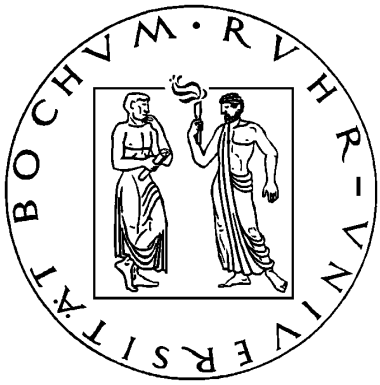
The LUCIFER Control Software: Operational and Observational Characteristics and NIR Spectroscopy of Galaxy Centers



Dissertation

zur
Erlangung des Grades
„Doktor der Naturwissenschaften"
der Fakultät für Physik und Astronomie
an der Ruhr–Universität Bochum

vorgelegt von
Dipl. Phys. Volker Knierim
aus
Mettingen

Bochum 2009

1. Gutachter: Prof. Dr. Ralf–Jürgen Dettmar
2. Gutachter: Prof. Dr. Susanne Hüttemeister

Datum der Disputation: 16.07.2009

This document has been created with LaTeX 2_ε

To my wife

Throughout this thesis the following typesetting conventions are used:

`typewriter` text indicates a software component, e.g., `GratingUnit` refers to the software service.

slanted text is used to identify physical components, .e.g., the *Grating Unit* of the LUCIFER instrument. Additionally, this style is also used to emphasize the corresponding text.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The night sky above the earth has fascinated humans at least since the beginning of civilization some 4000 years ago. For example, the Nebra Sky Disk is an impressive evidence of early astronomical interest (Meller 2002; Schlosser 2002). However, for more than three millennia observations were limited by the only available detector, the human eye. This changed when Galileo Galilei started to use a telescope to observe the universe 400 years ago. While the detections were still dependent on the eye, the light collecting area was not limited by the pupil of the eye anymore. It was now possible to see fainter objects that were previously unknown, and familiar objects, e.g., the moon and the five known planets, could be studied in greater detail because of the magnification telescopes provide. The new discoveries by Kepler led to a revolution of the view of the world, putting the earth, and thus also mankind, out of the center of the universe. Shortly afterwards, in 1687 Isaac Newtons "Philosophiae Naturalis Principia Mathematica" put physics on a new mathematical foundation.

The photographic plate replaced the eye as the primary detector in astronomy in the nineteenth century. It was now possible to image fainter astronomical objects with longer integration times and high spatial resolution. With the rapid evolution of electronics during the last century, other detectors like photomultipliers were developed. These had the benefit of higher dynamical range and sensitivity compared with photographic plates but lacked the spatial resolution of them. Only with the invention of Charge Coupled Devices (CCDs), which combine high sensitivity, greater dynamical range (10^5 compared to 10^3 for photographic plates, Kitchin 2008) and high spatial resolution, the "ideal" detector for optical astronomy was found. In the near infrared (NIR), similar progress has been made for detectors in the last decades.

While the detectors grew more and more sophisticated over time, so did the instruments that were built and attached to the telescopes. Today they provide different operation modes to allow imaging and/or spectroscopy, using focal reducers and different cameras to alter the optical setup of the telescope they are attached to. This allows to study different scientific questions in different wavelength regimes with the same telescope, maximizing its output. At the same time the complexity of, and technical challenges faced by the instruments have grown significantly. For example, for optical instruments only the detector, i.e., the Charge Coupled Device (CCD), needs to be cooled, in the NIR the whole instrument must ideally be cooled to suppress the thermal emission of the instrument itself.

A new instrument for the Large Binocular Telescope (LBT) is the LBT NIR spectroscopic Utility with Camera and Integral-Field Unit for Extragalactic Research (LUCIFER). It is fully cryogenic,

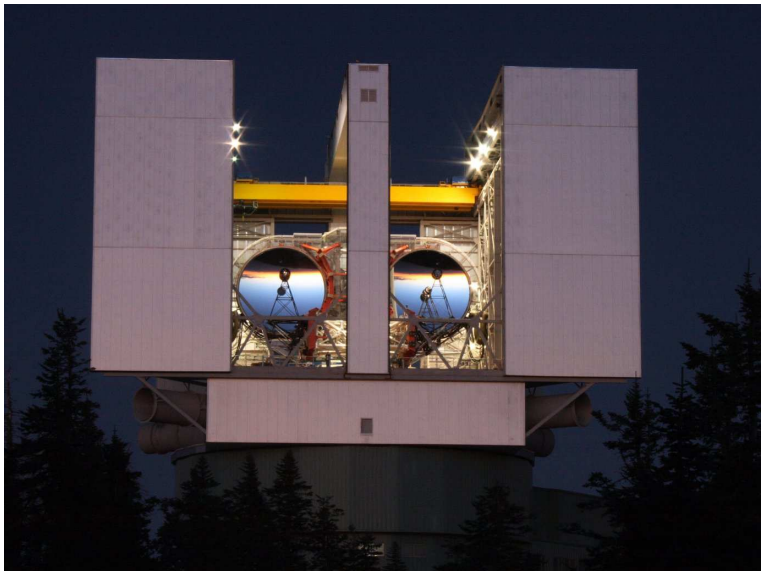


Figure 1.1: The LBT on Mt. Graham shortly after sunset. The photo was taken during the commissioning run in September 2008.

working at temperatures of T≈70 K and provides multiple operation modes for imaging, long slit spectroscopy (LSS) and multi object spectroscopy (MOS). Two identical instruments, LUCIFER 1 and LUCIFER 2 will finally be installed at the LBT. In the following, the term LUCIFER is used to describe properties that are common to both instruments. The individual instruments are explicitly referred to when necessary.

This chapter gives a brief overview, introducing the two main components, namely the LBT and the LUCIFER instrument. Following this description, some general particularities about observing in the NIR wavelength regime are provided. The chapter closes with an outline of the aims of this thesis.

1.1 The Large Binocular Telescope

The LBT is part of the Mt. Graham International Observatory near Safford, Arizona. It is located at Emerald Peak on Mt. Graham at an elevation of 3220 m. The LBT is the first telescope built with two 8.4 m mirrors on one common mount, providing the same light collecting area as one single mirror with 11.8 m diameter. The distance between the centers of the two mirrors is 14.4 m which results in the resolving power of an 22.6 m mirror when used in interferometric mode. It is currently the largest telescope in the world working at optical and NIR wavelengths (Hill & Salinari 2004). The main mirrors have a focal length of 9.6 m, which gives a focal ratio of f/1.14 for the primary mirrors and thus allows a compact telescope design. Different focal stations are used for each mirror. These are the prime focus, a direct Gregorian focus, three Bent Gregorian foci, plus one additional Nasmyth focus. The prime focus instruments as well as the secondary and tertiary mirrors are mounted on swing arms that can be moved in and out. Changing between the different foci is therefore quickly possible. Two of the Bent Gregorian foci will be used for interferometric instruments which, obviously, need both mirrors at the same time. The other focal stations are equipped with similar or twin instruments that can be used independently for each mirror. An

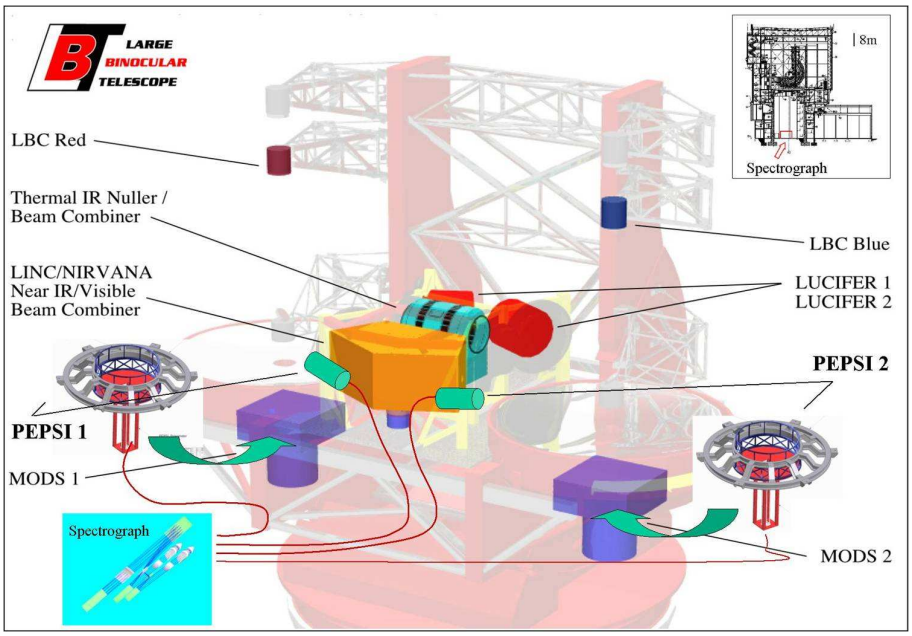


Figure 1.2: The location of the various instruments at the telescope. The image was taken from the Max-Planck-Institut für Astronomie, Heidelberg, Press Release 04-09-21, an online version is available at www.aip.de/pepsi. See also table 1.1 .

Table 1.1: Instruments for the LBT and their focal stations.

Focal station	Left Mirror	Right Mirror
Prime	LBC[a]Red	LBC Blue
Additional Nasmyth	PEPSI[b] 1	PEPSI 2
Direct Gregorian	MODS[c] 1/PEPSI 1	MODS 2/PEPSI 2
Bent Gregorian front	LUCIFER[d] 1	LUCIFER 2
Bent Gregorian center	LBTI[e]	
Bent Gregorian back	LINC/NIRVANA[f]	

[a]Large Binocular Camera
[b]Potsdam Echelle Polarimetric and Spectroscopic Instrument
[c]Multi Object Double Spectrograph
[d]LBT NIR spectroscopic Utility with Camera and Integral-Field Unit for Extragalactic Research
[e]LBT Intefrerometer
[f]LBT INteferometric Camera / Near-IR Visible Adaptive iNterferometer for Astronomy

exception is the PEPSI instrument. This instrument uses two foci (the Direct Gregorian and the additional Nasmyth) per mirror. The additional Nasmyth focus is mounted permanently, while the Direct Gregorian focus is shared with the MODS instrument, which has to be removed first when PEPSI should be used at this focal station. Tab. 1.1 and Fig. 1.2 give an overview of all instruments of the LBT.

When the telescope is fully operational, both secondary mirrors of the LBT will be adaptive and are designed to give a focal ratio of f/15 for the Gregorian foci instruments. The mirror shells

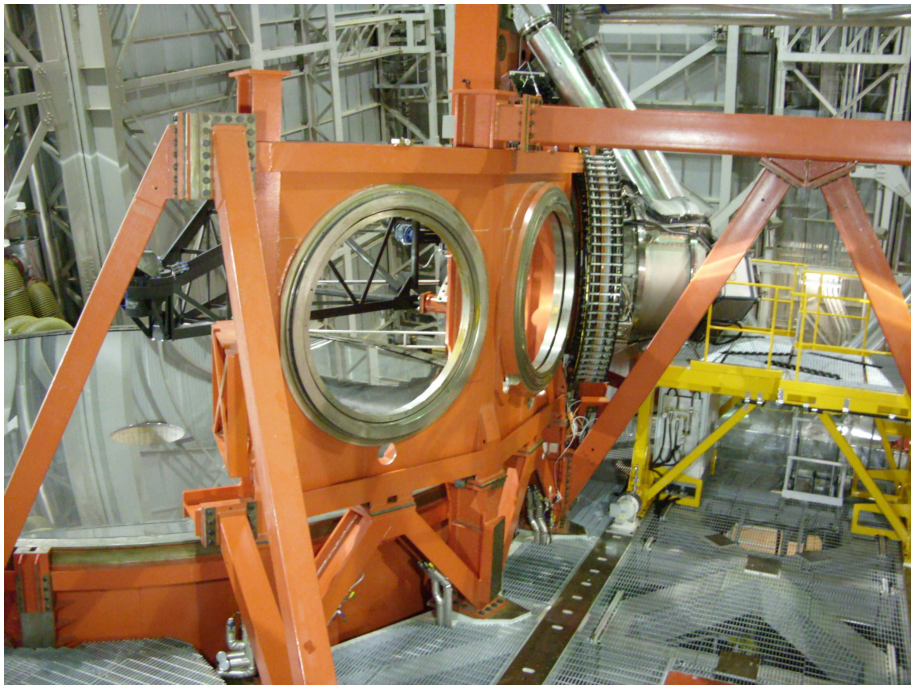


Figure 1.3: LUCIFER mounted at the LBT. To the left, the 8.4 m primary mirror of the telescope as well as the swing arm holding the tertiary mirror are visible. The yellow structure beneath the instrument is the bridge that is used when the mask cabinet exchange is made. The photo was taken during the commissioning run in February 2009.

with a diameter of 911 mm and a thickness of only 1.6 mm are mounted on 672 actuators, sitting on a hexapod structure (Martin et al. 2006). Due to delays in the construction of the thin mirror shells, a rigid secondary mirror has been constructed and is used for the commissioning of the first LUCIFER instrument. The commissioning of the adaptive secondary mirror will follow afterwards.

1.2 The LUCIFER instrument

The LUCIFER instrument is a collaboration between five different German institutions:

- Landessternwarte Heidelberg (LSW)
- Max-Planck-Institut für Astronomie, Heidelberg (MPIA)
- Max-Planck-Institut für Extraterrestrische Physik, Garching (MPE)
- Astronomisches Institut Ruhr–Universität–Bochum (AIRUB)
- Fachhochschule für Gestaltung Mannheim

The wavelength covered by the instrument ranges from 0.9 μm to 2.5 μm and thus includes the z, J, H and K NIR bands. As can be seen in Tab. 1.1, two LUCIFER instruments will be built and installed at the LBT. An integral field unit will not be installed in any of the instruments, however. Mandel et al. (2006) outlines the main optical characteristics of the LUCIFER instrument. These are summarized in Table 1.2.

Table 1.2: Optical Characteristics of the LUCIFER instrument.

Observing mode	Field of View	Resolution[a]
Seeing limited imaging	$4' \times 4'$	N.A.
Seeing limited LSS	$4' \times 4'$	10000
Seeing limited MOS	$4' \times 4'$	10000
Diffraction limited imaging	$30'' \times 30''$	N.A.
Diffraction limited LSS	$30'' \times 30''$	37100

[a]maximum possible resolution for this mode

There are five different observation modes foreseen for the instrument. Two of these modes are practically identical, namely the *seeing limited long slit spectroscopy* mode and the *seeing limited multi object spectroscopy* mode. Both use masks that are inserted into the focal plane of the instrument. The difference between the two modes lies in the availability of the masks for users. For LSS the masks are stored permanently in the instrument and available for all astronomers. The MOS masks are individually cut for each approved science project and inserted into the instrument prior to the planned observation. The main benefit of LUCIFER is, that this mask exchange can be carried out *while the instrument is cold* and thus no time is lost for warming up of the instrument and cooling it down again. To achieve this, the masks are inserted into a cabinet which is placed in one of the auxiliary cryostats. Two cryostats are then evacuated and cooled down the day before the mask exchange should take place. The empty cryostat is attached to an airlock located at the back of the instrument. The old cabinet that is currently inside of the instrument is moved through the airlock into the empty cryostat. Once this procedure is finished the first cryostat is removed and the second one – with new masks – is attached to the instrument. The cabinet with the new masks is moved through the airlock into the instrument. This mask exchange procedure can be carried out during daytime and therefore no time is lost for observations at night.

As can bee seen from Fig. 1.4 and Fig. 1.5, which show the optical path of the beam inside the LUCIFER instrument from two different viewing angles, the optical design of the instrument is very compact. The figures mentioned above as well as the following brief description are taken from Seifert (2002). Light enters the instrument through the entrance window, which is tilted by 15° and coated to reflect the visible light to the adaptive optics (AO) wavefront sensor. The focal plane is located 11.66 cm behind this entrance window and is the place where the MOS masks are inserted by the mask handling robot. After passing the focal plane the light enters the bent collimator of the instrument. The collimator consist of three lenses and four mirrors, two of which are movable. The first mirror (M1) is called the *Alignment Mirror* and intended to stay practically fixed during the operation of the instrument. It is only moved for initial optical alignment of the instrument. The second movable mirror (M4) is called *Compensation Mirror* and used to compensate movements of the image on the science detector caused by instrument flexure due to changing gravity vectors during observations. More information about the movable mirrors will be given in section 2.3.3.

The light propagates to the *Grating Unit* and is either reflected by a mirror (imaging mode of the instrument) or dispersed by one of the gratings in the spectroscopic mode. It subsequently passes through one of the three cameras called N1.8, N3.75, and N30, with focal lengths of

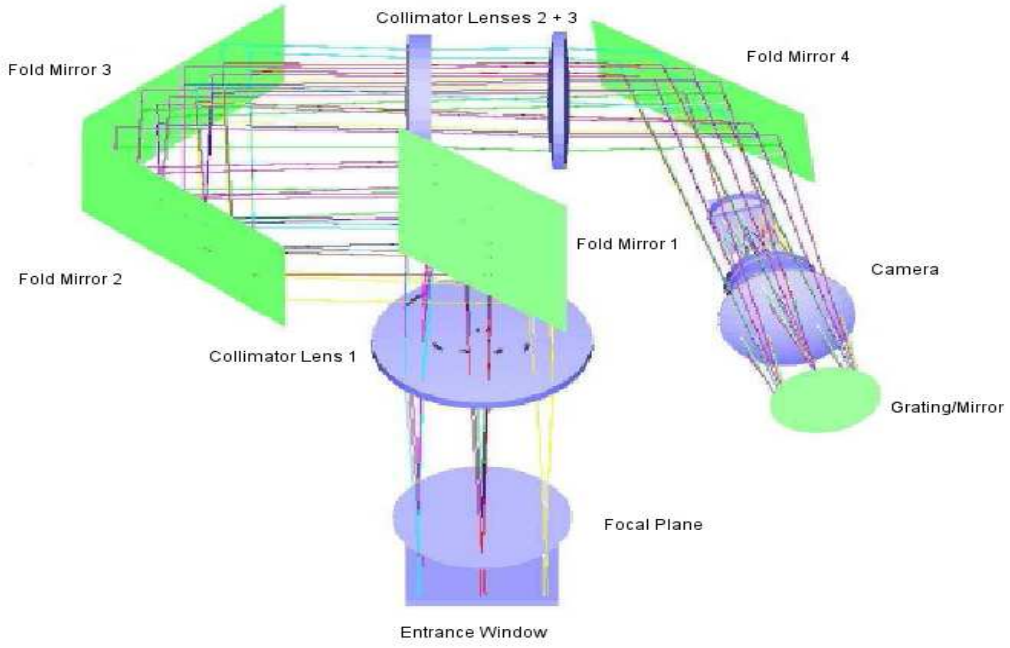


Figure 1.4: Optical path of the LUCIFER instrument tilted to show most optical components (Seifert 2002).

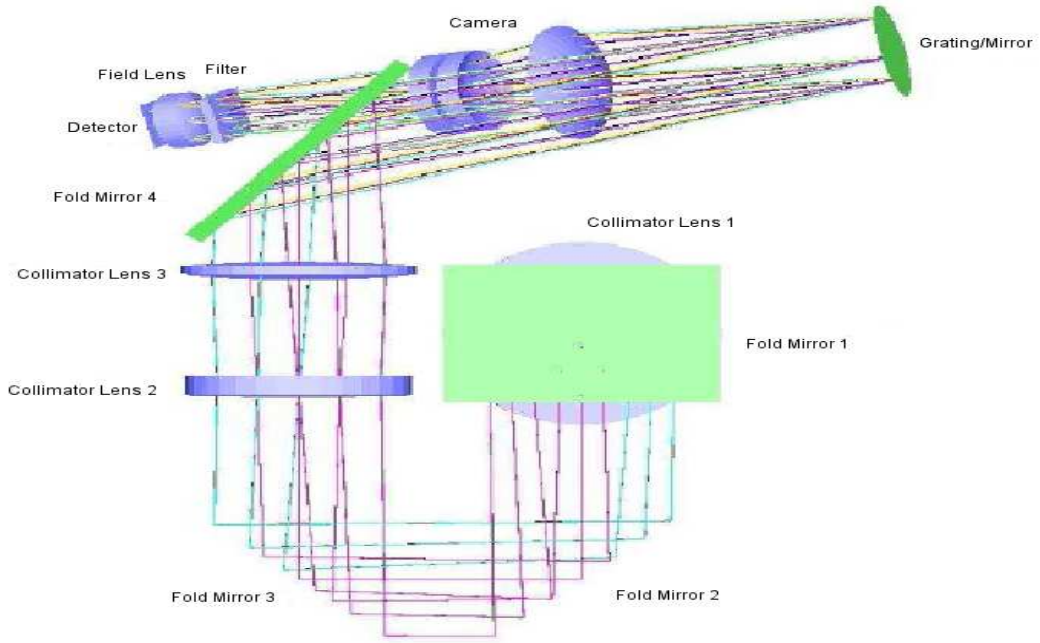


Figure 1.5: Optical path of the LUCIFER instrument (view from back) (Seifert 2002).

Table 1.3: List of the filters currently installed inside LUCIFER 1.

Position	Filter Wheel 1	Filter Wheel 2
1	Brackett-γ	clear
2	OH-hole 1060[a]	blind
3	clear	unused
4	blind	unused
5	Y1 1000	J
6	Y2 1070	unused
7	Paschen-γ	unused
8	Paschen-β	Ks
9	J-low 1200	unused
10	J-high 1297	z
11	PV-lens[b]	unused
12	OH-hole 1190	H
13	FeII 1644	unused
14	H_2 (1-0)S1	K
15	HeI 1086	order separation

[a] numbers indicate the central wavelength of the filter in nanometers
[b] pupil viewer

180 mm, 375 mm, and 3000 mm, respectively. The N1.8 and N3.75 cameras are used in seeing limited mode of the instrument and provide the full field of view (FOV) of $4' \times 4'$ of LUCIFER. The N30 camera is used in diffraction limited mode only and provides a FOV of $30'' \times 30''$. The light passed through both filter wheels, each of which can be equipped with up to 15 filters, before it finally reaches the science detector, which is sitting behind a field lens on a movable tray. Table 1.3 gives an overview of the filters that are currently installed inside LUCIFER 1.

1.3 NIR Astronomy

As mentioned at the beginning of this chapter, the origins of astronomy lie in the optical regime of the electromagnetic spectrum because these photons can be detected by the naked eye. From the beginning of modern astronomy in the 17$^{\text{th}}$ century, over nearly three centuries every telescope built was suited for optical wavelengths. Only with the development of modern electronics for radio receivers and the progress made in rocket science during the first half of the last century, astronomers were able to explore the universe at other wavelengths. While observations with radio telescopes can be carried out from the ground, the ability to launch satellites allowed the exploration of electromagnetic radiation that was inaccessible before because of the atmosphere of the earth. The transmission of electromagnetic waves in the atmosphere is shown in Fig. 1.6.

For NIR observations optical telescopes can be used. However, unlike its optical counterpart, NIR astronomy is a relatively young area mainly because suitable detectors are only available for the last couple of decades. Another problem is that although several windows exist between 0.8 μm and 2.5 μm where the atmosphere is transparent (see Fig. 1.6), even in these windows the atmosphere is emitting substantially because of its own average temperature of T$\approx$ 250 K and

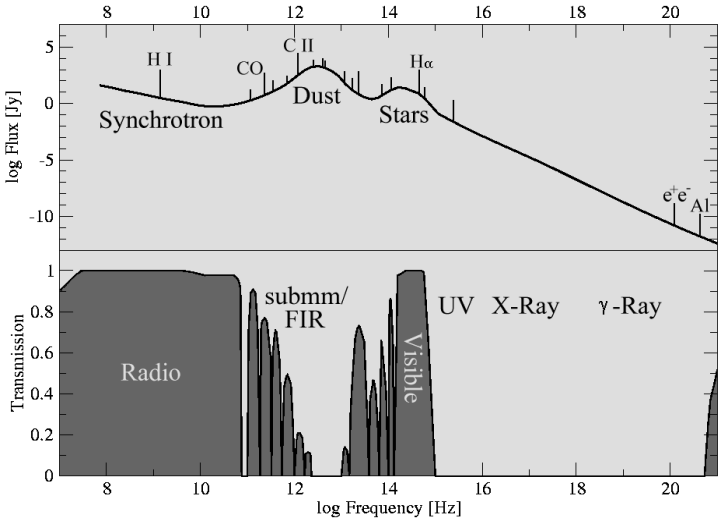


Figure 1.6: A sketch the spectral energy distribution of a spiral galaxy and some prominent emission lines (upper diagram). The transmission of the earths atmosphere (lower diagram).

line radiation of OH molecules. The glow from the sky is much brighter than most astronomical sources, implying that it must be removed from the observations before it is possible to analyze the data. Only modern detectors and data reduction software tools allow the efficient treatment of this data. The necessity to subtract the night sky brightness leads to significant overheads in the observations which can be larger than 50% of the total observation time. NIR spectroscopy is even more challenging than imaging, because the atmosphere has significant absorption lines even in the transparent windows of the spectrum. These absorption lines need to be carefully corrected, which requires additional calibration spectra and increases the overhead even more.

In spite of these inefficiencies and difficulties that are inherent in NIR astronomy the observations have a number of benefits over their optical counterparts. First, NIR emission from stars and other sources is much less absorbed by the interstellar medium than optical light. This allows to study objects, e.g., the galactic center, that would otherwise be heavily obscured. Secondly, the expansion of the universe and the subsequent redshift moves the well known optical spectra from distant galaxies into the near infrared when it reaches the earth. Thus, modern cosmology relies heavily on near infrared observations. Subsequently all modern observatories in the world have been or are being equipped with infrared instruments. One point that emphasizes the importance of infrared observations for the future of astronomy most is the fact that the James Webb Space Telescope (JWST), which will replace the Hubble Space Telescope (HST) in the near future, will operate only at wavelengths $\lambda \geq 0.6\mu$m (Gardner et al. 2006).

1.4 Aims and outline of the thesis

Understanding the driving mechanisms leading to the evolution of galaxies is of utmost importance for our understanding of the universe as a whole. In this context, the secular evolution of galaxies is one of the main research areas in astronomy nowadays. This thesis approaches this topic by studying the dominant stellar populations in the centers of galaxies of different Hubble type. This is done by using NIR K-band spectroscopy to measure absorption line indices of the CO

molecule. LUCIFER is an ideally suited instrument for this kind of research, providing unique new opportunities with its MOS mode. To prepare the planned LUCIFER observations spectra were obtained with a similar instrument from the European Southern Observatory (ESO), namely the Infrared Spectrometer and Array Camera (ISAAC) at the Paranal Observatory in Chile. These observations with ISAAC were successfully proposed during this thesis. During the commissioning of LUCIFER, additional spectra were obtained.

The focus of the software development for LUCIFER is to maximize the usability of the instrument for astronomers. Understanding the techniques of near infrared observations, and their application for the ISAAC observations, greatly helped to improve the software in this respect. It allowed for designing suitable software components to be able to provide convenient tools and user interfaces necessary for the operation of the instrument.

Details of the software package to control the LUCIFER instrument are presented in chapter 2. The software includes code that directly controls hardware components as well as more sophisticated services reflecting complex instrument components. Additionally, the observational and operational challenges of using a near infrared instrument are addressed by providing suitable tools to satisfy the needs of engineers as well as the needs of astronomers using the instrument during the night.

In chapter 3 the results of the ISAAC observations are discussed in the context of secular evolution. The data is analyzed using a new definition for the CO absorption strength (D_{CO}) that is proposed by Mármol-Queraltó et al. (2008). Because of the locations and width of the used absorption and continuum bands, this index is better suited to study galaxies than previous definitions. The observed galaxies are compared with each other and with other studies of stellar populations that used similar techniques.

Finally, the thesis is summarized and the future work in the project is outlined in chapter 4.

Chapter 2

The software

This chapter focuses on the main part of this thesis, the control software for the LUCIFER instrument. The full software package is presented first in an overview, which briefly summarizes the layout of the software by describing the functions of different layers representing increasing levels of abstraction. The description of the packages of the *Control Layer* follows, focusing in particular on the use of the `Journalizer` service, environment measurement, and the software control of the grating tilt and position switch readout electronics. After this, the general concepts of the *Instrument Layer* are described and explained using the specific examples of the `Grating Unit` and `Compensation Mirror` services. The last part of the description of the software package deals with the highest level of the software, namely the *Operation Layer*. Here, details about the user interaction principles that are applied for the software interaction are laid out. The management services of the software are introduced and different observation tools available for the astronomers are presented. At the end of the chapter a summary of the completed work and an outlook over the next necessary extensions of the software package for the future are presented.

2.1 Overview of the LUCIFER control software package

Most parts of the LUCIFER control software package (lcsp) are written in the *JAVA* programming language. Some external programs necessary for the operation of the instrument are written in other languages, namely `C` and `C++`. This includes the *GEneric InfraRed Software (GEIRS)* to control the detector read out. The telescope control commands that the instrument has to send to the LBT use the Instrument Interface (IIF). These software packages are written at the MPIA and in Tucson, respectively. To allow for a seamless integration into the control software, *JAVA* classes have been implemented to send the necessary commands to these external packages.

The software runs on a Sun Fire V880 server system under the *SOLARIS* operating system, because of hardware requirements for the detector readout electronics. However, the software is implemented as a distributed system, so it could be run on different computers using different operating systems (the latter as a benefit of the *JAVA* language) at the same time. This is briefly introduced in the next paragraph.

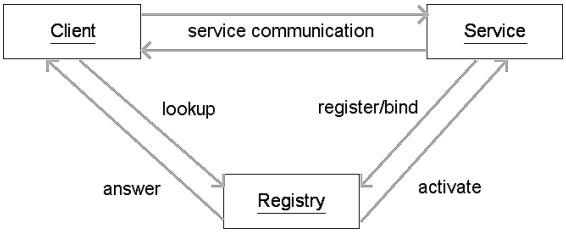


Figure 2.1: The communication for RMI services. First the `Service` registers itself with the `Registry`. If a `Client` needs to communicate with the service for the first time, it looks up the information using the `Registry`. The `Registry` answers with the connection details and starts (or "activates") the `Service` if it is not running. The communication between the client and the service is then performed directly.

2.1.1 General remarks

The LUCIFER control software is written as a distributed system in *JAVA* relying heavily on its remote method invocation (RMI) framework. This means that one or more services of the system can run on different computers that are connected with each other via a network. Since most of the software project is written completely in *JAVA*, using its native RMI functions is a natural choice. However, to realize the communication between the services, some common middleware like the Common Object Request Broker Architecture (CORBA) or the Internet Communications Engine (ICE) could also be used. In fact the communication with with the Telescope Control Software (TCS) is realized using ICE.

According to Grosso (2002, page 66) *"RMI is designed to make communication between two Java programs, running in separate JVMs*[1]*, as much like making a method call inside a single process as possible"*. From the programmers point of view this simply means that local objects and remote objects look the same. Using RMI interfaces to realize the distributed system saves a lot of programming time, because the necessary client side objects (commonly referred to as *stubs*) which handle the communication with the remote service via the network, are automatically generated by the compiler.

To manage the distributed system a lookup service is needed, to which all remote services register themselves (see Fig. 2.1). The clients use these lookup service to find out where the remote service can be contacted. Once the lookup has taken place the client communicates directly with the remote service. For RMI two lookup services, or registries, are provided: `rmiregistry` and `rmid`. The latter service allows to use *Activatable Remote Services*. This means that if a remote service, which is bound to the registry but currently not running, will be started when it is needed. This creates a very stable software system, because services are still available, even if they were suspended or have crashed for some reason. In this case a new instance is created automatically. Further discussion of RMI and its features would be far beyond the scope of this short introduction. For more details about this topic refer to, e.g., Grosso (2002).

2.1.2 Layered structure of the software

The software system is organized in a layered structure. Each layer reflects a different abstraction level of interaction with the instrument and/or solves different administration tasks for the system.

[1] Java Virtual Machines, i.e., processes

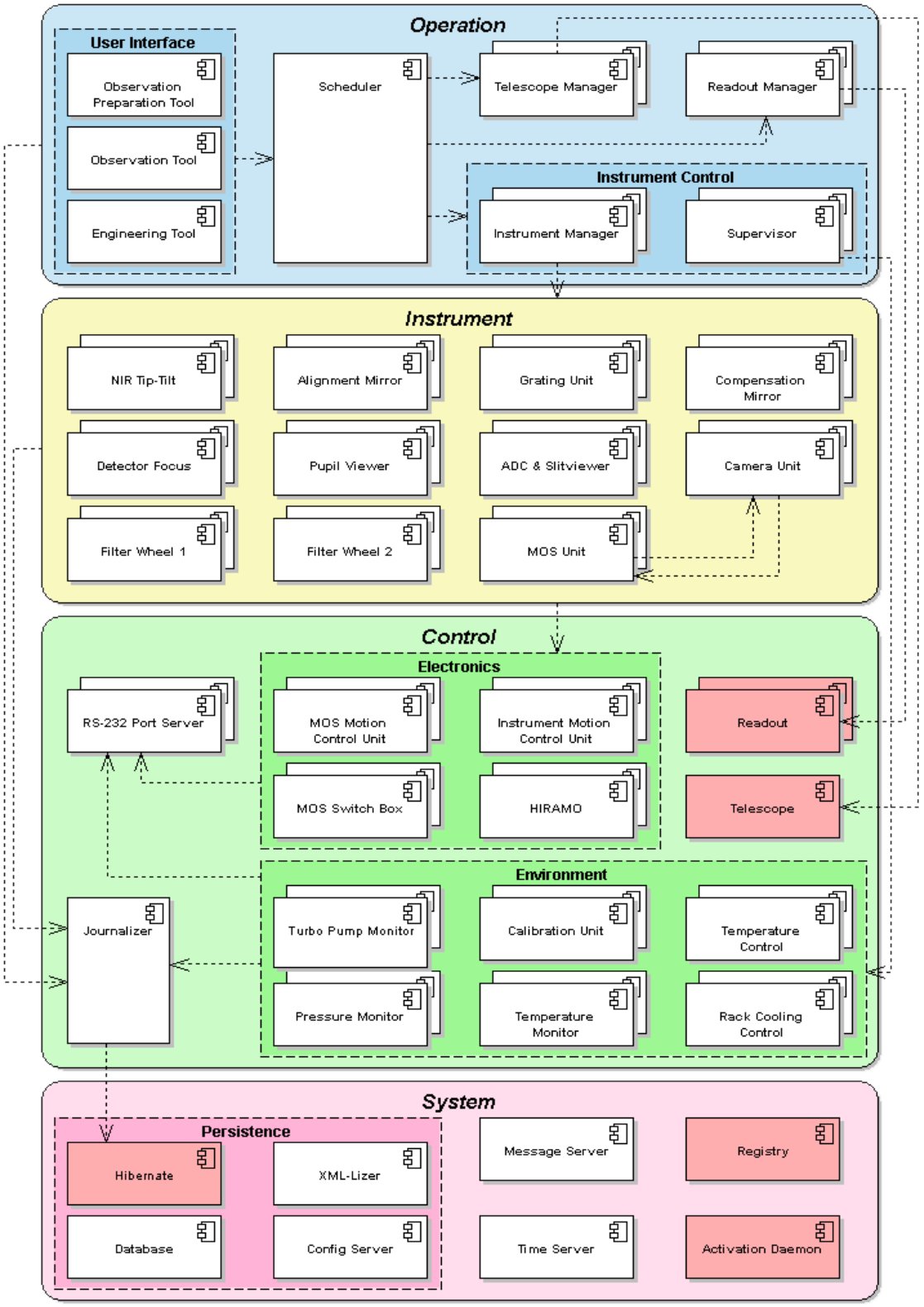


Figure 2.2: Overview of the LUCIFER control software package.

The structure of the software system is shown in Fig. 2.2. Using this approach, the complexity of the software in each layer remains manageable by shifting unnecessary difficulties to higher layer services or encapsulating them in lower layer ones. An example for the former benefit would be that, e.g., observation techniques need not to be known when writing code that controls a single instrument unit like the *Grating Unit*. An example for the latter benefit would be that, e.g. the same software controlling the *Grating Unit* does not need to deal with the details of sending the correct commands to the motor electronics to move the unit from one position to the next. Instead, the `GratingUnit` software service can focus on the main responsibilities of the unit, like the mapping of different tilt voltages to central wavelengths on the detector. See section 2.3.2 for more details about the `GratingUnit` software service.

The following subsections briefly describe the components of each layer of the lcsp as well as the main tasks that are solved by the corresponding layer. More details of the higher layers are given in Sects. 2.2 to 2.4.

System Layer

The lowest layer of the software package is the *System Layer*. It provides basic services necessary for the whole system. This includes a `ConfigService`, which provides functions to load and store configuration data dynamically for the distributed system. Additionally the `TimeService` provides time synchronization capabilities and the `MessageService` is used to store messages generated by the various services in a database and distribute them to any client which connects to the service. The *System Layer* of the software package was already implemented by K. Polsterer when the work for this thesis began. For details about this layer see Polsterer (in prep.).

Control Layer

The main task of the *Control Layer* is to provide a connection between the software and any hardware associated with the LUCIFER instrument. This includes components to control units of the instrument, i.e., motors and switches, as well as components for environment control, e.g., temperature and pressure measurements.

The `MCU`[2] service for example handles all communication with the motor control electronics. The `HIRAMO`[3] service controls the grating tilt voltage and is used to query the status of various switches inside the instrument using the hardware card which carries the same name as the service. The switches of the *MOS Unit* are controlled by the `SwitchBox` service, however. This is due to the fact that the *MOS Unit* hardware has been fully developed by the MPE, while the *HIRAMO* hardware has been developed at the MPIA.

Environment data is controlled by different services as well. Three separate temperature measurements are provided by the software. The `TemperatureMonitor` service measures and logs eight temperatures inside the instrument structure, while the `TemperatureControl` service controls two temperatures associated with the NIR science detector board. Finally, the `RackCoolingControl` service logs and controls the temperatures of the electronics rack outside the instrument. These electronics include the read-out electronics as well as the motor electronics mentioned above. The environment data control is completed with the measurement and logging of two instrument pressures. The `CalibrationUnit` service allows control of up to 8 lamps for

[2] Motion Control Unit
[3] HIgh Resolution Analog Measurement Output board

wavelength and detector sensitivity (i.e., flatfields) calibrations that are necessary for spectroscopic observations carried out with LUCIFER.

The logging of all instrument data (excluding messages, since these are stored by the `MessageService` directly) is done by the `Journalizer` service. Each service creates so–called `JournalizerObjects` when it needs to store new data and contacts the `Journalizer` which then takes care of the storage of this information in the database. The `Journalizer` can also be used to query the current instrument setup, since it keeps a collection of all instrument status objects it receives. Furthermore, the status of the instrument can be reconstructed from the database if this should be necessary. The `Journalizer` service plays an important role in the *Operation Layer* of the software because the functionality to get the instrument configuration without the need to interact with the hardware directly is used for user interaction. Details about this interaction with the instrument are given in Section 2.4.

The access to the two external software packages necessary for the operation of the instrument is located in the *Control Layer*, too. The first one, the `RMIGEIRSService` communicates with the *GEIRS* server using a socket connection. The service therefore acts like a client for the read out software and forwards all requests of the lcsp to this server. The second service, namely the `Telescope` service uses an ICE interface to communicate with the TCS which runs under the *CentOS* operating system on different machines.

Instrument Layer

The third layer of the software represents a software representation of the LUCIFER instrument itself. All components of the instrument that need to be controlled or somehow modified by the software are implemented as separate RMI services. This implies that the services in this layer rely heavily on the services in the *Control Layer*, since they know nothing (and should not need to know in the first place) about motor parameters or switch positions or other hardware related details. As a consequence, none of these services will run correctly, if needed services in the lower layer are not available. The *Instrument Layer* services are presented in detail in Section 2.3.

To model instrument setup changes in the *Instrument Layer*, the abstract concept of `States`, `Transitions` and `Sequences` has been used. It allows to view the different units as finite state machines with well defined conditions. This makes the operation of these units more robust and possible errors conditions can be detected easily. As an additional benefit complicated movements of a unit can be separated into smaller entities, which are easier to implement and debug from a programming side of view.

The services found in this layer focus on solving the operational difficulties of each represented instrument unit. For the `GratingUnit` service for example, different positions have to be selected, depending on the chosen observation mode. If one of the grating positions is selected, different tilt angles have to be applied to place a given central wavelength on the center of the detector. The conversion of the central wavelength to a corresponding tilt voltage is for instance one central requirement of the software at this level of interaction. Another example is the `CompensationMirror` service used to compensate image motions on the detector due to instrument flexure caused by changing rotation and elevation angles of the instrument and telescope, respectively. This service needs to calculate mirror movements depending on the current configuration of the instrument, i.e., the selected camera, rotator angle of the instrument, and the configuration of the telescope, i.e., the current elevation angle.

Although different services of the *Instrument Layer* require information from other services to function properly, direct interaction among the services is not desirable. This “horizontal”

interaction would induce a dependency between the involved services. As a result, the failure of one service may cause another service to fail as well. In the absence of horizontal interaction the other would not be affected, resulting in increased failure tolerance of the software. Furthermore synchronization problems are present for some services which put constraints on their operation. The `CompensationMirror` service provides a good example: when should the mirror be adjusted to compensate for flexure? This might depend on the observation that is currently running. Taking the just mentioned issues into account, it is clear that the software design cannot stop at this point, but instead one higher layer is needed to coordinate all *Instrument Layer* services. The *Operation Layer* of the software package provides these services.

Operation Layer

The *Operation Layer* is the highest layer of the LUCIFER control software package. It provides three fundamental services for the operation of the whole instrument, which are called the `InstrumentManager`, `ReadoutManager`, and `TelescopeManager`. As the name suggests, the `InstrumentManager` controls all services located in the *Instrument Layer* of the software package, providing the necessary information to each service when they need it and updating them in a synchronized way. New instrument setups are only committed to the relevant services when it is allowed to do so. i.e., when no exposures are carried out. This avoids the loss of precious observing time due to accidental setup changes and ensures secure operation of LUCIFER, which is especially important when it is used by observers that are inexperienced or unfamiliar with the instrument.

The `InstrumentManager` is supported by the `ReadoutManager` and the `TelescopeManager`, controlling the `RMIGEIRSService` and the `Telescope` service of the *Control Layer*, respectively. The use of additional management services in the *Operation Layer* which control just one service of a lower layer is necessary to allow coordinated setup changes of the telescope and the detector readout, thus allowing fully automated observations with the instrument.

Using the automatic observation capability of the software will be the main operation mode for the instrument since it provides the highest efficiency for observations in the NIR wavelength regime (see Section 3.2). However, the instrument can also be controlled manually using the same software services if this is desired.

All graphical user interfaces (GUIs) written for the software package also belong to the *Operation Layer*. These include the Observation Preparation Tool (OPT), written by Schimmelmann (2007). The other GUIs to mention here are the interfaces that control the three different manager services, as well as the additional engineer interfaces. Hidden in these interfaces is the mechanism how different users, i.e., engineers and observers, access the software. These principles together with a description of the important services of the *Operation Layer* are worked out in Section 2.4.

2.2 Control Layer

The *Control Layer* of the software package is the only interaction point between the software for the LUCIFER instrument and the hardware that controls and/or monitors it. Therefore all services in higher layers rely on the services of this layer, while these services themselves rely on the services provided by the *System Layer*. This includes configuration management and message storage and delivery for example. The services of the *Control Layer* can be separated into three different types: Services that control the instrument hardware (motors and switches), services that control

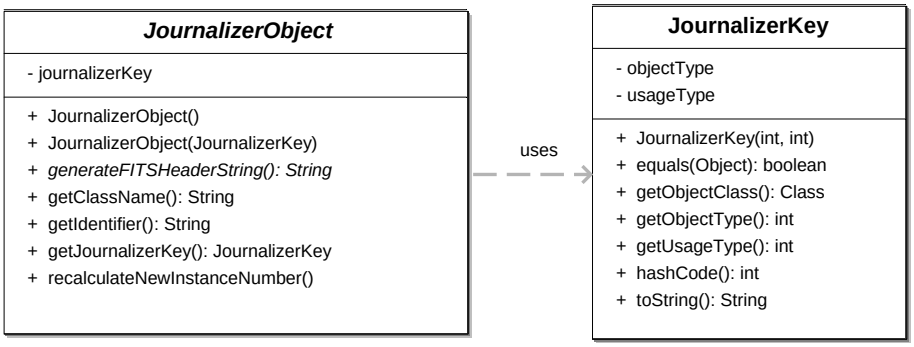


Figure 2.3: `JournalizerObject` and `JournalizerKey` class diagrams. For the `JournalizerKey` diagram the two important fields are shown.

environment measurement hardware (mainly temperature and pressure measurements) and services that control external software. Two services exist of the latter type, namely the `Telescope` and the `RMIGEIRSService`. These services controlling the external software packages are described at the end of this section. One service of this layer does not fit into the three categories that were mentioned above. This is the `Journalizer` service. A description of this service follows in the next section.

2.2.1 Journalizer

During the development of the software for the LUCIFER instrument it became clear that a central service, where all information about the instrument itself as well as other environmental data is stored, would be very beneficial. Since the LUCIFER control software package is realized as a distributed system of independent services, all of which could store the data they gather while they run on their own, it would be very laborious to query the current status of the instrument from the whole software system. Therefore the decision was made to create one service, which keeps a kind of *journal* about the status of the instrument. Subsequently, this service was called `Journalizer`. The service uses the *Hibernate* package[4] to store the data. This package allows the storage of any *JAVA* objects in relational databases. It does this using mapping files describing which fields of the object should be stored. The mapping files also specify the data type of these fields and table in the database where the objects should be stored. All *JAVA* objects that are stored by the `Journalizer` extend the abstract class `JournalizerObject` which is shown in Fig. 2.3. Because of this, the complexity of the service could be kept very simple as can be seen in Fig. 2.4 which shows the class diagram of the service interface. As a trade-off a corresponding mapping file has to be created for every `JournalizerObject`.

For the main functionality of the `Journalizer` service three methods are important: the `store(JournalizerObject)`, `get(JournalizerKey)` and `get(JournalizerKey, long)` methods. The first method is used to store a `JournalizerObject` to the database and the second method is used to retrieve the last `JournalizerObject` from the database. Since the `Journalizer` service is implemented in such a way that it keeps the last copy of any object it receives from the services of the software system in memory, the `get(JournalizerKey)` method does not query the database if the object was stored before. Instead the local copy is returned. This reduces queries to the database and improves the performance. Only if the `Journalizer` service

[4]for a detailed description about this package see `www.hibernate.org`

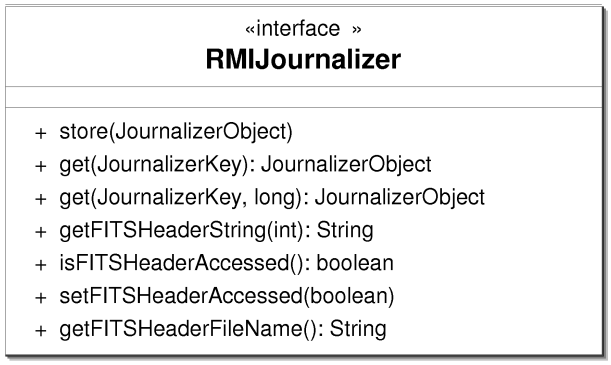


Figure 2.4: `Journalizer` service class diagram.

was recently started and a matching object is not stored in local memory the last object is retrieved from the database and returned to the calling service. The third method, `get(JournalizerKey, long)`, can be used to get a `JournalizerObject` as it was stored at a given time-stamp, which is passed as the second argument to the method. If there is no matching object for the exact time-stamp, the last object stored *before* this time-stamp is returned. This way, any object can be retrieved from the database and any instrument configuration at a given time can be reconstructed from the database in principle, assuming that the system runs normally and all services use the `Journalizer`.

The `JournalizerClient` which is used by all services to access the `Journalizer`, has a build-in queue for temporary storage of data that should be sent to the `Journalizer`. This is similar to the `MessageClient` of the `MessageService` package (see Polsterer in prep.). In the case that the `Journalizer` service is not available for the requesting client, new data that needs to be sent is first stored locally in memory and on disk. One possible scenario for such a situation might be the temporary loss of network connection. As soon as the `Journalizer` service is available again, the data that needs to be stored is submitted to the server. This provides that — under normal operating conditions — no data is lost.

As mentioned already, to store any kind of instrument data so–called `JournalizerObjects` are used. Figure 2.3 shows the class diagram of this abstract class. Since the class is abstract it cannot be instantiated itself, so subclasses of this class have to be created. These subclasses have fields that contain the relevant data for the service that uses them. To give typical examples, class diagrams of the `GratingUnitStatus` and the `ReadoutControlTemperatures` classes are shown in Fig. 2.5. In this figure, the left diagram shows the `JournalizerObject` subclass that is used by the `GratingUnit` service of the *Instrument Layer*, and the right diagram shows the `JournalizerObject` subclass that is used by the `TemperatureControl` service of the *Control Layer*. As can be seen in the two diagrams, each class holds different types of data: status information, i.e., position and grating tilt data and temperatures, respectively.

The `JournalizerObject` class uses so–called `JournalizerKeys` to distinguish between data from different services and determine where to store this data in the database. A `JournalizerKey` consists of two identifying numbers: the `UsageType`, which determines the *instrument* the data belongs to (i.e., LUCIFER 1 or LUCIFER 2) and the `ObjectType`, which characterizes the type of data stored in the `JournalizerObject` (temperature, pressure, etc.).

The Flexible Image Transport System (FITS) data format (Wells et al. 1981; Hanisch et al. 2001) is the standard file format used to store astronomical data. The header of FITS files contains

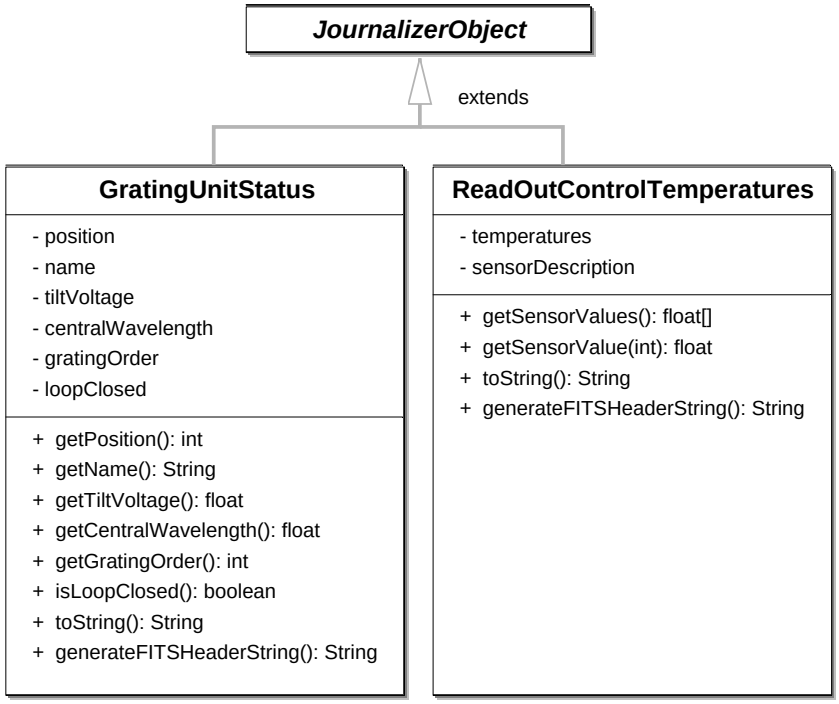


Figure 2.5: Examples for `JournalizerObjects`. The left diagram shows the status object stored for the *Grating Unit*, the right diagram the status object for the detector control temperatures.

information about the telescope and instrument setups at the time the data was taken, which is essential for the subsequent data reduction of these images. For creation of FITS header information the method `generateFITSHeaderString()` is used. To have the functionality of FITS header creation also encapsulated in the `Journalizer` service package is one additional benefit of the service and shows the strength of the concept. To create an "up to date" FITS header, one simply calls the `generateFITSHeaderString()` method of all stored `JournalizerObjects`. The `getFITSHeaderString(int)` method of the `Journalizer` service does exactly this. The argument required by the method determines the `UsageType`, which corresponds to the instrument (LUCIFER 1 or LUCIFER 2) the FITS header should be generated for. A typical FITS header of a LUCIFER image is shown in Appendix A.

Another benefit of the `Journalizer` service is that it allows a strict separation of observer interaction with the services from the *Instrument Layer*. Together with the manager services of the *Operation Layer* (see Section 2.4), the astronomer can interact with the software without the need to interact with the hardware directly. Although this might seem to be a big drawback at first, it introduces a very secure way to operate the instrument, eliminating the risk that the inexperienced or unfamiliar observer can do any harm to the instrument and greatly reducing the risk of lost observation time due to handling errors. The principles of user interaction with the instrument are laid out in detail in Section 2.4.1.

Summarizing the above, the usage of the `Journalizer` service allows:

- to store and retrieve the status of the two LUCIFER instruments at one single point of the software system,
- to rebuild an instrument state at any given time using the database, if everything runs

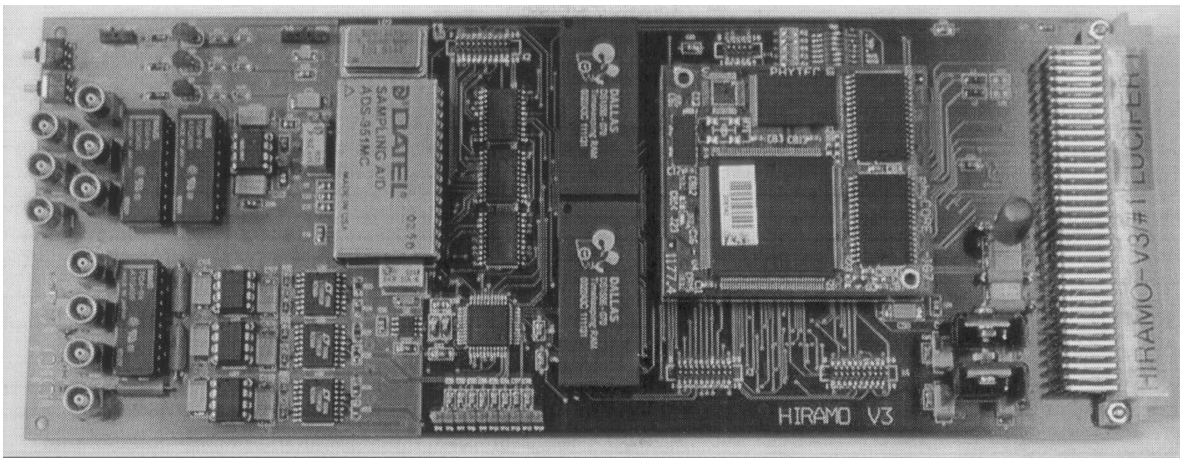


Figure 2.6: The *HIRAMO* electronic card (image taken from Lehmitz 2005).

correctly,

- easy generation of FITS header information, and finally
- the separation of observer interaction with the hardware of the instrument, introducing a high level of security during operation.

The work on the `Journalizer` package was split. The initial implementation and the functionality to create FITS header information was done during work for this thesis and the implementation to use the *Hibernate* package for data storage was done by Polsterer (in prep.).

2.2.2 Instrument and environment control

As noted earlier in this chapter, the main purpose of the *Control Layer* is the connection between the software and the hardware for the LUCIFER instruments. This section focuses on this point, presenting the instrument control services first, followed by a brief description of the environment control. Finally, the two services controlling the external software packages are introduced.

Instrument control

Instrument control can be split into two major categories:

- motor control and
- switch and other electronics control.

The motor control is realized by the `MCU` service. There are two instances of the `MCU` service running for each instrument: one service controls all instrument units excluding the *MOS Unit*, which is controlled by the second service (see Fig. 2.2). The switch and electronics control is realized by two separate service, called `HIRAMO` and `SwitchBox`. The `SwitchBox` service is responsible for all switches of the *MOS Unit* (see Polsterer in prep.). The `HIRAMO` service is discussed next.

Table 2.1: Switch value mapping to instrument unit positions for the *HIRAMO* card.

Bit Pos	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Instrument Unit	CC[a]		unused	CU[b]		FW[c]2				FW1				GU[d]		

[a]Cold Clamps for Filter Wheels
[b]Camera Unit
[c]Filter Wheel
[d]Grating Unit

HIRAMO The HIgh Resolution Analog Measurement Output board (Fig. 2.6) is a special purpose electronic card, mainly built to measure and regulate voltages with a very high resolution of 16 Bit (Lehmitz 2005). The voltage control and regulation functions of the board are used to apply a constant and precise tilt voltage to one of the three grating mounts of the *Grating Unit*. Three different input and output channels are used to control the grating tilt. One additional channel is used for reference voltage measurements. The card also has the capability to test the connection of the four different motor groups used for the LUCIFER instruments (excluding the *MOS Unit*). Furthermore, all switches for position measurements of four different units of the instrument are connected to the card and can be queried by a special software command. The switch value is coded as a 16 Bit integer that needs to be resolved to get the position of the corresponding unit. Table 2.1 shows the bit position mapping of the 16 Bit value. As can be seen from the table, the position of each unit itself is coded in a binary number, so the decimal position value has to be converted from this binary value. To request a position value from the `HIRAMO` service two methods must be called. The first one, `resolveSwitchValueBitPattern(int)` is used to get the correct bit pattern consisting of the starting bit and the bit length of the unit for which the position should be resolved. The unit identifier is passed as the argument. This pattern is subsequently used in the `getSwitchValue(SwitchValueBitPattern,boolean)` method, which then returns the decoded position value. The second parameter of this method determines if the switch value should be actively read from the hardware or if the buffered value should be used. This can safe traffic on the network and reduce the load of the electronics, especially if a lot of queries arrive in a short time period. Using two separate methods to query the position of the unit leaves the flexibility to quickly adopt the software should the electronics for LUCIFER 2 change. In this case only a minor part of the `HIRAMO` service has to be changed and these changes will be transparent for higher layer services. The class diagram of the `HIRAMO` service is shown in Fig. 2.7. Of all methods provided, only a few need to be examined in more detail, since these methods are relevant later on (see Section 2.3.2). All other methods are mainly for engineering purposes, i.e., to control the electronic card or the wiring of motors and will not be discussed here, since this would be beyond the scope of the description of the software.

The first relevant method is the `setGratingUnitTargetVoltage(float)` method. As the method name already suggests, the desired target voltage for the grating tilt regulation loop can be set. However, the grating tilt voltage is not yet applied by the electronics.To do this, the voltage regulation loop of the electronic has to be activated by calling the `activateLoop()` command. Then the desired voltage u_t is set by the electronics, which subsequently tilts the grating mount to an offset angle. The allowed values for the tilt voltage are in the range of $-5\,V < u_t < +5\,V$. The `setGratingUnitTargetVoltage(float)` method takes care that only sensible values can be set

Figure 2.7: `HIRAMO` service interface class diagram.

and throws an `Exception` otherwise. To keep the tilt angle — and thus the central wavelength on the detector — of the grating mount fixed, the applied voltage is kept constant by the regulation loop electronics as long as it is active. If the electronics work properly, the current applied voltage and the target voltage, which can be queried by calling `getGratingUnitTargetVoltage()`, should be identical. This can be tested using the `isTiltStable()` method. The regulation loop can be opened using the `deactivateLoop()` method and the current status of the loop can be queried with `getLoopStatus(boolean)`. Finally, to get the currently applied voltage, the `getGratingUnitCurrentVoltage()` method can be used.

Note that at this layer of the software, the `HIRAMO` service does not know anything about the possible central wavelength on the detector or the relation between tilt angle and tilt voltage, although this is of central importance for the spectroscopic operation of the LUCIFER instrument. The service does not need to care about these details because the higher layer services, the `GratingUnit` (cf. Section 2.3.2) and `InstrumentManager` (cf. Section 2.4.3), to be more specific, will handle these problems.

To conclude this section, Fig. 2.8 shows the graphical user interface that is used by engineers to control and manage the `HIRAMO` service. All relevant functions of the service are displayed in the user interface and can be accessed and modified. In the upper left part of the window, the switch value is displayed with all 16 bit positions. This display allows to easily identify if one of the switches is not operational, since the value for this bit would never change under these circumstances. For ease of use the decoded position values of each unit are shown next to this display. The lower part of the user interface controls the tilt regulation function of the service.

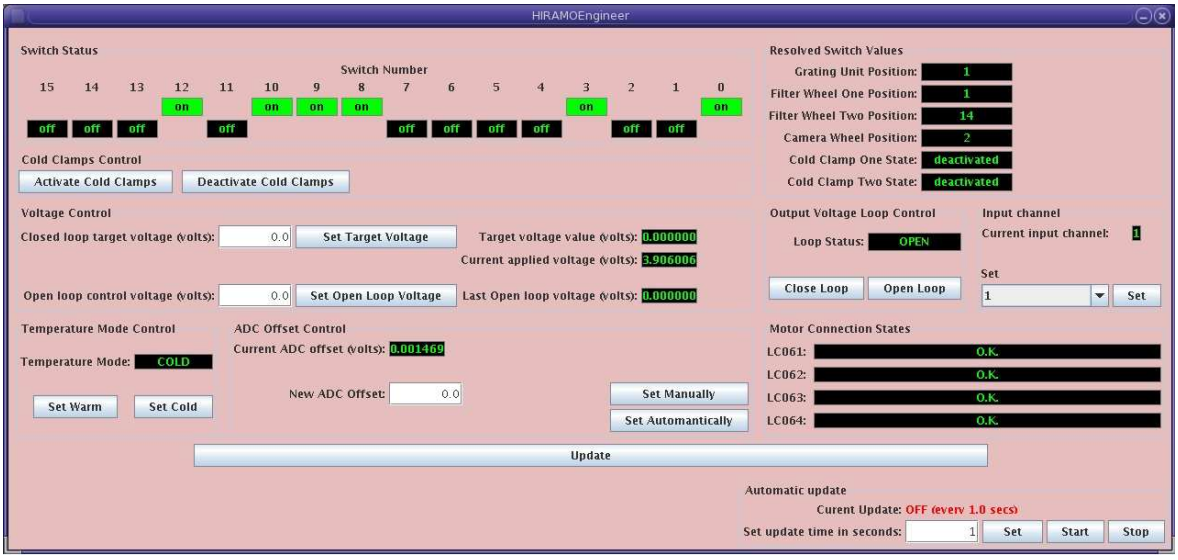


Figure 2.8: The graphical user interface for engineering access to the HIRAMO service.

Different tilt voltages can be set and read as well as input channels selected. The HIRAMO card can be configured for warm environments, i.e., when the instrument is at room temperature, or for cold environments, when the instrument is cooled to the operation temperature of 70 K. The different modes affect the resistance of the control loop electronics, to create the same conditions for both environments. Details are presented in Lehmitz (2005). Finally, the correct cabling of hardware to the card is displayed in the lower right part of the screen.

At this level of the software, no observer interaction is necessary or desirable, so no observer type user interface will be provided for the HIRAMO service.

Environment control

The LUCIFER control software package provides different services for environment control. Three separate services are responsible for temperature measurement.

1. The TemperatureMonitor service measures eight temperatures at different locations inside the instrument.

2. The TemperatureControl service does this for the detector and shield temperatures and

3. the temperatures of the electronic rack, where the control electronics for the instrument are located, are measured by the RackCoolingControl service.

Pressure measurement is realized by the PressureMonitor service. All services automatically log the measured values using the Journalizer service (see Section 2.2.1). Logging of data can also be disabled if storage of data is not necessary for the given instrument configuration or situation, i.e., when the instrument is warm and in the lab for maintenance. All logging services of the *Control Layer* have the ability to use static logging intervals (e.g., every 60 seconds), or use adaptive logging instead. When adaptive logging is used, the time interval is changed depending on how variable the measured data is. The interval stays between a minimum and maximum time, which can be specified in the service configuration. If the changes in data values are greater than

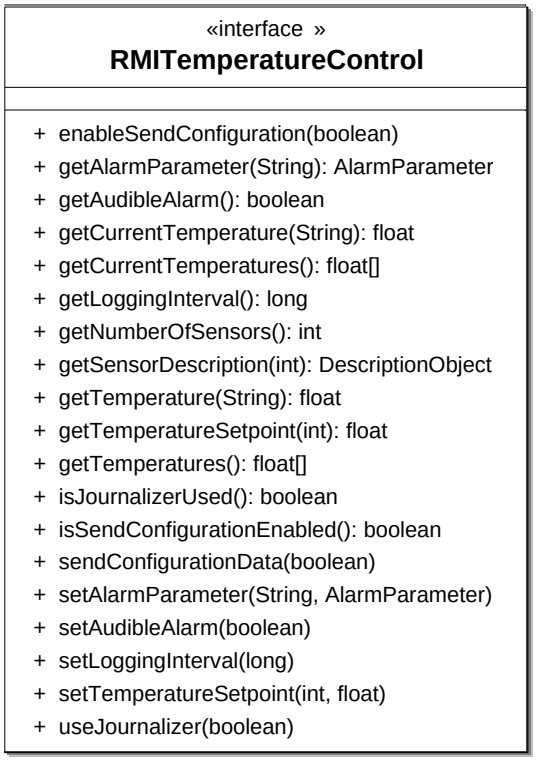


Figure 2.9: The class diagram of the `TemperatureControl` service as an example for environment control services.

a defined difference, the current interval is halved until the minimum logging time is reached. If the data stays below the defined difference for a repeated number of times, the current logging interval is doubled until the maximum time is reached. This way, changes in the data can be recorded with great precision, while constant data does not fill up the database quickly.

Temperature Control The detector temperatures are monitored and controlled by the `TemperatureControl` service which connects to a *LakeShore 331S Temperature Controller*. Fig. 2.9 shows the class diagram for the `TemperatureControl` service as an example for all environment control services. There are different methods provided to get the logged temperatures. The `getTemperatures()` method returns the temperatures as they were measured during the last automatic logging cycle, and thus these may be a few minutes old. To read the actual temperatures from the monitor the `getCurrentTemperatures()` method has to be used. It invokes an active query for the temperatures on the monitor hardware. This distinction for getting the temperatures (or pressures for the pressure logging service) is used in all logging services. Normally, the current values do not have to be actively queried from the hardware, since significant changes are slower than typical logging intervals. Two temperatures are logged and stabilized by the hardware, the temperature of the detector itself and the cold bench temperature (Lehmitz 2004). The regulation loop setpoints of the hardware temperature controller can also be set using the service, although this is normally configured on the device directly.

Eight temperatures at different position inside the instrument are monitored by the `TemperatureMonitor` service connected to a *LakeShore 218S Temperature Monitor*. The tem-

peratures are measured at the structure (top, bottom) and different units of the instrument.

For the third temperature measurements service, the `RackCoolingControl` service, four temperatures of the instrument rack are monitored. This service is connected to a *Jumo Imago 500* temperature controller (Lehmitz 2004). Three temperatures are measured for different parts of the rack: the motor electronics, the readout electronics and the temperature inside the rack itself. These are compared to the ambient temperature, which is measured by the fourth and last input channel of the controller. The controller hardware has the ability to regulate the temperatures of the given input channels using different output channels that drive heating or cooling units. If the ambient temperature is between two given temperature limits the software service sets the regulation target temperatures for the first three channels to the ambient temperature (with a ΔT of 0.5 K to minimize traffic to the monitor). If the ambient temperature drops below the minimum temperature, the target temperature is set to the minimum temperature, i.e., the rack is heated. If the ambient temperature rises above the maximum temperature, the target temperature is set to this maximum temperature, i.e., the rack is cooled.

Pressure Control The `PressureMonitor` service is able to measure two pressures simultaneously using a *Pfeiffer Vacuum TGP 262*. Normally only one gauge is used at a time and the other one kept in reserve and for comparison measurements, in case the active gauge provides unrealistic values. The software service can also be used to control setpoints for two regulation loops, similar to the `TemperatureController` service (see above). However, for the pressure control, these setpoints are configured at the device directly, too. Finally, the `TurboPumpMonitor` service can be used to monitor the pump revolutions and the used power of the *Pfeiffer Vacuum TCM 1601* turbomolecular pump when it is used during cooling–down or warming–up of the instrument.

External software control

As already mentioned at the beginning of this chapter, the software package for the LUCIFER instruments has to rely on two external software packages not developed at AIRUB. These are the *GEIRS* package for the readout of the HgCdTe Astronomical Wide Area Infrared Imager-2 (HAWAII2) detector, developed at the Max-Planck-Institut für Astronomie, Heidelberg and the IIF extension of the TCS developed in Tucson, Arizona. To be able to communicate with these packages, *JAVA* RMI services have been created that provide necessary functionality to the distributed system.

The `RMIGEIRSService` has been developed in the course of a diploma thesis (Muhlack 2006). For completeness, the service interface is shown in Fig. 2.10. The service needed to be extended considerably during the commissioning of the instrument. The changes were necessary in order to allow the control software to work with the current version of the *GEIRS* package, which evolved significantly since the time the original work was finished. In addition, new functionality that became apparent during the commissioning had to be implemented. This mainly includes the insertion of the FITS header generated by the `Journalizer` service (cf. Section 2.2.1) into the images written by *GEIRS*. FITS header information is currently written to a temporary text file and stored on disk. This file is subsequently added to the FITS image by *GEIRS*. This procedure will change in the future, however, when a more convenient way is implemented in the `readout` package. The `RMIGEIRSService` communicates with the *GEIRS* server using a socket connection and acts itself as a server for *GEIRS* to receive status information. For more information about this bidirectional communication and a detailed description of the package see Muhlack (2006).

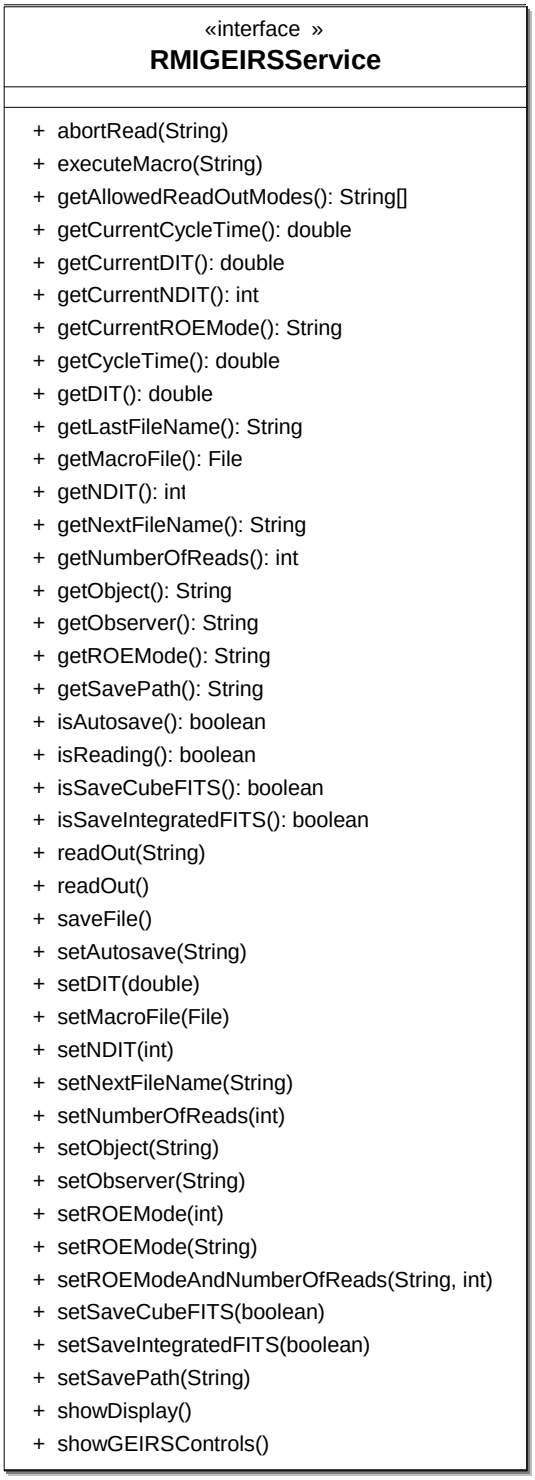


Figure 2.10: The class diagram of the `RMIGEIRSService` used to control the external *GEIRS* software package.

The `Telescope` service, responsible to provide commands to control the LBT has been implemented by Dr. M. Jütte using an ICE interface to the TCS. Since the work on this package is beyond the scope of this thesis, no further details are provided on this issue.

2.3 Instrument Layer

The next higher level above the *Control Layer* is the *Instrument Layer*. At this level, all physical units of the LUCIFER instrument are represented by dedicated software services. Each service provides the necessary functionality for the unit it represents and does not interact with other services of the *Instrument Layer*. Coordination of the different services of the *Instrument Layer* is done by the services of the highest layer of the software, the *Operation Layer* (cf. Section 2.4).

To write services that represent just one physical unit might look a bit narrow minded first, but it helps to keep the software complexity manageable. It means that only the control over, and tasks of one unit have to be provided. An additional benefit is that, should some part of the software or hardware of one unit fail for some reason, only that unit will be affected. Of course the problem will have to be fixed as soon as possible, but it might not affect the ongoing observations when the error occurred. For example, if the camera unit fails during the night but no other camera must be used, the instrument is still operational. If the software services would be more complex such an error could lead to the failure of the whole instrument in the worst case scenario.

This section begins with a description of the software concept used to model setup changes for the instrument. For the instrument units, this is the direct link to the services of the *Control Layer*. With the only exception of the service controlling the *MOS Unit*, which has been developed by Polsterer (in prep.), all services of the *Instrument Layer* (see Fig. 2.2) have been implemented in the scope of the present thesis. Using one of the most complex of all services, the `GratingUnit` service, as a first example, the fundamental concepts applied to all services of the *Instrument Layer* are examined in detail in Section 2.3.2. The section finishes with the `CompensationMirror` service, which is described as a second example. For this service to work properly, status information of different *Instrument Layer* services are necessary. This leads then to the highest level of the software, the *Operation Layer*, described in detail in Section 2.4.

2.3.1 Modeling instrument changes in software

At first it needs to be decided how changes of the instrument status can be modeled in a software solution. To do this, a detailed study was carried out of what happens when a unit (in this context understood as the real physical unit like, e.g., the *Grating Unit*) is moved inside the instrument.

First of all, the unit can be in an arbitrary position. This can be any position or *state*, the unit is physically able to move to or to be in. In the concept of the LUCIFER control software, this position is represented as a `State` object. The next step to model are then *changes* between these states or positions. The process of moving a unit from one `State` to another is described in the software by `Transition` objects. With these two objects, the software is already able to model instrument changes completely. However, to allow better structuring and error handling of the software, these two types were unified in one additional object which is called `Sequence`. A `Sequence` may consists of two `States` and one or more `Transitions`. The two `States` are called *precondition* and *postcondition*, the `Transitions` are the movements or actions necessary to move the unit from the precondition state to the postcondition state. Additionally, a `Sequence` may contain and

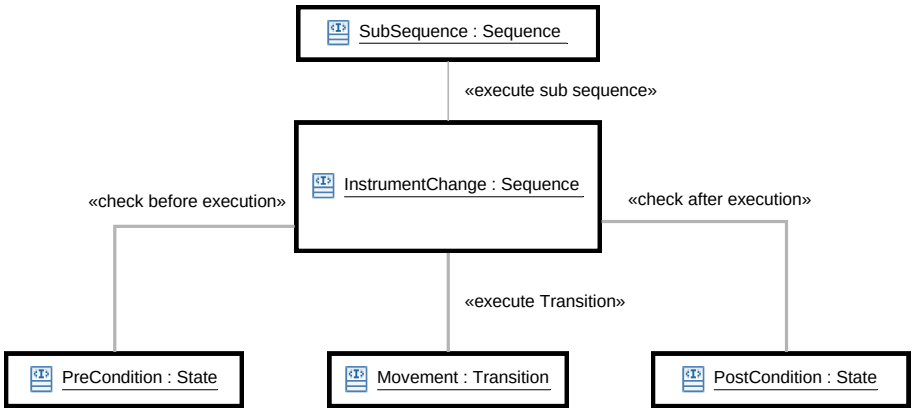


Figure 2.11: An object diagram showing the concept of modeling instrument changes using `Sequences`, `Transitions` and `States`.

execute other `Sequences`. This allows to group more complex instrument changes together in one entity and thus structures the available actions for one unit. Complicated movements of the units can be started in a safe way by executing only one sequence. Fig. 2.11 shows the abstract concept for an arbitrary instrument change. The `Sequence` to execute, named `InstrumentChange` in the figure, first checks the `PreCondition` state. If this check succeeds, the `Movement` transition is executed and finally the `PostCondition` state is evaluated. If this test also succeeds, the `Sequence` has been executed correctly. If the `InstrumentChange` sequence was to require the execution of another sub–sequence (as indicated in the figure), the checks for the sub–sequence would be performed in the same way as for the main–sequence. The precondition of the sub–sequence is checked before its execution and the postcondition is evaluated at the end. Then the execution of the main–sequence is resumed.

In the following a detailed description of all three abstract objects that are used to model changes inside the instrument is presented. The implementation of most of these abstract classes was done by Polsterer (in prep.) and is included here for completeness. The implementation of the software for the individual units of the instrument, including all `States`, `Transitions`, and `Sequences` that have been defined for them, is part of the present thesis.

States

In principle, one can distinguish two separate kinds of states:

1. the ones representing positions that can be queried by the control software using hardware switches or other kinds of measuring equipment (e.g. angle resolvers) and

2. all other states that can not be identified safely by the software.

The former of these are called *defined states*, the latter are consequently *undefined states*. To initialize a unit which is in an undefined state, it has to be moved to a defined state first. This can be done safely for nearly all units of the instrument with the exception of the *MOS Unit*. Once a defined state is reached, the unit is, or usually can be, initialized, again with the exception of the *MOS Unit*. See Polsterer (in prep.) for a detailed study about this subject.

The class diagram of the `State` interface is shown in the upper left part of Fig. 2.12. It contains only one method, which all `States` have to implement, the `isInState()` method, which

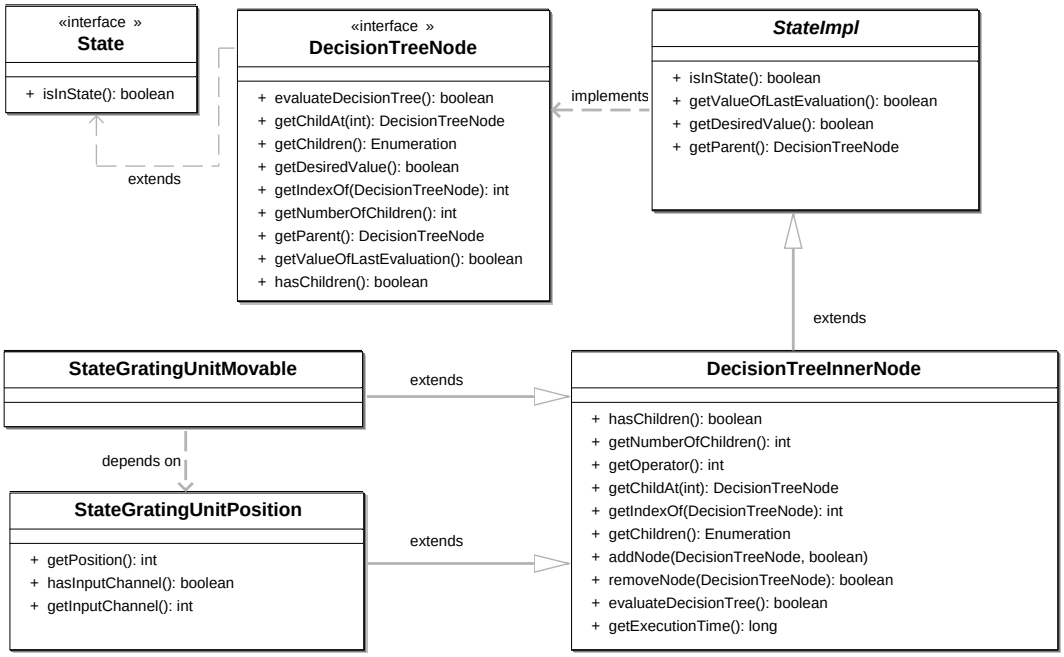


Figure 2.12: The complex dependencies of `State` objects. See text for details.

either returns `true` or `false`. To be able to create more complex states, each state can have any number of sub–states, organized in a so–called *decision tree* with nodes (which are defined as having one or more sub–states) and leaves (which do not have sub–states). A leave can simply be evaluated by checking the conditions defined for the `isInState()` method. For a more complex node, consisting of one or more sub–states, a call to the `isInState()` method has to evaluate the condition for all sub–states in the decision tree. This evaluation process is realized implicitly for all states that extend the `StateImpl` class shown in the upper right part of Fig. 2.12. The managing of the decision tree logic, like getting all sub–states, adding new nodes, etc., is defined in the `DecisionTreeNode` interface and implemented in the `DecisionTreeInnerNode` and `DecisionTreeLeaveNode` (the latter not shown in Fig. 2.12) classes.

The process of evaluating a complex state is best laid out using an example. The *Grating Unit*, which is presented in detail in Section 2.3.2, has a number of complex states, one of which is called `StateGratingUnitMovable`. This state is shown in the lower left part of Fig. 2.12. By extending the `DecisionTreeInnerNode` class it has all the functionality described above, of course. To use an object of this class, two parameters must be provided. The first information needed is how to query the current position of the *Grating Unit*. This is done by using a `Client` for the `HIRAMO` service (see Section 2.2.2), which is the direct interaction point between the *Instrument Layer* and the *Control Layer*. The second information needed for the state is an integer value, which determines the position of the unit that should be checked. This could also be expressed in a different way: the integer argument passed in the constructor call defines, which physical/logical state of the unit should be represented by this object in software. It can be seen, that the software concept of this layer does not deal with any hardware related details, i.e., which command syntax is used to read the current position from the electronics card, or where to find the address of the card in the instrument network. Instead, only the pure logic of the unit has to be, and is, modeled. This keeps the complexity of the software low and makes it more robust.

For the *Grating Unit* it was defined that the unit should be moved only, if no tilt voltage is applied in closed loop operation. This reduces stress on the regulation electronics and the hardware. To model the behavior in software, two conditions have to be checked. First, the position of the unit can be one of the grating positions, the mirror position, or the unit can be in an undefined state. Every `StateGratingUnitMovable` object therefore has one `StateGratingUnitPosition` object (shown in Fig. 2.12 in the lower left), representing one of the possible positions of the unit. To test if the unit is in the correct position, the `getPosition()` method of the `StateGratingUnitPosition` class is used. If the unit is not in a defined position, or not in the position it should be, this call will fail.

The `hasInputChannel()` method is used to test, if the position of the unit also represents one of the gratings positions. The name for the method is due to the fact that every grating has one specific input channel on the *HIRAMO* card associated to it. To determine if the grating unit is movable, the second condition has to be checked, namely if the regulation loop is currently closed or not. This is done by adding an additional sub–state to the decision tree. This state is called `StateHIRAMOLoopClosed` (not shown in Fig. 2.12). It is a leave state and can be evaluated directly, returning either `true` or `false`. Only if both tests succeed — the unit is in the correct position and the grating tilt loop is not closed — the `StateGratingUnitMovable` condition will be satisfied and the unit can be moved using a `Transition`.

Transitions

In the most general sense, a `Transition` is the action that is necessary to bring the instrument from one defined state to another. This action can be the movement of a motor, the selection of an input channel or the opening of the regulation loop, to give a few examples. However, between the execution of two transitions, an intermediate state does not have to be checked, if this is not needed. This might be the case because the state change of the first transition is a sub–state of the following one, and checking the final state is sufficient for a safe execution of a sequence (see next paragraph), for example.

Fig. 2.13 shows the dependencies for some transitions used in the software. Shown are all transitions that interact with the `HIRAMO` service (introduced in Section 2.2.2). The `Transition` interface shown at the top of the figure, defines the one method that all `Transitions` have in common: the `execute()` method.

To group all classes that interact with the *HIRAMO* card, the class `TransitionHIRAMO` (shown in the center of the diagram) has been defined. All transitions shown at the bottom of the diagrams are direct sub–classes of this `TransitionHIRAMO` class, which is indicated by the solid lines connecting all classes. The different vertical positions of these lower classes are chosen only to keep the size of the diagram small and do not represent different layers of inheritance. All classes have to define an implementation of the `execute()` method. This determines which actions have to be carried out by the specific `Transition`. For the example of a `TransitionOpenLoop`, this is the call of the `deactivateLoop()` method of the `HIRAMO` service (see Fig. 2.7). This call is forwarded using a `HiramoClient` object.

With the two types introduced so far, the `State` object and the `Transition` object, nearly everything is in place to work with the instrument from an abstract point of view. The last missing piece is described next.

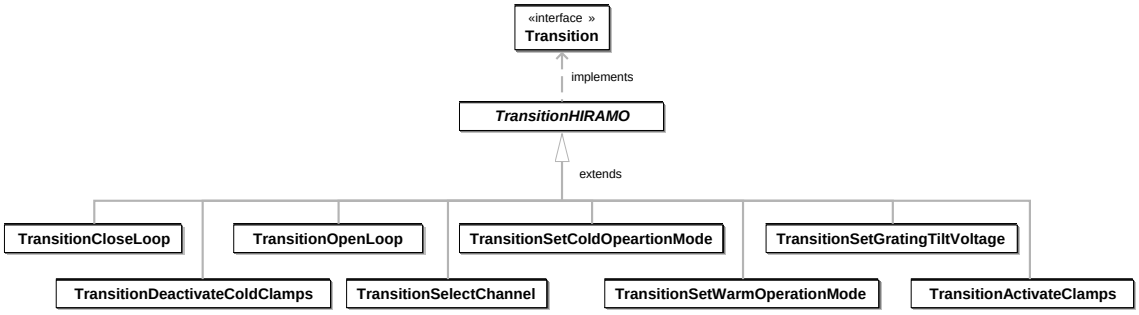


Figure 2.13: The dependencies between transitions shown for the example of HIRAMO service interactions.

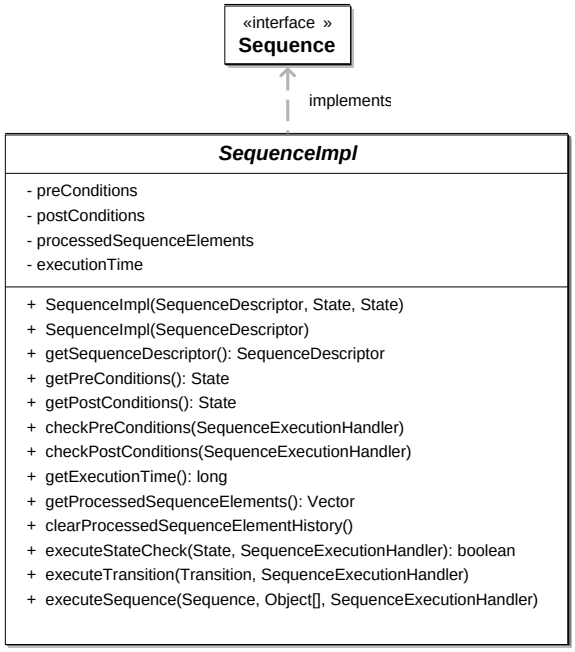


Figure 2.14: Class diagram of the SequenceImpl class which implements the Sequence interface and is the super class of all sequences.

Sequences

To structure the instrument changes that are modeled with States and Transitions, the Sequence object was introduced. By grouping the complex motions and transitions between states of the instrument together in one class, an easy-to-understand and easy-to-use mechanism has been created to work with the instrument. The class diagram of the SequenceImpl class is shown in Fig. 2.14. Two constructor methods are available to create a new Sequence object. Both constructor methods need a so–called SequenceDescriptor used to describe what the sequence is supposed to do. Additionally, pre– and postconditions can be set for the sequence using the second constructor. These conditions are State objects, as described above. The getPreConditions() and getPostCondition() methods can be used to return these State objects. The execution of sequences is normally started by calling the exceuteSequence(Sequence, Object[], SequenceExecutionHandler) method. The first parameter of this method defines the sequence

Figure 2.15: Class diagram of the `SequenceExecutionHandler` class.

to be executed. The second parameter is an array of objects, which can be used to control the execution of the sequence by specifying further constraints or setting values for pre– and postconditions, for example. The last parameter is `SequenceExecutionHandler` that should be used to manage the execution. The class diagram is shown in Fig. 2.15. Methods are provided to start and stop the execution of sequences and transitions and enabling or disabling pre– and postcondition checking. The functionality of this class is primarily needed by the *MOS Unit* with its complicated structure of sequences (for implementation details see Polsterer in prep.).

If pre– and postcondition checking is switched on and a precondition has been defined for the sequence, the handler will start to check this precondition. If it fails, an exception will be thrown. If the precondition check is passed, the handler will start to execute the actions, i.e., transitions or sub–sequences that have been defined. If sub–sequences are executed in between, their pre– and postconditions, if they are present, are checked before and after the execution of the corresponding sub–sequence. If only `Transitions` are defined, the post condition check is performed after successful completion of all of them. If an error occurs, an exception is thrown and the execution is halted. The corresponding unit maybe in an undefined state after such an error occurs and must be initialized again.

After successful execution of the sequence the unit is in the postcondition state and ready to execute another sequence, if needed. One example for this would be the movement of the *Grating Unit* from the mirror position to one of the grating positions. After this movement successfully finishes, the sequence to tilt the grating to a specific central wavelength on the detector can be executed. The unit and the corresponding `GratingUnit` software service is discussed in detail in the next section.

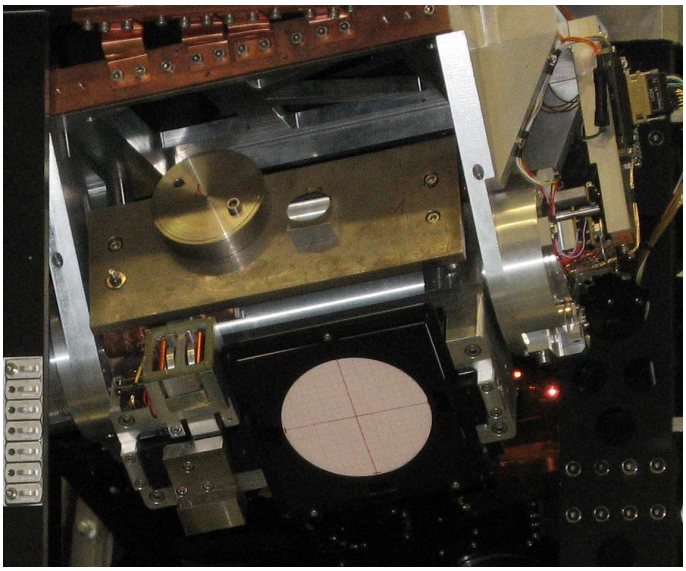


Figure 2.16: Photo of the *Grating Unit* mounted with a graticule and dummy weights in the lab in Heidelberg. The photo was kindly provided by M. Lehmitz.

2.3.2 Example: The `GratingUnit` service

The first service that is used to describe the main concepts of the *Instrument Layer* is the `GratingUnit` service, because it defines complex `Sequences` which are good examples for the implementation of this concept. As a second benefit, this service also uses `LookupTables`, which are common throughout this layer to store data that is fundamental for the operation of the instrument. First, the physical unit is presented in a short summary adapted from Seltmann et al. (2004), followed by the description of the software service.

The physical unit

Using the optical beam as a reference, the *Grating Unit* is located between the *Compensation Mirror*, which is described in detail in Section 2.3.3, and the *Camera Unit* with the three different cameras. Figs. 1.4 and 1.5 show the optical path of the instrument with the mirror position selected for the *Grating Unit*.

The *Grating Unit* can hold four optical components: one mirror and up to three gratings. Currently one high dispersion grating (Milton Ray: MR_35_53_*_877) can be used. The grating has 210 lines per mm and a blaze angle of 31.7° in first order. It is used in different orders (2^{nd} to 5^{th}) to cover the full wavelength range from 0.8 μm to 2.5 μm. Seifert (2002) gives a detailed description of it. Two other gratings are currently tested and will be added to the instrument in the near future. The position of the unit is measured using micro switches. Each position is also defined by a notch into which a ball bearing locks. This allows to switch off the motors when the unit is not moved. The three grating mounts are tilted using a moving magnet linear motor, consisting of two polarized permanent magnets, surrounded by two drive coils. A magneto-resistive element is used for measuring the tilt position (Seltmann et al. 2004). Fig. 2.16 shows a photo of the unit taken during the lab tests at the MPIA. One grating has a graticule mounted for optical alignment, the other unit positions are equipped with dummy weights for balancing.

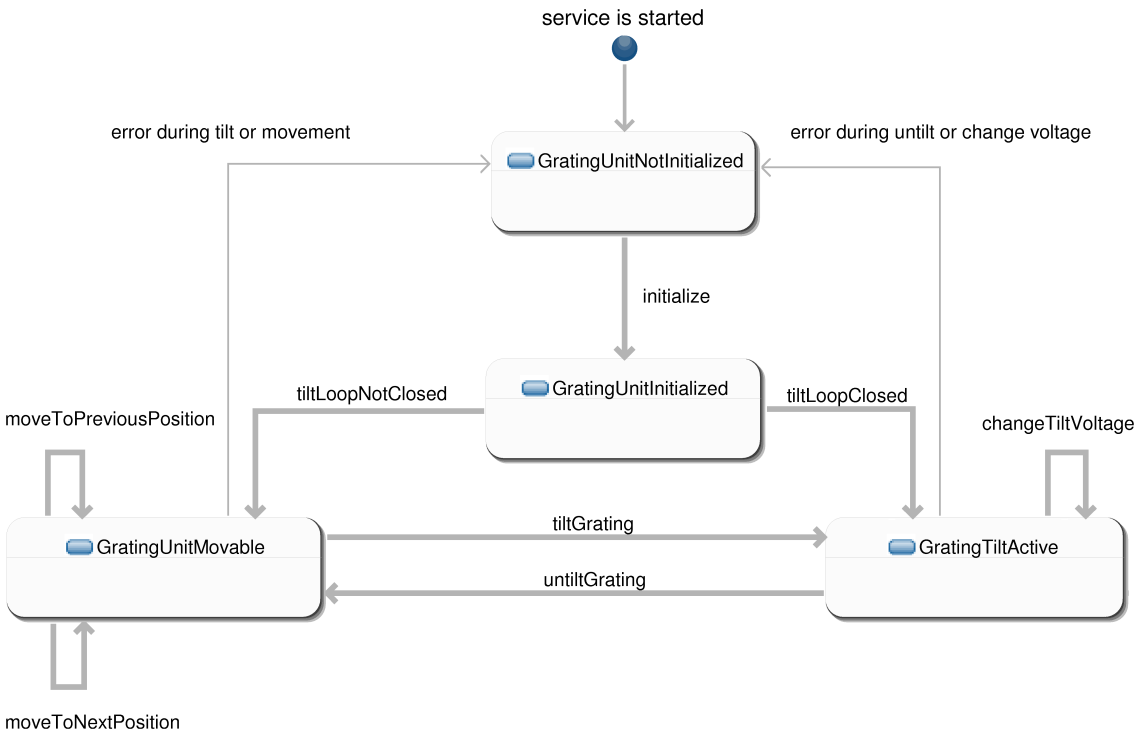


Figure 2.17: State diagram of the finite state machine for the `GratingUnit`. Normal transitions between states are marked with thick arrows, transitions due to errors are indicated by thin arrows.

The software service

As pointed out above, the software service uses a number of complex `Sequences` to operate the unit. Using this concept allows to model the physical unit like a *finite state machine* for which the possible states are shown in Fig. 2.17.

When the service initially starts, the machine is in the `GratingUnitNotInitialized` state. This is the default starting point for all *Instrument Layer* services. No active query for the current state of a unit is made automatically, since this might lead to unforeseen problems during the operation of the instrument or can even be dangerous during lab testing. To explain this reasoning, consider the following examples:

- The switch value decoding for the grating unit position is not working properly because of some hardware failure. This might lead the software service to assume that the unit is not initialized, even though it is operational, e.g., because the unit has been moved by an engineer to a defined position using the motor controls directly. It should stay in this position until the instrument can be maintained properly. The `GratingUnit` service is stopped by the engineer to avoid accidental movement of the unit. If the service would try to automatically initialize the unit when it is started again this would fail, leaving the instrument in an undefined state again because the motor would have moved.

- Even worse would be the scenario of a unit moving automatically because someone starts the software in the lab while the instrument is open and an engineer is working with his hands inside the instrument.

An initialization request must therefore be made explicitly. If the initialization succeeds, the unit is in one of the GratingUnitInitialized sub–states, depending on the conditions that are present when the unit is initialized. The first of these sub–states is the GratingUnitMovable state (see Fig. 2.12). This state represents the unit in one of the defined positions with the grating tilt loop open. If the position of the unit cannot be queried from the electronics, i.e., it is not locked in one of the notches, the GratingUnitMovable state is normally reached after a successful initialization, because the initialize() command tries to move the unit to the nearest defined position. Before this movement starts, the grating tilt loop is opened, regardless of the state of the tilt before. Here, it is assumed that if the current position of the unit cannot be determined from the hardware switches, no sensible position of the unit can be present, so opening the loop is no problem. If the position cannot be read because of some hardware failure (as in the first example mentioned above) but is in fact operational, there is no way for the software to determine this. When the unit is successfully moved to the next position, the unit is initialized in the GratingUnitMovable state of the finite state machine.

The second sub–state of the GratingUnitInitialized state is the GratingTiltActive state. After initialization this state can be reached, if the unit is in one of the defined positions *and* the grating tilt loop is closed. This scenario can be possible, when the software service was stopped, either on purpose or because of some other internal failure, and restarted at a later time. This is of course only possible if no interaction that changed the state has occurred in between with the physical unit.

If an error occurs during initialization of the unit, the finite state machine returns to the GratingUnitNotInitialized state. In this case, an engineer can try to interact with the unit using one of the services of the *Control Layer* services, namely the MCU or the HIRAMO service. This way it might be possible to set up the unit in such a way that further observations with LUCIFER are still possible. However, no higher layer functions are likely to work properly under these circumstances.

Depending on the state that is reached after initialization, different transitions are possible. If the unit is in the GratingUnitMovable state, it can be moved to the next or the previous position. If this transition succeeds, the unit is again in the GratingUnitMovable state. Alternatively, from the GratingUnitMovable state, the GratingTiltActive state can be reached via the tiltGrating transition. If an error occurs during any of these transitions, the resulting state is the GratingUnitNotInitialized state.

From the GratingTiltActive state, two transitions are possible, the changeTiltVoltage transition, which results in a greater or smaller tilt angle of the grating and keeps the unit in the same state, or the untiltGrating transition, bringing the unit to the GratingUnitMovable state. When the unit is in the GratingTiltActive state, it is locked there, so no accidental position changes are possible. Instead, before a movement of the unit can take place, the finite state machine must be in the GratingUnitMovable state, using the transition mentioned above. If any of the possible transition fails, the unit returns to the GratingUnitNotInitialized state.

The following example of a spectroscopic observation explains the normal operation of the *Grating Unit* using the finite state machine introduced above. The GratingUnit service is started and the unit is not initialized. An engineer, or the InstrumentManager service (see Section 2.4.3), initializes the unit. After the initialization routine, the unit is in the GratingUnitMovable state representing the mirror position. In this position an acquisition image is taken and the observer is satisfied with the on–sky positioning of the instrument. The desired long slit mask is inserted into the focal plane (using the MOSUnit service) and another image is taken to verify the slit positioning. Again the observer is satisfied with the result. To obtain the first spectrum of the object the *Grat-*

ing Unit must be moved to the high dispersion grating position. The finite state machine moves from the GratingUnitMovable state representing the mirror via the moveToPreviousPosition transition to the GratingUnitMovable state, representing the high dispersion grating position. Then the desired central wavelength is selected by tilting the grating with an appropriate tilt voltage, taken from a LookupTable. The finite state machine moves from the GratingUnitMovable state to the GratingTiltActive state via the tiltGrating transition. Once the state is reached the exposure of the spectrum can begin. When all necessary exposures are taken and another acquisition image should be taken, the finite state machine moves from the GratingTiltActive state to the GratingUnitMovable state via the untiltGrating transition and afterwards via the moveToNextPosition transition back to the GratingUnitMovable state representing the mirror position. Now the process just described can begin again.

Lookup Tables Another important function the GratingUnit service has to provide is the mapping of tilt voltages to central wavelengths on the detector. To perform this task, the service uses different LookupTables, one for each grating, that are loaded when the service is started and can also be changed while the service runs. Fig. 2.18 shows the class diagram for the most important classes of the lookupTable package. The abstract class LookupTable, shown in the upper left part of the figure is the super class for all lookup tables that are used by the software. The tables consist of LookupTableElements, which are shown in the lower left part of the diagram. The LookupTable class provides basic methods necessary to organize the table. This includes methods to add elements to the table, edit elements and methods to get elements or remove elements from the table. The latter two methods are overloaded, to allow to get (or remove) an element by specifying the position of the element in the table, or by specifying one particular element directly using its unique key.

Each LookupTableElement has to be unique and can only be added once to a table. This limitation is necessary to avoid errors due to duplicate entries in the table. If two indistinguishable elements were present with different data entries, it cannot be determined which of these elements will be returned when one of these elements is retrieved from the table and this might lead to unpredictable result when the software runs. To provide the possibility to distinguish different elements, a unique key has to be specified for each element. This key usually is one of the entries of the LookupTableElement, but can also consists of more entries or a value computed from multiple entries of the element. The implemented sub–classes of the LookupTableElement class provide this method in an appropriate way. The key value can be queried by the getKey() method and to test if an entry represents a key value the isKey() method can be used. Since by this definition two elements are considered equal when their keys are equal, the equals() and hashCode() methods, which are used by many *JAVA* classes to test for equality of objects, are overridden by the class LookupTableElement to provide the correct behavior.

The classes introduced so far are able to provide lookup tables that can be used to query values stored in the tables. This is sufficient for the FilterUnit service for example, which uses these tables to map filter names to filter positions and thus allows an easy selection of the correct filter. However, for many other *Instrument Layer* services, this functionality is not sufficient, because not all values that are required during the operation of the service can be stored in the table or it is just not sensible to do this, because it would require too much effort. To accommodate for this application, the LookupTable class is extended by the InterpolatingLookupTable class, which itself is extended by the TwoDInterpolatingLookupTable (see the right part of Fig. 2.18). These two classes provide a natural enhancement to the LookupTable class, by allowing the selection of

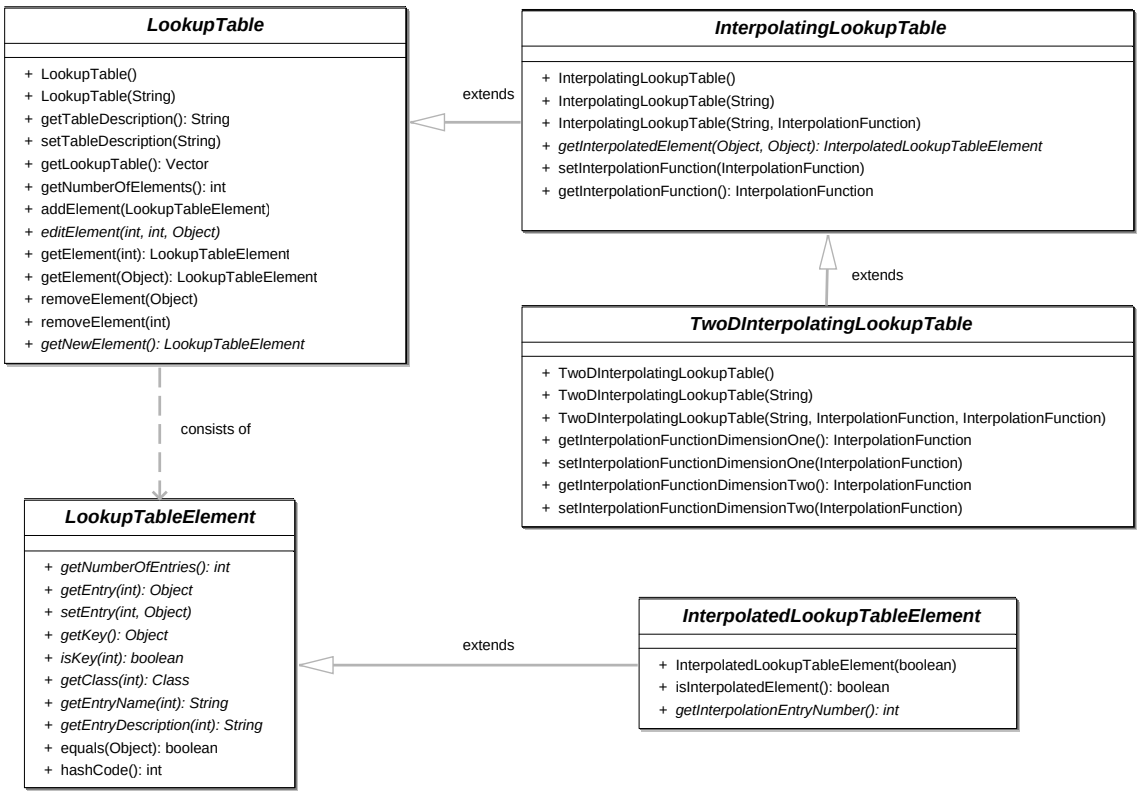


Figure 2.18: Class diagram of the main classes of the `LookupTable` package. Shown are the two basic abstract classes `LookupTable` and `LookupTableElement` on the left side as well as the extending classes for interpolated data on the right side of the diagram.

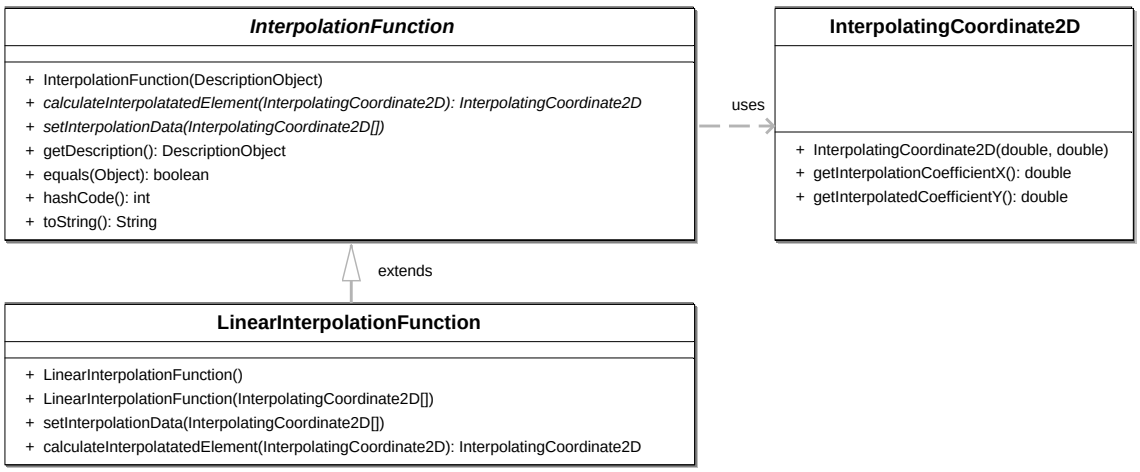


Figure 2.19: Class diagram showing the abstract `InterpolationFunction` class and one currently selectable implementation. The `InterpolatingCoordinate2D` class represents the type of objects that are used to interpolate data values.

InterpolationFunctions for one or two dimensions, respectively. It is important to note that the TwoDInterpolatingLookupTable uses *two* separate interpolation functions each providing a *one dimensional* interpolation for the selected dimension, instead of using *one* interpolation function for *two dimensional* interpolation. This allows to select different interpolation routines for each dimension and thus reflects the situation where the dependencies of interpolation values are likely to differ for separate dimensions. This is important for the CompensationMirror service, which is described in detail in Section 2.3.3.

For InterpolatingLookupTables, the getInterpolatedElement(Object,Object) method can be used to interpolate a given value from the table data. This value is specified as the first parameter of the method. The second parameter allows to specify an additional filter before the interpolation is started. The method returns an InterpolatedLookupTableElement object, whose class diagram is shown in the lower right section of Fig. 2.18. This class extends the LookupTableElement class by providing one additional field to distinguish interpolated values from "real" data that can be queried using the isInterpolatedElement() method. The getInterplolationEntryNumber(int) method may be used to test which of the entries of an element can be interpolated. For instance, it is possible to interpolate a tilt voltage from a given central wavelength, but not vice versa, since this is no requirement for the foreseen application of the software. Inverting the used InterpolationFunction may not be possible in general anyway, in which case reversing the lookup "direction" would also not be possible.

Fig. 2.19 shows the class diagram of the InterpolationFunction. Objects of this class are used by the lookup tables to calculate data values. As a super–class it defines the basic methods that all InterpolationFunctions provide to the tables using them. The computation is done with InterpolatingCoordinate2D objects that represent simple two dimensional coordinates with x and y values. It must be noted that the coordinates that are used to interpolate the data are not automatically taken from all of the values that are stored in the LookupTable. The coordinates must be set using the setInterpolationData(InterpolatingCoordinate2D[]) method explicitly. Of course the coordinates can be updated prior to any calculation. After a set of (x[n],y[n]) tuples has been set, for a given value x the corresponding y can be computed with getInterpolatedValue(InterpolatingCoordinate2D). If the calculation cannot be performed, i.e., because the interpolation data set is not sufficient, an exception is thrown describing the reason for the failure. The explicit selection of values allows to use only a subset of data pairs from the table for the interpolation of new data.

One implementation for an InterpolationFunction is shown in Fig. 2.19. It is called LinearInterpolationFunction and provides the functionality to interpolate any point (x,y) between two enclosing border points (x_1, y_1) and (x_2, y_2) using the formula:

$$y = \frac{y_1 - y_2}{x_1 - x_2} \times (x - x_1) + y_1. \tag{2.1}$$

Note that this is not the same as a using a linear (first order) polynomial to interpolate the data. In the latter case it would also be possible to extrapolate values beyond the "boundaries" of the table, whereas the given implementation always needs two boundary points for the interpolation. The gradient of the interpolation function will also change between neighboring intervals, which would not be the case for a linear polynomial fit. The lack of the ability to extrapolate data points beyond the boundaries is not a drawback in general. Instead, for some services, it can even be dangerous to extrapolate values. For the GratingUnit service, the tilt voltages that are computed from the table must always remain between some fixed boundaries ($\pm 5\,\mathrm{V}$) to avoid damage to the hardware. By specifying dedicated boundary values in the lookup tables, an inherent safety is

Figure 2.20: Class diagram of the `GratingUnit` service.

provided that only valid voltages are sent to the hardware. This introduces an additional level of security for the operation of the LUCIFER instrument.

The implementation of more complex interpolation functions (i.e., polynomials or splines) that also allow extrapolation of data values is very simple and straightforward using the framework that is in place already. The needed interpolation algorithms have to be written into the `calculateInterpolatedElement(InterpolatingCoordinate2d)` method of the corresponding `InterpolationFunction` sub–class and the new interpolation type can be used.

Implementation details for the `GratingUnit` service The class diagram of the `GratingUnit` service is shown in Fig. 2.20. Similar to the environment control services presented in Section 2.2.2, different methods are provided to get the position of the unit and the tilt voltages. Two methods are passive, i.e., return the buffered values, namely `getPosition()` and `getTiltVoltage()`, whereas `getCurrentPosition()` and `getCurrentTiltVoltage()` are actively sending a query to the underlying `HIRAMO` service of the *Control Layer*. Whenever one of the latter methods is called, the `Journalizer` (see Section 2.2.1) is updated as well.

The `GratingUnit` service uses up to three different lookup tables, one for each grating, that can be used to compute tilt voltages from central wavelength values. The tables can be loaded using the `setLookupTable(int, LookupTable)` method, with the first parameter specifying the grating for which the table should be used. The values that are stored in the tables can be seen in Fig. 2.21, showing the class diagrams for the `GratingTiltLookupTable` and `GratingTiltLookupTableElement` classes. Each element of the table consist of four entries: central wavelength, tilt voltage, grating order and tilt angle. The tilt angle is currently not used

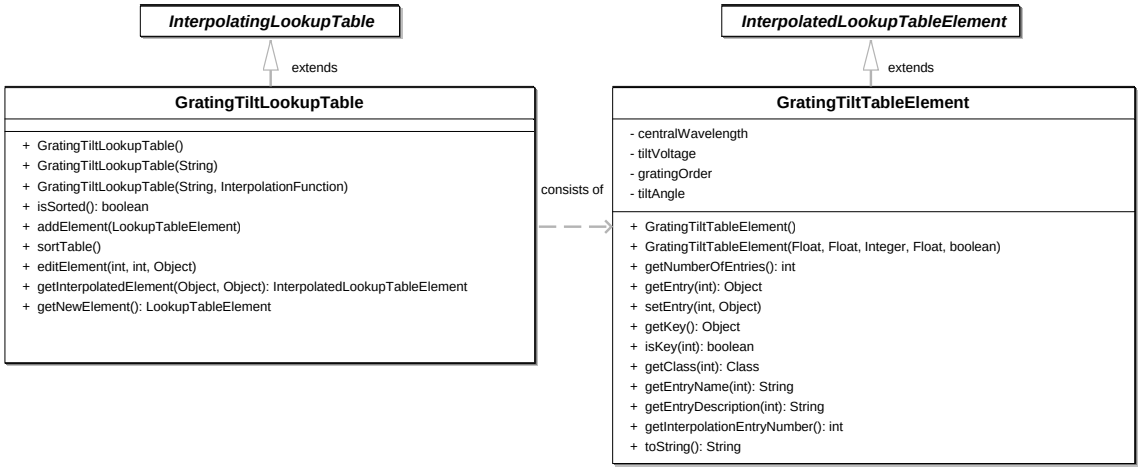


Figure 2.21: The structure of the `GratingTiltLookupTable` is illustrated in this figure using class diagrams. The table consists of `GratingTiltLookupTableElements`, which are shown on the right side. The super classes are shown above the two class diagrams.

by the software, it is included for additional information only. The key value (`k`) that is used to uniquely identify an element is calculated from the grating order (`GO`) and central wavelength (`CW`) fields, by the simple formula:

$$\mathtt{k} = \mathtt{GO} \times 100 + \mathtt{CW} \tag{2.2}$$

Since the central wavelength values are usually given in μm for the near infrared wavelength regime the values range from $0.9\,\mu$m to $\approx 2.5\,\mu$m and therefore the above formula computes a unique number for all sensible values. Additionally a check for sensible values is performed when a table entry is created.

With the functionality presented in the last section, the lookup table is able to map central wavelength values to tilt voltages. To be able to do this, the grating order has to be set using the `setGratingOrder(int)` method (see Fig. 2.20) first, because the gratings are used in different orders to center the desired central wavelength on the detector. This means that the same tilt voltage, corresponding to the same tilt angle, reflect different central wavelengths on the detector. Only if the grating order is also known, the mapping is unique. The broad band filters of the instrument are used to separate the overlapping orders during observations.

Calling the `tiltGratingToWavelength(float)` method will center the selected wavelength on the detector using the lookup table and interpolating values if necessary. To do the interpolation correctly and to make the tables more readable, the table values can be sorted using the key values for each entry, since the keys provide a logical sort order by the way they are calculated. This sorting of the table is done implicitly before the lookup of the tilt voltage is performed using the well established *quicksort* algorithm that is also used by Schimmelmann (2007), for example.

The method `tiltGratingToVoltage(float)` is for engineering purposes only, in order to provide a convenient way to set up the initial table values in the lab as well as for error checking in the case of some failure of the instrument hardware or software. This function simply calls the `setGratingUnitTargetVoltage(float)` of the `HIRAMO` service shown in Fig. 2.7.

To conclude this section Fig. 2.22 shows the engineering graphical user interface (GUI) for the `GratingUnit` service. The left side of the interface is dominated by the lookup table panel, allowing to edit existing tables as well as loading or creating new ones. Before data that has been

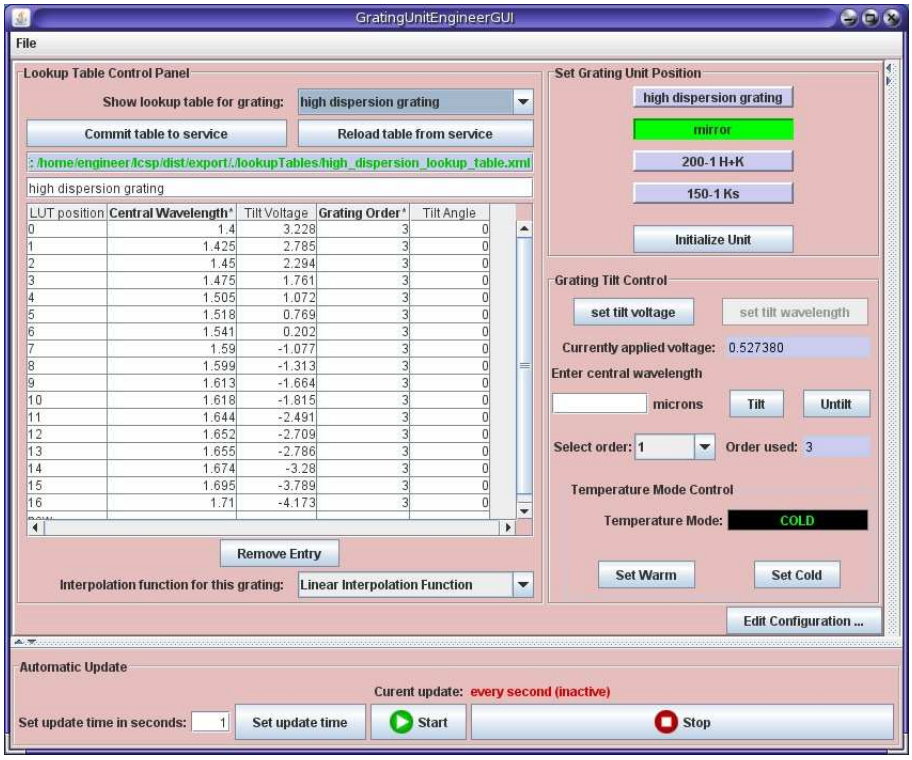


Figure 2.22: The `GratingUnit` engineer user interface. The lookup tables for the different gratings are shown on the left side, while the right side holds the panels to move the unit to different positions and to control the grating tilt.

changed is actually used by the service, the modified table has to be committed to the service first. The bold column descriptions indicate that these columns are used to compute the key value for the elements. For `GratingTiltLookupTables` these columns are the central wavelength and the grating order, as pointed out before. The current tilt values that are used for the H band can be seen in the figure.

The right part of the user interface is dominated by active control elements for the unit. In the top right section, the position of the unit can be selected and the unit can be initialized, if this is needed. Directly below this panel, the grating tilt control panel is located. The two buttons named "set tilt voltage" and "set tilt wavelength" change the display mode of the panel. In the mode shown in the figure, the central wavelength can be set, interpolating the tilt voltages from the table values if necessary. This is the normal mode when lookup tables have been created and the instrument is used by an engineer for tests of the `GratingUnit` service, for example. The other mode (not shown) allows to select a tilt voltage directly. This mode is useful when the table values have to be created or modified during lab tests, for instance. It allows to tilt the grating by applying the desired voltage directly. The resulting central wavelength on the detector can then be determined and the corresponding data added to the table. In both modes the grating order can be set using the corresponding selection field of the interface.

The "Temperature Mode Control" section of the user interface was added for convenience following a request by the engineers and is identical to the panel used in the engineering interface for the `HIRAMO` service (see Fig. 2.8). It allows to select the temperature mode for the *HIRAMO*

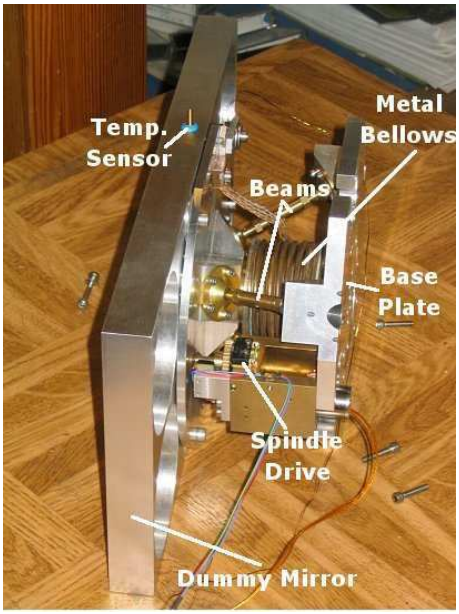


Figure 2.23: Photo of the *Compensation Mirror*, from Seltmann et al. (2004).

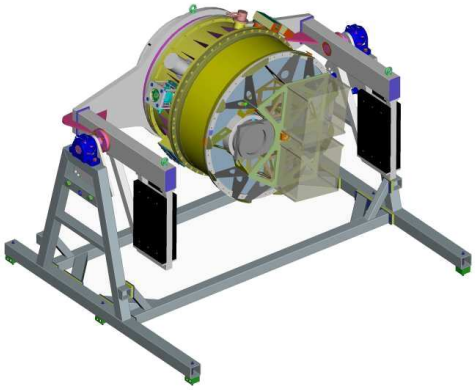


Figure 2.24: Drawing of the *Telescope Simulator*, taken from Seltmann et al. (2004). The instrument is seen from the back side.

electronic card. Details about this issue are given in Section 2.2.2.

2.3.3 Another Example: The `CompensationMirror` service

The `CompensationMirror` service is the second example that is used to describe the *Instrument Layer* services. One the one hand, it is more complex than the `GratingUnit` service with respect to lookup tables that are used and on the other hand, the movement of the physical unit is more simple, because only the `MCU` service is involved, i.e., there are no switches that have to be checked by the `HIRAMO` service. First the physical unit is introduced very briefly, then the software service follows afterwards.

The physical unit

Two identical mirrors are mounted inside the instrument that allow compensation of the movement of the image on the detector due to instrument flexure. This is necessary to keep the image motion on the detector in the sub–pixel range. Because of the compact optical design with four mirrors used in the collimator, it is impossible to stiffen the structure enough to fulfill this specification. Thus, active flexure compensation is required (Seifert, private correspondence). Referring to Fig. 1.4 in the optical path of the instrument, the movable mirrors are the first (M1) and the fourth (M4). The mirrors two and three (M2 and M3) are fixed. M1 is called *Alignment Mirror* and intended to be used for the initial alignment of the optics only, M4 is called *Compensation Mirror* and is used for flexure compensation. Both are identical, however, so in principal M1 could also be used to compensate instrument flexure. Fig. 2.23 shows the mirror mount with labels for the important parts. The mirrors can be moved around two axes using stepper motors connected to a spindle drive. The total tilt range is $\pm 1^\circ$ with an accuracy of ± 1 arcsecond. More details

about the physical unit are given in Seltmann et al. (2004).

The flexure of the instrument will depend on two variables, both of which have an effect on the resulting vector of the gravitational force acting on the instrument. The main influence on this force will be due to the *rotator angle* of the instrument. Another, less important influence, will be due to the *elevation angle* of the telescope. If the LUCIFER instrument was to be mounted parallel to the elevation axis of the telescope, this angle would effectively compensate some fraction of the rotation angle of the instrument. But since it is located at the "Bent Gregorian Front" focus, which is inclined with respect to the elevation axis, the resulting gravitational force on the instrument also depends on this angle.

The influence of different rotator and elevation angles can be tested in the lab using the *Telescope Simulator* (see Fig. 2.24). It is possible to rotate the instrument by $\pm 180^\circ$ and to tilt it by $\pm 30^\circ$, to simulate different elevation angles of the telescope (Seltmann et al. 2004). Detailed flexure tests have been made again at the telescope during the commissioning of the instrument. Results form these measurements are presented in the next paragraph after the structure of the lookup tables has been introduced.

The software service

Since both mirrors are physically identical, the `CompensationMirror` service is able to control the *Compensation Mirror* as well as the *Alignment Mirror*. To determine that the *Alignment Mirror* should be controlled, the command line argument `-controlAlignmentMirror` can be used when the service is started. To compensate the instrument flexure the mirror can be moved around two axes. To do this movement correctly, the position of the mirror must be known. Unfortunately, no encoders are available for the mirror control motors, so the absolute position of these motors, and thus the position of the mirror cannot be determined absolutely. The motion control electronics do have an internal step counter, but due to different effects, this step counter will not always represent the correct absolute motor position. The most likely effects for this are that the motor is loosing steps while it is working or a shift in the position due to some external force on the axis. This problem cannot be avoided completely, but to achieve a reproducible positioning of the motor an initialization routine has been implemented for the `CompensationMirror` service. This initialization tries to move both motors of the mirror to the negative limit switches, resetting the motor steps counter of the electronics to zero when the switch is reached. From this position, the motors move a predefined amount of steps to a nominal position that has to be determined by an engineer. This nominal position must be specified in units of motor steps for both motors and is stored in the service configuration file. Provided that no steps are lost during the movement from the negative limit to the reference position, a reproducible positioning of the mirror can be achieved. Of course this initialization has to be tested repeatedly, to confirm the positioning of the mirror. As one example for such a test, a special (pinhole) mask that creates one or more reference points on the detector image could be used, provided that the positioning of the mask in the focal plane is sufficiently accurate. The correct positioning of the mirror could then be confirmed by measurements of the reference point coordinates after the initialization and comparison with previous values.

The `MirrorTiltLookupTables` that are used by the `CompensationMirror` service, contain motor step values for both motors. These table values are defined as relative offsets for a motor to move in order to re–center a reference point, that can be freely chosen, back to its nominal position. Fig. 2.25 illustrates this. Suppose the reference point is moved on the detector by `dx` and `dy` pixel. In order to move the point back to its origin, the first motor of the *Compensation Mirror*

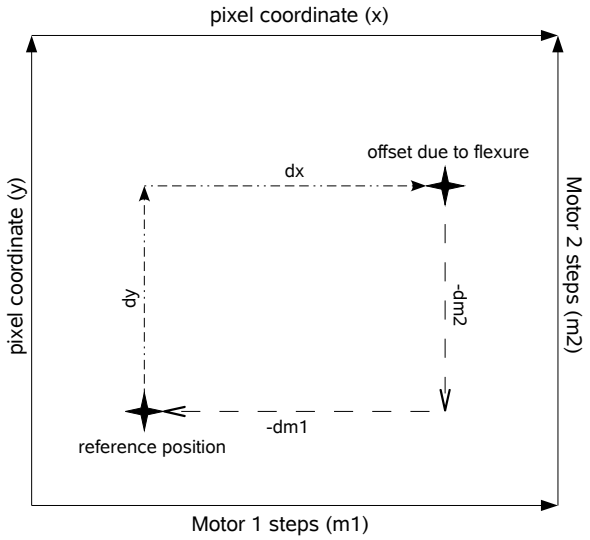


Figure 2.25: The flexure compensation principle: the reference point is shifted by dx and dy pixels. To compensate this, the motors have to be moved by −dm1 and −dm2 steps.

has to be moved by −dm1 steps and the second motor has to be moved by −dm2 steps. Note that for clarity reasons in Fig. 2.25 it is assumed that the pixel and motor coordinate systems are parallel. The procedure works just as well, if the two coordinate systems are rotated with respect to each other. This is the case for *Compensation Mirror* inside the LUCIFER instrument. There is a "crosstalk" of ≈4% between the two axes. This means that moving one motor for 100 steps in one direction moves the mirror an equivalent of 4 steps in the second direction. This makes the calculation of motor step values from pixel coordinates more complex but the procedure works as illustrated.

The service uses different lookup tables to determine the motor steps to compensate the instrument flexure. Analog to the `GratingUnit` service, these tables are loaded when the service is started, but can be altered and adjusted during operation of the service. For each of the three cameras, two lookup tables are used. The shift of the image on the detector obviously depends on the camera that is currently in the optical beam. Additionally, during the flexure tests it has become clear that a significant hysteresis is present when the instrument is rotated back and forth by a constant angle. Even after a full rotation by 360° a small residual flexure offset exists and the instrument is not back at its starting point. To accommodate for this, one lookup table is used for clockwise rotation and another one for counter–clockwise rotation. Figs. 2.26 and 2.27 show the latest measurements for the N1.8 camera. The telescope was positioned at zenith and the instrument rotated in 30° intervals from an angle of 0° to 360°. For each rotator angle position an image was taken with a sieve mask placed in the focal plane of the instrument. This configuration allowed to measure the movements of reference points for the different rotator angles precisely. As can be seen in the figures, the two curves show completely different shapes and are not closed. Flexure measurements have been made for various elevation angles of the telescope — horizon, 30°, 60°, and zenith — and the hysteresis can be seen at each elevation. The causes for this behavior is still under investigation. Tests with the `Compensation Mirror` service have shown that the flexure can be effectively compensated using different lookup tables for each rotation direction. The repositioning of the reference point has been better than 0.1 pixel when the instrument was rotated back and forth. However, the shape of the flexure curve seems

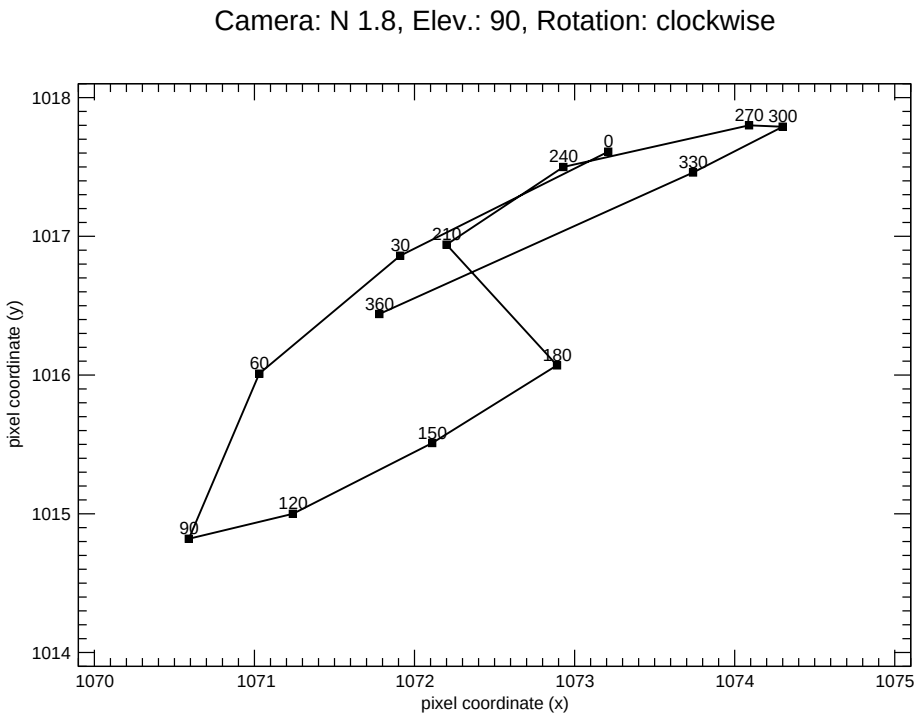


Figure 2.26: Flexure plot for the N1.8 camera for clockwise rotation with the telescope at zenith. Shown is the movement of a reference position for rotation angles from 0° to 360° in 30° intervals. The reference coordinates are pixel coordinates. The uncertainty of the position measurements is indicated by the size of the boxes.

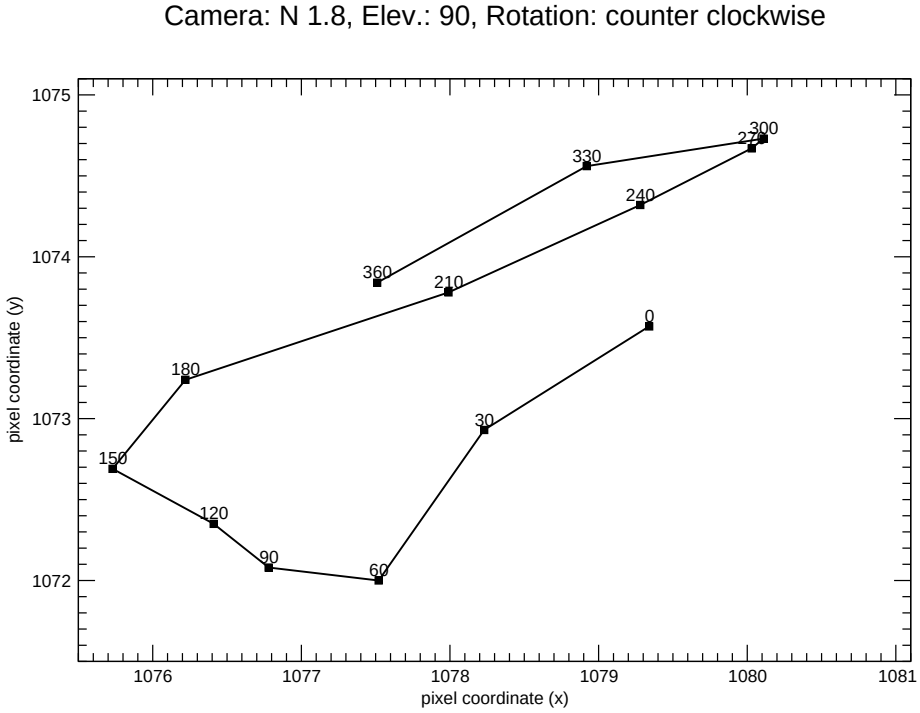


Figure 2.27: Same as in Fig. 2.26 but for counter–clockwise rotation.

to vary slightly, so that exact repositioning below 0.5 pixel could not yet be achieved every time. This value has been aimed for to minimize image shifts due to flexure below the expected image quality for operation of the instrument in AO mode (Seifert, private correspondence). The cause for this changes of the flexure curve need to be investigated further during the next months.

A natural choice for the key of an `MirrorTiltLookupTableElement` is the combination of a rotator and an elevation angle. For this reason the unique concatenation of both values to one string has been chosen. Further, to correctly interpolate values from the table data depending on the given rotator and elevation angles, the table must be sorted differently based on these angles. To achieve this, the `MirrorTiltLookupTableKey` class has been created. It is shown, together with the other lookup table classes for the `CompensationMirror` service, in the lower part of Fig. 2.28. To provide the ability to compare two keys, two methods can be used. The `greaterAs(MirrorTiltLookupTableKey, int)` returns `true` if the key object that is passed as a first parameter is smaller than the object for which this method was called. The method `smallerAs(MirrorTiltLookupTableKey, int)` returns `true` for the other case. The second parameter defines the sort order, which should be used to compare the elements. This is effectively defining whether to compare rotator angles or elevation angles. To test if two `MirrorTiltLookupTableKeys` are equal, the `equals(Object)` method can be used as usual. Together with the key, two motor step values are stored in the `MirrorTiltLookupTableElement`, which contain the necessary offsets from the reference position to compensate the flexure for the corresponding rotator and elevation angle.

In the left part of Fig. 2.28, the class diagram for the `MirrorTiltLookupTable` is shown. It extends the `TwoDInterpolatingLookupTable` class and the user is therefore able to define two distinct interpolation functions for each dimension. The first dimension is the rotator angle, the second dimension is the elevation angle. The interpolation functions for the angle `XX` can be set using the corresponding `setXXAngleInterpolationFunction(InterpolationFunction)` methods.

The sort order that is used for sorting the data entries can be set using the `setSortOrder(int)` method. The selected order is then passed to the compare methods of the `MirrorTiltLookupTableKey` objects as described above. When the data is interpolated the sort order is automatically adjusted for the needs of the current interpolation, however.

The description of the `CompensationMirror` service as an *finite state machine* is much simpler than for the case of the `GratingUnit`. Basically, only two states are possible:

- the unit is initialized or
- the unit is not initialized.

The difference between the two states is that for the former state the motor positions are known by the software service, while for the latter one, the positions are not known. Since there is no encoder available for any of the motors, this position value is somewhat arbitrary. Therefore, the service can be initialized in two ways. The first way is called passive initialization, and the current position of the motors are just read form the motion control electronics. This may be useful if the mirror is already in position (verified by taking an exposure, for example) and the service should be initialized without mirror movement. Another possible scenario for passive initialization would be the case of a temporary shutdown of the `CompensationMirror` service and a later restart, *without* any interaction with the mirror motors in between and assuming that the electronics are not switched off. The second mode is called active initialization and reflects the procedure presented earlier: both motors are moved to the negative limit first, the step counters

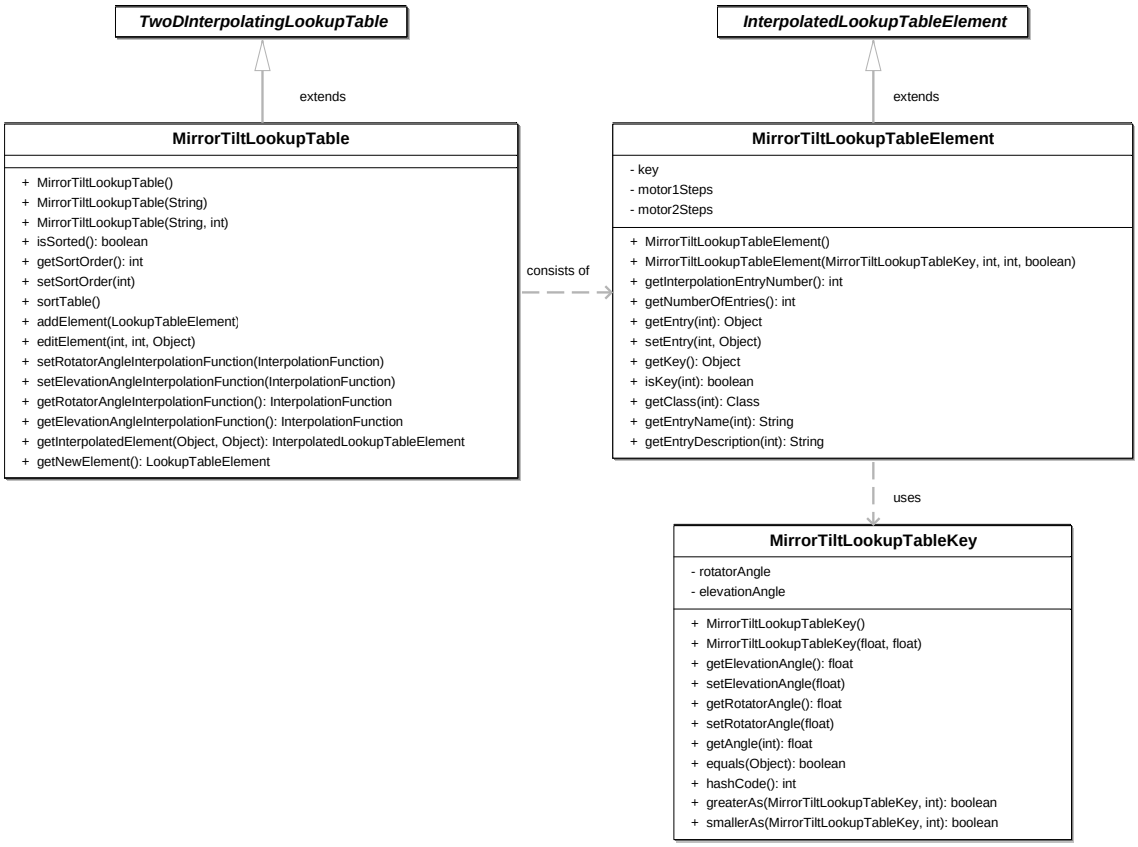


Figure 2.28: Class diagram of all classes necessary for lookup data for the `CompensationMirror` service.

are reset to zero and afterwards the motors are moved by the reference steps defined in the service configuration. The active initialization of the mirror is always recommended, if the current mirror position is not known a priori, or the motion control electronics have been reset or switched off. It also helps to avoid the problem that the mirror sometimes fails to move correctly when only small correction movements have been sent to the motors for a longer time period, i.e., a couple of days. This behavior has shown up during the final lab tests and the commissioning of the instrument, but can reliably been solved by driving the motors for a larger amount of steps back and forth.

The class diagram of the `CompensationMirror` service is shown in Fig. 2.29. The two different ways of initialization can be selected by using the `initializeMirror(boolean)` method, where the argument set to `true` defines active initialization. The mirror that is controlled by the service can be determined with the `getMirrorUsage()` method. Note that there is no `setMirrorUsage(int)` method, so changing the controlled mirror is not possible while the service runs. This has been done on purpose to avoid confusion and errors due to misconfiguration of the service while operating the instrument, i.e., due to mixed up lookup tables. Instead, two instances of the `CompensationMirror` service should be run, one controlling the *Compensation Mirror*, the second controlling the *Alignment Mirror* with different lookup tables for each of the services. These lookup tables can of course be modified or altered while the services are running.

Additional methods are provided to get and set the different values that are needed to interpolate the data from the current instrument and telescope configuration, like the camera position,

Figure 2.29: Class diagram of the `CompensationMirror` service.

rotator and elevation angles. It is important to note at this point, that the `CompensationMirror` service does not try to get the corresponding values on its own (i.e., by using a client for the other services). Instead, the values have to be set explicitly. This is done on purpose to ensure that the service can be used, even in the situation when other services are not available. The coordination of the necessary interaction between the service is left to the different *manager services* of the *Operation Layer* which will be described in detail in Section 2.4. Of course, the values can also be set directly by an engineer, if this should be needed.

The `adjustMirror()` method is used to directly adjust the mirror to compensate the instrument flexure depending on the currently set conditions. To test the interpolation routines before actually moving the mirror the `simulateAdjustMirror()` method must be used. It returns two integer values with the calculated motor steps for motor one and motor two, respectively. The motor positions can be read with different methods: the `getMotorPositions()` method returns the buffered values as they were last read by the service (passive read), while the `getCurrentMotorPositions()` method actively reads the values from the electronics. This behavior is analog to the `GratingUnit` service (see last section) and the environment control services introduced in Section 2.2.2.

To summarize the `CompensationMirror` service section, Fig. 2.30 shows the engineering user interface to the service. The mirror usage of the service is shown in the title section of the user interface. As can be seen in the figure, the *Compensation Mirror* is controlled in the presented example. Analog to the engineer interface for the `GratingUnit` service (see Fig. 2.22), the left part of the interface shows lookup table data that is used by the service. In the figure some of the data for the N3.75 camera can be seen for clockwise rotation of the instrument. As mentioned above the measurements where carried out using 30° intervals for the rotator angle at each chosen elevation of the telescope. The elevation of the telescope was also varied using 30° steps from

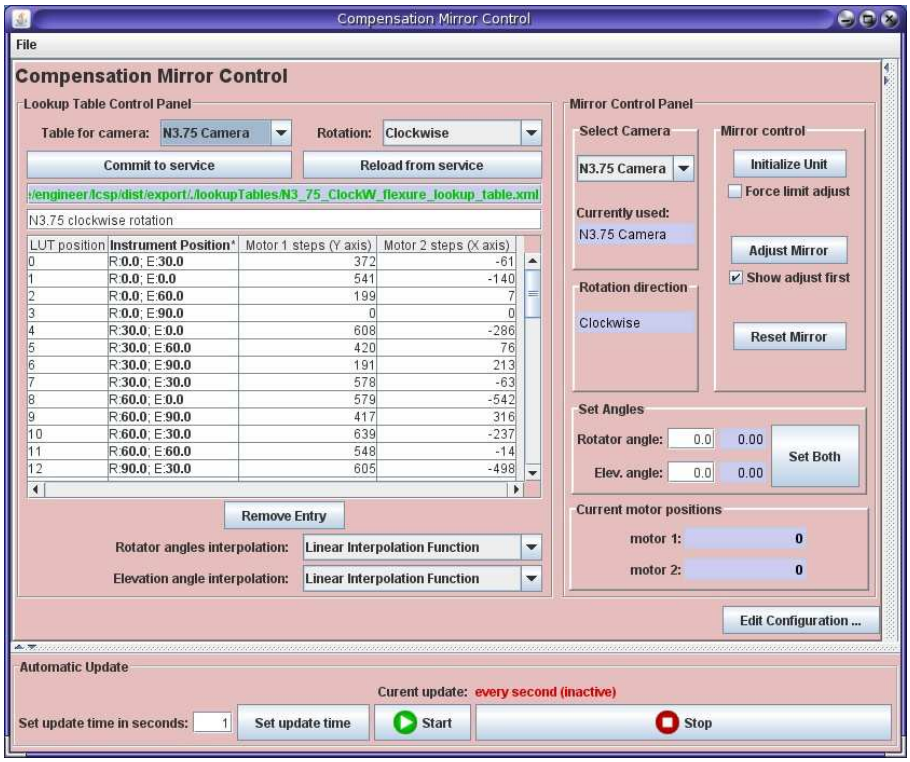


Figure 2.30: The engineer user interface for the `CompensationMirror` service. Lookup table data is placed left, the service control elements are located on the right side of the interface.

the horizon to the zenith. The lookup tables can be edited, saved or loaded locally. Like for the `GratingUnit`, before modified tables are actually used, they must be committed explicitly to the `CompensationMirror` service.

The right section of the engineer interface contains the control components. The current camera can be selected using the drop down GUI element. Below the "Select Camera" area the current rotation direction of the instrument is shown. This cannot be set directly, Instead the direction is computed from the changing rotator angle values. Next to these two components the "Mirror control" area is located. Three buttons are available, namely the "Initialize", "Adjust Mirror", and "Reset Mirror" buttons. The check–boxes beneath the upper two buttons determine the behavior of the corresponding buttons, with the "Force limit adjust" check–box specifying an active initialization when it is selected, and the "Show adjust first" check–box displaying the calculated motor steps for the mirror first, without actually moving it. The third button can be used to reset the mirror to the position it was before the flexure compensation has been started. The rotator and elevation angle can be set using the corresponding components, and the current mirror positions are shown in the bottom right section of the interface.

Altogether, the user interface looks very similar to the one used for the `GratingUnit` service (Fig. 2.22) to minimize the learning curve for handling of the instrument units for the engineers. This is the case for all other user interfaces of the *Instrument Layer* that use lookup table data, like the `DetectorUnit` and the `FilterUnit` services. Different from this layout are the more simple interfaces to the `CameraUnit` and `PupilViewer` services.

While the services of the *Instrument Layer* provide all the functionality necessary to *control*

the LUCIFER instrument and its components, it would be very cumbersome to actually *use* the instrument for NIR astronomical observations, if the software development would stop at this point. Effective observations require the automated interaction of three elements, namely the telescope, instrument and detector read–out. The detector read out is separated from the instrument in this case because it is controlled by an external software system. All of the needed functionality is located in the upper most layer of the LUCIFER control software package, the *Operation Layer*.

2.4 Operation Layer

The *Operation Layer* is the highest level of the LUCIFER control software package (see Fig. 2.2). In this layer, the components that are used to *operate* the instrument are located, including software to plan observations, software to carry out these observations in an automated fashion and software packages to manage the whole system to work seamlessly together.

The section begins with general remarks about the operation of the instrument, followed by a detailed overview about the user interaction principles that have been implemented to allow different kinds of access levels to the software system. The work on the *Operation Layer* is still in progress at the time of writing of this thesis. A substantial fraction has been implemented during the commissioning of LUCIFER 1 in the last months. Still, the planned operation mode of the instrument has not been completely finished. To nevertheless allow for an efficient handling of the instrument during that time, an intermediate tool for automated use of the instrument has been implemented. It enables the processing of text scripts and relies on the same operation mechanisms, i.e., the three *managers*, as the more sophisticated tools that are finally foreseen. These observation scripts are thus ideally suited to explain the principles and have been successfully used during the commissioning. They are presented in 2.4.2. The three *manager* services, the `Instrument Manager`, `Readout Manager`, and `Telescope Manager`, that are used to control the different subsystems are introduced after this.

The work on this layer is and has been carried out by different people, with a substantial fraction of the planning software as part of a diploma thesis (Schimmelmann 2007). The `Instrument Manager` and `Telescope Manager` have been implemented by Dr. M. Jütte. References to these works will be clearly stated.

2.4.1 Observing with LUCIFER

Gone are the times when astronomers interacted directly with the telescope, i.e., moving its axes to get a new pointing on the sky, or with the instruments they were using, like changing and developing photographic plates. Today, the operation of an astronomical instrument (or telescope) is more like controlling a complex machine used in industry. The main reason for this is that modern instruments have become much more complex than in past times, of course. Only a couple of decades ago observations were made with "simple" photographic plates, which had to be cut to the right shape, placed in the focal plane of the telescope and developed after the exposure had been taken. Observations with an instrument like LUCIFER are much more complex in detail. The instrument has hundreds of switches, many motors and complex electronic equipment, all of this is working at cryogenic temperatures in a vacuum, so no manual interaction with the hardware of the instrument is possible. However, the typical astronomer should not have to worry about these details. This means that nowadays, the control software has to take care that the observations are nearly as "simple" as with photographic plates.

The second important reason for the changing needs of operating an astronomical instrument is high efficiency. Observation time is getting more and more expensive, a single night at the LBT costs many 10.000 US dollars, for example. Since the weather conditions cannot be changed — even at the best astronomical sites in the world the weather is bad during some nights of the year — the available observation time must be used as effectively as possible. This puts high pressure on the performance of instrument and telescope. Especially for the instrument this means that setup changes must be done quickly, at best while the telescope is moving to the next target. For the case of near infrared observations, the efficient interaction between the telescope and the instrument is of utmost importance. Since these observation require many telescope movements between exposures (for details about this point see Section 3.2.2), an unnecessary loss of only a couple of seconds between every exposure and telescope movement can accumulate to many minutes or even hours for a whole night. The concepts that are applied to achieve an effective automated operation of the LUCIFER instrument are presented in Section 2.4.2.

Although high efficiency can best be addressed with an automated operation of the instrument, the operation of an instrument also always involves the interaction with users. The control software must account for the different interaction needs of these users. They can be distinguished into two main groups: *engineers* and *observers*. The former need access to all components of the instrument, without having to rely on the more complex software services, which might fail after all, while the latter want efficient observations, that at the same time can be changed quickly depending on the needs that arise during the night. Additionally, these observers are not, and need not to be, very familiar with the details of the instrument, so the interaction possibilities provided to them must ensure that no damage can be done to the instrument. The different interaction principles and the tools that are provided in the software package are presented in the following sections.

User interaction principles

The functionality that a software must provide to the user depends strongly on the intended type of user. If different user types are supposed to work with the same system, the necessary functionality also takes multiple characteristics. If one considers the product database system of a company, the possible user types could be customers that simply use the data that is stored in the system, e.g., by browsing through the catalog of available products of the company, employees that are allowed to edit and create new data, and database administrators that are able to change the layout of the system. Each of these users need a different interface to the system, graphically as well as with regard to the provided functionality.

For the application of an astronomical instrument, the main user types are *engineers* and *observers*. Included in the first group are the system engineers that work with the instrument while it is still being integrated in the lab, as well as the instrument scientists that are later responsible, when the instrument is already commissioned and running at the telescope. To the observer group belong literally all other users of the instrument that use it in order to carry out scientific observations. To represent these different user types, their access to the software and thus to the instrument itself, three different `Interfaces` have been created in the software:

1. `Basic`: this interface defines the most rudimentary access to a service that is possible. For some services this includes only the function to test whether the service is running or not.

2. `Observer`: this interface defines functions needed by the observer user group, i.e., astronomers.

3. `Engineer`: this interface provides full access to all methods offered by a service to allow full control over the system.

The implementation of this access levels to the software is shown in Fig. 2.31. On the left side of the diagram are the three main interfaces that were just introduced. As can be seen in the figure, the three interfaces are linked by inheritance, which means that each level of access automatically includes all methods that were defined at lower levels. `Observers` always have at least the same access as defined for the `Basic` interface and `Engineers` inherit the access from `Observer` (and subsequently also from `Basic`) levels. The right side of the figure shows the access that has been defined for the `GratingUnit` service (see Section 2.3.2). As can be seen in the figure, basic access to the `GratingUnit` service includes the ability to query the position of the unit (as it was last read by the service, see below) as well as the test if the service is initialized or running at all. `Observer` access to the service grants the further rights to set a new position for the unit, as well as the ability to tilt and untilt the gratings, to name the most important functions. The `Observer` interface therefore defines all methods that are necessary for operating the instrument from the point of view of an astronomer who wants to do his/her science. It does *not* include the ability to change the used lookup tables or modify the used interpolation functions. Further, any active interaction with the hardware as the `getCurrentPosition()` or `getCurrentTiltVoltage()` methods would carry out is also not available at the `Observer` level, but is exclusively reserved for `Engineer` access. If all three interfaces are taken together, the full functionality of the `GratingUnit` service is provided (compare with Fig. 2.20).

Furthermore, another distinction between the different access levels is made. Apart from providing varying *access methods* at different levels, changing *access types* are used also. Direct access, i.e., the method calls defined in the interface are directly forwarded to any of the services, is only possible at the `Engineer` level. The other two access levels only provide indirect access to the service, which means that the client that implements these interfaces does not contact the corresponding service directly. Instead, depending on the relevant service, one of the three *manager* services, i.e., the `InstrumentManager`, `ReadoutManager` or `TelescopeManager` is contacted and the request is send. The command that was issued by the observer is then carried out by the manager, by contacting the service of the *Instrument Layer* or *Control Layer* when it is save to do this. This is a necessary requirement to prevent damage to the instrument by an inexperienced observer.

The differences between the varying interaction principles are discussed in detail in the next paragraphs.

Engineers *vs.* **Observers**

How does access control work in practice? As an example consider again the `GratingUnit` service. If, for instance, the lookup table data for the `GratingUnit` needs to be changed because more precise measurements have been carried out since the table was set up, this cannot be done using the `Observer` interface, because at this level no method exists to change the lookup tables. This reflects the essential necessity that observers can not mess up the configuration of the instrument by accident. Therefore, the `GratingUnitEngineer` interface must be used. An obvious point to control the access to the software is the interface that is provided for the users. Thus, every graphical user interface of the control software extends one root class, which is called `LPanel` (see Fig. 2.32). It defines three constructor calls, one for each access level. Thus, the access that is possible with the `LPanel` is defined when the interface is created, by passing a client of appropriate

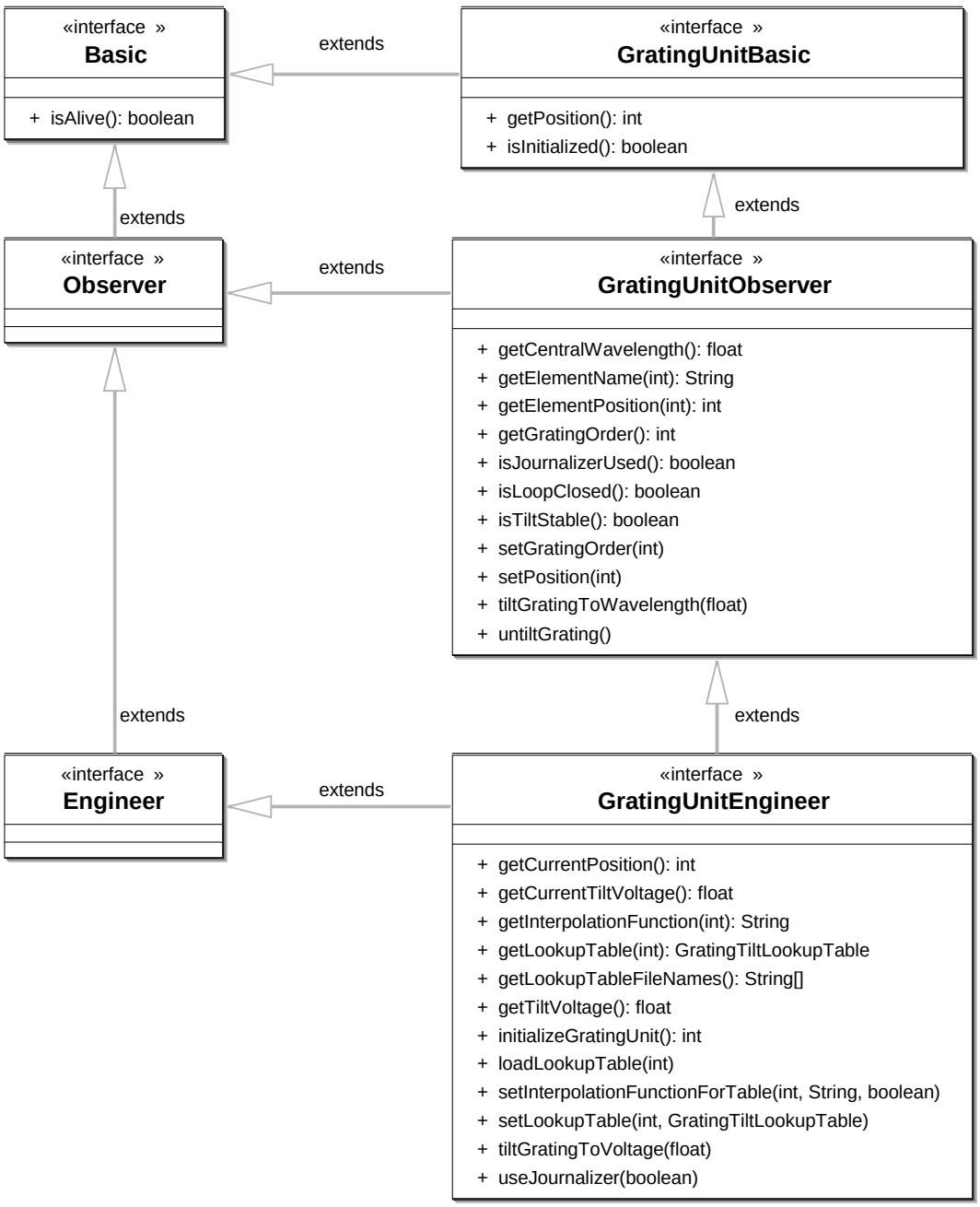


Figure 2.31: This class diagram illustrates the different service access levels. The general interfaces shown left distinguish between the main access to a service. On the right side of the figure, the different levels are shown for the example of access to the `GratingUnit` service.

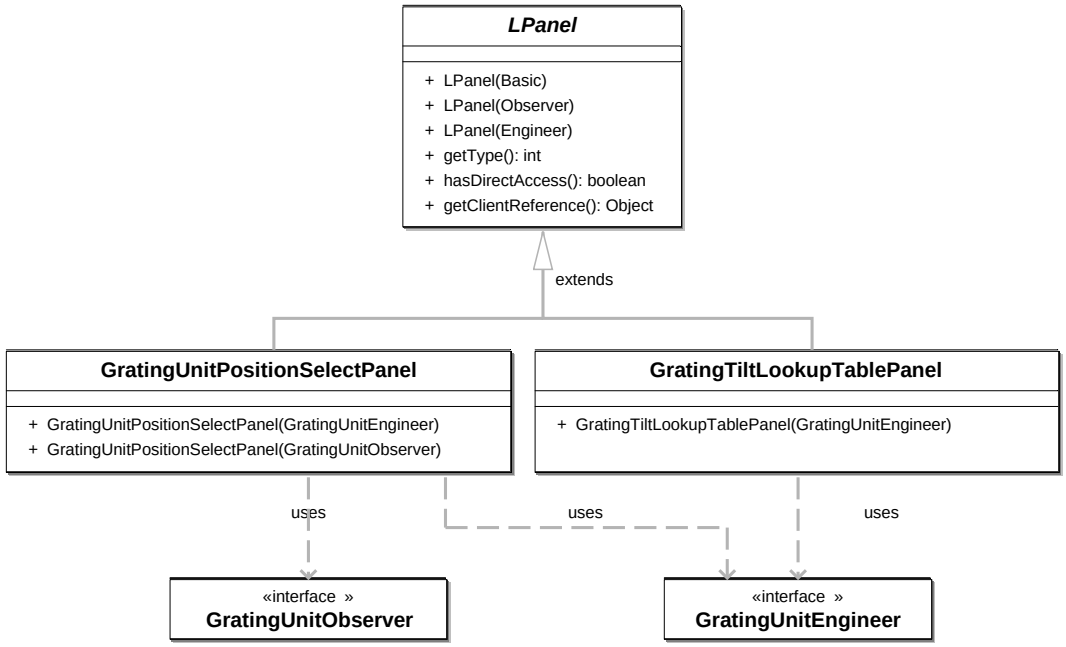


Figure 2.32: This class diagram illustrates how access to the `GratingUnit` service is managed for graphical user interface component. The `LPanel` class defines three constructor calls with different access types as a parameter. The `GratingTiltLookupTable` class only has one constructor, allowing `Engineer` access, while the `GratingUnitPositionSelectPanel` allows access for observers as well as engineers. For clarity, only the most important methods are shown.

access kind (i.e., `Basic`, `Observer`, or `Engineer`) to one of the three constructors. The different components that the panel consists of for data display and to control the underlying service can use this client to get their data or invoke actions by calling the `getClientReference()`. To test if the panel has engineering access to the underlying service, the method `hasDirectAccess()` is provided. This is used to change the background color of the panel, to visualize direct and indirect access, for example.

In the middle of Fig. 2.32, two implementations of a `LPanel` are shown, which are used by the graphical user interface for the `GratingUnit` service. First, the focus is set on the `GratingTiltLookupTablePanel` shown on the right side of the figure. It is the component that is used to modify lookup table data of the service. The panel has only one constructor, which takes a `GratingUnitEngineer` as an argument. This already defines that only clients with `Engineer` access to the service can be used with this panel, and therefore clients for `Observer` access do not get the chance to modify lookup table data by accident.

Apart from the fact that using the `LPanel` class easily allows access control to the services, another useful feature is provided as well. The same graphical components can be used for different access types. The `GratingUnitPositionSelectPanel`, shown on the left side of Fig. 2.32, illustrates this. The panel allows to select the position of the *Grating Unit* by clicking on the appropriate button. Obviously, this functionality is needed both by engineers and observers. Consequently, the `setPosition(int)` method is already defined in the `GratingUnitObserver` interface (see Fig. 2.31). The `GratingUnitPositionSelectPanel` defines two constructors, one for `Engineer` access, the other for `Observer` access. From the point of view of the panel components, it does not make a difference which client is actually used to invoke the `setPosition(int)`

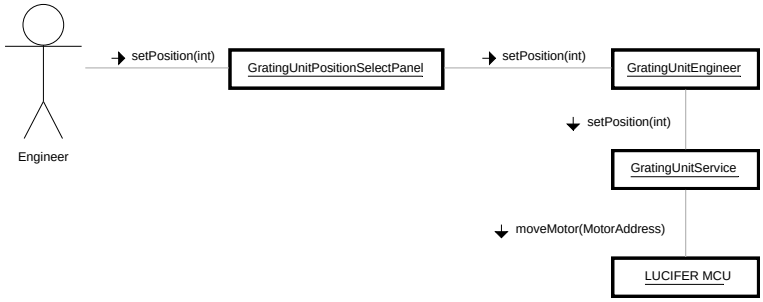


Figure 2.33: This collaboration diagram shows the involved classes when an engineer selects a new position for the `GratingUnit` service. The call is forwarded directly to the service by the client.

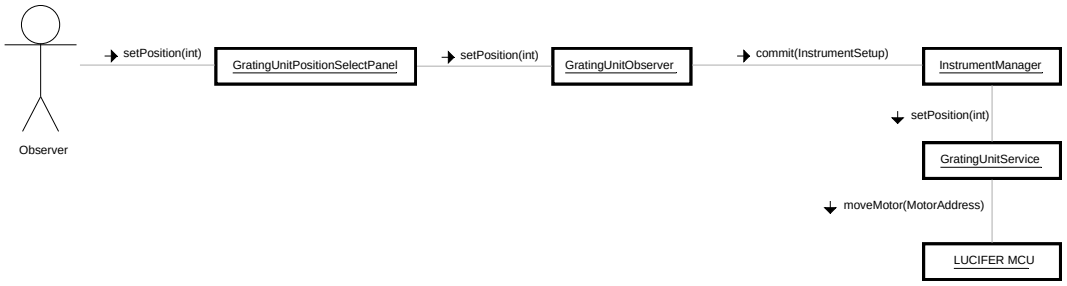


Figure 2.34: This collaboration diagram shows the involved classes when an observer selects a new position for the `GratingUnit` service. The call is forwarded as a new `InstrumentSetup` to the `InstrumentManager`. When it is safe to do so, the new position is committed to the `GratingUnitService`, which itself subsequently sets the position of the unit.

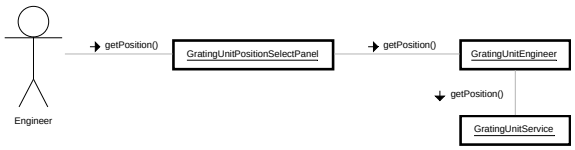


Figure 2.35: This collaboration diagram shows the involved classes when an engineer queries the position for the `GratingUnit` service. The call is forwarded directly to the service by the client.

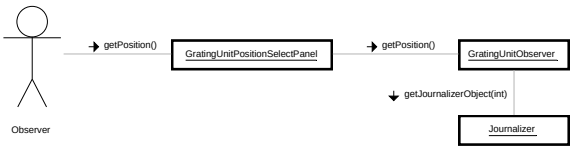


Figure 2.36: This collaboration diagram shows the involved classes when an observer queries the position for the `GratingUnit` service. The call is not sent to the service. Instead, the `Journalizer` service is contacted.

method, since for both types the method exists. This is always guaranteed because of inheritance, as is shown at the bottom of Fig. 2.32. The two access interfaces are the same as the ones shown on the right side of Fig. 2.31.

What will happen now, when different clients are used with the panel. Fig. 2.33 shows a collaboration diagram of the involved classes for engineer access. The engineer selects a new position for the unit by clicking on one of the buttons. This call is then sent from the panel to the `GratingUnitEngineer` client. Since the client has `Engineer` access to the `GratingUnit` service, the call is directly forwarded to it. Assuming that the physical unit is able to move, it will begin to do so immediately, because the `GratingUnit` service sends a `moveMotor(MotorAddress)` command to the `MCU` service for the instrument. This will happen regardless of the current states

of the other units, i.e., even if an exposure is currently taken. It is assumed that the engineers know what they are doing and are experienced enough not to make mistakes easily.

If we look at `Observer` access on the other hand, the situation is different. Fig. 2.34 shows this case. The observer selects a new position for the unit by clicking on a button of the panel. This call is sent to the `GratingUnitObserver` client. This time, the client creates a new `InstrumentSetup` and sends it to the `InstrumentManager`. When it is safe to do this, i.e., no exposure is currently being taken, the `InstrumentManager` sets the new position of the unit by contacting the `GratingUnit` service. As can be seen by this example, there is no direct interaction between the observer and the `GratingUnit` service, although from the point of view of the graphical component, no difference exists for the two cases.

`Observer` access does not require direct interaction, even to get a position from any service. Figs. 2.35 and 2.36 show this for the `Engineer` and `Observer` access levels, respectively. For `Engineer` access the call of the `getPosition()` method is forwarded to the `GratingUnit` service directly, as usual. The `GratingUnitObserver` client, does *not* contact the `GratingUnit` service for the position (see Fig. 2.36). Instead, the `Journalizer` (Section 2.2.1) is queried for the last `GratingUnitStatus` object. The position of the unit can be extracted from this object by the `GratingUnitObserver` client using the `getPosition()` method (see Fig. 2.5 on page 29).

Special needs for Observers

As mentioned in the introduction of the *Operation Layer* section, modern astronomical observations require efficient operation of telescopes and instruments. This implies that, apart from distinct access rights between engineers and observers concerning the direct control of the instrument, observers need more tools at their hand to operate the instrument than the simple direct setup of the instrument using some buttons to select different cameras, filters and gratings. Especially for near infrared astronomy, observations have to be well planned before they are supposed to be carried out (see chapter 3). Most observatories with large telescopes today have tools that allow astronomers to prepare their observations when they successfully proposed for observation time. One prominent example is the Phase 2 Proposal Preparation (P2PP) tool that is used for all active telescopes at ESO.

It was thus straightforward to decide that for the LUCIFER control software, an OPT would be highly beneficial. The development of the tool was part of a diploma thesis (Schimmelmann 2007). However, the OPT has not been used to plan observations with the LUCIFER until now because it relies on some parts of the *Operation Layer* that are not fully operational yet. One missing service is the `Scheduler`, which has been designed and implemented as a prototype in the scope of Schimmelmann (2007). For the commissioning of the instrument the prototype could not be used, however, because it could not yet be adapted to the current state of the software. Therefore, a another tool has been developed to fill the "gap" until the *Operation Layer* is fully completed. It allows the parsing and processing of text files that describe the observations to be carried out in a fully automated way.

2.4.2 Observation scripts

Efficient observations require a good coordination between the telescope and instrument. For the instrument, an additional separation for the detector and all other units is made in the software architecture. These three entities, telescope, instrument and detector, must be controlled in such a way that the different work packages that need to be carried out are distributed effectively.

This task will finally be handled by the Scheduler service. But since it is not available yet (see above), the decision was made to use scripting files to carry out the observations automatically. The scripts are simple text files that can be edited easily. They have the following structure:

```
[START_INSTRUMENT_SETUP]
INSTRUMENT PARAMETER          =value                #comment
...
[END_INSTRUMENT_SETUP]

[START_TELESCOPE_SETUP]
TELESCOPE PARAMETER           =value                #comment
...
[END_TELESCOPE_SETUP]


[START_READOUT_SETUP]
READOUR PARAMETER             =value                #comment
....
[END_READOUT_SETUP]


[START_OBSERVING_SETUP]
OBSERVATION ELEMENT           =element parameters   #comment
...
[END_OSERVING_SETUP]
```

Every script can have up to four sections. These sections are defined by the special tokens [START_xxx_SETUP] and [END_xxx_SETUP]. The four sections of the script are mapped to the following objects that describe the observation:

1. An InstrumentSetup containing all information about the configuration of the LUCIFER instrument. It is evaluated by the InstrumentManager.
2. A TelescopeScriptInformation with all data that is necessary for pointing the telescope to a given target. It is evaluated by the TelescopeManager service.
3. The ReadoutSetup holds the necessary information for the ReadoutManager.
4. The observation itself is contained in an ObservationScriptSetup object.

Fig. 2.37 shows a class diagram of the ObservationScriptParser and the important classes mentioned above. The observation script file that should be analyzed is passed as an argument when the ObservationScriptParser object is created. By calling the different parsing methods, parseInstrumentSetup(), parseTelsecopeSetup(), parseReadoutSetup() and parseObservationSetup(), all necessary information is created. Any errors present in the script will be detected and a subsequent ScriptParseException will be thrown. The checking includes both the syntax of the script as well as a basic check for valid parameters, i.e. , for allowed instrument configurations.

If some sections are not wanted or not needed in a script they can be left out. This is useful for several situations, e.g., if no InstrumentSetup needs to be specified, because the instrument

is already in the correct configuration. Additionally, this introduces a great flexibility into the scripts, since they can also be used for taking calibration data for the instrument, like flat fields or focus sequences. For these calibration scripts no `Telescope Setup` is necessary, for instance. This feature to be able to leave out parts of the script also allows very flexible and easy operation of the instrument during astronomical observations, because it allows different scripts to be used for target acquisition and science observations. This is explained in more detail at the end of this section.

For the setup of the telescope, instrument and detector the parameters are straightforward. Schimmelmann (2007) gives a detailed description of this. How the scripts model an astronomical observation is more complex, however. The `ObservationScriptSetup` class which describes these observations is shown in Fig. 2.38. The class is a container for `ObservationSetupElements`. Using the `addNewElement(ObservationSetupElement)` method, the parser adds different elements to this objects as they are encountered in the script. The elements can be retrieved later by their position in the setup and `getNumberOfElements()` can be used to query the total number. This allows to loop through the whole setup, executing one element after the other.

The `ObservationSetupElement` class and the subclasses that have been defined so far are also shown in Fig. 2.38. The class has one attribute, a field to determine that the current element describes an acquisition position. When this attribute is set to `true` the automatic execution of the script is paused until the observer explicitly resumes it. One possible application is the review of an initial pointing of the telescope at the beginning of an observation. If the astronomer is not satisfied with the pointing of the telescope he/she can correct it before the observation script is resumed. Although such an acquisition is most commonly used at the beginning of a script, it can be used anywhere in the setup.

Four types of `ObservationSetupElements` have been defined so far. These are shown with their most important methods in the lower part of Fig 2.38. Only two of the types are used for scientific observations, namely the `OffsetPosition` and the `PointingPosition`. The former moves the telescope by a small offset from the initial pointing, the latter can be used to point the telescope to any position in the sky. This is useful for observations of large targets, where the telescope has to be moved to sky positions far away from the object. The `FlatPosition` and `FocusPosition` types are used for automatic calibration measurements with the instrument. The former type allows to take so–called flatfields to determine the detector sensitivity, the latter can be used to measure the internal focus of LUCIFER. Each `ObservationSetupElement` has multiple parameters, e.g., offset values or focus ranges. An exception to this rule is the `FlatPosition`, which does not need any parameters. It simply triggers the start of an exposure of the detector. A description of all parameters that are available would go beyond the scope of this section. A summary is shown in Fig 2.38 which displays the getter methods — and thus the parameters — for every type of `ObservationElement`. Additionally, a template script showing all possible parameters is printed in Appendix B.

To provide more flexibility for the scripts in the future, it is sufficient to define new types of `ObservationElements`. This has already been done in the past, e.g. , the `FocusPosition` type was not planned at the beginning. It was implemented following a request from the instrument engineers for an automated procedure of focus measurements. For the near future some template types are foreseen to automate imaging scripts even more, than is currently possible. One proposed type will allow "jittered" imaging. The astronomer will specify the number of exposures to take within a certain distance from a reference point, rather than every offset explicitly. The template will then compute random offsets of the telescope for the given constraints. This will increase the flexibility of observations with LUCIFER even further.

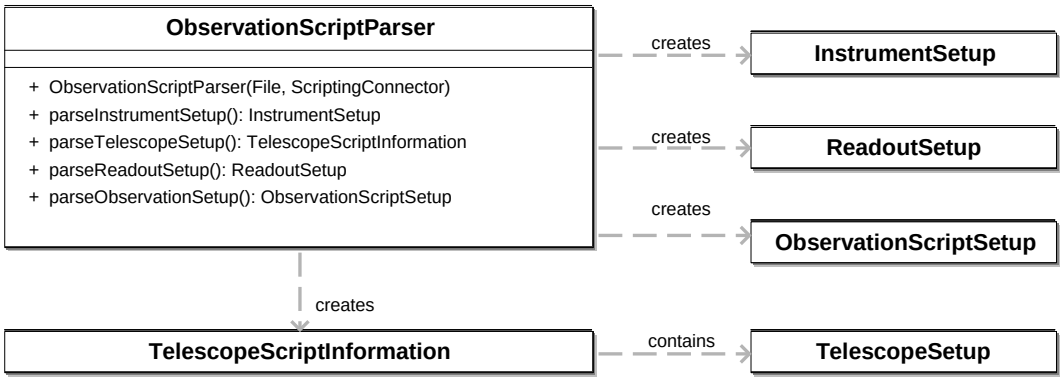


Figure 2.37: This class diagram shows the `ObservationScriptParser` and the relevant classes of objects that are created when an observation script is read.

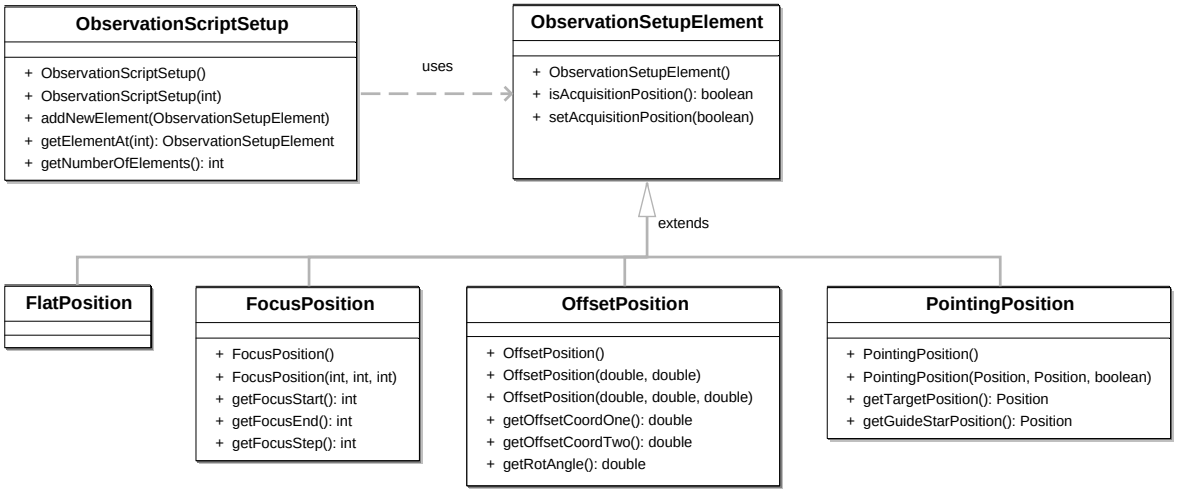


Figure 2.38: This class diagram shows the various `ObservationElements` that can be used in observation scripts. These are stored in an `ObservationScriptSetup` objects, for which the class diagram is shown in the upper left part of the figure.

How are these scripts executed, after they have been parsed? This task is completed by the `ScriptExecutionHandler` shown in Fig. 2.39. It gets up to four setups, if all four are specified in the script, from the `ObservationScriptParser` and sends three of them (for the instrument, telescope, and readout) parallel to the responsible manager. "Committing a setup" has become the usual terminology for this sending. The `commitInstrumentSetup()` instructs the `InstrumentManager` (see next section) to change the setup of LUCIFER for the next observation. At the same as the instrument setup is being sent, the `commitTelescopeSetup()` is executed to preset the telescope to the desired position using the `TelescopeManager`. The wanted detector configuration parameters are committed to the `ReadoutManager` also. Since the needed time to configure all three units is different, with the telescope preset normally taking longest, and the detector setup being the fastest, the continuation of the script execution has to be paused until everything is ready. To test this the `isInstrumentReady()`, `isTelescopeReady()`, and `isReadoutReady()` methods are used. Any errors during the initial setup are reported with the `setError(CommitException)` method, which causes the stopping of the script execution

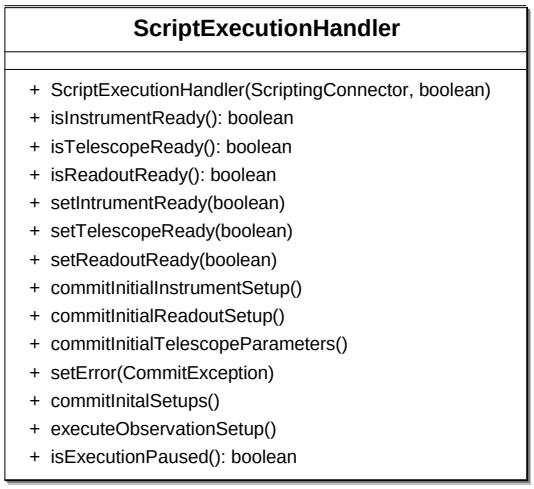


Figure 2.39: The class diagram of the `ScriptExecutionHandler`.

and reporting the failure to the user. After the initial setup has successfully been committed, the `executeObservationSetup()` method is called. It loops through all elements of the `ObservationScriptSetup`, processing one element after another. For each element the specified `ReadoutSetup` of the script is executed.

The capability of scripts is explained best by an example of how to use them for observations. For instance, the acquisition and observation strategy for spectroscopy is the following: two scripts with different instrument setups are created and executed subsequently. The first script does the presetting of the telescope and target acquisition using the imaging setup of the instrument.The second script — without a `TelescopeSetup`, thus leaving the telescope position unchanged — changes LUCIFER to a spectroscopic mode and defines the exposure parameters for the science spectra, i.e., central wavelength, exposure times, number of exposures per position, etc. The second script further defines the offsets of the telescope to create a suitable observing pattern, like "nodding on the slit" (see Section 3.2.2). Alternatively the initial setup can be done manually using the graphical user interfaces. A script is used afterwards to take the exposures automatically. During the commissioning this procedure has turned out to allow very efficient and flexible operations with the instrument.

Regardless of the operation type, i.e., automatic or manual, that is used to control the instrument, different setups have to be processed. This leads to three very important services that are used in the *Operation Layer*: the *manager* services.

2.4.3 The different managing services

Why does the `ScriptExecutionHandler` need to contact the different managers to commit the setups and not contact the *Instrument Layer* services directly? The answer to this question is the principle of encapsulation of complexity into different objects to make the software more robust, readable, and maintainable. To setup the instrument, telescope or detector is a difficult task that needs careful thinking. We consider the necessary tasks for the `InstrumentManager` first. Most of the work on the `InstrumentManager`, including the creation of the user interface was done by Dr. M. Jütte. The implementation of the flexure compensation mechanism was done in the scope of this thesis, however.

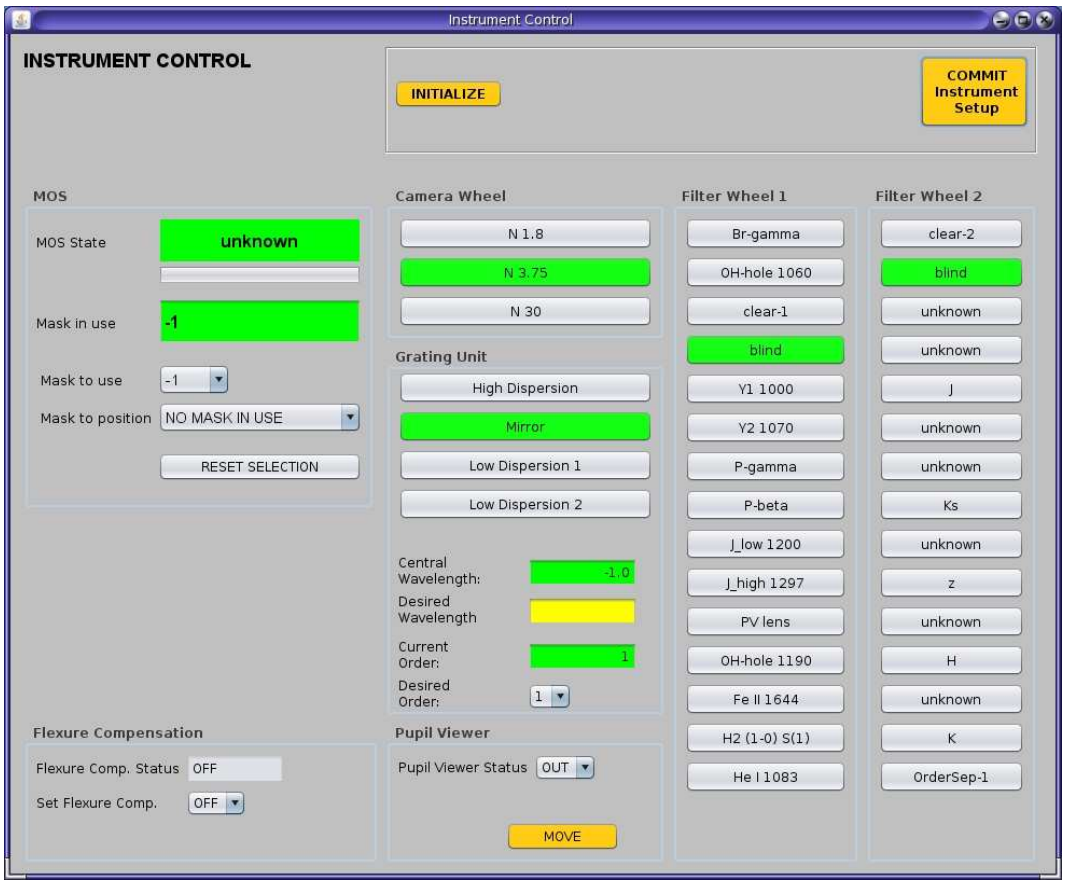


Figure 2.40: The `InstrumentManager` GUI.

InstrumentManager

Since the services of the *Instrument Layer* need information from other services of this layer to solve their tasks correctly, they have to get this information somehow. One possibility would be to let the services communicate with each other directly. This approach is however very error prone in the case of a failing service that is needed by a couple of other services. One example for such a service is the `CameraUnit` service. The information about the currently selected camera is needed by the `DetectorFocus` service and the two services controlling the *Alignment*– and *Compensation Mirror*. For the former service, the focus position obviously depends on the selected camera (and on the selected filters), for the latter two services, the movement of the image on the detector will also vary for different camera optics. The `GratingUnit` service is not affected, since the central wavelength on the detector should not change with the camera, only the usable wavelength range.

If the `CameraUnit` service were to be deactivated or become inoperable for some reason, i.e., a hardware failure of the camera wheel, and the dependent services would get their information directly from the `CameraUnit` service, they would be inoperable as well. The consequence could be that the whole instrument would become unusable, while in principle it could still be used, assuming that one camera is still in the optical path. If the `InstrumentManager` is responsible, on the other side, this is not a problem. The other services can be used as usual, only the selection of a new camera would be prevented.

Another great benefit to avoid direct communication between the services of the *Instrument Layer* is the arising complexity of information exchange. Again for the example of the `CameraUnit`, this means that this service needs to inform the depending services about a camera change. This would introduce a level of complexity for the software service, that is not needed for the operation of the physical unit. The coordination of the instrument is much easier if it is managed by one central instance instead of many independent services. As a final benefit the current setup of the instrument is available at one central place.

When the `InstrumentManager` gets a setup from the `ScriptExecutionHandler`, it configures the units that need to change according to the information specified in the setup. Furthermore, it updates the services that depend on setups of other services, like the `CompensationMirror` or the `DetectorFocus`, with the new information. When the `InstrumentManager` has successfully set up the instrument, a notification is sent to the `ScriptExecutionHandler`.

If the flexure compensation is switched on, information about the current elevation angle of the telescope is needed also. To get this information, the `TelescopeManager` must be contacted, so communication between the three manager services is necessary. The current information is queried in regular intervals for the current values which are subsequently send to the `CompensationMirror` service. When no exposure is currently taken the mirror is adjusted to compensate any instrument flexure.

Fig. 2.40 shows the main observer GUI for the LUCIFER instrument. This user interface communicates with the `InstrumentManager` in the same way the `ScriptExecutionHandler` does. By selecting the elements represented by the buttons the astronomer creates a new `InstrumentSetup` which is send to the instrument when "Commit Instrument Setup" is clicked. As can nicely been seen, all the complexity of the instrument is hidden from the astronomer. Only the necessary information is shown and asked. The tilt of the *Grating Unit* can only be specified by the central wavelength, and not by the tilt voltage, for example, since the astronomer does not need to know this detail. When changes are made to the instrument via any other source, e.g., observation scripts, the GUI is updated automatically. It may therefore also be used to check the correct execution of scripts.

TelescopeManager

The telescope control package was implemented by Dr. M. Jütte. The following description gives a short summary of the interaction between the instrument and the telescope and has been added to this chapter for completeness.

The main task of the `TelescopeManager` is of course the control of the telescope and the processing of the `TelescopeSetups` that it gets from the `ScriptExecutionHandler`. To do this, the `Telescope` service of the *Control Layer* is used which communicates with the TCS using an ICE interface provided by J. Borelli who is working at Max-Planck-Institut für Astronomie, Heidelberg. The setup includes pointing and tracking information for the selected astronomical target, as well as the desired telescope mode, i.e., if active or adaptive optics are to be used. The current status values of the telescope, like the different coordinates — right ascension, declination, azimuth and elevation — have to be queried in regular and small time intervals to provide the necessary information to the `Instrument Manager` for flexure compensation.

The main observer interface for the `TelescopeManager` is shown in Fig. 2.41. The basic layout is similar to that of the `InstrumentManager` GUI shown in Fig. 2.40. The top right button commits a new `TelescopeSetup` to the telescope. The middle section shows current status data for the telescope, while the lower part of the GUI allows to set new values for the next setup.

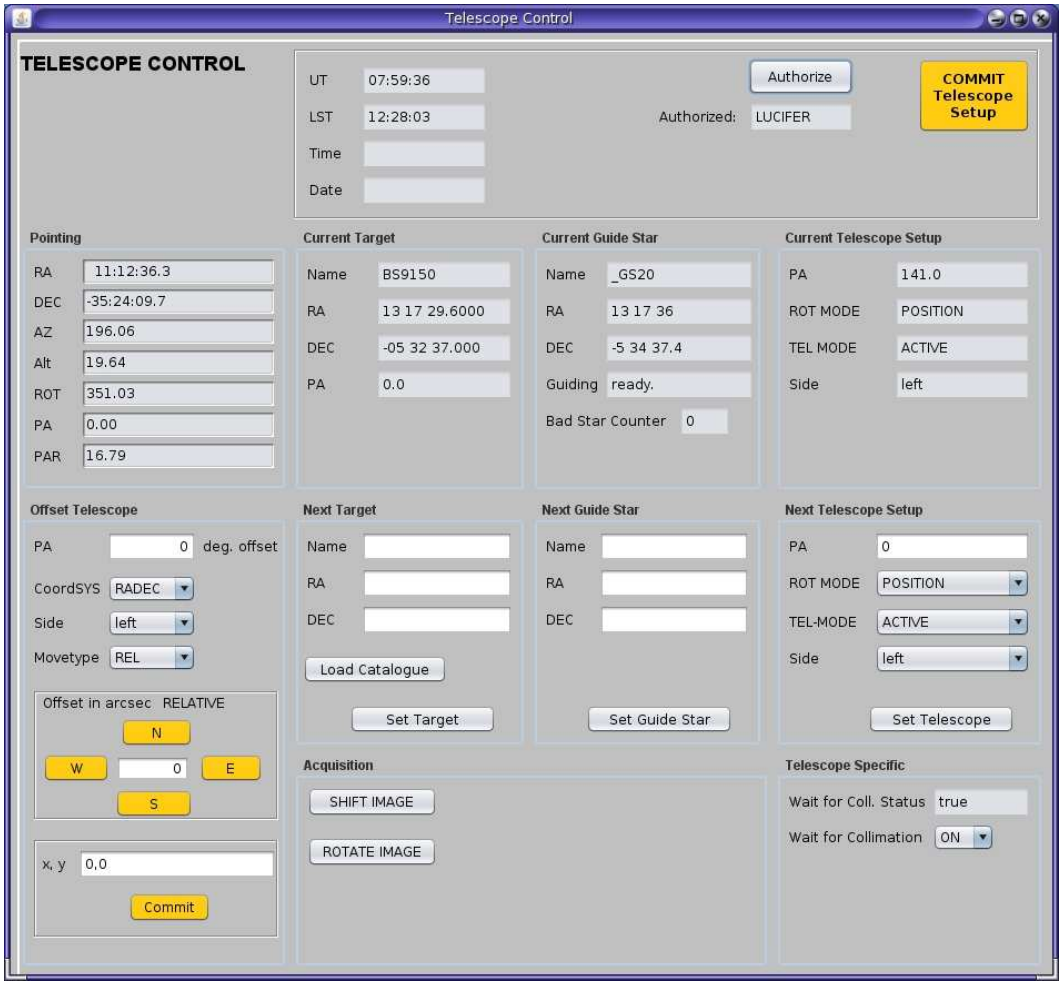


Figure 2.41: The `TelescopeManager` GUI.

These values will be sent to the `TelescopeManager` when the "Commit Telescope Setup" button is pressed. An exception is the lower left area of the interface which lets the user send small offsets directly to the telescope. After more experience had been gained during the commissioning, this functionality has been proven necessary.

ReadoutManager

The last of the three manager services of the *Operation Layer* is the `ReadoutManager`. It processes `ReadoutSetups` with integration time and readout mode information, that it gets from the `ScriptExecutionHandler`. The manager communicates with the *GEIRS* software, developed at the Max-Planck-Institut für Astronomie, Heidelberg, using the `readout` package (see Section 2.2.2). One important extension over the *Control Layer* service, that the `ReadoutManager` has to provide, is the functionality to synchronize readout cycles with setup changes of the instrument and the telescope. This synchronization has to work in two directions, i.e., no exposure should be started when any setup changes are made to the instrument and telescope on the one hand, while no setup changes of the telescope or instrument should start during an exposure on

Figure 2.42: The `ReadoutManager` GUI.

the other hand. Normally this can be managed by the `ScriptExecutionHandler`, by making any changes before the exposure is started. If the flexure compensation is turned on, however, the `Instrument Manager` needs to know, if an exposure is taken before it adjusts the mirror. It can therefore request a pausing of the readout as soon as possible, i.e., between two subsequent reads. When the `Instrument Manager` has finished the movement of the mirror, it informs the `Readout Manager` that the next exposure can be taken.

Like the other two manager services introduced before, the `ReadoutManager` can be controlled manually by observers. The user interface is shown in Fig. 2.42. The left region of the GUI shows the current status of the manager, indicating if an exposure is currently taken by a countdown clock. The right part allows active control of the service by setting new values and starting and stopping exposures. As with the other manager services, setup changes done by scripts are displayed automatically, so the GUI can be used to check the correct processing of the observation scripts.

2.5 Summary and Outlook

In this chapter the software package for the LUCIFER instrument was presented. Most of the software is written in *JAVA* as a distributed system using the RMI features that are provided with the language. The software is separated into four different layers, each solving the necessary tasks for changing levels of abstraction. The *Control Layer* (Section 2.2) provides the services for communication with the different hardware units for instrument control and environment measurements. Also located in this layer are the services that connect to external software packages, namely the service for control of the *GEIRS* software and the `Telescope` service for control of the TCS system. The *Instrument Layer* (Section 2.3) represents the software abstraction for the LUCIFER instrument. All movable units of the instrument are modeled by software services like the `GratingUnit` or the `CompensationMirror`. These have been tested and are working very

well and reliably. All services have been implemented and tested extensively, with one exception: the physical unit for the atmospheric dispersion corrector (ADC) has not been built yet, since it is only needed for observations using adaptive optics and these are not scheduled to start before 2010. Subsequently, the `ADC` software service has not been implemented yet. The `MOSUnit` service of the *Instrument Layer* is presented in detail in Polsterer (in prep.). Finally, the *Operation Layer* (Section 2.4) addresses the specific problems that arise for the astronomical observation with the instrument. To allow the automatic operation of the instrument for optimal performance, an `ObservationScriptParser` and a `ScriptExecutionHandler` have been developed. Three manager services, the `Instrument–`, `Telescope–`, and `ReadoutManager`, are used to coordinate the necessary setup steps for observations with LUCIFER. These managers have been developed partly by Dr. M. Jütte (`TelescopeManager` and most parts of the `InstrumentManager`).

Commissioning of LUCIFER 1 at the LBT has started in September 2008. From the beginning the instrument and the control software have been very reliable and performed above expectations. The structure of the different layers of the software, breaking complex tasks down into more simple entities, has been proven to be the correct approach for managing the complexity of the instrument and is working very well in practice. The software provides for very efficient operations of LUCIFER by the use of scripts. These allow a thorough planning of the science observations and a flexible as well as efficient execution. Direct interaction with the main observer GUIs allow quick setup changes also, if needed.

The future work on the software includes the finalization of the *Operation Layer* with the main focus on the full integration of the OPT and `Scheduler` service created by Schimmelmann (2007). These need to be adapted to the current version of the software. The `ADC` service will be implemented by the time of the commissioning of LUCIFER 1 for adaptive optics observations. The completion of the LUCIFER 2 instrument, together with the necessary extensions of the software will be worked on at the same time. Finally, the binocular operation of both instruments at the LBT are scheduled for 2010 and beyond.

Chapter 3

Stellar populations in galaxy centers

The understanding of galaxy formation and evolution is essential for our understanding of the universe. The central parts of spiral galaxies, the bulges, play a vital role in this regard. In the scenario of hierarchical evolution of galaxies the bulges are thought to be older parts of their host galaxies and have formed before the disk. However, in the past decade more and more evidence has been found that this scenario might not be the only possible one. The bulges of spiral galaxies might have formed slowly, i.e., by secular processes. This second scenario is subsequently called secular evolution of galaxy bulges.

The LUCIFER instrument will be able to contribute significantly to this research because of its unique MOS capabilities in the near infrared. In the K–band the absorption features of the CO molecule around 2.3 μm are frequently used to study stellar populations of galaxies. This technique is applied in this thesis to study the central parts of galaxies of different Hubble type. These are expected to show different spectral characteristics in the region of the CO absorption bands.

To prepare the LUCIFER observations NIR spectra have been successfully proposed during this thesis and were taken with a similar instrument, ISAAC at the Very Large Telescope (VLT) of ESO. During the commissioning of LUCIFER, additional long slit spectra have been taken for one of the galaxies that was also observed with ISAAC. The main part of this chapter deals with the analysis and discussion of these spectra.

At the beginning of this chapter an overview over recent studies and findings regarding the (secular) evolution of galaxies is presented. In Section 3.2 a short overview of the particularities of near infrared observations is given. Although there are many similarities with optical observations, some fundamental differences exist. The data that was gathered during this thesis is presented in Section 3.3. The CO absorption strength is measured using the new D_{CO} index defined by Mármol-Queraltó et al. (2008). The results of the analysis are presented and discussed. The chapter closes with a summary of the results and an outlook on how LUCIFER can contribute to NIR astronomy in the future.

3.1 Secular evolution of galaxies

Galaxies exist in various kinds of shapes and sizes. For a better understanding of these objects, some kind of classification is very helpful. A crude distinction based solely on morphology leads to three fundamental classes: elliptical–, spiral–, and irregular galaxies. This was done for the

first time by Hubble (1926). Sandage (1961) provided typical objects for each class. The galaxies were classified as E0 to E7 for the ellipticals, with higher numbers reflecting higher ellipticity. S0 galaxies are an intermediate type between ellipticals and spirals, which were originally classified into Sa to Sc types and SBa to SBc types. The difference between the two spiral types is that the latter show a *bar* — an elongated structure — in their central regions, while the former do not. The letters 'a' to 'c' were used to distinguish between more strongly wound spiral arms ('a') and less strongly wound ones ('c'). The last type, the irregular galaxies, show no apparent global structure. This initial classification has been extended and revised several times since its initial proposal by Hubble to be able to accommodate and distinguish better between transition objects, for example by de Vaucouleurs (1963) and de Vaucouleurs et al. (1991). Sandage (2005) gives a detailed review about the history of galaxy classification. But even with this revised classification some galaxies, e.g., the galaxies of Arp (1966), do not fit well in any of the classes. Other classification schemes for galaxies exist, that are based on other properties of galaxies than morphology, e.g., spectroscopy (Morgan & Mayall 1957) or more recently combining different internal and external galaxy properties into a new "multidimensional" system (Conselice 2006).

From the beginning of galaxy classification the question arose, if the different classes represent different physical properties which could lead to insights of their evolutionary stage. Even though Hubble himself separated between "early" and "late" types, he noted that this terminology was solely based on a sequential, not a temporal, meaning of the objects in the classification. Subsequent studies of galaxies showed fundamental physical differences between the different types, though. On the one hand, most ellipticals were found to be nearly void of cold gas (e.g., Knapp et al. 1978, and references therein), and complementary observations showed that they do not have recent star formation. On the other hand, spiral galaxies contain more gas and are actively forming stars. The fraction of the gas content is rising with the spiral galaxy type. Early types (Sa) have the lowest fraction of gas, while the later types have increasingly more gas (Roberts & Haynes 1994). Kennicutt (1998a) presents a review about differences in star formation with the Hubble type, suggesting fundamental physical differences among the galaxies. The recently found bimodal separation of global galaxy colors, grouping them into "blue cloud" (mostly late type spirals and irregular galaxies) and "red sequence" (mostly early type spirals and ellipticals) companions, is another indication for a physically motivated separation between galaxy types (Driver et al. 2006).

Indications for transitions from one galaxy type to another were observed as well. Dressler (1980) formulated the morphology density relation (MDR), finding that the number of elliptical and S0 galaxies increases in the centers of galaxy clusters, while spiral galaxies decrease in number for higher density regions. By looking at more distant clusters at redshift $z\sim0.5$ Dressler et al. (1997), and later Smith et al. (2005) and Postman et al. (2005) for $z\sim1$ found that the MDR changed from the present day numbers. Higher z clusters show a lack of S0s and an increase in spirals, supporting the picture of the transition, or evolution, from spiral galaxies to S0 and elliptical types. This has recently been questioned by van der Wel et al. (2007) and Holden et al. (2007), however. These newer studies see no significant evolution of the MDR since $z\sim0.8$, at least for the higher mass galaxies.

To explain the different galaxy types and the formation and evolution of them, different models were proposed and the two most prominent are examined in more detail next.

3.1.1 Structure formation in the universe

Eggen et al. (1962) suggested the rapid collapse of a protogalactic cloud as a formation scenario

for galaxies. This idea was extended by Gott & Thuan (1976), who proposed different evolution scenarios for spirals and ellipticals to explain their various characteristics. In this picture, ellipticals originated in denser environments of the universe and used up all of their gas in the initial collapse that formed these galaxies. Spirals form in less dense environments and only a smaller fraction of the initial gas is converted to stars which subsequently form the bulge of the spiral galaxy. The remaining gas is rearranged and flattened to form the disk of the galaxy.

Another scenario is examined by Baugh et al. (1996). In this model, galaxies evolve to different types more than once. Star formation is induced in the disks of spiral galaxies, which frequently undergo merger events with other galaxies. After such a merger event an elliptical galaxy is formed (Barnes 1992; Barnes & Hernquist 1996). Onward accretion of surrounding gas by the newly formed elliptical can create a new disk (Springel & Hernquist 2005). The central elliptical part becomes the bulge of the new spiral galaxy. This process can occur multiple times, in which case the galaxy changes its type from spiral to elliptical types back and forth. The largest elliptical galaxies underwent the highest number of merger events in this scenario. These results have recently been confirmed by the study of De Lucia et al. (2006), who estimate the most massive ellipticals to have approximately five progenitors, whereas the less massive elliptic galaxies with $M_\star < 10^{11}\,M_\odot$ only have two. Naab & Ostriker (2009) put more strict constraints on the formation of the most massive ellipticals. According to their study, these cannot form in a single merger event involving only two spiral galaxies. Instead, mergers of multiple progenitors are necessary to form a giant present day elliptical.

These two evolution scenarios can explain the existence of two of the main galaxy types, ellipticals and spiral galaxies. Two main constituents of spiral galaxies are present by both theories as well:

1. The *spiral disk* is the decisive place for star formation in the model suggested by Baugh et al. (1996), implying that a primordial gas cloud collapses to a disk before significant star formation starts. Gott & Thuan (1976) propose star formation earlier on, if the gas cloud is dense enough and the collapse times τ_c are short. As mentioned before this process creates the elliptical galaxies. A disk forms mainly in less dense environments where τ_c is larger and only a smaller fraction of the gas cloud is turned into stars in the initial collapse.

2. The central *bulge* — seen edge–on as a more or less spheroidal component, face–on as a central light concentration of spiral galaxies — is produced by mergers of galaxies (Baugh et al. 1996) or by the collapse of the protogalaxy itself (Gott & Thuan 1976).

In both scenarios the evolution of galaxies is mainly triggered by *external* processes, i.e., the encounter of two galaxies and subsequent merging and interaction processes, or by the surrounding environment. Difficulties are present in both models, however:

- The nature of irregular galaxies is not well understood in both models.

- It remains unclear why some spiral galaxies show a central bar, (Fig. 3.1), while other galaxies do not (Fig. 3.2).

- The bulges of spiral galaxies differ greatly in their sizes and morphologies. Some spirals have no or very small bulges (Kormendy 2008) while the bulges of other galaxies have peculiar shapes that are far from spherical. Instead these bulges appear more box/peanut (b/p)

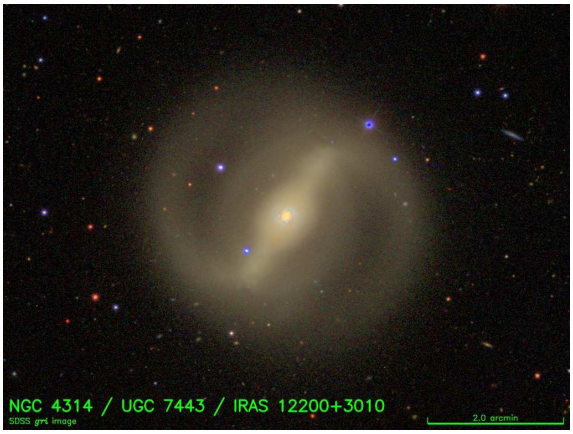


Figure 3.1: SDSS image of NGC 4314, a spiral galaxy with a strong bar.

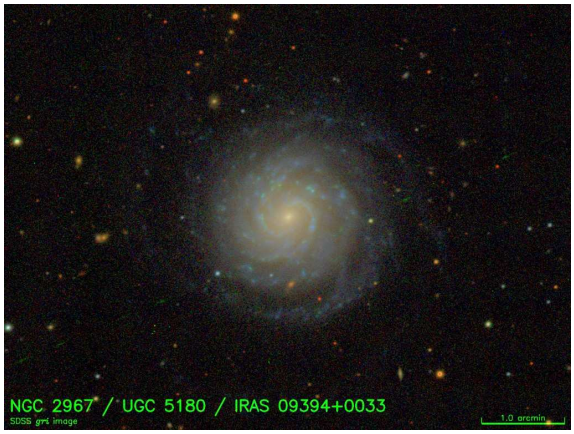


Figure 3.2: SDSS image of NGC 2967, an unbarred spiral galaxy.

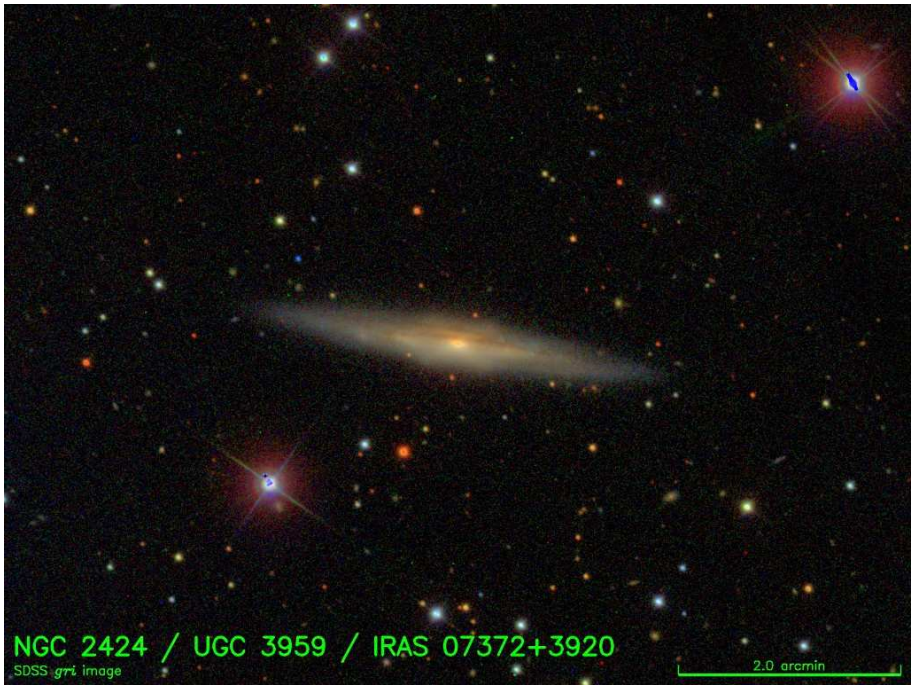


Figure 3.3: SDSS image of the galaxy NGC 2424, which shows the peanut (looking like a ⋈) bulge nicely.

shaped (Lütticke et al. 2000a,b, 2004). An example of a peanut shaped bulge is shown in Fig. 3.3[1].

With regard to this last point, Kormendy (1982, 1993), noticed that the bulges of some galaxies show high rotational velocities and flat profiles that are similar to those of galaxy disks (see next section). Flat profiles of the bulges would mean that they might have formed by different processes

[1]Figs. 3.1, 3.2 and 3.3 are three color composites of images taken with the 'g', 'r', and 'i' filters reproduced with kind permission by David W. Hogg, Michael Blanton, and the Sloan Digital Sky Survey (York et al. 2000).

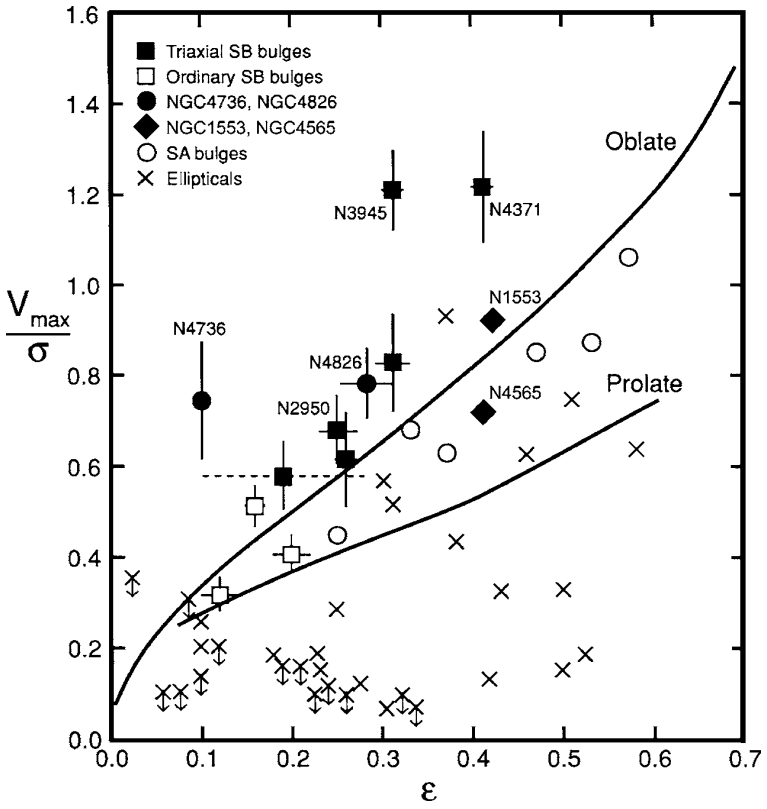


Figure 3.4: $V_{max}/\sigma - \epsilon$ diagram from Kormendy & Kennicutt (2004). Objects above the "oblate" line are dominated by rotation. See text for details.

than the collapse or merger events mentioned above. This implies that the structure formation in galaxies is not only triggered by external properties, i.e., the surrounding environment, but also by internal processes. This slow — compared to typical collapse timescales — "modification" of galaxies by purely internal mechanisms is called *secular evolution*.

3.1.2 Classical bulges *vs.* pseudo bulges

Athanassoula & Martínez-Valpuesta (2008) discuss three different definitions for a galaxy bulge: photometric, morphological and kinematic. The photometric one defines the bulge as the "extra light" around the galaxy's center that remains when the disk brightness profile is modeled and subsequently subtracted from the total brightness profile of the spiral galaxy. For some bulges this inner brightness profile is found to follow that for elliptical galaxies — the famous $r^{1/4}$ law by de Vaucouleurs (1948) — indicating a similar origin and connection between bulges and elliptical galaxies. Using a generalized form of the law $\mu \propto r^{1/n}$ defined by Sersic (1968), Andredakis et al. (1995) find that the profile of bulges are better approximated using different *Sersic indices* $n \leq 2$ for the late type spirals and $n \geq 2$ for earlier types. The morphological classification defines the bulge as the extended structure that is visible above the disk of a spiral galaxy seen edge–on. The kinematic definition compares the maximum radial velocity V_{max} to the velocity dispersion σ. The dominance of rotation is analyzed by a plot of V_{max}/σ versus the ellipticity ϵ of the objects (Binney 1978). This plot is shown in Fig. 3.4, taken from Kormendy & Kennicutt (2004). A recent version

of this plot can be found in Erwin (2008). Larger values of V_{max}/σ for a given ϵ indicate stronger rotation. Elliptical galaxies lie below a line which characterizes "oblate" objects. This line takes the flattening of objects due to rotation into account, leading to higher ϵ values. Bulges of early type spirals follow this line closely, as well as medium mass ellipticals (Kormendy 1982; Kormendy & Illingworth 1982; Illingworth & Schechter 1982). Bulges of later type spirals are found above this line, indicating a more disky structure. Looking at the famous velocity dispersion for elliptical galaxies by Faber & Jackson (1976), it is found that bulges of early type spirals (Sa) follow this relation well, while those for late type spirals do not (Kormendy & Kennicutt 2004).

From the discussion above one can draw the conclusion that all three definitions for bulges do not describe an unique class of object. This has lead to the terminology *classical bulge* for those that do share the properties of elliptical galaxies, and *pseudo bulge* (Kormendy 1993) for those objects that do not. As more and more evidence has been found that the pseudo bulges themselves do not represent a single class of objects, Athanassoula (2005) divides the pseudo bulges into two categories, b/p bulges and disky bulges. Thus, three types of bulges can be distinguished.

Classical Bulges

Classical bulges are more common in early type spiral galaxies and are similar to elliptical galaxies in their properties. They are characterized by spheroidal or triaxial shapes, and Sersic indices $n \geq 2$ (Andredakis et al. 1995). Kinematically they are dominated by random motions and follow the "oblate" line in the V/σ *vs.* ϵ relation (Erwin 2008; Kormendy & Kennicutt 2004). Drory & Fisher (2007) show that galaxies belonging to the "red sequence" (Strateva et al. 2001; Balogh et al. 2004; Baldry et al. 2004; Driver et al. 2006) usually have classical bulges, while those of the "blue cloud" have pseudobulges. The many similarities of classical bulges with elliptical galaxies suggest a connatural formation scenario. As outlined in Section 3.1.1 this could be merger events between two galaxies or the collapse of the protogalactic gas cloud that led to the first star formation. Thomas & Davies (2006) investigate the ages of elliptical galaxies and spiral bulges. Their results show that the age is related stronger to the mass of the object than to the type of galaxy. Massive elliptical galaxies are old, while lower mass elliptical and bulges have younger ages around 2-3 Gyrs. Ellipticals and bulges are indistinguishable if they have comparable masses, suggesting that the bulges are not significantly affected by the presence of a disk. From their sample, Thomas & Davies (2006) conclude that the secular evolution of bulges is only dominant for spiral galaxies later than Sc. However, Gadotti (2009) finds offsets in the mass–size relations of ellipticals and classical bulges that might suggest different formation scenarios and question the picture that classical bulges are small ellipticals surrounded by spiral disks.

b/p Bulges

The works of Bureau & Freeman (1999), Lütticke et al. (2000b), Aronica et al. (2003) and Athanassoula (2005) have shown that one class of pseudo bulges, the b/p shaped ones, are in fact parts of galaxy bars seen edge–on. Using N-body simulations and bar orbit theories (see for example Englmaier & Gerhard 1997), Patsis et al. (2002), Athanassoula (2003), Debattista et al. (2006), and others can explain the shape of b/p bulges by instabilities in stellar orbits. These x_1 orbits are very elongated and form the bar (Figs. 2, 5, and 8 in Englmaier & Gerhard 1997). The instabilities raise material in the z direction, out of the disk, causing the characteristic $\bowtie$ shaped bulge (Patsis et al. 2002). Athanassoula (2008) gives an overview of the time evolution of b/p bulges, describing that the vertical structures that form the characteristic shape appear after the

bar has had a maximum strength and is weakening again. Once formed, the b/p gets stronger with time by secular evolution processes. It should be mentioned that once the structure has been formed it does not fade away over the time of the simulations (12 Gyrs).

Disky Bulges

Disky bulges are the other class of pseudo bulges and are extensively reviewed in Kormendy & Kennicutt (2004). They form by transport of gas into the centers of galaxies, leading to significant star formation there, since the surface density of the gas Σ_{gas} (in $M_\odot\,pc^{-2}$) and the star formation rate (SFR) Σ_{SFR} (in $M_\odot\,year^{-1}\,kpc^{-2}$) are connected by the *Schmidt law* $\Sigma_{SFR} \propto {\Sigma_{gas}}^{1.4}$ (Schmidt 1959; Kennicutt 1998b). The gas transported into the center is often arranged in rings (Kormendy 2008, and references therein). Nuclear bars and spirals are other structures found in these disky bulges. The SFR resulting from the gas transport is comparable to that of the host galaxy (Fisher 2006). The brightness profile of the host galaxies shows a rise in the center, which is better approximated by lower Sersic indices of $n \leq 2$ (Fisher & Drory 2008; Kormendy & Kennicutt 2004), but this is not a strict criterion for disky bulges.

Gadotti (2009) analyzes a large data set from the Sloan Digital Sky Survey (SDSS) performing a decomposition of the brightness profiles into bulge, bar and disk components. His results show that disky bulges contribute 3% to the stellar content of the local universe, while 25% are contributed by classical bulges. The former are less concentrated than the latter and currently undergoing intense star formation. The same trend is indicated by Peletier et al. (2008), who used the Spectroscopic Areal Unit for Research on Optical Nebulae (SAURON) instrument (Bacon et al. 2001) to study the stellar populations of the central regions of 42 spiral galaxies. They find a significant trend towards higher but quiescent star formation for later types. The centers of earlier types show less star formation in general. However, starbursts are more common for these objects. Combining their data with that of Kuntschner et al. (2006) for ellipticals and S0s reveals a global trend. The earliest type galaxies show a low SFR, which is increasing towards later types. S0's and early type spiral galaxies can show a starburst, however, raising their SFR above those of the latest types. Hathi et al. (2009) have analyzed the stellar ages of 34 bulges of late type galaxies in the Hubble Ultra Deep Field (HUDF). They find that these bulges contain significantly younger stars than elliptical galaxies. They also do not see a significant trend between the stellar ages and redshift, interpreting the results with a much longer star formation time in these bulges compared to early type galaxies.

The differentiation between classical and pseudo bulges is not exclusive, however. Erwin (2008) shows examples of galaxies that have clear kinematic and photometric indications for a disky bulge *as well as* for a classical one. The same conclusions are drawn by Peletier et al. (2008) for two galaxies they studied. Thus classical and pseudo bulges might exist simultaneously in a galaxy, maybe reflecting different evolutionary phases of it. This scenario is also supported by MacArthur et al. (2009) in their study of radially resolved long slit spectra of star forming spiral galaxies. They conclude that although secular evolution effects lead to a young stellar population of the bulges which contributes for up to 70% of the optical light, only 20% of the total mass of the bulge are represented by this young population. Thus, 80% of the mass of these bulges resides in older stars.

Concluding this section it can be said that no formation and evolution scenario for galaxies, either driven mainly by collapse or by mergers, is completely ruled out by current observations. However, in the last couple of years many evidence has been found that the evolution of spiral galaxies is not only affected by external but also significantly by slower secular processes. This

evolution severely affects the centers of spiral galaxies. The b/p shaped bulges are well understood as the inner parts of galaxy bars seen edge–on. Material is lifted out of the plane of the galaxy by perturbations and instabilities of elongated orbits. Disky bulges are mainly seen in late type spiral galaxies and can be distinguished from classical spheroidal bulges using the following indicators (Kormendy & Kennicutt 2004). Disky bulges:

- consist primarily of young stars, gas and dust with no indication for a merger,
- show a nuclear spiral, bar or ring,
- have a low Sersic index n ≤ 2,
- are more rotation dominated than classical bulges in the $V/\sigma - \epsilon$ diagram and
- do not fit well in the Faber & Jackson (1976) diagram.

3.2 Properties of the NIR regime

Although the near infrared wavelengths are adjacent to optical wavelengths — this is the reason why optical telescopes can also be used for infrared observations — the observational techniques differ considerably from their optical counterparts. The main reason is the influence of the earths atmosphere, which is significantly larger in the near infrared than in the optical. Another reason for deviations between the optical and NIR observations is of technical nature, since the detectors used for the latter differ from the ones used for the former.

3.2.1 Influence of the earths atmosphere

The atmosphere of the earth has several influences on the information that reaches us from distant sources. These influences can be grouped into different categories. Léna (1988) distinguishes the following distortions:

1. absorption,
2. emission,
3. scattering, and
4. turbulence.

Absorption

The atmosphere of the earth is transparent for electromagnetic radiation in certain windows only (see Fig. 1.6). The absorption depends on different mechanisms for different frequencies. For example, for wavelengths smaller than 300 nm electronic transitions of O_3 and O_2 as well as the ionization of O_2 are the main cause for absorption, while in the infrared (up to 100 μm) rotational–vibrational absorption of the H_2O and CO_2 molecules are dominant (Léna 1988). For the NIR K–band the telluric absorption of water vapor is the prevalent source for absorption of the photons. A plot of the measured transmission of the atmosphere in the K–band is shown in Fig 3.5. The data was taken at the Kitt Peak Observatory in Arizona and is made available from the ESO[2].

[2] http://www.eso.org/sci/facilities/paranal/instruments/isaac/tools/spectroscopic_standards.html

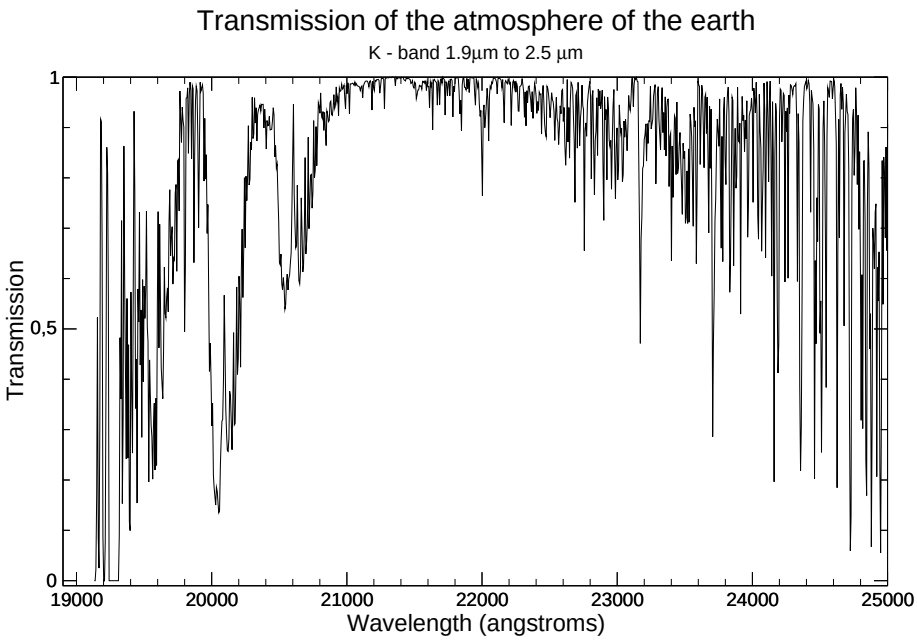


Figure 3.5: The transmission of the atmosphere over Kitt Peak between 1.9 μm and 2.5 μm. The data was provided on the ESO ISAAC website.

Since the column height of water vapor has a significant impact on the absorption, the measured transmission values for Kitt Peak, which is considerably lower than Mt. Graham (2070 m compared to 3220 m), will deviate quantitatively from those for the LBT. However the figure gives a good impression about the main telluric features present in this wavelength range, e.g., the two broad absorption features between 20000 Åand 21000 Å. To correct for telluric absorption features in infrared spectra, stars with well known energy distribution — telluric standards — are observed. More information about the reduction process is given in Section 3.3.

Emission

The atmosphere also emits radiation at various wavelengths. This radiation is composed of discrete lines as well as of continuum emission. The line emission, or airglow, consists of many constituents. For the near infrared the dominant source of emission line radiation are the OH radicals in the stratosphere (Léna 1988). This emission varies significantly over scales $\geq 1'$ as well as on time scales in the order of minutes.

The continuum emission of the sky is mainly caused by thermal emission of the atmosphere itself, since it is in local thermodynamical equilibrium up to heights of 60 km. Above this altitude collisions between molecules become too infrequent (Léna 1988).
Using *Wien's law*:

$$\lambda_{\mathrm{max}} \times \mathrm{T} \approx 2.90 \times 10^{-3}\,\mathrm{m\,K} \tag{3.1}$$

the peak of the emission of a black body with a temperature of $\mathrm{T} = 250\,\mathrm{K}$ (taken as the average temperature of the atmosphere) lies at a wavelength $\lambda_{\mathrm{max}} \approx 11.6\,\mu$m. Due to fluctuations in the atmosphere the temporal variability the thermal emission comparable to the variability of the line emission. Spatial variations are caused by the increasing line of sight through the atmosphere for growing zenith distances.

Because of the variability of the sky brightness, sky subtraction is a difficult but vital task when NIR observations are performed. Sky frames have to be taken frequently before and after the science exposures, if the sky brightness cannot be calculated from the science frames directly, i.e., for extended objects. Since the thermal emission of the atmosphere becomes more important for longer wavelength, in the K–band around 2.3 μm the observations begin to be background dominated and allow only short integration times. For shorter wavelength in the z– and J–bands (0.9 μm and 1.2 μm, respectively) line emission is more dominant.

Scattering

The light that enters the atmosphere is scattered by molecules and aerosols. For molecules, Léna (1988) gives the scattering cross–section σ_R due to *Rayleigh scattering* as

$$\sigma_{\mathrm{R}}(\lambda) = \frac{32\,\pi^3}{3\mathrm{N}^2}\,\frac{(\mathrm{n}-1)^2}{\lambda^4}\,. \tag{3.2}$$

Here, N represents the molecular number density and n the refractive index. The cross–section is highly dependent on the wavelength of the incident radiation ($\sigma_{\mathrm{R}}(\lambda) \propto \lambda^{-4}$), with smaller wavelength being much more affected. This is the reason for the blue color of the daytime sky. *Rayleigh scattering* is not isotropic but has a $1 + \cos^2(\theta)$ dependence, θ representing the angle between the incident and scattered light. Aerosols are larger than molecules, so the approximation that the scattering particles are small compared to the wavelength of the photons used for *Rayleigh scattering* is no longer valid. For this case the scattering cross section has to be calculated using *Mie's theory* (Mie 1908).

When the intensity of the scattered sunlight becomes comparable to the emission of the atmosphere, astronomical observations are no longer limited by the daylight. This condition is valid for sub–millimeter and radio wavelengths, so these telescopes can be used independent of the time of day.

Turbulence

The atmosphere is not static and has perturbations with scales ranging from low and high pressure zones (cyclones and anticyclones) with several hundred kilometers in size down to a few millimeters for distortions due to friction between individual air "cells". Temporal variations can also be as fast as milliseconds. The turbulence is caused by different effects, i.e., wind blowing over surface irregularities, or shear between two laminar flows of air. Additionally the refractive index of air changes with temperature and H_2O variations (Léna 1988).

All of these effects result in a blurring and twinkling of the stars. This is also called "seeing" and is severely limiting the spatial resolution of astronomical images. Even for the best telescope sites, the seeing is usually no better than 300 milliarcsecond (mas). Using the well known formula

$$\alpha = \frac{1.22\,\lambda}{\mathrm{D}} \tag{3.3}$$

for the minimum angle α (in radians) under which the Airy disks of two stars can be distinguished for a circular aperture of size D at a given wavelength λ, 300 mas correspond to a telescope with D = 1.6 m at λ = 2.0 μm. Thus large telescopes are limited by atmospheric turbulence unless some technical solution — adaptive optics (AO) — is used.

3.2.2 NIR observations

Apart from effects of the atmosphere, which cause differences for NIR observations compared with optical astronomy, the detectors used to collect the information received by the telescope are also different. This leads to altered observation strategies which are discussed briefly in this section.

NIR detectors

The sensitivity of CCDs is rapidly decreasing in the NIR, with a limit at $\lambda \approx 1\,\mu$m (Kitchin 2008). For longer wavelengths photoconductive cells are used. The principle detection process of this detector type is the change in conductivity of a semiconductor with varying illumination. The difference of the conductivity is caused by electrons that are lifted from the valence band into the conduction band by the inner photo effect. The conductivity is then measured using a small bias current. Although many different semiconductors exist that are sensitive in the near infrared, see table 1.4 in Kitchin (2008) for example, Mercury–Cadmium–Telluride (HgCdTe) is common, and this material is used for the HAWAII2 detector of LUCIFER as well. Unlike CCDs every pixel of the HAWAII2 detector can be read out independently and has its own wiring circuit. This implies several differences between the readout of the LUCIFER detector and CCDs. Since pixel can be read out non destructively, several reads can be performed on short timescales at the beginning of the integration and again at the end. The resulting image is computed by subtracting the values read at the end from those read at the beginning and averaging these differences over the number of reads. This readout mode is called multiple end point read (mer). Another possible way to get an image from the detector is to reset the chip first and to perform one read at the beginning of the exposure and one at the end. The resulting signal is calculated by the difference between the two reads. This mode is called double correlated read (dcr) because the two reads are correlated to the same reset level (C. Storz, priv. correspondence). Since only two reads are made, the readout noise is higher in the dcr mode than in the mer mode. Therefore, dcr is used with a high signal to noise ratio (S/N), i.e., for imaging. Muhlack (2006) gives a detailed description of the other possible readout modes for the HAWAII2 detector used in LUCIFER.

Since the pixel are read out individually, using a shutter is usually not necessary for NIR instruments. The detector is constantly illuminated unless blind filters are used. Therefore, a minimum detector integration time (DIT) exists because not all pixel of the detector can be read out simultaneously. This minimum DIT depends on details of the readout electronics, i.e., the number and clocking of readout channels, but is usually in the order of a couple of seconds for the 2048^2 pixel array. For LUCIFER the smallest minimum DIT$\approx$2 s for the dcr mode. A secondary effect of the limited readout speed is that the signal in different regions of the chip is detected at different times. For the chip used inside LUCIFER which has four quadrants with 1024×1024 pixel, this means that row 1024 is read the minimum DIT seconds after the first row of the quadrant, for example. This implies that the flux of row 1024 has been collected the minimum DIT time *after* the flux in the first row. Fig. 3.6 illustrates the layout of detector readout for the HAWAII2 chip used in LUCIFER (Ligori 2004).

Because of the larger thermal noise in the infrared, the detectors need to be cooled down more than CCDs usually need to. For the NIR temperatures of 70 K are usually sufficient, for longer wavelengths temperatures of liquid helium (4 K) or even down to several hundred mK are necessary to reduce the thermal noise of the detector.

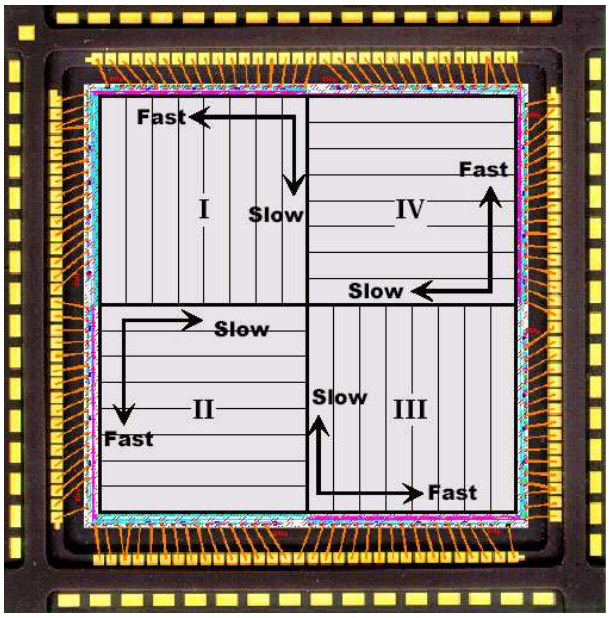


Figure 3.6: This figure illustrates the detector readout quadrants for the HAWAII2 detector used in LUCIFER. The figure was taken from Ligori (2004).

Observation strategies

The high variability of the sky background in the NIR forces the observations to account for this problem. The sky has to be measured as frequently as it typically varies, which is in the order of minutes. Many different methods exist to approach this problem, but only the most common ones will be presented here. A recent summary of the different observing techniques for ISAAC at ESO can be found in Mason & Schmidtobreick (2009), but the best strategy clearly depends on the unique characteristics of the used instrument.

To reduce the overheads, it is desirable to be able to determine the sky from the science exposures. To achieve this, a technique called "jittering" is frequently used for imaging. The telescope is offset by small amounts many times on the science target. By combining adjacent exposures with suitable removal of the stars, a calibration frame can be constructed and subsequently used to subtract the sky. This is of course only possible, if there is "enough" sky in the images. For extended sources the sky has to be constructed from a nearby region, that is relatively clean, i.e., void of stars. Although this introduces a huge overhead — the exposure time of the sky and the object have to be the same — it is sometimes the only possible option. The fact that for observations in the K–band at 2.3 μm the thermal emission of the sky saturates the detectors within a couple of seconds increases the overheads even more.

For NIR long slit spectroscopy (LSS) the same strategies can be applied as for imaging. However, the integration times can be longer and are usually limited by the variability of the background, i.e., in the order of minutes. For compact objects like stars and small galaxies a technique called "nodding on the slit" is used. Between two exposures the object is moved to different parts of the slit. The measured sky above and below the object in one exposure can then be used to subtract the sky in the another one. For extended objects that fill most of the slit, separate sky spectra have to be taken. As for imaging, this creates large overheads for the observations.

Taking all this into account it becomes clear that astronomy in the NIR requires well planned observations. It is very inefficient to carry out these observations by manually offsetting the telescope and starting the integrations. The choice of suitable sky positions and/or observation patterns is also crucial for the success of the science program. It is therefore standard for NIR instruments to provide some kind of tool to allow automatic observations. For LUCIFER this has been realized with the implementation of observation scripts, which are described in detail in Section 2.4.2.

3.3 Data Analysis and Results

To address the question of the dominant stellar population in the center of different galaxy types, NIR spectra in the K–band from 2.2μm to 2.4μm are analyzed. The NIR has many benefits over optical wavelengths because the former is much less affected by dust than the latter. Dust is common in the centers of spiral galaxies, particularly late time spiral galaxies seen face–on, which frequently show disky bulges (Section 3.1.2). For galaxies seen edge–on dust absorption is always a problem, since the centers are hidden behind the disk and its dust. To determine the dominant stellar populations, the strong absorption bandheads of the CO molecule (see Kleinmann & Hall 1986) are used. The absorption features of the CO molecule have been widely used for studies of stellar populations. The most recent studies for elliptical galaxies include Davidge et al. (2008), Silva et al. (2008) and Mannucci et al. (2001). Ivanov et al. (2000) and James & Seigar (1999) used the CO feature in their analysis of spiral galaxies. Mármol-Queraltó et al. (2008) gives are more complete list including also other galaxy types.

3.3.1 CO index definitions

Kleinmann & Hall (1986) gave a first definition of the spectroscopic CO index, but the definition has been changed by different authors in the last two decades (Frogel et al. 2001; Puxley et al. 1997, see also Fig. 3.7). For the analysis done in the scope of this thesis a new definition of the CO index introduced by Mármol-Queraltó et al. (2008) is used. This new index definition is less sensitive to wavelength and flux calibration, spectral resolution and S/N of the data. This is important to produce stable index values in the case of a poorly known or calibrated velocity dispersion or radial velocity. Mármol-Queraltó et al. (2008) define the index as a generic discontinuity. This is the ratio between the average fluxes in the continuum and absorption bands. They used the general definition

$$D_{\text{generic}} \equiv \frac{\sum_{i=1}^{n_c} \int_{\lambda_{c,i_1}}^{\lambda_{c,i_2}} F_{c,i}(\lambda) d\lambda}{\sum_{i=0}^{n_c} (\lambda_{c,i_2} - \lambda_{c,i_1})} \Bigg/ \frac{\sum_{i=1}^{n_a} \int_{\lambda_{a,i_1}}^{\lambda_{a,i_2}} F_{a,i}(\lambda) d\lambda}{\sum_{i=0}^{n_a} (\lambda_{a,i_2} - \lambda_{a,i_1})} \tag{3.4}$$

for the index. In the above formula $F_{c,i}(\lambda)$ and $F_{a,i}(\lambda)$ are the fluxes in the continuum and absorption bands, respectively. The total number of absorption bands is given by n_a, the number

Table 3.1: Definition of the continuum and absorption bands used for the D_{CO} index defined in Mármol-Queraltó et al. (2008). See text for details.

Band	λ_{x,i_1} [μm]	λ_{x,i_2} [μm]
continuum 1 (x=c, i=1)	2.246	2.255
continuum 2 (x=c, i=2)	2.271	2.277
absorption 1 (x=a, i=1)	2.288	2.301

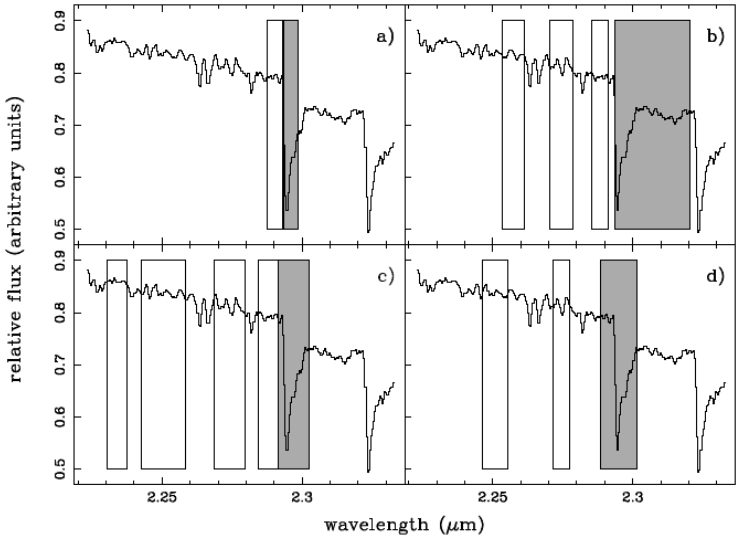


Figure 3.7: The different CO index definitions that are commonly used. Absorption bands are shown in gray, continuum bands in white. a) Kleinmann & Hall (1986), b) Puxley et al. (1997), c) Frogel et al. (2001) and d) Mármol-Queraltó et al. (2008) from which the figure was taken and that is used in this work.

of continuum bands by n_c. λ_{a,i_1} and λ_{a,i_2} define the lower and upper wavelength limits for the i^{th} absorption band (analog for the continuum with subscript 'c'). The index D_{CO} is defined by specifying the absorption bands and continuum bands. According to Table 4 of Mármol-Queraltó et al. (2008), two continuum bands ($n_c = 2$) and one absorption band ($n_a = 1$) are used. The limiting wavelengths for each band are shown in Table 3.1.

With the definition of the generic discontinuity as specified in Equation 3.4 the variance of the measured index is given by (Mármol-Queraltó et al. 2008):

$$\sigma^2[D_{generic}] = \frac{\mathcal{F}_c^2\sigma^2_{\mathcal{F}_a} + \mathcal{F}_a^2\sigma^2_{\mathcal{F}_c}}{\mathcal{F}_a^4}, \tag{3.5}$$

with

$$\mathcal{F}_x \equiv \Theta \frac{\sum_{i=1}^{n_x} \sum_{k=1}^{N^i_{pixel}} F_{x,i}(\lambda_k)}{\sum_{i=1}^{n_x} (\lambda_{x,i_2} - \lambda_{x,i_1})}. \tag{3.6}$$

Thus, $\mathcal{F}_x$ with x='a' or x='c' is the total flux per wavelength unit in the absorption and continuum bands, respectively. In Equation 3.6, Θ is the linear dispersion in Å/pixel, N^i_{pixel} the number of pixel of the i^{th} band and λ_k the central wavelength of the k^{th} pixel. The last missing piece of Equation 3.5 is the variance of the total flux $\mathcal{F}_x$ which is defined by:

$$\sigma^2_{\mathcal{F}_x} = \Theta^2 \frac{\sum_{i=1}^{n_x} \sum_{k=1}^{N^i_{pixel}} \sigma^2_{F_{x,i}}(\lambda_k)}{\left[\sum_{i=1}^{n_x} (\lambda_{x,i_2} - \lambda_{x,i_1})\right]^2}. \tag{3.7}$$

Here, $\sigma^2_{F_{x,i}}(\lambda_k)$ is the random error in the k^{th} pixel. It has been measured as the standard deviation of the fluxes in the continuum bands, which are dominated by the pixel noise, since they have been chosen so that they do not show any absorption or emission lines.

To compute the D_{CO} index, the flux in the two continuum bands is measured and then summed. This value is divided by the width of the continuum bands to get the mean. The result is finally divided by the mean flux measured in the absorption band. Mármol-Queraltó et al. (2008) provide a library of stars with spectral types and corresponding D_{CO} values for field and cluster stars that can be used for a first comparison. Additionally, the authors also provide conversion functions for the other definitions of the CO index, so the data from this work can be compared with results from other publications listed above. For the equivalent width (EW) defined by Puxley et al. (1997) (EW_{Puxley})

$$D_{CO} = 1.0488(\pm 0.0033) + 0.0051(\pm 0.0001) EW_{Puxley} \tag{3.8}$$

and for Frogel et al. (2001) (EW_{Frogel})

$$D_{CO} = 1.0507(\pm 0.0031) + 0.0077(\pm 0.0005) EW_{Frogel} + 0.00007(\pm 0.00002) EW^2_{Frogel}. \tag{3.9}$$

The EWs in the last two equations must be given in Ångstrom. Figs. 3.8, 3.9, and 3.10 show the measured index values for the field stars presented in Mármol-Queraltó et al. (2008) for $[Fe/H] \leq -1$, $-1 < [Fe/H] < 0$ and $[Fe/H] \geq 0$, respectively. The index values are plotted against the luminosity class of the stars. Intermediate class stars have been grouped into the higher luminosity class, with class I taken as the highest. Two global trends are visible. For lower metallicity the D_{CO} values are generally lower than for higher metallicity and within a given metallicity main sequence stars have lower index values than giants. The largest spread in index values can be seen for giants (luminosity class III) and supergiants (class I), but the mean values for these classes are higher than for main sequence stars. These general trends reflect the fact that the CO absorption features in stars are strongest in the spectra of giant and supergiant stars, including asymptotic giant branch (AGB) stars. Their strength increases with

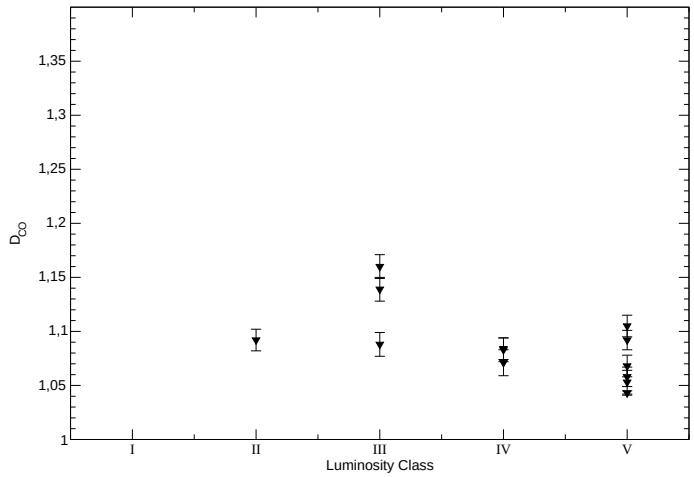


Figure 3.8: The D_{CO} values from Mármol-Queraltó et al. (2008) for stars with [Fe/H] $\leq$ –1. The measured index values are plotted against the luminosity class.

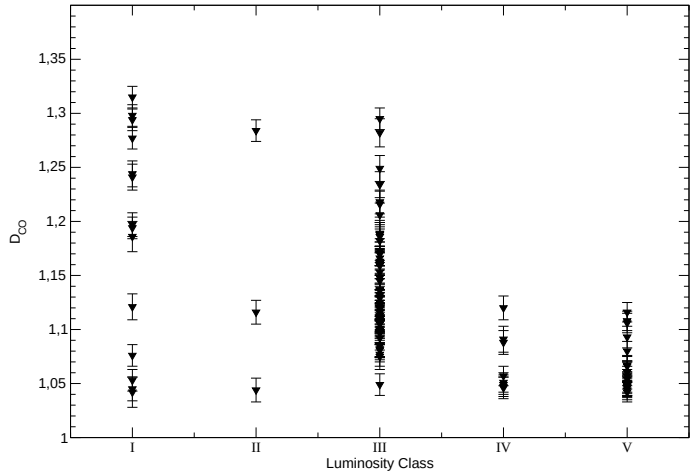


Figure 3.9: Same as Fig. 3.8 but for $-1 < [Fe/H] < 0$.

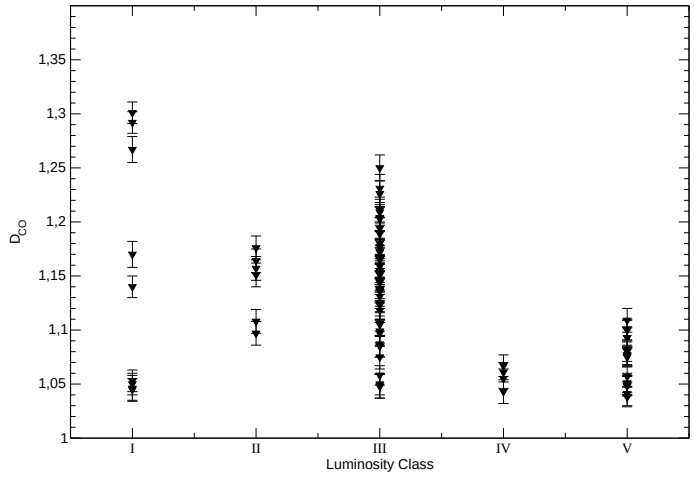


Figure 3.10: Same as Fig. 3.8 but for [Fe/H] ≥ 0.

Table 3.2: Summary of the ISAAC observations of program 078.B-0234(B).

Galaxy Name	Hubble Type	z[a]	Date[b]	Time[c] [s]	Comments
M 95	SB(r)b	0.002595	03.01.07	900	disky bulge
NGC 1032	S0/a	0.008986	27.12.06	2340	edge on classical bulge
NGC 3585	E7/S0	0.004667	05.01.07	600	elliptical galaxy
			17.01.07	600	
NGC 4593	(R)SB(rs)b	0.009000	09.02.07	720	disky bulge

[a]heliocentric
[b]Observation date(s)
[c]Total integration time on target

increasing luminosity, increasing metallicity and decreasing temperature (Mobasher & James 2000, and references therein).

A possible dependency of the higher CO absorption strength, corresponding to higher D_{CO} values, with younger mean ages has been proposed by James & Seigar (1999). They based their assumption on theoretical works by Rhoads (1998) and Origlia et al. (1999), who find an EW — as defined by Puxley et al. (1997) — of $EW_{Puxley} = 5.3 \pm 0.5$ for a young stellar population with $T = 10^7$ yrs (log T = 7). Comparing this value with the results of James & Mobasher (1999), measuring $EW_{Puxley} = 2.8 \pm 0.1$ as the lowest value for isolated elliptical galaxies with pure old stellar populations and high metallicity, a relation between age and CO absorption strength might be justified. Using Equation 3.8, these EWs correspond to D_{CO} =1.192±0.007 and D_{CO} =1.319±0.028 for a predominantly old and young stellar population, respectively. Cesetti et al. (2009) derived a template spectrum of an old elliptical galaxy with a mean age of log T = 9.98 and [Fe/H] = 0.16. The D_{CO} value computed from this spectrum (Fig. D.1) corresponds to D_{CO}=1.163±0.008. These values will be used for a rough comparison with the measured index values of this thesis.

3.3.2 ISAAC observations

The Infrared Spectrometer and Array Camera (ISAAC) instrument is operated by ESO at the VLT on Cerro Paranal in the Atacama desert in Chile. It is a NIR imager and spectrograph, similar to LUCIFER. ISAAC has no MOS capabilities and the detector is only half the size, however. A detailed description of the instrument is given in the operation manual (Mason & Schmidtobreick 2009) available from the ESO homepage[3].

Four galaxies of different types where observed during the observation program 078.B-0234(B). The bulges of M 95 and NGC 4593 have been identified as disky bulges by Jogee et al. (2005) and by Kormendy et al. (2006), respectively. NGC 1032 has a classical bulge (Lütticke et al. 2000a) and NGC 3585 is a late type elliptical galaxy. In the context of secular evolution of galaxies, the first two galaxies and the last two should be similar in their characteristics. The question addressed here is if these similarities are also present in the NIR K–band spectra of the CO absorption features. The data was taken in five nights between December 2006 and February 2007. One galaxy (NGC 3585) was observed in two nights. Details about the observation run are presented in Table 3.2. The observations where carried out in the long slit spectroscopy (LSS)

[3]www.eso.org

Table 3.3: List of telluric standard stars used for ISAAC observations. The spectral type and magnitudes are quoted from data provided by ESO.

Galaxy Name	Telluric Standard	Spectral type	K [mag]
M 95	Hip016792	G2V	7.508
NGC 1032	Hip010449	G1V	7.519
NGC 3585	Hip027727[a]	G1V	7.522
	Hip056398[b]	G1V	7.894
NGC 4593	Hip064878	G3V	7.234

[a]05.01.2007
[b]17.01.2007

mode with the 0.″6 slit and at a central wavelength of 2.2μm. This setup covered the complete wavelength range of the K–band.

Data reduction

The data was reduced following the recommendations given in the "Data Reduction Cookbook" for ISAAC (Mason 2007). The Image Reduction and Analysis Facility (IRAF) package was used to reduce the data, with the exception of the initial correction for electrical ghosts. These were corrected using the `ghost` recipe provided by ESO for the `ESO-eclipse` data reduction pipeline. This correction removes the crosstalk between two lines — 512 rows apart — that occurs during the readout of the chip. After ghost removal, the data were corrected for the dark current and flatfielded using the master calibration frames provided by the ISAAC data reduction pipeline. For wavelength calibration and distortion correction, the Xenon and Argon spectra where used. These were provided in three separate frames, one frame with the both lamps off, serving as a dark for this calibration, and two frames with the Xenon or the Argon lamps on, respectively. The dark frame was subtracted from the illuminated frames and the results were summed to give the final Xenon–Argon calibration frame. The combined frame was used to calculate the dispersion relation and distortion correction using the IRAF tasks `identify, reidentify` and `transform`. The Xenon–Argon spectrum was then self calibrated using the `fitcoords` task. The measured wavelengths in the self calibrated spectra where exact to $\lambda \approx 1$ Å and y ≈ 1 pixel over both dimensions of the frames, i.e., over the dispersion and spatial axes, respectively. For the K–band this accuracy corresponds to an error of 0.05% and a spectroscopic resolution of 7.1 Å per pixel. The wavelength calibration was performed for all five nights separately, each with similar results as mentioned before.

After the spectra have been wavelength calibrated, sky spectra were subtracted from the object spectra to remove the sky background. Figs. 3.11 and 3.12 show an example spectrum of NGC 3585 before and after sky removal, respectively. As can be seen, the emission lines are well corrected for. The spectra have been extracted at various positions along the spatial direction using the `apall` task. For each galaxy three to seven spectra could be extracted (details are given in the next section).

Flux calibration was done implicitly with the correction for telluric absorption (see Section 3.2.1). At the end of each science observation a telluric standard star was observed at a similar airmass as the science observations. The stars observed where G-dwarfs, similar to our

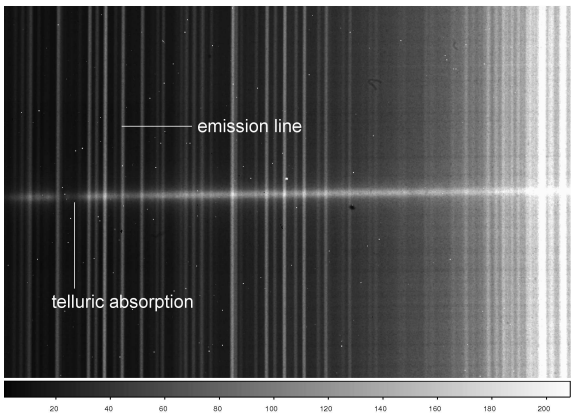


Figure 3.11: One spectrum of NGC 3585 before sky subtraction. The emission lines as well as broad telluric absorption bands are clearly visible.

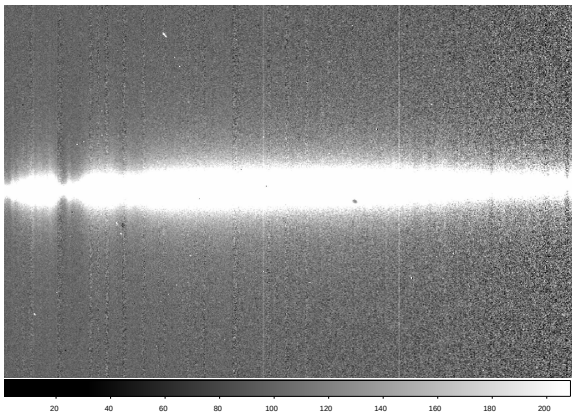


Figure 3.12: The same spectrum after the sky has been subtracted. Apart from some residuals the emission lines have vanished. Note that the telluric absorption is still present, however.

sun and are listed in Tab. 3.3. The reduction of these spectra has been done in the same manner as for the science data. In particular, the same flatfields where used for the standard stars and the galaxy spectra to achieve a relative flux calibration. Since the telluric standard stars have the same spectral type as the sun, the intrinsic absorption lines of the telluric standard can be corrected for by dividing a solar spectrum into the telluric spectrum. A high resolution (R=40000) solar spectrum is provided by ESO. Before the two spectra were divided, the solar spectrum has been converted to the ISAAC resolution (R=4400). The correction for the telluric features was carried out using the `telluric` task in IRAF. Using this task, the telluric spectrum can be shifted and scaled to achieve an optimal correction. The ability to shift the spectrum allows to correct small variations in the wavelength calibration that might be present. Scaling the spectrum allows to account for airmass differences between science and calibration exposures. After the telluric lines have been corrected, the final flux calibration was done by multiplying the spectrum with a blackbody curve for a temperature of $T \approx 5800\,K$, .i.e., the sun's temperature. Although this is not an absolute flux calibration, it is appropriate for the measurement of the D_{CO} index, because the unknown factor to achieve the absolute calibration cancels out in the division of the continuum bands and the absorption band (see Equation 3.4). The reduction described above is illustrated in Figs. 3.13 to 3.15, which show the original spectrum, the spectrum after telluric correction and the final flux calibrated spectrum, respectively.

Since the telluric features change with airmass, the individual spectra could not be combined before the telluric correction but had to be corrected individually. Variations in the strength of the absorption lines between one and the next exposure were significantly different, so the shifting and scaling values found for one was not a good fit for the next. Additionally, the automatic fitting algorithm of the IRAF task did not perform well. It tended to minimize the noise at the edges of the spectra, and not the telluric features. Thus, manual adjustment of the parameters resulted in a much better reduction of the telluric lines.

In the end, all individual spectra of the same spatial region were combined into one final spectrum. This spectrum was corrected to the rest frequencies with respect to the earth, using the radial velocities given in the NASA/IPAC Extragalactic Database (NED) and correcting these for annual effects due to the rotation of the earth around the sun. After this correction the spectra

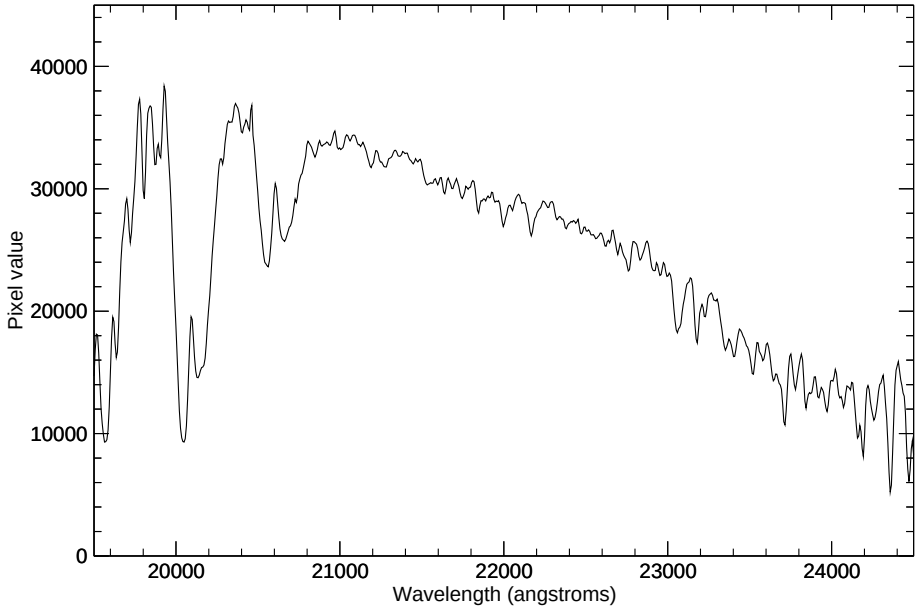


Figure 3.13: A spectrum of NGC 3585 before the telluric correction (compare with Fig. 3.5). The two broad absorption features between 2.0 μm (20000 Å) and 2.1 μm are prominent, although there are many other, less obvious, ones. The flux values are arbitrary.

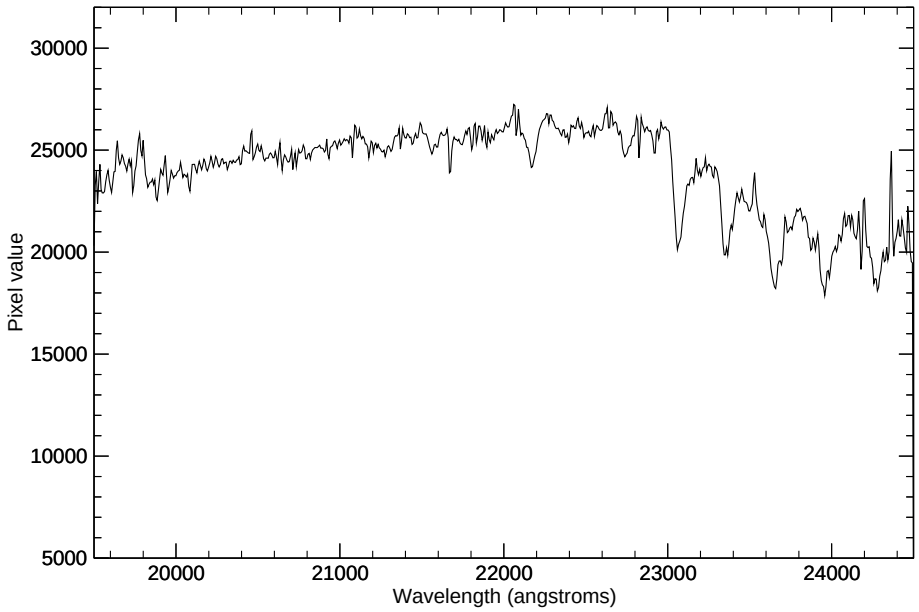


Figure 3.14: The same spectrum after the telluric features have been corrected. The broad features between 2.0 μm and 2.1 μm are no longer visible.

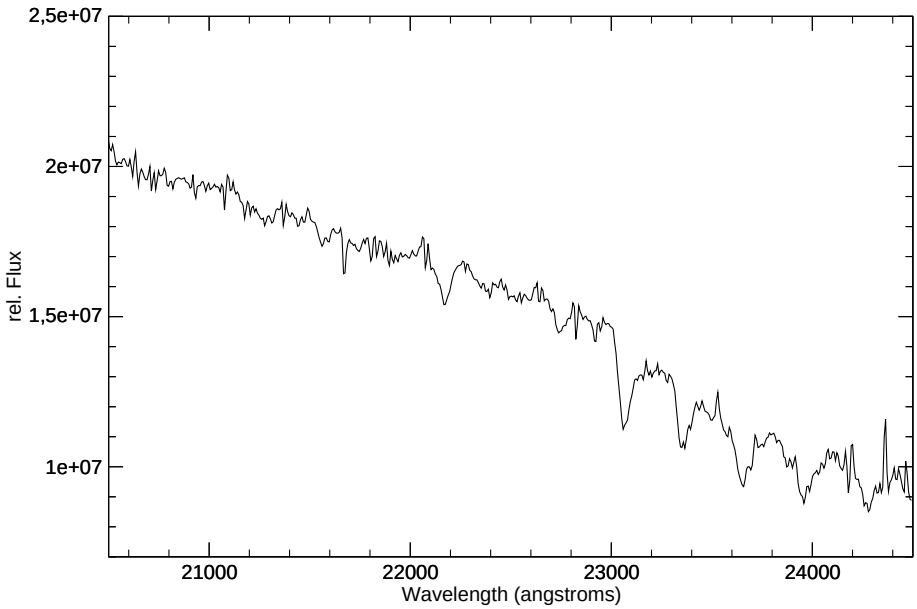


Figure 3.15: The flux calibrated spectrum. Note that the spectrum has been trimmed to the important range for the data analysis.

could be analyzed and the D_{CO} index could be measured. The results for each galaxy are presented next.

Analysis and results

The D_{CO} values and their corresponding errors were computed according to Equations 3.4 and 3.5. To calculate the integral flux values of the continuum and the absorption bands defined in Table 3.1 the `tables` package of IRAF has been used. These fluxes were then divided by the width of the bands. To determine the measured uncertainty the sum of the fluxes in the three bands has been calculated (see Eq. 3.6) together with their variances. The latter was determined by taking the standard deviation of the flux values of the continuum bands as an estimate for the random error of each pixel. The variance of the absorption was taken as the mean of the variances of the continuum bands.

In the following paragraphs each studied galaxy is presented individually. Afterwards, the new data for all four galaxies is compared to the results of similar stellar population and age studies for different galaxy types that are published in the literature.

M 95 For M 95 seven spectra could be extracted, the highest number of all galaxies presented in this thesis. The slit was oriented at a position angle of –65° directly along the bar of the galaxy, cutting through the center (see Fig. 3.16). The spectra have been extracted from the two dimensional spectrum by adding up several lines, depending on their distance from the center of the galaxy. The center is defined by the maximum flux in spatial dimension in this regard. The regions, or "bins", that were added have been chosen to cover equally large regions of the galaxy whenever possible. When the S/N decreased to values where the spectrum could barely be identified, the last bin was chosen to be twice the size of the previous one.

Throughout this thesis the NED value for the Hubble flow distance relative to the 3 K cos-

Table 3.4: Summary of the properties of the spectra of M 95. Negative distance values represent positions north–west of the center, positive distances positions south–east of it (slit position angle = –65°).

Spectrum[a]	Added up region[b] [″]	mean distance [pc]	D_{CO}[c]	$\sigma_{D_{CO}}$[d]
a)	−13.23 to −7.35	–772	1.130	0.098
b)	−7.35 to −4.41	–441	1.129	0.073
c)	−4.41 to −1.47	–221	1.185	0.045
d)	±1.47	0	1.140	0.029
e)	1.47 to 4.41	221	1.176	0.036
f)	4.41 to 7.35	441	1.177	0.040
g)	7.35 to 13.23	772	1.172	0.110

[a] see Figs. 3.16 and C.1
[b] distance from center
[c] CO index as defined by Eq. 3.4
[d] Eq. 3.5

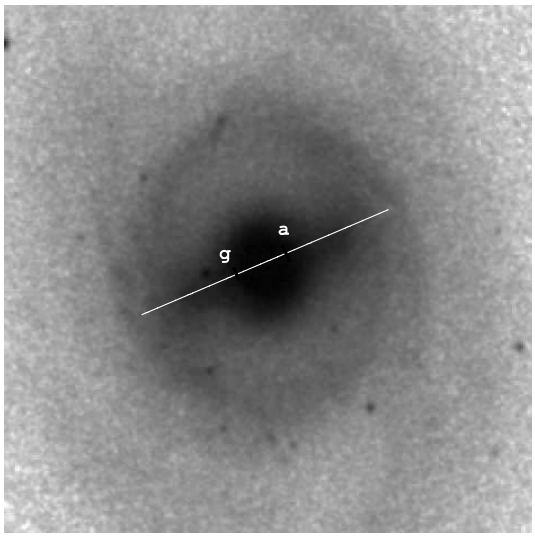


Figure 3.16: DSS image (infrared) of M 95. North is up, east is left. The image size is $4' \times 4'$. The $2'$ long slit at the position angle of –65° is indicated. The outermost distances from which spectra have been analyzed are also indicated by 'a' and 'g'(see Table 3.4).

mic microwave background (D_{3K}) will be taken as the reference, using the Hubble constant $H_0 = 73\,\mathrm{km\,s^{-1}\,Mpc^{-1}}$. For M 95 this is $D_{3K} = 15.4 \pm 1.1\,\mathrm{Mpc}$. This corresponds to an angular scale of $\Delta = 75\,\mathrm{pc}/''$ projected on the sky. Using this scale the outermost bins that were added up correspond to mean distances of roughly 750 pc from the center of the galaxy. Table 3.4 summarizes the basic properties of the different spectra including the measured values and errors for the D_{CO} index. The location of the outermost spectra is indicated in Fig. 3.16 by the characters 'a' and 'g' and all seven are shown in Appendix C (Fig. C.1).

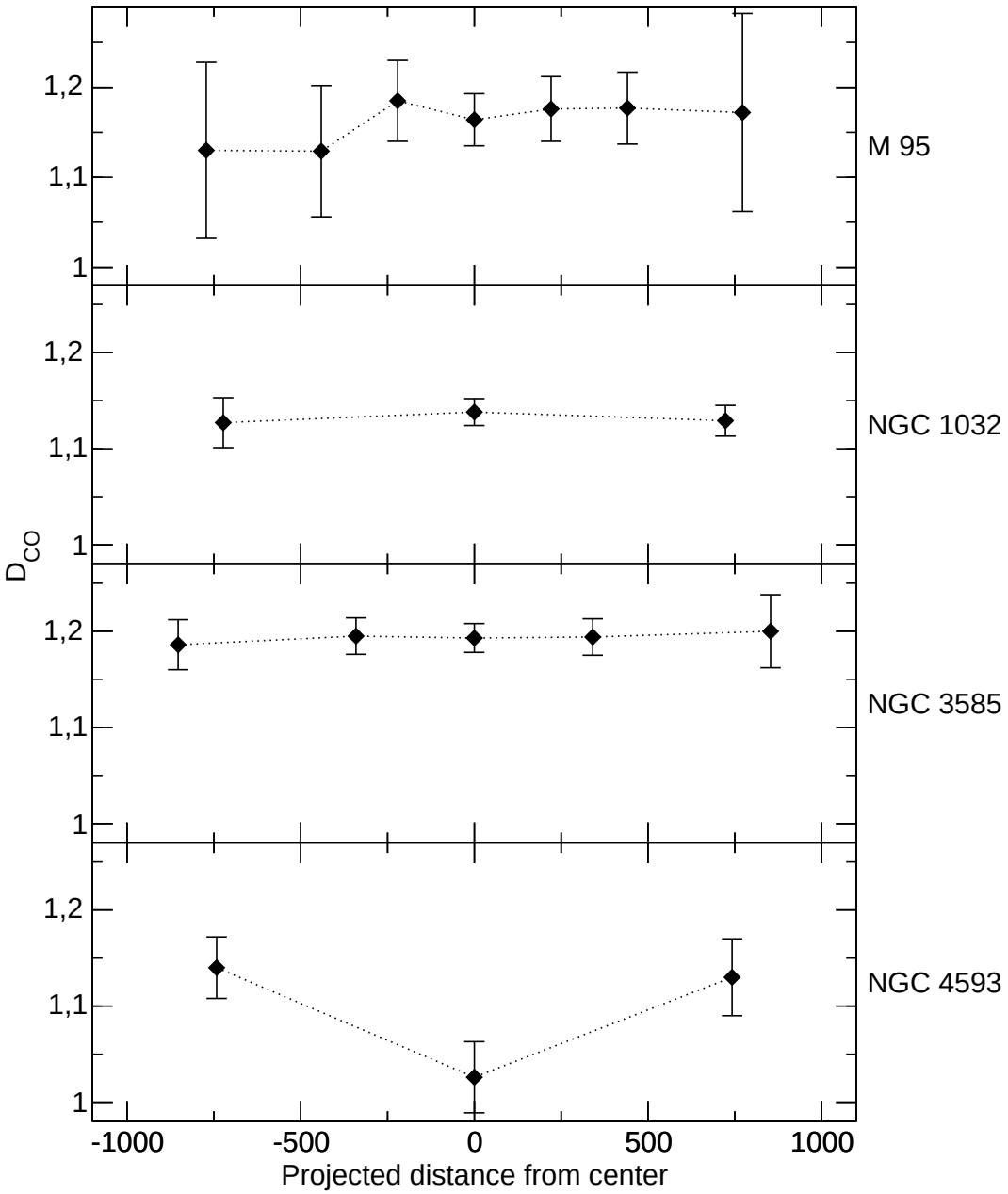


Figure 3.17: The measured D_{CO} values and their errors plotted against the projected distance for all galaxies examined in this thesis. The plots from top to bottom are for M 95, NGC 1032, NGC 3585 and NGC 4593, respectively. The plot for M 95 and for NGC 4593 show variations in the D_{CO} index values along the slit, while the index values for NGC 1032 and NGC 3585 remain constant. See text for further discussion.

Table 3.5: Summary of the properties of the spectra of NGC 1032. Negative distance values represent positions north of the center, positive distances positions south of it (slit position angle = 0°). The column definitions are the same as in Table 3.4.

Spectrum[a]	Added up region [″]	Mean distance [pc]	D_{CO}	$\sigma_{D_{CO}}$
a)	−6.62 to −2.21	–723	1.127	0.026
b)	±2.21	0	1.138	0.014
c)	2.21 to 6.62	723	1.129	0.016

[a]see Figs. 3.18 and C.2

The D_{CO} index shows a trend to higher values in the center of M 95 after being constant for the two outermost bounds at the north–western end of the slit. This is shown in Fig. 3.17 in the uppermost graph. The highest value for the D_{CO} index value is reached for the nearest bin representing the position 221 pc north–west of the center (labeled 'c' in Table 3.4 and Fig. C.1). The values stay constant, taking their uncertainties into account, from the center until the outermost bin at the south–eastern end.

The center of M 95 has been found to host a circumnuclear starburst (Hägele et al. 2007, and references therein). Thus a contribution of a young stellar population is likely to be present in the center of the galaxy. Using J-K photometry, Elmegreen et al. (1997) derive an age of log T = 7.1 for the stellar population of several hotspots in the circumnuclear region. They also find indications for heavy dust absorption in the central region of this galaxy. Comparing Fig. 6 of Elmegreen et al. (1997) with the slit positioning of this thesis, two hotspots (labeled 'C' and 'G' by the authors) might have contributed to the spectra 'f' and 'b', respectively. Unfortunately, this cannot be determined more precisely because of the remaining positioning uncertainty of the slit and the lack of exact coordinates in the reference figure in Elmegreen et al. (1997). If there is a contribution of the hotspots to the spectra, the same age of log T = 7.1 found by the authors for both hotspots leads to different D_{CO} values. For spectra 'b' and 'g' the values are of 1.129±0.073 and 1.177±0.0040, respectively. Both values agree within their uncertainties, however. Looking at the innermost parsecs of M 95 with the HST, Sarzi et al. (2005) find a luminosity weighted age of log T = 9.39 when they apply a model of multiple starburst of various ages. Thus the center of the galaxy is dominated by older stellar populations. Comparing the D_{CO} values of M 95 with the results from the other galaxies (see below), this would imply that the young stellar population found by Elmegreen et al. (1997) is not covered by the ISAAC observations of this thesis.

NGC 1032 Three spectra could be extracted for NGC 1032 (see Tab. 3.5 and Fig. C.2). In Fig. 3.18 the outermost positions for the extracted spectra are marked together with the slit, which was oriented at a position angle of 0°, i.e., cutting diagonally through the bulge and the the center. The extracted spectra were centered on the peak of the brightness profile and at two positions equally distant from this peak. The prominent dust lane of the galaxy disk that is visible in the image could not be identified in the spectrum, indicating that the spectral features are indeed dominated by the stellar component and not severely affected by dust. Thus the bins of the extracted spectra represent two bulge positions north and south of the center and the center itself. Using the NED parameters of $D_{3K} = 33.8 \pm 2.4$ Mpc and $\Delta = 164\,\mathrm{pc}/''$ the two bulge

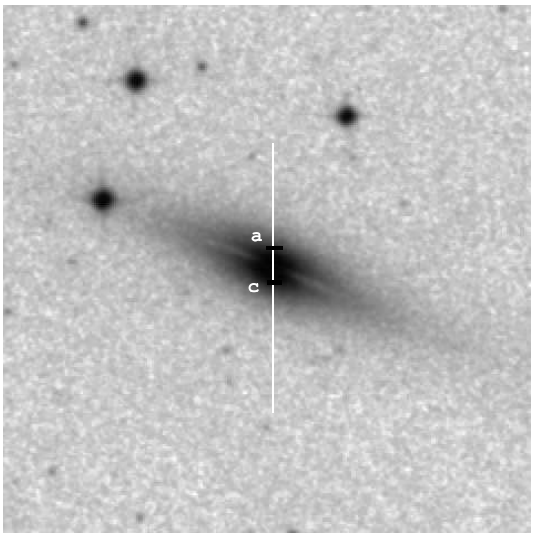


Figure 3.18: DSS image (infrared) of NGC 1032 ($4' \times 4'$). The position angle of the slit is 0°. North is up, east is left. The outermost distances from which spectra have been analyzed are indicated by 'a' and 'c' (Table 3.5).

spectra were taken at a mean distance of ≈720 pc from the center.

Looking at Table 3.5 and Fig. 3.17 no significant trend or change of the D_{CO} index values is visible, suggesting that the dominant stellar population does not change significantly with distance from the center. The mean values are below those of M 95 and NGC 3585 and are among the lowest that are found in the current sample. Although lower D_{CO} values for NGC 4593 have been found these are probably due to different physical effects in that galaxy. This is discussed below.

NGC 1032 has a spheroidal bulge (Lütticke et al. 2000a) and has been used, e.g., by Bureau et al. (2006) in their study of pseudobulges as part of a control sample representing classical, merger built bulges. Thus, this bulge is expected to share similar properties with elliptical galaxies. Terlevich & Forbes (2002) found an age of log T = 9.49 and a metallicity [Fe/H] > 0.5 for this galaxy, somewhat older than the value for the center of M 95. The lower D_{CO} value of NGC 1032 would then fit to the assumption that younger populations have higher CO absorption strengths by trend. But the overall picture is more complicated. As mentioned above, the properties of the bulge of NGC 1032 should be similar to elliptical galaxies, like NGC 3585.

NGC 3585 The properties of the five spectra for NGC 3585 are summarized in Table 3.6 and the spectra are shown in Fig. C.3. The slit was positioned at an angle of 0°, cutting the galaxy in north-south direction. The slit positioning and the limits where the outermost spectra were obtained are indicated in Fig. 3.19. As usual the center has been defined by the peak of the brightness profile, for which a spectrum covering the central $2\farcs94$ of the galaxy was extracted. Two spectra of the same spatial bin where extracted next to the center, while the outermost spectra cover twice the area of the galaxy projected on the plane of the sky. The NED values of $D_{3K} = 23.9 \pm 1.7$ Mpc and $\Delta = 116\,\mathrm{pc}/''$ have been used to compute the distances from the center.

Similar to the result found for NGC 1032, the D_{CO} index values do not show any dependence on the distance from the center and are basically flat (Fig. 3.17). The dominant stellar population

Table 3.6: Summary of the properties of the spectra of NGC 3585. Negative distance values represent positions north of the center, positive distances positions south of it (slit position angle = 0°). The column definitions are the same as in Table 3.4.

Spectrum[a]	Added up region [″]	Mean distance [pc]	D_{CO}	$\sigma_{D_{CO}}$
a)	−10.29 to −4.41	–853	1.186	0.026
b)	−4.41 to −1.47	–341	1.195	0.019
c)	±1.47	0	1.193	0.015
d)	1.47 to 4.41	341	1.194	0.019
e)	4.41 to 10.29	853	1.200	0.038

[a] see Figs. 3.19 and C.3

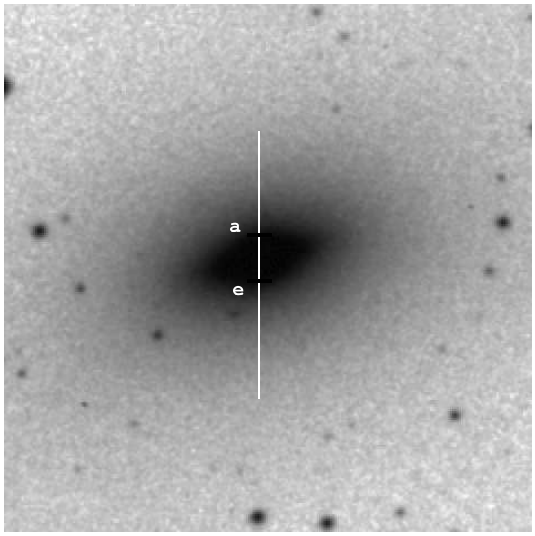


Figure 3.19: DSS image (infrared) of NGC 3585 ($4' \times 4'$). The position angle of the slit is 0°. North is up, east is left. The outermost distances from which spectra have been analyzed are indicated by 'a' and 'e' (Table 3.6).

is therefore most likely homogeneously distributed throughout the analyzed region. The measured values are much higher than the ones for NGC 1032 and by trend also higher than the ones for M 95. The values agree very well with the results of similar studies of elliptical galaxies carried out by Silva et al. (2008), Mobasher & James (2000) and James & Mobasher (1999). As for NGC 1032, Terlevich & Forbes (2002) determine the age of NGC 3585 to log T = 9.49 and a metallicity [Fe/H] > 0.5. Since the age and metallicity for both galaxies are identical within the 20% uncertainty mentioned by Terlevich & Forbes (2002), the different D_{CO} values are very conspicuous, since they have been measured with high S/N and cannot be matched even when the maximum uncertainties are taken into account.

James & Seigar (1999) mention other effects that can change the absorption strength of the CO bands apart from age and metallicity. The first of these is an extremely young starburst

log T $\leq$ 7, which can be ruled out for NGC 1032. The other possibility is significant contribution by a possible non stellar continuum, e.g., by dust. To affect the continuum significantly, the dust would have to be heated by star forming regions to temperatures of several hundred Kelvin (James & Seigar 1999). While this is not ruled out in principle for the center of NGC 1032, the similar D_{CO} values measured for the bulge of the galaxy cannot be explained. A significant contribution by a non thermal continuum is also highly unlikely for the bulge of NGC 1032. Summarizing the above, the discrepancy between the D_{CO} values found for NGC 1032 and NGC 3585 remains peculiar when both galaxies share the same age and metallicity.

NGC 4593 The fourth galaxy studied in this thesis is NGC 4593. Three spectra were extracted from the observations (Tab. 3.7 and Fig. C.4). The center of this galaxy is extremely bright (see discussion below), so the central spectrum was extracted from the innermost 1.″47 and the two other spectra were combined from the adjacent regions. The slit was oriented at an position angle of 53°, stretching over most of the bar of the galaxy. This is illustrated together with the positions of the outermost spectra in Fig. 3.20. Using $D_{3K} = 41.7 \pm 3.0$ Mpc and $\Delta = 202\,\mathrm{pc}/''$ (NED) these two bulge spectra have a mean distance of $\approx$740 pc from the center. The extreme steepness of the brightness profile along the slit for this galaxy means that a substantial amount of light in the outer two spectra comes from regions significantly closer to the values mentioned above, however.

Looking at the lowermost panel of Fig. 3.17, an interesting slope is visible. The D_{CO} index value for the center is considerably lower than the ones found for the adjacent regions. In fact, it is the lowest value found for all galaxies studied in this thesis. The two bulge spectra have low values compared to NGC 3585, more similar to those found for NGC 1032, and the north–western part of the M 95 spectra. The extremely low value that is found for the center is real and not due to an measurement uncertainty, however. Looking at the spectrum for the center shown in Fig. C.4, one can see that it is basically flat, showing no signs of CO absorption. The cause for this continuum slope lies in the contribution of the active galactic nucleus (AGN) that is located in the center of the galaxy (Kollatschny & Fricke 1985, and references therein). NGC 4593 is classified as a Seyfert galaxy (Seyfert 1943) and the bright nucleus with its non stellar continuum completely dominates the spectrum. Thus, any CO absorption that might be due to stars in the central region of the galaxy is hidden. The effect is most probably also significant for the other two spectra obtained from the offset positions and thus affecting the measured D_{CO} values. Due to the excess of non stellar emission, these measured values are lower compared to the undisturbed ones. A qualitative interpretation of the index values regarding age and metallicity of the dominant stellar population is therefore very difficult based on the current set of data.

Comparison of the index values with other publications After the individual discussion of the examined galaxies, the new results are now compared to other publications that used the CO absorption feature for stellar population studies. To compare the older results with the new index definition of Mármol-Queraltó et al. (2008), the conversion formulas (Eqs. 3.8 and 3.9) were used to calculate the D_{CO} values from the published ones in other systems.

Fig. 3.21 shows the measured D_{CO} values for elliptical galaxies *vs.* the Hubble type defined in de Vaucouleurs et al. (1991). The elliptical galaxy NGC 3583 (brown) and the classical bulge of NGC 1032 (orange) are shown as larger circles with their measured error values. The data points from the other publications are not shown with their uncertainties, since this would overcrowd the diagram significantly. The uncertainties are in the order of 0.005 in absolute index values.

Table 3.7: Summary of the properties of the spectra of NGC 4593. Negative distance values represent positions north–east of the center, positive distances positions south–west of it (slit position angle = 53°). The column definitions are the same as in Table 3.4.

Spectrum[a]	Added up region [″]	Mean distance [pc]	D_{CO}	$\sigma_{D_{CO}}$
a)	−6.62 to −0.74	–742	1.140	0.032
b)	±0.74	0	1.026	0.037
c)	0.74 to 6.62	742	1.130	0.040

[a]see Figs. 3.20 and C.4

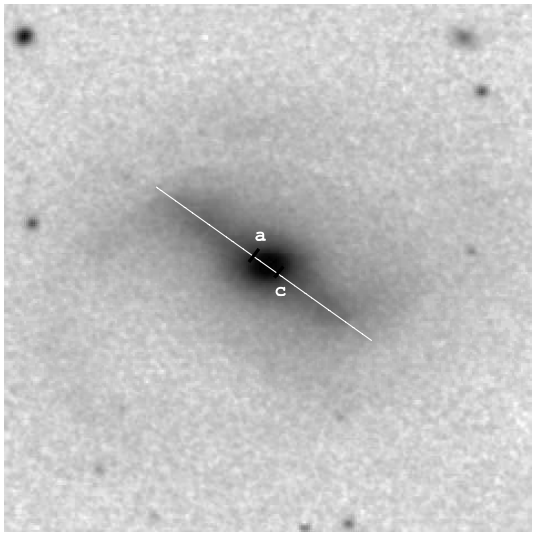


Figure 3.20: DSS image (infrared) of NGC 4593 ($4' \times 4'$). The position angle of the slit is 53°. North is up, east is left. The outermost distances from which spectra have been analyzed are indicated by 'a' and 'c' (Table 3.7).

NGC 3585 is classified as type '–5' in de Vaucouleurs et al. (1991), but has been offset by a small amount in Fig. 3.21 to make the comparison with other elliptical galaxies easier.

The smaller circles shown in red, green, and blue are the values published by James & Mobasher (1999), Mobasher & James (2000), and Silva et al. (2008), respectively. The first two publications used the EW definition of Puxley et al. (1997) for their analysis. The results were converted into the D_{CO} values using Eq. 3.8. Silva et al. (2008) used the definition by Frogel et al. (2001) for their study of ellipticals in the Fornax cluster. These values were subsequently converted using Eq. 3.9 into the D_{CO} index.

James & Mobasher (1999) divided their sample of galaxies into cluster ellipticals and isolated ones, including galaxies that belong to small groups in the second category. These field ellipticals have been included in Fig. 3.21, since NGC 3585 itself has no close neighbors. The sample of Mobasher & James (2000) is separated into galaxies which are closer than 0.2° angular distance from the center of the cluster they belong to and those that are further out. The galaxies with

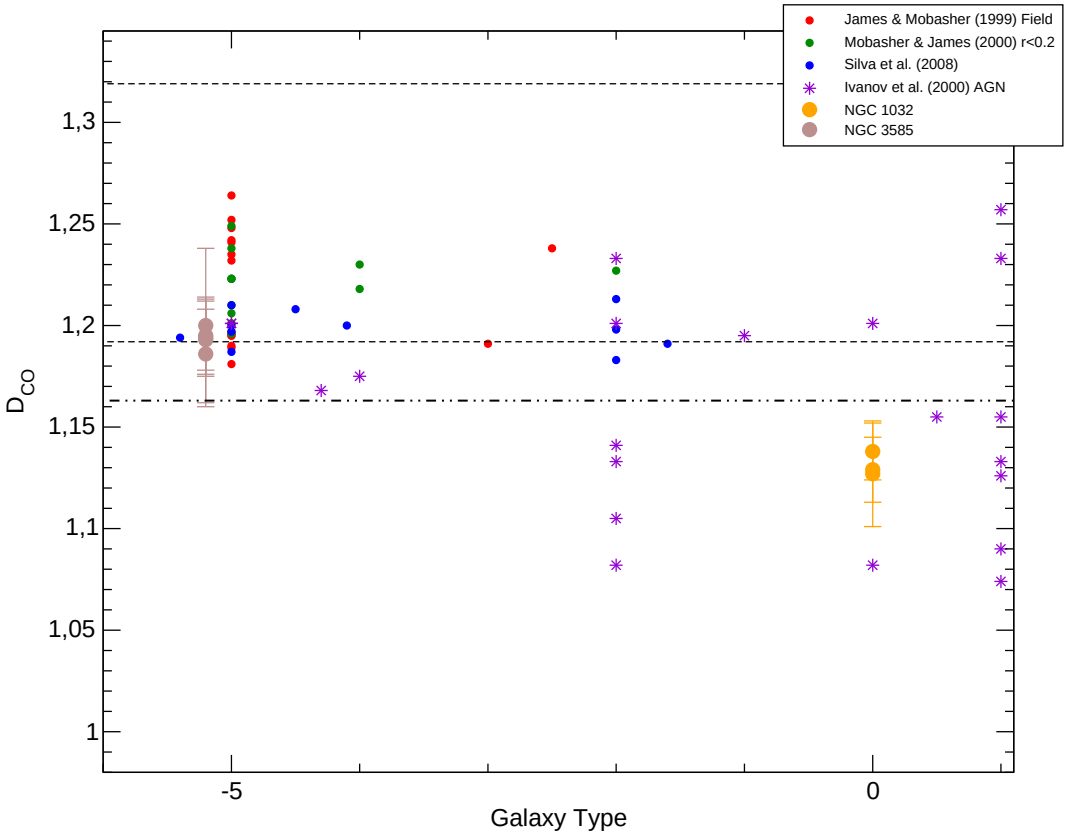


Figure 3.21: Comparison of the new D_{CO} values with other published values for elliptical galaxies. The index values are plotted against the Hubble type (de Vaucouleurs et al. 1991). The two proposed values for a purely old and a young stellar population (James & Seigar 1999) are indicated by dashed lines at D_{CO}=1.192 and D_{CO}=1.319, respectively. See text for discussion. The value of D_{CO}=1.163 derived from the template spectrum of Cesetti et al. (2009) is shown as a double dot–dashed line.

$r < 0.2^{\circ}$ are shown in Fig. 3.21.

Finally, a subset of galaxies from Ivanov et al. (2000) are shown in Fig. 3.21 as purple stars. This study focused on AGN and starburst galaxies, measuring the CO absorption with an spectroscopic index comparable to the one defined by Doyon et al. (1994). These values were converted to the EW defined by Puxley et al. (1997) by the formula published therein and subsequently transformed to D_{CO} values using Eq. 3.8. Three galaxies from Ivanov et al. (2000) shown in Fig. 3.21 are "true" ellipticals (without an AGN) and were included as reference galaxies in that publication. The data point with galaxy type '–4.3' belongs to NGC 1144, a peculiar AGN that is interacting with NGC 1141 and appears elliptical on DSS images. The other data points from Ivanov et al. (2000) belong to AGN with S0 type host galaxies. It must be mentioned that only those galaxies from the other publications for which the galaxy type could be inferred are shown in Fig. 3.21.

Looking at the data points for NGC 3585 it is obvious that this galaxy is comparable to many other ellipticals studied in the selected paper. All points lie close to the line that defines a metal rich, predominantly old stellar population, as suggested by James & Seigar (1999) but are considerably higher than the value derived from the template spectrum of Cesetti et al. (2009) for

an old elliptical. The values for NGC 3585 are significantly smaller as those for elliptical galaxies found in small groups (the red circles with higher D_{CO} values in Fig. 3.21), for which James & Mobasher (1999) suggest that their larger index values are due to frequent minor mergers, and thus younger stellar populations. The authors exclude metallicity differences as the cause for the varying CO absorption strengths based on their measurements.

The measured D_{CO} values for NGC 1032 show striking difference compared with the other elliptical galaxies. These values lie unambiguously below the ones found for elliptical galaxies, even lower than the value found for the spectrum of Cesetti et al. (2009). The best agreement for NGC 1032 exists with the galaxies studied by Ivanov et al. (2000). Although similar in type, all of these galaxies have an AGN in their center, which can affect the D_{CO} values quite significantly. NGC 1032 shows no sign of an AGN, however. Instead it has been found to contain a classical bulge which is thought to be old and similar to ellipticals. At least from the results of the CO absorption strength, this similarity is definitely not present. The cause for the discrepancy is difficult to understand, taking the results of Terlevich & Forbes (2002) into account, who find comparable metallicities and ages for both NGC 1032 and NGC 3585 (see above). If the general trend of lower CO absorption strengths, equivalent to lower D_{CO} index values, corresponds to older mean stellar populations (James & Seigar 1999) is valid, either the metallicity of NGC 1032 must be lower than that of NGC 3585 or the mean ages must be different. On the one hand, Michard (2006) find a slightly younger age of $\log T = 9.235$ for NGC 3585. On the other hand, the fact that the D_{CO} index for NGC 1032 nearly reaches the line defined by Cesetti et al. (2009) for an elliptic galaxy with $\log T = 9.978$ might suggest that the dominant stellar population of NGC 1032 is considerably older than that of NGC 3585. The results for NGC 1032 and NGC 3585 also support the results of Gadotti (2009), who find differences between ellipticals and classical bulges in the mass–size relation that might indicate different formation scenarios.

Fig. 3.22 shows the comparison of the two spiral galaxies studied in this thesis with other galaxies studied using the same technique. M 95 (green diamonds) and NGC 4593 (purple squares) are both classified as galaxy type '3' in de Vaucouleurs et al. (1991). To better distinguish between the values for both galaxies, the type for M 95 has been slightly shifted from the published one. For completion, the values for NGC 1032, which were discussed in the last paragraph already, are also shown as orange circles in Fig. 3.22. The values are compared to the results from James & Seigar (1999), who studied the three galaxies NGC 613 (galaxy type '4'), NGC 628 (type '5') and NGC 7741 (type '6'). The values were converted from the published EW values into the D_{CO} index using Eq. 3.8. Another comparative sample is given by the galaxies studied by Ivanov et al. (2000). The starburst sample is shown as green stars, the AGN sample as purple stars (as already introduced in Fig. 3.21). Galaxies with peculiar classifications (types '90' and '99') are included as type '12' in Fig. 3.22 to keep the diagram compact. Only the galaxies from Ivanov et al. (2000) for which a type classification was available are shown in the Figure.

Most of the measured D_{CO} values for M 95 lie close to the ones derived for the starburst sample of Ivanov et al. (2000). All values are lower than those of the starburst sample, however. The two lowest values found for the north–western center of M 95 (see Table 3.4 and Figure 3.16) come close to this region only when the maximum uncertainties are taken into account. They are comparable to the lowest values found by James & Seigar (1999) for their galaxy sample. The low indices are explained by the authors as being affected by significant amounts of dust or other non stellar emission. Both explanations may also be valid for the case of M 95. The lowest two values for M 95 also lie significantly below the line defined by the template spectrum of Cesetti et al. (2009) for an old elliptical galaxy. The other D_{CO} indices for this galaxy agree fairly well with this line, however. If the D_{CO} index is governed by the mean stellar population and not seriously

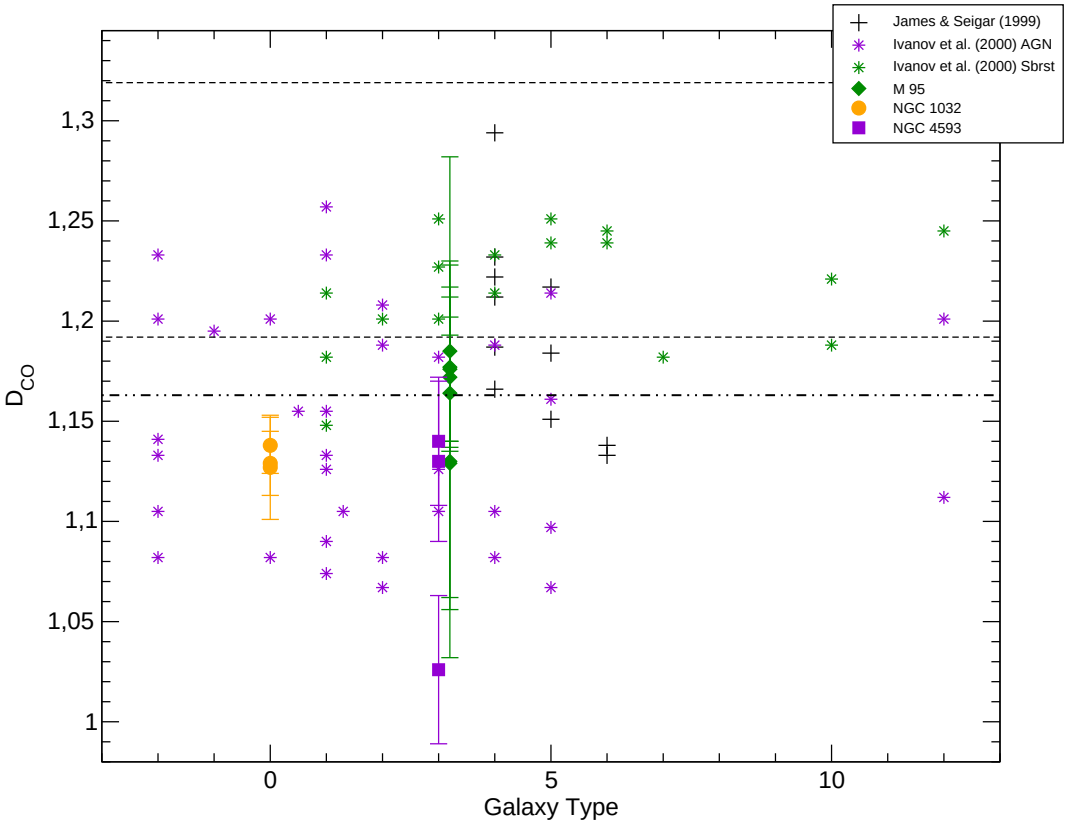


Figure 3.22: Same as Fig. 3.21 but for spiral galaxies, including starburst galaxies and AGN. The galaxy types for peculiar galaxies from de Vaucouleurs et al. (1991) ('90' and '99') are shown as type '12' to keep the diagram compact. See text for discussion.

affected by a non stellar contribution the overall result for the bulge of M 95 is that it consists of a significant fraction of older stars, and the slit did not cover any star forming regions in the eastern circumnuclear starburst ring (see Fig. 6 of Elmegreen et al. 1997). This result is consistent with the results of MacArthur et al. (2009) or Peletier et al. (2008), who find that pseudo bulges also show significant contributions of older stellar populations.

For NGC 4593 (purple squares), the extremely low D_{CO} index measured for the central AGN clearly stands out. Even taking the uncertainties into account it is among the lowest values that have been found for other AGN by Ivanov et al. (2000). The two adjacent regions that were measured show values that are more consistent with this comparative sample. They also agree well with the lowest indices found for M 95. On the one hand, this might suggest that the affected regions in both galaxies have a significant contribution of non stellar light in their NIR spectra. On the other hand, the measured D_{CO} indices for NGC 4593 and M 95 also agree fairly well to the ones found for NGC 1032. There is no indication for a starburst or AGN in this galaxy, however.

Concluding this discussion, Fig 3.23 shows the combined data that was already presented in Figs. 3.21 and 3.22, but covering all galaxy types. The main result from this diagram is that no overall trend between the D_{CO} index and galaxy type is visible. Interestingly the scatter of index values is lowest for the elliptical galaxies, and increases significantly for galaxy types ≥ -3. It it thus evident that the strength of the CO absorption is affected more by internal physical processes and environments of the galaxies than by the dominant stellar population. The age–metallicity

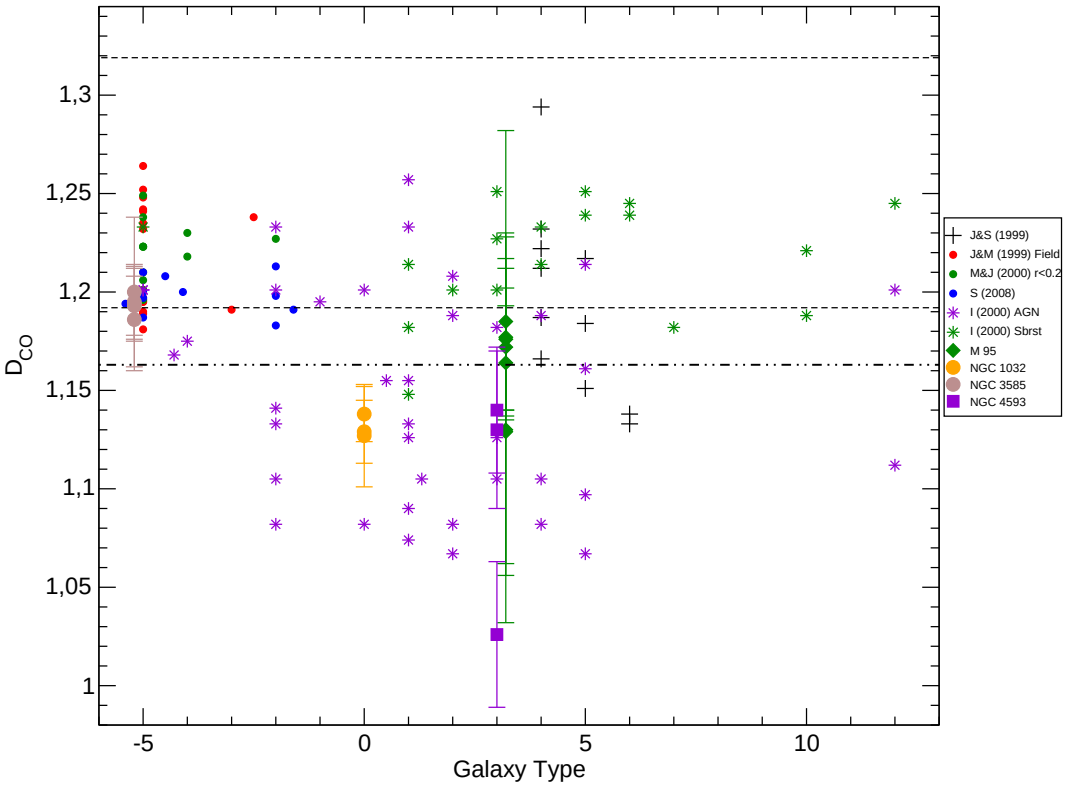


Figure 3.23: Same as Figs. 3.21 and 3.22, but showing all galaxy types. No significant trend of the D_{CO} values with Hubble type can be inferred. See text for discussion.

degeneracy is another problem that has to be taken care of when the CO absorption is used to study the ages of galaxies.

The general trend that lower index values are an indication for an older stellar population is in agreement with the measured value of D_{CO}=1.162±0.008 for the template elliptical galaxy spectrum for an old stellar population derived by Cesetti et al. (2009). The value postulated by James & Seigar (1999) for a very young stellar population is not found for any of the studied galaxies, however. One possible explanation for this result is that other processes that are common in star forming regions, like non stellar continuum emission by dust, affects the CO absorption strength and thus the D_{CO} index, dragging it to lower values.

3.3.3 LUCIFER observations

It was planned to examine the CO absorption bands at 2.3 μm with LUCIFER during the commissioning of the instrument at the LBT. Two targets that were suitable for both observations with ISAAC and LUCIFER were chosen as targets for the spectroscopic commissioning. These targets were M 95 and NGC 4593. Due to scheduling problems only NGC 4593 could be observed, however. After the data had been reduced it turned out that it could not be used for a meaningful scientific examination, unfortunately. This data analysis of the LUCIFER spectra of NGC 4593 and the problems that were encountered are described in the following.

Data reduction

The LUCIFER observations of NGC 4593 were carried out during the commissioning run in January 2009. The data was acquired on the night of the 13.01.2009. Due to the time of the year the galaxy could be observed only in the last quarter of the night, and the spectra were taken shortly before sunrise. In total, five object spectra with a total integration time of 10 minutes were planned, but due to imperfect positioning of the telescope, the last three object spectra were lost because the object could not be moved back on the slit after sky frames had been taken. This resulted in only two usable object spectra of NGC 4593 with two minutes integration time each. After the object spectra had been taken, the star Hip55122 (spectral type B8) was observed to correct for telluric features. This observation were severely limited by the increasing background due to the twilight.

The data was corrected for the dark current of the detector using calibration frames taken the day before and the morning after the observations. The flat field was computed from one image with the halogen lamps switched on that was subtracted from the next image with the lamp off, thus removing the dark current of the detector. This flat field was severely limited by flux differences over the spectral range due to the diffusion disk of the calibration unit that was out of specifications and caused a significant flux deficit beyond 2.2 μm. This problem has been fixed after the maintenance of the unit in March 2009. A second problem became also apparent at this time. A significant back reflection exists in the right area of the detector for pixel values of $x \geq 1000$. The cause of this reflection is currently being investigated. The flatfielded spectra were subsequently wavelength calibrated and distortion corrected using calibration frames of Argon emission lines. The error in the wavelength calibration was lower compared to the ISAAC calibrations (see Section 3.3.2). The self calibrated spectrum of the emission line data showed distortion corrections better than one pixel and the wavelength calibration was on average better than 0.5 Å. The spectral resolution was 2.3 Å per pixel, which is better than a factor of two compared to the ISAAC one (7.1 Å per pixel) and near the maximum resolution of LUCIFER which is R=5000 in seeing limited mode in the K–band. The following sky subtraction revealed some serious problems. It was not possible to remove the sky emission from the data without creating strong residuals. The reason for this is probably mainly due to the observation time early in the morning before sunrise. For the sky frames the same location as for the ISAAC observations was chosen but this offset was apparently too large at the time with increasing twilight. Even an attempt to remove the sky emission from the object frame did not result in a good subtraction because the sky lines changed in intensity over the field of view of the detector.

After the same reduction was performed for the telluric standard star the problems increased even further. The sky became so variable that it was impossible to extract a suitable sky spectrum from the only available object spectrum. Consequently, it was not possible to correct the galaxy spectrum for telluric features and to flux calibrate the spectrum. A scientific analysis of the data taken with LUCIFER was therefore not possible, unfortunately.

3.4 Summary and Outlook

The CO absorption features prominent in the K–band were studied in this thesis. ISAAC spectra of four galaxies of different Hubble type have been analyzed in the scope of the new D_{CO} index definition of Mármol-Queraltó et al. (2008). The addressed question focuses on the evolution of spiral galaxies in the context of secular driven processes of bulge formation. The monolithic collapse model introduced by Eggen et al. (1962) understands the bulge as the oldest part of spiral

galaxies, similar to elliptical galaxies. In many ways these "classical" bulges of some spirals share many properties of ellipticals. In the scenario of secular evolution, the bulges of some galaxies share parameters that are common to the spiral disks that surround them (disky bulges).

The results from this thesis regarding this question are ambiguous. No clear indication is seen that the stellar population of the two studied spiral galaxies, M95 and NGC 4593, which have been found to host a disky bulge, have young stellar populations in their centers. However, the classical bulge of NGC 1032 shows a significantly different D_{CO} index than the elliptical galaxy NGC 3585. This finding is confirmed by the comparison of the index values with other elliptical galaxies. Regarding the numerical value for D_{CO} only, most regions studied for M 95 are very similar or at least not well distinguishable from ellipticals. If the simple assumption that the strength of the CO absorption feature is dominated only by the stellar population is applied, this result leads to the conclusion that the stellar population of the center of M 95 is similar to the ones found in ellipticals. This finding is supported by the measured D_{CO} index for the template elliptical from Cesetti et al. (2009), which fits well to most of the values measured for M 95. However, different internal processes and environments in a galaxy can severely affect the measured D_{CO} index. A prominent example is the value measured for the AGN of NGC 4593, which is among the lowest values that has been found for any galaxy. The main result from the analysis is that the measurement of the CO absorption itself is not sufficient to constrain the dominant stellar population. The influence of other physical processes and environmental conditions has to be determined by additional measurements. The low index value found for NGC 1032, which is significantly lower than those found for the elliptical galaxies is remarkable. It would therefore seem worthwhile to extend the study of CO absorption strengths to galaxies showing a classical bulge to constrain the range of D_{CO} values using a larger sample of galaxies. This would then clarify if NGC 1032 is an exceptional case or if there is a general trend present which has to be explained by theories for structure formation in the universe.

A meaningful LUCIFER spectrum to address this scientific question could not be acquired during the commissioning of the instrument so far. The spectra that were obtained suffer from internal, i.e., calibration, issues as well as external, i.e., sky subtraction and telluric correction. However, the measured high spectral resolution of the instrument promises a significant impact regarding this question in the future. The high resolution of LUCIFER will allow to measure the D_{CO} index with much greater accuracy compared to other current NIR spectrographs. Additionally, using the unique MOS capabilities of the instrument will allow to study environmental effects of the CO absorption strength with much improved efficiency compared to the current possibilities. This efficiency grows even further when both LUCIFER instruments will be available and able to carry out observations with the LBT in parallel.

Chapter 4

Conclusions and future work

One of the main research areas of present day astronomy is to understand the formation and evolution of galaxies. In this context, the currently extensively studied question of secular evolution of galaxies has been examined in this thesis by means of NIR spectroscopy in the K–band. The two LUCIFER near infrared instruments for the LBT will be able to make substantial contributions to this research in the future.

On of the main intentions behind the thesis has been to maximize the astronomical usage of the LUCIFER instrument. The needs by different user groups, i.e., engineers and observers, differ greatly. The control software has to has to take care of this problem and provide convenient and easy to use tools for both user groups. Security issues for the operation of the instrument that arise have to be solved, too. Taking all of these points into account, it becomes clear that the design of the control software plays an vital part for the overall success of an instrument like LUCIFER.

Most of the the software for LUCIFER has been written in *JAVA* using its RMI capabilities to create a distributed system. In this paradigm, every service that is essential for the operation of the instrument or the management of the software is realized as an individual process that provides the needed functionality to other services or programs, e.g., user interfaces. The separation of different tasks into individual processes greatly improves the overall stability and availability of the system. The software is structured in four different layers, where higher layers generally solve more complex tasks. This keeps the complexity within each layer manageable which also helps to increase the stability of the software.

The lowest layer, the *System Layer*, consists of services that are necessary to manage the distributed system. It provides configuration control and other vital functions.

The *Control Layer*, located directly above the *System Layer* is the main interaction point between the hardware of the LUCIFER instrument and contains all services that communicate directly with hardware components. This includes services that are responsible to control the motors and query the position switches of the instruments, as well as services that query environment data, e.g., temperatures and pressures inside LUCIFER. Control of the external packages for the detector readout and telescope control are located in this layer as well. Finally, the `Journalizer` service that keeps track of the overall state of the instrument and generates the FITS header information is also part of the *Control Layer*.

The third layer, the *Instrument Layer*, models all physical units of the instruments that need to be controlled with corresponding software services. These services rely on the ones provided

in the *Control Layer* to work properly. The focus for these services lies on the tasks that have to be solved for the corresponding unit, rather than on the problem of how to move a motor, for example. One excellent example for this concept is the `GratingUnit` service. From the users point of view, either the mirror or one of the gratings have to be placed in the optical beam, depending if the intended science observations are imaging or spectroscopy. For spectroscopy, the central wavelength that should be used for the observations is a second parameter. To be able to provide the necessary functionality to the users of the instrument, the `GratingUnit` service needs to move the physical unit to different positions and apply a specific voltage to tilt the gratings. These more basic tasks are solved by the services located in the *Control Layer*. The responsibility of the `GratingUnit` is to select the correct position of the unit and to set the correct tilt voltages depending on the central wavelength chosen by the user. To do this, the necessary tilt voltages can be interpolated using `LookupTables`.

The *Operation Layer*, which represents the highest layer of the control software, also directly benefits from the increasing level of abstraction. All user interfaces as well as the three main managing services for the instrument, telescope and the readout are located here. The manager services solve the highest level complexities, which include execution of the necessary actions to change the instrument from one setup to another. The user interfaces used by astronomers directly communicate with the manager services. Services of lower layers are not directly accessible. This ensures a safe operation of the instrument by an inexperienced observer since the managers take care of the correct order in which the commands are send to the instrument in order to execute the desired setup change. Engineers have full control over the complete software system by user interfaces that allow them to interact directly with individual services of lower layers. The efficient operation of the instrument is provided by the use of observation scripts that allow a fully automatic observation of an astronomical object, including necessary telescope offsets between individual exposures. Complex observing programs can be created before the observations are carried out and executed later with minimum overheads.

The development of the observation scripts greatly benefited from the experience gained with the preparation of the observations with ISAAC that were analyzed in this thesis. Four galaxies of different Hubble types have been observed with the VLT, in order to study the dominant stellar population in their centers. NIR K–band long slit spectra where taken and the CO absorption feature at 2.3 μm measured using the new D_{CO} index definition proposed by Mármol-Queraltó et al. (2008). For each galaxy, multiple spectra could be extracted along the slit. The measured index values were analyzed for each galaxy individually, focusing on the change of the index values with regard to the distance of the center. For two galaxies, NGC 1032 — showing a classical bulge — and NGC 3585 — an elliptical — no significant change of the index values with distance from the center could be found. This means that the mean stellar population is very homogeneous throughout the examined region, if the CO absorption is dominated by the stellar population. For the two spiral galaxies showing a disky bulge in their center and thus signs of secular evolution processes at work, the D_{CO} values vary along the slit. The responsible processes causing this change are likely to be different, however. For M 95 the differences could be due to a change in the stellar population. Alternatively, influences due to a significant contribution by a non stellar continuum are possible. For NGC 4593, the spectrum is dominated by the extremely bright AGN that is located in the center of that galaxy. The D_{CO} value measured for the center is one of the lowest value found for any galaxy so far. Because of the extreme brightness of the central core the low index values measured for the adjacent regions are also likely to be caused by contamination of the spectrum from the center.

Comparing the newly found D_{CO} values with other ones published in the literature shows

no global trend of index values with Hubble type. The spread of measured values is smallest for elliptical though and increases significantly for galaxies with types ≥ -3. This leads to the conclusion that local environmental effects have to be taken into account when the index values are analyzed. Interesting is the result that the index values found for the classical bulge of NGC 1032 are considerably lower than the ones found for NGC 3585. Since the CO absorption of NGC 3585 is similar to that found for other elliptical galaxies the value for NGC 1032 can be compared in a broader context. Both galaxies have been found to have the same same mean age and metallicity (Terlevich & Forbes 2002), ruling out two important factors that affect the CO absorption index. Other physical effects which are likely to affect the index values, e.g., non stellar emission of dust or AGN are very unlikely for the bulge of NGC 1032 as well. The values found for NGC 1032 also lie lower than the one that was measured for a template spectrum of an old elliptical galaxy derived by Cesetti et al. (2009). This could indicate an predominantly old stellar population, considerably older than for most elliptical galaxies, or a significantly different one. Taking all of these indications together, one possible consequence is, that the classical bulges found in spiral galaxies might not simply be small ellipticals surrounded by stellar disks. This result thus supports the same findings of Gadotti (2009).

For spiral galaxies the spread in the measured CO absorption strengths is considerably larger than for ellipticals. The D_{CO} values found for NGC 4593 are dominated by the non stellar contribution of the AGN. Any conclusions for dominant stellar populations are thus difficult to achieve. Most values found for M 95 are in good agreement with the ones found in elliptical galaxies with old populations. If the stellar continuum dominates the spectra the stellar population of the pseudo bulge of M 95 thus will constitute of a significant fraction of older stars. Similar results have recently been found by MacArthur et al. (2009) for other galaxies.

The observations of NGC 4593 that were carried out with LUCIFER 1 during the commissioning of the instrument were severely affected by external and internal problems, so that no scientific meaningful spectrum could be extracted, unfortunately. Nonetheless, the data help to understand the current issues and provided important insights on how to fix them in the near future.

LUCIFER 1 will be available for the scientific community within the next months. The control software is ready and working very reliably and above expectations. It allows efficient and save operation of the instrument. The next steps are the integration of the `Scheduler` service into the current design of the *Operation Layer* and the extension for the AO operation of the instrument which is due to start in 2010. Finally, the operation of both LUCIFER instruments together at the LBT has to be supported.

By carrying out the first spectroscopic study using the new D_{CO} index of Mármol-Queraltó et al. (2008) for galaxies, differences have been found between elliptical galaxies and classical bulges. Although the current sample is definitely to small to draw any general results, it is proposed to extend the study of galaxies showing a classical bulge with regard to the CO absorption in the K–band. By comparing the results of CO absorption measurements for many galaxies, the main result of this thesis is that the CO index may be dominated in numerous spiral galaxies by internal physical properties leading to non stellar radiation rather than by the stellar populations.

Appendix A

A typical FITS header

```
SIMPLE  =                    T
BITPIX  =                   32
NAXIS   =                    2
NAXIS1  =                 2048
NAXIS2  =                 2048
COMMENT =
BSCALE  =                  1.0
BZERO   =                  0.0
COMMENT =
COMMENT =
COMMENT =
COMMENT =
MJD-OBS =       54844.50903705 / Modified julian date 'days' of observation end
DATE-OBS= '2009-01-13T12:13:00.8010'/ UT-date of observation end
DATE    = '2009-01-13T12:13:01.4414'/ UT-date of file creation
UT      =           43980.8010 / '12:13:00.8010' UTC (sec) at EOread
LST     =         44755.640000 / local siderial time: 12:25:55 (EOread)
ORIGIN  = 'Mount Graham, MGIO, Arizona'
OBSERVER= 'master'
TELESCOP= 'LBT'
FRATIO  = 'F/15'
INSTRUME= 'Lucifer'
OPTIC   = 'high res.'
ELECGAIN=             4.100000 / electrons/DN
ENOISE  =            12.000000 / electrons/read
ELECTRON= 'MPIA IR-ROelectronics Vers. 2'
STRT-INT=           43959.6450 / '12:12:39.6450' start integration (sec) (UT)
STOP-INT=           43980.7963 / '12:13:00.7963' stop integration (sec) (UT)
OBJECT  = 'M95'
EXPO_NO =                  544 / exposure/read counter
FILENAME= 'luci_20090112_0156.fits'
```

```
TPLNAME = '/home/luci/GEIRS/MACROS/darks.mac'/ macro/template name
TIMER0  =                 1984 / milliseconds
TIMER1  =                 2000 / milliseconds
TIMER2  =                16029 / microseconds
PTIME   =                    2 / pixel-time-base index
READMODE= 'o2.double.corr.read'/ read cycle-type
IDLEMODE= 'break'              / idle to read transition
SAVEMODE= 'o2.double.corr.read'/ save cycle-type
CPAR1   =                    1 / cycle type parameter
ITIME   =            2.000000 / (on chip) integration time [secs]
CTIME   =            4.105992 / read-mode cycle time [secs]
CRATE   =            0.243547 / read-mode cycle rate [Hz]
HCOADDS =                    1 / # of hardware coadds
PCOADDS =                    1 / # of coadded plateaus/periods
SCOADDS =                    5 / # of software coadds
NCOADDS =                    5 / effective coadds (total)
EXPTIME =           10.000000 / total integ. time [secs]
FRAMENUM=                    1 / INTEGRAL OF 5
SKYFRAME= 'unknown'
SAVEAREA= '[1:2048,1:2048]'
SOFTWARE= 'GEIRS Vers. hwplx-r251.sv9b-x04 (Sep 17 2008, 22:35:45)'
COMMENT = 'your comment'
CRVAL1  =                  189.98377500 /
CRVAL2  =                   -5.25700000 /
RA      =          '12:39:56.106000000'/ RA at MJD-OBS
DEC     =         '-05:15:25.210000000'/ DEC at MJD-OBS
RADECSYS=                        'FK5'/
CRPIX1  =                  1025.000000 /
CRPIX2  =                  1050.000000 /
SECPIX1 =                    -0.120000 /
SECPIX2 =                     0.120000 /
CDELT1  =                   0.00003333 /
CDELT2  =                   0.00003333 /
CTYPE1  =                    'RA---TAN'/
CTYPE2  =                    'DEC--TAN'/
CROTA1  =                   306.070000 /
CROTA2  =                   306.070000 /
CD1_1   =                   0.00001962 /
CD1_2   =                   0.00002694 /
CD2_1   =                  -0.00002694 /
CD2_2   =                   0.00001962 /
OBJRA   =                '12 39 56.106'/ RA requested
OBJDEC  =                '-05 15 25.21'/ DEC requested
TELALT  =                    51.823708 / LBT mount altitude at MJD-OBS
TELAZ   =                   173.926315 / LBT mount azimuth at MJD-OBS
PARANGLE=                      -4.9126 / Parallactic angle at start (deg)
POSANGLE=                      53.9300 / position angle at start (deg)
```

```
ROTANGLE=                   66.7085 / rotator angle at start (deg)
M1-X    =                  0.217489 / X pos of PM
M1-Y    =                 -0.876797 / Y pos of PM
M1-Z    =                  0.435177 / Z pos of PM
M1RX    =                -45.722590 / X rot of PM
M1RY    =                 -2.801875 / Y rot of PM
M1RZ    =                  0.000000 / Z roy of PM
M1CTEMP =                 -0.035333 / temp of PM
M2-X    =                 -7.010000 / X pos od SM
M2-Y    =                  0.862000 / Y pos of SM
M2-Z    =                  0.000000 / Z pos of SM
M2RX    =                114.280000 / X rot of SM
M2RY    =                181.800000 / Y rot of SM
M2RZ    =                  0.000000 / Z rot of SM
M2CTEMP =                 -0.031667 / temp of SM
M3TIP   =                  0.000000 / tip of TM
M3TILT  =                  0.000000 / tilt of TM
M3PSTN  =                  0.000000 / position of TM
M3ZROT  =                  0.000000 / Z rot of TM
M3CTEMP =                 -0.001333 / temp of TM
TTEMP201=                  2.699000 / telescope temp at sensor 201
TTEMP202=                  6.182000 / telescope temp at sensor 202
TTEMP203=                  0.546000 / telescope temp at sensor 203
TTEMP204=                  0.369000 / telescope temp at sensor 204
TTEMP205=                  2.494000 / telescope temp at sensor 205
TTEMP206=                  4.403000 / telescope temp at sensor 206
TTEMP207=                  1.394000 / telescope temp at sensor 207
TTEMP208=                  1.302000 / telescope temp at sensor 208
TTEMP209=                  1.312000 / telescope temp at sensor 209
TTEMP210=                  1.394000 / telescope temp at sensor 210
TTEMP301=                  0.220000 / telescope temp at sensor 301
TTEMP302=                  0.145000 / telescope temp at sensor 302
TTEMP303=                 -1.113000 / telescope temp at sensor 303
TTEMP304=                 -0.814000 / telescope temp at sensor 304
TTEMP305=                 -0.131000 / telescope temp at sensor 305
TTEMP306=                 -0.660000 / telescope temp at sensor 306
TTEMP307=                  0.215000 / telescope temp at sensor 307
TTEMP308=                 -0.139000 / telescope temp at sensor 308
TTEMP309=                  0.033000 / telescope temp at sensor 309
TTEMP310=                 -0.659000 / telescope temp at sensor 310
SMTTEMP =                  0.100000 / ambient temp SMT weather station
SMTPRES =                699.691700 / pressure at SMT weather station
SMTHUM  =                 24.100000 / humidity at SMT weather station
SMTDWPT =                -18.402560 / dewpoint at SMT weather station
LBTTEMP =                  0.000000 / ambient temp at LBT weather station
LBTPRES =                  0.000000 / pressure at LBT weather station
LBTHUM  =                  0.000000 / humidity LBT weather station
```

```
LBTDWPT =                    0.000000 / dewpoint LBT weather station
GUIRA   =                '12 40 13.354'/  RA of guide object
GUIDEC  =                '-05 13 26.44'/  DECS of guide object
STATLMP1=                       'OFF'/ Ne
STATLMP2=                       'OFF'/ Ar
STATLMP3=                       'OFF'/ Xe
STATLMP4=                       'OFF'/ HALO 1
STATLMP5=                       'OFF'/ HALO 2
STATLMP6=                       'OFF'/ HALO 3
RACKTEM1=                     284.79 / rack sensor 1 (I-Rack)
RACKLVL1=                      100.0 / output level channel 1
RACKTEM2=                     288.36 / rack sensor 2 (motion control electron
RACKLVL2=                      100.0 / output level channel 2
RACKTEM3=                     287.13 / rack sensor 3 (readout electronics)
RACKLVL3=                      100.0 / output level channel 3
RACKTEM4=                     277.30 / rack sensor 4 (ambient)
RACKLVL4=                   200004.0 / unused
RACKCLG =                        'ON'/ cooling of the electronics rack
PRESSUR1=                  5.0900E-7 / mbar
PRESSUR2=                  6.9000E-7 / mbar
DETTEMP1=                      77.00 / input channel A (detector)
DETTEMP2=                      60.00 / input channel B (cold bench)
INSTEMP1=                      66.30 / sensor 1 (structure bottom)
INSTEMP2=                      70.32 / sensor 2 (focal plane unit)
INSTEMP3=                      68.89 / sensor 3 (getter unit)
INSTEMP4=                      69.17 / sensor 4 (structure top)
INSTEMP5=                      66.96 / sensor 5 (camera unit)
INSTEMP6=                      76.97 / sensor 6 (grating unit)
INSTEMP7=                      76.69 / sensor 7 (MOS unit)
INSTEMP8=                     502.50 / sensor 8 (MOS unit)
M4M1POS =                          0 / position of mirror 4 motor 1
M4M2POS =                          0 / position of mirror 4 motor 2
CAMPOS  =                          2 / position of camera unit
CAMNAME =           'N3.75 Camera'/ camera name
PIXSCALE=                    0.12000 / arcsec/pixel
INSFOCUS=                       -350 / detector focus (steps from reference)
FILTPOS1=                          3 / position of FW1
FILTER1 =                 'clear-1'/ name of filter in FW1
FILTPOS2=                         14 / position of FW2
FILTER2 =                        'K'/ name of filter in FW2
GRATPOS =                          2 / position of the grating unit
GRATNAME=                  'mirror'/ name of the grating unit element
GRATVOLT=                   0.16781 / tilt voltage
GRATWLEN=                'not used'/ central wavelength (microns)
GRATORDE=                'not used'/ used grating diffraction order
GRATLOOP=                    'OPEN'/ regulation loop status
MOSPOS  =          'no mask in use'/ position of the MOS unit
```

```
MASKSLOT=                       'unknown'/ number of the used mask
MASKID  =                       'unknown'/ ID of the used mask
MASKNAME=                       'unknown'/ description of the used mask
PVSTATUS=             'PUPIL_VIEWER_OUT'/ position of pupil viewer
END
```

Appendix B

A template observation script

```
[START_INSTRUMENT_SETUP]
CAMERA                  =N3.75                  # N1.8|N3.75|N30
FILTER_ONE              =clear-1                # names of filters in FW1
FILTER_TWO              =Ks                     # names of filters in FW2
GRATING_UNIT            =mirror                 # mirror|high_dispersion|
                                                # 200_H+K|150_Ks
CENTRAL_WAVELENGTH      =                       # value or leave blank
ORDER                   =                       # value or leave blank
MASK                    =                       # IDxxx = mask id
                                                # NBxx  = cabinet position
                                                # leave blank for no mask
MASK_POSITION           =no_mask_in_use         # mask_in_fpu|
                                                # mask_in_turnout|
                                                # no_mask_in_use
FLEXURE_COMP            =on                     # enable flexure compens
[END_INSTRUMENT_SETUP]


[START_TELESCOPE_SETUP]
TARGET_NAME             =NGC4593                # any name or leave blank
TARGET_COORD            =12 39 39.4 -05 20 39   # hh mm ss dd mm ss
GUIDE_NAME              =                       # any name or leave blank
GUIDE_COORD             =12 39 46.4 -05 18 56   # hh mm ss dd mm ss
ROT_ANG                 =50                     # in degrees
ROT_MODE                =position               # position|parallactic|idle
[END_TELESCOPE_SETUP]


[START_READOUT_SETUP]
DIT                     =10                     # any positive value
NDIT                    =2                      # any positive value
NINT                    =2                      # any positive value
```

```
ROE_MODE                =mer                        # o2dcr|mer (#reads)
SAVE_MODE               =INTEGRATED                 # integrated|cube|normal
SAVE_PATH               =/data/luci/20090211        # absolute save path
FILENAME                =script_                    # root of the filenames
[END_READOUT_SETUP]


[START_OBSERVING_SETUP]
OFFSET_TYPE             =relative                   # relative|absolute
COORD_SYS               =DETXY                      # DETXY|RADEC
ACQUISITION             =20 20                      # offset the telescope and
                                                    # wait for acknowledge
OFFSET                  =10 00                      # offset the telescope
OFFSET                  =10 00 45.6                 # same as above
                                                    # but with a rotator angle
                                                    # offset of 45.6

# a new telescope pointing can be specified with the following command:
POINTING                =06 40 40 21 23 30 06 40 59 21 24 00
#                        hh mm ss dd mm ss hh mm ss dd mm ss (guide star)

FOCUS                   =-420 -200 10               # adjust focus from -420
                                                    # to 200 in steps of 10
FLAT                                                # readout the detector
                                                    # for flats, no parameters
[END_OSERVING_SETUP]
```

Appendix C

ISAAC spectra of the galaxies

The following pages show spectra of the different galaxies that are examined in this thesis. For each galaxy, several spectra could be extracted along the slit. The number of extracted spectra varied from three (the center and one adjacent regions on each side) for NGC 1032 and NGC 4593, to seven spectra for M 95 (the center and three adjacent regions on each side). The regions — or bins — were chosen to provide for equal spatial separation with a reasonable S/N after extraction. If the S/N dropped significantly, the bin size was doubled and this bin was then the outermost one. The only exception is NGC 4593 for which the central bin was chosen to cover only the innermost region hosting the AGN and the other two bins extending to the edges of the detectable emission.

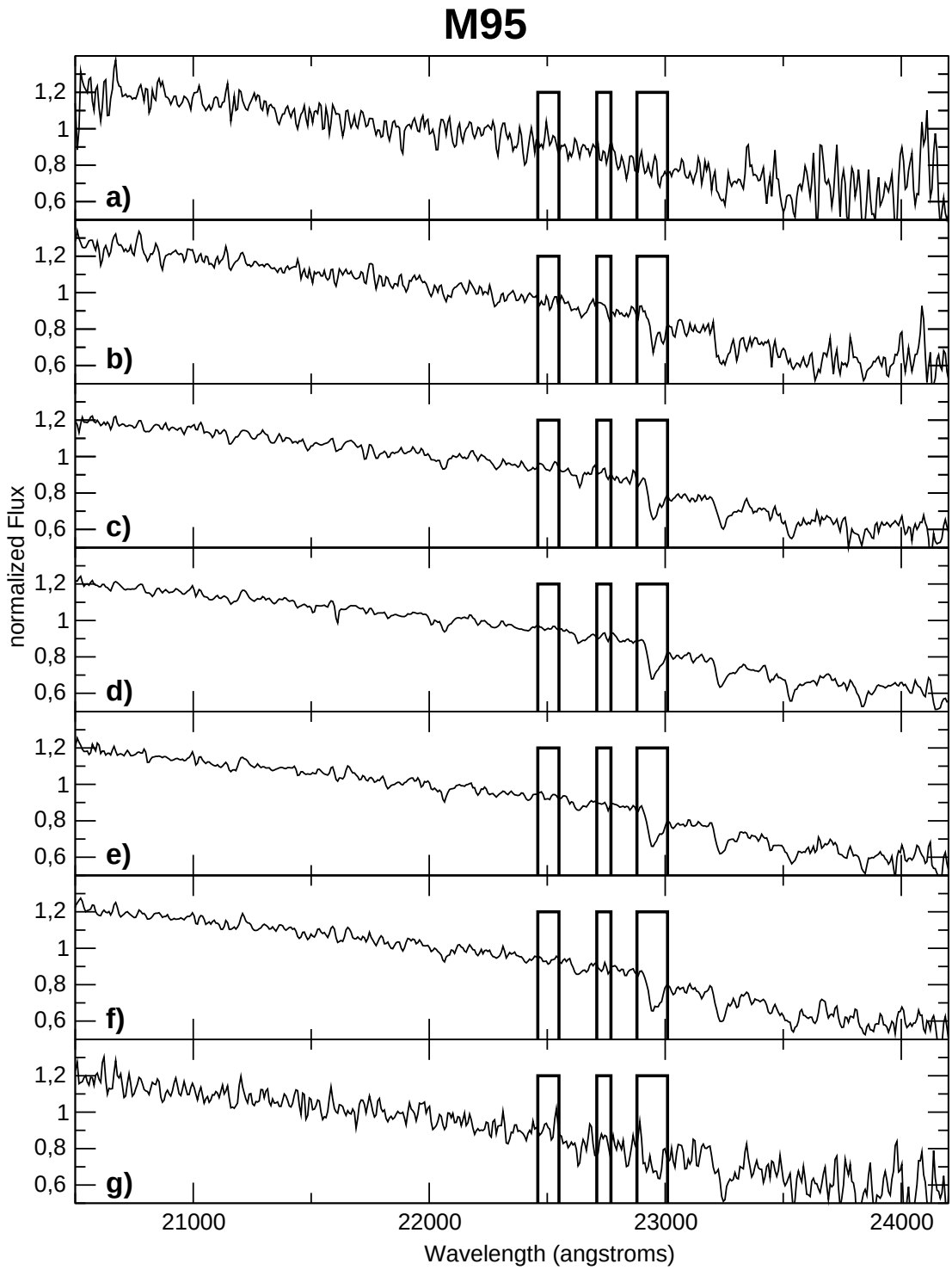


Figure C.1: Spectra of M 95 for various distances from the center. The spectra have been normalized to $F(\lambda = 22000\,\text{Å}) \equiv 1$. The absorption and the two continuum bands that have been used for the D_{CO} index calculation are indicated. The projected mean distances from the center are a) -772 pc, b) -441 pc, c) -221 pc, d) 0 pc, e) 221 pc, f) 441 pc, and g) 772 pc.

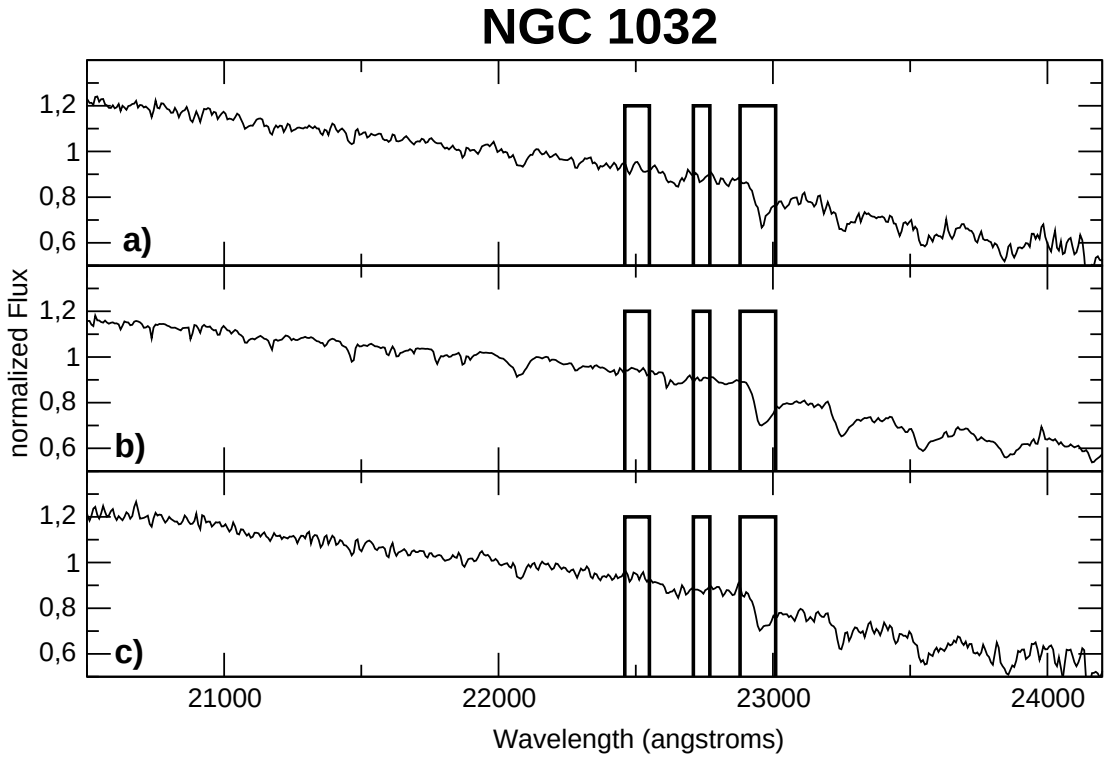


Figure C.2: Spectra of NGC 1032 for various distances from the center. The spectra have been normalized to F($\lambda = 22000$ Å) $\equiv 1$. The absorption and the two continuum bands that have been used for the D_{CO} index calculation are indicated. The projected mean distances from the center are a) -723 pc, b) 0 pc, and c) 723 pc.

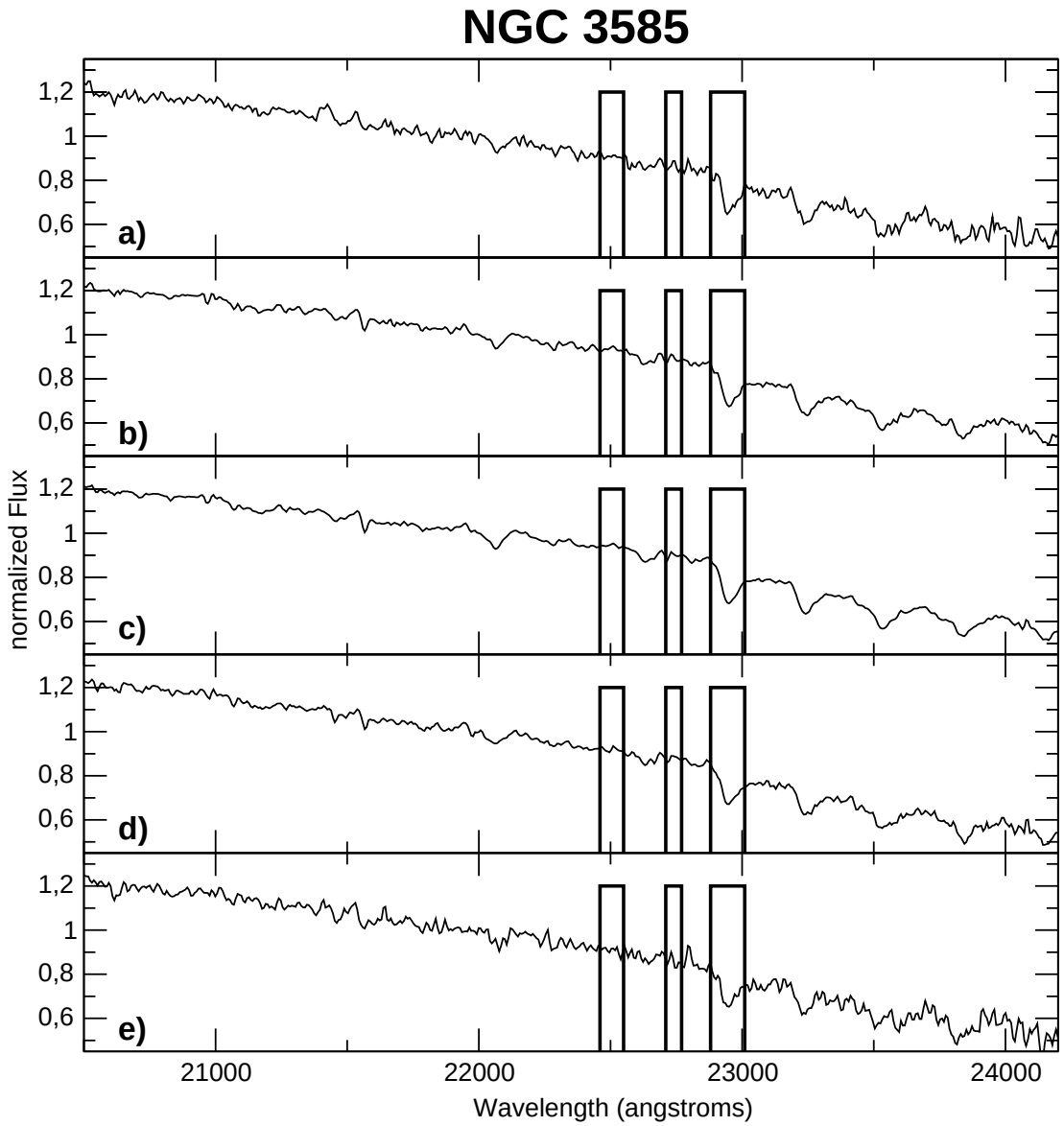


Figure C.3: Spectra of NGC 3583 for various distances from the center. The spectra have been normalized to $F(\lambda = 22000\,\text{Å}) \equiv 1$. The absorption and the two continuum bands that have been used for the D_{CO} index calculation are indicated. The projected mean distances from the center are a) -853 pc, b) -341 pc, c) 0 pc, d) 341 pc, and e) 853 pc.

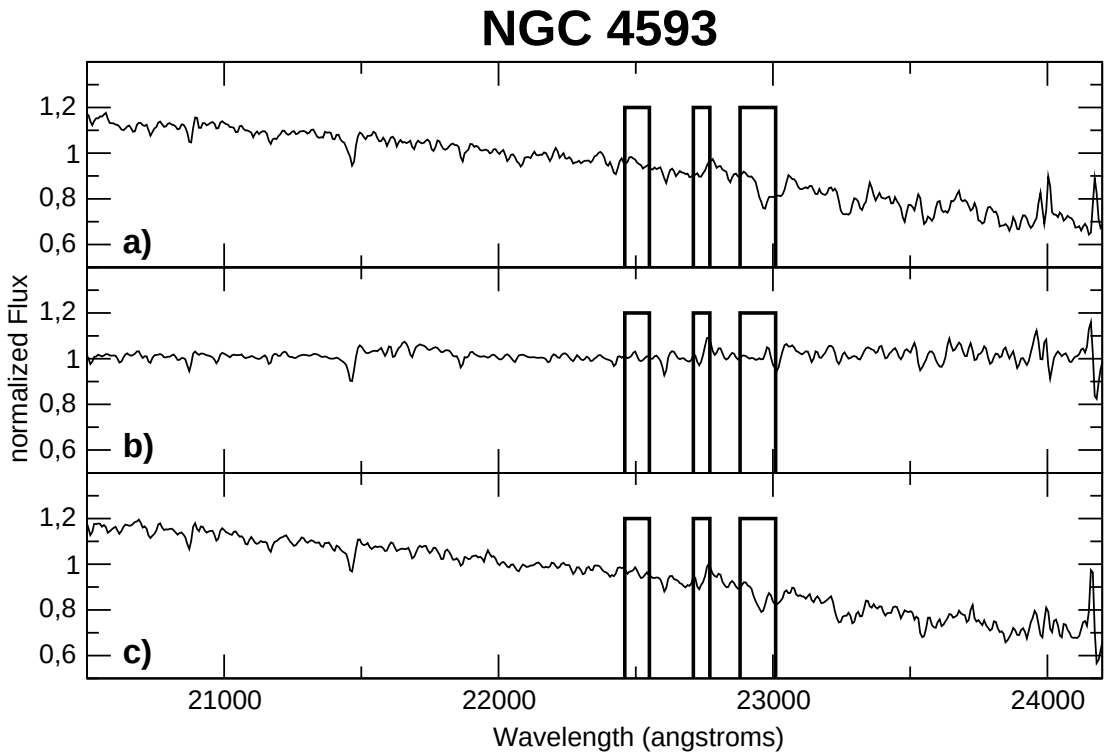


Figure C.4: Spectra of NGC 4593 for various distances from the center. The spectra have been normalized to $F(\lambda = 22000\,\text{Å}) \equiv 1$. The absorption and the two continuum bands that have been used for the D_{CO} index calculation are indicated. The projected mean distances from the center are a) -742 pc, b) 0 pc, and c) 742 pc. Note that the "absorption feature" near 21500 Å is a residual of the telluric correction.

Appendix D

Template spectrum of an elliptical galaxy with an old stellar population

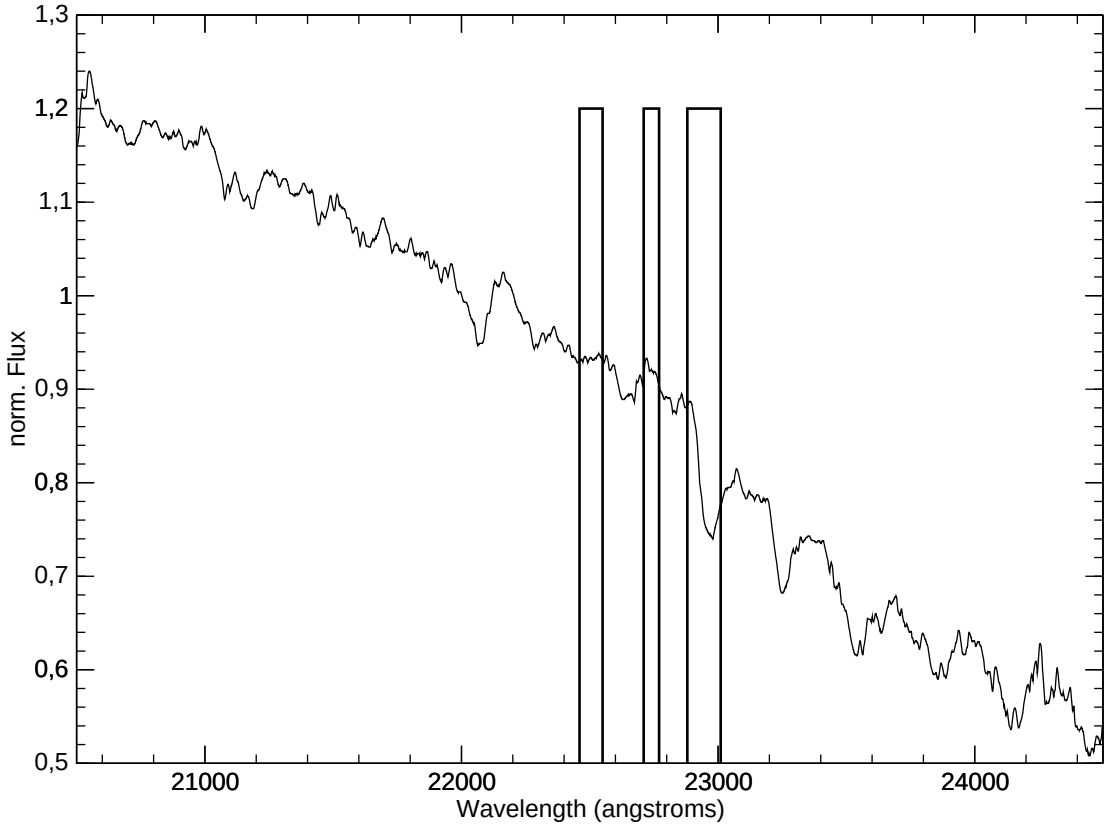


Figure D.1: The template spectrum of an elliptical galaxy from Cesetti et al. (2009) for a mean age of the stellar population of log T = 9.98. The absorption and two continuum bands used to calculate the D_{CO} values are indicated. The spectrum is normalized to $F(\lambda = 22000\,\text{Å}) \equiv 1$.

Abbreviations

ADC atmospheric dispersion corrector

AGB asymptotic giant branch

AGN active galactic nucleus

AIRUB Astronomisches Institut Ruhr–Universität–Bochum

AO adaptive optics

b/p box/peanut

CCD Charge Coupled Device

CORBA Common Object Request Broker Architecture

dcr double correlated read

DIT detector integration time

DSS Digital Sky Survey

ESO European Southern Observatory

EW equivalent width

FITS Flexible Image Transport System

FOV field of view

GEIRS GEneric InfraRed Software

GUI graphical user interface

HAWAII2 HgCdTe Astronomical Wide Area Infrared Imager-2

HIRAMO HIgh Resolution Analog Measurement Output board

HST Hubble Space Telescope

HUDF Hubble Ultra Deep Field

ICE Internet Communications Engine

IIF Instrument Interface

IRAF Image Reduction and Analysis Facility

ISAAC Infrared Spectrometer and Array Camera

JVM Java Virtual Machine

JWST James Webb Space Telescope

LBT Large Binocular Telescope

LBTI LBT Intefrerometer

LBC Large Binocular Camera

lcsp LUCIFER control software package

LINC/NIRVANA LBT INteferometric Camera / Near-IR Visible Adaptive iNterferometer for Astronomy

LSS long slit spectroscopy

LSW Landessternwarte Heidelberg

LUCIFER LBT NIR spectroscopic Utility with Camera and Integral-Field Unit for Extragalactic Research

mas milliarcsecond

MCU Motion Control Unit

MDR morphology density relation

mer multiple end point read

MODS Multi Object Double Spectrograph

MOS multi object spectroscopy

MPIA Max-Planck-Institut für Astronomie, Heidelberg

MPE Max-Planck-Institut für Extraterrestrische Physik, Garching

NED NASA/IPAC Extragalactic Database

NGC New General Catalogue

NIR near infrared

OPT Observation Preparation Tool

P2PP Phase 2 Proposal Preparation

PEPSI Potsdam Echelle Polarimetric and Spectroscopic Instrument

RMI remote method invocation

S/N signal to noise ratio

SAURON Spectroscopic Areal Unit for Research on Optical Nebulae

SDSS Sloan Digital Sky Survey

SFR star formation rate

TCS Telescope Control Software

VLT Very Large Telescope

Bibliography

Andredakis, Y. C., Peletier, R. F., & Balcells, M. 1995, MNRAS, 275, 874

Aronica, G., Athanassoula, E., Bureau, M., Bosma, A., Dettmar, R.-J., Vergani, D., & Pohlen, M. 2003, Ap&SS, 284, 753

Arp, H. 1966, ApJS, 14, 1

Athanassoula, E. 2003, MNRAS, 341, 1179

—. 2005, MNRAS, 358, 1477

Athanassoula, E. 2008, in IAU Symposium, Vol. 245, IAU Symposium, 93–102

Athanassoula, E., & Martínez-Valpuesta, I. 2008, in Astronomical Society of the Pacific Conference Series, Vol. 390, Pathways Through an Eclectic Universe, ed. J. H. Knapen, T. J. Mahoney, & A. Vazdekis, 454–+

Bacon, R., Copin, Y., Monnet, G., Miller, B. W., Allington-Smith, J. R., Bureau, M., Carollo, C. M., Davies, R. L., Emsellem, E., Kuntschner, H., Peletier, R. F., Verolme, E. K., & de Zeeuw, P. T. 2001, MNRAS, 326, 23

Baldry, I. K., Glazebrook, K., Brinkmann, J., Ivezić, Ž., Lupton, R. H., Nichol, R. C., & Szalay, A. S. 2004, ApJ, 600, 681

Balogh, M. L., Baldry, I. K., Nichol, R., Miller, C., Bower, R., & Glazebrook, K. 2004, ApJ, 615, L101

Barnes, J. E. 1992, ApJ, 393, 484

Barnes, J. E., & Hernquist, L. 1996, ApJ, 471, 115

Baugh, C. M., Cole, S., & Frenk, C. S. 1996, MNRAS, 283, 1361

Binney, J. 1978, MNRAS, 183, 501

Bureau, M., Aronica, G., Athanassoula, E., Dettmar, R.-J., Bosma, A., & Freeman, K. C. 2006, MNRAS, 370, 753

Bureau, M., & Freeman, K. C. 1999, AJ, 118, 126

Cesetti, M., Ivanov, V. D., Morelli, L., Pizzella, A., Buson, L., Corsini, E. M., Dalla Bontà, E., Stiavelli, M., & Bertola, F. 2009, A&A, 497, 41

Conselice, C. J. 2006, MNRAS, 373, 1389

Davidge, T. J., Beck, T. L., & McGregor, P. J. 2008, ApJ, 677, 238

De Lucia, G., Springel, V., White, S. D. M., Croton, D., & Kauffmann, G. 2006, MNRAS, 366, 499

de Vaucouleurs, G. 1948, Annales d'Astrophysique, 11, 247

—. 1963, ApJS, 8, 31

de Vaucouleurs, G., de Vaucouleurs, A., Corwin, Jr., H. G., Buta, R. J., Paturel, G., & Fouque, P. 1991, Third Reference Catalogue of Bright Galaxies (Volume 1-3, XII, 2069 pp. 7 figs.. Springer-Verlag Berlin Heidelberg New York)

Debattista, V. P., Mayer, L., Carollo, C. M., Moore, B., Wadsley, J., & Quinn, T. 2006, ApJ, 645, 209

Doyon, R., Joseph, R. D., & Wright, G. S. 1994, ApJ, 421, 101

Dressler, A. 1980, ApJ, 236, 351

Dressler, A., Oemler, A. J., Couch, W. J., Smail, I., Ellis, R. S., Barger, A., Butcher, H., Poggianti, B. M., & Sharples, R. M. 1997, ApJ, 490, 577

Driver, S. P., Allen, P. D., Graham, A. W., Cameron, E., Liske, J., Ellis, S. C., Cross, N. J. G., De Propris, R., Phillipps, S., & Couch, W. J. 2006, MNRAS, 368, 414

Drory, N., & Fisher, D. B. 2007, ApJ, 664, 640

Eggen, O. J., Lynden-Bell, D., & Sandage, A. R. 1962, ApJ, 136, 748

Elmegreen, D. M., Chromey, F. R., Santos, M., & Marshall, D. 1997, AJ, 114, 1850

Englmaier, P., & Gerhard, O. 1997, MNRAS, 287, 57

Erwin, P. 2008, in IAU Symposium, Vol. 245, IAU Symposium, 113–116

Faber, S. M., & Jackson, R. E. 1976, ApJ, 204, 668

Fisher, D. B. 2006, ApJ, 642, L17

Fisher, D. B., & Drory, N. 2008, AJ, 136, 773

Frogel, J. A., Stephens, A., Ramírez, S., & DePoy, D. L. 2001, AJ, 122, 1896

Gadotti, D. A. 2009, MNRAS, 393, 1531

Gardner, J. P., Mather, J. C., Clampin, M., Doyon, R., Greenhouse, M. A., Hammel, H. B., Hutchings, J. B., Jakobsen, P., Lilly, S. J., Long, K. S., Lunine, J. I., McCaughrean, M. J., Mountain, M., Nella, J., Rieke, G. H., Rieke, M. J., Rix, H.-W., Smith, E. P., Sonneborn, G., Stiavelli, M., Stockman, H. S., Windhorst, R. A., & Wright, G. S. 2006, Space Science Reviews, 123, 485

Gott, III, J. R., & Thuan, T. X. 1976, ApJ, 204, 649

Grosso, W. 2002, Java RMI, 1st edn. (O'Reilly & Associates, Inc.)

Hägele, G. F., Díaz, Á. I., Cardaci, M. V., Terlevich, E., & Terlevich, R. 2007, MNRAS, 378, 163

Hanisch, R. J., Farris, A., Greisen, E. W., Pence, W. D., Schlesinger, B. M., Teuben, P. J., Thompson, R. W., & Warnock, III, A. 2001, A&A, 376, 359

Hathi, N. P., Ferreras, I., Pasquali, A., Malhotra, S., Rhoads, J. E., Pirzkal, N., Windhorst, R. A., & Xu, C. 2009, ApJ, 690, 1866

Hill, J. M., & Salinari, P. 2004, in Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, Vol. 5489, Ground-based Telescopes. Edited by Oschmann, Jacobus M., Jr. Proceedings of the SPIE, Volume 5489, pp. 603-614 (2004)., ed. J. M. Oschmann, Jr., 603–614

Holden, B. P., Illingworth, G. D., Franx, M., Blakeslee, J. P., Postman, M., Kelson, D. D., van der Wel, A., Demarco, R., Magee, D. K., Tran, K.-V., Zirm, A., Ford, H., Rosati, P., & Homeier, N. 2007, ApJ, 670, 190

Hubble, E. P. 1926, ApJ, 64, 321

Illingworth, G., & Schechter, P. L. 1982, ApJ, 256, 481

Ivanov, V. D., Rieke, G. H., Groppi, C. E., Alonso-Herrero, A., Rieke, M. J., & Engelbracht, C. W. 2000, ApJ, 545, 190

James, P. A., & Mobasher, B. 1999, MNRAS, 306, 199

James, P. A., & Seigar, M. S. 1999, A&A, 350, 791

Jogee, S., Scoville, N., & Kenney, J. D. P. 2005, ApJ, 630, 837

Kennicutt, Jr., R. C. 1998a, ARA&A, 36, 189

—. 1998b, ApJ, 498, 541

Kitchin, C. R. 2008, Astrophysical Techniques, 5th edn. (CRC Press, Taylor & Francis Group)

Kleinmann, S. G., & Hall, D. N. B. 1986, ApJS, 62, 501

Knapp, G. R., Kerr, F. J., & Williams, B. A. 1978, ApJ, 222, 800

Kollatschny, W., & Fricke, K. J. 1985, A&A, 143, 393

Kormendy, J. 1982, ApJ, 257, 75

Kormendy, J. 1993, in IAU Symposium, Vol. 153, Galactic Bulges, ed. H. Dejonghe & H. J. Habing, 209–+

Kormendy, J. 2008, in IAU Symposium, Vol. 245, IAU Symposium, 107–112

Kormendy, J., Cornell, M. E., Block, D. L., Knapen, J. H., & Allard, E. L. 2006, ApJ, 642, 765

Kormendy, J., & Illingworth, G. 1982, ApJ, 256, 460

Kormendy, J., & Kennicutt, Jr., R. C. 2004, ARA&A, 42, 603

Kuntschner, H., Emsellem, E., Bacon, R., Bureau, M., Cappellari, M., Davies, R. L., de Zeeuw, P. T., Falcón-Barroso, J., Krajnović, D., McDermid, R. M., Peletier, R. F., & Sarzi, M. 2006, MNRAS, 369, 497

Lehmitz, M. 2004, LUCIFER Final Design Report Control Electronics, "LBT-LUCIFER-TRE-010"

—. 2005, LUCIFER Technical Manual: High resolution analog mesurement and output board, HIRAMO

Léna, P. 1988, Observational Astrophysics, 1st edn. (Springer)

Ligori, S. 2004, LUCIFER Final Design Report Detector Subsystem, "LBT-LUCIFER-TRE-012"

Lütticke, R., Dettmar, R.-J., & Pohlen, M. 2000a, A&AS, 145, 405

—. 2000b, A&A, 362, 435

Lütticke, R., Pohlen, M., & Dettmar, R.-J. 2004, A&A, 417, 527

MacArthur, L. A., González, J. J., & Courteau, S. 2009, MNRAS, 369

Mandel, H. G., Appenzeller, I., Seifert, W., Baumeister, H., Dettmar, R.-J., Feiz, C., Gemperlein, H., Germeroth, A., Grimm, B., Heidt, J., Herbst, T., Hofmann, R., Jütte, M., Knierim, V., Laun, W., Luks, T., Lehmitz, M., Lenzen, R., Polsterer, K., Quirrenbach, A., Rohloff, R.-R., Rosenberger, J., Weiser, P., & Weisz, H. 2006, in Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, Vol. 6269, Ground-based and Airborne Instrumentation for Astronomy. Edited by McLean, Ian S.; Iye, Masanori. Proceedings of the SPIE, Volume 6269, pp. 62693F (2006).

Mannucci, F., Basile, F., Poggianti, B. M., Cimatti, A., Daddi, E., Pozzetti, L., & Vanzi, L. 2001, MNRAS, 326, 745

Mármol-Queraltó, E., Cardiel, N., Cenarro, A. J., Vazdekis, A., Gorgas, J., Pedraz, S., Peletier, R. F., & Sánchez-Blázquez, P. 2008, A&A, 489, 885

Martin, H. M., Brusa Zappellini, G., Cuerden, B., Miller, S. M., Riccardi, A., & Smith, B. K. 2006, in Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, Vol. 6272, Advances in Adaptive Optics II. Edited by Ellerbroek, Brent L.; Bonaccini Calia, Domenico. Proceedings of the SPIE, Volume 6272, pp. 62720U (2006).

Mason, E. 2007, Paranal Science Operations INSTRUMENT data reduction cookbook, VLT-MAN-ESO-14100-4031 Issue 80.0

Mason, E., & Schmidtobreick, L. 2009, ISAAC User Manual, VLT-MAN-ESO-14100-0841 Issue 84.0

Meller, H. 2002, Archäologie in Sachsen-Anhalt, 1, 7

Michard, R. 2006, A&A, 449, 519

Mie, G. 1908, Ann. Phys., 330, 377

Mobasher, B., & James, P. A. 2000, MNRAS, 316, 507

Morgan, W. W., & Mayall, N. U. 1957, PASP, 69, 291

Muhlack, T. 2006, Der LUCIFER NIR Spektrograph: Detektor und Ausleseprozess, Diplomarbeit Ruhr–Universität–Bochum

Naab, T., & Ostriker, J. P. 2009, ApJ, 690, 1452

Origlia, L., Goldader, J. D., Leitherer, C., Schaerer, D., & Oliva, E. 1999, ApJ, 514, 96

Patsis, P. A., Skokos, C., & Athanassoula, E. 2002, MNRAS, 337, 578

Peletier, R. F., Ganda, K., Falcón-Barroso, J., Bacon, R., Cappellari, M., Davies, R. L., de Zeeuw, P. T., Emsellem, E., Krajnović, D., Kuntschner, H., McDermid, R. M., Sarzi, M., & van de Ven, G. 2008, in IAU Symposium, Vol. 245, IAU Symposium, 271–276

Polsterer, K. L. in prep., PhD thesis, Ruhr–Universtität Bochum

Postman, M., Franx, M., Cross, N. J. G., Holden, B., Ford, H. C., Illingworth, G. D., Goto, T., Demarco, R., Rosati, P., Blakeslee, J. P., Tran, K.-V., Benítez, N., Clampin, M., Hartig, G. F., Homeier, N., Ardila, D. R., Bartko, F., Bouwens, R. J., Bradley, L. D., Broadhurst, T. J., Brown, R. A., Burrows, C. J., Cheng, E. S., Feldman, P. D., Golimowski, D. A., Gronwall, C., Infante, L., Kimble, R. A., Krist, J. E., Lesser, M. P., Martel, A. R., Mei, S., Menanteau, F., Meurer, G. R., Miley, G. K., Motta, V., Sirianni, M., Sparks, W. B., Tran, H. D., Tsvetanov, Z. I., White, R. L., & Zheng, W. 2005, ApJ, 623, 721

Puxley, P. J., Doyon, R., & Ward, M. J. 1997, ApJ, 476, 120

Rhoads, J. E. 1998, AJ, 115, 472

Roberts, M. S., & Haynes, M. P. 1994, ARA&A, 32, 115

Sandage, A. 1961, The Hubble atlas of galaxies (Washington: Carnegie Institution, 1961)

—. 2005, ARA&A, 43, 581

Sarzi, M., Rix, H.-W., Shields, J. C., Ho, L. C., Barth, A. J., Rudnick, G., Filippenko, A. V., & Sargent, W. L. W. 2005, ApJ, 628, 169

Schimmelmann, J. 2007, Benutzerunterützte Planung und Durchführung astronomischer nah-infrarot Beobachtungen unter Berücksichtigung spezieller Optimierungsprobleme des LUCIFER Instruments, Diplomarbeit Universität Dortmund

Schlosser, W. 2002, Archäologie in Sachsen-Anhalt, 1, 21

Schmidt, M. 1959, ApJ, 129, 243

Seifert, W. 2002, LUCIFER Final Design Report Optics, "LBT-LUCIFER-TRE-009"

Seltmann, A., A., S., & W., S. 2004, LUCIFER Final Design Report Mechanics, "LBT-LUCIFER-TRE-014"

Sersic, J. L. 1968, Atlas de galaxias australes, ed. J. L. Sersic

Seyfert, C. K. 1943, ApJ, 97, 28

Silva, D. R., Kuntschner, H., & Lyubenova, M. 2008, ApJ, 674, 194

Smith, G. P., Treu, T., Ellis, R. S., Moran, S. M., & Dressler, A. 2005, ApJ, 620, 78

Springel, V., & Hernquist, L. 2005, ApJ, 622, L9

Strateva, I., Ivezić, Ž., Knapp, G. R., Narayanan, V. K., Strauss, M. A., Gunn, J. E., Lupton, R. H., Schlegel, D., Bahcall, N. A., Brinkmann, J., Brunner, R. J., Budavári, T., Csabai, I., Castander, F. J., Doi, M., Fukugita, M., Győry, Z., Hamabe, M., Hennessy, G., Ichikawa, T., Kunszt, P. Z., Lamb, D. Q., McKay, T. A., Okamura, S., Racusin, J., Sekiguchi, M., Schneider, D. P., Shimasaku, K., & York, D. 2001, AJ, 122, 1861

Terlevich, A. I., & Forbes, D. A. 2002, MNRAS, 330, 547

Thomas, D., & Davies, R. L. 2006, MNRAS, 366, 510

van der Wel, A., Holden, B. P., Franx, M., Illingworth, G. D., Postman, M. P., Kelson, D. D., Labbé, I., Wuyts, S., Blakeslee, J. P., & Ford, H. C. 2007, ApJ, 670, 206

Wells, D. C., Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 363

York, D. G., Adelman, J., Anderson, Jr., J. E., Anderson, S. F., Annis, J., Bahcall, N. A., Bakken, J. A., Barkhouser, R., Bastian, S., Berman, E., Boroski, W. N., Bracker, S., Briegel, C., Briggs, J. W., Brinkmann, J., Brunner, R., Burles, S., Carey, L., Carr, M. A., Castander, F. J., Chen, B., Colestock, P. L., Connolly, A. J., Crocker, J. H., Csabai, I., Czarapata, P. C., Davis, J. E., Doi, M., Dombeck, T., Eisenstein, D., Ellman, N., Elms, B. R., Evans, M. L., Fan, X., Federwitz, G. R., Fiscelli, L., Friedman, S., Frieman, J. A., Fukugita, M., Gillespie, B., Gunn, J. E., Gurbani, V. K., de Haas, E., Haldeman, M., Harris, F. H., Hayes, J., Heckman, T. M., Hennessy, G. S., Hindsley, R. B., Holm, S., Holmgren, D. J., Huang, C.-h., Hull, C., Husby, D., Ichikawa, S.-I., Ichikawa, T., Ivezić, Ž., Kent, S., Kim, R. S. J., Kinney, E., Klaene, M., Kleinman, A. N., Kleinman, S., Knapp, G. R., Korienek, J., Kron, R. G., Kunszt, P. Z., Lamb, D. Q., Lee, B., Leger, R. F., Limmongkol, S., Lindenmeyer, C., Long, D. C., Loomis, C., Loveday, J., Lucinio, R., Lupton, R. H., MacKinnon, B., Mannery, E. J., Mantsch, P. M., Margon, B., McGehee, P., McKay, T. A., Meiksin, A., Merelli, A., Monet, D. G., Munn, J. A., Narayanan, V. K., Nash, T., Neilsen, E., Neswold, R., Newberg, H. J., Nichol, R. C., Nicinski, T., Nonino, M., Okada, N., Okamura, S., Ostriker, J. P., Owen, R., Pauls, A. G., Peoples, J., Peterson, R. L., Petravick, D., Pier, J. R., Pope, A., Pordes, R., Prosapio, A., Rechenmacher, R., Quinn, T. R., Richards, G. T., Richmond, M. W., Rivetta, C. H., Rockosi, C. M., Ruthmansdorfer, K., Sandford, D., Schlegel, D. J., Schneider, D. P., Sekiguchi, M., Sergey, G., Shimasaku, K., Siegmund, W. A., Smee, S., Smith, J. A., Snedden, S., Stone, R., Stoughton, C., Strauss, M. A., Stubbs, C., SubbaRao, M., Szalay, A. S., Szapudi, I., Szokoly, G. P., Thakar, A. R., Tremonti, C., Tucker, D. L., Uomoto, A., Vanden Berk, D., Vogeley, M. S., Waddell, P., Wang, S.-i., Watanabe, M., Weinberg, D. H., Yanny, B., & Yasuda, N. 2000, AJ, 120, 1579

Acknowledgements

First of all my sincere gratitude goes to my supervisor Prof. Dr. Ralf–Jürgen Dettmar for giving me the opportunity to write this thesis and for his various comments and suggestions throughout the last years. This significantly improved the thesis and lead to the current version. Furthermore I would like to thank Prof. Dr. Susanne Hüttemeister for her offer to be the second referee.

I owe Dr. Marcus Jütte my deep gratitude for his ongoing support and inspiring discussions within the LUCIFER software team during the past years. His motivating approaches to various problems which were encountered while the project moved on helped a lot in not loosing the aims of the thesis out of sight. In many long nights at the LBT during the various commissioning phases of LUCIFER 1, he was the guarantor for the good atmosphere within the commissioning team with his refreshing personality.

I would like to especially thank the core LUCIFER commissioning team, particularly Dr. Walter Seifert, Dr. Nancy Ageorges and Michael Lehmitz for their willingness to answer any technical or scientific questions that arose and for providing the necessary data while preparing the software for LUCIFER, making the instrument ready for the LBT and during the commissioning at the telescope. The commissioning of LUCIFER 1 was a life changing experience for me and it would have been much more difficult without the support and encouragement of you all. Thank you very much, "never seen this before".

Kai Polsterer opened up a new perspective in software development which definitely helped to improve the structure and the stability of the software package. I thank him for introducing me to this kind thinking.

Dr. Giuseppe Aronica and Dr. Clemens Trachternach are thanked for enlightening discussions about astronomy and physics during the last couple of years. Both also helped to improve the text of the thesis significantly by their thorough and accurate proof reading of the various drafts of the document. Many other present and former colleagues of the AIRUB also contributed substantially to this thesis, by solving computer and network problems, administrative tasks, and motivating discussions. This includes Tim Falkenbach, Dr. Olaf Schmithüsen, Dr. Eva Jütte, Birgitta Burggraf, Dr. Janine van Eymeren, and Dr. Nicola Bennert, although this list is definitely not complete.

Finally, I am deeply indebted to my family, especially my beloved wife Elena, who often had to wait for me till late in the evening and to live without me for several weeks during the last year. I also thank my mother, my sisters and my brother for their continuing support over many years. All of them believed in me and without them this work would not have been possible in the present form.

This research has made use of several astronomical databases: the NASA Extragalactic Database (NED) operated by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration, the NASAs Astrophysics Data System (ADS) Bibliographic Services, as well as the Online Digitized Sky Survey 2 (DSS2) server at the ESO/STECF Archive produced by the Space Telescope Science Institute through its Guide Star Survey group. Additionally, NSO/Kitt Peak FTS data used here were produced by NSF/NOAO. This work is supported by the German federal department for education and research (BMBF) under the project numbers: 05 AL2VO1/8, 05 AL2EIB/4, 05 AL2EEA/1, 05 AL2PCA/5, 05 AL5VH1/5, 05 AL5PC1/1, and 05 A08PCA.

Curriculum Vitae

	Personal details
Name	Volker Knierim
Address	Buscheyplatz 17
	44801 Bochum
Nationality	german
Date of Birth	30.06.1975 in Ibbenbüren
	Education
1981–1985	"Paul–Gerhard–Schule", Mettingen (primary school)
1985–1994	"Kardinal–von–Galen Gymnasium", Mettingen (secondary school)
Jun. 1994	Abitur (A-levels) - University entrance qualification
1995–2003	Study of Physics at the Ruhr–Universität–Bochum
1999–2000	Study of Mathematical Physics at the
	"National University of Ireland (NUI) Galway" (two semsters abroad study)
Dec. 2003	Diploma in Physics
2004–2009	PhD at the "Astronomisches Institut Ruhr–Universität–Bochum"