Optical measuring system using a camera and laser fan-out for narrow mounting on a miniaturized submarine

Martin Berglund
In loving memory of my grandfather Erik Sundström
Abstract

Optical measuring system using a camera and laser fan-out for narrow mounting on a miniaturized submarine
Martin Berglund

The aim was to develop, manufacture and evaluate diffractive lenses, or diffractive optical elements (DOE), for use in correlation with a camera to add perspective in pictures. The application is a miniaturized submarine developed in order to perform distant exploration and analysis in harsh and narrow environments. The idea is to project a laser pattern upon the observed structure and thereby add geometrical information to pictures acquired with an onboard CMOS camera. The design of the DOE-structures was simulated using the optimal rotational angle method (ORA). A set of prototype DOEs were realized using a series of microelectromechanical system (MEMS) processes, including photolithography, deposition and deep reactive-ion etching (DRIE). The projected patterns produced by the manufactured DOEs were found to agree with the simulated patterns except for the case where the DOE feature size was too small for the available process technology to handle. A post-processing software solution was developed to extract information from the pictures, called Laser Camera Measurement (LCM). The software returns the x, y and z coordinate of each laser spot in a picture and provides the ability to measure a live video stream from the camera. The accuracy of the measurement is dependent of the distance to the object. Some of the patterns showed very promising results, giving a 3-D resolution of ~0.6 cm, in each dot, at a distance of 1 m from the camera. Lengths can be resolved up till 3 m distance from the submarine.
Sammanfattning

Optical measuring system using a camera and laser fan-out for narrow mounting on a miniaturized submarine

Martin Berglund

Tillämpningen finns i en miniatyriserad ubåt framtagen för utforskning och analys av svåråtkomliga och trånga håligheter. Målet var att designa, tillverka och utvärdera en diffraktiv lins (DOE) för användning tillsammans med en kamera för att skapa perspektiv i bilder. Idén var att projicera ett lasermönster på objektet och därmed lägga till geometrisk information till bilderna tagna med CMOS kameran.

Utformningen av DOE-strukturen simulerades med the optimal rotational angle method (ORA). En uppsättning av prototyp DOE-linser tillverkades med hjälp av en serie mikrostrukturteknikprocesser, bland annat fotolitografi, deponering och plasmaetsning. Mönster projicrerade med de tillverkade DOE-linserna stämde väl överens med önskade mönster, med undantag för de DOEs där strukturstorleken underskred processens begränsningar. En programvara, kallad Laser Camera Measurement (LCM), utvecklades för att extraera information från bilderna. Programvaran returnerar x, y, och z koordinaterna för varje laserpunkt i en bild och ger möjlighet att mäta i en kontinuerlig videoström från kameran. Mätosäkerheten är beroende av avståndet till objektet. Vissa mönster gav mycket lovande resultat, med en 3-D upplösning på ~0.6 cm, i varje punkt, på ett avstånd av 1 m från kameran. Längder kan upplösas upp till 3 m från kameran där ett så kallat far-field uppstår.
# Contents

1 Introduction 1
   1.1 History of DOE technology ......................................... 1

2 Theory 5
   2.1 Diffraction .............................................................. 5
      2.1.1 Scalar diffraction theory ....................................... 5
      2.1.2 Huygens-Fresnel principle ..................................... 7
      2.1.3 Fresnel approximation .......................................... 8
      2.1.4 Fourier scalar diffraction ..................................... 9
   2.2 Kinoform .............................................................. 11

3 Simulation 13
   3.1 Iterative Fourier Transform Algorithm ............................. 15
   3.2 Optimal-Rotational-Angle Method ................................... 15

4 Manufacturing 19
   4.1 Detailed processes ................................................... 19

5 Design 23
   5.1 Projected patterns .................................................... 23
   5.2 Evaluation method .................................................... 23
      5.2.1 Setup ............................................................... 23
      5.2.2 LCM Software ..................................................... 23

6 Results 31
   6.1 Evaluation of structures ............................................. 31
   6.2 Fan-out and noise ..................................................... 34
   6.3 Camera + DOE evaluation ............................................. 34
   6.4 Module in water ....................................................... 34
   6.5 Etch rates of SiO$_2$ .................................................. 37

7 Discussion 39
   7.1 Manufacturing process ............................................... 39
   7.2 Design ................................................................. 40
   7.3 Software ............................................................... 40
   7.4 Future improvements ................................................. 40

8 Conclusions 43

9 Acknowledgements 45
References

A DOECAD - Source

B LCM - Source

B.1 LCM_gui.m ......................................................... 87
B.2 FFA_init.m ........................................................... 94
B.3 FFA_ident.m .......................................................... 94
B.4 LCM_beams.m .......................................................... 95
B.5 LCM_calib.m ............................................................ 95
B.6 LCM_cam_angle.m ....................................................... 95
B.7 LCM_frame_trigger.m .................................................. 96
B.8 LCM_genpic.m ........................................................... 96
B.9 LCM_ident.m ............................................................. 97
B.10 LCM_interp.m .......................................................... 98
B.11 LCM_linematch.m ...................................................... 98
B.12 LCM_live.m ............................................................. 99
B.13 LCM_measure.m ......................................................... 100
B.14 LCM_hierarchy .......................................................... 101
Chapter 1

Introduction

In the project Deeper Access, Deeper Understanding (DADU) a miniaturized submarine, which can perform distant exploration and analysis in harsh and narrow environments, is developed. It will be able to reach previously unexplorable sub-glacial lakes through narrow glacial bore holes, to explore and make measurements. The submarine itself is not larger than a 50 cl soda bottle[1], bringing with it instruments such as a camera, sonar [2] and a particle sampler. The communication with the submarine is done via a fiber optical cable, which can reach several hundreds of meters and will supply real time control, bi-directional data transfer and power transfer. A first prototype of the submarine can be viewed in Figure 1.1.

Pictures captured with a normal camera are without perspective. This is a problem with most underwater imaging, the pictures appear flat and do not convey how far away the object is, its shape or size. In order to add this extra information to the pictures one can project a well known pattern onto the object. This pattern has to be in focus independent of the distance from the camera within a reasonable region, and therefore has to be collimated. In order to achieve this a Diffractive Optical Element (DOE) can be used that re-shapes a laser beam into a fan-out pattern.

A DOE is a lens that changes the phase and/or amplitude of the incoming beam in order to change its propagation properties, thus giving the advantage of being able to redistribute the intensity of the beam into a desired pattern. A kinoform is a DOE that modulates only the phase of an incoming coherent light-wave, preferably a laser, and splits the beam into a desired pattern.

This master thesis describes the design, manufacturing and evaluation of several such different kinoform prototypes. These kinoform lenses are to project a laser pattern to give perspective in pictures taken with the camera on-board the DADU submarine.

1.1 History of DOE technology

Throughout history light has been a central part of philosophy and science, and the theories of the lights nature have always been debated. In ancient India, according to the philosophical schools of Samkhya, from around the 6th–5th century BC, light is one of five fundamental elements, while at the same time the Vaisheshika school taught light to be a stream of fire-atoms.

Empedocles, a Greek philosopher (490–430 BC), postulated that light-rays came out of the observers eyes and interacted with those from the sun. This became the generally accepted theory of light for a long time to come. Euclid, another Greek philosopher (325–265 BC), is the first known author who studied optics from a geometrical point of view, but he
didn’t touch on the subject of what light consisted of or where it came from. In year 984 the Persian mathematician, Ibn Sahl (940-1000), wrote *On Burning Mirrors and Lenses* in which he describes a mathematical formula for optical refraction, similar to Snell’s law.

Later Ibn al-Haytham, (965–1040), formulated the first comprehensive and systematic alternative to Greek optical theories [3], and with his book *Kitab al Manazir* (*Book of optics* in English) he laid the foundations for modern optics. Ibn al-Haytham’s achievements in optics were numerous. Among other things he revolutionized the way of visualizing light. He insisted that vision only occurred because of rays entering the eye from outer sources.

Up until now lenses were made out of water-filled glass bulbs and polished crystals, often quartz. During the middle ages glass-lenses were invented and enabled more variations in the applications, and the first wearable glasses were invented 1284 by the Italian scientist Salvino D’Armate (1258–1312).

These kinds of lenses were possible to design by treating light as rays which travel at straight lines in a homogeneous medium until entering a medium with another refractive index. At the interface the ray is bent according to Snell’s law, this being empirically discovered by Willebrord Snell (1580–1626), in 1621.

At this time in history effects that couldn’t be explained with particle rays, called diffraction, were studied by Francesco Grimaldi (1618–1663) by illuminating a rod with a small lamp source and observing that the shadow didn’t have sharp edges as expected. This effect wasn’t explained until light was described as an electromagnetic wave. One of the first scientists who actively proposed that light was a wave was Christiaan Huygens (1629–1695). Huygens stated that each point on the wavefront of a disturbance can be considered to be a new disturbance (see Figure 1.2). This is a fundamental insight in order to explain diffractive optics. However the general physical interpretation of light at the time was that it was a stream of particles and this theory was supported by among others the notable Isaac Newton (1643–1727) who rejected the wave theory.

In the beginning of the 19th century the theory of light being a wave was confirmed by Thomas Young’s (1773–1829) famous interference experiment and independently confirmed by Augustin Fresnel (1788–1827). The idea was a radical one at the time, because it stated...
CHAPTER 1. INTRODUCTION

1.1. HISTORY OF DOE TECHNOLOGY

Figure 1.2: The figure shows how the disturbance from each point on a primary wavefront creates secondary wavefronts that adds up to a tertiary wavefront according to Huygens.

that under proper conditions, light could be added to light and produce darkness.

James Maxwell (1831–1879) later identified light as an electromagnetic wave when studying electromagnetism, after finding that the group velocity calculated from the empiric vacuum parameters; permeability $\mu_0 = 4\pi \cdot 10^{-7} \frac{\text{Vs}}{\text{Am}}$ and permittivity $\varepsilon_0 \approx 8.854 \cdot 10^{-12} \frac{\text{As}}{\text{Vm}}$ were the same as the speed of light in vacuum $c_0 \approx 2.998 \cdot 10^8 \frac{\text{m}}{\text{s}}$

$$c_0 = \frac{1}{\sqrt{\varepsilon_0 \mu_0}} \quad (1.1)$$

This was a discovery of enormous importance which explained the interference and diffraction effects, but this wasn’t obvious at first.

Gustav Kirchhoff (1824–1887) was the first one attempting to explain the effects discovered by Fresnel and Young. Kirchhoff’s work was then further developed by Arnold Sommerfeld (1868–1951), who by removing one of Kirchhoff’s approximations developed the Rayleigh-Sommerfeld diffraction theory.

Both the Kirchhoff and the Rayleigh-Sommerfeld theory share some major simplifications in order to be simple solutions to the Maxwell equations. Most importantly, the electromagnetic field is treated as a scalar field. Fortunately these theories yield good results under the following two conditions [4]:

- The diffracting aperture must be large compared with the wavelength
- The diffracting fields must not be observed too close to the aperture, i.e. only valid in the far-field.

Maxwell’s discovery that light was in fact electromagnetic waves consequently gave rise to the theory of the aether, a medium in which light was thought to propagate in. After several failed experiments to prove the existence of the aether, the theory was more or less disproven and later in 1905 Albert Einstein (1879–1955) introduced his Special Theory of Relativity where he postulated the Principle of Invariant Light Speed [5]

Any ray of light moves in the “stationary” system of coordinates with the determined velocity $c_0$, whether the ray be emitted by a stationary or by a moving body.

This means that electromagnetic waves could propagate through free space and has the same speed in all reference frames. This meant the end of the aether theory.

3
Einstein’s analysis of the photoelectric effect demonstrated that light also possessed particle-like properties. This was further confirmed by Compton scattering discovered by Arthur Compton (1892–1962) in 1927. This Particle-wave duality was explained by Louis de Broglie (1892–1987) the same year and confirmed two years later when diffraction of electrons showed that all matter possess this duality.

Light can therefore be seen as photons, mass-less energy packets, that like all the other elementary particles have both wave and particle characteristics. To understand the nature of light it is easiest to explain light as waves when considering propagation and as particles when considering absorption and emission of light.

With the development of quantum physics in the first half of the 20th century came the laser and in 1969 the first kinoform was constructed. Designing kinoforms requires large computational power and has become easier as computer power has grown.
Chapter 2

Theory

2.1 Diffraction

In optics one should not confuse the phenomenon \textit{diffraction} with \textit{refraction}. Refraction can be defined as the bending of light rays that occur when they pass through a region in which there is a gradient of the local velocity of propagation of the wave. The most common case of this phenomenon is when the light ray enters a medium with a different refractive index thus changing the propagation velocity. The incident light is bent at the interface according to \textit{Snell’s law}

\[ n_1 \sin \theta_1 = n_2 \sin \theta_2, \]

(2.1)

where \( n_1 \) and \( n_2 \) are the refractive indexes, \( \theta_1 \) and \( \theta_2 \) are the refractive angles from the surface normals for medium 1 and 2 respectively. Figure 2.1a shows an example of this effect where \( n_1 < n_2 \) which gives \( \theta_1 > \theta_2 \).

Diffraction on the other hand has been defined by Sommerfeld\cite{Sommerfeld} as

“any deviation of light rays rectilinear paths which cannot be interpreted as reflection or refraction”.

The most common example of diffraction is when an obstacle, or aperture, is introduced in a monochromatic light-wave and the light “bends” around the obstacle. Another example is a grating that is illuminated by a monochromatic and coherent light-beam i.e. a laser, and the one spot falling upon the grating becomes a series of fan-out dots radiating out of the grating. The equation

\[ d \sin \theta_m = m\lambda, \]

(2.2)

where \( d \) is the grating constant, or the center-to-center distance between the openings in the grating, and \( \theta_m \) is the angle from the spot \( m=0 \) (called the zeroth order) to a spot with the index \( m \), where \( m \) is an integer \( \neq 0 \), Figure 2.1b.

2.1.1 Scalar diffraction theory

In order to predict diffraction pattern from complex boundary conditions one needs a sufficient model of how light behaves. The mathematical representation of light is obtained by
solving Maxwell’s equations in regions free of charges and currents.

\[ \nabla \cdot \mathbf{E} = 0, \quad (2.3) \]
\[ \nabla \cdot \mathbf{H} = 0, \quad (2.4) \]
\[ \nabla \times \mathbf{E} = -\mu_0 \frac{\partial \mathbf{H}}{\partial t}, \quad (2.5) \]
\[ \nabla \times \mathbf{H} = \varepsilon \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t}, \quad (2.6) \]

where \( \mathbf{E} \) is the electric vector field, \( \mathbf{H} \) is the magnetising vector field, \( \mu, \varepsilon \) is the regions relative permeability and permittivity. Maxwell’s equations can be solved in different ways and with different approximations. The unapproximated solution gives a complete description of the \( \mathbf{E} \) and \( \mathbf{H} \) field, however this is seldom needed and some simplifying approximations can be done. Some assumptions can be made upon the dielectric in which we solve Maxwell’s equation:

- Isotropic. The properties of the media are independent of the direction of the polarization of the wave.
- Homogeneous. The permittivity is constant in the media.
- Non-dispersive. The permittivity is independent of the waves wavelength.
- Nonmagnetic. The magnetic permeability is always equal to vacuum permeability, i.e. \( \mu = 1 \).

To solve Maxwell’s equations, with stated simplifications, one starts with applying the curl operator, \( \nabla \times \), of (2.5) and utilizing relation (F.99) in reference [7].

\[ \nabla \times (\nabla \times \mathbf{E}) = \nabla (\nabla \cdot \mathbf{E}) - \nabla^2 \mathbf{E} = -\mu_0 \frac{\partial (\nabla \times \mathbf{H})}{\partial t}. \quad (2.7) \]
Using (2.3), (2.6) and (2.7), the following expression is obtained

\[
\nabla^2 E = \mu_0 \varepsilon_0 \frac{\partial^2 E}{\partial t^2},
\]

(2.8)

where \( \varepsilon_0 \mu_0 = \frac{1}{c_0^2} \) and \( \varepsilon = n^2 \). The constant \( c_0 \) is the speed of light in vacuum and \( n \) is the refractive index of the media. Thus, the equation (2.8), can be rewritten as

\[
\nabla^2 E - \frac{n^2}{c_0^2} \frac{\partial^2 E}{\partial t^2} = 0,
\]

(2.9)

which can be solved for each of \( E \)'s components individually. These components are scalars and (2.9) becomes a system of scalar wave equations

\[
\begin{align*}
\nabla^2 E_x - \frac{n^2}{c_0^2} \frac{\partial^2 E_x}{\partial t^2} &= 0 \\
\nabla^2 E_y - \frac{n^2}{c_0^2} \frac{\partial^2 E_y}{\partial t^2} &= 0 \\
\nabla^2 E_z - \frac{n^2}{c_0^2} \frac{\partial^2 E_z}{\partial t^2} &= 0.
\end{align*}
\]

(2.10)

This is only possible when components of \( E \) doesn't couple to each other which is the case if the propagation media is homogeneous. If all the assumptions earlier are fullfilled it is possible to summarize the behavior of each contribution of all components of \( E \) through one scalar wave equation,

\[
\nabla^2 u (r, t) - \frac{n^2}{c_0^2} \frac{\partial^2 u (r, t)}{\partial t^2} = 0.
\]

(2.11)

The next step is to make the equation time-independent. A light wave will always be oscillating with the frequency \( f = \frac{\omega}{2\pi} = \frac{c}{\lambda} \). This means that one can make the ansatz

\[
u (r, t) = U (r) e^{-i\omega t},
\]

(2.12)

where \( U (r) \) is the time-independent part of the scalar solution. Insertion of this into equation (2.11) gives

\[
(\nabla^2 + k^2) U (r) = 0,
\]

(2.13)

where \( k = \frac{2\pi}{\lambda} \) is called the wavenumber. This equation is called Helmholtz Equation and describes the time-independent amplitude and phase of the \( E \)-field in a source-free space that follows the approximations made earlier.

### 2.1.2 Huygens-Fresnel principle

When considering diffraction of light by an opaque screen with an aperture, Figure 2.2, boundary conditions are crucial. Using Green's Theorem one can reduce a volume integral into a surface integral. This theorem can be stated as follows [4]:

Let \( U (r) \) and \( G (r) \) be any two complex-valued functions of position, and let \( S \) be a closed surface surrounding a volume \( V \). If \( U, G \) and their first and second partial derivatives are single-valued and continuous within and on \( S \), then we have

\[
\iiint_V (U \nabla^2 G - G \nabla^2 U) \, dv = \iint_S \left( U \frac{\partial G}{\partial n} - G \frac{\partial U}{\partial n} \right) \, ds,
\]

(2.14)
where $\partial/\partial n$ signifies a partial derivative in the outward normal direction at each point on $S$.

Using $G$ and $U$ as solutions to Helmholtz equation (2.13), simplifying and integrating [6], the Huygens-Fresnel principle is obtained

$$U(r_0) = \frac{1}{i\lambda} \iiint_{\Sigma} U(r_1) \frac{\exp(ikr_{01})}{r_{01}} \cos \theta ds.$$  \hspace{1cm} (2.15)

$U(r_0)$ is the observed field in a point $r_0$ expressed as a superposition of diverging spherical waves originating from secondary sources $U(r_1)$ located within the aperture $\Sigma$. $G(r)$ is set to a radiating source $\sim \exp(ikr) \cos \theta$ which is one of the possible solutions to Helmholtz equation (2.13). $r_{01}$ is the vector from point $r_0$ in the diffraction plane to the point $r_1$ in the aperture plane. $\theta$ is the angle between $r_{01}$ and the aperture normal $n$ which is close to zero in the far-field and the cosine term is often approximated to be one since the angle $\theta$ is small.

### 2.1.3 Fresnel approximation

To simplify the Huygens-Fresnel principle one introduces an approximation of the distance $r_{01}$ between the two planes $U(r_1) \equiv U(x, y)$ and $U(r_0) \equiv U(\xi, \eta)$. This approximation is based on the binomial expansion of $\sqrt{1 + b}$, where $b$ is less than unity, which is given by

$$\sqrt{1 + b} = 1 + \frac{1}{2} b - \frac{1}{8} b^2 + .... \hspace{1cm} (2.16)$$
Using this expansion and only using the two first terms the following expression is obtained

\[ r_{01} = z \sqrt{1 + \left( \frac{x - \xi}{z} \right)^2 + \left( \frac{y - \eta}{z} \right)^2} \approx z \left[ 1 + \frac{1}{2} \left( \frac{x - \xi}{z} \right)^2 + \frac{1}{2} \left( \frac{y - \eta}{z} \right)^2 \right]. \]  

(2.17)

\( z \) is the distance between the two planes.

Insertion of this approximation into (2.15) and factorising the term \( e^{ikz} \) outside the integral, yields

\[
U(x, y) = e^{ikz} i\lambda z \hat{\Sigma} \left\{ U(\xi, \eta) e^{i\frac{k}{2\lambda}(\xi^2 + \eta^2)} \right\} e^{-i\frac{2\pi}{N_{\Sigma}}(x\xi + y\eta)} d\xi d\eta.
\]

(2.18)

which can be recognized as a Fourier transform of the complex field multiplied with a quadratic phase exponential.

### 2.1.4 Fourier scalar diffraction

The quadratic phase exponentials in (2.18) can be included in the expressions \( U(\xi, \eta) \) and \( U(x, y) \) since this only is a part of the two unknown fields. The amplitude factors, \( \frac{e^{ikz}}{i\lambda z} \), can be set equal to one. This is an unphysical approximation that states that the amplitude of the light-wave does not diminish as the wave propagates and that the first and second order maxima’s will have the same intensity. However, these factors are irrelevant when only predicting the diffraction pattern; the intensity will only be relative, i.e. the pattern will look the same at different distances only with less power density. Applying these approximations to (2.18) the following expression is obtained

\[
U(x, y) = \int_{\Sigma} U(\xi, \eta) e^{-i\frac{2\pi}{N_{\Sigma}}(x\xi + y\eta)} d\xi d\eta.
\]

(2.19)

Comparing this expression with a Fourier transform between two-dimensional complex planes \( F[\mathbb{C}^2] \rightarrow \mathbb{C}^2 \), Figure (2.3), one could identify (2.19) as a Fourier transform of the source. Using discrete Fourier transform and introducing the discrete dimensionless coordinates \( a \) and \( b \) on the two surfaces, (2.19) substitutes into the following expression

\[
\hat{U}(a_1, a_2) = \int_{\Sigma} U(b_1, b_2) \exp \left( \frac{2\pi}{N_{\Sigma}} (a_1 b_1 + a_2 b_2) \right) db_1 db_2,
\]

(2.20)

where \( N_{\Sigma} \) is the number of points in one row or column of the aperture \( \Sigma \), which is set to be quadratic. When the phase and amplitude is known in the aperture area the resulting diffraction pattern projected in the far-field can be determined. However, because of the properties of Fourier transforms this formula can be reversed and the original state in the aperture plane can be determined if the projected pattern is known.

\[
U(b_1, b_2) = \int \hat{U}(a_1, a_2) \exp \left( -i\frac{2\pi}{N_{\Sigma}} (a_1 b_1 + a_2 b_2) \right) da_1 da_2
\]

(2.21)

There are some restrictions to this method:

- Diffraction field \( \psi \) have to be in the far-field. This means that this is not valid close to the aperture
- Incident light has to be monochromatic
Figure 2.3: Visualization of the two complex Fourier planes that represents the aperture plane $\Sigma$ and the diffraction plane $\psi$. 
• Diffraction plane $\psi$ only describes the zeroth order diffraction pattern, although this pattern repeats itself for each higher order.

• Intensity in the diffraction plane is not dependent on the distance, which means that the total power that falls upon the aperture is the same as that in the diffraction plane independent of the distance.

• Aperture shouldn’t have features smaller than a few wavelengths.

• Fan-out angle of the pattern is determined by the feature size of the aperture in relation to the wavelength of the incident light.

The fan-out angle from a kinoform, i.e., the angle from the center of the diffraction pattern to one of its sides, $\theta$, can be determined using the formula of the first diffractive minimum of a grating $b \sin \theta' = \lambda$, where $b$ is the feature size and $\theta'$ is the angle of a feature in the diffraction plane. The total height of the fan out can now be expressed as $h = N_s \tan \left( \arcsin \frac{\lambda}{b} \right)$, where $h$ is the total height and $N_s$ is the number of points, or pixels, in the kinoform. This gives the following expression for the angle from the center to one of the edges:

$$\theta = \arctan \left( \frac{n}{2} \tan \left( \arcsin \frac{\lambda}{b} \right) \right),$$

where $n$ is the number of features in the aperture plane and $b$ is the size of the features.

### 2.2 Kinoform

Normally in optics, diffraction effects are accomplished with apertures that modulates the intensity through slits and gratings. A kinoform is a diffractive optical element that modifies the phase of the light incident upon it. This means that the boundary condition on the $\Sigma$-plane will look like

$$U(b) = U_0 \exp \left( -i \phi(b) \right),$$

where $U_0$ is a constant and $\phi(b)$ is the phase-shift after the light has passed through the kinoform. In this case one approximates the incident light as having constant intensity over the whole surface. Usually one also set the incident lights phase to be constant so that the phase-shift in the outgoing light-wave is the same as the kinoform, Figure 2.4. This phase-shift is usually produced by letting the light travel different distances through a media with higher diffractive index than air, thus giving the wavefront new phases in each pixel.
Figure 2.4: Incoming plane wave from the left. Phase-shifting kinoform in the middle. Near-field phase-shifted light wave coming out to the right.
Chapter 3

Simulation

When designing a kinoform from a desired diffraction pattern a complication occurs; a kinoform only changes the phase $\phi$ of the wave and not the amplitude. It is therefore not possible to set the amplitude $U(a)$ to a pattern of real amplitudes and then transform it back to the kinoform-plane using Formula (2.21). This is because the resulting $U(b)$ will have an amplitude-shift in addition to a phase-shift. This could be solved by making the aperture both shift phase and amplitude e.g. with filters, but this would complicate the manufacturing considerably and intensity would be reduced.

Instead the phaseshift resulting in the amplitude closest to the desired pattern has to be determined. In this case, where the desired pattern is complicated, the phase-shift is determined by simulation on a computer. There are several algorithms developed to simulate this. Two of these algorithms were used in a MATLAB-application developed in the course of designing the kinoform for this thesis. The script is called *Diffractive Optical Element Computer Aided Development* (DOECAD) and utilizes MATLABs GUIDE-system to produce the graphical user interface (GUI) seen in Figure 3.1.

DOECAD features:

- 2-D & 3-D visualization of both the Kinoform phase-shift and the resulting intensity
- Export and import capabilities of the arrays into
  - Gray-scale bitmaps (*.bmp)
  - Matlab arrays with absolute values (*.mat)
- Additional export capabilities of the kinoform design into the Auto-CAD open file format (*.dxf)
  - As an exact 3-D cad blueprint of the resulting surface
  - As a layer-based blueprint for lithography masks
- Two different algorithms for simulating a kinoform design
  - Iterative Fourier Transform Algorithm (IFTA)
  - Optimal Rotational Angle method (ORA)
- Real-time progress updates while simulating
- A calculator for manufacturing parameters such as height and width of kinoform features.
- Quick explanatory help buttons to explain some of the features and a tutorial to show the basic work-flow.
Figure 3.1: A picture of DOECAD GUI during simulation.
3.1 Iterative Fourier Transform Algorithm

IFTA is the most basic form of simulating a kinoform. The principle is to transform a desired diffraction plane \( \hat{U}_0(a) \in \mathbb{R}^2 \) into the phase-plane. Using only the phase \( \phi(b) \) gained from this transformation, the phase in the diffraction plane is determined through inverse transformation. Multiplying the desired intensity with the calculated phase a more realistic \( \hat{U}(a) \) is obtained. When the diffraction plane \( \hat{U}(a) \) contains a phase it becomes far more likely to produce an intensity-independent kinoform. This is iterated a number of times to get good stability, Figure 3.2.

One of the primary advantages with IFTA is that when used in combination with the Fast Fourier Transform (FFT), a computational algorithm, this algorithm becomes a very fast one. The disadvantages with using IFTA is that it produces fairly low quality results no matter how many iterations are done, the algorithm doesn’t converge after a few hundred iterations, the result doesn’t improve and can even revert backwards a small amount. Another disadvantage is that the resulting phase-shift is continuous which makes manufacturing harder and degrades the result even more when only an approximation of the phase can be accomplished.

The IFTA algorithm is usually better for a few symmetrical dots in the diffraction plane and not very good with continuous illuminated areas. A symmetric pattern produces a symmetrical result and usually a more continuous pattern where the phase does not look like white noise, Figure 3.3.

3.2 Optimal-Rotational-Angle Method

The ORA method [8] was developed by Jörgen Bengtsson in 1994. The main idea is to calculate the contribution from each pixel individually and optimize its phase to produce as high and even amplitude as possible in each desired spot in the diffraction plane. The
3.2 OPTIMAL-ROTATIONAL-ANGLE METHOD  

CHAPTER 3. SIMULATION

Figure 3.3: The resulting phase-shifts, calculated with IFTA, from the same symmetrical dot-pattern, centered in (a) and moved a few pixels off center in (b). The phase is represented by gray-scale where white is a full phase-shift, $\phi = 2\pi$, gray is half phase-shift $\phi = \pi$ and black is no phase-shift, $\phi = 0$. The effect of having a symmetrical pattern is obvious when comparing the results, (a) almost only contains two phase-shifts, 0 (black) and $\pi$ (gray), whereas (b) is more of a discontinuous blur. This has significant impact of how easy the pattern is to realize.

The sum of all the complex contributions can be visualized as a chain of vectors in the complex plane, See Figure 3.4. Afterwards each pixel in the kinoform is evaluated individually. Each pixel in the kinoform only contributes with one link to each amplitude-chain in the diffraction plane. The algorithm then steps through each pixel in the kinoform and changes it to the optimal phase. The optimal phase is the phase that gives the best quality, $Q$, according to

$$Q = \sum_n (A_n - A_{n,\text{org}})^2,$$

where $A_n$ is the real-valued amplitude in each spot $n$ in the diffraction-plane that the current phase of the kinoform produces. $A_{n,\text{org}}$ is the desired amplitude in each spot. When changing the phase of one pixel, $Q$ varies. The lower the value of $Q$ is, the better.

Hence only the amplitude in each illuminated spot is evaluated and the dark areas are just ignored. If the sum of the desired amplitude $A_{n,\text{org}}$ is chosen to be too low there is going to be a lot of noise in the diffraction-plane, while if chosen too high, the chains of intensity contributions will never reach the desired amplitude in the whole picture, thus creating an
CHAPTER 3. SIMULATION  
3.2. OPTIMAL-ROTATIONAL-ANGLE METHOD

Figure 3.4: Illustrates the adding of small complex contributions from each point in the kinoform to the amplitude in a point in the diffraction plane. One can also observe the amplitude difference between points $A_1$ and $A_2$ when changing the phase from $\phi_1$ to $\phi_2$ of a single pixel contribution.

uneven distribution of intensity. The process of checking each pixel in the kinoform plane is iterated until the desired quality of the picture is reached.

One of the features of this method is that the different phases the kinoform can assume, or rather have to assume, a restricted set of phases.

One problem that occurs is that the zeroth order, the pixel in the absolute middle of the picture in the diffraction plane, gets a large amount of intensity if its not suppressed. The best way of suppressing the zeroth order is to, if the spot is not a part of the desired pattern, add it to the desired pattern with the desired intensity $A_{\text{zeroth order}} = 0$. This means that the zeroth order spot is weighted equally as all the other illuminated spots.

The fact that each pixel in the phase has to be evaluated individually in a number of iterations means that this method is slow and the workload increases with each bright spot and at higher resolutions of the aperture plane $\Sigma$.

Some of the advantages of the ORA method are:

- Intensity in each spot can be set individually
- Zeroth order suppression
- A quality goal that ends the algorithm when done
- Discrete phase-shifts
- Converges toward a better result in each iteration

These advantages outweighed the long simulation time and therefore this was the method used to design the realized kinoforms in this thesis. An example of the result from the ORA method compared to the IFTA can be observed in Figure 3.5. One can observe that the
Figure 3.5: The resulting kinoform-phase from the ORA method. This is the same diffraction pattern that was used for Figure 3.3. The phase was calculated using only two phases $0$ and $\pi$, for the sake of comparison with the IFTA result.

phase is much smoother, but not centered, as in the case of the IFTA result. This is because the initial phase in the ORA-algorithm was randomized.

Another interesting property of a kinoform is that if the pattern is repeated, it will fit perfectly to itself and the joints will not be distinguishable. Because of this property the pattern can be shifted arbitrarily. This feature is also used to create large kinoforms from small patterns. The pattern is simply repeated over and over until a large enough area is created, hence one only has to simulate a small area and the kinoform is independent of where the intensity of the laser beam is.
Chapter 4  
Manufacturing

The fabrication process used to realize the kinoform structure on glass is the same used to produce micro electromechanical systems (MEMS). MEMS manufacturing refers to the fabrication of devices with at least some of its dimensions in the micrometer range [9]. At the beginning of its history MEMS was almost exclusively based on thin and thick film processes and materials borrowed from IC fabrication, such as UV-lithography, single-crystal Si wafers, techniques for removal (etching) and deposition of materials. Later new branches of processes and materials have been developed and established.

When working with MEMS processes it is important to keep the work-sample as clean as possible to avoid unwanted particles to be introduced to the structures, therefore almost all process-equipment is located in a so called clean-room. A clean-room has a low level of environmental pollutants, such as dust, airborne microbes, aerosol particles and chemical vapors. Engineers working in clean-rooms have to wear protective clothing so as not to contaminate the environment. This includes gloves, face masks and coveralls.

The idea was to realize the kinoform structure on a standard 4 inch pyrex glass wafer. The structures were etched into a thin layer of silicon dioxide, SiO$_2$, that was deposited on one side of the pyrex wafer. Silicon dioxide has a well defined and even refractive index, and is a good material to realize micro-structures in.

First the pyrex glass wafer was cleaned and a film of SiO$_2$ was depositioned on top of the wafer. Then a mask of Al was depositioned on top of the SiO$_2$ to act as a mask in the SiO$_2$-etching step. The Al-layer was then etched using plasma etch with S1813 positive photoresist used as mask. Then the structures was etched in the SiO$_2$ using dry plasma etch.

4.1 Detailed processes

The first step in the manufacturing process was to clean the pyrex wafer using a standard RCA process. This was done to make sure no organic materials or particles were left. The last RCA-bath was not used since the HF would dissolve the pyrex, rendering it useless. The first RCA cleanser is: five parts water, H$_2$O, one part 27% ammonium hydroxide, NH$_4$OH and one part 30% hydrogen peroxide, H$_2$O$_2$.

After cleaning the wafer, SiO$_2$ was deposited on the topside. This is done with a Von Ardenne CS 730S magnetron sputter. The program used 10 cycles with a sputtering step, see Table 4.1, for 10 min, then having a cooldown period between each step. The entire sputtering process time took about 5.5 h and rendered a stress free SiO$_2$ layer, which was about 1.2 $\mu$m thick.
The refractive index and the thickness of the SiO\textsubscript{2} layer was measured using an ellipsometer, Rudolph Research Auto Bi - II, and an interferometer, Leica Ergolux AMC MPV-SP.

On top of the SiO\textsubscript{2} layer a 120-150 nm thick film of Al was deposited using a \textit{Von Ardenne CS 730S} magnetron sputter, see Table 4.1.

In order to transfer the patterns from the CAD-file onto a wafer and realize them, photolithography with positive photoresist was used. The photoresist, S1813, was spun onto the Al-layer, 6000 rpm during 30 s, resulting in a 1.05 μm thick layer. Then the wafer were soft-baked on a hotplate at 115 °C for 60 s, in order to vaporise some of the solvent in the photoresist to make it solid. When exposed to UV-light the bonds in positive photoresist are dissociated. The structures were transferred onto the wafer using a lithography mask with UV-light projected through it using a Karl Süss Ma 6/Ba 6. The settings used was hard contact, 6 s exposure and 40μm alignment gap. The different layers and a schematic of the lithography step can be viewed in Figure 4.1.

The exposed wafer was then developed in one part Microposit developer and four parts H\textsubscript{2}O for 15 s, then rinsed in water for 1 min, spun dry, then hard-baked at 115 °C during 1 min. In order to remove any residue of photoresist a \textit{Tepla 300} plasma-strip was used, see Table 4.1. The result was a thin film of photoresist with the structures etched in, Figure 4.1.

The aluminum was etched using dry etch with an ICP PlasmaTherm SLR, Figure 4.1 and Table 4.1. In order to passivate the surface after the etch-process, the wafer was dipped in 40 °C H\textsubscript{2}O for 5 min.

The photoresist etch mask was then removed using the \textit{Tepla 300} plasma-strip, see Table 4.1.

Using \textit{Advanced Vacuum Vision 320} dry etching, the structures were etched into the SiO\textsubscript{2} layer with the Al as etch mask, Figure 4.1. The program used varied with each iteration of the process. That is, when etching the alignment marks an etch depth of about 1.1μm was desired, while the different structure depths depended on the refractive index of the SiO\textsubscript{2}-layer and the wavelength of the laser. This step, and which mask used in the lithography step, was the only variation of the process iterations. When the passmarks were etched, a working pressure of 20 mTorr and 400 W was used in 30 minutes. The DOE structures were etched using 70 mTorr and 550 W and different times were used in each step.

The Al mask was removed using wet etch, 29 parts H\textsubscript{3}PO\textsubscript{4}, 5 parts CH\textsubscript{3}COOH and 1 part HNO\textsubscript{3}.

These steps were iterated three times. First one for the init-mask, containing alignment marks, saw-marks and index numbering for the structures. Second and third with different structure-masks to realize the four level pattern, Figure 4.1.
CHAPTER 4. MANUFACTURING

4.1. DETAILED PROCESSES

Figure 4.1: The production steps illustrated step-by-step. a) Photo lithography, b) Developed photo resist, c) Al etched with photo resist as mask, d) SiO\(_2\) etched with Al as mask, e) Wafer after Al strip resulting in a 2 level lens, f) 2nd Photo-lithography, g) Developed photo-resist, h) Al etched with photo-resist as mask, i) SiO\(_2\) etched with Al as mask, e) Wafer after Al strip resulting in a 4 level lens.

<table>
<thead>
<tr>
<th>Process</th>
<th>Gas composition</th>
<th>Working pressure</th>
<th>Power</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiO(_2) deposition</td>
<td>1 parts Ar, 2 parts O(_2)</td>
<td>6 \cdot 10^{-3}\text{Torr}</td>
<td>300 W</td>
<td>10 min x 10</td>
</tr>
<tr>
<td>Al deposition</td>
<td>Ar</td>
<td>6 \cdot 10^{-4}\text{Torr}</td>
<td>600 W</td>
<td>60 s</td>
</tr>
<tr>
<td>Plasmastrip residue removal</td>
<td>O(_2) 50 ml/min, N(_2) 50 ml/min</td>
<td>248 mTorr</td>
<td>50 W</td>
<td>20 s</td>
</tr>
<tr>
<td>Al ICP dry etch</td>
<td>2 parts O(_2), 25 parts BCl(_3), 13 parts Cl(_2)</td>
<td>6 mTorr</td>
<td>RF1 10W, RF2 500W</td>
<td>1 min</td>
</tr>
<tr>
<td>Plasmastrip PR removal</td>
<td>O(_2) 50 ml/min</td>
<td>210 mTorr</td>
<td>1000 W</td>
<td>20 min</td>
</tr>
<tr>
<td>SiO(_2) dry etching</td>
<td>9 parts CHF(_3), 1 part O(_2)</td>
<td>20 / 70 mTorr</td>
<td>400 / 550 W</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 4.1: The specific parameters of the plasma-based processes.
Chapter 5

Design

The design of the miniaturized submarine, Figure 5.1 and 5.2, restricts the camera, lights and laser to be placed in the front end of the submarine body. The configuration used for this thesis is with the camera slightly off center and the laser on the opposite side, as opposed to in Figure 5.2, in order to get as much spacing as possible between these two. This configuration results in a center-to-center spacing of 2.4 cm. The idea of the laser pattern was to act as a visual aid to pictures taken using the camera and the dots should thus much as possible, for better space recognition.

5.1 Projected patterns

The patterns used in the first iteration of the manufacturing were created to fill up the entire visual areas. Patterns where each dot is easily related to others in a logical pattern such as grids and equidistant dots, Figure 5.3, to give as much information of the illuminated area as possible, were chosen.

5.2 Evaluation method

5.2.1 Setup

In order to test multiple lenses, a plastic rig was manufactured, using Fused Deposition Modelling (FDM) in a Dimension 3-D Printer, Figure 5.4. This configuration allowed for easy switching between DOE-lenses, different laser sources and a movable camera. The camera, Logitech Quickcam S5500, was separated from its casing and mounted 2.4 cm from the center of the laser-DOE configuration. The laser source, HL6714G, 670 nm, 10 mW Hitachi Laser Diode, a side-emitting laser diode, were mounted in an adjustable casing, Figure 5.4. An adjustable plastic focusing laser lens, f=3 mm, were mounted in front of the laser.

The rig was connected to a computer and a power source. In order to mimic conditions similar to that of the submarine submerged in water, without the need of waterproofing the rig, the rig was placed outside of a filled aquarium, Figure 5.5

5.2.2 LCM Software

To collect and analyze the pictures of the projected pattern, a dedicated software was developed, called *Laser-Camera Measurement* (LCM). The basic idea was to make a reference picture and detect how each dot moves in relation to this reference in order to evaluate the change of position for each dot. The source code can be viewed in Appendix B.
5.2. EVALUATION METHOD

CHAPTER 5. DESIGN

Figure 5.1: The illustration shows the different parts of the DADU submarine.

Figure 5.2: Close-up on the shell of the bow assembly. The camera is placed in the middle, lights and laser is placed in the slots around it.
Figure 5.3: The 12 different patterns that were chosen to be projected by the DOE-lenses.
Figure 5.4: Exploded view schematics of the plastic test rig.
The software was developed in MATLAB and uses the Image Acquisition Toolbox to communicate with the camera. The software can be used to analyze pictures captured in other programs as well. The interface workflow is linear and only designed to get the x, y and z coordinates of each dot in the picture.

In order to make any measurement with LCM the following is needed:

- One reference picture of the projected pattern at a known distance on a plane surface perpendicular to the camera view.
- The xyz coordinates of the camera focal point at the photodetector and the DOE.
- The cameras line of sight in angles, from the center to the top and from the center to the side edge.

These parameters are put into the settings part of the program and the physical setup is then modelled in the program, Figure 5.6. The view is a representation of what the LCM software knows of the current setup, the laser- and camera angles and positions. From this data the software can analyze a picture captured with the setup and return the xyz coordinates for each point.

The program utilizes a laser detection algorithm that searches for high intensity clots in the red channel of each picture. It also weights the center of the clot and uses the coordinate of the center to describe each laser points position. This increases the precision since the laser-point can get a center with higher resolution than the picture taken.

The basic parts of the program are:

- LCM_gui.m - The graphical user interface that links all the subroutines together into a whole. This function handles errors, plotting, loading and saving of data.
- FFA_ident.m and LCM_ident.m - Identifies the laser-points in a picture and returns their position in the picture. The only difference between these two is that LCM_ident.m asks for parameters such as maximum number of points and sensitivity until the user is satisfied, whereas FFA_ident.m gets these parameters from the start and handles the data in a more streamlined way. This is the choke-point of the live-view.
Figure 5.6: The laser fan-out and camera view shown in a 3-D view. The camera field of view is marked by blue and the laser beams are marked by red.
• LCM\_calib.m - Takes the position of the projected pattern, the laser disposition and the camera angles and returns the angles of the laser fan-out.

• LCM\_cam\_ang.m - Takes the positions of each dot in a picture and returns the angle at which the light from that dot enters the camera.

• LCM\_linematch.m - Finds the best intersection of the camera-beam and the laser-beam and returns the xyz-coordinate of the intersection.

• LCM\_measure.m - Handles calls to LCM\_linematch.m and LCM\_cam\_ang.m, filters the results and returns them.

LCM also has a live-view section in which a video stream is used to get live measurements. Since the live video stream requires quick computations, a table with the height corresponding to a laser-point position in a camera picture is first compiled before the streaming takes place. This method utilizes some other scripts:

• LCM\_live.m - Sets the camera feed and the data plots and connects the feed to the trigger-function.

• LCM\_frame\_trigger.m - Is called every time a new frame is sent from the video-stream. It calls FFA\_ident.m then finds the xyz-value out of a precompiled table for each position.

• FFA\_init.m - Generates a table with a predefined resolution and precision. This is a slow process and needs several minutes to complete, even for pictures with low resolution.

Some of the notable features of LCM is:

• Able to interpolate a surface from a set of xyz-positions.

• Handles different types of pictures and cameras, with different resolutions.

• Can save and load different setups.

• Can generate pictures from a given setup and a surface, eg. a sphere.

• Can compare the generated picture with the original surface.

• Easy interface for plotting different data-sets in a plot, such as the camera-laser setup, laser beams, the picture taken, the laser-dots and the interpolated surface.
Chapter 6

Results

6.1 Evaluation of structures

Through the course of manufacturing the first iteration of DOE-lenses, different evaluations were carried out, primarily in order to tune the process but also to evaluate the process itself.

The Lithography step was best evaluated after the Al-etch was done. Figure 6.1 shows the resolution on one of the first batches in a testing structure placed near the pass-marks. The testing structures works as follows; there’s a space Figure 6.1(a), or a beam, Figure 6.1(b), with decreasing width, from 0.4 \( \mu m \) to 3.2 \( \mu m \), with an increment of 0.2 \( \mu m \), in each step. This is meant to show how under/over exposed the photoresist mask is.

The corners of the structures were visually evaluated, Figure 6.2. One can clearly see that the structures were underexposed in Figure 6.2(a) while this error doesn’t affect the larger structures, (b).

The alignment error were measured on the finished structures using a microscope, Figure 6.3. The SiO\(_2\) structures was possible to see due to standing waves in the thin oxide-layer.

During the manufacturing process the topological features of the structures were measured using an atomic force microscope (AFM). In Figure 6.4 and 6.5 the four different heights of the DOE-surfaces can be observed, including the alignment error and the rounded edges. The ringe at the edges is probably an artifact from the AFM needle settings.
Figure 6.1: Pictures showing the resolution of the lithography. The lighter area is the aluminum mask and the darker is SiO$_2$. The beams have a width from 0.4 $\mu m$ to 3.2 $\mu m$ with an increment of 0.2 $\mu m$. (a) shows spaces and (b) beams of aluminum.

Figure 6.2: Picture of the edge-sharpness in different structure sizes, (a) 1.5 $\mu m$ and (b) 10 $\mu m$. 

Figure 6.3: Picture taken with microscope on finished structures. The different heights in the structure are visualized by different colors. In picture (a) and (b) the alignment error is apparent. Small structures, 1.5 μm, in (a). Medium structures, 5 μm, in (b). Large structures, 10 μm, in (c).
6.2 Fan-out and noise

The intensity and noise was measured using a Hamamatsu CMOS camera. The fan-out was merely confirmed to coincide with theoretical values using simple geometry. Pictures were taken with the pattern projected onto a white screen, which was photographed using the Hamamatsu camera. The pictures were then analyzed in MATLAB and compared to simulated intensity-maps of the pattern, Figure 6.6. The zeroth order is present in both (a) and (b) in Figure 6.6, but in the camera picture the high intensity of the spot causes the camera-chip to “bleed” into nearby pixels.

6.3 Camera + DOE evaluation

The testing module and LCM program was first tested in air to confirm its basic utility. The testing screen was put on a mobile base enabling measurements at different distances and angles. An example of the resulting LCM outputs can be seen in Figure 6.7, where a cylinder were put in front of the camera. The red stars in Figure 6.7 marks the spot where the laser beams hit the surface of the cylinder.

6.4 Module in water

The water testing was carried out using white plastic screens in a water-tank, Figure 6.8. The software was evaluated in regards to its capability to find the laser-spots when passing through water, Figure 6.9.

The pictures in Figure 6.9 shows the steps taken when measuring. First the calibration picture of a flat screen at a known distance, this result is then compared to the laserdots in
Figure 6.5: An AFM measurement of the surface of a 10 µm structure. The 4 levels can be seen in the line profile view.
Figure 6.6: A comparison of intensity of the simulated and the produced projected pattern.

Figure 6.7: The result of a measurement of a cylinder in air using the LCM software in combination with the testing module.
6.5 Etch rates of SiO$_2$

In order to predict the etch rate of the SiO$_2$ layer during manufacturing, measurements, Table 6.1, were noted during the process. The Al layer was measured using the knowledge of its initial and final thickness and extrapolated in between. The first two wafers, Si #1 and Si #2, were both test wafers to test the process.

Pyrex #1 were manufactured to be used with green laser ($\lambda = 532\,nm$), a SiO$_2$ layer with refractive index $n = 1.47$, and structures with steps of 286 nm.

Pyrex #2 were manufactured to be used with red laser ($\lambda = 670\,nm$), a SiO$_2$ layer with refractive index $n = 1.47$, and structures with steps of 359 nm.
Figure 6.9: The three steps of a measurement. First a) the calibration picture only has to be captured once per laser-camera setup. b) the picture of a object which is to be measured. c) the 3-D result can also be output as cartesian coordinates with respect to the camera.
Chapter 7

Discussion

7.1 Manufacturing process

The manufacturing steps that proved to be most important with respect to the quality of the diffraction pattern was found to be mask alignment and reactive ion etching of SiO$_2$ (depth accuracy).

As seen in Table 6.1 the etch rates of both the SiO$_2$ and the Al were different for each process cycle. This made it more cumbersome to stop and remove the Al mask to see the resulting etch depth. The measure depth in Al+SiO$_2$ layers were pretty easy to measure, using either the AFM or a profilometer in the clean-room. This issue could have been resolved if test-structures were put on the wafer and if more rigorous testing on what affected the etch rate in the SiO$_2$ etch were performed. Also, the structure where the measurements of the etch-depth were performed was one of the saw-marks on the wafer, which was a rather large structure and therefore had to assume uniform etch-rates independent of structure area and depth in order to be comparable to DOE structure depth.

Because of the size of the smaller structures, the alignment of the lithography mask is crucial. The error of Karl Süss mask aligner is about 0.2 µm, the lesser alignment steps the better. The first obvious improvement to the manufacturing process would be to put the initial alignment marks on the same mask as the first structure-mask. This way only one alignment is necessary and the total process time would be decreased by a third.

Another manufacturing step that required much more time than necessary was the lithography mask export. This was because of limitations in the software, the structures contained too much data and therefore made the program act up on occasions. The issues were finally resolved using a custom built version of the software that could handle more data, in combination with exporting only one structure at a time.

Because of the problems with alignment and etch-rates the easiest improvement of the process would be to only make a two level lens with a symmetrical pattern (asymmetry isn’t possible with two layers), and make those structures small enough. Another way of avoiding the alignment problem would be to use nanoinprinting technology with polymers. The master would be manufactured using a 128 layer gray-scale lithography process, thereby avoiding the alignment-issues, giving a higher possible resolution to the structures and many more layers to work with.

In spite of all these possible improvements of the process this first iteration of the manufacturing resulted in a fairly good result above expectations, Figure 6.6.
7.2 Design

The initial idea of the design was to give pictures captured with the submarine camera depth, but because of the narrow mounting of the laser and camera the projected pattern only deforms by rather deep structures. This problem is, however, an advantage when using a computer to find each spot location, this gives a rather deep range of possible measurable distances.

The precision of the LCM-measurement is dependent on the distance to the measured point. As a measuring tool it has a far-field where it cannot resolve the distance, which depends on the spacing between the camera and the DOE. In the tested case, 2.4 cm spacing, the far-field started at a distance of about 3 m.

When producing DOE-lenses for use with the LCM-tool, other parameters have to be taken into account account. Every dot should be easy to identify from the other dots and more intensity in each spot is preferred, which means a more sparse fan-out.

As mentioned before a revamp of the manufacturing process would enable less noise and more fan-out and therefore more distinct dots.

A design change that was discussed but never tested was to coat the SiO\textsubscript{2} structures with a layer of EpoClad polymer through spinning. EpoClad has a well defined refractive index when cured. It would act as a protective layer and resulting in a smooth surface. This however, would mean that the structures would have to be deeper to compensate for the lower difference in refractive index between the materials.

7.3 Software

The two software solutions developed for this project were both developed in MATLAB and are therefore dependent on some toolboxes, for example the Image Acquisition toolbox is required to use a web-cam in the LCM-interface. This is a trade-off, as some of the algorithms used are optimized and very well written and on the other hand restricts the platform mobility.

The LCM software added functionality to the design that by far exceeded the initial goal. With the function of live measurements, evaluations of objects sizes and shapes are improved significantly. The LCM software lacks one function that, in the future, would be preferable; the ability to take high-resolution pictures while using the live view to make a more precise measurement of an object.

The source code is separated into independent modules for further development and optimization.

The DOECAD software makes it possible to design new patterns with ease. The current algorithm is in need of an optimization revamp. A manually constructed Fourier-transform was written to handle the ORA algorithm, it should however be possible to utilize the built in FFT in MATLAB, which would significantly reduce the simulation time.

7.4 Future improvements

The easiest thing to improve in the current setup would be to switch the laser source and its lens to more suited ones. The wavelength should be shorter, water absorbs longer wave-lengths. The power of the laser could be higher, at about 30 mW, this shouldn’t pose a problem since the laser will only be in use in short bursts when measuring topology. The laser optics that collimates the beam should be switched for one that gives a very small dot, a waist of about 0.3 mm would suffice to cover at least one basic cell of the kinoform. These improvements would lead to more precise measurements and easier-to-find dots.
The LCM software has a lot of potential improvements. When a setup is calibrated a mask could be generated that would remove all colour data outside of the tracks where the dots will move. Another part that has a lot of potential is the dot-identifier algorithm. An example would be identifying a cluster of small dots as one and identify its center, or adding a noise filter that would smooth the dots and the noise, making it easier to identify.

The most important thing to improve on is the pattern chosen for projection. The patterns should be sparse, with a large enough fan-out angle to cover the whole picture, and should be designed so that the dots will not cross each others paths when moving in the picture. An example of this can be observed in Figure 7.1.
Chapter 8

Conclusions

- A first iteration of the DOE-lenses were successfully manufactured and tested with great success.
- A testing rig was assembled and confirmed to work in the desired environment.
- Most of the designed patterns were unfit to use with the LCM software. Sparse patterns would have been preferred.
- A tool, DOECAD, was developed for further development of new DOE-lenses.
- A tool, LCM, was developed in order to post process pictures and to measure live feeds from the testing rig. LCM was also prepared for further development in order to add additional features and improve on the existing ones.
Chapter 9

Acknowledgements

First of all, I would like to thank my supervisor, Jonas Jonsson, for extensive help with design and manufacturing, valuable input and proofreading.

I would also like to thank Henrik Kratz for great input throughout my work with this thesis and for proofreading. Mikael Karlsson and Jörgen Bengtsson helped me getting started with the simulation, thank you.

I could not have finally succeeded with the mask export without the help and perseverance of Angelo Pallin.

A big thanks goes to Hugo Nguyen, Ville Lekholm, Sara Lotfi, Kristoffer Palmer, Anders Persson, Johan Sundqvist, Greger Thornell for all the assistance and to the entire material science group for making my time here so pleasant.

Karin Markgren, thank you, for all the love and support you provide, and for proofreading the theses.
References


Appendix A

DOECAD - Source

```matlab
function varargout = DOECAD(varargin)

% Begin initialization code

% Begin initialization code

% --- Executes just before DOECAD is made visible.
function DOECAD_OpeningFcn(hObject, eventdata, handles, varargin)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Outputs from this function are returned to the command line.
function varargout = DOECAD_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

function radio_ora_Callback(hObject, eventdata, handles)

% --- Executes on button press in radio_ifta.
function radio_ifta_Callback(hObject, eventdata, handles)

% --- Executes on button press in transform_pic_to_phase.
function transform_pic_to_phase_Callback(hObject, eventdata, handles)

method = get(handles.transform_method, 'Value');
switch get(handles.pop_pic_type, 'Value')
    case 1
        picture = handles.pic_org;
    case 2
        picture = handles.pic;
end
switch method
    case 1 % Direct
        if (size(picture,1) > 0)
            handles.phase = pictophase(picture);
            guidata(hObject, handles);
            redraw(handles);
        end
    case 2 % IFTA
        if (size(picture,1) > 0)
            iteration = str2num(get(handles.iterations, 'String'));
            handles.phase = ifta(picture, iteration, handles);
            guidata(hObject, handles);
            redraw(handles);
        end
    case 3 % ORA
        if (size(picture,1) > 0)
            if get(handles.pop_pic_type, 'Value') == 2
                warndlg('You are transforming a result with ORA! This will take some time... Consider this a warning.', 'Transforming result with ORA!');
            end
            iteration = str2num(get(handles.iterations, 'String'));
            quality = str2num(get(handles.quality, 'String'));
            intensity = str2num(get(handles.intensity, 'String'));
            levels = str2num(get(handles.levels, 'String'));
            [filename, pathname] = uiputfile(['*.bmp', 'Save Snapshots As']);
            if isequal(filename, 0) | isequal(pathname, 0)
                disp('User selected Cancel')
            else
                handles.phase = ora(picture, levels, intensity, fullfile(pathname, filename), quality, iteration, handles);
                guidata(hObject, handles);
                redraw(handles);
            end
        end
end

function ifta_iterations_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ifta_iterations_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in export_phase.
function export_phase_Callback(hObject, eventdata, handles)
value = get(handles.export_method, 'Value')
switch value
    case 1 % Bitmap
        [filename, pathname] = uiputfile('*.*','Save Phase As');
        if isequal(filename,0) | isequal(pathname,0)
            disp('User selected Cancel')
        else
            imwrite(handles.phase/2/pi, fullfile(pathname, filename));
        end
    case 2 % DXF (Layers)
        [filename, pathname] = uiputfile('*.*','Save Phase As');
        if isequal(filename,0) | isequal(pathname,0)
            disp('User selected Cancel')
        else
            export_lens_to_dxf(handles.phase, fullfile(pathname, filename));
        end
    case 3 % DXF (3D)
        [filename, pathname] = uiputfile('*.*','Save Phase As');
        if isequal(filename,0) | isequal(pathname,0)
            disp('User selected Cancel')
        else
            edxf(handles.phase, fullfile(pathname, filename));
        end
    case 4 % MAT
        [filename, pathname] = uiputfile('*.*','Save Phase As');
        if isequal(filename,0) | isequal(pathname,0)
            disp('User selected Cancel')
        else
            save(fullfile(pathname, filename), 'phase');
        end
end

% --- Executes on button press in import_pic.
function import_pic_Callback(hObject, eventdata, handles)
[filename, pathname] = uigetfile({'*.bmp;*.png;*.mat','Bitmaps & MAT-file (*.mat);*.*', 'All Files (*.*)'}, 'Pick a file');
if isequal(filename,0)
    disp('User selected Cancel')
else
    switch filename(end-2:end)
        case 'mat'
            picture=[];
            load(fullfile(pathname, filename), 'picture');
            if size(picture,1)==0
                disp(['User selected ', fullfile(pathname, filename)]);
            else
                disp(get(hObject,'BackgroundColor'))
            end
        case 'png'
            disp(['User selected ', fullfile(pathname, filename)]);
        end
    end
end
warndlg ( 'The selected Mat-file does not contain the variable picture', 'Warning' );

end

pic = picture;

pic = pic / max ( max ( pic ));

otherwise

[ pic, map ] = imread ( fullfile ( pathname, filename ));

if size ( map, 1 ) > 0

pic = double ( pic ) / ( size ( map, 1 ) - 1 ) ;

end

else

pic = sum ( pic, 3 );

pic = double ( pic / max ( max ( pic ))) ;

end

end

% Plottar bilden

handles.pic_org = pic ;

set ( handles.pop_pic_type, 'Value', 1 );

guida ( hObject, handles );

redraw ( handles );

end

% --- Executes on button press in export_pic.

function export_pic_Callback ( hObject, eventdata, handles )

[ filename, pathname ] = uiputfile ( {'*.bmp', 'Bitamp (*.bmp)'; '*.mat', 'MAT-file (*.mat)'),'Save As' );

if isequal ( filename, 0 ) | isequal ( pathname, 0 )

disp ( ' User selected Cancel ' );

else

switch get ( handles.pop_pic_type, 'Value' );

case 1

picture = handles.pic_org;

case 2

picture = handles.pic;

end

if size ( picture, 1 ) > 0

disp ( [ 'User selected ', fullfile ( pathname, filename ) ] )

switch filename ( end = 2 : end )

case 'mat'

save ( fullfile ( pathname, filename ), 'picture' );

case 'bmp'

imwrite ( picture / max ( max ( ( picture ))) , fullfile ( pathname, filename ) );

end

disp ( ' Empty picture ' )

end

end

% --- Executes on button press in import_phase.

function import_phase_Callback ( hObject, eventdata, handles )

[ filename, pathname ] = uigetfile ( {'*.bmp;*.png;*.mat' ; '*.mat'; 'MAT-file (*.mat)'; 'All Files (*.*)'}, 'Pick a file' );

if isequal ( filename, 0 )

disp ( 'User selected Cancel' )

else

disp ( [ 'User selected ', fullfile ( pathname, filename ) ] );

switch filename ( end = 2 : end )

case 'mat'

phase = [];

load ( fullfile ( pathname, filename ), 'phase' );

if size ( phase, 1 ) == 0
warndlg('The selected Mat-file does not contain the variable phase', 'Warning');

end

case 'bmp'

[phase, map] = imread(fullfile(pathname, filename));
if size(map, 1) > 0
    phase = 2 * pi * double(phase) ./ (size(map, 1) - 1);  % renormaliserar bilden till skala 0-1
else
    phase = sum(phase, 3);
    phase = double(phase ./ max(max(phase)));  
end

end

% Plottar bilden
handles.phase = phase;
guider(hObject, handles)
redraw(handles);

end

% --- Executes on button press in transform_phase_to_pic.
function transform_phase_to_pic_Callback(hObject, eventdata, handles)

if (size(handles.phase, 1) > 0)
    handles.pic = abs(fftshift(fft2(exp(i * handles.phase))));
    set(handles.pop_pic_type, 'Value', 2);
guider(hObject, handles);
    redraw(handles);
end

% --- Executes on button press in transform_phase_to_pic.
function transform_phase_to_pic_Callback(hObject, eventdata, handles)

if (size(handles.phase, 1) > 0)
    handles.pic = abs(fftshift(fft2(exp(i * handles.phase))));
    set(handles.pop_pic_type, 'Value', 2);
guider(hObject, handles);
    redraw(handles);
end

% --- Executes on selection change in ora_quality.
function ora_quality_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ora_quality_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% --- Executes on selection change in ifta_iter.
function ifta_iter_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function ifta_iter_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end
function []=eedxf ( handles , fname)
    fas=handles.phase;
    depth=2*str2num(get(handles.man_depth,'string'))./1000;
    if depth==0
        depth=1;
    end
    fas = fas./2./pi.*depth;
    I=1:antal;
    X_=(I-floor(antal/2));
    Y_=(ceil(antal/2)-I);
    [X_,Y_]=meshgrid(X_,Y_);
    for a=1:antal
        for b=1:antal
            Z((2*a-1):2*a,(2*b-1):2*b)=[fas(a,b);fas(a,b);fas(a,b);fas(a,b)];
            X((2*a-1):2*a,(2*b-1):2*b)=[X_((a,b)-0.5,X_((a,b)+0.5,X_((a,b)-0.5,X_((a,b)+0.5).*pixel;
            Y((2*a-1):2*a,(2*b-1):2*b)=[Y_((a,b)+0.5,Y_((a,b)+0.5,Y_((a,b)-0.5,Y_((a,b)+0.5).*pixel;
        end
    end
    writedxf(fname,X,Y,Z);
end

function []=writedxf ( fname ,X,Y,Z)
    fid=fopen(fname,'w');
    fprintf(fid,'0
SECTION
2
HEADER
0
ENDSEC
0
SECTION
2
ENTITIES
0
');
    for a=1:size(X,1)-1
        for b=1:size(X,2)-1
            fprintf(fid,'3DFACE
8
0
0
Y %.4f
20
%.4f
30
%.4f
',X(a,b),Y(a,b),Z(a,b));
            fprintf(fid,'10
%.4f
20
%.4f
30
%.4f
',X(a+1,b),Y(a+1,b),Z(a+1,b));
            fprintf(fid,'11
%.4f
21
%.4f
31
%.4f
',Y(a+1,b+1),Y(a+1,b+1),Z(a+1,b+1));
            fprintf(fid,'12
%.4f
22
%.4f
32
%.4f
',X(a+1,b+1),Y(a+1,b+1),Z(a+1,b+1));
            fprintf(fid,'13
%.4f
23
%.4f
33
%.4f
',X(a,b+1),Y(a,b+1),Z(a,b+1));
        end
    end
    fprintf(fid,'0
');
    fclose(fid);
end

% --- Executes on selection change in export_method.
function export_method_Callback(hObject , eventdata, handles)
% --- Executes during object creation, after setting all properties.

function export_method_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
end

function []=export_lens_to_dxf(fas, fname, pixel_size)
antal=size(fas,1);
I=1:antal;
X_=(I-ceil(antal/2)).*pixel_size;
Y_=(ceil(antal/2)-I).*pixel_size;
[X_,Y_]=meshgrid(X_,Y_);
step=pixel_size/2;
fas=fas/4;
fid=fopen(fname,'w');
fprintf(fid,'999
created by Matlab\nSECTION\n2\nENTITIES\n0
');
for a=1:antal
    for b=1:antal
        fprintf(fid,'POLYLINE
8
Phase %u
66
1
10
0.0
20
0.0
30
0
70
1
0
',fas(a,b));
        fprintf(fid,'VERTEX
8
Phase %u
',fas(a,b));
        fprintf(fid,'10
%.6f
20
%.6f
30
0.0
',X_(a,b)-step,Y_(a,b)+step);
        fprintf(fid,'0
');
        fprintf(fid,'VERTEX
8
Phase %u
',fas(a,b));
        fprintf(fid,'10
%.6f
20
%.6f
30
0.0
',X_(a,b)+step,Y_(a,b)+step);
        fprintf(fid,'0
');
        fprintf(fid,'VERTEX
8
Phase %u
',fas(a,b));
        fprintf(fid,'10
%.6f
20
%.6f
30
0.0
',X_(a,b)+step,Y_(a,b)-step);
        fprintf(fid,'0
');
        fprintf(fid,'VERTEX
8
Phase %u
',fas(a,b));
        fprintf(fid,'10
%.6f
20
%.6f
30
0.0
',X_(a,b)-step,Y_(a,b)-step);
        fprintf(fid,'0
');
    end
fprintf(fid,'SEQEND
8
Phase %u
0
');
end
APPENDIX A. DOECAD - SOURCE

401
402
403 fprintf(fid,'
ENDSEC
0
nEOF
');
404 fclose(fid);
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
function iterations_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function iterations_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function quality_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function quality_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function fas = pictophase(original_bild)

fas = (fft2(abs(fftshift(fft2(exp(i*angle(fft2(original_bild))))))));
fas = atan2(imag(fas),real(fas))+pi;

function fas = ifta(original_picture,loops,handles)

set(handles.status,'Visible','On');
set(handles.busy,'Visible','On');
fas = (fft2(abs(fftshift(fft2(exp(i*angle(fft2(original_picture))))))));
fas = atan2(imag(fas),real(fas));
for foo = 1:loops
    set(handles.status,'String',['Iteration: ',num2str(asdasddd),'/',num2str(loops)]);
    bild = fftshift(fft2(exp(i*fas)));
    bild = original_picture.*exp(i*angle(bild));
    fas = (fft2(abs(fftshift(fft2(exp(i*angle(fft2(bild))))))));
    fas = atan2(imag(fas),real(fas));
    pause(eps);
end

set(handles.status,'Visible','Off');
set(handles.busy,'Visible','Off');
fas = fas*pi;

function kino_fas = ora(original_picture,levels,intensity_factor,fname,quality,iterations,handles)

%initializing variables
avail = linspace(0,(2*pi-2*pi/levels),levels);
number_of = size(original_picture,1);
I=1:number_of;
X=ceil(number_of/2);
Y=ceil(number_of/2)-I;
[X,Y]=meshgrid(X,Y);
n_tot=0;
z_pos=0;
set(handles.status,'Visible','On');
set(handles.busy,'Visible','On');
set(handles.status,'String',['Preparing data...']);
pause(eps);
kino_fas=avail(ceil(size(avail,2)*rand(size(original_picture))));
for c = 1: number_of
    for v = 1: number_of
        if (original_picture(c, v) > 0.9 | (c == ceil(number_of/2) & v == ceil(number_of/2)))
            n_tot = n_tot + 1;
            n_x(n_tot) = X(c, v);
            n_y(n_tot) = Y(c, v);
            n_I_org(n_tot) = original_picture(c, v);
            bild = sum(sum(exp(i*kino_fas) .* exp(i*(X*n_x(n_tot)+Y*n_y(n_tot)) + 2*pi/number_of)));
            n_fas(n_tot) = angle(bild);
            n_I(n_tot) = abs(bild);
        end
    end
end
pause(eps);
end
n_I_org = intensity_factor * n_I_org * (number_of^2) / sqrt(n_tot - 1);
r = 1;
Q(r) = sum((n_I - n_I_org).^2);
set(handles.status, 'String', ['Iteration: ', num2str(r), ' Quality: ', num2str(Q(r)),'%']);
pause(eps);
done = false;
while (done == false)
    for i_x = 1: number_of
        for i_y = 1: number_of
            alpha = (n_x(i_x, i_y)+n_y(i_x, i_y))*2*pi/(number_of);
        end
    end
end
suggestion = 1;
for u = 1: size(avail, 2)
    suggestion_bild(:, u) = (n_I .* exp(i*kino_fas) - exp(i*(kino_fas(i_x, i_y)+alpha)) + exp(i*(avail(u)+alpha)));
    suggestion_I(:, u) = abs(suggestion_bild(:, u));
    suggestion_q(u) = sum((suggestion_I(:, u)^2 - n_I_org).^2);
    if suggestion_q(u) < suggestion_q(suggestion)
        suggestion = u;
    end
end
kino_phase(i_x, i_y) = avail(suggestion);
% Calculates new phase and intensity
% in the picture-frame
n_phase=(angle(suggest_bild(:,suggest)));

n_I=suggest_I(:,suggest);

end

pause(eps);

% Calculates the quality in the whole picture
r=r+1;
Q(r)=sum((n_I-n_I_org).^2);

set(handles.status,'String',['Iteration : ',num2str(r),'

Quality : ',num2str(100*(Q(r)/Q(r-1))),'='%]);

pause(eps);

if(Q(r)/Q(r-1)>quality/100 | r>iterations)
done=true;
end

disp '-------------

if(size(fname)>0)
imwrite(rot90(kino_phase,2)/2/pi,fname)

end

set(handles.status,'Visible','Off');
set(handles.busy,'Visible','Off');
kino_phase=rot90(kino_phase,2);

function intensity_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function intensity_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0, ...)
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');

end

function levels_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function levels_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'), get(0, ...)
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');

end

function redraw(handles)

    n_1= str2num(get(handles.man_n_1,'String'));
    n_2= str2num(get(handles.man_n_2,'String'));
    lambda= str2num(get(handles.man_lambda,'String'));
    angle= pi*str2num(get(handles.man_angle,'String'))/360;
    ant=size(handles.phase,1);
    if (ant>0)
       axes(handles.plot_phase);
       value=get(handles.pop_phase,'Value');
pix = lambda/\sin(\tan(2\cdot \tan(\text{angle})/\text{ant}));
depth = lambda/(n_1-n_2)/2;
switch value
  case 1
    imagesc(handles.phase);
  case 2
    mesh(handles.phase);
end
set(handles.man_pixel,'String',num2str(pix));
set(handles.man_depth,'String',num2str(depth));
end

switch get(handles.pop_pic_type,'Value');
  case 1
    axes(handles.plot_pic);
    imagesc(0);
    if (size(handles.pic_org,1)>0)
      value = get(handles.pop_pic,'Value');
      switch value
        case 1
          imagesc(handles.pic_org);
        case 2
          mesh(handles.pic_org);
      end
    end
  case 2
    axes(handles.plot_pic);
    imagesc(0);
    if (size(handles.pic,1)>0)
      value = get(handles.pop_pic,'Value');
      switch value
        case 1
          imagesc(handles.pic);
        case 2
          mesh(handles.pic);
      end
    end
end
colormap(gray);

% --- Executes on selection change in pop_pic.
function pop_pic_Callback(hObject, eventdata, handles)
  redraw(handles);
end

% --- Executes during object creation, after setting all properties.
function pop_pic_CreateFcn(hObject, eventdata, handles)
  if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
  end
end

% --- Executes on selection change in popupmenu6.
function popupmenu6_Callback(hObject, eventdata, handles)
end
% --- Executes during object creation, after setting all properties.
function popupmenu6_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function pop_phase_Callback(hObject, eventdata, handles)
    redraw(handles);

function pop_phase_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function man_lambda_Callback(hObject, eventdata, handles)
    redraw(handles);

function man_lambda_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function man_angle_Callback(hObject, eventdata, handles)
    redraw(handles);

function man_angle_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function man_pixel_Callback(hObject, eventdata, handles)
    redraw(handles);

function man_pixel_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end

function man_depth_Callback(hObject, eventdata, handles)
    redraw(handles);

function man_depth_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject,'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
function man_n_1_Callback(hObject, eventdata, handles)
    redraw(handles);
end

function man_n_1_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject, 'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject, 'BackgroundColor', 'white');
    end
end

function man_n_2_Callback(hObject, eventdata, handles)
    redraw(handles);
end

function man_n_2_CreateFcn(hObject, eventdata, handles)
    if ispc && isequal(get(hObject, 'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject, 'BackgroundColor', 'white');
    end
end

% --- Executes on button press in pushbutton20.
function pushbutton20_Callback(hObject, eventdata, handles)
    helpdlg('Transform into the Phase-plane with one of the 3 methods.
    - FFT (Instant) Transforms the picture to its phase
    - IFTA (fast) Iterative Fourier Transform Algorithm
    * Iterations: Number of iterations back and forth
    * ORA (slow): The Optimal Rotation Angle Method developed by J"orgen Bengtson
    + + Iterations: Maximum number of iterations
    + + Quality [%]
    + + Intensity factor: Increasing this reduces noise but decreases uniformity
    + + Levels: The number of different phaseshifts between 0 and 2pi i. e. 4 gives [0 pi/2 pi 3pi/2]', 'Transform');
end

% --- Executes on button press in pushbutton21.
function pushbutton21_Callback(hObject, eventdata, handles)
    helpdlg('Export the phases in one of the three formats
    - Bitmap: A bitmap with the phase 0 at black and 360 at white
    - DXF (Layers): each phase in a layer (up to a maximum of 64 layers)
    - DXF (3D): A 3D model of the lens with attributes specified in files below', 'Export phase');
end

% --- Executes on button press in pushbutton22.
function pushbutton22_Callback(hObject, eventdata, handles)
    helpdlg('Import a bitmap containing the desired pattern to be projected White=full Black=no intensity', 'Import Picture Help');
end

% --- Executes on button press in pushbutton23.
function pushbutton23_Callback(hObject, eventdata, handles)
    helpdlg('Transforms the phase to the picture-plane.', 'Transform');
end

% --- Executes on button press in pushbutton24.
function pushbutton24_Callback(hObject, eventdata, handles)
helpdlg('1: Draw a picture of desired pattern white is bright.

2. Import the picture in the
leftmost area
3. Transform into the Phase-plane with one of the 3 methods.
   -- FFT (Instant) Transforms the picture to its phase
   -- IFTA (fast) Iterative Fourier Transform
   -- ORA (slow): The Optimal Rotation Angle Method developed by Jörgen Bengtson

(4). Transform back to see the resulting pattern from this phase
5. Export the phases in one of the three formats
   -- Bitmap: A bitmap with the phase 0 at black and 360 at white
   -- DXF (Layers): each phase in a layer
   -- DXF (3D): A 3D model of the lens with attributes specified in fields below.

% --- Executes on selection change in pop_pic_type.
function pop_pic_type_Callback(hObject, eventdata, handles)
redraw(handles);

% --- Executes during object creation, after setting all properties.
function pop_pic_type_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject, 'BackgroundColor'), get(0, 'defaultUicontrolBackgroundColor'))
    set(hObject, 'BackgroundColor', 'white');
end

function Untitled_1_Callback(hObject, eventdata, handles)
function Untitled_2_Callback(hObject, eventdata, handles)

function h1 = DOECAD_LayoutFcn(policy)
persistent hsingleton;
if strcmpi(policy, 'reuse') && ishandle(hsingleton)
    h1 = hsingleton;
end

appdata = []; appdata.GUIDEOptions = struct(...
    'active_h', [], ...
    'taginfo', struct(...
    'figure', 2, ...
    'pushbutton', 25, ...
    'uipanel', 14, ...
    'axes', 4, ...
    'radiobutton', 10, ...
    'text', 28, ...
    'edit', 15, ...
    'popupmenu', 9, ...
    'slider', 2, ...
    'override', 1, ...
    'release', 13, ...
    'resize', 'simple', ...
    'accessibility', 'callback', ...
    'mfile', 1, ...
    'callbacks', 1, ...)
singleton, 1, ...
syscolorfig, 1, ...
blocking, 0, ...
lastSavedFile, 'C:\Documents and Settings\Martin\Desktop\DOECAD.m');
appdata.lastValidTag = 'figure1';
appdata.GUIDeLayoutEditor = [];
'CreateFcn', @local_CreateFcn, appdata);}
appdata = [];
appdata.lastValidTag = 'transform_pic_to_phase';
h4 = uicontrol(...
'Parent', h3,...
'Units', 'normalized', ...
'Callback', 'DOECAD("transform_pic_to_phase_Callback", gcbo,[],guidata(gcbo))', ...
'Position', [0.492211838006231 0.683673469387755 0.214953271028037 0.2346938775102], ...
'String', 'Start', ...
'Tag', 'transform_pic_to_phase', ...
'Behavior', get(0,'defaultuicontrolBehavior'), ...
'CreateFcn', @local_CreateFcn, appdata);}
appdata = [];
appdata.lastValidTag = 'transform_method';
h5 = uicontrol(...
'Parent', h3,...
'Units', 'normalized' ,...
'BackgroundColor', [1 1 1],....
'Callback', 'DOECAD("transform_method_Callback", gcbo,[],guidata(gcbo))', ....
'Position', [0.121495327102804 0.663265306122449 0.35202492211838 0.224489795918367], ....
'String', {'FFT (Instant)'; 'IFTA (Fast)'; 'ORA (Slow)' },....
'Style', 'popupmenu',  ....
'Value', 1,...
'CreateFcn', @local_CreateFcn, 'DOECAD("transform_method_CreateFcn", gcbo,[],guidata(gcbo))', appdata); ....
'Tag', 'transform_method', ...
'Behavior', get(0,'defaultuicontrolBehavior'));
appdata = [];
appdata.lastValidTag = 'iterations';
h6 = uicontrol(...
'Parent', h3,...
'Units', 'normalized' ,...
'BackgroundColor', [0.752941176470588 0.752941176470588 0.752941176470588], ...
'Callback', 'DOECAD("iterations_Callback", gcbo,[],guidata(gcbo))', ...
'Enable', 'inactive', ...
'ForegroundColor', [0.501960784313725 0.501960784313725 0.501960784313725], ...
'HorizontalAlignment', 'right', ...
'Position', [0.121495327102804 0.163265306122449 0.1526479750778 0.214285714285714], ...
'String', '100', ...
'Style', 'edit', ...
'CreateFcn', @local_CreateFcn, 'DOECAD("iterations_CreateFcn", gcbo,[],guidata(gcbo))', appdata); ....
'Tag', 'iterations', ...
'Behavior', get(0,'defaultuicontrolBehavior'));
appdata = [];
appdata.lastValidTag = 'text4';
h7 = uicontrol(...
'Parent', h3,...
'Units', 'normalized', ...
'}
961 'Position', [0.121495327102804 0.459183673469387 0.1651090342679 13 0.153061224489796], ...
962 'String', 'Iterations:', ...
963 'Style', 'text', ...
964 'Tag', 'text4', ...
965 'Behavior', get(0, 'defaultuicontrolBehavior'), ...
966 'CreateFcn', (@local_CreateFcn, '', appdata));

appdata = []; 
appdata.lastValidTag = 'quality'; 

h8 = uicontrol(... 
'Parent', h3, ... 
'Units', 'normalized', ... 
'BackgroundColor', [0.752941176470588 0.752941176470588 0.752941176470588], ... 
'Callback', 'DOECAD(''quality_Callback'', gcbo, [], guidata(gcbo))', ... 
'Enable', 'inactive', ... 
'ForegroundColor', [0.501960784313725 0.501960784313725 0.501960784313725] ,...
978 'HorizontalAlignment', 'right', ... 
979 'Position', [0.314641744548287 0.163265306122448 0.1526479750778 82 0.214285714285714] ,...
980 'String', '99.99', ...
981 'Style', 'edit', ...
982 'CreateFcn', (@local_CreateFcn, 'DOECAD(''quality_CreateFcn'', gcbo, [], guidata(gcbo)), appdata), ...
983 'Tag', 'quality', ...
984 'Behavior', get(0, 'defaultuicontrolBehavior'));

appdata = []; 
appdata.lastValidTag = 'text5'; 

h9 = uicontrol(... 
'Parent', h3, ... 
'Units', 'normalized', ... 
'BackgroundColor', [0.752941176470588 0.752941176470588 0.752941176470588], ... 
'Callback', 'DOECAD(''intensity_Callback'', gcbo, [], guidata(gcbo))', ... 
'Enable', 'inactive', ... 
'ForegroundColor', [0.501960784313725 0.501960784313725 0.501960784313725] ,...
998 'HorizontalAlignment', 'right', ... 
1000 'Position', [0.305295950155763 0.459183673469387 0.1775700934579 44 0.153061224489796] ,...
1001 'String', 'Quality: [%]', ...
1002 'Style', 'text', ...
1003 'Tag', 'text5', ...
1004 'Behavior', get(0, 'defaultuicontrolBehavior'));

appdata = []; 
appdata.lastValidTag = 'intensity'; 

h10 = uicontrol(... 
'Parent', h3, ... 
'Units', 'normalized', ... 
'BackgroundColor', [0.752941176470588 0.752941176470588 0.752941176470588], ... 
'Callback', 'DOECAD(''intensity_Callback'', gcbo, [], guidata(gcbo))', ... 
'Enable', 'inactive', ... 
'ForegroundColor', [0.501960784313725 0.501960784313725 0.501960784313725] ,...
1014 'Tag', 'intensity', ...
appdata = [];
appdata.lastValidTag = 'text6';
h11 = uicontrol(...
    'Parent',h3,...
    'Units','normalized'....
    'HorizontalAlignment','left'....
    'Position',[0.501557632398754 0.438775510204081 0.274143302180685
               0.17349387755102]....
    'String','Intensity factor:....
    'Style','text'....
    'Tag','text6'....
    'Behavior',get(0,'defaultuicontrolBehavior')....
    'CreateFcn',{@local_CreateFcn ,'' ,appdata});

appdata = [];
appdata.lastValidTag = 'levels';
h12 = uicontrol(...
    'Parent',h3,...
    'Units','normalized'....
    'BackgroundColor',[0.752941176470588 0.752941176470588
                     0.752941176470588]....
    'Callback','DOECAD(''levels_Callback'',gcbo,[],guidata(gcbo))'....
    'Enable','inactive'....
    'ForegroundColor',[0.501960784313725 0.501960784313725
                     0.501960784313725]....
    'HorizontalAlignment','right'....
    'Position',[0.753894080996885 0.163265306122448 0.15264797507782
               0.214285714285714]....
    'String','4'....
    'Style','edit'....
    'CreateFcn',{@local_CreateFcn ,'' ,guidata(gcbo))'....
    'Tag','levels'....
    'Behavior',get(0,'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'text7';
h13 = uicontrol(...
    'Parent',h3,...
    'Units','normalized'....
    'HorizontalAlignment','left'....
    'Position',[0.744548286604362 0.448979591836735 0.24299065420567
               0.163265306122449]....
    'String','Levels:'....
    'Style','text'....
    'Tag','text7'....
    'Behavior',get(0,'defaultuicontrolBehavior')....
    'CreateFcn',{@local_CreateFcn ,'' ,appdata});

appdata = [];
appdata.lastValidTag = 'pushbutton20';
h14 = uicontrol(...
    'Parent',h3,...
    'Units','normalized'....
    'Callback','DOECAD(''pushbutton20_Callback'',gcbo,[],guidata(gcbo))'....
    'Position',[0.707165109034268 0.683673469387755 0.0498442367601246
               0.2346937755102]....
    'String','?';
'Tag', 'pushbutton20', ...
'Behavior', get(0, 'defaultuicontrolBehavior'), ...
'CreateFcn', (@local_CreateFcn, '', appdata));

appdata = [];
appdata.lastValidTag = 'import_pic';

h15 = uicontrol(...
'Parent', h2,...
'Units', 'normalized', ...
'Callback', 'DOECAD(''import_pic_Callback'', gcbo, [], guidata(gcbo))', ...
'Position', [0.703170028818444 0.267782426778243 0.1988472622478 39 0.0481171548117155], ...

'appdata = [];
appdata.lastValidTag = 'export_pic';

h16 = uicontrol(...
'Parent', h2,...
'Units', 'normalized', ...
'Callback', 'DOECAD(''export_pic_Callback'', gcbo, [], guidata(gcbo))', ...
'Position', [0.0345821325648415 0.265690376569038 0.1988472622478 39 0.0481171548117155], ...

'appdata = [];
appdata.lastValidTag = 'plot_pic';

h17 = axes(...
'Parent', h2,...
'Position', [0.069164265129683 0.355648535564854 0.86743515850144 0.629707112970711], ...

'CameraPosition', [0.5 0.5 9.16025403784439], ...
'CameraPositionMode', get(0, 'defaultaxesCameraPositionMode'), ...
'Color', get(0, 'defaultaxesColor'), ...
'ColorOrder', get(0, 'defaultaxesColorOrder'), ...
'LooseInset', [0.144912718204489 0.157334963325183 0.105897755610973 0.107273838630807], ...

'XColor', get(0, 'defaultaxesXColor'), ...
'YColor', get(0, 'defaultaxesYColor'), ...
'ZColor', get(0, 'defaultaxesZColor'), ...
'Tag', 'plot_pic', ....

'h18 = get(h17,'title');

set(h18, ...
'Parent', h17,...
'Units', 'data', ...
'FontUnits', 'points', ...
'BackgroundColor', 'none', ...
'Color', [0 0 0], ...
'EdgeColor', 'none', ...
'EraseMode', 'normal', ...
'DVIMode', 'auto', ...
'FontAngle', 'normal', ...
1132 'FontName', 'Helvetica', ...
1133 'FontSize', 10, ...
1134 'FontWeight', 'normal', ...
1135 'HorizontalAlignment', 'center', ...
1136 'LineStyle', '-', ...
1137 'LineWidth', 0.5, ...
1138 'Margin', 2, ...
1139 'Position', [0.498355263157895 1.02181208053691 1.00005459937205], ...
1140 'Rotation', 0, ...
1141 'String', '', ...
1142 'Interpreter', 'text', ...
1143 'VerticalAlignment', 'bottom', ...
1144 'ButtonDownFcn', [], ...
1145 'CreateFcn', (@local_CreateFcn, [], ''), ...
1146 'DeleteFcn', [], ...
1147 'BusyAction', 'queue', ...
1148 'HandleVisibility', 'off', ...
1149 'HelpTopicKey', '', ...
1150 'HitTest', 'on', ...
1151 'Interruptible', 'on', ...
1152 'SelectionHighlight', 'on', ...
1153 'Serializable', 'on', ...
1154 'Tag', '', ...
1155 'UserData', [], ...
1156 'Behavior', struct(), ...
1157 'Visible', 'on', ...
1158 'XLimInclude', 'on', ...
1159 'YLimInclude', 'on', ...
1160 'ZLimInclude', 'on', ...
1161 'CLimInclude', 'on', ...
1162 'ALimInclude', 'on', ...
1163 'Clipping', 'off';
1164 h19 = get(h17, 'xlabel');
1165 set(h19,
1166 'Parent', h17,
1167 'Units', 'data',
1168 'FontUnits', 'points',
1169 'BackgroundColor', 'none',
1170 'Color', [0 0 0],
1171 'LineColor', 'none',
1172 'EraseMode', 'normal',
1173 'DVIMode', 'auto',
1174 'FontAngle', 'normal',
1175 'FontName', 'Helvetica',
1176 'FontSize', 10,
1177 'FontWeight', 'normal',
1178 'HorizontalAlignment', 'center',
1179 'LineStyle', '-',
1180 'LineWidth', 0.5,
1181 'Margin', 2,
1182 'Position', [0.498355263157895 -0.0788590604026846 1.00005459937205],
1183 'Rotation', 0,
1184 'String', '',
1185 'Interpreter', 'text',
1186 'VerticalAlignment', 'cap',
1187 'ButtonDownFcn', [],
1188 'CreateFcn', (@local_CreateFcn, [], ''),
1189 'DeleteFcn', [],
1190 'BusyAction', 'queue',
1191 'HandleVisibility', 'off',
1192 'HelpTopicKey', '',
1193 'HitTest', 'on',...
1196 'Interruptible','on',...
1197 'SelectionHighlight','on',...
1198 'Serializable','on',...
1199 'Tag',''...
1200 'UserData',[],...
1201 'Behavior',struct(),...
1202 'Visible','on',...
1203 'XLimInclude','on',...
1204 'YLimInclude','on',...
1205 'ZLimInclude','on',...
1206 'ClimInclude','on',...
1207 'ALimInclude','on',...
1208 'Clipping','off');
1209
1210 h20 = get(h17,'ylabel');
1211
1212 set(h20,...
1213 'Parent',h17,....
1214 'Units','data',....
1215 'FontUnits','points',....
1216 'BackgroundColor','none',....
1217 'Color',[0 0 0],....
1218 'EdgeColor','none',....
1219 'EraseMode','normal',....
1220 'DVMode','auto',....
1221 'FontAngle','normal',....
1222 'FontName','Helvetica',....
1223 'FontSize',10,...
1224 'FontWeight','normal',....
1225 'HorizontalAlignment','center',....
1226 'LineStyle','-',....
1227 'LineWidth',0.5,....
1228 'Margin',2,....
1229 'Position',[-0.09375 0.4949644295302 1.00005459937205],....
1230 'Rotation',90,....
1231 'String',''....
1232 'Interpreter','tex',....
1233 'VerticalAlignment','bottom',....
1234 'ButtonDownFcn',[],....
1235 'CreateFcn',{@local_CreateFcn,[],''}....
1236 'DeleteFcn',[],....
1237 'BusyAction','queue',....
1238 'HandleVisibility','off',....
1239 'HelpTopicKey',''....
1240 'HitTest','on',....
1241 'Interruptible','on',....
1242 'SelectionHighlight','on',....
1243 'Serializable','on',....
1244 'Tag',''....
1245 'UserData',[],....
1246 'Behavior',struct(),....
1247 'Visible','on',....
1248 'XLimInclude','on',....
1249 'YLimInclude','on',....
1250 'ZLimInclude','on',....
1251 'ClimInclude','on',....
1252 'ALimInclude','on',....
1253 'Clipping','off');
1254
1255 h21 = get(h17,'zlabel');
1256
1257 set(h21,...
1258 'Parent',h17,....
1259 'Units','data',....
APPENDIX A. DOECAD - SOURCE

```
    'FontUnits', 'points', ...
    'BackgroundColor', 'none', ...
    'Color', [0 0 0], ...
    'EdgeColor', 'none', ...
    'EraseMode', 'normal', ...
    'DVIMode', 'auto', ...
    'FontAngle', 'normal', ...
    'FontName', 'Helvetica', ...
    'FontSize', 10, ...
    'FontWeight', 'normal', ...
    'HorizontalAlignment', 'right', ...
    'LineStyle', '-', ...
    'LineWidth', 0.5, ...
    'Position', [-0.0838815789473684 1.09228187919463 1.00005459937205], ...
    'Rotation', 0, ...
    'String', '', ...
    'Interpreter', 'tex', ...
    'VerticalAlignment', 'middle', ...
    'ButtonDownFcn', [], ...
    'CreateFcn', @(local_CreateFcn, [], '') ....
    'DeleteFcn', [], ...
    'BusyAction', 'queue', ...
    'HandleVisibility', 'off', ...
    'HelpTopic', '', ...
    'HitTest', 'on', ...
    'Interruptible', 'on', ...
    'SelectionHighlight', 'on', ...
    'Serializable', 'on', ...
    'Tag', '', ...
    'UserData', [], ...
    'Behavior', struct(), ...
    'Visible', 'off', ...
    'XLimInclude', 'on', ...
    'YLimInclude', 'on', ...
    'ZLimInclude', 'on', ...
    'CLimInclude', 'on', ...
    'ALimInclude', 'on', ...
    'Clipping', 'off');
    appdata = [];  
    appdata.lastValidTag = 'pop_pic';
    h22 = uicontrol(...
      'Parent', h2, ...
      'Units', 'normalized', ...
      'BackgroundColor', [1 1 1], ...
      'Callback', 'DOECAD (''pop_pic_Callback'', gcbo, [], guidata(gcbo))', ...
      'Position', [0.472622478386167 0.265690376569038 0.2161383285302 0.0481171548117155], ...
      'String', {'2D'; '3D'}, ...
      'Style', 'popupmenu', ...
      'Value', 1, ...
      'CreateFcn', @(local_CreateFcn, 'DOECAD(''pop_pic_CreateFcn'', gcbo, [], guidata(gcbo)), appdata)', ...
      'Tag', 'pop_pic', ...
      'Behavior', get (0, 'defaultuicontrolBehavior'));
    appdata = [];  
    appdata.lastValidTag = 'pushbutton22';
    h23 = uicontrol(...
      'Parent', h2, ...
      'Units', 'normalized', ...
```

'Callback', 'DOECAD(''pushbutton22_Callback'', gcbo, [], guidata(gcbo))',
'Position', [0.902017291066282 0.267782426778243 0.046109510086 55],
'String', 'Callback',
'Tag', 'pushbutton22',
'Behavior', get(0, 'defaultuicontrolBehavior'),
'CreateFcn', (@local_CreateFcn, '', appdata);
appdata = [];
appdata.lastValidTag = 'pop_pic_type';

h24 = uicontrol(...
'Parent', h2,...
'Units', 'normalized',
'BackgroundColor', [1 1 1],
'Callback', 'DOECAD(''pop_pic_type_Callback'', gcbo, [], guidata(gcbo))',
'Position', [0.244956772334294 0.265690376569038 0.21613832853 59],
'String', { 'Orginal'; 'Result' },
'Style', 'popupmenu',
'Value', 1,
'CreateFcn', (@local_CreateFcn, 'DOECAD(''pop_pic_type_CreateFcn'', gcbo, [], guidata(gcbo)), appdata),
'Tag', 'pop_pic_type',
'Behavior', get(0, 'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'uipanel9';

h25 = uipanel(...
'Parent', h1,...
'FontSize', 12,
'Title', 'Kinoform (Phase) Plane',
'TitlePosition', 'centertop',
'Tag', 'uipanel9',
'Behavior', get(0, 'defaultuipanelBehavior'),
'Clipping', 'on',
'Position', [0.498567335243553 0 0.5 0.998188405797101],
'CreateFcn', (@local_CreateFcn, '', appdata);

appdata = [];
appdata.lastValidTag = 'asdasdasd';

h26 = uipanel(...
'Parent', h25,...
'Title', 'Export kinoform',
'Tag', 'asdasdasd',
'Behavior', get(0, 'defaultuipanelBehavior'),
'Clipping', 'on',
'Position', [0.0318840579710145 0.0246212121212121 0.94202898550 7246],
'CreateFcn', (@local_CreateFcn, '', appdata);

appdata = [];
appdata.lastValidTag = 'export_phase';

h27 = uicontrol(...
'Parent', h26,...
'Units', 'normalized',
'Callback', 'DOECAD(''export_phase_Callback'', gcbo, [], guidata(gcbo))',
'Position', [0.4330218068535682 0.779310344827586 0.214953271028037 0.158620689655172],
'String', 'Export',
'Tag', 'export_phase';
1381 'Behavior', get(0,'defaultuicontrolBehavior'),...
1382 'CreateFcn', @local_CreateFcn, '', appdata); ...
1383
1384 appdata = [];  
1385 appdata.lastValidTag = 'export_method';
1386
1387 h28 = uicontrol(...
1388 'Parent',h26,...
1389 'Units','normalized',...
1390 'BackgroundColor',[1 1 1],...
1391 'Callback','DOECAD(''export_method_Callback'',gcbo,[],guidata(gcbo))',...
1392 'Position',[0.0280373831775701 0.786206896551724 0.352024922118 380.151724137931035],...
1393 'String', ( 'Bitmap'; 'DXF (Layers)'; 'DXF (3D)'; 'Matlab (.mat)' ),...
1394 'Style', 'popupmenu',...
1395 'Value', 1,...
1396 'CreateFcn', (@local_CreateFcn, 'DOECAD(''export_method_CreateFcn'',gcbo,[],guidata(gcbo))', appdata), ...
1397 'Tag','export_method',...
1398 'Behavior', get(0,'defaultuicontrolBehavior'));
1399
1400 appdata = [];  
1401 appdata.lastValidTag = 'uipanel12';
1402
1403 h29 = uipanel(...
1404 'Parent',h26,...
1405 'Title','Manufacturing',...
1406 'Tag','uipanel12',...
1407 'Behavior',get(0,'defaultuipanelBehavior'),...
1408 'Clipping','on',...
1409 'Position',[0.0155763239875389 0.0275862068965517 0.96573208722 74140.779310344827586],...
1410 'CreateFcn', (@local_CreateFcn, 'DOECAD', '', appdata) );
1411
1412 appdata = [];  
1413 appdata.lastValidTag = 'man_lambda';
1414
1415 h30 = uicontrol(...
1416 'Parent',h29,...
1417 'Units','normalized',...
1418 'BackgroundColor',[1 1 1],...
1419 'Callback','DOECAD(''man_lambda_Callback'',gcbo,[],guidata(gcbo))',...
1420 'HorizontalAlignment','right',...
1421 'Position',[0.0947712418300654 0.505154639175258 0.114379084967 320.216494845360825],...
1422 'String', '670',...
1423 'Style', 'edit',...
1424 'CreateFcn', (@local_CreateFcn, 'DOECAD(''man_lambda_CreateFcn'',gcbo,[],guidata(gcbo))', appdata), ...
1425 'Tag','man_lambda',...
1426 'Behavior', get(0,'defaultuicontrolBehavior'));
1427
1428 appdata = [];  
1429 appdata.lastValidTag = 'man_angle';
1430
1431 h31 = uicontrol(...
1432 'Parent',h29,...
1433 'Units','normalized',...
1434 'BackgroundColor',[1 1 1],...
1435 'Callback','DOECAD(''man_angle_Callback'',gcbo,[],guidata(gcbo))',...
1436 'HorizontalAlignment','right',...
1437 'Position',[0.0915032679738562 0.0927835051546392 0.1209150326797390.216494845360825],...
1438 'String', '20',...
'Style', 'edit', ...
'CreateFcn', (@local_CreateFcn, 'DOECAD(@man_angle_CreateFcn', gcbo, [], guidata(gcbo)), appdata) ...
'Tag', 'man_angle', ...
'Behavior', get(0, 'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'text9';
h32 = uicontrol(...
'Parent', h29,...
'Units', 'normalized'...
'Position', [0.0588235294117647 0.742268041237113 0.183006535947 0.22680412371134],...
'String', 'Wavelength:', ...
'Style', 'text', ...
'Tag', 'text9', ...
'Behavior', get(0, 'defaultuicontrolBehavior'), ...
'CreateFcn', (@local_CreateFcn, 'text', appdata));

appdata = [];
appdata.lastValidTag = 'text10';
h33 = uicontrol(...
'Parent', h29,...
'Units', 'normalized'...
'Position', [0.0326797385620915 0.350515463917526 0.235294117647 0.154639175257732],...
'String', 'Fan-out angle:', ...
'Style', 'text', ...
'Tag', 'text10', ...
'Behavior', get(0, 'defaultuicontrolBehavior'), ...
'CreateFcn', (@local_CreateFcn, 'text', appdata));

appdata = [];
appdata.lastValidTag = 'text11';
h34 = uicontrol(...
'Parent', h29,...
'Units', 'normalized'...
'HorizontalAlignment', 'left', ...
'Position', [0.238562091503268 0.123711340206186 0.0751633986928105 0.154639175257732],...
'String', 'nm', ...
'Style', 'text', ...
'Tag', 'text11', ...
'Behavior', get(0, 'defaultuicontrolBehavior'), ...
'CreateFcn', (@local_CreateFcn, 'text', appdata));

appdata = [];
appdata.lastValidTag = 'text13';
h35 = uicontrol(...
'Parent', h29,...
'Units', 'normalized'...
'HorizontalAlignment', 'left', ...
'Position', [0.238562091503268 0.123711340206186 0.0751633986928105 0.154639175257732],...
'String', 'deg', ...
'Style', 'text', ...
'Tag', 'text13', ...
'Behavior', get(0, 'defaultuicontrolBehavior'), ...
'CreateFcn', (@local_CreateFcn, 'text', appdata));
APPENDIX A. DOECAD - SOURCE

```matlab
appdata = [];
appdata.lastValidTag = 'man_pixel';

h36 = uicontrol(...
    'Parent', h29, ...
    'Units', 'normalized', ...
    'BackgroundColor', [0.854901960784314 0.854901960784314 0.854901960784314], ...
    'Callback', 'DOECAD(''man_pixel_Callback'', gcbo, [], guidata(gcbo))', ...
    'Enable', 'off', ...
    'HorizontalAlignment', 'right', ...
    'Position', [0.643790849673203 0.505154639175258 0.215686274509804 0.216494845360825], ...
    'String', '0', ...
    'Style', 'edit', ...
    'CreateFcn', (@local_CreateFcn, 'DOECAD(''man_pixel_CreateFcn'', gcbo, [], guidata(gcbo)), appdata), ...
    'Tag', 'man_pixel', ...
    'Behavior', get(0, 'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'man_depth';

h37 = uicontrol(...
    'Parent', h29, ...
    'Units', 'normalized', ...
    'BackgroundColor', [0.854901960784314 0.854901960784314 0.854901960784314], ...
    'Callback', 'DOECAD(''man_depth_Callback'', gcbo, [], guidata(gcbo))', ...
    'Enable', 'off', ...
    'HorizontalAlignment', 'right', ...
    'Position', [0.643790849673203 0.0927835051546392 0.215686274509804 0.216494845360825], ...
    'String', 'Pixel size:', ...
    'Style', 'text', ...
    'CreateFcn', (@local_CreateFcn, 'DOECAD(''man_depth_CreateFcn'', gcbo, [], guidata(gcbo)), appdata), ...
    'Tag', 'man_depth', ...
    'Behavior', get(0, 'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'text16';

h38 = uicontrol(...
    'Parent', h29, ...
    'Units', 'normalized', ...
    'HorizontalAlignment', 'right', ...
    'Position', [0.705882352941177 0.742268041237113 0.183006535947712 0.22680412371134], ...
    'String', 'Pixel size:', ...
    'Style', 'text', ...
    'CreateFcn', (@local_CreateFcn, 'text16', appdata), ...
    'Tag', 'text16', ...
    'Behavior', get(0, 'defaultuicontrolBehavior'));

appdata = [];
appdata.lastValidTag = 'text17';

h39 = uicontrol(...
    'Parent', h29, ...
    'Units', 'normalized', ...
    'Position', [0.679738562091503 0.350515463917526 0.235294117647059 0.154639175257732], ...
```
'String', 'Pi etch depth:
'Style', 'text'
'Tag', 'text17'
'Behavior', get(0, 'defaultuicontrolBehavior')
'CreateFcn', (@local_CreateFcn, '', appdata)

appdata = [];
appdata.lastValidTag = 'text18';
h40 = uicontrol(...
'Parent', h29, ...
'Units', 'normalized', ...
'HorizontalAlignment', 'left', ...
'Position', [0.88562091503268 0.536082474226804 0.05882352941176 0.154639175257732], ...
'String', 'nm'
'Style', 'text'
'Tag', 'text18'
'Behavior', get(0, 'defaultuicontrolBehavior')
'CreateFcn', (@local_CreateFcn, '', appdata)

appdata = [];
appdata.lastValidTag = 'text19';
h41 = uicontrol(...
'Parent', h29, ...
'Units', 'normalized', ...
'HorizontalAlignment', 'left', ...
'Position', [0.88562091503268 0.123711340206186 0.07516339869281 0.2154639175257732], ...
'String', 'nm'
'Style', 'text'
'Tag', 'text19'
'Behavior', get(0, 'defaultuicontrolBehavior')
'CreateFcn', (@local_CreateFcn, '', appdata)

appdata = [];
appdata.lastValidTag = 'man_n_1';
h42 = uicontrol(...
'Parent', h29, ...
'Units', 'normalized', ...
'BackgroundColor', [1 1 1], ...
'Callback', 'DOECAD(''man_n_1_Callback'', gcbo, [], guidata(gcbo)), ...
'HorizontalAlignment', 'right', ...
'Position', [0.415032679738562 0.505154639175258 0.11437908496732 0.2166494845360325], ...
'String', '1.444'
'Style', 'edit'
'CreateFcn', (@local_CreateFcn, 'DOECAD(''man_n_1_CreateFcn'', gcbo, [], guidata(gcbo)), appdata) , ...
'Tag', 'man_n_1'
'Behavior', get(0, 'defaultuicontrolBehavior'))

appdata = [];
appdata.lastValidTag = 'man_n_2';
h43 = uicontrol(...
'Parent', h29, ...
'Units', 'normalized', ...
'BackgroundColor', [1 1 1], ...
'Callback', 'DOECAD(''man_n_2_Callback'', gcbo, [], guidata(gcbo)), ...
'HorizontalAlignment', 'right', ...
APPENDIX A. DOECAD - SOURCE

1613 'Position', [0.411764705882353 0.0927835051546392 0.120915032679 0.216494845360825] ,...
1614 'String', '1',...
1615 'Style', 'edit',...
1616 'CreateFcn', (@local_CreateFcn,'DOECAD(''man_n_2_CreateFcn'',gcbo,[]),guidata(gcbo)''), appdata) ,...
1617 'Tag', 'man_n_2',...
1618 'Behavior', get(0,'defaultuicontrolBehavior')) ;
1619
1620 appdata = [] ;
1621 appdata . lastValidTag = 'text20' ;
1622
1623 h44 = uicontrol(...
1624 'Parent', h29, ...
1625 'Units', 'normalized', ...
1626 'HorizontalAlignment', 'right', ...
1627 'Position', [0.359477124183007 0.752577319587629 0.2222222222222 0.216494845360825] ,...
1628 'String', 'n ( kinoform): ',...
1629 'Style', 'text', ...
1630 'Tag', 'text20', ...
1631 'Behavior', get(0,'defaultuicontrolBehavior')) ,...
1632 'CreateFcn', (@local_CreateFcn, '', appdata) ) ;
1633
1634 appdata = [] ;
1635 appdata . lastValidTag = 'text21' ;
1636
1637 h45 = uicontrol(...
1638 'Parent', h29, ...
1639 'Units', 'normalized', ...
1640 'Position', [0.339869281045752 0.350515463917526 0.2614379084967 0.154639175257732] ,...
1641 'String', 'n ( surrounding): ',...
1642 'Style', 'text', ...
1643 'Tag', 'text21', ...
1644 'Behavior', get(0,'defaultuicontrolBehavior')) ,...
1645 'CreateFcn', (@local_CreateFcn, '', appdata) ) ;
1646
1647 appdata = [] ;
1648 appdata . lastValidTag = 'pushbutton21' ;
1649
1650 h46 = uicontrol(...
1651 'Parent', h26, ...
1652 'Units', 'normalized', ...
1653 'Callback', 'DOECAD(''pushbutton21_Callback'',gcbo,[]),guidata(gcbo)''), ..., ...
1654 'Position', [0.64797507788162 0.779310344827586 0.0498442367601246 0.158620689655172] ,...
1655 'String', '?', ...
1656 'Tag', 'pushbutton21', ...
1657 'Behavior', get(0,'defaultuicontrolBehavior')) ,...
1658 'CreateFcn', (@local_CreateFcn, '', appdata) ) ;
1659
1660 appdata = [] ;
1661 appdata . lastValidTag = 'import_phase' ;
1662
1663 h47 = uicontrol(...
1664 'Parent', h26, ...
1665 'Units', 'normalized', ...
1666 'Callback', 'DOECAD(''import_phase_Callback'',gcbo,[]),guidata(gcbo)''), ..., ...
1667 'Position', [0.721739130434783 0.335227272727273 0.2 0.0435606060606061] ,...
1668 'String', 'Import', ...
1669 'Tag', 'import_phase', ...
1670 'Behavior', get(0,'defaultuicontrolBehavior')) ,...
h50 = get(h48,'xlabel');
set(h50,...
'Parent',h48,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'fontWeight','normal',...
'HorizontalAlignment','center',...
'LineStyle','-',...
'LineWidth',0.5,...
'Margin',2,...
'Position',[0.496677740863787 -0.0780730897009967 1.00005459937 205],...
'Rotation',0,...
'String',' ',...
'Interpreter','tex',...
'VerticalAlignment','cap',...
'ButtonDownFcn',[],...
'CreateFcn', @local_CreateFcn, [], '' },... 
'DeleteFcn',[],...
'BusyAction','queue',...
'HandleVisibility','off',...
'HitTest','on',...
'Interruptible','on',...
'SelectionHighlight','on',...
'Serializable','on',...
'Tag',' ',...
'UserData',[],...
'Behavior',struct(),...
'visible','on',...
'XLimInclude','on',...
'YLimInclude','on',...
'ZLimInclude','on',...
'AlimInclude','on',...
'Clipping','on');

h51 = get(h48,'ylabel');
set(h51,...
'Parent',h48,...
'Units','data',...
'FontUnits','points',...
'BackgroundColor','none',...
'Color',[0 0 0],...
'EdgeColor','none',...
'EraseMode','normal',...
'DVIMode','auto',...
'FontAngle','normal',...
'FontName','Helvetica',...
'FontSize',10,...
'fontWeight','normal',...
'HorizontalAlignment','center',...
1861 'UserData', [], ...  
1862 'Behavior', struct(), ...  
1863 'Visible', 'off', ...  
1864 'XLimInclude', 'on', ...  
1865 'YLimInclude', 'on', ...  
1866 'ZLimInclude', 'on', ...  
1867 'CLimInclude', 'on', ...  
1868 'ALimInclude', 'on', ...  
1869 'Clipping', 'off');  
1870  
1871 appdata = {};  
1872 appdata.lastValidTag = 'transform_phase_to_pic';  
1873  
1874 h53 = uicontrol(...  
1875 'Parent', h25, ....  
1876 'Units', 'normalized', ...  
1877 'Callback', 'DOECAD(''transform_phase_to_pic_Callback'', gcbo, [], guidata(gcbo))'; ...  
1878 'Position', [0.0898550724637681 0.335227272727273 0.240579710144 928 0.0435606060606061], ...  
1879 'String', '<< Transform', ...  
1880 'Tag', 'transform_phase_to_pic', ...  
1881 'Behavior', 'get(0, 'defaultuicontrolBehavior'), ...  
1882 'CreateFcn', (@local_CreateFcn, ' ', appdata);  
1883  
1884 appdata = {};  
1885 appdata.lastValidTag = 'pop_phase';  
1886  
1887 h54 = uicontrol(...  
1888 'Parent', h25, ....  
1889 'Units', 'normalized', ...  
1890 'BackgroundColor', [1 1 1], ...  
1891 'Callback', 'DOECAD('''pop_phase_Callback'', gcbo, [], guidata(gcbo))''', ...  
1892 'Position', [0.411594202898551 0.337121212121212 0.2869565217391 3 0.0416666666666667], ...  
1893 'String', '{ '2D'; '3D' }', ...,  
1894 'Style', 'popupmenu', ...,  
1895 'Value', 1, ...  
1896 'CreateFcn', (@local_CreateFcn, 'DOECAD('''pop_phase_CreateFcn'', gcbo, [], guidata(gcbo))', appdata), ...  
1897 'Tag', 'pop_phase', ...  
1898 'Behavior', 'get(0, 'defaultuicontrolBehavior')); ...  
1899  
1900 appdata = {};  
1901 appdata.lastValidTag = 'pushbutton23';  
1902  
1903 h55 = uicontrol(...  
1904 'Parent', h25, ....  
1905 'Units', 'normalized', ...  
1906 'Callback', 'DOECAD('''pushbutton23_Callback'', gcbo, [], guidata(gcbo))'', ...  
1907 'Position', [0.330434782608696 0.335227272727273 0.0463768115942029 0.0435606060606061], ...  
1908 'String', '?', ...  
1909 'Tag', 'pushbutton23', ...  
1910 'Behavior', 'get(0, 'defaultuicontrolBehavior'), ...  
1911 'CreateFcn', (@local_CreateFcn, ' ', appdata); ...  
1912  
1913 appdata = {};  
1914 appdata.lastValidTag = 'pushbutton24';  
1915  
1916 h56 = uicontrol(...  
1917 'Parent', h1, ...  
1918 'Units', 'normalized', ...  
1919 'Callback', 'DOECAD('''pushbutton24_Callback'', gcbo, [], guidata(gcbo))'', ...
APPENDIX A. DOECAD - SOURCE

1920 \[\text{'Position'}, [0.335243553008596 0.0 0.166189111747851 0.0923913043478261]\], ...
1921 \[\text{'String'}, \text{'Tutorial'}, ...
1922 \[\text{'Tag'}, \text{'pushbutton24'}, ...
1923 \[\text{'Behavior'}, \text{get}(0, \text{'defaultuicontrolBehavior'}), ...
1924 \[\text{'CreateFcn'}, (\text{@local_CreateFcn}, ' ', \text{appdata}) \];
1925
1926 \text{appdata} = []; \text{appdata}.lastValidTag = 'Untitled_2';
1927
1928 \text{h57} = \text{uicontextmenu}(...
1929 \[\text{'Parent'}, \text{h1}, ...
1930 \[\text{'Callback'}, \text{DOECAD}('untitled_2_callback', gcbo, [], guiData(gcbo)), ...
1931 \[\text{'Tag'}, 'untitled_2', ...
1932 \[\text{'Behavior'}, \text{get}(0, \text{'defaultuicontextmenuBehavior'}), ...
1933 \[\text{'CreateFcn'}, (\text{@local_CreateFcn}, ' ', \text{appdata}) \];
1934
1935 \text{appdata} = []; \text{appdata}.lastValidTag = 'Untitled_2';
1936
1937 \text{h58} = \text{uipanel}(...
1938 \[\text{'Parent'}, \text{h1}, ...
1939 \[\text{'Title'}, 'Status', ...
1940 \[\text{'Tag'}, 'uipanel13', ...
1941 \[\text{'Behavior'}, \text{get}(0, \text{'defaultuipanelBehavior'}), ...
1942 \[\text{'Clipping'}, 'on', ...
1943 \[\text{'Position'}, [-0.00143266475644699 0 0.330945558739255 0.0978260869565217], ...
1944 \[\text{'CreateFcn'}, (\text{@local_CreateFcn}, ' ', \text{appdata}) \];
1945
1946 \text{appdata} = []; \text{appdata}.lastValidTag = 'busy';
1947
1948 \text{h59} = \text{uicontrol}(...
1949 \[\text{'Parent'}, \text{h58}, ...
1950 \[\text{'Units'}, 'normalized', ...
1951 \[\text{'ForegroundColor'}, [1 0 0], ...
1952 \[\text{'Position'}, [0.299559471365639 0.5 0.343612334801762 0.578947368421053], ...
1953 \[\text{'String'}, '\text{'== Busy =='}, ...
1954 \[\text{'Style'}, 'text', ...
1955 \[\text{'Tag'}, 'busy', ...
1956 \[\text{'Behavior'}, \text{get}(0, \text{'defaultuicontrolBehavior'}), ...
1957 \[\text{'Visible'}, 'off', ...
1958 \[\text{'CreateFcn'}, (\text{@local_CreateFcn}, ' ', \text{appdata}) \];
1959
1960 \text{appdata} = []; \text{appdata}.lastValidTag = 'status';
1961
1962 \text{h60} = \text{uicontrol}(...
1963 \[\text{'Parent'}, \text{h58}, ...
1964 \[\text{'Units'}, 'normalized', ...
1965 \[\text{'Position'}, [0.0 0.33954558739255 0.0978260869565217 0.330945558739255], ...
1966 \[\text{'String'}, '', ...
1967 \[\text{'Style'}, 'text', ...
1968 \[\text{'Tag'}, 'status', ...
1969 \[\text{'Behavior'}, \text{get}(0, \text{'defaultuicontrolBehavior'}), ...
1970 \[\text{'Visible'}, 'off', ...
1971 \[\text{'CreateFcn'}, (\text{@local_CreateFcn}, ' ', \text{appdata}) \];
1972
1973 \text{hsingleton} = \text{h1};
1974
1975 \text{appdata} = []; \text{appdata}.lastValidTag = 'status';
1976
1977 \text{appdata} = []; \text{appdata}.lastValidTag = 'status';
1978
1979 \% --- Set application data first then calling the CreateFcn. 
1980
function local_CreateFcn(hObject, eventdata, createfcn, appdata)
    if ~isempty(appdata)
        names = fieldnames(appdata);
        for i = 1:length(names)
            name = char(names(i));
            setappdata(hObject, name, getfield(appdata, name));
        end
    end
    if ~isempty(createfcn)
        eval(createfcn);
    end
end

% --- Handles default GUIDE GUI creation and callback dispatch
function varargout = gui_mainfcn(gui_State, varargin)
    gui_StateFields = {'gui_Name',
                      'gui_Singleton',
                      'gui_OpeningFcn',
                      'gui_OutputFcn',
                      'gui_LayoutFcn',
                      'gui_Callback'};
    gui_Mfile = 'gui.m';
    for i = 1:length(gui_StateFields)
        if ~isfield(gui_State, gui_StateFields{i})
            error('Could not find field %s in the gui_State struct in GUI M-file %s', gui_StateFields{i}, gui_Mfile);
        elseif isequal(gui_StateFields{i}, 'gui_Name')
            gui_Mfile = [gui_State.(gui_StateFields{i}), '.m'];
        end
    end
    numargin = length(varargin);
    if numargin == 0
        % DOECAD
        gui_Create = 1;
        elseif isequal(ishandle(varargin{1}), 1) && ispc && iscom(varargin{1}) && isequal(varargin{1}.gcb, 1)
            % DOECAD(ACTIVEX, ...)
            gui_Create = 1;
        elseif ischar(varargin{1}) && numargin>1 && isequal(ishandle(varargin{2}), 1)
            % DOECAD('CALLBACK', hObject, eventdata, handles, ....)
            gui_Create = 0;
        else
            % DOECAD(...)
            gui_Create = 1;
        end
        if gui_Create == 0
            varargin{1} = gui_State.gui_Callback;
            if nargout
                [varargout{1:nargout}] = feval(varargin{:});
            else
                feval(varargin{:});
APPENDIX A. DOECAD - SOURCE

2043          end
2044  else
2045       if gui_State.gui_Singleton
2046          gui_SingletonOpt = 'reuse';
2047       else
2048          gui_SingletonOpt = 'new';
2049       end
2050
2051         \% Open fig file with stored settings. Note: This executes all
2052         \% specific CreateFunctions with an empty HANDLES structure.
2053
2054         \% Do feval on layout code in m-file if it exists
2055         \% feval on layout code in m-file if it exists
2056         if ~isempty(gui_State.gui_LayoutFcn)
2057            gui_hFigure = feval(gui_State.gui_LayoutFcn, gui_SingletonOpt);
2058            \% openfig (called by local_openfig below) does this for gui
2059            \% without
2060            \% the LayoutFcn. Be sure to do it here so gui show up on screen.
2061            movegui(gui_hFigure,'onscreen')
2062         else
2063            gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
2064            \% If the figure has InGUIInitialization it was not completely
2065            \% created
2066            \% on the last pass. Delete this handle and try again.
2067            if isappdata(gui_hFigure, 'InGUIInitialization')
2068                delete(gui_hFigure);
2069                gui_hFigure = local_openfig(gui_State.gui_Name, gui_SingletonOpt);
2070            end
2071         end
2072
2073         \% Set flag to indicate starting GUI initialization
2074         setappdata(gui_hFigure, 'InGUIInitialization',1);
2075
2076         \% Fetch GUIDE Application options
2077         gui_Options = getappdata(gui_hFigure, 'GUIDEOptions');
2078
2079         \% If user specified 'Visible','off' in p/v pairs, don't make the
2080         \% figure
2081         \% Generate HANDLES structure and store with GUIDATA. If there is
2082         \% user set GUI data already, keep that also.
2083         \% data = guidata(gui_hFigure);
2084         handles = guihandles(gui_hFigure);
2085         if ~isempty(handles)
2086            if isempty(data)
2087               data = handles;
2088            else
2089               names = fieldnames(handles);
2090               for k = 1:length(names)
2091                   data.(char(names(k))) = handles.(char(names(k)));
2092               \end
2093           end
2094       end
2095       guidata(gui_hFigure, data);
2096   end
2097
2098         \% If user specified 'Visible','off' in p/v pairs, don't make the
2099         \% visible.
gui_MakeVisible = 1;
for ind=1:2:length(varargin)
    if length(varargin) == ind
        break;
    end
end
len1 = min(length('visible'),length(varargin(ind)));
len2 = min(length('off'),length(varargin(ind+1)));
if ischar(varargin(ind)) && ischar(varargin(ind+1)) && ...
    strcmpi(varargin(ind),'visible',len1) && len2 > 1
    if strcmpi(varargin(ind+1),'off',len2)
        gui_MakeVisible = 0;
    elseif strcmpi(varargin(ind+1),'on',len2)
        gui_MakeVisible = 1;
    end
end

% Check for figure param value pairs
for index=1:2:length(varargin)
    if length(varargin) == index || ~ischar(varargin(index))
        break;
    end
    try set(gui_hFigure, varargin(index), varargin(index+1)), catch break, end
end

% If handle visibility is set to 'callback', turn it on until finished
% with OpeningFcn
gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
if strcmpi(gui_HandleVisibility, 'callback')
    set(gui_hFigure,'HandleVisibility', 'on');
end
feval(gui_State.gui_OpeningFcn, gui_hFigure, [], guidata(gui_hFigure), varargin{:});
if ishandle(gui_hFigure)
    % Update handle visibility
    set(gui_hFigure, 'HandleVisibility', gui_HandleVisibility);
    % Make figure visible
    if gui_MakeVisible
        set(gui_hFigure, 'Visible', 'on')
        if gui_Options.singleton
            setappdata(gui_hFigure, 'GUIOnScreen', 1);
        end
    end
    % Done with GUI initialization
    rmappdata(gui_hFigure, 'InGUIInitialization');
end

% If handle visibility is set to 'callback', turn it on until finished
% with
% OutputFcn
if ishandle(gui_hFigure)
    gui_HandleVisibility = get(gui_hFigure,'HandleVisibility');
    if strcmpi(gui_HandleVisibility, 'callback')
        set(gui_hFigure, 'HandleVisibility', 'on');
    end
    gui_Handles = guidata(gui_hFigure);
else
    gui_Handles = [];
end
if nargout
    [varargout{1:nargout}] = feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
else
    feval(gui_State.gui_OutputFcn, gui_hFigure, [], gui_Handles);
end
if ishandle(gui_hFigure)
    set(gui_hFigure,'HandleVisibility', gui_HandleVisibility);
end

function gui_hFigure = local_openfig(name, singleton)
    setappdata(0,'OpenGuiWhenRunning',1);

    % openfig with three arguments was new from R13. Try to call that first, if
    % failed, try the old openfig.
    try
        gui_hFigure = openfig(name, singleton, 'auto');
    catch
        % OPENFIG did not accept 3rd input argument until R13.
        % toggle default figure visible to prevent the figure
        % from showing up too soon.
        gui_OldDefaultVisible = get(0,'defaultFigureVisible');
        set(0,'defaultFigureVisible','off');
        gui_hFigure = openfig(name, singleton);
        set(0,'defaultFigureVisible',gui_OldDefaultVisible);
    end
    rmappdata(0,'OpenGuiWhenRunning');
Appendix B

LCM - Source

B.1 LCM_gui.m

function LCM_gui()
clear all;
close all;
load('LCM_default_setting.mat');
the_end=false;
while ~the_end
  m=menu('LCM Menu - ',setting_name,...
     'Undersök en bild',...
     'Live feed',...
     'Capture',...
     'Ladda XYZ',...
     'Generera en bild',...
     'Inställningar',...
     'Avsluta');
  switch m
    case 1
      try
        g_analysis(ltp, lxyz, r, c_ang,h);
        catch
          uipause(errordlg(lasterr));
      end
    case 2
      try
        g_live_m(ltp, lxyz, r, c_ang);
        catch
          uipause(errordlg(lasterr));
      end
    case 3
      try
        g_capture();
        catch
          uipause(errordlg(lasterr));
      end
    case 4
      try
        pic=[];XYZ=[];
        uipause('LOAD');
        if size(pic,1)>0 && size(XYZ,1)>0
          g_handle_XYZ(ltp, lxyz, r, c_ang, pic, XYZ,h,ctp);
        end
        catch
          uipause(errordlg(lasterr));
      end
    end
  end
end
clear pic XYZ;
case 5	ry
    g_generate(ltp, lxyz, r, c_ang, h);
catch
    uiwait(errordlg(lasterr));
end
case 6	ry
    [ltp, lxyz, r, c_ang, h, setting_name]=g_settings(ltp, lxyz, r, c_ang, h, setting_name);
catch
    uiwait(errordlg(lasterr));
end
otherwise
    the_end=true;
end
case 6
end
end

function [ltp, lxyz, r, c_ang, h, setting_name]=g_settings(ltp, lxyz, r, c_ang, h, setting_name)
done_g_s=false;
while ~ done_g_s
    switch menu(["Inställningar - ", setting_name,'
    'Visa setup','
    'Andra',...
    'Spara','Ladda','Set as default','Huvudmeny'])
case 1 %view setup
    l_beam=LCM_beams(ltp, lxyz, [min(r) max(r)]);
    c_beam=LCM_beams(LCM_cam_ang([1 1;1 1], c_ang),[0 0],[min(r) max(r)]);
    [x_m,y_m]=meshgrid([min(c_beam(1,:)) max(c_beam(1,:))],...
    [ min(c_beam(2,:)) max(c_beam(2,:))]);
    z_m=x_m;z_m(:,1)=h;
    figure(1);
    plot3(lxyz(1),lxyz(2),lxyz(3),'r*');
    hold on;
    plot3(0,0,0,'bo');
    mesh(x_m,y_m,z_m,'EdgeColor',[0 0 0], 'FaceColor','none');
    for i=1:size(l_beam,3)
        plot3(l_beam(:,1,i),l_beam(:,2,i),l_beam(:,3,i),r');
    end
    for i=1:size(c_beam,3)
        plot3(c_beam(:,1,i),c_beam(:,2,i),c_beam(:,3,i),'b','LineWidth',2);
    end
    title(setting_name);
    axis equal;
    uiwait(1);

    case 2 %edit
    prompt={["Enter the distance of interest (on the form 1:10:200) [cm]:'....
    'Enter the distance of the calibration picture [cm]:'....
    'Enter the camera angles &hight; width\&hight; ["(o)]:'....
    'Enter laser position from the camera \(x \& y \& z\) [cm]:'....
    'Enter setup name:']};
    name='Input for Peaks function';
    numlines=1;
defaultanswer = {[\texttt{num2str}(r(1)),':',\texttt{num2str}(r(2)-r(1)),':',
\texttt{num2str}(r(end))],...
\texttt{num2str}(h),... 
\texttt{num2str}(\texttt{round}(c\_ang(1)*1800/pi)/10),'; ',
\texttt{num2str}(\texttt{round}(c\_ang(2)*1800/pi)/10),... 
\texttt{num2str}(lxyz(1)),',',\texttt{num2str}(lxyz(2)),',',
\texttt{num2str}(lxyz(3))],...
\texttt{setting\_name};

options.Resize = 'on';
options.WindowStyle = 'normal';
done = false;

while ~ done

answer = inputdlg (prompt , name , numlines , defaultanswer , options);

try
    if size(answer,1)>0
        r=\texttt{eval} (\texttt{char} (answer(1)));
        h=\texttt{eval} (\texttt{char} (answer(2)));
        c\_ang=\texttt{eval} ([['',\texttt{char} (answer(3)),'']]*pi/180;
        lxyz=\texttt{eval} ([['',\texttt{char} (answer(4)),'']]);
    \texttt{setting\_name} = \texttt{char} (answer(5));
    \texttt{end}
    done = true;
    catch
        \texttt{uwait} (\texttt{errordlg} ('Felaktig inmatning ', 'Fel'));
        defaultanswer = answer;
    \texttt{end}
\texttt{end}

if size(answer,1)>0
    [filename , pathname] = \texttt{uigetfile}({'*.jpg;*.tif;*.png;*.gif
    ;*.bmp','All Image Files';...
    '*.*','All Files' }, 'Pick a
calibration picture');
    if isequal(filename,0)
        \texttt{disp}('User selected Cancel')
    else
        calib\_im\_raw = \texttt{imread} (\texttt{fullfile} (pathname , filename));
        [\texttt{row} , \texttt{col} , \texttt{calib\_pic}] = \texttt{LCM\_ident} (calib\_im\_raw);
        \texttt{ctp} = \texttt{LCM\_cam\_ang} (calib\_pic , c\_ang , col , row);
        \texttt{ltp} = \texttt{LCM\_calib} (lxyz , ctp , h);
        \texttt{end}
    \texttt{end}
\texttt{end}
case 3 % Save
    \texttt{uisave} ({'\texttt{ltp}' , 'lxyz' , 'r' , 'c\_ang' , 'h' , 'setting\_name'},'
    LCM\_New\_setting.mat');
case 4 % Load
    \texttt{uiopen} ('LOAD');
case 5 % Make default
    \texttt{save} ('LCM\_default\_setting.mat' , 'ltp' , 'lxyz' , 'r' , 'c\_ang' , 'h' ,
    'setting\_name');
case 6 % Main menu
    done\_g\_s = true;
\texttt{end}
\texttt{end}
\texttt{end}

function g\_analysis (ltp , lxyz , r , c\_ang , h)
    [filename , pathname] = \texttt{uigetfile}({'*.jpg;*.tif;*.png;*.gif;*.bmp','All
    Image Files';...
    '*.*','All Files' }, 'Pick a
picture');
    if isequal(filename,0)
disp('User selected Cancel')
else
  disp(['User selected ', fullfile(pathname, filename)]);
  pic_raw = imread(fullfile(pathname, filename));
  [row col pic] = LCM_ident(pic_raw, size(ltp,2));
  ctp = LCM_cam_ang(pic, c_ang, col, row);
  XYZ = LCM_measure(lxyz, ltp, ctp, 0.1, r);
  try
    g_handle_XYZ(ltp, lxyz, r, c_ang, pic, XYZ, h, ctp);
  catch
    uiwait(errordlg(lasterr));
  end
end

function g_handle_XYZ(ltp, lxyz, r, c_ang, pic, XYZ, h, ctp)
done = false;
while ~done
  ma = menu('Analys', 'Plotta', 'Spara', 'Huvudmeny');
  switch ma
    case 1
      ok_button = 1;
      options = {'User points', 'Laser Beams', 'Camera Beams', 'Picture',
                 'Setup', 'Interpolated surface (linear)', ...
                 'Interpolated surface (cubic)', 'Interpolated surface
(nearset)', 'Facit'};
      fig = 10;
      while ok_button == 1
        [selected, ok_button] = listdlg('PromptString', 'Select
stuff to plot', 'ListString', options);
        if find(selected == 1)
          figure(fig);
          plot3(XYZ(1,:), XYZ(2,:), XYZ(3,:), 'r*');
          hold on;
        end
        if find(selected == 2)
          figure(fig);
          l_beam = LCM_beams(ltp, lxyz, [min(r) max(r)]);
          hold on;
          for i = 1:size(l_beam, 3)
            plot3(l_beam(1,:,i), l_beam(2,:,i), l_beam(3,:,i), 'r');
          end
        end
        if find(selected == 3)
          figure(fig);
          c_beam = LCM_beams(ctp, [0 0 0], [min(r) max(r)]);
          hold on;
          for i = 1:size(c_beam, 3)
            plot3(c_beam(1,:,i), c_beam(2,:,i), c_beam(3,:,i), 'b');
          end
        end
        if find(selected == 4)
          figure(fig);
          c_frame = LCM_beams(LCM_cam_ang([1 1; 1 1], c_ang), [0 0 0], r);
          [foo, index] = min((h - c_frame(3,:,1)).^2);
          x_p = [min(c_frame(1,index,:)) max(c_frame(1,index,:))];
          y_p = [min(c_frame(2,index,:)) max(c_frame(2,index,:))];
          [x_p, y_p] = meshgrid(x_p, y_p);
          z_p = x_p .* z_p + h;
          hold on;
        end
    end
  end
end

end
surf(x_p,y_p,z_p,'CData',pic,'FaceColor','texturemap')

clear c_frame x_p y_p z_p;

if find(selected == 5)
    figure(fig);
    hold on;
    c_beam=LCM_beams(LCM_cam_ang([1 1;1 1],c_ang),[0 0 0],[min(r) max(r)]);
    plot3(lxyz(1),lxyz(2),lxyz(3),'r*');
    plot3(0,0,0,'bo');
    for i=1:size(c_beam,3)
        plot3(c_beam(1,:,i),c_beam(2,:,i),c_beam(3,:,i),'b','LineWidth',2);
    end
end

if find(selected == 6)
    LCM_interp(XYZ,'linear',fig);
    hold on;
end

if find(selected == 7)
    LCM_interp(XYZ,'cubic',fig);
    hold on;
end

if find(selected == 8)
    LCM_interp(XYZ,'nearest',fig);
    hold on;
end

if ok_button==1
    figure(fig);
    axis equal;
end

if find(selected == 9)
    FACIT=[];
    uiopen('LOAD');
    if size(FACIT,1)>0
        figure(fig)
        mesh(FACIT(:,1),FACIT(:,2),FACIT(:,3),...
             'FaceColor',[0 0 0],'FaceColor','none');
        hold on;
    end
    clear FACIT;
end

if ok_button==1
    figure(fig);
    set(fig,'RendererMode','Manual')
    set(fig,'Renderer','OpenGL')
    hold off;
    uiwait(fig);
end

case 2
    uisave({'XYZ','pic','ctp'},'XYZ_New.mat');
case 3
    done=true;
end
end

function g_generate(ltp, lxyz, r, c_ang, h)
l_beam=LCM_beams(ltp, lxyz, r);
x=linspace(min(l_beam(1,:)), max(l_beam(1,:)),50);
y=linspace(min(l_beam(2,:)), max(l_beam(2,:)),50);
\[ [X \, Y] = \text{meshgrid}(x, y); \]
\[ x = X; y = Y; \]
\[ Z = \text{zeros}(\text{size}(X)) + h; \]
\[
\text{prompt} = \{'\text{Enter the formula for } Z(X, Y, h): }' \ldots \\
'\text{Enter picture dimensions } Z1024 \, 6802'\}; \\
\text{name}' = 'Input for Peaks function'; \\
\text{numlines}' = 1; \\
\text{defaultanswer}' = \{'\text{peaks}(X/20, Y/20) * 10', '1728 2304'\}; \\
\text{options}' = \{'\text{Resize}' = 'on', \\
'\text{WindowStyle}' = 'normal', \\
'\text{Interpreter}' = 'tex'\}; \\
\text{done}' = \text{false}; \\
\text{while } \sim \text{ done} \\
\text{answer}' = \text{inputdlg}(\text{prompt}, \text{name}, \text{numlines}, \text{defaultanswer}, \text{options}); \\
\text{try} \\
\text{if size(\text{answer},1)} > 0 \\
\text{Z} = \text{Z} + \text{real}((\text{eval}(\text{char}(\text{answer}(1))))); \\
picdim = \text{eval}(['\),' \text{char}(\text{answer}(2)), ']'); \\
\text{end} \\
\text{done}' = \text{true}; \\
\text{catch} \\
\text{uiwait(\text{errordlg('Felaktig inmatning','Fel')});} \\
\text{defaultanswer}' = \text{answer}; \\
\text{end} \\
\text{end} \\
\text{if size(\text{answer},1)} > 0 \\
\text{FACIT(:,1)} = X; \\
\text{FACIT(:,2)} = Y; \\
\text{FACIT(:,3)} = Z; \\
\text{[pic, ctp]} = \text{LCM\_genpic(lxyz, ltp, FACIT, r, picdim, c\_ang)}; \\
\text{done} = \text{false}; \\
\text{while } \sim \text{ done} \\
\text{mp}' = \text{menu('Resultat', 'Plotta', 'Spara Bild', 'Spara Höjder', 'Huvudmeny')}; \\
\text{switch mp} \\
\text{case 1} \\
\text{figure(1)} \\
\text{mesh}(X, Y, Z, ['EdgeColor', [0 0 0], 'FaceColor', 'none']); \\
\text{axis equal}; \\
\text{uiwait(1)}; \\
\text{case 2} \\
\text{[filename, pathname] = uiputfile({'*.jpg;*.tif;*.png;*.gif \\
*:.*;*.bmp', 'All Image Files'},... \\
'*.*', 'All Files'), 'Save Image';... \\
'GEN\_newfile.bmp');} \\
\text{if isequal(filename, 0)} \\
\text{disp('User selected Cancel') } \\
\text{else} \\
\text{imwrite(pic, fullfile(pathname, filename));} \\
\text{end} \\
\text{case 3} \\
\text{uisave({'FACIT'}, 'FACIT\_New.mat');} \\
\text{case 4} \\
\text{done} = \text{true}; \\
\text{end} \\
\text{end} \\
\text{end} \\
\text{function g\_live\_m(ltp, lxyz, r, c\_ang)} \\
\text{close all;} \\
\text{done}' = \text{false};
table=[];
n=size(ltp,2);
sensitivity=0.9;
while ~ done
    ma=menu('Live View', 'Live', 'Visa tabell', 'Generera Tabell', 'Spara tabell', 'Ladda tabell', 'Huvudmeny');
    switch ma
        case 1 % live
            try
                answer=inputdlg('Sensitivity', 'Set sens', 1, {num2str(sensitivity*100)});
                if size(answer,1)>0
                    sensitivity=eval(char(answer))/100;
                    l_beam=LCM_beams(ltp, lxyz, r);
                    axis_lim=[min(l_beam(1,:)), max(l_beam(1,:)), ... min(l_beam(2,:)), max(l_beam(2,:)), ... min(r), max(r)];
                end
            catch
                uiwait(errordlg(lasterr));
            end
        case 2 % Singlett
            try
                h=figure(1); subplot(1,2,1);
                imagesc(table(:, :, 3)); colormap(hot);
                subplot(1,2,2);
                mesh(table(:, :, 3)); colormap(hot);

                set(h,'RendererMode','Manual')
                set(h,'Renderer','OpenGL')
                uiwait(h);
            catch
                uiwait(errordlg(lasterr));
            end
        case 3 % Generera
            prompt='Enter picture dimmensions Z1024 680Z';
            name='Input dimensions';
            numlines=1;
            defaultanswer=['480 640'];
            options.Resize='on';
            options.WindowStyle='normal';
            options.Interpreter='tex';
            good_ans=false;
            while ~ good_ans
                try
                    answer=inputdlg(prompt, name, numlines, defaultanswer, options);
                    if size(answer,1)>0
                        picdim=eval(['[',char(answer),']']);
                        pic=zeros(picdim);
                        good_ans=true;
                    end
                catch
                    uiwait(errordlg(lasterr));
                end
            end
        case 4 % spara tabell
        end
    end
end

function g_capture()
    hwInfo = imaqhwinfo('winvideo');
    device1 = hwInfo.DeviceInfo(1);
    vidFormat = device1UPPORTED_FORMATS;
    vidFormat = vidFormat([1 7 4]);
    sel_res = listdlg('PromptString','Select a Resolution','SelectionMode','single','ListString',vidFormat);
    try
        vidobj = videoinput('winvideo',1,char(vidFormat(sel_res)),'ReturnedColorSpace','rgb');
        pic = getsnapshot(vidobj);
        [filename, pathname] = uiputfile({'*.jpg;*.tif;*.png;*.gif;*.bmp','All Image Files';... '.*','All Files'},'Save Image',... 'Capture_New.bmp');
        if isequal(filename,0)
            disp('User selected Cancel')
        else
            imwrite(pic, fullfile(pathname, filename));
        end
    end
end

function table = FFA_init(lxyz, ltp, c_ang, pic, r)
if nargin==0
    uiopen('LOAD');
    pic=zeros(round([480 640]/2));
    r=[10 300];
end
answer = inputdlg('Tolerance','Set tol',1,{num2str(0.2)});
tol = eval(char(answer));
pic = pic+1;
ctp = LCM_cam_ang(pic, c_ang);
XYZ = LCM_measure(lxyz, ltp, ctp, tol, r, true);
table(:,1) = reshape(XYZ(1,:), size(pic));
table(:,2) = reshape(XYZ(2,:), size(pic));
table(:,3) = reshape(XYZ(3,:), size(pic));
if nargin==0
    n= size(ltp, 2);
    uisave({'table','n'},'TABLE_new_table.mat');
end
end

function g_capture()
APPENDIX B. LCM - SOURCE

B.4 LCM_BEAMS.M

```matlab
function table=FFA_init(lxyz, ltp, c_ang, pic, r)
    if nargin==0
        uiopen('LOAD ');
        pic=zeros(round([480 640]/2));
        r=[10 300];
    end
    answer=inputdlg('Tolerance ','Set tol',1,{num2str(0.2)});
    tol=eval(char(answer));
    pic=pic*0+1;
    ctp=LCM_cam_ang(pic, c_ang);
    XYZ = LCM_measure(lxyz, ltp, ctp, tol, r, true);
    table (:,:,1)=reshape(XYZ(1,:),size(pic));
    table (:,:,2)=reshape(XYZ(2,:),size(pic));
    table (:,:,3)=reshape(XYZ(3,:),size(pic));
    if nargin==0
        n=size(ltp,2);
        uisave({'table', 'n'}, 'TABLE_new_table.mat');
    end
end
```

B.4 LCM_beams.m

```matlab
function beams=LCM_beams(ltp, l_xyz, r)
    l_t=ltp(1,:);
    l_p=ltp(2,:);
    z=r*cos(l_t)+l_xyz(3);
    x=r*(sin(l_t).*cos(l_p))+l_xyz(1);
    y=r*(sin(l_t).*sin(l_p))+l_xyz(2);
    beams(1,:,:)=x;
    beams(2,:,:)=y;
    beams(3,:,:)=z;
end
```

B.5 LCM_calib.m

```matlab
function [ l_t_p ] = LCM_calib(l_xyz, c_tp, z)
    c_t=c_tp(1,:);
    c_p=c_tp(2,:);
    c_r=z./cos(c_t);
    x=c_r.*sin(c_t).*cos(c_p)+l_xyz(1);
    y=c_r.*sin(c_t).*sin(c_p)+l_xyz(2);
    z=l_xyz(3);
    l_t_p=[acos(z./sqrt(x.^2 + y.^2 + z.^2)); atan2(y,x)];
end
```

B.6 LCM_cam_ang.m

```matlab
function [ c_tp ] = LCM_cam_ang(pic, c_ang, col, row)
    %returns theta & phi from a binary picture and the camera agles
    pic=pic(:,:,:);1;
    if (nargin==1)
        c_ang=[20; 20]*pi/180;
    end
end
```
B.7 LCM_frame_trigger.m

```matlab
function LCM_frame_trigger(obj, event, hImage)

% Display the current image frame.

global n_global;
global sensitivity_global;
global table_global;
global hPoints_global;
global f_global;
figure(f_global);
set(hImage, 'CData', event.Data);
frame = event.Data;
try
    [row col laser] = LCM_fast_ident(frame(:,:,1), n_global, sensitivity_global);
    row_round = round(size(table_global,1)*row/size(laser,1));
    col_round = round(size(table_global,2)*col/size(laser,2));
    for i = 1:size(row,2)
        XYZ(:,i) = table_global(row(i),col(i),:);
    end
    set(hPoints_global, 'XData', XYZ(1,:), 'YData', XYZ(2,:), 'ZData', XYZ(3,:));
    % FFA_trigger_interp(XYZ, hPoints_global)
catch
    disp(lasterr);
end
subplot(1,3,2);
global asd;
disp(asd);
image(laser>0);
%title(['Z:', num2str(XYZ(3,:)), ', ' rad:, num2str(row,:), ', ' col:, num2str(col,:))]);
axis image;
hold on;
plot(col, row, 'r*');
hold off;
drawnow
```

B.8 LCM_genpic.m

```matlab
function [pic, c_tp] = LCM_genpic(l_xyz, l_tp, XYZ, r, picdim, c_ang)
beams = LCM_beams(l_tp, l_xyz, r);
d_xyz = LCM_intersect(XYZ, beams);
```
```matlab
4 c=(picdim - [1 1]) ./ (2* tan(c_ang));
5 x=d_xyz(1,:);
6 y=d_xyz(2,:);
7 z=d_xyz(3,:);
8 col=round(c(2)*x./z+(picdim(2)+1)/2);
9 row=round(c(1)*y./z+(picdim(1)+1)/2);
10 col=col(find(col>=0 & col<=picdim(2)));
11 row=row(find(row>=0 & row<=picdim(1)));
12 col=col(find(col>=0 & col<=picdim(2)));
13 row=row(find(row>=0 & row<=picdim(1)));
14 if(nargout == 1)
15 pic=zeros(picdim);
16 for i=1:size(col,2)
17 pic(row(i),col(i))=1;
18 end
19 elseif (nargout == 2)
20 pic=zeros(picdim);
21 for i=1:size(col,2)
22 pic(row(i),col(i))=1;
23 end
24 c_tp=[acos(z./sqrt(x.^2 + y.^2 + z.^2)) ; atan2(y,x)];
25 else
26 error('Fel antal utpar. ');
27 end
28 end

B.9 LCM_ident.m

```
B.10 LCM_interp.m

```matlab
function [ X, Y, Z ] = LCM_interp( XYZ, type, v)
if nargin==1
    type='linear';
end
x = XYZ(1,:);
y = XYZ(2,:);
z = XYZ(3,:);
[X, Y]=meshgrid(linspace(min(x),max(x),200),linspace(min(y),max(y),200));
Z=griddata(x,y,z,X,Y,type);
if nargin==3
    figure(v);
    mesh(X,Y,Z,'EdgeColor',[0 0 1],...
    'FaceColor',[1 1 1],...
    'FaceLighting','none',...
    'EdgeLighting','flat');
end
end
```

B.11 LCM_linematch.m

```matlab
% A1=[0 0 0]';
% A2=[5 5 5]';
% B1=[0 5 5]';
% B2=[5 5 0]';
% mA = dot(cross(B2-B1,A1-B1),cross(A2-A1,B2-B1));
```
function xyz=LCM_linematch(A1, A2, B1, B2, tol, r)
  if nargin==4
tol=1e-2;
end
B0=zeros(3, size(B2,2));
dist(1:size(B2,2))=0;
for i=1:size(B2,2)
v1=cross(A2-A1,B2(:,i)-B1);
  nA = dot(cross(B2(:,i)-B1,A1-B1),v1);
  nB = dot(cross(A2-A1,A1-B1),v1);
  d = dot(v1,v1);
  B0(:,i) = B1 + (nB/d)*(B2(:,i)-B1);
  dist(i)=sqrt(sum((A1 + (nA/d)*(A2-A1)-B0(:,i)).^2));
end
dist(B0(3,:)<min(r))=NaN;
dist(B0(3,:)>max(r))=NaN;
[mini ix]=min(dist);
if mini<tol
  xyz=B0(:,ix);
else
  xyz=[NaN NaN NaN]';
end

B.12 LCM_live.m

function LCM_live(table,n,sensitivity,ax_lim)
close all;
global n_global;
global sensitivity_global;
global table_global;
global hPoints_global;
global f_global;
table_global= table;
sensitivity_global = sensitivity;
n_global=n;
hwInfo = imaqhwinfo('winvideo');
device1 = hwInfo.DeviceInfo(1);
vidFormat=device1.SupportedFormats;
vidFormat=vidFormat([11 14 8]);
sel_res = listdlg('PromptString','Select a Resolution','SelectionMode','single','ListString', vidFormat);
vidobj = videoinput('winvideo',1, char(vidFormat(sel_res)),'ReturnedColorSpace','rgb');
vidRes = get(vidobj, 'VideoResolution');
f = figure('Visible','off');
f_global=f;
imageRes = fliplr(vidRes);
subplot(1,3,1);
hImage = imshow(zeros(imageRes));
axis image;
subplot(1,3,2);
hLaser = imshow(zeros(imageRes));
axis image;
colormap(gray);
subplot(1,3,3);
hPoints_global=plot3(0,0,NaN,'r*');
axis(ax_lim)
B.13 LCM_measure.m

```matlab
function XYZ = LCM_measure(l_xyx, l_tp, c_tp, tol, r, verb)
    if nargin < 4
        tol = 0.2;
    end
    if nargin < 6
        verb = false;
    end

    l_beam = LCM_beams(l_tp, l_xyx, [0 1]);
    c_beam = LCM_beams(c_tp, [0 0 0], [0 1]);
    proc = 0;
    n = size(c_beam, 3);
    XYZ = zeros(3, n);
    if verb
        h = waitbar(0, 'Please wait...');
    end

    for c = 1:n
        XYZ(:, c) = LCM_linematch(c_beam(:, 1, c), c_beam(:, 2, c), l_beam(:, 1),
                                   squeeze(l_beam(:, 2, :)'), tol, r);
        if verb
            if proc < round(c/n*100)
                proc = round(c/n*100);
                waitbar(c/n);
            end
        end
    end
    if verb
        close(h);
    end
end
```
B.14 LCM hierarchy

Figure B.1: The hierarchy of the LCM software.