordinate system 0 are defined as:

$$P_0 = H_{0L}P_L \tag{2.2}$$

In following text, detailed description of homogeneous transformations between each neighbour systems are presented.

2.5.1 Transformation from Measured Point to Laser Emitter

Let's have point P_L, which distance from laser emitter inside laser scanner has been measured by laser beam. Its homogeneous coordinates in emitter coordinate system L are:

$$P_L = \begin{bmatrix} d \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.3}$$

where d is distance of point from laser emitter. This value is returned by laser scanner, together with measuring angle α.

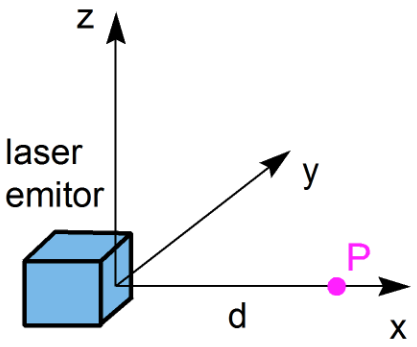


Fig. 2.11: Coordinate system L

2.5.2 Transformation from Laser Emitter to Scanner

There is a sweeping mechanism, which rotates with this laser source in xy plane of laser scanner's coordinate system S. Coordinate systems S of each used scanner are shown on fig. 2.12.

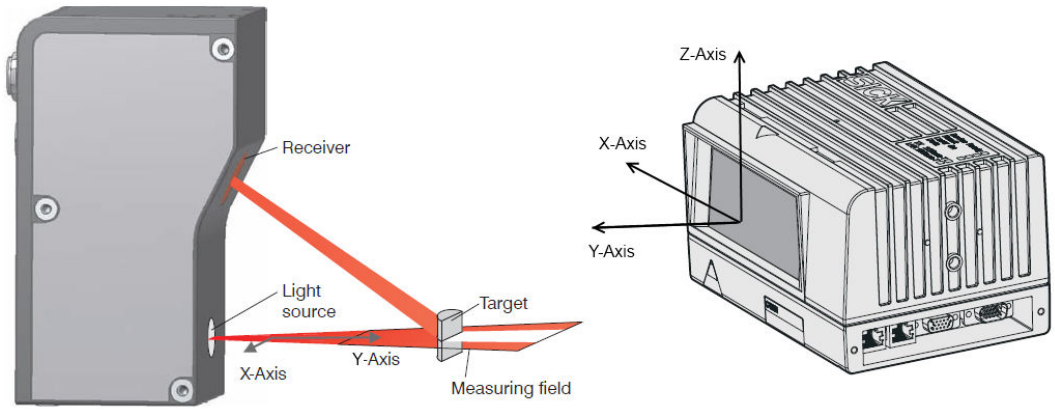


Fig. 2.12: Coordinate system S of each used scanner [8][13]

Because system L is system S rotated along z axis by angle α, homogeneous transformation between coordinate systems L and S is defined by matrix H_{SL}:

$$H_{SL} = \begin{bmatrix} \cos(180-\alpha) & -sin(180-\alpha) & 0 & 0 \\ sin(180-\alpha) & \cos(180-\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

where α is instant deviation from $-x$ axis of coordinate system S and is also returned by laser scanner.

2.5.3 Transformation from Scanner to End-Point

This laser scanner is mounted on the manipulator's end-point. Centroid of manipulator's end-point defines origin of coordinate system E. Homogeneous transformation from system E to S is combination of translation in all axes and rotation described by RPY model [14]:

$H_{ES} =$

$$\begin{bmatrix} c(u_t)\ c(v_t) & c(u_t)\ s(v_t)\ s(w_t) - c(w_t)\ s(u_t) & s(u_t)\ s(w_t) + c(u_t)\ c(w_t)\ s(v_t) & x_t \\ c(v_t)\ s(u_t) & c(u_t)\ c(w_t) + s(u_t)\ s(v_t)\ s(w_t) & c(w_t)\ s(u_t)\ s(v_t) - c(u_t)\ s(w_t) & y_t \\ -s(v_t) & c(v_t)\ s(w_t) & c(v_t)\ c(w_t) & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

where $c(x) = cos(x)$, $s(x) = sin(x)$, x_t, y_t and z_t is translation of system S in system E along appropriate axis, u_t is roll, v_t is pitch and w_t is yaw of system S in coordinate system E. These parameters describes mounting of laser scanner and could be acquired from documentation of each laser scanner and its mounting holder.

This homogeneous matrix defines dimensions and constitution of scanner. Be aware, that in our case, only this matrix is different for SICK LMS and MicroEpsilon Scanner. Other matrices H_{ab} are same for both scanners.

2.5.4 Transformation from End-Point to Manipulator

Actual position of manipulator's endpoint (eg. origin of system E) is placed inside own manipulator's coordinate system M. Homogeneous transformation from system M to E is once again combination of translation in all axes (x,y and z) and rotation described by RPY model (u is roll, v is pitch and w is yaw)[14]:

$$H_{ME} = \begin{bmatrix} c(u)\ c(v) & c(u)\ s(v)\ s(w) - c(w)\ s(u) & s(u)\ s(w) + c(u)\ c(w)\ s(v) & x \\ c(v)\ s(u) & c(u)\ c(w) + s(u)\ s(v)\ s(w) & c(w)\ s(u)\ s(v) - c(u)\ s(w) & y \\ -s(v) & c(v)\ s(w) & c(v)\ c(w) & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

where $c(x) = cos(x)$ and $s(x) = sin(x)$, x, y and z is actual position of manipulator's end-point in own manipulator's coordinate system M, u is its roll, v is pitch and w is yaw. These parameters are time-variant and could be acquired from manipulators driver.

2.5.5 Transformation from Manipulator to Default System

The coordinate system of manipulator M is placed inside a default coordinate system 0, which we want to transform measured points into. The homogeneous transformation from system 0 to M is H_{0M}. In our case, system 0 is identical with system M. If it is not, another matrix similar to H_{ME} could be used.

2.5.6 Scanner Parameters for Homogeneous Transformations

MicroEpsilon Scanner

Scanner mounted on the manipulator's end-point has the same orientated coordinate system as end-point, only translated. Vector of parameters for this scanner is then:

$$C_{\mu E} = (x_t; y_t; z_t; u_t; v_t; w_t) = (0; 43, 0; 27, 5; 0; 0; 0) \tag{2.7}$$

SICK LMS400 Scanner

Scanner mounted on the manipulator's end-point has coordinate system rotated by 90° along the X axis and then translated. Vector of parameters for this scanner is then:

$$C_{LMS} = (x_t; y_t; z_t; u_t; v_t; w_t) = (0; -123, 4; 86, 5; 0; 0; 90) \tag{2.8}$$

2.6 Summary

This chapter showed mechanical constitution of 3D scanner, described communication with partial devices and illustrated principles of scanner's operation. In last part, important equations were derived using homogeneous transformation. These equations are essential for transforming measured data into 3D space, what is purpose of this 3D scanner.

3 CALIBRATION PROCEDURE

All the theoretical knowledge derived in the chapter above is practically usable just in case, that scanner holder and box of laser scanner has exactly the same dimensions as indicated by manufacturer, exactly even surfaces, perpendicular sides, etc. It is clear, that all these preconditions could not be exactly satisfied, but in this case, the uncertainty following from violation of preconditions influences our results so much, that we cannot neglect them.

As a reason of this, some indispensable deviation will be present and must be considered in design. We can involve this uncertainty to the scanner – end-point transformation constants:

$$\begin{aligned} x_t &= x_{t0} + x_e & u_t &= u_{t0} + u_e \\ y_t &= y_{t0} + y_e & v_t &= v_{t0} + v_e \\ z_t &= z_{t0} + z_e & w_t &= w_{t0} + w_e \end{aligned}$$

where x_t is real value of transformation parameter, x_{t0} is theoretically computed value of transformation parameter and x_e is parameter uncertainty. Solving of calibration procedure can be considered as looking for 6 calibration constants – x_e, y_e, z_e, u_e, v_e and w_e.

3.1 Choosing Calibration Object

Choice of proper calibration object significantly influences not just complexity of calibration algorithm but also final precision of computed calibration constants. There are two possibilities:

- **Using very precise calibration object** – Calibration algorithm is very simple, but accuracy of calibration constants depends significantly on the precision of calibration object. Manufacturing of high precision object is also expensive.
- **Using generic calibration object** – Manufacturing of this object is very cheap and also possible accuracy is theoretically higher, but complexity of the calibration algorithm is much more higher, in some cases could be so complex, that is non-solvable.

Consider our particular situation: There is a scanning system constituted as on fig. 3.1. Manipulator's end-point positioning

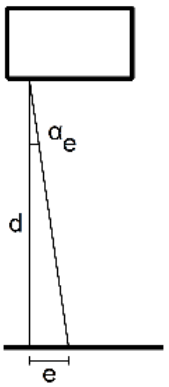


Fig. 3.1: Illustrating required accuracy

accuracy is $\pm 0.02\ mm$ [5], accuracy of laser scanner is up to $\pm 0.04\ mm$ [13]. Laser scanner is capturing object from distance $d = 400\ mm$. To reach the value of positioning error e same as scanner's accuracy, maximal angular error α_e can be:

$$\alpha_e = \arctan\left(\frac{e}{d}\right) = 0.008° \tag{3.1}$$

From this consideration follows, that required accuracy of calibration constants should be in range of hundredths of degree. As mentioned above, accuracy of first proposed solution of calibration procedure (using very precise calibration object) is approximately same as accuracy of manufacturing of very precise object, so we want to have a calibration object manufactured with accuracy in range of hundredths of degree. Norm ISO 8407 defines accuracy $\pm 0.60°$ for the highest accuracy class [16], common angular accuracy of iron-made machine-manufactured products of high-accuracy class is about $\pm 0.15°$, the best accuracy available on the market reaches up to $\pm 0.02°$ [15]. From this follows, that approach to calibration procedure based on very precise object cannot be used in this case and approach based on second proposed possibility must be used: using generic calibration object.

3.2 Calibration Procedure Fundamental Principles

As mentioned above, calibration based on generic object significantly increases complexity of solution and in some cases could be even non-solvable. From this reason, we are expecting calibration scene composed from cuboid laying on the flat ground as shown on fig. 3.2.

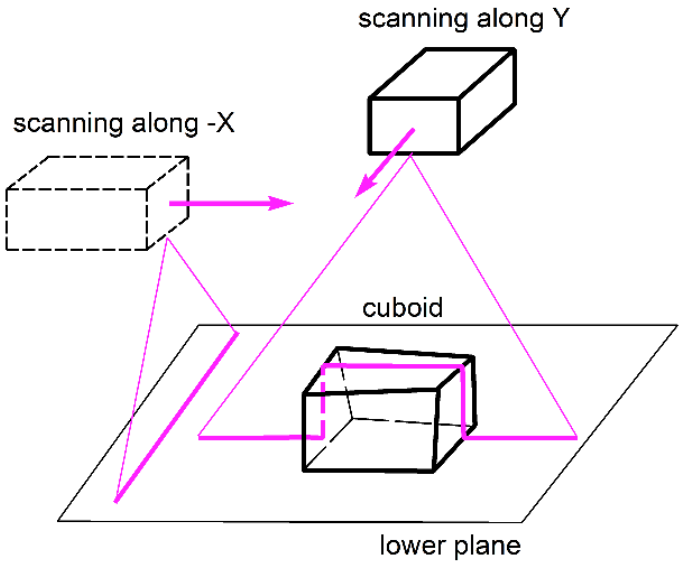


Fig. 3.2: Illustrating calibration principle.

The "generality"of this scene rests in fact, that there are no requirements on perpendicularity of cuboid sides, on cuboid edge dimensions, on planarity of lower plane, etc. We are just expecting some object, which is approximately cuboid laying on something similar to the plane. This is a big advantage of this calibration procedure, because this object is very cheap and calibration quality is time-invariant.

As shown on fig. 3.2, we are scanning the same scene two ways:

- Parallel equidistant scanning along positive Y axis
- Parallel equidistant scanning along negative X axis

After that, we have 2 images of same scene, which are similar, but distorted relatively to each other. Because particular system deviations (expressed by calibration constants) causes different distortion in case of scanning along -X axis and Y axis, calibration constants should be able to be computed from these distortions.

3.3 Numeric Approach to Calibration

One of possible approaches is the numeric one. When we know the scanning trajectory and scanner-mounting transformation matrix, we can reverse-compute pure data acquired from laser scanner and recompute them using the new scanner-mounting transformation matrix. By this way, we can recompute already measured points to the form like its are acquired with scanner with calibration constants applied. This transformation is defined as:

$$x_1 \xrightarrow{T} x_2 \qquad T = H_1^{-1} \cdot H_{ESnew} \cdot H_{ESold}^{-1} \cdot H_1 \tag{3.2}$$

where H_1 is transformation defined by scanning trajectory (actual position of scanner in manipulator coordinates), H_{ESold} is an scanner – end-point transformation without calibration constants applied and H_{ESnew} is an scanner – end-point transformation with calibration constants already applied.

Using error function (see section 3.3.1), we can evaluate correlation between 2 images acquired by scanning with different trajectories. Then, we can change calibration constants, recompute measured points and evaluate correlation once again. By computing all the variations of calibration constants and choosing the combination with maximal correlation, we can compute values of calibration constants.

On Intel Core2Duo E7400 with 2 GB RAM memory, computation of error function for correlation takes approximately 850 ms. Considering resolution of calibration constants 0.01 and range $\pm 1^\circ$ in case of rotation and $\pm 5mm$ in case of translation, time required for test all the combinations is:

$$t = 200^3 \cdot 1000^3 \cdot 0.85 = 6.8 \cdot 10^{15}\ s \tag{3.3}$$

From this reason, this approach cannot be used if significant decrease of tested combinations not applied.

3.3.1 Error Function Used for Computing Correlation

In order to evaluate correlation of two scans, error function has been designed. For each point from one scan, distance to every point from second scan is computed and square of distance to the nearest point is added to the sum:

$$f_{err}(I_1, I_2) = \sum_{i=1}^{Count(I_1)} \min(\parallel I_1(i)I_2(j) \parallel)^2 \qquad j \in (1; Count(I_2)) \tag{3.4}$$

where I_1 and I_2 are scans which correlation we are looking for and $\parallel xy \parallel$ is a distance between points x and y.

Due to decrease of elapsed computational time, the nearest point is not computed for every point from first scan. Each sixth sample in each profile is taken into consideration, what approximately corresponds with 1 mm distance between points.

3.4 Analytic Approach to Calibration

Another possible approach is the analytic one. In the following text, we are trying to derive analytical equations and methodology, which leads to value of each calibration constant is computed. In first draft, exact computing of calibration constants from generic cuboid image was planned, but from complexity reasons, some simplifications were applied.

In the following, each calibration constant is discussed separately. At first part of each section, constraining conditions are mentioned, and consequences resulting from their violation are discussed. Then, the analytical solution is proposed.

3.4.1 Deviation of Roll Rotation u_e

Constraining Conditions

Flat ground, which calibration object lays on, is laying in plane, which normal has the same direction as Z axis. In other words, flat ground is exactly perpendicular to the Z axis. If this condition is not satisfied, each 1° of its declination causes approximately 0.05° error in resulting value of u_e. It was experimentally detected, that declinations bigger than 1° is easily look-distinguishable, so we are able to compute u_e with accuracy ±0.05°.

Solution

Consider scanner performing scanning procedure as shown on fig. 3.3. In real, it is scanning profile AB, but because it "doesn't know"about its deviation u_e, displays measured data like $A'B'$. Our goal is to find relation between declination of flat ground β_y and u_e.

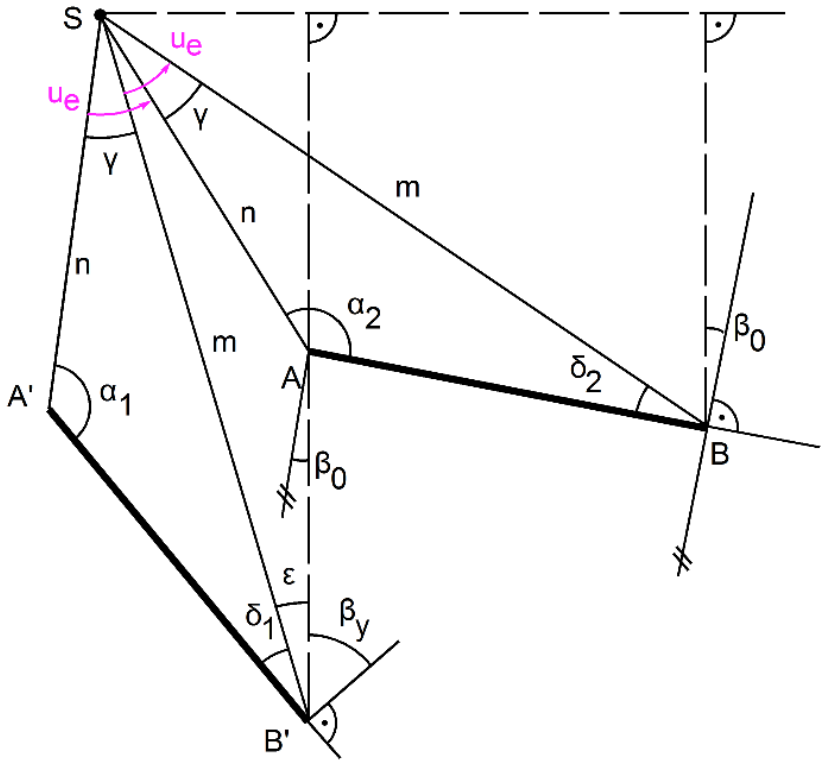


Fig. 3.3: Prove of relation between flat ground declination and u_e.

Triangles ABS and $A'B'S$ are the same, only rotated, because they had two legs of same length and same angle between them. From this follows:

$$\alpha_1 = \alpha_2 \qquad \delta_1 = \delta_2 \tag{3.5}$$

From triangle $A'B'S$ follows equation for δ_1:

$$\delta_1 = 180° - \gamma - \alpha_1 \tag{3.6}$$

We can declare angular equation for point A:

$$\alpha_2 + 90° + \mid \angle SAB' \mid -\beta_0 = 360 \tag{3.7}$$

We can declare angular equation for point B':

$$\delta_1 + \varepsilon + \beta_y = 90^\circ \tag{3.8}$$

Angle ε is defined as:

$$\varepsilon = 180 - (u_e - \gamma) - \mid \angle SAB' \mid \tag{3.9}$$

By substituting δ_1 by equation 3.6 and ε by combination of equations 3.7 and 3.9 in equation 3.8, we have:

$$90^\circ - \gamma - \alpha_1 + [180^\circ - (360^\circ - \alpha_2 - 90^\circ + \beta_0) - (u_e - \gamma)] + \beta_y = 0 \tag{3.10}$$

$$\alpha_2 - \alpha_1 - \beta_0 - u_e + \beta_y = 0 \tag{3.11}$$

Applying equation 3.5 on equation 3.11 results to:

$$\beta_y = \beta_0 + u_e \tag{3.12}$$

where β_y is measured declination of flat ground in direction of scanned profile and β_0 is real declination of flat ground in direction of scanned profile.

We proved the relation between value of measured declination of flat ground and value of calibration constant u_e. Note, that this relation would be valid just for declination in direction of profile. In direction of scanner movement (which is perpendicular on profile direction), the measured declination will be only the original declination of flat ground. Equation 3.12 also considers only value of declination, not its direction. When considering also direction, we can summarize this in table 3.1:

Tab. 3.1: Measured declination of flat ground in particular cases

	Scanning along Y axis	**Scanning along -X axis**
Declination α in direction of -X axis	$\alpha_y = \alpha_0$	$\alpha_x = \alpha_0 + u_e$
Declination β in direction of Y axis	$\beta_y = \beta_0 - u_e$	$\beta_x = \beta_0$

where α_x, α_y, β_x and β_y are measured declinations and α_0 and β_0 is a real declination of flat ground.

Combination of equations in table 3.1 results to final equation for u_e:

$$u_e = \alpha_x - \alpha_y = \beta_x - \beta_y \tag{3.13}$$

This calculation is realized at program as following: At first, the whole scan is divided into two parts – points belonging to the flat ground and points belonging

to calibration object. This dividing function is based on profile derivation, where differences between neighbour points are computed, and then edges are detected from peaks of these differential data.

After that, only points of flat ground are passed to the plane fitting function, which is based on Least Squares Method and computes normal vector of best fitting plane. This vector is then decomposed to vectors along X, Y and Z axis, from which magnitudes, the declinations α_x, α_y, β_x and β_y are computed. The value of u_e is then computed from equation 3.13. If equations $\alpha_x - \alpha_y$ and $\beta_x - \beta_y$ gives the same results, it means, that constraining conditions are satisfied. Otherwise, they are not.

3.4.2 Deviation of Pitch Rotation v_e

Constraining Conditions

Calibration object is exact cuboid, with all the sides perpendicular to the neighbour side. Axis of the cuboid is oriented along the scanning direction. These preconditions are very hard, but its violation is penalized with error only 0.01° per 1° of declination of sides or declination of cuboid axis relative to scanning direction (approximately).

Note: Generic solution in case of violation of constraining conditions also exists, but just in case, that other calibration constants are zero. This is the reason, why special solution with constraining conditions is used.

Solution

Error parameter v_e causes declination of sides perpendicular to scanner's movement. Sides parallel to scanner movement direction are slightly shifted, but it's normal has stays unchanged. This situation is displayed in horizontal projection (fig. 3.5).

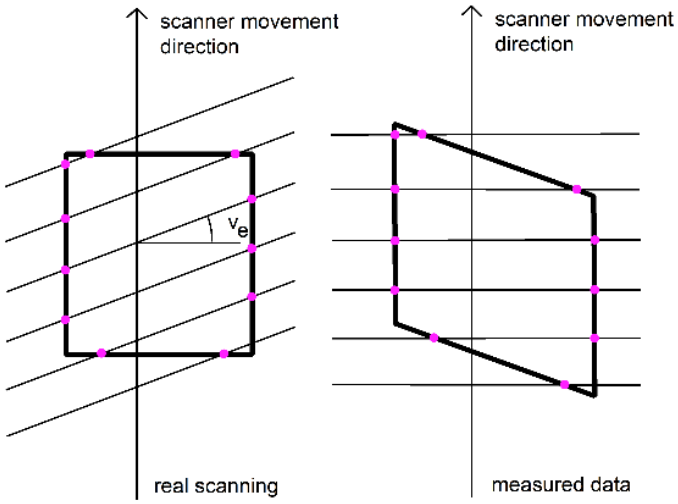


Fig. 3.4: Illustrating influence of v_e - horizontal projection.

Satisfying the conditions above makes the solution very simple. Two of corners are enlarged by size of v_e and two of them are same size reduced. This can be proved by following:

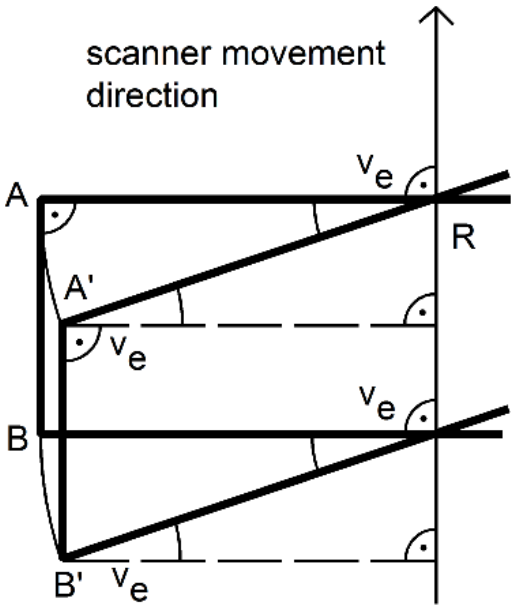


Fig. 3.5: Prove of relation between corner angle and v_e.

When taking the scanner movement direction as y axis of planar coordinate system, where x axis is perpendicular to y axis, the relation between point $A = [x_A, y_A]$ and $A' = [x_{A'}, y_{A'}]$ is given by equations:

$$x_{A'} = x_A \cdot \cos(v_e) \qquad\qquad y_{A'} = y_A - x_A \cdot \sin(v_e) \tag{3.14}$$

The movement of new point depends only on x coordinate, so if points A and B has same x coordinate, also points A' and B' will have the same x coordinate, what signalizes, that line segments AB and $A'B'$ are parallel.

When line segments AB and $A'B'$ are parallel, their perpendicular lines must be also parallel, and line $\overleftarrow{RA'}$ going through two parallel lines has the same declination for both of them. This proves, that all of angles marked at fig. 3.5 as v_e. And this consequently proves declaration above fig. 3.5.

Note, that the corner, which is enlarged when scanning along Y axis is reduced when scanning along -X axis and vice versa. Then v_e can be computed from equation:

$$v_e = f(A'B', A'_1B'_1) = \mid \angle A'_1EA' \mid \qquad E = \overleftrightarrow{A'B'} \cap \overleftrightarrow{A'_1B'_1} \tag{3.15}$$

where points A' and B' are acquired when scanning along Y axis and points A'_1 and B'_1 are acquired when scanning along -X axis.

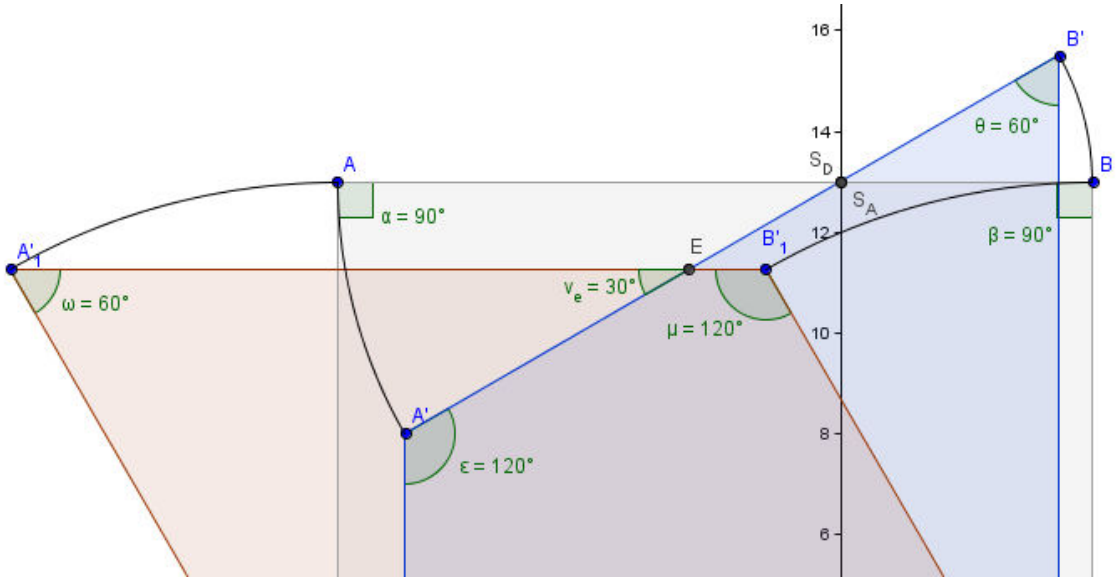


Fig. 3.6: Illustrating computation of v_e.

This equation is graphically illustrated on fig. 3.6:

This calculation is realized at program as following: Using derivation, where differences between neighbour points are computed, the corners are detected from peaks of these differential data. After that, v_e is computed using equation 3.15.

3.4.3 Deviation of Yaw Rotation w_e

Constraining Conditions

Conditions are the same as at section 3.4.2, calibration object is exact cuboid, with all the sides perpendicular to the neighbour side. Axis of the cuboid is oriented along the scanner movement direction. These preconditions are very hard, but its violation is penalized with error only 0.01° per 1° of declination of sides or declination of cuboid axis relative to scanner movement direction (approximately).

Solution

Let's define term "facing side"for purposes of this section: Facing side is a side of calibration cuboid, which is perpendicular to scanner's movement and is simultaneously facing to the scanner moving direction in case that w_e orientation is positive or it is rearing to the scanner moving direction in case that w_e orientation is negative.

Error parameter w_e causes declination of facing side, the opposite side has no declination. Comparing to the v_e, where horizontally-projected shape of cuboid was

deformed, here vertically-projected shape is deformed. The most illustrative is visualization of w_e impact on fig. 3.5.

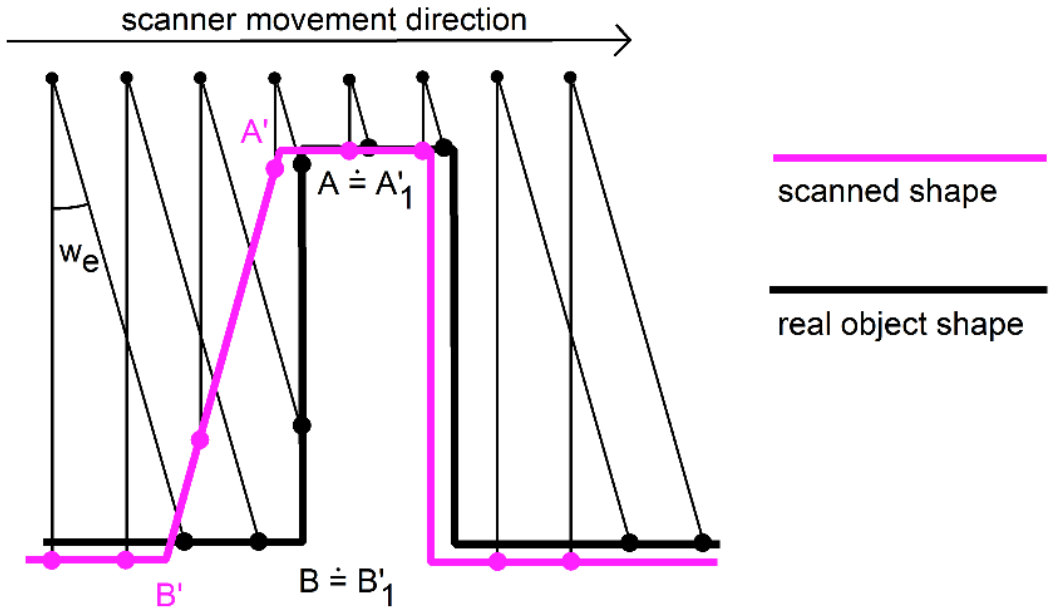


Fig. 3.7: Illustrating influence of w_e - vertical projection.

Declination of facing side caused by w_e impact is defined exactly the same way as declination caused by v_e:

$$w_e = f(A'B', A'_1B'_1) = \mid \angle A'_1EA' \mid \qquad E = \overleftrightarrow{A'B'} \cap \overleftrightarrow{A'_1B'_1} \tag{3.16}$$

where points A' and B' are acquired when scanning along Y axis and points A'_1 and B'_1 are acquired when scanning along -X axis. Note, that in case of sides, which are not facing sides, their normal stays unchanged, what is figured on fig. 3.5 by $A \doteq A'_1$ and $B \doteq B'_1$.

Validity of this preposition was proved at section 3.4.2.

This calculation is realized at program as following: Using derivation, where differences between neighbour points are computed, the vertical sides are detected from peaks of these differential data. After that, facing sides are detected using the declination between sides scanned along Y and sides scanned along -X. The not facing sides has almost no declination. Then, w_e is computed for facing sides using equation 3.16.

3.4.4 Deviation of X Translation x_e and Z Translation z_e

Constraining Conditions

All rotation error parameters must be zero. Solution methods for other calibration constants are independent on other error parameters, so these calibration constants can be computed in advance, then acquired data can be recalculated in order to suppress all rotation error parameters. After that, calculation of x_e and z_e can be processed.

Solution

Error parameters x_e and z_e causes translation of points just in XY plane. All the points are translated by same value, so no deformation of object shape is performed. Object translation based on x_e and z_e shows fig. 3.8

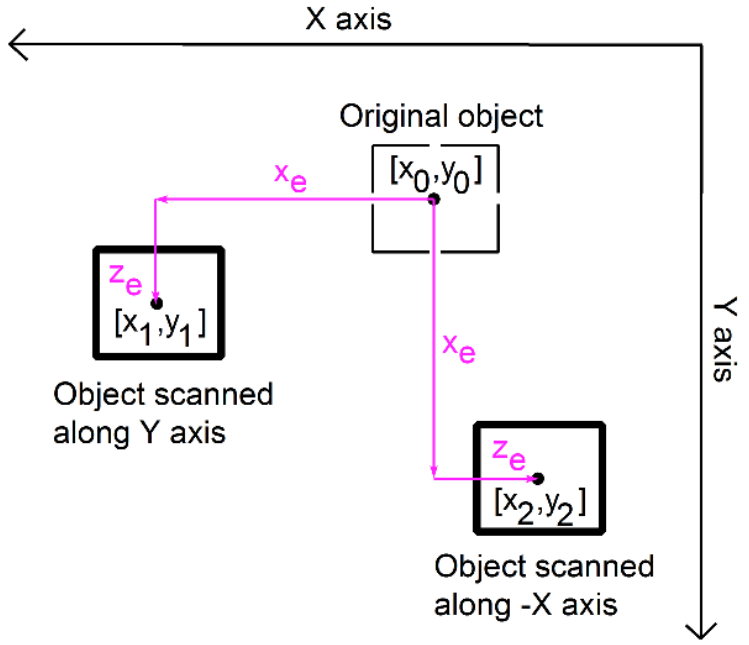


Fig. 3.8: Illustrating influence of x_e and z_e - horizontal projection.

From fig. 3.8 follows equations for coordinates of scanned objects:

$$x_1 = x_0 + x_e \tag{3.17}$$

$$y_1 = y_0 + z_e \tag{3.18}$$

$$x_2 = x_0 - z_e \tag{3.19}$$

$$y_2 = y_0 + x_e \tag{3.20}$$

where points x_1 and y_1 are coordinates of object scanned along Y axis and x_2 and y_2 are coordinates of object scanned along -X axis.

By combination of these equations, we gets directly equations for evaluating x_e and z_e:

$$x_e = \frac{x_1 - x_2 - y_1 + y_2}{2} \qquad z_e = \frac{y_1 - y_2 + x_1 - x_2}{2} \tag{3.21}$$

This calculation is realized at program as following: At first, the whole scan is divided into two parts – points belonging to the flat ground and points belonging to calibration object. This dividing function is based on profile derivation, where differences between neighbour points are computed, and then edges are detected from peaks of these differential data.

After that, only points of calibration object are used to compute the centroid of object for both data acquired by scanning along Y axis and -X axis. From these centroids, x_e and z_e is calculated using equations 3.21. If Z coordinates of both centroids are not almost equal, it means, that constraining conditions are not satisfied.

3.4.5 Deviation of Y Translation y_e

Error parameter y_e causes shifting of all points along Z axis and doesn't matter on which direction data were scanned. It means, that we are not able to compute the y_e value from these data. But our goal is to produce device, which is able to make model of 3D object, where right relation between points is necessary. We don't want to have this object correctly place in our coordinates. Because of this, calibration constant y_e in not computed and stay zero.

3.5 Hybrid Calibration

Each of proposed methods for evaluating calibration constants has their advantages and disadvantages. Numerical approach is very accurate, but unbearably slow, what makes it useless. Analytic approach is fast, but its accuracy depends on constraining conditions, which are almost never satisfied.

Hybrid calibration combines both of proposed methods in order to use their advantages and suppress their disadvantages. At first, calibration constants are computed using analytic method. The u_e is evaluated with accuracy approximately $\pm 0.05°$, v_e and w_e are evaluated with accuracy approximately $\pm 0.02°$ and x_e and z_e are evaluated with sufficient accuracy just by using analytic approach. After that, the numeric approach is used, with significantly smaller number of iteration comparing to using only numeric method. Time required for test all the combinations is:

$$t = 20 \cdot 6 \cdot 6 \cdot 0.85 = 10.2 \text{ min} \tag{3.22}$$

Using numeric method, we reach the sufficient accuracy of all calibration constants in tolerable time.

3.6 Summary

In this chapter, the most difficult part of this thesis has been presented. During the solution of calibration procedures, lot of different approaches was tested, but lot of them has important disadvantage, what made them useless. Final solution, which combines analytic and numeric approach seems to be fast enough with sufficient accuracy. Big advantage of this calibration procedure is its cheap solution because of no necessity of precise calibration object.

4 CREATING 3D MODELS

This chapter describes next software layer of this solution, which encloses kernel software mentioned in previous chapter. This layer consists from Trajectory Planning Block, Surface Generator Block and File Saver. Placings of these blocks in the entire system are shown on fig. 4.1 and it's functions are described in further text.

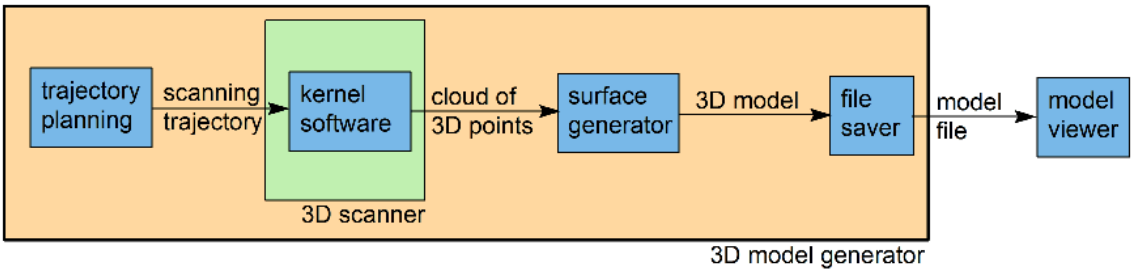


Fig. 4.1: Blocks of solution's higher software layer.

4.1 Trajectory Planning

As described in section 2.4, kernel low level software receives trajectory in form of list of points, where each of them is described by three Cartesian coordinates of translation and three RPY coordinates of rotation. This list is processed by Trajectory Realization block which moves with robotic manipulator's end-point and sends triggering commands to the laser scanner when desired point has been reached.

In order to make entire system user-friendly as much as possible, desired trajectory is not written as a list of point directly. There is an application called Trajectory Manager, which handles with the code written in our own scripting language for describing a trajectory and compiles it to the form of list of points.

Trajectory descriptions written in this scripting language are stored in the text files with *.3dtraj extension. If script passes the syntax checking procedure, it can be compiled to the list-of-point form, which is stored in files with *3dctraj extensions. This file is readable for Trajectory Realization block.

Besides common functions for text-file handling like opening, saving, exporting and erasing, also sophisticated Check Syntax module has been programmed. It not just check right syntax of code, but also determines purpose of syntax error, like a missing semicolons, brackets, wrong number of parameters, parameters out of permitted range, undefined point references, etc. If script passes syntax check, trajectory is compiled and saved to the trajectory storage of main program, from which can be easily executed. Trajectory Manager can be also launched from main program.

User interface of Trajectory Manager tool is displayed on fig. 4.2.

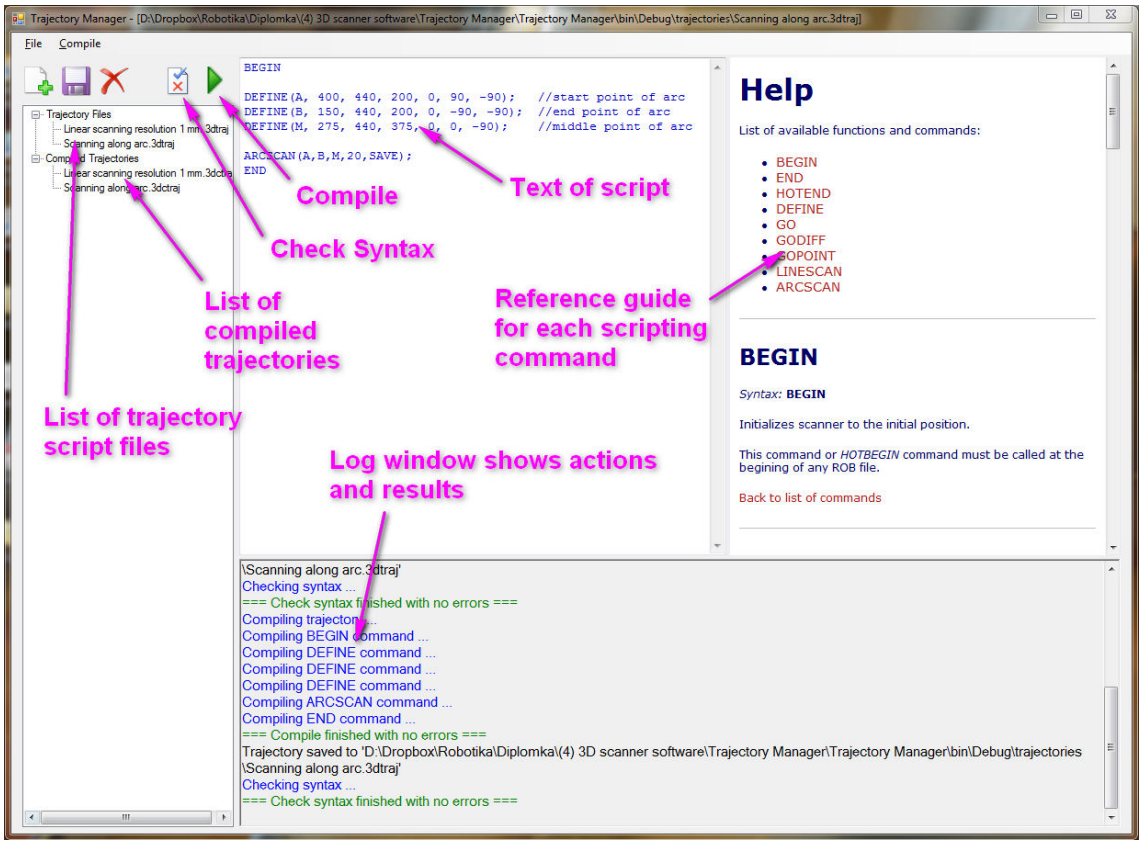


Fig. 4.2: Trajectory Manager user interface.

The most frequently used scanning trajectory is equidistant parallel scanning. Compare to direct defining list of points, the recipe for this trajectory has only 3 lines of simple code. Example of pure profile data acquired with this trajectory are shown on fig. 4.3.

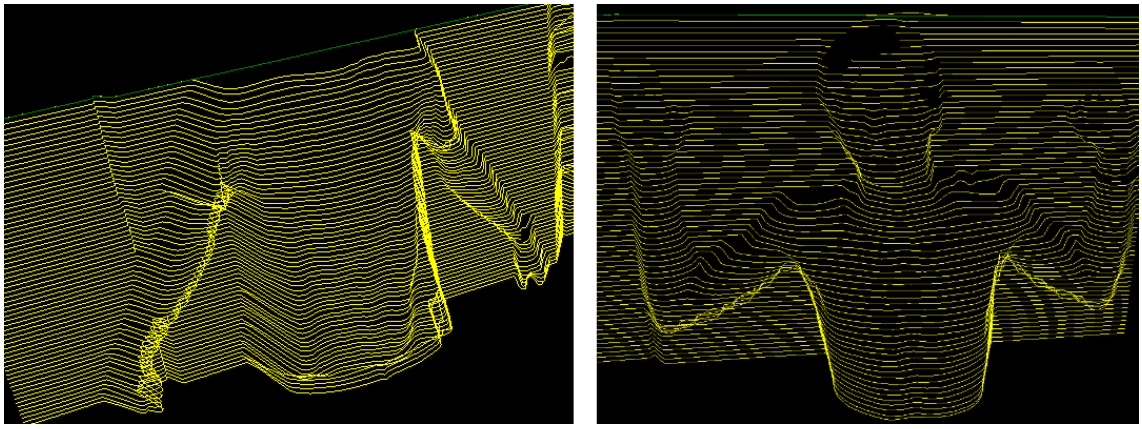


Fig. 4.3: Illustration of equidistant parallel scanning.

4.1.1 Scripting Language for Trajectory Description

The first version of our scripting language contains just few essential commands for trajectory describing, but in future work it will be extended with more complex functions. Actually supported commands are:

- **BEGIN** – each file must begin with this command, what ensures begining of manipulator's movement in initial position.
- **END** – each file must be terminated with this command, what ensures returning of manipulator's end-point back to the initial position.
- **HOTEND** – in some very specific cases, it could be necessary to leave the manipulator in other position than initial. If HOTEND command is used, manipulator's end-point is not returned to the initial position. It is recommended to use in every case END command instead of this one.
- **DEFINE** – This command defines a point, which is labelled with unique name and described by 6 coordinates. Such points could be used in following commands. The reason is to limit repeated writing of coordinates as function parameters. For example, the ARCSCAN function receives 3 points instead of 3×6 coordinate parameters.
- **GO** – Moves robotic end-point to position described by 6 absolute coordinates. Optionally, saves actual profile to the memory before moving.
- **GODIFF** – Moves robotic end-point to position described by 6 coordinates relative to actual position of end-point. Optionally, saves actual profile to the memory before moving.
- **GOPOINT** – Moves robotic end-point to position described by point, defined by DEFINE function in advance. Optionally, saves actual profile to the memory before moving.
- **LINESCAN** – Moves from start point to the ending point using linear trajectory, and during this movement saves a mesured profile each x millimetres (x can be defined).
- **ARCSCAN** – Moves from start point to the ending point using arc-shaped trajectory, which is described by third point laying on this arc. This movement is realized in x steps, where x can be defined.

More detailed description of each command syntax and it's function is shown directly in the window of Trajectory Manager tool. It provides fast access to reference guide for trajectory programmers.

Besides these commands, also comments in C++ style code are allowed to make a trajectory recipe comprehensible.

4.2 Generating Surface from Set of Points

As shown on fig. 4.1, output data from kernel 3D scanner are in form of 3D-point cloud. It means, that unorganized set of three-dimensional data is produced and sent into the Surface Generator block, where smooth shaded surface should be generated.

Because computer graphic is based on triangles, which define surface, this task can be described also as looking for links between given points in order to produce triangles, which visualize surface of scanned object.

In this project, the Delaunay Triangulation has been used, which is a common mesh generation method in scientific computation [17]. It is so frequently used, that there is a Delaunay Triangulation function already built in MATLAB [18]. From fast developing and easy testing reasons, work of Surface Generation block is performed outside our main application, in MATLAB environment. Scanned data are saved to file, which is processed by MATLAB script producing the list of triangles. This list is included into the model file and loaded by 3D Model Viewer, which is detailed in following chapter.

Example of acquired point cloud representing human fingers is shown on fig. 4.4 and visualization of surface created from outputted triangle list is shown on fig. 4.5.

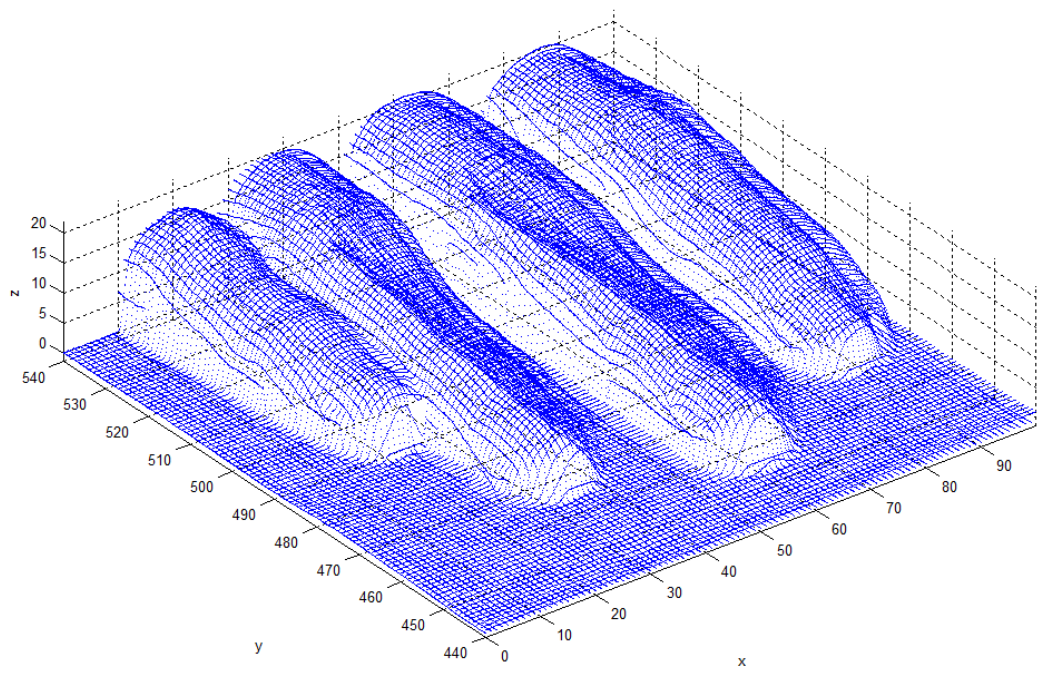


Fig. 4.4: Input cloud point given to Surface Generator.

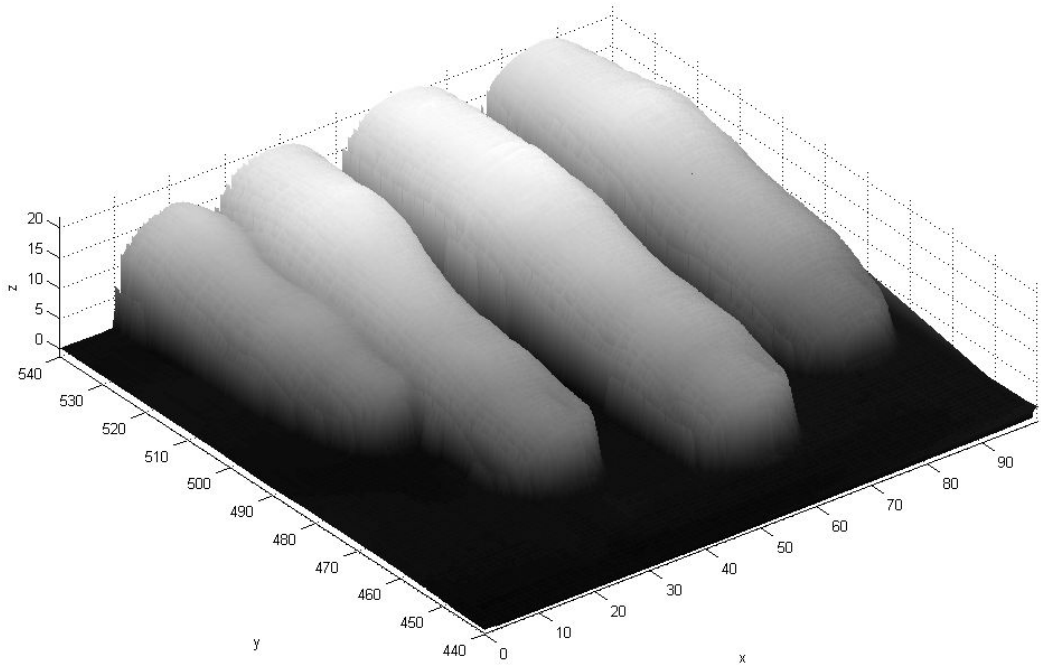


Fig. 4.5: Visualization of outputting triangle list.

This method uses all measured data and doesn't deal with outlying points and noise in signal. Also its requirements on computation time are relatively high. From this reason, this approach is temporary, and that is why it is not embedded in main program.

In my future work, the Surface Generator block will be embedded in main application. Simple Delaunay Triangulation will be replaced with fast method described in [19], which is based on modified Level-Set function inspired by fluid flow. This method accelerates computation and also deals with noise in measured data.

4.3 Saving Models

3D models covered by surface should be saved for future viewing. Due to very specific application, no common file format is used and our own file structure has been proposed. Format for the model saving was designed with aim to be flexible and easily extensible for future abilities. More accent is also put on the backward-compatibility. Figure 4.6 shows outline of the file structure.

There is a text file, which consists from several blocks. Each block contains string identifier, which describes type of data within the block. This identifier is followed by number of text lines containing data of this block. Then, data itself are stored.

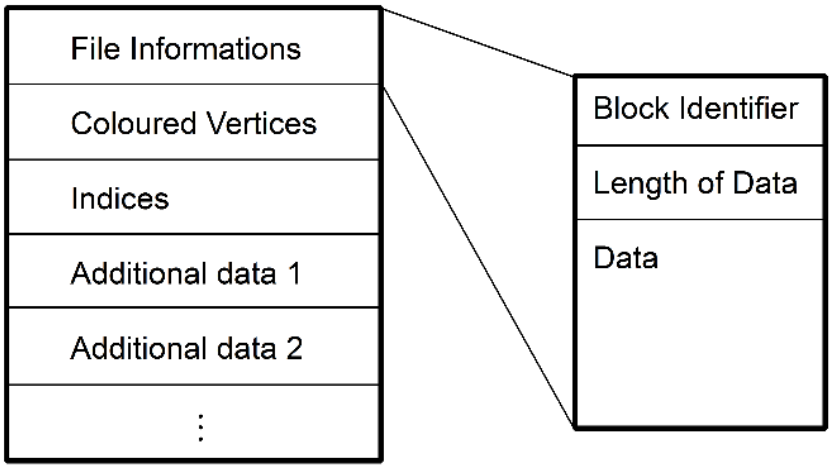


Fig. 4.6: Structure of the file for saving models.

This approach supports backward compatibility – if some new feature is added, new block is appended to the file. If application version does not supports this feature, program reads the identifier which is unknown for it and skip this block over.

There are 3 data blocks in this program version:

- **File Informations** – Section containing metadata describing this file. There are informations about author, date, time, description of scanned object, model dimensions, used units, etc. These metadate are stored in form of XML, what empowers unlimited extensions of this block, if desired.
- **Coloured Vertices** – This section contains vertices itself. Each vertex is described by it's coordinates and colour. In default, all the vertices has the same colour, but its could be coloured according to the point remission, point reliability, etc. Data are stored in the same way as stored in the graphic device, so its could be loaded directly to it's buffer.
- **Indices** – There are links between vertices stored in this section, which defines particular graphic triangles. Data are stored in the same way as stored in the graphic device, so its could be loaded directly to graphic device's buffer.

4.4 User Interface

All these features are enclosed in program with graphic user interface, which is shown on fig. 4.7. It allows scanning along predefined trajectories (created in Trajectory Manager tool, see 4.1) or using the manual mode, where scanner is moved to desired position by specifying it's coordinates and saving of actual profile is performed by appropriate button. Scanned data are shown in main window if form of yellow line

per each saved profile. Projection of these data can be adjusted using the mouse or keyboard. Zooming, rotating and moving of the view is allowed.

Scanned data can be saved to file or covered by surface and sent to the 3D Model Viewer application, which also part of this work. This application is described in section 5.

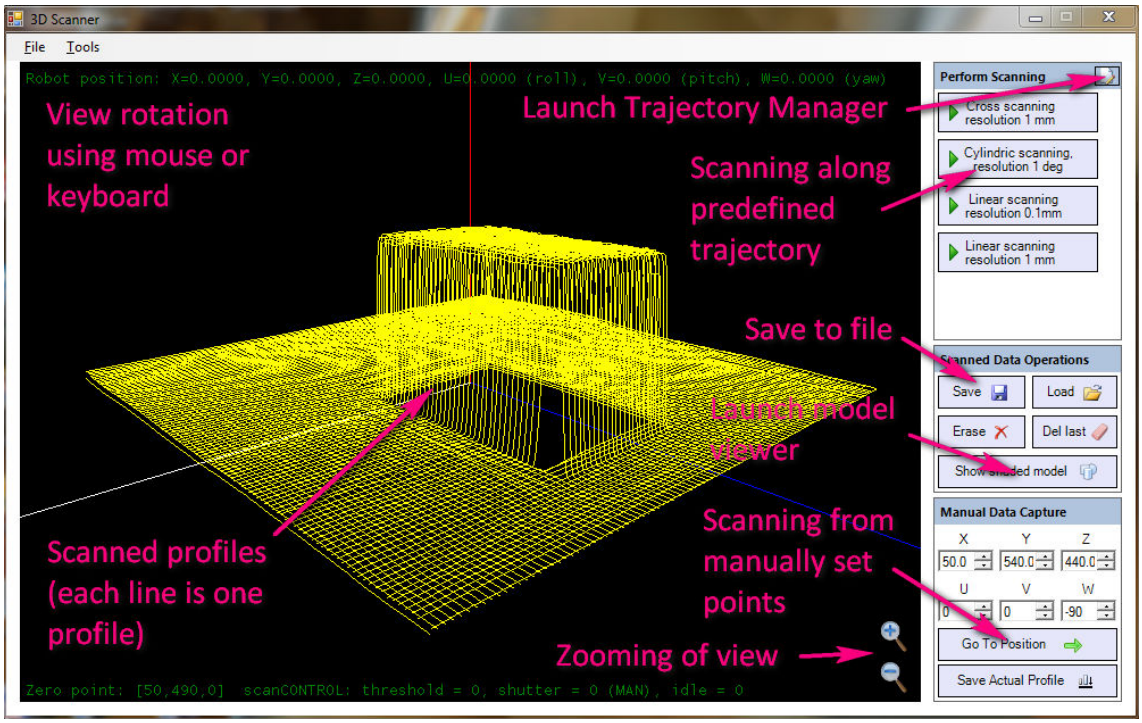


Fig. 4.7: Structure of the file for saving models.

4.5 Summary

This chapter describes enclosing higher layer of the whole scanning system. It contains Trajectory Planning Block, which is equipped with own scripting language allowing trajectory description in simple way and Surface Generator Block which is building surface from measured point cloud. This block is temporary designed, much robust and faster surface generator will be implemented in my future work.

Significant part of this software layer is graphic user interface, which allows controlling all the possible functions from one easy-comprehensible environment.

In this chapter, also the file structure for saving models is described, which is simple and easily extensible.

5 3D MODEL VISUALIZATION

This chapter describes showing of finished 3D models already saved in file. Also brings some samples of human body scans.

5.1 Loading Models

As mentioned above in section 4.3, file structure is inspired with memory structure of graphic device buffer. Blocks of vertices and indices in file are stored in the very similar form as in vertex and index buffer, so it can be directly copied to the graphic device's memory.

5.2 Displaying Models

3D model viewer is an application created in XNA. The main window with displayed sample model is shown on figure 5.1.

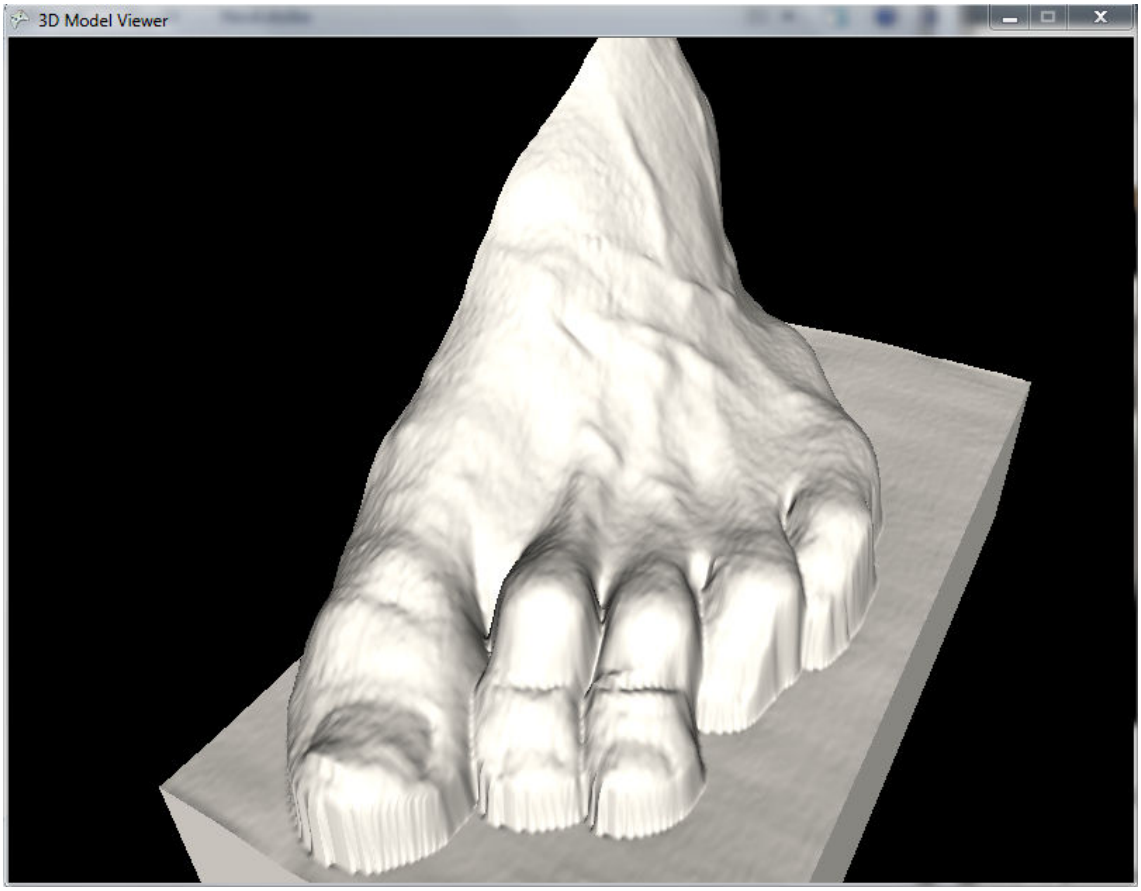


Fig. 5.1: Application window of 3D Model Viewer

It's a first prototype having the only one ability: it performs showing of 3D model saved in file. The control interface of application is focused on ability to show model in any angle of view and at any zoom. Commanding of view is performed by:

- Left and right arrow key on keyboard provides the horizontal rotation of model.
- Up and down arrow key on keyboard empowers vertical rotation of model (tilting).
- By scrolling of mouse-wheel, the zoom can be adapted.
- When Shift key is pressed, the arrow keys on keyboard are used for moving model in appropriate direction.
- Rotation can be also easily performed by mouse dragging in appropriate direction.
- Dragging when Shift key is pressed causes model moving in appropriate direction.

Application can be easily extended with other features depending on purpose of this scanner, what could be done in my future work on this project. For example, measuring volume of model, measuring diameter of particular parts of model, smoothness or many other features in dependence of device's usage.

5.2.1 Basic Principles

The main application engine acquires data from file, which structure is described above. These data are represented with 3 coordinates in Cartesian coordinate system. By analysing the position of each point in neighbourhood, the normal vector is computed. This vector is necessary for computing shadows which creates the 3D illusion. Coordinates, normal vector and color (white for each point presently), these 3 informations are saved in structure, which is called vertex. Afterwards, the array of vertices is sent to the buffer of graphic device. By joining each 3 closest points, the triangles are compiled and array containing information which vertex belongs to which triangle is also sent to the graphic device's buffer. According these informations, the graphic device creates the model which is a set of thousands of triangles. Using the transformation matrices defining the angle of view, graphic device decides if each triangle is inside the view and eventually how to draw this triangle on the screen. This process is performed each time, when any changes in view are made.

5.3 Sample models

There are demonstrating models displayed on fig. 5.2. First three screenshots comes from models scanned using triangulation scanner MicroEpsilon. Its accuracy is

clearly visible – particular pleats of skins on fingers are visible on second image, and particular vessels are distinguishable on third image.

Last picture is taken from model measured by SICK laser scanner. As visible, it is possible to make scans of much bigger objects. However, the resolution is worse, but the important features still remains. Breathing of scanned human causes waves on created model in place of his belly. This can cause a problem when scanning living creatures. Possible solution could be speeding up of whole scanning process.

There are also visible undetectable parts on the top of the human head. There is a big dispersion of light in hair and sufficiently strong signal is not returned.

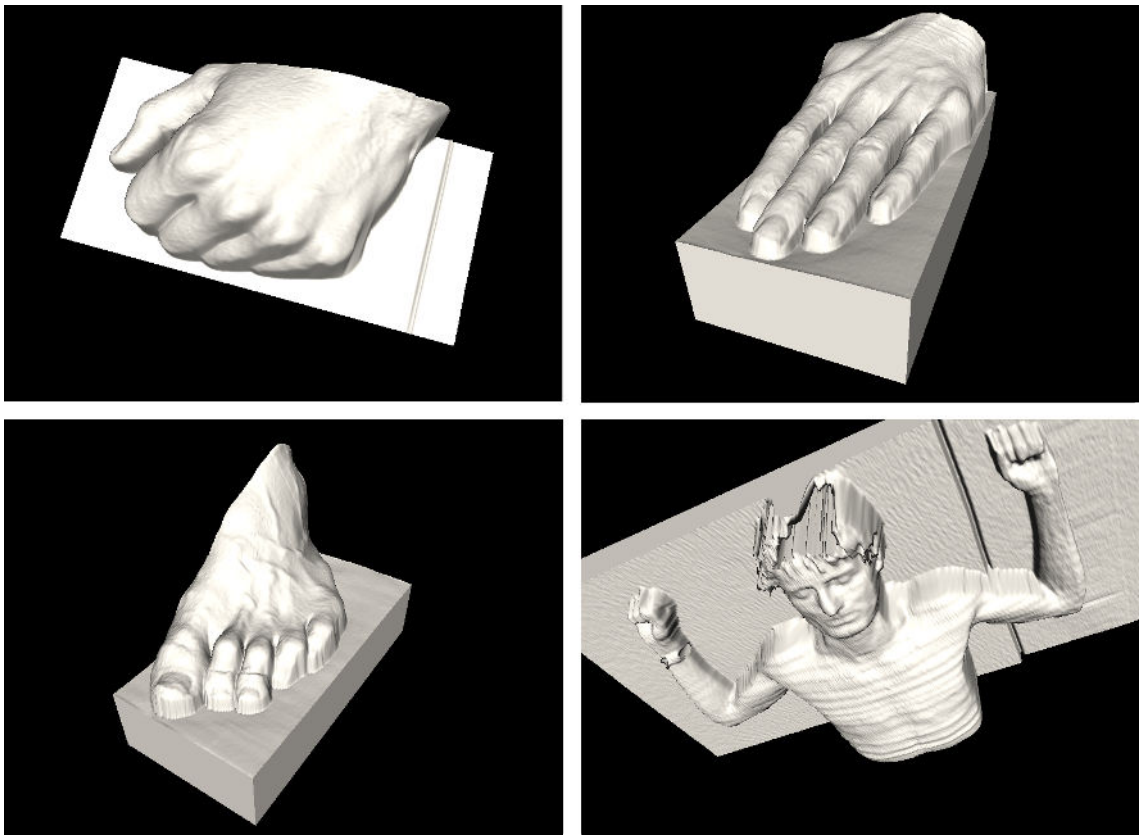


Fig. 5.2: Comparison of models from different scanners.

5.4 Summary

This chapter describes 3D model visualizing application created in XNA. It provides basic viewing operation like a translation and rotation of model and will be extended in future work. Proposed extensions could be precise measuring of volume, area, distances and other values for exact analysing of scanned objects.

6 CONCLUSION

This thesis summarizes principles of three-dimensional scanning system based on robotic manipulator and laser scanner and describes practical construction of such device. Solution is universal and not dependent on actually used devices.

This approach brings more flexibility on scanned object — almost any type of object can be scanned. In case of large structures we can use TOF laser scanner if high accuracy is not so important for us, in case of tiny structures, we can use triangulation-based scanners. If we want to scan large objects with high resolution, this could be solved by triangulation-based scanner together with wide operation area manipulator and suitably designed scanning trajectory.

Because of that, the resulting device has a wide area of usage. It can be used for scanning complex objects, like a historically valuable objects, which could be preserved by this way for next generations. It can be used also in health care domain for scanning parts of human body in order to be able to compare body status along the time. In this case, it could bring significant savings in acquisition costs and operation costs with keeping quality of service at the same level.

Result of this work is fully-functioning device with controlling software. Final 3D Scanner is just ready to be used.

This project can be combined with other sensors such as CCD camera or thermo vision, what empowers creating of coloured three-dimensional models. This could be aim of my future work. Also the next commands for trajectory planning and better surface generation would be developed in future.

BIBLIOGRAPHY

[1] ŠOLC, F.; ŽALUD, L. *Robotika*. University mimeographed, VUT Brno, 2006.

[2] *robomatrix.org* [online]. 2005, last update 2011 [cited 22. 04. 2012]. Link: <http://www.robotmatrix.org/>.

[3] *EPSON Robots - official web pages* [online]. 2011, last update 2011 [cited 26. 04. 2012]. Link: <http://www.epsonrobots.com/>.

[4] SKAŘUPA, J. *Průmyslové roboty a manipulátory*. University mimeographed, VŠB-TU Ostrava, 2007.

[5] *Epson C3 Compact 6-Axis Robot Manual* [online]. Rev. 4, 2010, last update 2010 [cited 22. 04. 2012]. Link: <http://www.robots.epson.com/downloads/manuals/EPSON_C3_Robot_Manual(R4).pdf>.

[6] SHAN, J.; TOTH, Ch. *Topographic Laser Ranging and Scanning:Principles and Processing*. CRC Press, 2008, ISBN 1420051423.

[7] PASCHOTTA, R. *Encyclopedia of Laser Physics and Technology*. [online]. 2009, last update 2010 [cited 27. 04. 2012]. Link: <http://www.rp-photonics.com/encyclopedia.html>.

[8] *LMS 400 Laser Measurement System Operating Instructions* [online]. 2006, last update 2006 [cited 28. 04. 2012]. Link: <https://www.mysick.com/saqqara/pdf.aspx?id=im0010698>.

[9] *Epson Robot Controller RC170/RC180 Manual* [online]. Rev. 13, 2011, last update 2011 [cited 28. 04. 2012]. Link: <http://www.robots.epson.com/downloads/manuals/EPSON_RC170_RC180_Controller_Manual(R13).pdf>.

[10] *Epson RC+ 5.0 User's Guide* [online]. Rev. 1, 2011, last update 2011 [cited 28. 04. 2012]. Link: <http://www.robots.epson.com/downloads/manuals/EPSON_RC+50_Users_Guide-RC170_RC180(V54R1).pdf>.

[11] FIREŠ, M. *Obslužný program pro robotický manipulátor EPSON*. Brno, Brno University of Technology, Faculty of electrical engineering and communication, 2011. 35 p. Semestral thesis, supervised by doc. Ing. Luděk Žalud, Ph.D..

[12] *Microsoft: Next Generation of Games Starts With XNA* [online]. 2004, last update 2004 [cited 28. 04. 2012]. Link: <https://www.microsoft.com/en-us/news/press/2004/mar04/03-24xnalaunchpr.aspx>.

[13] *Instruction Manual scanCONTROL* [online].
2008, last update 2010 [cited 03. 01. 2013]. Link: <http://www.micro-epsilon.cz/download/manuals/man–scanCONTROL-2700-2800-Quick-Manual–en.pdf>.

[14] *Robotics Kinematics and Dynamics* [online]. 2012, last update 2012 [cited 04. 01. 2013].
Link: <http://en.wikibooks.org/wiki/Robotics_Kinematics_and_Dynamics/Description_of_Position_and_Orientation>.

[15] VODIČKA, J. *Verbal information*. mechanical technician, Technická 12, 616 00 Brno, SE1.106 [cited 14. 05. 2013].

[16] PROCHÁZKA, P.; ROZSÍVALOVÁ, Z. *Materiály a technická dokumentace, část: Technická dokumentace*. University mimeographed, VUT Brno, 2006.

[17] CHEN, M. *A Parallel 3D Delaunay Triangulation Method*. Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium, 2011., p. 26-28

[18] *Delaunay traingulation - MATLAB delaunay* [online].
2008, last update 2013 [cited 15. 05. 2013]. Link: <http://www.mathworks.com/help/matlab/ref/delaunay.html>.

[19] MARCON, M.; PICCARRETA, L.; SARTI, A.; TUBARO, S. *Fast PDE approach to surface reconstruction from large cloud of points*. Computer Vision and Image Understanding 112, 2008., p. 274-285

LIST OF SYMBOLS, PHYSICAL CONSTANTS AND ABBREVIATIONS

c The speed of light $[m \cdot s^{-1}]$

d_{obj} Distance from laser scanner to the object $[m]$

H_{0L} Homogeneous transformation from measured data to default system $[-]$

H_{0M} Homogeneous transformation from manipulator's to default system $[-]$

H_{ME} Homogeneous transformation from end-point's to manipulator's system $[-]$

H_{ES} Homogeneous transformation from scanner's to end-point's system $[-]$

H_{SL} Homogeneous transformation from laser emitter's to scanner's system $[-]$

P_L Vector of homogeneous coordinates of measured point $[mm]$

P_0 Vector of homogeneous coordinates of point transformed into my system $[mm]$

t_{of} Time difference between laser beam's generation and detection $[s]$

x, y, z Translation of end-point in manipulator's coordinate system $[mm]$

u, v, w Rotation of end-point in manipulator's coordinate system [deg]

x_t, y_t, z_t Translation of scanner zero-point in end-point's coordinate system $[mm]$

u_t, v_t, w_t Rotation of scanner zero-point in end-point's coordinate system [deg]

x_e, y_e, z_e Translation uncertainty of laser scanner mounting $[mm]$

u_e, v_e, w_e Rotation uncertainty of laser scanner mounting [deg]

LIST OF APPENDICES

A LMS400 LASER SCANNER DRIVER OPERATING MANUAL

Author	Bc. Adam Chromý
Version	1.0.0
Release date	26. 3. 2012
Description	C# driver for configuring and controlling operation of SICK LMS400 laser scanner.
Licence	GNU GPL

The best way to explore all the functions and find out how to use this driver is to see the demonstrating program, which is also attached.

For brief overview of driver's and demonstrating program's features see the attached demonstrating video.

A.1 Description of Public Methods

Open

Syntax:

```
void Open()
```

Requires power-supplied LMS400 device connected to the driver by Ethernet. This method opens TCP/IP communication between computer and device. If device is successfully connected, `OnConnected` event occurs.

Close

Syntax:

```
void Close()
```

Terminates connection with device. If successfully disconnected, `OnClosed` event occurs.

StartMeas

Syntax:

```
void StartMeas(EMeasuredData whatToMeasure =
   EMeasuredData.DistanceOnly)
```

Starts continuous measuring process. If it is successful, `OnMeasureStart` event occurs. Data are available in `RawData` property and each time when they have been refreshed, `OnMeasured` event occurs. Continuous measurement can be stopped by `StopMeas`. Parameter `whatToMeasure` defines which value is measured (distance, remission). If omitted, only distance is measured.

TriggeredMeas

Syntax:

```
void TriggeredMeas(EMeasuredData whatToMeasure =
   EMeasuredData.DistanceOnly)
```

Starts measurement in mode where measuring is triggered by selected trigger. You can select one of available triggers by `SelectTrigger` method. For software triggering use `SingleMeas` method instead this one. Data are available in `RawData` property and each time when they have been refreshed, `OnMeasured` event occurs. Parameter `whatToMeasure` defines which value is measured (distance, remission). If omitted, only distance is measured.

SingleMeas

Syntax:

```
void SingleMeas(EMeasuredData whatToMeasure)
void SingleMeas(int count = 1)
void SingleMeas(EMeasuredData whatToMeasure, int count)
```

Performs `count` measurements of value defined by `whatToMeasure`. If `count` is omitted, one measurement is performed. Data are available in `RawData` property and each time when they have been refreshed, `OnMeasured` event occurs. If `whatToMeasure` is omitted, only distance is measured.

StopMeas

Syntax:

```
void StopMeas()
```

Stop continuous measurement (if it is actually performed) and if this action is successful, `OnMeasureStop` event occurs.

ConfigByFreqAndAng

Syntax:

```
void ConfigByFreqAndAng(int Frequency, float AngularResolution,
   float StartingAngle = 55.0F, float ScanWidth = 70.0F)
```

Configures angular resolution and scanning frequency. Not every combination of these parameters is possible, so device computes nearest possible combination and set parameters to these values. Then `OnParametersChanged` event occurs. Values of parameters which were **really** used in configuration of device can be obtained from `InfoLMS` property.

Warning: Device allows to configure value, which are out of range guaranteed by vendor. Check `ValueQuality` at `InfoLMS`, which refers to actual device precision (more at `InfoLMS` section).

ConfigByFreq

Syntax:

```
void ConfigByFreq(int Frequency, ERoughAngularValue
   AngularResolution, float StartingAngle = 55.0F, float
   ScanWidth = 70.0F)
```

Configures angular resolution and scanning frequency. Device tries to set up **exact value of scanning frequency**, appropriate angular resolution is counted by device. Then `OnParametersChanged` event occurs. Values of parameters which were **really** used in configuration of device can be obtained from `InfoLMS` property.

Warning: Device allows to configure value, which are out of range guaranteed by vendor. Check `ValueQuality` at `InfoLMS`, which refers to actual device precision (more at InfoLMS section).

ConfigByAng

Syntax:

```
ConfigByAng(ERoughFrequency Frequency, float AngularResolution,
   float StartingAngle = 55.0F, float ScanWidth = 70.0F)
```

Configures angle resolution and scanning frequency. Device tries to set up **exact value of angular resolution**, appropriate scanning frequency is counted by device. Then `OnParametersChanged` event occurs. Values of parameters which were **really** used in configuration of device can be obtained from `InfoLMS` property.

Warning: Device allows to configure value, which are out of range guaranteed by vendor. Check `ValueQuality` at `InfoLMS`, which refers to actual device precision (more at InfoLMS section).

SaveParameters

Syntax:

```
void SaveParameters()
```

Saves actual configuration parameters as default settings, which are applied to device after start up (power on). These values are saved in device's EEPROM, so no power supply is required to store them.

Run

Syntax:

```
void Run()
```

Exits configuration mode of device. This step is performed automatically when any measuring command is received.

Reset

Syntax:

```
void reset()
```

Switch off device and switch it on again. Then default values are applied to the configuration parameters. During this process, TCP/IP connection with driver is lost because of lost of power supply. It leads to device disconnecting, so it is necessary to reconnect by `Open` method after device has been restarted.

Synchronize

Syntax:

```
void Synchronize(ESyncMode SyncMode, int PhaseOffset = 0)
```

Set up synchronizing when working of two devices in pair is required. First device is configured as Master and second as Slave. Devices must be connected by System connection. See Operating Instructions for details.

MedianFilter

Syntax:

```
void EnableMedianFilter()
void DisableMedianFilter()
```

Enables/disables Median Filter. Median value is chosen from group of 9 values:

- actual value of appropriate point
- actual value of first point left
- actual value of first point right
- previous value of appropriate point
- previous value of first point left
- previous value of first point right
- second-last value of appropriate point
- second-last value of first point left
- second-last value of first point right

EdgeFilter

Syntax:

```
void EnableEdgeFilter()
void DisableEdgeFilter()
```

Enables/disables Edge Filters. The edge filter prevents incorrect/extreme distance values at edges that result from it not being possible to determine a distance value for the previous or next point (e.g. if the previous/next measured point was too dark or outside the measuring range of 3 metres).

RangeFilter

Syntax:

```
void EnableRangeFilter(float BottomLimit, float TopLimit)
void DisableRangeFilter()
```

Enables/disables Range Filter. Only values bigger than `BottomLimit` and smaller than `TopLimit` are provided. Other values are set to 0.

MeanFilter

Syntax:

```
void EnableMeanFilter(int NumberOfMeans)
void DisableMeanFilter()
```

Enables/disables Mean Filter. Mean is counted from `NumberOfMeans` last values.

Note: applying mean filter slows the measurement, the scanning frequency is then `NumberOfMeans`-times smaller then configured.

SelectTrigger

Syntax:

```
SelectTrigger(ETriggerSource StartSource, ETriggerSource
   StopSource)
```

Selects trigger to begin and end measurement. For more information see Operating Instructions.

A.2 Description of Public Parameters

RawData

Structure containing set of measured points. Each point contains values of distance, remission and angle of scanning. If measuring of parameter is not performed (depends on configuration) or measured value is incorrect (out of range, etc.), the value is 0. For recommended use see demonstrating program.

InfoLMS

Contains information about configuration of device. `InfoLMS` is a structure containing following properties:

- `UserLevel` – user level of actually connected user
- `StartTrigger`, `StopTrigger` – actually selected triggers of measuring
- `Name` – name of the device, same as Name property.
- `FirmwareVersion` – version of device firmware
- `ScanningFrequency` – actual scanning frequency. If frequency is not acquired yet, the value is 0.
- `AngularResolution` – actual scanning frequency. If frequency is not acquired yet, the value is 0.
- `ValueQuality` – quality index of actual combination of angular resolution and scanning frequency. If 10, device is operating with best quality, if 7, device is operating with precision defined in datasheet, if less than 7, device is operating ot of the range.

Name

Returns name of device. In most cases the return value will be "LMS400_XX00" depending on the device version.

Connected

Readonly parameter which returns true if device is actually connected to driver, otherwise it returns false.

A.3 Description of Events

OnConnected

Syntax:

```
public event ChangedEventHandler OnConnected;
```

Occurs when driver become connected with device by TCP/IP. When link is not successfully established, an exception is thrown.

OnClosed

Syntax:

```
public event ChangedEventHandler OnClosed;
```

Occurs when driver is successfully disconnected from the device. This event occurs also in case when device is not disconnected properly, but also exception is thrown.

OnMeasureStart

Syntax:

```
public event ChangedEventHandler OnMeasureStart;
```

Occurs when continuous measurement is successfully started by `StartMeas` method.

OnMeasureStop

Syntax:

```
public event ChangedEventHandler OnMeasureStop;
```

Occurs when continuous measurement is successfully stopped by `StopMeas` method.

OnMeasured

Syntax:

```
public event ChangedEventHandler OnConnected;
```

Occurs every time when cyclic message with measured data is received and data are processed and prepared for reading. Is recommended to use this event to further processing of received data. Event handling method receives measured data as a parameter in form of `RawDataSet` object.

OnParametersChanged

Syntax:

```
public event ParametersEventHandler OnParametersChanged;
```

Occurs when scanning frequency, angular resolution or quality index has been changed. Is recommended to use this event to acquire real values of these settings decided by device (which may vary from desired values).

OnParametersSaved

Syntax:

```
public event ChangedEventHandler OnParametersSaved;
```

Occurs when actual configuration has been set as default configuration and saved into the device memory.

OnSynchronizingChanged

Syntax:

```
public event ChangedEventHandler OnSynchronizingChanged;
```

Occurs when synchronizing has been switched on, or synchronizing role has been changed from one role to another (master <> slave).

OnFilterChanged

Syntax:

```
public event ChangedEventHandler OnFilterChanged;
```

Occurs when at least one of possible filters is switched on or off, or its parameters have been changed.

OnException

Syntax:

```
public event ExceptionEventHandler OnException;
```

Occurs when exception is thrown. Event carries information about exception stored in object of `EventArgs` type. Object contains text message describing exception and date and time when exception occured.

B MICROEPSILON LASER SCANNER DRIVER OPERATING MANUAL

Author	Bc. Adam Chromý
Version	1.0.0
Release date	16. 7. 2012
Description	C# driver for configuring and controlling operation of MicroEpsilon scanCONTROL 2750-100 laser scanner.
Licence	GNU GPL

The best way to explore all the functions and find out how to use this driver is to see the demonstrating program, which is also attached.

B.1 Description of Public Methods

ConnectETH

Syntax:

```
public bool ConnectETH(string IPaddress)
```

Connects to the device defined by `IPaddress` using Ethernet interface. This method opens TCP/IP communication between computer and device. If device is successfully connected, `OnConnected` event occurs.

ConnectSerial

Syntax:

```
public bool ConnectSerial(uint portNumber)
```

Connects to the device defined by `portNumber` using Serial interface. If device is successfully connected, `OnConnected` event occurs.

ConnectFirewire

Syntax:

```
public bool ConnectFirewire(uint nodeID)
```

Connects to the device defined by `nodeID` using Firewire interface. If device is successfully connected, `OnConnected` event occurs.

Open

Syntax:

```
public void Open()
```

Connects to device using Ethernet and default IP defined in constants. If successfully disconnected, `OnClosed` event occurs.

Disconnect, Close

Syntax:

```
public bool Disconnect()
public void Close()
```

Terminates connection with device. If successfully disconnected, `OnClosed` event occurs and in case of `Disconnect` also returns true.

StartMeas

Syntax:

```
public bool StartMeas()
```

Starts continuous measuring process. If it is successful, `OnMeasureStart` event occurs and true is returned.

StopMeas

Syntax:

```
public bool StopMeas()
```

Stop continuous measurement (if it is actually performed) and if this action is successful, `OnMeasureStop` event occurs and true is returned.

GetValues

Syntax:

```
public XZpoint[] GetValues()
```

Returns set of points from one actual profile. Data are returned in form of array of public structures `XZpoint`

IsAlive

Syntax:

```
public bool IsAlive()
```

Returns true if driver is actually connected to device.

B.2 Description of Public Parameters

Interface

Returns actually selected interface for communication.

Resolution

Number of points returned in profile.

PacketDelay

Minimal time between two packets in us [0-1000 allowed]. Neccessary when more devices connected.

DeviceType

Type (version) of connected device.

Name, DeviceName

Returns name, type and version of connected device.

FirmwareVersions

Returns array of DSP, FPGA1 and FPGA2 firmware versions.

SerialNumber

Returns serial number of device.

ShutterTime

Defines time when shutter is on. Defined in number of 10us, allowed range 1 .. 4095 (eg. 10..40950us).

IdleTime

Defines time when shutter is off. Defined in number of 10us, allowed range 1 .. 4095 (eg. 10..40950us).

AutoShutter

Enables automatic regulation of shutter time depending on scanned object.

AmbientLightSupression

Enables supression of ambient light.

VideoFilter

Enables video filter.

ReflectionThreshold

Threshold for the selection of reflections. At targets with several reflections this may lead to better results.

Connected

Returns true if driver is actually connected to device.

Measuring

Returns true if device is actually measuring. Some functions are disabled then.

InvalidPointsFilter

Enables interpolating of invalid points.

LaserPower

Enables interpolating of invalid points.

XRange

Sets range of scanning in X axis.

MedianFilter

Enables/disables median filter.

AverageFilter

Enables/disables average filter.

Params

Returns actual configuration of device packed in one object.

B.3 Description of Events

OnConnected

Syntax:

```
public event ChangedEventHandler OnConnected;
```

Occurs when driver become connected with device by TCP/IP. When link is not successfully established, an exception is thrown.

OnClosed

Syntax:

```
public event ChangedEventHandler OnClosed;
```

Occurs when driver is successfully disconnected from the device. This event occurs also in case when device is not disconnected properly, but also exception is thrown.

OnMeasureStart

Syntax:

```
public event ChangedEventHandler OnMeasureStart;
```

Occurs when continuous measurement is successfully started by `StartMeas` method.

OnMeasureStop

Syntax:

```
public event ChangedEventHandler OnMeasureStop;
```

Occurs when continuous measurement is successfully stopped by `StopMeas` method.