Manual: Performance of wafer-fused VECSEL under high power operation

Stefan Keller

February 26, 2015

Abstract

`github.com/stefantkeller/VECSELsetup`

Contents

0 Refresher

This section is for if your computer is already set up, the measurement routine is adapted to your needs, and you simply want to refresh your memory how to get (re)started.

1. record a measurement with `routine_measurement.py`
   i.e. adjust the variables `samplename`, `path_to_meas_folder`, `heatsink_start`, `pump_end`, `use_spectrometer`, `shuffle_pump`
   (as explained in section 2)

2. calibrate the setup with `routine_calibration.py`
   If you still have a valid calibration you can skip this. But if you didn't measure for a while, you probably want to recalibrate! (as explained in section 3)

3. evaluate the measurement with `light_light_to_file.py` specify the path to the measurement logfile `logfile` and the path to the calibration data `calib_folder`. This script returns a file that you can open for example with Excel; the top row states all sorts of information, what were the measurement conditions, and what the columns mean at all.

1 Introduction

1.1 Installation

This is a Python library. If you don't have Python installed yet, I recommend you install "Anaconda" (`http://continuum.io/downloads`) for Python 2.7 (it's free!). Also, you have to have installed the drivers from National Instrument (`ni.com`), in order to communicate with GPIB (etc... (NI drivers are a mess, what exactly you need, you have to figure out yourself)).

To install this library from github,

1. click on "Download ZIP",

2. extract the .zip

3. copy the folder to where ever you want it

4. adjust the PYTHONPATH

(at least, that's the proper way to do it, I guess). And, if you are familliar with git ... you know what to do.

If you don't care about "proper" and you simply want it to work (I don't know though what this breaks along the way...): after extracting the .zip, copy the folder to (something like; if you work with Anaconda as recommended above)

- Windows: `C:\Anaconda\Lib\site-packages`
- Mac: `/Users/yourusername/anaconda/Lib/python2.7/site-packages`

this path is already in the Pythonpath, and Python will find it.

For measurements (the stuff in `meas/`) you need pyvisa
(`https://pyvisa.readthedocs.org/en/master/`).
For evaluation (the stuff in `eval/`) you need to also install errorvalues
(`https://github.com/stefantkeller/errorvalues`).
Install it with the same procedure as listed above. If you intend to use the calibration in `exp/eval/calibration.py` as it is, this is going to write an automated report (tuck

Figure 1.1: PyScripter is a convenient tool to work with python scripts. When you have opened one of the examples from the folder `exp/` you can simply click on the launch button, the scripts starts and writes the output in one of the subwindows – try it out, should be selfexplanatory. The stop button lets you to interrupt the script if required. For example, `exp/meas/temp_logger.py` logs the heat sink temperature in an infinite loop; in order to stop the logging you have to interrupt it. This interruption raises a `KeyboardInterrupt` error. This is supposed to happen.

together some plots) as a pdf; ready for print. This relys on LaTeX(`pdflatex`) to be installed on your system.

If measurement and evaluation happens on two different computers (recommended) you probably install only those dependencies that you need. That's ok, the error handling in `__init__.py` catches `ImportError`s and ignores them. However, if nothing happens at all, you might want to check whether you have installed, what you're supposed to have installed (maybe everything is caught and subsequentially ignored?).

1.2 File hierarchy

The different folders have different purposes:

- meas, setup control to record measurements
- eval, scripts for evaluation of measurements
- exp, examples of working measurement routines (using meas), and evaluation (using eval)
- doc, documentation

1.3 Usage

Copy the files stored from the folder `exp` in your working directory. The example `exp/meas/routine_measurement.py` is the most exhaustive for setup control / recording measurements. While `exp/eval/light_light.py` highlights the evaluation part.

There is not graphical user interface (GUI), as one might know from LabView. Instead, with this library you have full controll over what the software does. But this comes with the price to actually having to read the lines of code – instead of looking at obscure icons. There are only few lines of code to read, in order to understand what's going on; and those are accompanied with comments. So go ahead and read it. You best start with the example scripts in the folder `exp`.

These lines of code you can read with any text editor of your choosing. On Windows I recommend PyScripter (`https://code.google.com/p/pyscripter/`). It brings in some simple GUI features, so you can edit and launch your measurement routines easily, see Fig. 1.1.

1.4 Disclaimer

Some paragraphs in this documentation are copy-paste from the master project report, during whose period the presented scripts have established themselves. A back-up copy

of said report you will find at
`github.com/stefantkeller/VECSELMasterProjectReport`.

2 Measurement Routine

The routine is written in Python. This brings in several advantages: First of all we can actually look through the code and comment on it, where clarification is needed. The Python syntax is simple enough – basically, English with peculiar grammar – so if you can read this documentation, you will understand the program just as well. LabView for example fails at exactly this point; it's very hard to maintain, and the different sections of the script are difficult to interpret (let alone the litteral spaghetti code). And lastly, with Python we don't depend on a third party license. Again, LabView and Matlab fail at this point.

2.1 The routine

You either look at the example given in `exp/eval/routine_measurement.py` and read it through, or you continue to read here. The following long text covers the same.

We choose a set of values corresponding to a) the current of the pump laser, and b) the temperatures of the heat sink. We specify further, a path where to store the results. And lastly, we choose on how often every power current is ought to be measured – repeated measurements in order to obtain the errorbars that inform us on the reliability of the results.

At each temperature, we set the power source to the requested currents, and read out the power meters. The results of each power meter is written in its own file. Each of these files starts with a header line, that contains the state of the relevant settings of the device in question. Consequentially, if we doubt the integrity of the measurements we can look at this header line and at least know what state the device has reported to be in. The information about the power source are also written to its own file, containing the set and actual current. Hence, at each temperature we generate one file for the power source, plus one for each power meter.

At the end of these measurements we write a line in a logfile: The set temperature, the actually reached temperature, the filenames of the files with the results of the different devices, along with a timestamp so we know which measurement took how long. The timestamp also allows us to connect certain effects to the time of the day it was measured. The approach with a logfile and the separate files of each device (whose names are automatically stored in the logfile), facilitates the analysis; all the information a analysis-script needs is specified in the logfile.

At each heat sink temperature we irradiate the sample with different pump powers. Each pump is repeated N times over all. The pump order is selected at random, as illustrated in Fig. 2.1. Thanks to the random sampling the measurement results are detached from the lab environment – most notably time-independent. The heat sink cannot control its temperature with absolute precision. Figure 2.2 illustrates this issue: It shows the actually present heat sink temperature for six set temperatures. In the left column the temperatures are plotted in chronological order. We can identify, the temperature drifts. The right column shows the same temperatures, but corresponding to the set pump. The repeated measurements see a spread of different temperatures but the temporal drifts can not be resolved.

In contrast, Fig. 2.3 shows the heat sink temperature during a measurement without the random pump selection. In this case, clearly, we cannot talk about a single heat sink temperature for all the pump settings for this specific set heat sink temperature. The

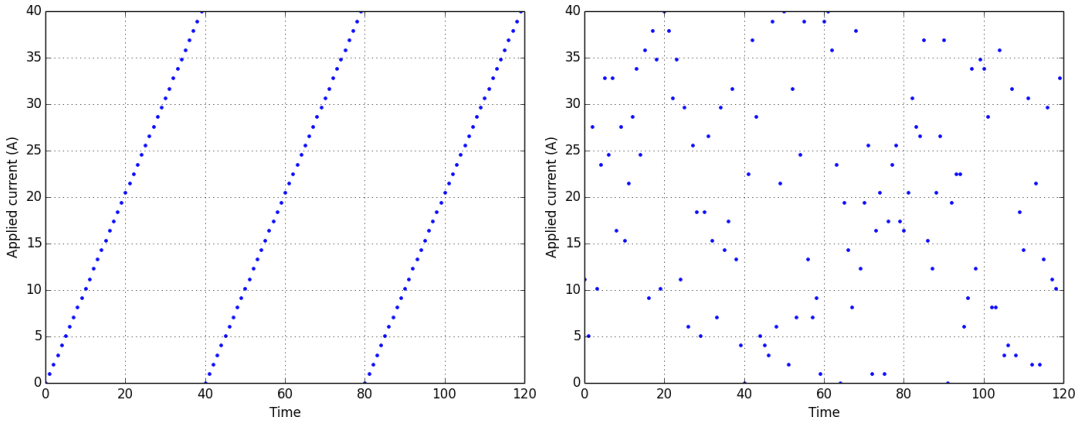

Figure 2.1: Two examples to apply various pump settings, with a repetition rate of 3. Left: A ramp. Right: Random sampling.

resulting measurements are highly repeatable, but only given the same pump order. This pseudo-stability we exploit during the calibration process of the different beam samplers and detectors: During the calibration we don't care about reproducibility but solely about the repeatability of two consecutive measurements.

The power meter average each measurement point over 200 samples, of which each one sample takes approx. 3 ms [1].

The order of heat sink temperature is still a ramp: I expect the setting of a new temperature to be somewhat time consuming. Therefore, we want to make use of the previously set temperature. The measurement routine looks at the specified temperature range, picks the one closest to room temperature, and increases to every second entry. Once the highest temperature is reached, the residual temperatures are picked, in descending order, until the lowest temperature is reached. From there we heat back up to room temperature. This routine is illustrated in Fig. 2.4.

As mentioned already, we repeated the measurement of each pump setting N times. With this repetition we obtain a measure for how well we know the underlying true value. We are hence interested in the mean of these single measurements and the resulting unbiased standard error [2]

$$\Delta x = \sqrt{\frac{1}{N(N-1)} \sum_{i=1}^{N} (x_i - \frac{1}{N} \sum_{i=1}^{N} x_i)^2}. \tag{2.1}$$

For the uncertainties attached to quantities obtained through fits, I use a so-called Jackknife [3] approach: In a nutshell, this method allows to estimate the influence of the single measurement points on the fit parameters without working through the covariance matrix of the fit. The resulting error value is directly related with the unbiased standard error (2.1), used for the rest of the report.

Working with errorbars on data is tedious. Not only do you have to manipulate the data themselves, you always have to carry with you the uncertainty value associated with each of the data points. This is why the evaluation scripts rely heavily on errorvalues; a python library designed for lazy people who don't want to keep track of the uncertainties by hand (`https://github.com/stefantkeller/errorvalues`).

2.2 Safety precautions

The goal of this script is to automate as much as possible. Ideally, we want to install a new VECSEL, align the output coupler, press start, and return some time later to a

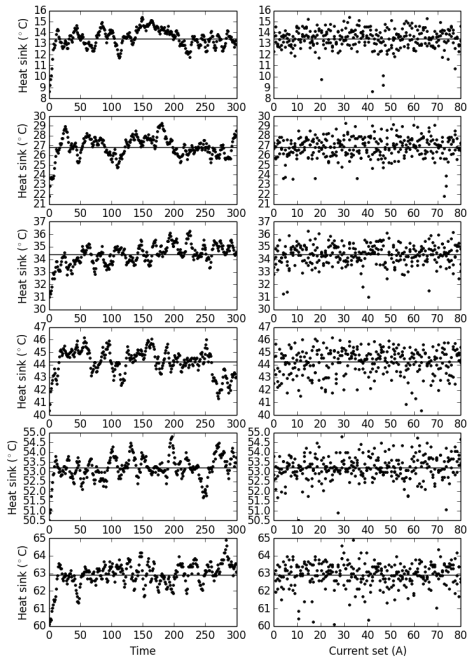

Figure 2.2: The heat sink temperature fluctuates over time (left). Thanks to the random sampling addressed in Fig. 2.1 the measurements don't see these drifts (right).

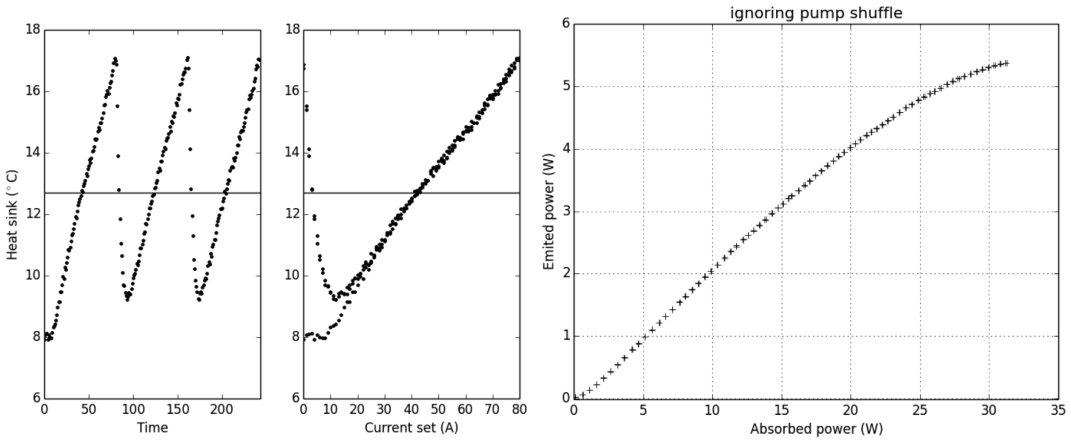

Figure 2.3: In contrast to Fig. 2.2, when we ignore the random pump sampling highlighted in Fig. 2.1, the temperature seen by the single pump settings differ strongly from the average heat sink temperature (left). The resulting LL-characteristic has very little noise on its data points (right). But these small error bars dismiss the fact that the underlying points were measured under very different conditions; eroding the significance of this low noise.

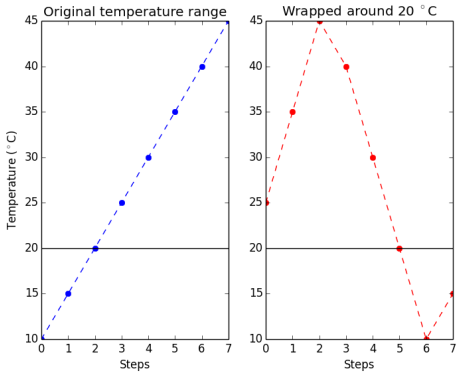

Figure 2.4: An example of how a given temperature range is wrapped around the room temperature (here assumed to be 20 °C).

complete data set for this specific VECSEL under test. For this we need to be sure our measurement routine handles potential errors appropriately.

For this we must not rely on software. Instead, we have to implement the safety precautions on hardware side. In software we can try to mitigate potential problems through proper error handling. I don't know how this would be done in LabView. In Matlab and Python this is performed through so-called `try`/`catch` and `try`/`except` handles, respectively. With it I have implemented that if anything goes wrong software side, the power source is shut down (This order itself is so low-level, that it *should* always work.). For example, one of the devices could send an unexpected answer, the heat sink doesn't reach its requested temperature within reasonable time, etc. For the unlikely event that even the power source shut down doesn't work, we have to implement the safety precautions on hardware side.

Our power source has two ways to be shut down: We can disable the current, or we can set the current to zero. In order to disable the current, we first have to ask the power source, whether the current is currently applied. If it is, we toggle it like a light-switch to shut off. However, querying the current state is error-prone. Hence, if an error is caught we leave the shutter to be in what ever state it is and only set the current to zero. Potentially it is still applied. But the light output at 0 A is barely detectable – but there still *is* a leak flow of photons. Writing to the power source should not cause any new problems, so simply writing 0 A should go through.

On the hardware side, the "laser on" light in front of the lab is controlled by a logic tied to the power source: As soon as the power source applies a certain specified voltage the laser warning turns on. During the measurements this safety sign therefore is always on. Our setup is not fit for an inter-lock solution, hence the safety responsibility lies solely on you, the operator. This means, you have to wear you safety goggles, and the setup is to be closed up, Fig. 2.5.

As a second safety precaution `exp/meas/VECSELSetupFake.py` allows us to test modifications on the measurement routine with fake devices. In order to connect the measurement routine with the measurement devices we have to specify a protocol. In the new script we do this by selecting an external file that contains the initiation details. By specifying the fake protocol we can modify the routine without the real devices. This separation in code is, for one, good practice, and secondly convenient if we cannot use one of the devices due to a revision or whatever. To be detached from the physical devices was not possible with the old script – which is one of the reasons why it was possible to toggle the power source shutter while setting the wavelength of the power meter.

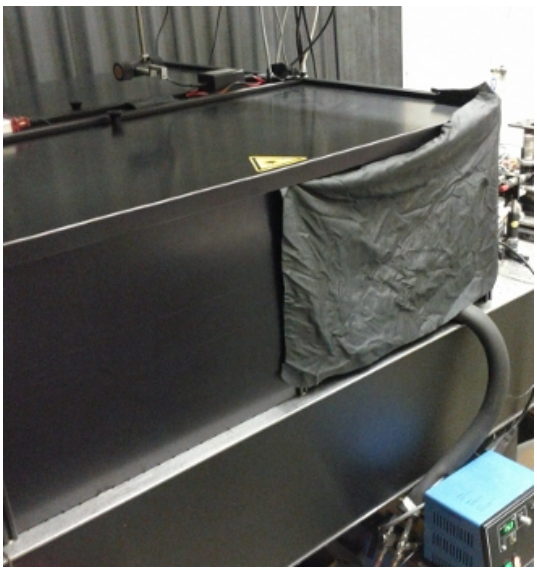

Figure 2.5: During operation the setup has to protect the Along the primary beam line we use a beam dump. None the less, there is a lot of scattered light and reflections from everywhere. To cope with that, close the walls and cover. The curtain allows you to reach into the setup, in order to align the xyz-stage during high power operation.

3 Calibration

3.1 Principle

Figure 3.1 shows a photograph of the setup. Before we can use it to characterize samples, we have to characterize the setup first. Following the beam path, we start with the lens system indicated as *pump*. This input beam is targeted at the sample. Part of this pump beam is sampled by a beam sampler (BS), a glas plate with high damage threshold and anti reflective coating on one side. This sampled light is directed towards a detector. The pump light is reflected off the sample, collimated by a lens, and again sampled as for the input. The lasing output passes the output coupler (OC), a collimating lens, and another beam sampler (coated for the emission wavelength). This beam sampler we eventually use to record the spectrum of the emission.

In order to calibrate this setup we need to know how to correlate the readings of the single detectors with the actually present powers. For this we place the thermal power meter at the four indicated positions. With the thermal power meter we can validate the setup up to 40 W, in principle.

Position 1 requires to remove the sample. We're interested in the correlation between the readings of the detector after the beam sampler and the measurements at the sample position. From this same measurement we can also extract a look-up-table what current setting of the pump laser corresponds to what output power.

On position 2 we measure the beam power present after the collimating lens. This we relate with the readings of the detector after the beam sampler, gathered during the measurements at position 3 (below). Make sure (IR-viewer) you capture all of the reflected light within the aperture of the power meter. Clipping during the calibration renders everything else useless.

At position 3 we need two measurements: once with and once without the beam sampler. The difference between these two measurements lets us correct for the fraction of light the beam sampler directs away.

The pump order during calibration is a ramp (not random sampling as for the measurement routine, section 2). As shown in Fig. 2.3 this way we arrive in a highly repeat-

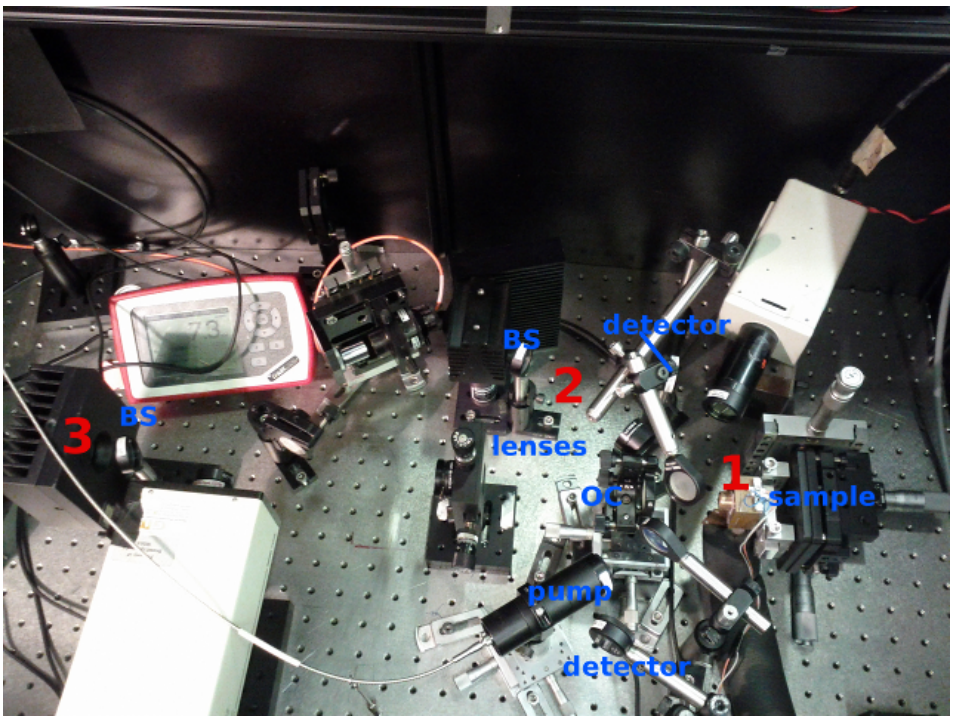

Figure 3.1: In order to calibrate the setup we have to measure the actually present power at the indicated positions 1–3.

able situation, using the same exact ramp.

3.2 Implementation

The routine in `exp/meas/routine_calibration.py` needs the following 4 files. The calibration evaluation routine in `exp/eval/calibration.py` should also clear things up, what the following comments mean.

Between these measurements you move only the thermal power meter to the different positions indicated in Fig. 3.1. Do not touch the other equipment! With this procedure we gauge the photo detectors (after beam splitters to protect from high power) to the thermal power meter. When placing the thermal power meter, you have to make sure the exposure is below its damage threshold. A Thorlabs thermal power meter S314C, for example, has a power range limited up to 40W.

`1_pump_calib.csv` :
Place the thermal power meter at the sample position; position 1. Choose the variable `pump_end` equal to the pump current corresponding to the maximally allowed pump power (or less, to be on the safe side). (Note: `pump_end` will be a lower value than during actual operation. This is made possible because there is a linear relation between pump current and the power seen at position 1. We're going to extrapolate based on this linear relationship. Ensure yourself, this is actually linear! (Otherwise you have to find a new solution how to do the job.))

`2_refl_calib.csv` :
Thermal power meter goes in the reflection channel, between collimating lens and beam sampler; position 2. Variable `pump_end` now determines up to what power you can calibrate the setup over all. Make sure the power exposure at this point doesn't exceed the power range of the power meter. Caveat: If you replace the sample under test

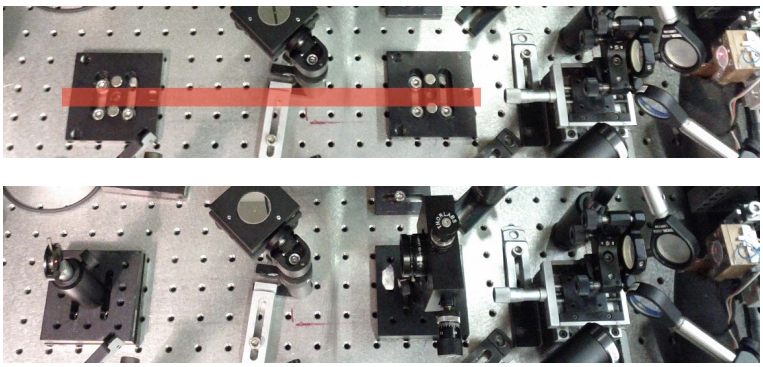

Figure 4.1: Top: Removeable posts to define the beamline. Bottom: HeNe beam has to be directed through the two irises. With these two irises the position of the HeNe source is detached from the beam line.

without changing the alignment, you can – in principle – use an old calibration. But, if this sample has a higher reflectivity, you run the risk of entering an uncalibrated range.

`3_emission_calib.csv` :
Thermal power meter goes at its final position in the laser emission path – position 3 – without the spectrometer beam sampler.

`4_emission_calib.csv` :
Thermal power meter goes at its final position in the laser emission path – position 3 – with the spectrometer beam sampler in place. Measurement 3 and 4 allow you to find the influence of said beam sampler.

In practice you will probably gather these four measurement in reverse order. After you have conducted the measurements (e.g. with `routine_measurement.py`), you run this script for 4, remove the spectrometer beam sampler and measure 3, and so on. This way you're sure to account for the actual alignment present during the measurements.

3.3 Evaluation

The example routine `exp/eval/calibration.py` works with these mentioned files. It is easiest to look at the code and learn based on the comments provided alongside. The `main()` function in these scripts you can use to calibrate one of the specific parts manually.

4 Measurement alignment process, description of

In order to align the different parts we employ a HeNe laser. Figure 4.1 shows the beam line orthogonal to the sample surface. The beam line is defined by two removable magnetic posts. These posts allow us to remove and replace a mount reproducible. The beam line is ultimately defined by two irises. As such, we can place the HeNe source anywhere on the table; we simply have to guide its beam through these two irises.

For the alignment, in a first stage, we align the sample orthogonal to said beam line: We remove – or flip, Fig. 4.2 – all other components along the beam line and leave only the sample. We adjust the orientation of the sample such that the back-reflection is directed at the HeNe, Fig. 4.3. Once the sample is aligned we place the output coupler back in the beam path and repeat the same procedure.

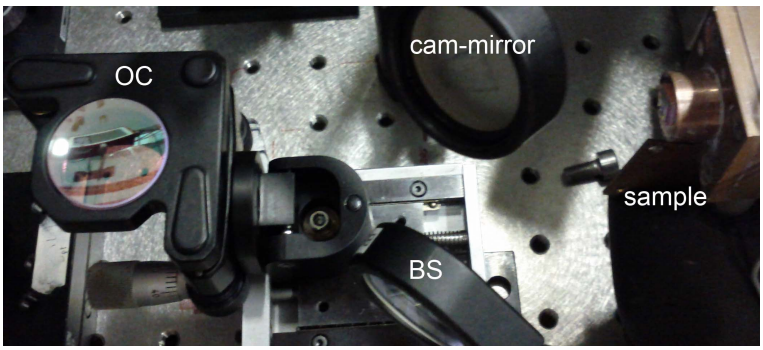

Figure 4.2: First, we remove / flip all components along the beam line except the sample. In this configuration we align the sample surface orthogonal to the HeNe.

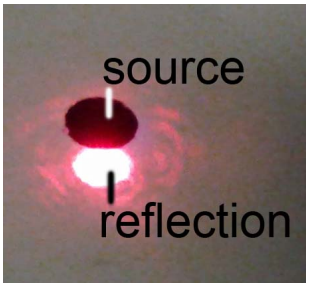

Figure 4.3: By arranging the reflected beam to coincide with the HeNe source position we ensure the orthogonality of the component.

In a last step we have to irradiate the sample with the 980 nm pump. A camera along the emission beam line sees the photoluminescence resulting from the pump on the sample. This camera is equipped with a long-pass filter in order to see only the photoluminescence and not the pump light. With this camera, initially, we see two spots corresponding to pump and reflection from the output coupler. Because of the aforementioned alignment with the HeNe laser these two spots should already be close, see Fig. 4.4. We bring these two spots to overlap, using the xyz-stage of the output coupler. Once the two spots do overlap, and the pump is above threshold, the sample starts lasing.

Figure 4.5 shows an overview of the different components. The pump light is directed

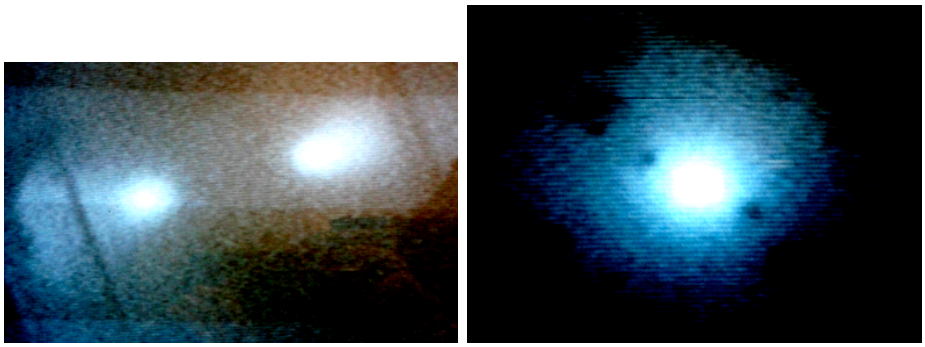

Figure 4.4: After the pre-alignment with the HeNe, the two spots – originating from the pump and the output coupler reflection – are close by (left). By optimizing the alignment of the output coupler, these two spots have to be brought to overlap (right). Given this configuration, we have to increase the pump power, and at threshold we obtain laser emission. For the fine-alignment we have to replace the camera with a power meter and adjust for maximum output power.

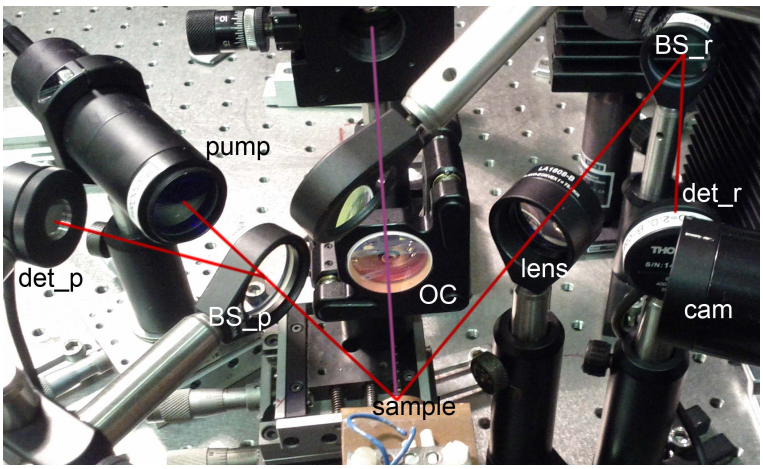

Figure 4.5: Overview of the different components incorporated for the cavity.

via fiber to a lens system that images the light onto the sample. With beam sampler BS_p we extract a fraction of the pump and direct it to detector det_p. This way we have a realtime reading of the pumped power. A considerable part of the pump light is reflected off the sample. This light diverges. First, we thus have to collimate it. For this we install a lens with appropriate focal length and distance from the sample. Of this reflected beam we again sample a fraction with BS_r, and direct this to detector det_r. The largest fraction of the reflected beam is directed to the beam dump. By sampling pump and reflection in this geometry we avoid high power readings on the detectors.

5 Sample surface microscopy

Take a picture of the non-radiative defects of the sample under the microscope. For this connect the pump light with the microscope illumination, and place the camera, including a long-pass filter so we see only the photoluminescence and block the pump, as illustrated in Fig. 5.1.

Note: Whenever you change the fiber connected to the pump diode, ensure the fiber is intact – to avoid damages portrayed in Fig. 5.2 – and make sure to have the best possible diode-to-fiber coupling. For the first task use the fiber inspection microscope. For the latter (see diode manual!) apply a certain pump, unscrew the fiber connection at the pump diode module, and turn the fiber end while observing the change in output power. Lock it for the maximum configuration.

With a microscope magnification of 5×, take a picture of all four corners. For this you can use the fully equiped NI-MAX, which came along with the other NI drivers, see Fig. 5.3. The overlapping parts of these images allow you to staple together the whole photograph.

References

[1] Thorlabs, *Power meter operation manual*, http://www.thorlabs.de/thorcat/19500/PM100USB-Manual.pdf

[2] R. J. Barlow, *Statistics – A guide to the use of statistical methods in the physical sciences*, The Manchester Physics Series, Wiley, (1999)

[3] B. Efron, G. Gong, *A Leisurely Look at the Bootstrap, the Jackknife and Cross-Validation*, The American Statistician, Vol 37, No 1 (Feb 1983), pp. 36-48

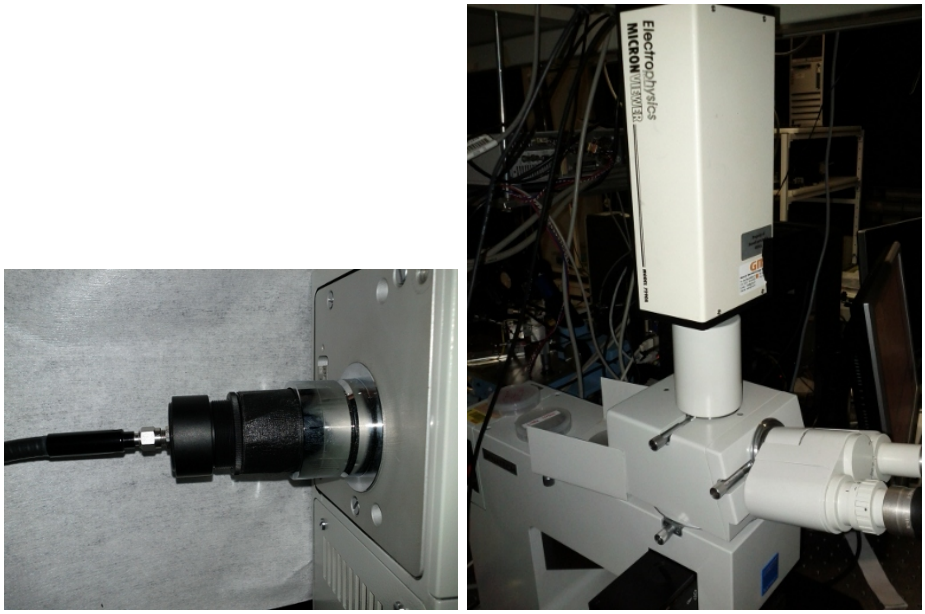

Figure 5.1: Direct the pump light to the microscope for illumination. With this light source, and a long-pass filter on the camera, the non-radiative defects are directly visible.

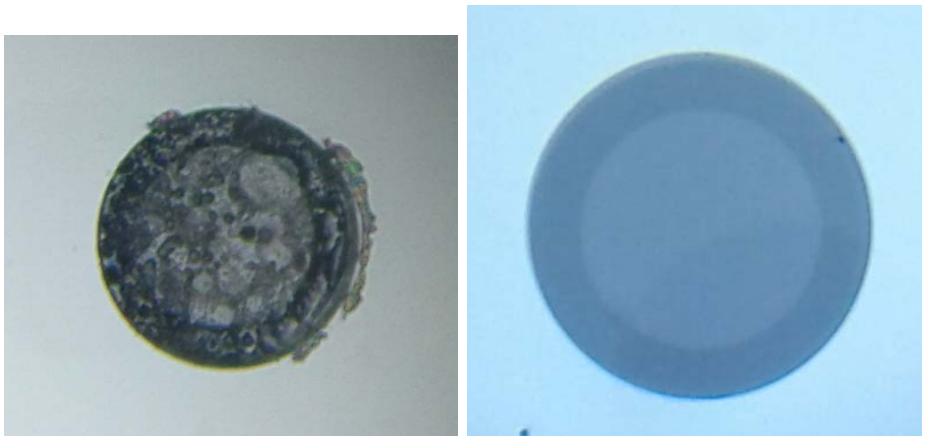

Figure 5.2: Left: This fiber end is burnt; it delivers a beam of bad quality. Rigth: This is how all fiber interfaces are supposed like; it is intact.

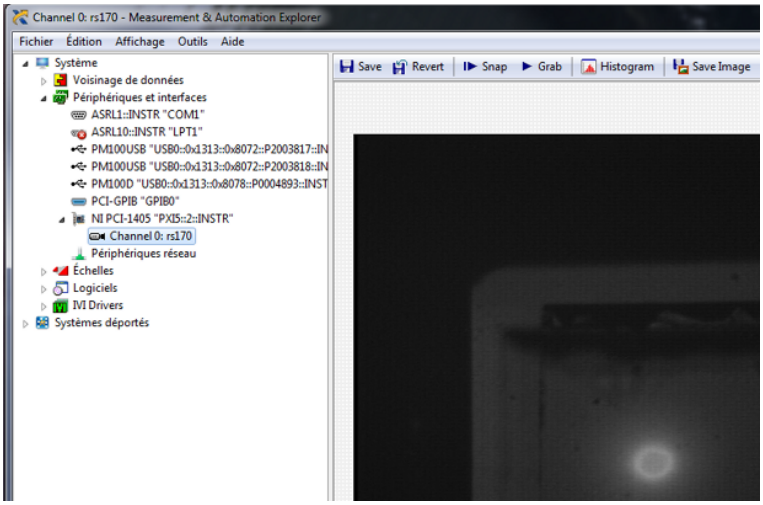

Figure 5.3: National Instrument's measurement and automation explorer (NI MAX) allows us to control the camera.