Freie Universität Berlin
Fachbereich für Mathematik und Informatik
Institut für Informatik

Masterarbeit

Large Scale Supercomputer Assisted and Live Video Encoding with Image Statistics

18. August 2016

Bearbeitet von:
Hauke Jürgen Mönck
Brentanostr. 63
12163 Berlin
hauke_moenck@gmx.de

Gutachter : Prof. Dr. Raúl Rojas
Betreuer : Prof. Dr. Tim Landgraf

Abstract

The primary purpose of this thesis is to elaborate mechanisms for encoding images as videos for live recordings and post-hoc encoding of large amounts of images on a supercomputer. A major focus was set on quality preservation and optimizing the recording setup.

Researching social insects behaviour has always been a human interest for various reasons. To examine colonies having many individuals and complex social interactions, large amounts of sample data are necessary. In the Beesbook project this sample data comes in form of images of marked honeybees. These images shall be evaluated automatically using custom software, which requires excellent image quality and resolution. This master thesis introduces various improvements to the acquisition process to obtain images of optimal quality at minimal cost. Also large amounts of existing data are required to be compressed, aided by a supercomputer. For the live recordings a custom IR lighting system was introduced, recording software using a GPU encoder was developed and image statistics for calibration and surveillance have been introduced. Also software was created to automatically do large scale compression of videos.

Evaluation has shown that the newly introduced lighting system not only comes at 23.7% of the price of a conventional lighting system, but also has good illumination properties. Different mechanisms are provided to analyse images and configure recording setup for optimal quality. Existing data could be compressed using the HEVC video codec from 289 TB to 80.6 TB, saving 72.1% of space with negligible loss in image quality. Finally recording software was developed to achieve this level of compression during live recording.

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Diese Arbeit wurde keiner anderen Prüfungsbehörde in gleicher oder ähnlicher Form
vorgelegt.

Berlin, den 18. August 2016

.....
Hauke Jürgen Mönck

Acknowledgements

Besides the author there are several people involved in the successful elaboration of a thesis. Hereby I want to thank some of them.

I would first like to thank my thesis advisor Prof. Dr. Tim Landgraf of the Biorobotics Lab at Freie Universität Berlin. The door to Prof. Landgrafs office was always open whenever I ran into a trouble spot or had a question about my research and writing. He always steered me in the right the direction whenever I needed it.
Also I am grateful to Mr. Fernando Wario Vazques. He has made available his support in a number of ways. I would like to express my gratitude to all the other members of the Biorobotics Lab as well, for any informative conversation and the good working atmosphere they provided.
Furthermore I would like to thank Mr. Lutz Freitag, who always made available his support and shared his knowledge about electronics.

Finally, I must express my very profound gratitude to my parents Jürgen and Petra Mönck for providing me with support and encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Hauke Jürgen Mönck

Contents

1. Introduction **7**

1.1. Biorobotics Lab . . . 7
1.2. Overview . . . 8
1.3. Experiments . . . 9
1.3.1. Legacy Honeybee Experiment System . . . 9
1.3.2. Bumblebee Training . . . 10
1.3.3. Bumblebee Experiment Setup . . . 11
1.4. Improving the Recording . . . 12
1.5. Considered Compression Schemes . . . 14

2. State of the Art **15**

2.1. Tracking of insects . . . 15
2.2. Live Video Encoding . . . 16
2.2.1. Infrared Camera Flash . . . 16
2.3. Post-Hoc Encoding . . . 17

3. Supercomputer Assisted Large Scale Video Encoding **18**

3.1. Hardware and Software Facilities . . . 18
3.2. Code and Adjustments . . . 19
3.3. Choice of Codec . . . 21
3.4. Container Format . . . 21
3.5. Compression Quality . . . 22
3.5.1. Choosing the Quality . . . 22
3.5.2. Choosing the Size and Encoding Time Trade-off . . . 25
3.6. Verification . . . 26
3.6.1. Timelines . . . 26
3.6.2. Frame Counting . . . 27

4. GPU Assisted Video Recording **28**

4.1. Software . . . 29
4.1.1. Acquisition Application . . . 30
4.1.2. Watcher Application . . . 32
4.1.3. Storage Management Application . . . 37
4.2. Hardware . . . 37
4.2.1. Lighting . . . 37

4.2.2. Recording Hardware 41

5. Evaluation **44**

5.1. Supercomputer Assisted Video Encoding 44

5.1.1. Filesize and NPL cost 44

5.1.2. Verification 44

5.2. Live encoding 45

5.2.1. Lighting 45

5.2.2. Image Statistics 47

5.2.3. Stability 47

6. Discussion **50**

6.1. Supercomputer Assisted Video Encoding 50

6.2. Live Video Encoding 50

7. Concluding Remarks **52**

7.1. Supercomputer Assisted Video Encoding 52

7.2. Live Video Encoding 52

Appendices **54**

A. Calculations and Tables **55**

A.1. HEVC And AVC Compression Ratio For Beesbook Recordings 55

A.2. Lighting System Cost Calculation 55

A.3. Disk Space Cost Calculation 55

B. Glossary **57**

1. Introduction

Collective intelligence is a form of intelligence that emerges from individual agents collaborating. The agents sample local information, process these information and communicate them to their peers. This enables the agents to do consensus decision making, delegation and organization of tasks, etc.. Observing the set of clients as a whole, a collective intelligence emerges. Many social insects use collective intelligence in practice. In the Biorobotics Lab of the Free University Berlin honeybees (apis mellifera) are the major model organism. Other animals, such as bumblebees (bombus) and guppies (poecilia reticulata) are also surveyed.

By observing these organisms behavioural models may be derived. These models not only describe individual behaviour, but also how swarm behaviour arises from the individuals. In this sense, the individual organism acts as a client, the swarm intelligence being a form of collective intelligence. Bees in general have a sophisticated social structure and behavioural patterns of which major parts remain unresearched. Seeley described in *Wisdom of the Hive*(37) a multitude of experiments for researching honeybees behaviour, worker bees egg laying properties for instance. The experiments required an extensive amount of manual preparation and evaluation. It is assumed that highly complex observations are required to fully capture the swarms intelligence in all its parts.(44) Hence more complex and extensive observations are expected to be required. Looking back at the experiments described by Seeley, more automated observations are required to gain deeper insights.

1.1. Biorobotics Lab

The Biorobotics Lab developed an elaborate automatized observation scheme to engage on the previously illustrated issue. In the experiment a beehive consisting of one comb accommodating from 1000 to 2000 bees at a time is being recorded continuously for several weeks. There are four high-resolution cameras set up for the recording. The recording is being done for several weeks during summer, when the bees are most active and produce honey, depending on temperature.(18) Each bee is marked using a unique tag on the back of its thorax as depicted in figure 1.1. In recent experiments applying the tags is done once a bee hatches from its larva state. For each tag the day it was attached to a bee is noted. In consequence the date of hatching for every individual bee is known. In previous experiments all existing bees have been marked at once in the beginning. In post-hoc analysis the positions of the individual tags are found for every frame in the

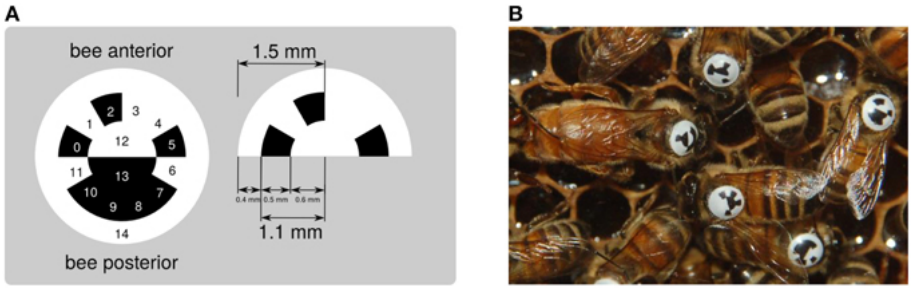

Figure 1.1.: Tags for bee and bumblebee identification. *(A) The circular matrix design exhibits 15 regions used to represent a unique binary code (cells 0 to 11), to encode the tags orientation (cells 12 and 13) and produce a clear contrast with the bee thorax (ring 14). (B) Tagged bees inside the hive.* (44)

high-resolution recordings. Additionally low resolution cameras observe the hive at 100 Hz, enabling detection of waggle dances. Having the tag positions, motion vectors may be calculated. Sets of motion vectors can be combined to find any position of the honeybees of their entire lifecycle in the hive. Merged with the knowledge about the waggle dances, eventually individual behaviour and interactions may be deduced. Ultimately swarm behaviour can be inferred.

Besides the honeybee experiment there is a bumblebee experiment with a slightly different nature. It is a continued experiment from Jin et al.(23) Here the bumblebees have their wings cut and are assigned slightly bigger tags than depicted in figure 1.1 to fit their thorax. The bumblebees tags are 4 mm in diameter. They are living in a small colony of around 50 individuals in a terrarium for the hive. A second terrarium, called arena, may be accessed to collect food in several training sessions. The purpose of the experiment is to acquire information about the bumblebees strategies of finding food sources.

To enable the algorithms to generate reliable information from the recordings, image quality is an important factor. This means properties such as image sharpness, contrast, lighting, resolutions etc. are of major importance. As a high image quality is required to reliably detect tags, large amounts of data being generated is the logical consequence. Optimizing these quality measures while decreasing the amount of data to be stored is an aim of this thesis.

1.2. Overview

The Biorobotics Lab used the legacy system for honeybee recordings in 2014 and 2015. This thesis transfers the legacy system into an advanced system in two major work packages as depicted in figure 1.2. In work package 1 a new data format is found to most efficiently encode existing image data. The amount of existing image data is large and was hence encoded using a supercomputer into the new format. This work package was

	Legacy	Advanced	Work Package
Existing Data	JPEG image compression, ca. 160 TB per season	HEVC video compression, ca. 40 TB per season	WP 1
Hardware	Constant lights, lower contrast	asynchronous flashing lights, high contrast	WP 2.1
Recording Software	JPEG image compression on CPU, high bandwidth and large files	HEVC video compression on GPU, low bandwidth and smaller files	WP 2.2

Figure 1.2.: Overview of the thesis structure. The experiment setups of the Biorobotics Lab are composed of recording hardware, software and resulting data. In the course of the data this previous system, henceforth called legacy system, will be transferred into a new system, called advanced system, in two work packages. Work package two is divided in two subpackages, for hard- and software respectively.

handled in chapter 3. Work package 2 revolves around improving the legacy hardware and recording software. Improvements in the hardware may yield more contrast rich and eventually better recorded images. Improvements to the software are necessary not only to comply the changes in hardware but also to generate the video data. The video data shall be stored in the new format according to elaborations done in work package 1 live.

1.3. Experiments

1.3.1. Legacy Honeybee Experiment System

There are four high-resolution cameras directed at the beehive, two at each side as depicted in figure 1.3. These cameras are connected to a single computer which manages the acquisition process. The images are compressed in JPEG format and bundled to TAR archives of 256 files. As the total amount data is to vast for being stored locally, it is sent to the Cray supercomputer. The Cray is a supercomputer by the HLRN, who are sponsoring the Biorobotics Lab with computational time and storage. A local NAS can buffer data for a short period, in case the Cray encounters maintenance or other downtimes. It will also store striping data. This is, four times a day a short recording sequence is permanently being stored on NAS. It is done to even out permanent or temporary unavailability of the Cray in the future. Recordings have been done from 24.07.2014 to 18.09.2014 and from 19.08.2015 to 26.10.2015, so far yielding 132TB and 157TB[1] of data respectively.

The hive is protected by a glass pane. The FLEA 3 *FL3-U3-88S2C-C* camera is being used in combination with a regular lens.[2] Additionally there is one modified PS-Eye camera on each side, having an IR filter installed. Constant infra-red lighting using a set of 24 lamps of type TV6700 was used to illuminate the hive.(7) These lamps have a

[1] Space effectively used on file systems. Season 2014 on home, GPFS and 2015 on gfs1, Lustre file system

[2] Available online at: `http://www.rmaelectronics.com/ricoh-fl-cc1214a-2m-2-3-12mm-f1-4-manual-iris-c-mount-lens-2-megapixel-rated/`, retrieved 13.06.2016

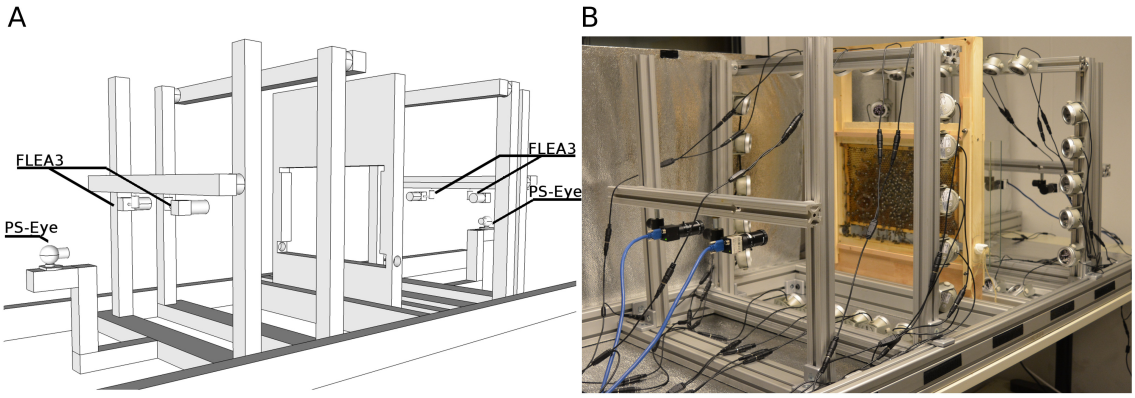

Figure 1.3.: The legacy honeybee experiment setup. A shows a schematic of the setup. There are three cameras at each side of the observation hive, two high resolution FLEA3 cameras and one modified PS-Eye camera. The PS-Eye cameras have their white light filter replaced by an IR filter and their lens replaced. The constant lights are positioned in a circular fashion around the cameras. In B a photo of the setup 2015 can be seen, not including the PS-Eye cameras. Images by Fernando Wario Vazques.

light sensor, which dims the IR LEDs accordingly. These sensors have been deactivated to guarantee the constant lighting to the beehive. The lamps emit light of 840 nm centroid wavelength which is invisible to bees and bumblebees. The range of visible light of either species goes up to round about 600 nm.(26) (38) This corresponds to the colour orange in human perception.

The bees may leave the hive to the exterior of the building through a small tunnel. The hive itself is inside an observation room which is kept at around 30°C at all time. This will guarantee bee activity throughout the recording time to a certain degree.(18)
In this thesis several improvements to this setup will be introduced.

1.3.2. Bumblebee Training

So far it is fully understood whether and to what extend bumblebees use landmarks and their visual capabilities to locate themselves and potential food sources. The purpose of the training is to train bumblebees to find the food according to landmarks. To achieve unambiguous results other means of orientation have to rendered inefficient. For example clearing the floor to eliminate odour tracks creating an indistinctible environment apart from explicitly presented landmarks.

The Bumblebees have been trained four times a week in cooperation with Fernando Wario Vazques and Franziska Lojewski. The training was done Monday to Thursday, with a day of testing on Friday.

There is one training session a day, consisting of four iterations. Each iteration lasts one hour. In the beginning of each training iteration, the food is positioned relative to

the same landmarks. In the subsequent training iteration, the new position of the food is

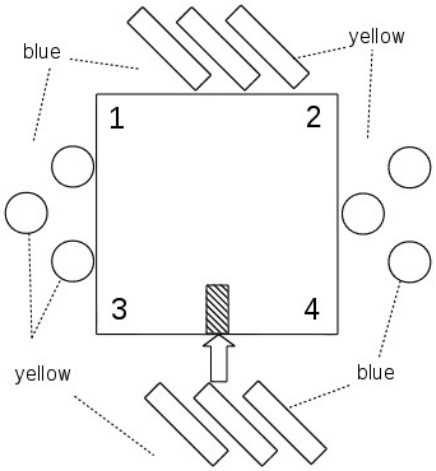

Figure 1.4.: Sketch of the bumblebee experiment panorama layout. The entrance to the arena is marked with an arrow. Within the arena, numbers 1 to 4 indicate possible food locations. The circles and stripes represent an example panorama design.

chosen randomly from the four positions which can be seen in figure 1.4. The panorama has to be turned, so that the food again is in the correct relative position. Bumblebees have to be placed into the hive again manually if there are still some in the arena.

When all iterations are done, the tunnel to the arena is being closed and the white light is switched off. The recording software on the PC corresponding to the arena is stopped while there is no ongoing experiment.

Testing is done the same way as training, but no food is supplied. The Panorama is just rotated and the bumblebees are enabled to enter the arena. This will test whether the bumblebees apply the learned behaviour without having other indicators for the food.

1.3.3. Bumblebee Experiment Setup

The bumblebee experiment was conducted in 2016. It was the first to use the advanced system and served as a testbed. The arena box, the hive box and the frame have been acquired from a regular building supplies store. The framing is closed from all sides using chipboard. The front is not nailed but only attached using magnets as to not disturb the bumblebees unnecessarily.

Everything in the arena is designed to be as symmetric as possible. In figure 4.2.1 one may notice that there are two extra holes where are no actual cameras. They are just there to keep up the symmetry. This is done so the panorama is the only possible landmark for orientation.

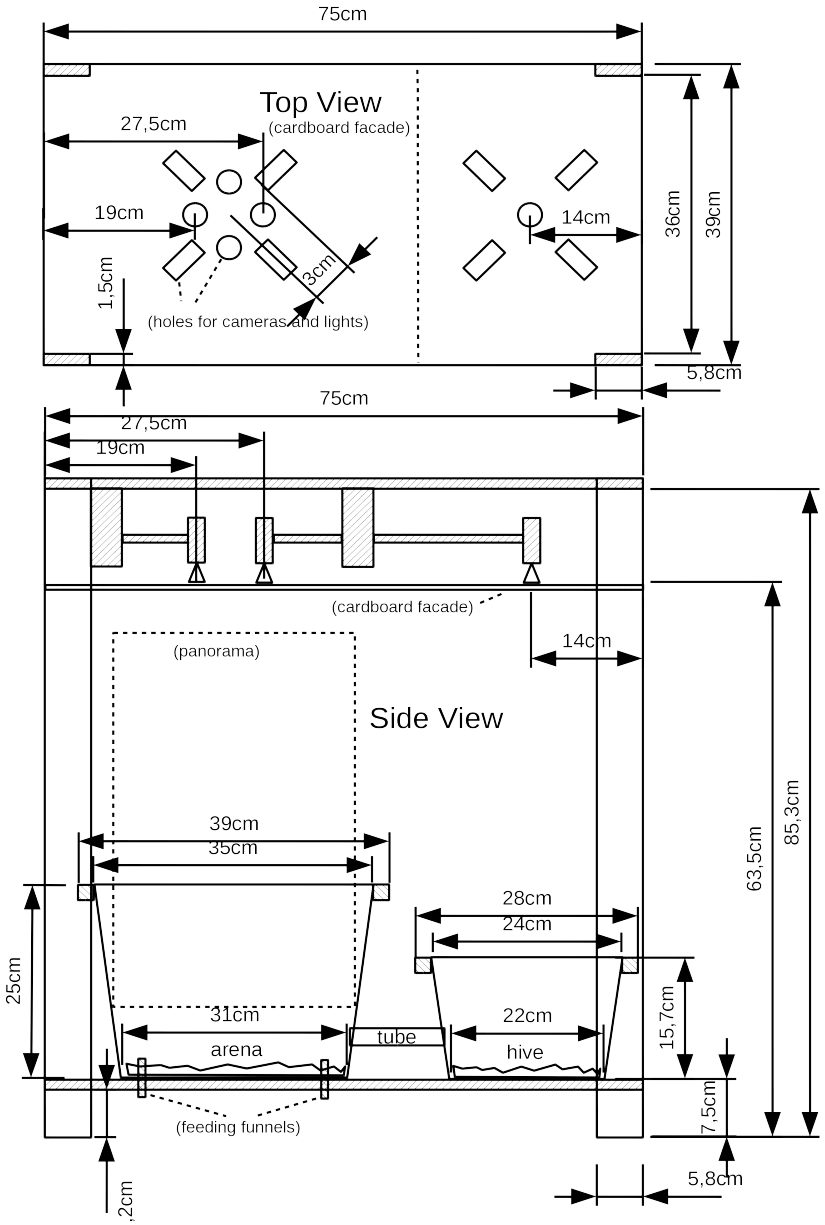

Figure 1.5.: Sketch of the bumblebee experiment setup. In the upper portion the top view can be seen. Note that the holes for the cameras and lights are symmetric on both, horizontal and vertical axis. In the lower portion, the side view of the setup can be seen. The exact positioning of the boxes is not indicated. In fact, the boxes may be moved around freely. To acquire a reproducible positioning of the boxes during the experiment, markers on the floor have been used. The dashed line indicates where a vertical cardboard sheet may be added to shut off the hive from the arena in terms of lighting.

1.4. Improving the Recording

As mentioned previously, improving the image quality while decreasing the amount of data to store is the purpose of this thesis. This comes in two aspects. First, the recoding

system needs to be improved. There are different schemes to be considered to compress the image data. The live encoding schemes will be elaborated in section 1.5. To improve the quality as per previously mentioned methods of the camera recordings, several steps have been taken.

- The recording system was extended to contain a configuration mode. In this configuration mode, multiple quality measures of the recorded images are taken live. These measures may be optimized by manually adjusting the camera lens, lighting, etc..
- A unique solution to lighten the beehive was developed. The previous setup used constant lighting. This cause low contrast images, because the light emitted on one side of the hive will light up the comb itself, as the combs consist of light-transmissive wax. This results in bright combs on either side. Finally it leads to *bright* bees on *bright* background, or in other words, poor contrast. This is why flashing the individual sides asynchronously will improve contrast to the recordings. As the framerate is less or equal 4 FPS at all times, this is feasible.
- The improved encoding scheme leads to smaller files at equivalent quality. This comes with multiple advantages. The recording quality may be improved using the capacities which were acquired by improving the encoding scheme. Longer potential downtimes of the Cray may be buffered and longer stripe sequences may be saved. This will improve the overall reliability of the recording system. In the future it might enable local storage and postprocessing of the recorded data.

These issues are being addressed in chapter 4.

Improving future recordings is essential as there are vast amounts of recorded data in existence. Since the data accounts to as much as 289 TB, transferring it to local machines for procession and re-transferring it to the Cray seems an inept task. Inept, because the bandwidth to and from Cray as well as the local resources are confined. However the computational resources of the Cray are immense, as will be described in section 3.1. But re-encoding the data on Cray is subtle task due to the amounts of data and parallelizing the task. The existing data is encoded in JPEG format and hence comes with compression artefacts. Which parameters must be applied to guarantee acceptable quality for the recognition tasks? How to mux the videos to provide optimal accessibility for various tasks?
The task also includes building unique software for the Cray architecture for encoding and attaining a strategy to deploy the input data to a multitude of processes concurrently.

1.5. Considered Compression Schemes

As pointed out previously, the amounts of raw image data is very large and storing them compressed in JPEG format is not sufficient. Different approaches to compress the videos have been considered.

The first and surely best compressing approach was as following. The images of bees are being stored as low resolution and low quality videos. Before discarding the original images the localizer is being run on them. The localizer is a software which finds all tags in the images with an accuracy of 90.6%.(44) Note that more recent developments may enable the localizer to reliably detect more than 99% using CNNs. The localized tags are then copied next to each other into a new image, preserving the very most of their original quality. But the problems about this approach are manifold. The major issue is most of the original data getting lost. This excludes the possibility to later improve the localizer and retrospectively increase the number of detected tags. As mentioned in section 1, acquiring enough usable high quality data is crucial. Hence this method of compression is quite dangerous, as it is yet unknown what quality and meta data are actually required. Furthermore other tasks apart from tracking the bees may hardly be done on data compressed this way. One example for this is the classification of objects in the beehive as described by Ziese.(48) There the images were used to classify portions of the beehive using neural networks. As shown by Ziese, more fine grained classification (e.g. classify a bees head and thorax) require a higher level of image details than basic classification (e.g. classify bees and combs). Secondly compression using better image compression codecs were considered. The recently developed BPG codec(4) was the most promising candidate, according to studies as done by Hofbauer et al.(19) However, the compression rate was still insufficient. Finally compression using videos was considered. The datasets compress well as videos, as the images are of sequential nature and contain temporal redundancy. Encoding may be feasible using different encoders. Video encoders will be elaborated in section 2.2.

2. State of the Art

2.1. Tracking of insects

Animal tracking and behavioural analysis is a well studied field in biological science. Tracking larger animals can be done using GPS trackers as done by Patrick et al.(6), for instance. When tracking insects, such as honeybees and bumblebees, the issue of the individuals being very small arises. Seeley observed a beehive of 2000 to 4000 bees by attaching circular markers on the thorax.(37) The tags had a distinct colour every 500 tags and an Arabic number from 1 to 500 printed on them. Additionally they have been marked using a colour dot on their abdomen to distinguish which feeder they visited. This enabled him to do detailed observations of individual behaviour. However, this method does not scale well, as the tags are not well suited for machine readability and hence automatized recognition. Seeley read all the tags manually and noted his observations by pen or voice recording. This method is not suitable for large scale applications due to its labour extensive nature.

A more automatized approach has been done by Mersch et al.(27) The recording was done using a FCi4-14000 monochrome camera at 4560×3048 pixels. They have been tracking an ant colony by attaching square tags, similar to QR codes.(13) In the given setup, this resulted in a size of 25–30 pixels per marker, each marker having a side length of 1.6mm. This yields a resolution of 17–18pixels/mm. Every marker encodes 36 bits, 26 of them being error correction.

Tracking was done live using the ARTag library. While high resolution recordings were discarded after processing, low resolution samples were kept to visualize and verify tracking results. The results of the tracking seem appealing: 225 immobile tags could be recognized with 99% accuracy. Yet only 88±17% of attached tags. In worst case, these results are just slightly better as the results achieved by Wario et al with an accuracy of 65.98%.(44) Using the recording system of Mersch et al has been considered in the first place. The cameras owned by Wario et al have just a slightly smaller resolution, but have a regular colour CMOS sensor. The CMOS image sensors use a Bayer pattern. The tags used by Mersch et al have been tried on the bees. However, the bees will start removing the tags from each others back. The tags have to be square, as this is a proprietary solution. Downscaling and applying them to a round, bent physical tag structure is not feasible either. They either are to small or have a too high curvature to be ever readable as a whole.

2.2. Live Video Encoding

For many years, begin ning as of 2003, the AVC codec dominated the market in terms of compression to quality ratio.(35)(46)(33) Recently codecs have been developed which compete with AVC about this position. Ahead of all, there are HEVC and its free counterpart VP9. Studies have shown that these codecs come in fact with a better compression to quality ratio, HEVC performing best.(36) HEVC has seen uses in science and economy to compress video data more efficiently. For example Xiao et al developed a background subtraction method for HEVC for Chinese public surveillance camera systems.(47) This approach is aimed to be less computation extensive but not having a significantly better compression to quality ratio. The video streaming platform Netflix now offers streaming their media in HEVC, yet it is unknown what kind of encoding technology they use for encoding.(1)

As HEVC is still a young codec, there are few choices when it comes to encoding. Available maintained software encoders are *DivxLabs x265*,(10) *Kvazaar*,(43) the *Heinrich Hertz Institute's* HEVC encoder(5) and *MulticoreWare x265*(20) which is being used by ffmpeg. There are only two consumer feasible products for encoding HEVC via hardware fixed functions.
The first is Intel Media Server Studio.(22) Support is limited to more recent Xeon and 4'th, 5'th or 6'th generation CPUs. This framework also supports encoding VP9 via fixed hardware functions.
The second is the NVEncoder by NVidia. The NVEncoder supports encoding HEVC videos as of the second generation Maxwell GPUs as well as AVC on Keppler GPUs or newer.(30) For consumer (Geforce desktop) GPUs encoding is limited to 2 parallel encoding sessions per system. System in this case means per OS installation, which is a licensing limitation.(29)

2.2.1. Infrared Camera Flash

As lighting is a crucial part of the recording setup, it will also be considered thoroughly in this thesis. When it comes to IR lighting systems, there is a multitude of different IR camera flashes and IR lights which might be suitable for the use case of the Biorobotics Lab. For instance, the IR camera flash *Meike MK930*[1]. Flashes like this usually come as a camera mount and are battery powered, as in the given example. This might be an obstacle for the particular usecase but does not rule out the possibility to use it entirely. There are also IR flash cameras for animal surveillance like the *Browning Command OPs*[2]. These

[1]Specifications see `http://www.lightinthebox.com/meike-mk930-camera-flash-speedlite-for-canon-speedlite-dslr-400d-450d-500d-550d-600d-650d-1100d-vs-yongnuo-yn-560-ii_p1525323.html`
[2]Specifications see `http://www.trailcampro.com/collections/red-glow-infrared-flash-trail-cameras/products/2016-browning-command-ops?variant=17398568965`

recording systems come as an off-the-shelf solution with camera-flash synchronisation and usually have flashing intervals of up to 5 seconds. Simple boards with LEDs are also available. Some of them are easy to power manually to acquire a flash. High powered flashing LED systems do not seem to exist off-the-shelf.

2.3. Post-Hoc Encoding

As mentioned previously in section 1.3.1, the Cray supercomputer provided by the Zuse Institute Berlin is available for computation for the Beesbook project. The previous section 2.2 has listed all available software and hardware encoders. Hardware encoders inarguably outperform software encoders in regular desktop scenarios. This is intuitive since they are dedicated hardware which implements some or all of a software encoders functionality. However the Cray supercomputer offers vast computational resources in form of CPUs.(3) Software encoders offer a fairly more fine grained configuration interface than hardware encoders, leaving more space for optimization. (42) (31) Due to the given CPU resources software encoders have been used in the post-hoc case.

3. Supercomputer Assisted Large Scale Video Encoding

In the legacy system the Beesbook project stored recorded image files in JPEG format bundled in TAR files on the Cray supercomputer provided by the HLRN. These account to approximately 132 TB for the recordings of season 2014 and 157 TB of season 2015 as mentioned in the introduction section 1.3.1.

Storing the recordings in JPEG format requires more space then video compression. This is intuitive since JPEG only compresses a single image at a time, not exploiting temporal redundancy. Using suitable video compression the recordings can be processed in a much more effective way. The reasons to improve the compression of the recordings are simple. First, it is necessary to acquire a scientifically sufficient amount of statistics. Observing a single colonies for two or three seasons is hardly sufficient. If more seasons could be observed, models could be verified better and the weight of any discovery may be greater. But storing more than three seasons at most is beyond the limitations of the supercomputer using legacy JPEG compression. Also using the current compression schemes, the data throughput per second from recording station to final storage is high. A raw image is 4000x3000 pixels in size, yielding 12 MB of data. Compressed as JPEGS at 80% quality the image accounts to round about 3 MB. At 3 FPS this requires a connection an storage throughput of 3 MB * 3 FPS * 4 Cameras = 36 MB/s. Acquiring more frames at higher quality is desirable, yet hardly feasible. This chapter elaborates different compression schemes, quality considerations and finally describes how the data was re-encoded in the advanced system. It describes work package 1 as per section 1.2.

3.1. Hardware and Software Facilities

The HLRN is supporting the project by offering accounts to the supercomputer, a file quota of about 800 TB maximum and a certain amount of NPL. NPL measures the amount of computation time a certain account may use. The supercomputer is organized into different kinds of nodes, however for this project only the computation nodes and management nodes are relevant. See table 3.1 for all relevant nodes. The NPL only accounts to computations on the computation nodes while managements nodes may be used at will, but they are shared between users and the number of nodes is limited. They are not meant to be used for computing, but for managing data input and output of the computation

nodes. The computation nodes run in a batch system. This means there is a set of nodes and a job queue. Jobs may be submitted by a user to the job queue and are processed by a subset of free nodes. The Scheduler prefers jobs occupying more nodes.(3) However, a scheduled amount of larger jobs may leave some node offcuts. Smaller jobs may trickle into these offcuts. This is also called *Sandkornprinzip*.

The login nodes major purpose is interactive use, as any process running longer than

node type	#CPU	CPU model	CPU #cores	CPU clock	total RAM
login(1..2)	2	E5-2670	8	2.6GHz	256GiB
login(3..4)	2	E5-2650v2	8	2.6GHz	256GiB
data(1..4)	2	E5-2609	4	2.4GHz	64GiB
MPP2 compute	2	E5-2680v3	12	2.5GHz	64GiB
MPP2 compute	2	E5-2680v3	12	2.4GHz	64GiB
(...)					

Figure 3.1.: Table of relevant node types in the cray supercomputer. For the full list see: `https://www.hlrn.de/home/view/System3/CrayHardware`

approximately 30 minutes will be killed automatically.

Software is provided via modules.(15) Using modules users can easily load, unload and switch environmental applications. A variety of software and software libraries is available for loading via modules on Cray. For instance cmake, automake, QT, OpenCV, etc.. Unfortunately for the encoding task an encoding library and possibly a front end is required which are not provided. Consequently these and their dependencies have to be built manually. There exist many off-the-shelf solutions which may be able to do the task. As the front end the application *ffmpeg* was chosen, *HEVC* as the video encoder and Matroska as the container format. Using ffmpeg we can easily read different input formats, encode the videos and mux them into any common container format. Furthermore it offers convenience features, such as glob pattern input. Code adjustments to Wichmanns observer, the choice of codec and choice of container are elaborated in the subsequent chapters.

3.2. Code and Adjustments

Wichmann has developed at tool for the project to batch analyse the recorded images on Cray.(45) The major functionality is as follows:

- Extract and distribute a folder of TAR files contents to several job directories.
- Each job may occupy several computation nodes, thus having a variable amount of cores.
- Jobs are automatically submitted and run while the next jobs are being prepared and submitted up to a configurable limit.

- Job termination is being detected and results are saved to a results folder.
- Clearing the input data is left to the process. Not deleted input will be considered unprocessed, retrieved by the image distribution machinery and eventually redistributed for procession.
- In case of a crash the process may be restarted at any time thanks to a python shelf file.

Although slight alterations are required, this software is a well suited foundation to organize the encoding process.

The distribution of images was changed into a more sophisticated manner as round robin will break the sequential order of the images. For each core a given number of subsequent images X shall be supplied. Sequentiality is determined by the file name of the images. Every image is named according to pattern 3.1.[1]

$$Cam_\#CAMID\#_YYYYMMDDhhmmss_iiiiii.jpeg \tag{3.1}$$

Iff there is an image file with a time stamp less or equal one second later than its predecessor it is a subsequent image. The file naming enables ordering them lexicographically and hence find a subset of sequential images in a subset of extracted files easily using conventional sorting algorithms. If there is no sequence of size Y with $Y \geq X$, then the core will be supplied with Y images only. This will result in shorter videos and shorter encoding time which will eventually lead to wasting a small amount of NPL. In those cases it can be hardly be avoided that some cores may be idle while waiting for the other cores on a node to finish.
The resulting videos are named by the pattern according to pattern 3.2.

$$\begin{aligned} &Cam_\#CAMID\#_YYYYMMDDhhmmss_iiiiii_TO_ \\ &Cam_\#CAMID\#_YYYYMMDDhhmmss_iiiiii.mkv \end{aligned} \tag{3.2}$$

Deleting the input is delegated to the results collecting process. This is because deletion can not be done atomically in this case. In the previous scenario, each image was analysed independently. So if the image exists, it was not analysed fully yet, otherwise it is. A video encoding process would need to delete some thousands of input images. If the process gets killed while deleting, the original amount needs to be recalculated from the output file name. But it is also necessary to check whether the video was encoded completely. It may have aborted, leaving a broken video file. To simplify this, ffmpeg was modified to create a *.dne* file when done with error code 0. This signals the results collector that the images may be deleted and the output is correct. If this file does not exist upon completion all output will be deleted and the input retrieved. There are cases in which a minor error, like

[1] Recent adjustments use UTC full length timestamp format

a single broken frame, will still lead to ffmpeg terminating with error code 0. In these cases the error will be printed to console. To detect and resolve these issues, the console output is forwarded to a text file and analysed whether it contains *error* or similar keywords. If so, all the input and output is moved to a separate error directory for further manual investigation. This approach seems feasible, as this happens rarely. Retrospectively this happened twice in 130773 video files of an average frame count of 1024 frames. In both cases a single image was unreadable.

3.3. Choice of Codec

Rerabek et al have shown that HEVC is currently the best codec in terms of ratio of file size and quality.(36) The average bit rate savings compared to AVC was calculated to be 57.3% and 33.6% compared to VP9. In terms of perceived quality, as measured in PSNR, HEVC was similarly ahead. However, the image data of Beesbook is different from the sample videos used in any papers to the best of our knowledge. The main difference is the static background, grey scale as colour space and a frame rate of around 3 images per second. For this reason HEVC compression was compared to AVC using ffmpeg as seen in appendix A.1.

The experiment has shown that HEVC performs worse than estimated from Rerabek et al's findings, but still performs significantly better than AVC. The file size of the HEVC video is 56% to 71% the size of the AVC counterpart at roughly the same quality according to the codecs CRF function. This is not an exact comparison, as the CRF implementations may vary.(8) It is, however, good enough to verify the results of Rerabek et al. The PSNR has not been used here, as there is no proven correlation between PSNR and the classification accuracy of computer vision algorithms.

3.4. Container Format

Beesbook is comprised of many different sub-projects, each relying on the recorded informations. Thus, the video data will be accessed by many heterogeneous clients. The major focus has been set on load-ability in C++, Python, the JAABA framework and consequently Matlab.[2] It also needs to be capable of storing videos of 3 FPS frame rate and ideally officially support HEVC as contained video stream. Multiple sources state that Matroska is the only container to officially support HEVC besides MPEG4.(2)(24) MPEG4 is supposed to work well, but does not support a frame rate of 3 FPS.(9) HEVC containing Matroska containers can be loaded and decoded using the OpenCV framework in C++ and Python. This has been tested in Ubuntu 15.10 using the sample code provided along with this thesis. The JAABA framework is based on Matlab is also able to load Matroska containers. Matlab uses the gstreamer plugin on Linux operating systems and

[2]The JAABA framework requires the Matlab Compiler Runtime: `http://jaaba.sourceforge.net/`

Direct Show on Windows respectively to decode the video stream.(21) Both are verified to be compatible with MKV/HEVC.(2)

3.5. Compression Quality

For encoding the existing images of season 2014 and 2015 research on the optimal encoding quality and encoder parametrization was done. When encoding a video one will encounter an optimization triangle of file size, encoding time and video quality as depicted in figure 3.2. To solve this problem, first an optimal quality configuration was chosen using the current tag detection pipeline of the Beesbook project. Secondly an encoder configuration was found which compresses very well on a single CPU core in feasible time. So the bias here was set on file size.

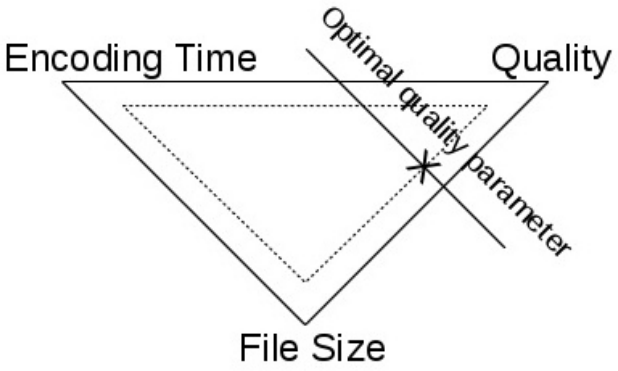

Figure 3.2.: Parameter optimization problem. The dashed line indicates the point at which optimization of one parameter comes at inappropriately high cost of the other parameters. The 'optimal quality parameter' line is found using the pipeline. The optimal configuration point X is determined manually.

3.5.1. Choosing the Quality

For later analysis of the videos it is not important how the videos look, but how well the tags are detectable by the pipeline. For this purpose, evaluation of the quality is solely based on the pipeline. The tags also should still be human-readable, but this criteria is likely to match machine-readability. In the pipeline, tags are processed in multiple steps: Preprocessing, Localizing, Ellipse Fitting, Grid Fitting and Decoding. The approach of reading tags is the following. Tags are localized first and false positives are filtered using a neural network. Details are described by Wario et al.(44) For every found 100x100 pixel partition of the image a 3D model of a tag is fitted on the image partition. This requires finding the orientation, rotation and size of the tag. Finally the decoder can read the bits from the 3D model. For a few images Ground Truth data is available.[3] The Ground Truth data was manually created and describes all tags in the respective image almost

[3]Available online at: `https://github.com/BioroboticsLab/deeplocalizer_data`, retrieved 13.06.2016

perfectly. The scheme for finding the optimal quality setting is to encode a video containing the Ground Truth frame in different quality settings. Then the pipeline is run on this frame and the Ground Truth frame and performance is being compared. Comparing the in HEVC re-encoded frame with the original JPEG pendant seems biased, as there will always be a quality loss unless encoding lossless, which is not feasible. In fact, the used version of the NVEncoder will crash upon using lossless encoding settings.[4] But as will be shown, in near-lossless situations the performance does barely change but for noise.

As a quality metric for encoding the CRF parameter was used. CRF encoding uses a variable bitrate in general. However, *how* the bitrate is chosen in accordance to the parameter is implementation dependent. So it is only safe to compare results within a certain encoder implementation. Recent encoders also involve the framerate in the CRF function.(8)(21) The following measures are based on 25 FPS. Videos on Cray also have been encoded using 25 FPS for the sake of comparability. In a later muxing step, videos have been changed to 3 FPS and muxed into MKV containers. The reason to do encoding and muxing in two steps is, that the HLRN NPL for the Cray had to be used by a given deadline. However the Biorobotics Lab committee did not finally decide on the requirements on muxing information at that given point.

Three different measures have been developed. Let E_x be the encoded frame of CRF value x in the video and G the corresponding Ground Truth frame. Let there be a set of tags T per frame and y $\in$ 1..*Number of tags in the frame*. Every single tag can now be identified as T_y for its respective frame. Y is and index set of the tags in the Ground Truth frame.

First, compare the pixel coverage of the entire mid ring of a tag. Let there be the set of pixels $S^O_{E_x,T_y}$. These are the pixels covered by the mid ring of tag T_y in E_x and S^O_{G,T_y} in G respectively.

$$\sum_{y \in Y} \frac{|S^O_{E_x,T_y} \cap S^O_{G,T_y}|}{|S^O_{G,T_y}|} / |Y| \tag{3.3}$$

Second, compare the pixel coverage of every single mid ring. Let there be the set of pixels $S^I_{E_x,T_y,M_z}$. These are the pixels covered by the mid ring section M_z. Let M be the set of mid ring sections and z an individual section, z $\in$ 1 .. 12.

$$\sum_{y \in Y} \frac{\sum_{z=0}^{11} |S^I_{E_x,T_y,M_z} \cap S^I_{G,T_y,M_z}|}{\sum_{z=0}^{11} |S^I_{G,T_y,M_z}|} / |Y| \tag{3.4}$$

[4]Tested using video driver Linux-x86_64-358.16, Cuda 7.5.18, NVEnc 5.0.1

Third, compare the hamming distance of the decoded tag bits in S_{E_x,T_y} and S_{G,T_y}.

$$\sum_{y \in Y} HAM(S_{E_x,T_y}, S_{G,T_y})/|Y| \tag{3.5}$$

The index y in T_y per algorithm design always points to the best fitting tag in regard to the Ground Truth tag set. For S_{G,T_y} there is obviously a one-to-one relation. For S_{E_x,T_y} the best fitting tag is selected, disregarding which candidate is the best choice according to the pipeline. These measures have been plotted as performance against CRF value. As all of them indicate the same point of decreased detection performance, only one of them is shown in figure 3.4.

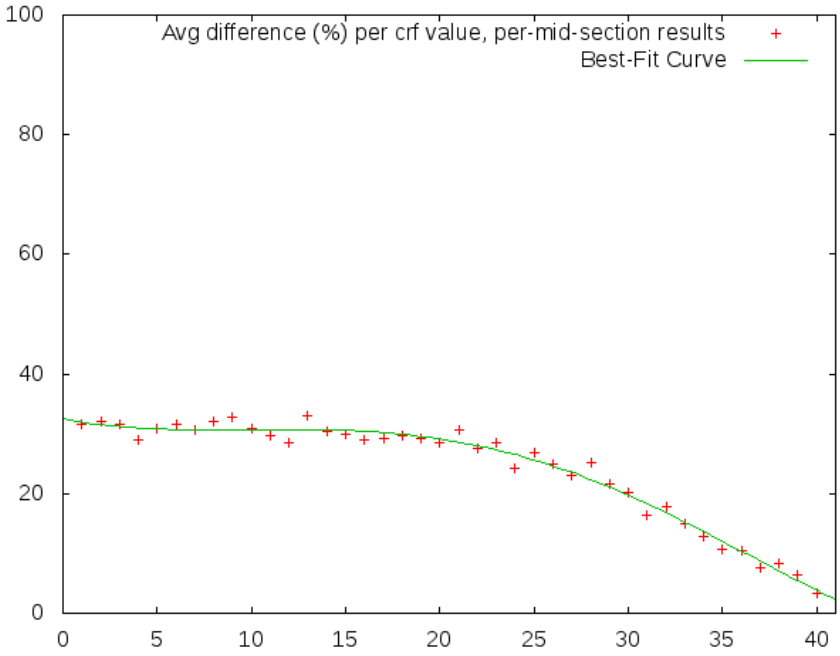

Figure 3.3.: Average difference in % per CRF value, mid-ring results for camera 1. An insignificant slope can be seen from 0 to 20, which is dominated by noise. At roughly 20 the downturn starts and becomes significant at 25.

As it is yet unknown which algorithms will be used in the future, CRF was chosen to be 20. Besides this insight the plots revealed that the performance per camera is different. Unfortunately these measures are not feasible to be used for camera calibration in the future, as each calibration step would require creation of new Ground Truth data.

As can be seen in the formulas 3.3 to 3.5, poor localizer performance is penalized strongly. A failure in the localizer accounts to zero pixels for the entire respective Tag. This was found to be acceptable, as the localizer is the most reliable but important component. Not penalizing a localizer failure strongly would result in high evaluation noise in videos of poor quality. E.g. when not taking into account not localized tags, a blank image would perform perfectly. In a blank image all found tags would have been decoded

perfectly, which are none. Having found the optimal quality parameter for the video, the the optimal configuration X as in figure 3.2 can be found.

3.5.2. Choosing the Size and Encoding Time Trade-off

The HEVC encoder has a multitude of options which impact the file size to encoding time trade-off. To simplify this for end users, the CLI encapsulates 29 of them in the *-preset* switch.(20) The user may choose from *ultrafast* to *veryslow* or *placebo*. However, these were optimized for natural videos. The beehive recordings are special in terms of colour, framerate and so on. Finding the optimal solution here means doing a gradient descent on 29 dimensions. However a gradient descent will only find a global optimum if the only local optimum is the only optimum. It can be assumed that it is not the case that the loss function is convex, meaning there is more than one local optimum in this parameter search.

As encoding a single sample takes significant time[5], exhaustively doing this kind of parameter search is not feasible. Instead a more manual approach was done. First, a suitable preset parameter was found at which the improvement in filesize becomes insignificant. As

filesize	preset	gain
22613412	superfast	-
14841618	fast	34.37%
14889477	medium	-0.32%

Figure 3.4.: Gain in filesize compared to the respective previous preset. The gain here describes the percentage at which a file is smaller than it's predecessor. E.g. if A is 100 MB and B is 90 MB the gain from A to B is 10%. Irrelevant results have been truncated. As it can be seen, there is a significant gain from *superfast* to *fast*. From *fast* to *medium* the gain is small enough to be masked by noise which explains the negative gain.

shown in figure 3.4, the gain from *fast* to *medium* is -0.32%. The gain is small enough to be covered by noise. For this reason *fast* was chosen to be the optimal preset. The documentation now yields suitable start values to simplify the manual parameter optimization.

A script was created to further optimize parameters thoroughly. While specifying some of the parameters set by the preset *fast*, the preset has to be set explicitly as well. The preset is applied first, then the fine grained parameters, regardless of the ordering. The parameter description of *–ctu* states the following: *The larger the maximum CU size, the more efficiently x265 can encode flat areas of the picture, giving large reductions in bitrate. However this comes at a loss of parallelism with fewer rows of CUs that can be encoded in parallel, and less frame parallelism as well.*(20) CUTs are *Coding Tree Units* and describe blocks of size 8x8 to 64x64 in which HEVC fragments the input image. A CTU gets partitioned into CUs, *Coding Units*. The videos contain large portions of what basically is background. High *ctu* settings will encode this well but penalize parallelism. As every

[5]Encoding 30 frames ranges from 128s to 573s on the test hardware. For details see the CD.

video is encoded on a single core, the penalty does not apply in this usecase. Furthermore it seems reasonable to choose the minimal cu size to be large. This is because the framerate is low and hence the objects move large distances between two frames, compared to natural videos.

Further parameters, namely *early-skip*, *rd*, *fast-intra*, *tskip-fast* and *rdoq-level* may considerably improve encoding time, but the penalty on filesize is not clear and were hence tested. The HEVC encoder estimates motions in the video throughout time and creates a so called motion vector. *me*, max-merge and *merange* are impacting the search pattern, length and temporal resolution of these motion vectors.(14) It is neither obvious nor determinable other than through testing how well motion estimation works using small vectors. Again the object motions in the videos are likely to be large due to the low framerate. For this reason, increasing the search range was evaluated. Additionally to the performance optimizing parameters, the two parameters b-adapt=0 and rc-lookahead=5 have been set. This will reduce the memory usage of the encoding process, which is necessary as it may run out of memory otherwise. The impact on the motion estimation is thought to be minimal, as the temporal resolution is sparse and hence normal parameters may hardly be applicable. All test results are attached to this thesis on CD. The used parameter set can be seen in listing 3.1.

Listing 3.1: Final Parameter Set

```
-preset fast -x265-params rd=2:ctu=64:min-cu-size=32:
  max-cu-size=64:early-skip=1:fast-intra=1:tskip-fast=1:
  rdoq-level=0:me=1:max-merge=5:merange=114:b-adapt=0:rc-lookahead=5
```

3.6. Verification

The legacy image data acquired in the experiments in 2014 and 2015 is crucial to the project. Hence data loss must be avoided at any time. To guarantee the encoded videos are not missing any data, steps have been taken to validate them.

3.6.1. Timelines

Every video has a start timestamp and an end timestamp in its title. From these timestamps timelines have been created. For verification purposes it is not relevant whether a certain period of time is available in one or multiple videos. The purpose of the timelines is to verify whether they are available at all or not.

To generate the timelines, all timestamp pairs have been put into a set. If the end time of one matches the start time of another, they are merged into a single pair. If they overlap or one contains the other, they are also merged. These steps are done exhaustively

on the entire set.
A similar process was done on the original images as well. If and only if two images are consecutive by lexicographic order among all images and their timestamps differ by less than one second, they are considered consecutive. The algorithm sorts the images by lexicographic order and go through them one by one, following the order. Along the way, pairs equivalent to the videos pairs are created, if the videos have been created correctly.

The resulting sets of pairs are effectively timelines. They have been checked for discrepancies using a python script and eventual gaps have been fixed manually. Additionally known downtimes may be inserted into the sets and annotated. The sets finally have been visualized using the google timelines API.(16) As displaying an entire season in one timeline might not be appropriate in terms of size, there is also a version which splits the timelines day wise. An example is given in figure 3.5.

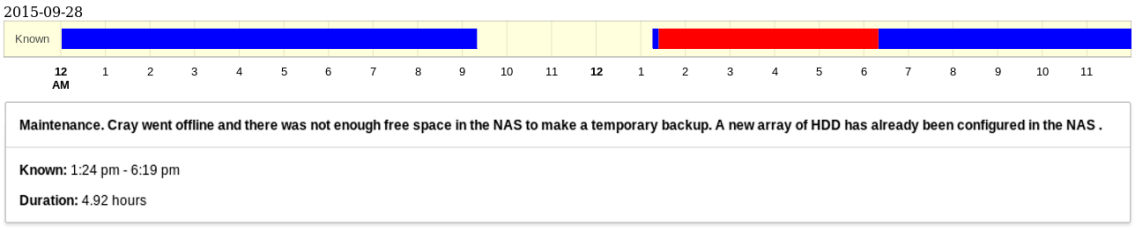

Figure 3.5.: Timeline of the video data of 28.09.2015. Hovering a section may give details. Annotated sections are always red, found video data blue and gaps blank. In this example, details for the annotated downtime (red) are shown in the info box below.

3.6.2. Frame Counting

In the preceding section the timelines have been used to verify completeness of the video data. However one can not be sure that the video contains the amount of frames according to the timespan in its title. This means it is not sure for an individual file whether content is consistent to the title. Inconsistencies might occur due to errors while providing the images and erroneous images which get skipped during creation of the video. To ensure consistency the frames in every video have been counted using *ffmpeg*. Frame counting can be easily done using the command in listing 3.2. Again found gaps and malformed videos have been fixed manually.

Listing 3.2: Counting frames using ffmpeg

```
ffmpeg -i VIDEO -vcodec copy -acodec copy -f null /dev/null 2>&1
    | grep 'frame=' | cut -f 2 -d ' '
```

4. GPU Assisted Video Recording

This chapter elaborates work package 2 as described in section 1.2. That is, transferring and improving the legacy hard- and software into advanced ones. This includes improvements to the lighting, camera calibration, image compression and finally the file transfer. In the legacy system the hive was illuminated with constant IR light of 84 0nm. This required a large amount of IR lights: 22 lamps each having 24 LEDs. The LEDs have very directed light (approx. $\pm 5°$) and hence produce a great deal of reflection on the glass pane and the tags. The lighting from one side is back lighting to the cameras of the respective other side. This is unwanted, as it decreases contrast in the recorded images. The ambition is to create more diffuse lighting and reduce or eliminate back lighting.

Previously the recording application *bb_ImageAcquisition* saved the retrieved images in JPEG format, all of the configuration parameters being hardcoded. The JPEG images were saved to a ramdisk, as the produced workload was large. A single image was approximately 3 MB. This results in a workload per camera as calculated in equation 4.1.

$$4\ cameras\ *\ 3\ frames/s\ *\ 3\ MB/frame\ =\ 48\ MB/s \tag{4.1}$$

The images were then stored in a TAR archive and again written to the ramdisk. As TAR does not use compression,(25) this causes in one read and two writes of the workload calculated in 4.1. The archives are then sent to a long term storage on the HLRN supercomputer Cray, as well as stored on a local NAS. The NAS is caching the data in case of Cray failure or maintenance, as well as storing stripes of data every day on long term. Doing this concludes in one more read. So the workload totals to 2 reads and writes of each image, which are 192 MB/s. This can not be handled by a consumer hard drive. It would also greatly impact the lifetime of multi hard drive or SSD setups. The recordings last 9 weeks, which concludes in a data amount as per equation 4.2. The calculation is an estimate. Since there are maintenance breaks, failures and varying file sizes, the actual recorded amount is 157 TB. Hence equation 4.2 may be considered a worst case scenario. This scenario does not cover any gaps in the data, like gaps due to long Cray maintenance or system failure.

$$48\ MB/s\ *\ 604800\ s/w\ *\ 9\ w\ =\ 261273600\ MB\ \approx\ 261\ TB\ ^{1} \tag{4.2}$$

The objective here is to reduce the workload and most of all, reduce the total amount of

[1] s/w is seconds per week.

produced data.

All of these targets are contained by a very limited yet not strictly defined financial resources.

4.1. Software

In section 2.2 the state of the art for video encoding was elaborated. Choosing which technology to use comes with certain obstacles. First of all, is live software encoding feasible? Video encoding in general is a very computationally laborious task. The recording machines must be able to encode 4 videos at 4000x3000 pixels and 3 frames per seconds in real time. As shown in the appendix in table A.1, the fastest tested AVC software encoding test ran in 1.5 s. This is just about feasible for real time encoding. There are 4 cameras at 3 frames per second, eventually producing an average workload of 60% to the CPU. However the compression ratio is poor compared to the HEVC codec or even h264 at other configurations. This may be improved to an unsatisfying amount by using 4 recording machines. Looking at the results of the post-hoc encoding, it is known that a season may produce 40 TB of HEVC encoded data. The table also tells us that there is a saving of 8MB from approx. 50 MB to 42 MB between certain settings. Thinking of 42 MB as 100%, 50 MB is round about 120%. 120% of 40 TB are 48 TB. A reasonably cheap local storage is the 8 TB HDD of Seagate with approximately 248€ per unit.[2] As file security is highly desirable, at last four extra TB are required to do some kind of RAID. So 8 MB extra in the table here result in 372€ extra cost per season. It can be concluded that hardware encoders are financially advantageous in this usecase.

The two feasible options for hardware encoding are the NVEncoder(30) and the Intel Media Server Studio.(22) As two systems and hence two NVidia GPUs are required, hardware costs are round about 330€[3] minimum. The cheapest Intel CPU having the desired hardware support appears to be the E3-1286v3 for 229€[4]. The major cost factor is the licence required to use the Intel Media Server Studio. The only licence with HEVC capability is the *Intel Media Server Studio Professional Edition* for 3999$.[5]
The NVidia GPU was chosen for multiple reasons. First, the GPUs are by far the cheapest solution. Second, other tasks such as mentioned in section 1.5 heavily profit from a powerful GPU. As the recordings of bees take approximately 9 weeks and the bumblebees

[2]Price per amazon, retrieved 21.06.2016: `https://www.amazon.de/Seagate-Archive-interne-Festplatte-Cache/dp/B00QGFEQXU/ref=sr_1_sc_1?ie=UTF8&qid=1466504694&sr=8-1-spell&keywords=seagote+8TB+storage`

[3]Price per amazon, retrieved 03.05.2016: `http://www.amazon.de/MSI-V320-059R-GTX-950-Graphics/dp/B013SUP76U/ref=sr_1_1?s=computers&ie=UTF8&qid=1462280394&sr=1-1&keywords=gtx+950`

[4]Price taken from Computeruniverse.net, retrieved 03.05.2016: `https://www.computeruniverse.net/products/90648641/intel-xeon-e5-2603-v3.asp`

[5]Price taken from Intel homepage, retrieved 05.05.2016: `https://software.intel.com/en-us/intel-media-server-studio/try-buy`

6 weeks per year, there is a decent amount of spare time for the GPUs to serve a dual purpose. Finally there is already one *Geforce 960 GTX* GPU available at the lab which could be used for feasibility checks and development. The Biorobotics Lab finally acquired two *Geforce 960 GTX* GPUs. The MSI edition having 4 GB of RAM was chosen in favour to the *Geforce 950 GTX*, which does not seem to come with 4 GB of memory.

4.1.1. Acquisition Application

For the acquisition and encoding task, existing software was extended. The application bb_imageAcquisition was developed by Fernando Wario in order to save the recordings in JPEG format back in 2014. A branch for video encoding was created and is publicly available.[6] Previously the camera configurations were fixed in the source code and for every camera there was a thread retrieving the images. The camera was triggered by its internal software. So the retrieving thread had to retrieve the image, store it as JPEG and log the event within 3 FPS. If the thread was to slow to do so, caused by random hard drive load for instance, data loss may have occurred.
The acquisition software was extended to process the input as displayed in figure 4.1.

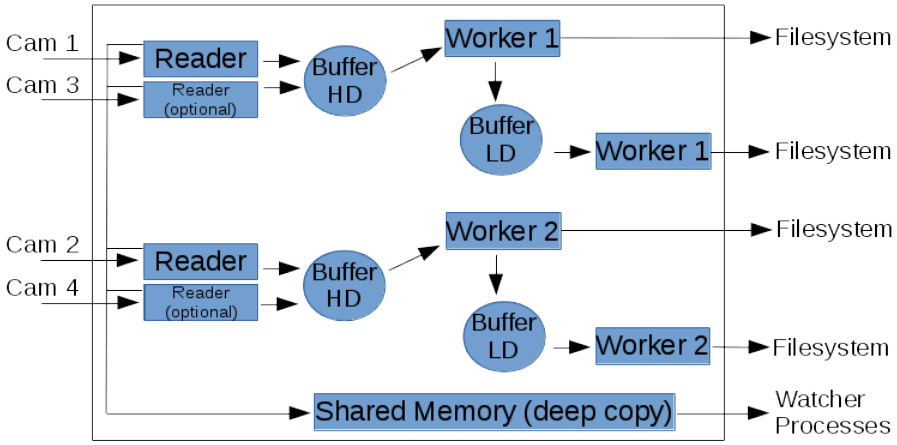

Figure 4.1.: Software architecture of the advanced acquisition application. Each worker may hold one encoding session at any time. Interconnections depend on buffer order and may be changed by switching around buffers as well. HD buffer entries 1 and 3 in the JSON file are connected to worker one and 2 and 4 to worker two respectively.

Each camera is still being read by a single reader thread. Every reader thread holds a pointer to a concurrent linked list buffer structure. Let this structure be the HD buffer. Each HD buffer has a corresponding configuration in a JSON file. This configuration specifies how the input is supposed to be acquired and how it will be encoded by the GPU encoder. Former implies all the camera configurations: Camera serial number, logical camera ID, resolution, shutter time, gain and so forth. Buffers are storing raw images. The GPU encoder unfortunately does not support monochrome chroma format, so the grayscale images are converted by the encoder thread first.(30) Let the grayscale value be

[6]Repository link: `https://git.imp.fu-berlin.de/bioroboticslab/bb_imgacquisition`

W, equation 4.4 can be derived from equation 4.3, as W = R = G = B.(40) The U and V channels always have the constant value of 128 and are hence omitted in the conversion process. Pointers to a constant memory only containing the same value are passed for performance reasons.

$$Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16 \tag{4.3}$$

$$Y = 0.895 * W + 16 \tag{4.4}$$

The workers are doing the encoding. A single worker may be assigned one or two HD buffers as well as corresponding LD buffers, each of them being registered to a *writeHandler* which will be elaborated later. LD buffers may be enabled or disabled via configuration file. Similarly to HD buffers, the LD buffers specify their input, but here the input has to be an HD buffer. Whilst encoding the HD buffer, the processed image will not be discarded but formatted according to the LD buffer and written to it. Quality settings for the encoding may be applied for every buffer regardless of kind via configuration file. In this fashion one may specify a high resolution and high quality HD buffer and a corresponding low resolution, low quality LD buffer.

The worker will always choose the buffer which consumes the most memory and is feasible for encoding, preferring HD buffers. LD buffers may be infeasible for the following reason. Assume an LD buffer has the exact same quality and resolution settings as its HD counterpart. For example, let the HD buffers frames per video be 10 and the LD buffers 11. As the HD buffers images are transformed into LD buffers alongside encoding, the LD buffer will hold 10 images after completion of encoding of the HD buffer. Upon selecting the next buffer to be encoded, the LD buffer may not be chosen, as it will wait for the 11'th frame infinitely. The only other mean to prevent this lock is to directly transform the HD buffers into LD or do the transformation on demand. However the use case is to have an LD video of a length which is k times the length of the HD video. So either of the proposed approaches would exhaust unnecessary amounts of memory. Switching between buffers while encoding is not possible, as only 2 encoding sessions may be opened at once due to licensing restrictions. Ending one encoding session effectively means breaking the exploitation of sequentiality, as compression over time works only within one encoding session. It would therefore render video encoding useless in the worst case. However, video streams can be concatenated.

The encoding function itself is based on the NVEncoder SDK examples. An encoder object is being configured and an encoder session is opened via framework functions. Images are retrieved from the buffers as mentioned previously in a loop and passed to the encoder framework function. Results are passed to the writeHandler. The input image is

transformed and passed to a LD buffer or discarded.[7] Once all the frames are encoded, the encoding session is being closed and the thread returns to buffer selection.

Unfortunately only one process at once may have access to the camera devices. However for many purposes it is desired to grab images at any time. For example to see the contents of the hive live without opening it and possibly disturb the bees. To enable this a simple shared memory interface was created. The shared memory holds a deep copy of the most recent image and timestamp, but does not hold any cache. The interface comes with two advantages: First, any operation on raw images may be implemented, patched and tested without interrupting the acquisition process. Second, the complexity in the acquisition process remains constant. New features are always critical to the stability of the recordings, which are highest priority because they have to run through several weeks of testing until considered stable.

Write Handler

Files are named according to the pattern as described in section 3.2. To separate data procession from output, the writeHandler class was created. The writeHandler only operates on the virtual camera ID, as well as the configurable output and temporary folders. To prevent the storage application to attempt to send malformed or unfinished videos to their final destination, they are held in a temporary folder. Once a buffer is associated with the writeHandler, a video data file and a textfile are created. A writing process may simply use the public lock handle to write a data frame. Then the log function has to be used to log the timestamp of the frames. They will be formatted according to conventions, written and immediately flushed to the textfile. Upon destruction of the writeHandler, text and data files are moved to the output directory. The log function will keep track of the first and last timestamp and name the video data file accordingly. If the application crashes at any point, partial video files can be recovered thanks to the text file. The text file holds the timestamps of all frames written so far. Having interrupted the encoding session will only harm decoding and playback of the video if half a frame was written back to the video data file. This happens if the application crashed during file writing. However this does not render the entire data useless, but will only cause errors upon processing the data.

4.1.2. Watcher Application

An application was created to watch the acquisition processes output. There are two main functionalities. The first is to always grab the most recent frame and display it in a window. This is done using the QT GUI framework. The second is to save the most current picture or a series of pictures as RAW images, JPG or PNG. For the later two

[7] Creation is optional and may be configured

formats QT functions are used as well, while RAW images are just a copy of the shared memory.

Live Statistics

Some image statistics are done live on the images every minute using the shared memory interface. Though the live statistics code is part of the original bb_imgacquisition code it functions as a watcher in a separate process. This is because the acquisition process uses the same code for the calibration task. This may be outsourced into a library and separate projects later.

Live statistics are done for multiple reasons. One may want to find out whether a camera got decalibrated during the running experiment. This might happen accidentally or on purpose, when feeding or changing the experiment setup slightly. As manual evaluation is a cumbersome and error prone task, support via image statistics is desirable. The statistics may indicate differences between cameras as well as whether a single camera has changed in recording quality. Additionally other informations can be derived. For example bee activity in general or in specific parts of the hive.

The question is, what properties can be used to determine the quality of an image? The focus was set on three measures: Noise, sharpness and contrast. Farooque et al described image noise as a random variation of brightness and colour in the image.(12) In the paper different kinds of noise are being identified and measures for removing noise are analysed. Our goal is not to remove, but only to detect and measure the quantity of noise. Some kinds of noise are of much less interest, because we can not prevent them from occurring. An example for this is quantization noise. Quantization noise may occur due to discretization of image values. It usually is uniform for the entire image and hence can hardly be filtered by a median filter, for instance.
According to Farooque, gaussian noise is *sensor noise caused by poor illumination and/or high temperature, and/or transmission e.g. Electronic circuit noise.* These issues may be resolved by changing the hardware setup and shall be detected and measured. A mean to remove this kind of noise is the median filter. The idea is now to filter an image using the median filter and calculate the sum of square differences of the original and filtered images according to formula 4.5. Here N and M are the respective image dimensions and E is the median filter operation.

$$Noise(I) = \frac{\sum_x^N \sum_y^M |I(x,y) - E(I)(x,y)|^2}{N * M} \tag{4.5}$$

The same approach can be applied to image contrast. According to Starck et al an image can be contrast enhanced using histogram equalization, wavelet- and curvelet transformation. (39) It was concluded that curvelets enhance contrast the best, particularly if there is a considerable amount of image noise. In our usecase however we do not want to

enhance an image but rather find a measure for contrast. In the setup some regions of the image are rather irrelevant, as they are not in the scope of the experiment setup. So improving contrast by lighting in regions exterior to the setup may improve the measure although relevant sections of the image may or may not become worse. There are two intuitive solutions to this problem. The first is to crop the image to the relevant sections. The second approach is to compare the image to an image having good contrast. The good image can be acquired by taking a picture from the camera and enhance the contrast using third party tools. Then the image is provided as *refIm.jpg* to the application and colour histograms are compared. Again the sum of square differences is taken. Additionally the contrast ratio has been calculated. The contrast ratio is defined as lowest pixel value divided by highest pixel value found in a ROI as illustrated in equation 4.6.

$$ContrastRatio(I) = \frac{min(I)}{max(I)} \tag{4.6}$$

Obviously this metric is prone to errors caused by outliers, such as dead pixels and over-exposed areas which are of no interest. For this reason the image is first filtered with a median filter. Then five ROIs are selected in the pattern of the five face of a regular dice where the size is adjustable. Finally the average of the five regions is taken.

Sharpness is the most important factor in adjusting the setup. Most digital cameras are able to focus the camera lens automatically. However the hardware being used in section 1.3.1 requires manual focusing. There have been several studies about how to find the overall sharpnes of an image. Gross et al have been evaluating variance, sum modulus difference and perceptual sharpness metric for video colonoscopy.(17) Moreover Chern et al have compared a multitude of methods as depicted in figure 4.2 Results are sobering, as almost all thinkable methods are able to do the job. Yet there are differences in the performance of the algorithms. For example the steepness of the sharpness function near the sharpest point and the computation time. The sum modulus difference and perceptual sharpness metric have been chosen for implentation. SMD chosen because it is computationally inexpensive and performed well in both mentioned papers. Calculation was done as in equation 4.7 according to Gross's paper.(17)

$$S_{SMD}(I) = \sum_{i=1}^{M-1} \sum_{j=1}^{N-1} (|I(i,j) - I(i-1,j)| + |I(i,j) - I(i,j-1)|) \tag{4.7}$$

The perceptual sharpness metric was chosen since it was not analysed by Chern et al and hence completes the view on suitable sharpness metrics. Here the image is split into blocks of 8x8 pixels. Each block is applied an discrete cosine transform as in quation 4.8.

$$DCT_{m,n}(k_1,k_2) = \sum_{i=0}^{7} \sum_{j=0}^{7} x_{i,j} cos[\frac{\pi}{8}(i+\frac{1}{2})k_1] cos[\frac{\pi}{8}(j+\frac{1}{2})k_2] \tag{4.8}$$

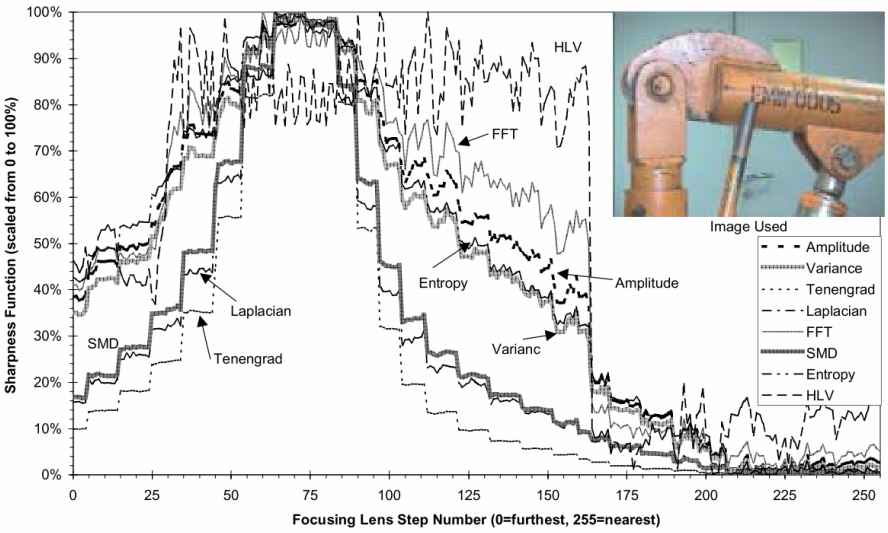

Figure 4.2.: Plot on the performance of image sharpness functions. It can be seen that almost all intuitively meaningful functions work reasonably well to for finding the correct global maximum. (28)

The coefficients of the first row and column are then weighted according to equation 4.9 and added as in equation 4.10.

$$SSF(d) = d^{0.269}(-3.533 + 3.533d)exp(-0.548d) \tag{4.9}$$

$$\hat{S}_{m,n} = \sum_{d=1}^{8} SSF(d)[DCT_{m,n}(0, d-1) + DCT_{m,n}(d-1, 0)] \tag{4.10}$$

Finally the blocks are added and weighted by image size as in 4.11.

$$S_{PSM} = \frac{1}{M * N} \sum_{m=1}^{M/8} \sum_{n=1}^{N/8} \hat{S}_{m,n} \tag{4.11}$$

All of the the functions described in this section have been implemented. Their results are written live to a textfile along with the timestamp of the corresponding image and camera ID.

Memory Footprint

Images in buffers are handled using the standard libraries shared pointers. The reference counting of shared pointers guarantees the memory to be freed once at no point in the code a valid reference to the memory is left. This effectively means that the reader can simply pass the same pointer to the HD buffer and the shared memory thread. Despite the threaded nature there is no need to negotiate when or by whom the memory has to be freed.

Memory consumption is highly dependent on the number of images which need to be

cached before processing. This number depends on the number of frames per videos, whether LD videos should be done or not and the HD buffer to worker connections. Images are held uncompressed in buffers.

The usecase is HD images of dimensions 4000x3000 consuming 12 MB of memory and LD images of dimensions 2000x1500 consuming 2.86 MB each. The videos are of length 1024 frames HD and LD at 3 frames per second. There are two workers per system and one encoding session per worker. According to NVidias documentation the Maxwell 2'nd generation HEVC encoder is able to process at least 133 frames per second, depending on configuration.(30) So a buffer containing 1024 images can be processed in approximately 7.7 s. Let us assume two cameras are connected and assigned two different workers. In this example, each may only process its assigned HD buffer or the corresponding LD buffer. As the HD buffer is processed, the LD buffer will accumulate 2.9 GB of data. In the process of encoding the LD buffer, the HD buffer will again acquire 288 MB of images. That is, image size into images per second into encoding time rounded up. As evaluating build up in HD against freed memory from LD buffer may be fragile to lag spikes caused by external applications, it seems safe to round the total memory usage up to 3.4 GB per worker. This results in 6.8 GB total peak memory usage in this use case.
Consider another use case. There are four cameras of the same configuration in one system. Now also two HD buffers need to be cached, resulting in four buffers served per worker in total. While encoding HD buffer 1, HD buffer 2 and LD buffer 1 build up memory to 12.3 GB + 2.9 GB = 15.2 GB. Encoding HD buffer 2 causes LD buffer 2 to build up. After this there is a total memory consumption of 288 MB (HD 1) + 6.8 GB (LD 1+2). Next up are the two LD buffers. Eventually the peak memory usage is 15.2 GB per worker, concluding in 30.4 GB per system.

Calibration

The application has a separate calibration mode. This mode may be launched by running it with the parameter *–calibrate*. It will not do the normal recording procedure, but only calculate the quality measures as described in section 4.1.2. This is done in real time in a best-effort manner. As the whole task is computation extensive, the full framerate might not be achieved. Hence it is advised to calibrate only one camera at once. The results are printed in the console, formatted as a table. The user may now change the setup to optimize the quality measures live.
However, for some setups they might be misleading. For example, in the bumblebee hive the focal point is desired to be on the hive. The hive is somewhat higher than the floor. Since the floor consists of cat litter it has a lot of structure. As a result, an optimal perceived sharpness metric will have the focal point on the floor. This issue may be tackled by setting the resolution and region of interest of the camera accordingly. This will enable the user to set the focus an arbitrary rectangular subimage.

4.1.3. Storage Management Application

The storage management application is called *bb_imageStorage* and was originally created by Christian Tietz.(41) The legacy version accumulated the images created by *bb_imageAcquisition* into TAR archives. These archives are then sent to the Cray supercomputer. Features were as following.

- Transfer TAR archives automatically to Cray using SCP and keyfile authentication.
- Cache files to a temporary directory in case of Cray failure or maintenance to a separate directory.
- Upload cached files automatically as soon as possible.
- Monitor Cray status and alert the user via console ouput.
- Store stripe data to a separate folder as per given schedule.
- All sources and targets are local folders, except for the Cray. To use a NAS, folders need to be mounted locally.
- Sources and targets are hard coded. Changes require recompilation.
- Qmake project files for compilation on Windows and Linux

The functionality was altered to meet the requirements of the video recordings, though the workflow remains unchanged. The project was converted to CMake to meet the Biorobotics Lab standards and all configurations are now done via JSON configuration file. Accumulating files is no longer necessary and hence file IO has been reduced. Video files and textfiles take the place of the TAR files. The application can and has to distinct between LD videos and HD videos. They are named identically and are uploaded to different directories on Cray. In case of Cray failure or maintenance, LD videos will be discarded as they can be reproduced from the HD videos.

4.2. Hardware

4.2.1. Lighting

As mentioned in section 1.1, to acquire pictures in a good quality lighting is crucial. The legacy recording system, as introduced in section 1.3.1, used a high number of constant lights. This setup comes with many disadvantages.

- Conventional LEDs have a small lighting angle and hence cause high reflection on tags, glass panes or similar.
- Positioning a high number of lights to light the scene equally well is an elaborate task.

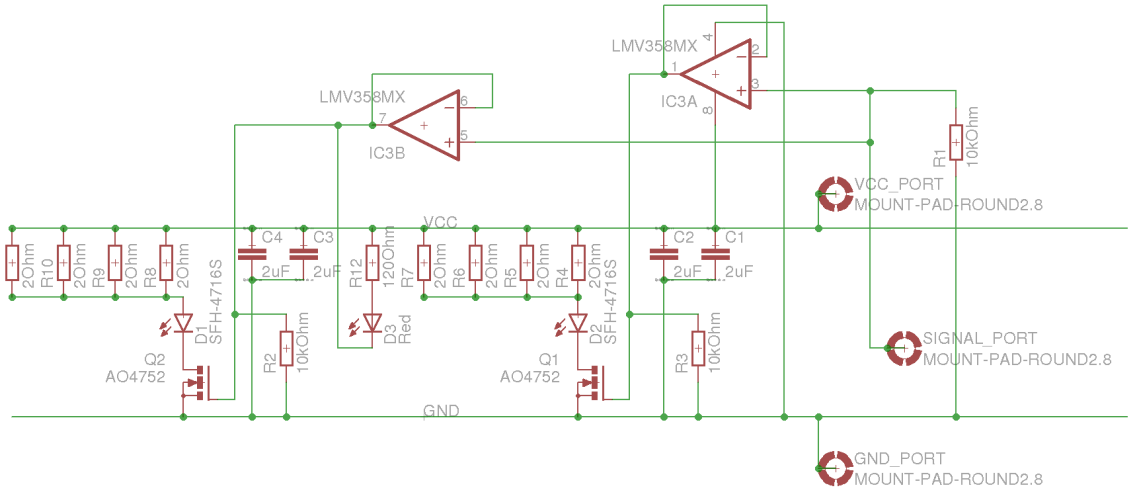

Figure 4.3.: Circuit diagram of the LED boards.

- Most of the light emitted is not required. 3 frames per second and a shutter time of 5ms result in 1.5% of the light actually being used.
- The overall system is expensive. It accounts to round about 1300€, as calculated in appendix A.2.

Board Design

For the advanced hardware setup new LED boards were designed. They run at 5V. Hence regular computer power supply units can be used to power them. Each board has a status LED indicating three different statuses. In diagram 4.3 this is LED *D3*. Off means there is no power supplied to the board. The red LED *D3* being on means there is power, but the board is not flashing. Flickering means that the board is flashing, where one off-phase indicates one flash. The resistors R4 to R7 and R8 to R11 respectively are parallel series resistances to the LEDs. As the power consumption of 4 A is rather high, four resistors have been chosen to prevent overheating. Q1 and Q2 are MOSFET gates to switch the LEDs on and off. The power supply may not adequately supply the power instantly, as the surge current is high. The capacitors *C1* to *C4* are smoothing this current and guarantee a short response time while relieving stress on the power supply. The status LED *D3* is connected between VCC and the trigger line of the MOSFET. As a result it will switch of whenever the MOSFET is being triggered. The voltage between VCC and signal becoming negative causes the LED to switch of. Due to its diode functionality there is almost no current flowing back to VCC. If There is no signal coming from the signal amplifier, the LED is on and nothing else will happen. The current drains over the pull down resistor *R2*. The voltage on the signal line will be round about 1.5V compared to ground. This way the MOSFET will not be triggered, as the trigger voltage is round about 3V between gate and drain.

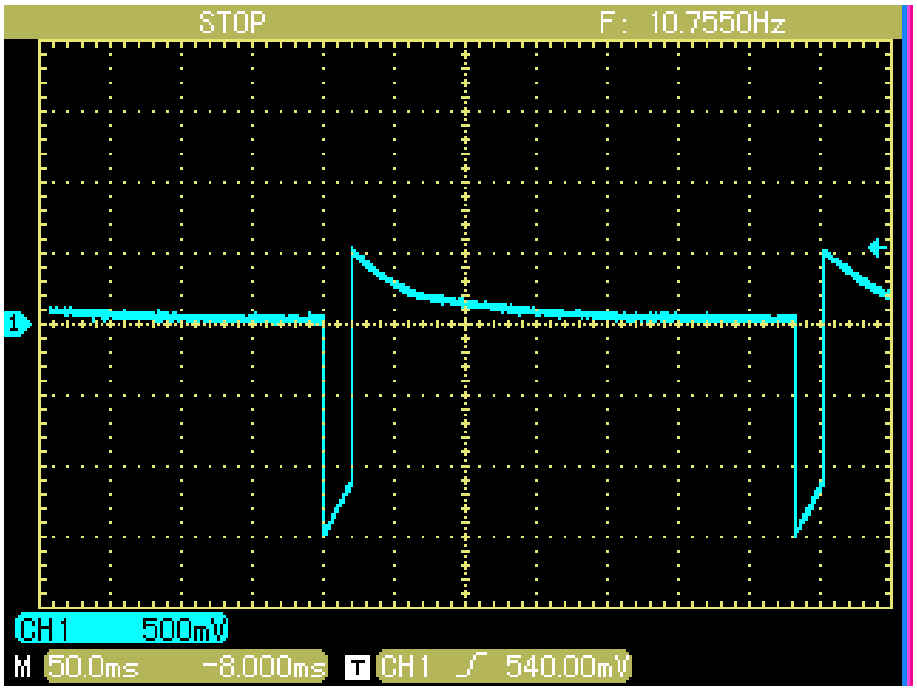

Figure 4.4.: Voltage between the LED anode and ground during flashing at 3 flashes per second. Measured using an oscilloscope. In this test circuit, only one LED was used and no signal amplifiers. The time delta per grid box is 50ms. The base voltage is 3V.

The power consumption of the boards is pulse wise in nature. Yet the power supply has to be able to serve every peak, not the average. In figure 4.4 the LED phases can be seen clearly. Switching on the LED causes the voltage to fall by 1.5V. The power supply slowly recovers from this surge. After switching of the LED the voltage goes up to 0.5V relative to 3V. This is because the capacitors are recharging. Eventually the capacitors are not fully sufficient to smooth the current. Therefore all the voltages on the power supply will drop shortly. When connecting other devices one has to be aware that the supplied voltage is not smooth and constant. A connected light bulb, for instance, would flicker on every flash of the LEDs.

The layout of the board is single-sided. Power lines have been designed as polygons where possible. The outer polygon is ground and the inner one is VCC. So in figure 4.5 the upper connector is ground, the middle one VCC and the bottom one the trigger. The drill holes do not come with any pads to save space, so using synthetic screws is highly advised. The Beesbook setup uses polypropylene screws. LED *D3* in this design is sketched to be on top of the board. However it is advised to solder it facing back to not obstruct the light emission of the IR LEDs.[8]

However since the IR light is now flashing at 3 Hz the PS-Eye cameras can not record

[8]Minor changes to the ordered version included. See CD for details.

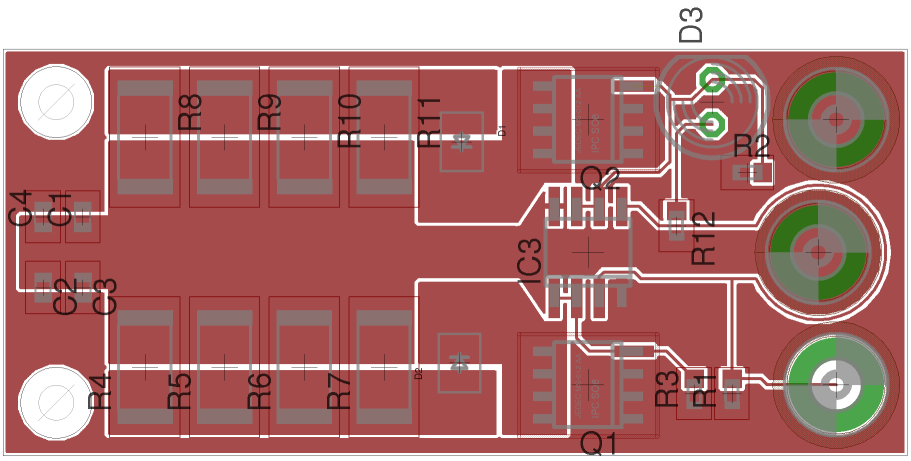

Figure 4.5.: Circuit board layout

under the IR light at 100 Hz any more. Some frames will be too dark as there is hardly light in between the flashes while some are overexposed, resulting in a flickering video. To resolve this issue the white light filters were installed again into the PS-Eye cameras. Panels of red LEDs are used to illuminate the hive constantly for the low resolution recordings. This way two distinct kinds of recordings can be done simultaneously using light which is non-visible to the honeybees.

Setup

This subsection describes how to wire the parts of the LED boards and cameras. All wiring is according to the Biorobotics Lab usecase.

An Arduino Duemilanove was used to create the trigger impulses for camera and flash boards. The signal for two flash boards was bundled into a single PWM port. As there are only two sides of the beehive only two different signals are required. Splitting it further into two PWM ports was done to increase the signal strength per trigger line and to increase flexibility of the setup. The cameras are also triggered in pairs of two. As the flashing is done asynchronously, two pairs of different flashing cycles have been connected to a single power supply. This way a single power supply only has to provide for two flash boards at a time.

Three wires are soldered to the LED boards as depicted in the top two images of figure 4.6, the smaller connector being the trigger line. Any small connector will fit this purpose, for example *JST-PH-2.0 2-Pin* connectors. Whenever a signal is applied to this line, the LEDs will turn on. The MOLEX connector Standard .093" matches the power supply of +5V as seen in common computers. Note that the power lines are inverted here and will not fit on the regular connectors of computer power supplies. This is done to prevent naive use, most of all connecting too many LED boards to a single power supply.

To connect the trigger lines of the boards to the Arduino, Cat-7 twisted pair wires have been used. One end was cut open and a 1-Pin connector plug was attached to each lead of the wire. On the other end, each twisted pair was bundled to a single one-pin male jumper wire. This way a single PWM port of the Arduino may trigger two LED boards simultaneously as described previously.

Cameras are triggered using a separate wire. Matching the camera, 8-pin connectors with part number *HR25-7TR-8SA* were connected to a Cat-7 twisted pair wire in a similar fashion as the trigger for the LED boards. To trigger the cameras only GPIO ports 1 (trigger in) and 5 (GND) are required.(34) These two were connected to a twisted pair each, the striped wire being ground. On the other end, all ground leads were bundled to a single one-pin male black jumper wire. The others were bundled in pairs of two.

Finally a simple bridge which short wires multiple one-pin connectors was created as in section C of figure 4.6. This is used as a compensating current bridge. This bridge is important: The voltage a power supply creates is relative to its ground potential. If the voltages among all connected devices are not relative to the same ground potential, undesired behaviour may occur. For example, the signal might be +5V relative to the Arduino's ground but only +2V relative to the power supplies ground. As a result, the LED will not switch on. In other cases too high voltage or power consumption may damage devices. Note that to create the same potential on all connected devices, a small current may be flowing from one connected devices ground to another. To minimize this, one can supply all devices from the same power outlet. It is a good idea to check the compensating flow using a multimeter. In the tested experiment setup, the compensating flow was hardly measurable using Lab devices.

A black one-pin male jumper wire has been soldered to the ground of each power supply to be connected to the compensating current bridge.

If the Arduino is running while switching on the power supplies, damage may be caused to them. There is round about a 6%[9] chance of the the switch-on time of the LEDs to stack with the inrush current of the capacitors. If the power supply is not powerful enough to serve this, the fuse will break.

4.2.2. Recording Hardware

The cameras being used are *FL3-U3-88S2C-C* by point grey. The maximum resolution is 4096 x 2160 at 21 fps but may be set to 4000x3000 at 15 FPS for free running trigger mode. Note that in external trigger mode only half the frame rate can be achieved in

[9]There are three flashes each lasting 20 ms in one second, resulting in a 6% chance

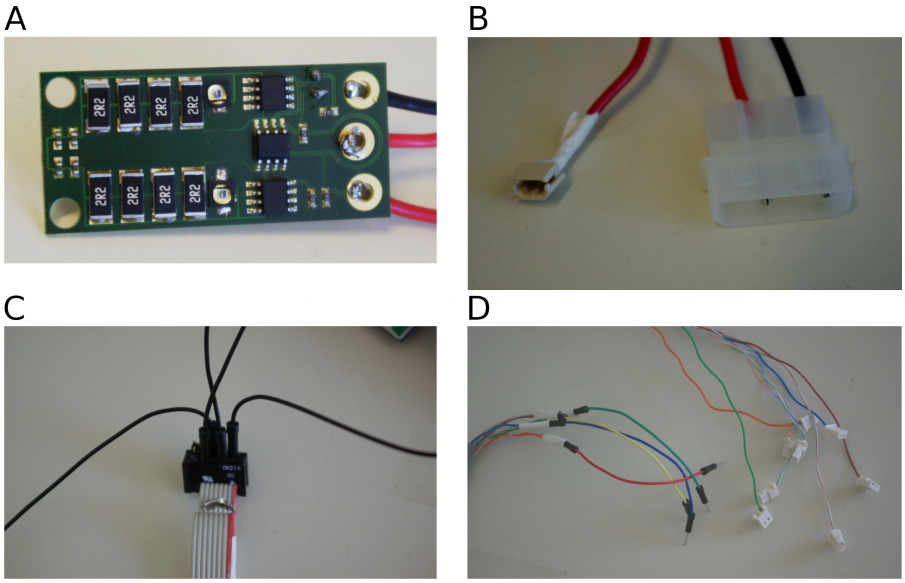

Figure 4.6.: LED boards and wiring. Top left: The LED board. Top right: MOLEX connectors of the LED board. Bottom left: Compensating current bridge. Bottom right: Trigger wire

general.(34) The cameras offer an USB3 interface for retrieving images and configuring the cameras via software. They also provide 8 GPIO ports connected via Hirose HR25 8 pin connector.[10] The GPIO ports are used in this scenario to do external triggering only, as described in the previous section.
In the advanced recording system two recording machines are being used, each coming with the following features.

- At least 16 GB RAM
- NVIDIA GTX960 GPU including 4 GB memory
- At least a dual core CPU running at 3 GHz
- Two USB3 interfaces
- Gigabit Ethernet adapter
- A dedicated HDD for the recording cache, having 3 TB disk space

The expected data throughput per camera can be estimated as follows. The throughput may vary according to video content, camera configuration and encoder configuration.
For HD videos at 1024 frames, 3 FPS, 4000x3000, CRF 20 or equivalent the expected file size is 400 MB to 900 MB.

$$Workload = (900MB * 3frames/s)/(1024frames) * 2 = 5.27MB/s \tag{4.12}$$

[10]Hirose HR25 8 pin connector part number HR25-7TR-8SA

where the *2 accounts to writing and reading each file.
For LD videos at 1024 frames, 3 FPS, 2000x1500, CRF 30 or equivalent the expected file size is 5 MB to 40 MB.

$$Workload = (40MB * 3frames/s)/(1024frames) * 2 = 0.24MB/s \tag{4.13}$$

So the total HDD workload per second is

$$2 * 5.27MB/s + 2 * 0.24MB/s \approx 11MB/s \tag{4.14}$$

In conclusion, with two cameras the worst-case throughput per HDD is 22 MB/s, which is feasible for consumer HDDs. To transfer this amount of data, Gigabit Ethernet may be necessary.

5. Evaluation

Both the evaluation and discussion sections come in two parts. This first part is evaluating the supercomputer assisted large scale video encoding and the second is the live video encoding.

5.1. Supercomputer Assisted Video Encoding

5.1.1. Filesize and NPL cost

The legacy data of the seasons 2014 and 2015 accounted for 132 TB and 157 TB respectively as mentioned in section 1.3.1. Encoding both seasons in HEVC format yielded 44.3 TB and 36.3 TB in the advanced system. Despite smaller input 2014 was encoded into larger files. The recording quality of 2014 has been worse than 2015 and due to time constraints exact analysis was only done for 2015. As the parameters were unsuitable for 2014 data, the quality settings have been set higher by estimating. The estimation included a certain buffer to be safe. In hindsight, 2014 video data has probably better quality than actually required.
The generated previews total to 389 GB for season 2014 and 483 GB for season 2016.
According to HLRN account information, the overall video encoding process used 21063 NPL, including test runs. Creating the LD videos consumed round about 2000 NPL. On MPP2 12 Cores (0.5 nodes) cost 1.2 NPL per hour. 21063 NPL correspond to 210630 core hours or ca. 24.04 core years which have been used for the HD videos. In practice the image analysis, i.e. detection of all tags of all images for an entire season, accounts to 120346 NPL, which is 1203460 core hours on MPP2 or 137.38 core years. In conclusion the NPL costs of video encoding are not negligible, but relatively small.

5.1.2. Verification

The verification methods discovered various issues and anomalies in the Beesbook legacy data and newly encoded data.

- **Error logging to textfiles** resulted in two erroneous videos total, both from season 2014. In both cases ffmpeg reported a broken frame which was skipped. These issues have been fixed manually by repeating the previous frame, replacing the broken one. Then the video was encoded again.

- **Timelines** have shown various gaps. The majority of them resulted from known issues during the start of the recordings. At that time reconfiguration and connectivity issues caused gaps frequently. For example in camera 0, season 2015 there have been 41 gaps found in the first two days and 15 more in the rest of the season. 3 gaps were reported maintenance. 5 gaps are due to broken or incomplete TAR files. Small missing sections of less than a 30 seconds caused 4 gaps. 3 more larger gaps have been found, one of which was caused by Cray failure. Detailed statistics can be found on CD.

- By **counting frames** some unusually short videos have been discovered. For instance, in season 2015 seven videos have been found. Six of them have been to short due to aforementioned gaps. Only one was too short for unknown reasons. This file has been re-encoded manually.

- Checking the videos **file size** being greater than zero did not result in any errors found.

- The **loss in quality** can not be quantified any better as has been done in chapter 3.5.1. The reason for this is, that previously analysed images can not be analysed again in their video format due to NPL cost. Analysing smaller samples has been done in section 3.5.1. Any upcoming image analysis will be done using a fully restructured pipeline and are hence not comparable.

5.2. Live encoding

5.2.1. Lighting

The camera trigger synchronization with the LED flash works seamless. The videos do look perfectly illuminated and no negative effects of the flashing can be seen. The histogram in 5.2 shows that all image information seem to be captured in the recent recordings.
Finding the exact difference in illumination is subtle, though. The vendor of the legacy LED lamps does not specify what kind of LEDs are embedded. The lamps have been taken apart, the LEDs tested thoroughly and compared to available specified ones. The LED *TSHG5210* by Vishay has been thought equivalent or better. In figure 5.1 the radiant flux of the *SFH 4716S* used in the LED boards is depicted. The LEDs of the advanced system run at 4 A for 20 ms, which should yield $3.5 \times \Phi_e$ (rounded values) The total radiant flux is specified as $\Phi_e = 1030$ mW whereas the Vishay LED emits $\Phi_e = 55$ mW. Scaling the HPFL (high power flashing LED) up to 4 A we get $\Phi_e = 1030\ mW \times 3.5 = 3605\ mW$. This makes a single HPFL emit 65.5 times more light than the regular ones by Vishay. As there are 24 LEDs per lamp, a HPFL may replace 2.73 lamps. There are 4 LED boards per side of the beehive, each having two HPFL. In theory these can replace $2.73 \times 8 = 21.8$ lamps. In the original setup 22 lamps have been used per side. Consequently the LED boards should be able to replace the lamps sufficiently. However the lighting system using

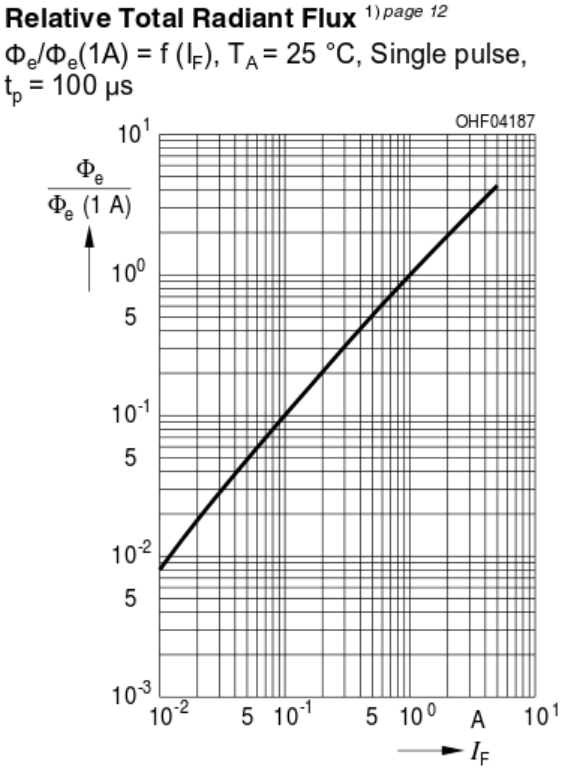

Figure 5.1.: Radiant flux of the *SFH 4716S* by Osram. (32)

the LED boards does not seem to yield as much light: Closing the aperture to f-stop 16 the resulting images are too dark, some portions being entirely black, so further investigations have been done.

In the legacy 2015 setup the aperture of the cameras was closed to f-stop 16, which is the minimum for this aperture. Due to the wave nature of light this causes a certain degree of diffraction. The shutter time was 8ms. Douvos provided a formula to calculate the optimal aperture, which is given in equation 5.1.(11)

$$N_{optimum} = \sqrt{375(\frac{u_n f}{u_n - f} - \frac{u_f f}{u_f - f})} \tag{5.1}$$

Where u_n is the nearest object, u_f the furthest object and f the focal length of the lens. Here the nearest object is 48cm, the furthest 51cm and the focal length is 12mm. Inserting the values in equation 5.1 we get approximately 0.083 or $\frac{1}{12}$ as an optimal f-stop, disregarding lens aberration. In conclusion the scenery was previously overly illuminated while it is now illuminated below par.

Nonetheless the advanced setup for 2016 was done the following way. The shutter time was set to 5ms, the gain to 17 and the f-stop to 8. This results in darker looking images, but the entire brightness spectrum is still being captured. In figure 5.2 the brightness

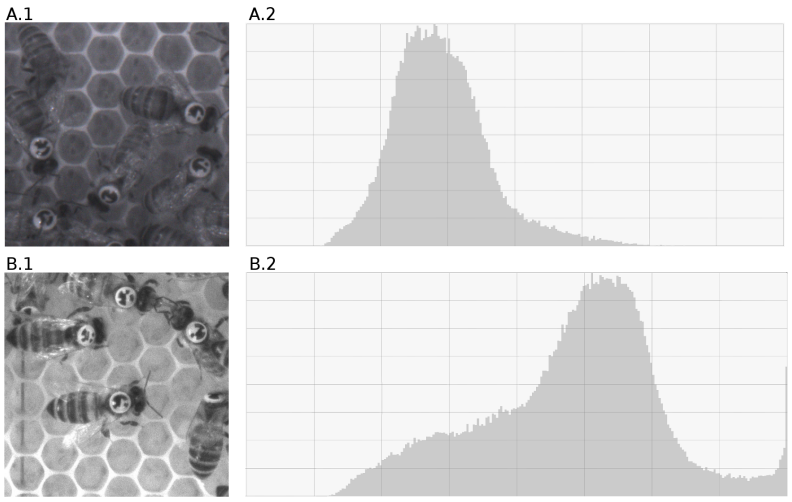

Figure 5.2.: Comparison of illumination. A.1 recording example of 2016. A.2 corresponding colour histogram to A.1, where left is darkest and right is brightest. B.1 recording example of 2015. B.2 corresponding colour histogram to B.1. In B.2 different classes can be seen. The gaussian peak represents the combs, whereas the level to the left represents the bees bodies. The slight peak to the very right represents overexposed image portions. A.2 shows that the comb and bee classes have merged apparently. The overly illuminated sections are missing and the gaussian moved slightly to the left.

histograms of 2015 recordings and recent ones are compared.

5.2.2. Image Statistics

Image statistics have been evaluated by plotting a sequence of results as a graph. A simple scenario has shown that the algorithms respond as expected and desired. Results are plotted in figure 5.3. DCT was excluded since it behaves very similar to SMD but is more computationally expensive. The contrast ratio was excluded due to very high standard deviation. It can be clearly seen that all metrics responded strongly when the aperture was opened and the image became nearly white. After adjusting gain and focus it stabilized at expected values. The image became more noisy and less sharp. The contrast did not exactly return to its previous value, in this case indicating the images are having a better contrast. However the variance increased although the image is in fact less sharp, which also contradicts the results of the SMD.

5.2.3. Stability

Prior to executing the bumblebee experiment, recording has been started for three weeks without experiment to test the advanced hard- and software. At first, the image statistics and image acquisition were run in a single process as different threads. This had the advantage that shared pointers could be used to free image memory by reference counting. In the first two weeks the acquisition crashed regularly every few days. Eventually a freeze occurred which could not be recovered by the watchdog script, as the application

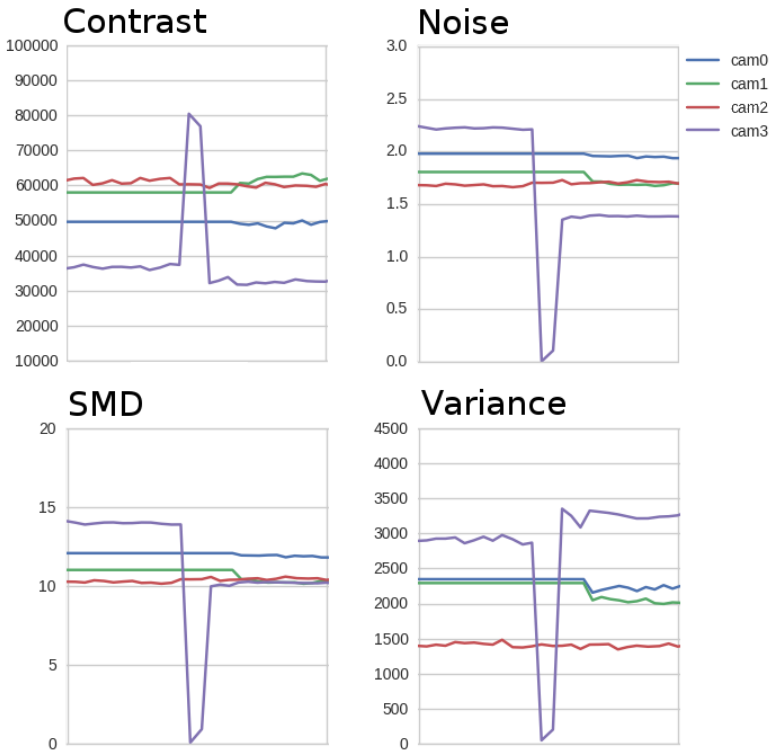

Figure 5.3.: Evaluation of Image Statistics. All cameras are recording the hive. Camera 0 to 2 are reference cameras. Their internal gain parameter was set to 17, the lens focused manually and the aperture closed to fstop 8, which seems to be a near-optimal setup. Camera 3 was selected for testing. The gain was set to 0, the lens was refocused and the aperture opened to fstop 2. Note that values may not be comparable between cameras since different portions of the hive have been recorded. Plots created by Benjamin Wild.

never terminated. Due to this issues further improvements have been done: To debug any upcoming errors an event handler was integrated. This event handler will catch every segmentation fault and print the stack trace to the console. For keeping track of any relevant informations, the acquisition process is now being run in a GNU screen session. The sessions output is being dumped to log files. The image statistics functions are likely to be changed in the future and are not crucial to the acquisition process. For this reason the application has been modified to run statistics in a separate process with a separate watchdog. The images are acquired using the shared memory interface. Furthermore within the acquisition process a kind of watchdog thread was created. This thread watches all buffers in existence. If a buffer has seen a change once it is considered active. If an active buffer is not affected by any change within 60 seconds, the entire process is killed and restarted. Which buffer has been inactive buffer gets logged to console.
Additionally a script was created to post on the Biorobotics Lab slack channel whenever the software crashed. Along provided is a short description of the error, e.g. a segmentation fault, SDK related error or camera connectivity issues. The error level indicates whether it is self-recoverable or not.

After these changes crashes rarely occurred. The remaining crashes originated from an SDK function. The SDK errors are catchable and can be resolved by restarting the application. Recovering from these errors may be possible but was not implemented, as the error is not easily reproducible. Statistically this kind of error occurred twice a week for which reason the loss was thought to be negligible. When the error gets caught the restart is done quickly. Possible gaps in the recorded data amount to as much as one minute. The reason may be the watchdog threads timeout of one minute for a thread to be considered dead.

6. Discussion

6.1. Supercomputer Assisted Video Encoding

In the course of this thesis all of the existing legacy data has been encoded as videos successfully. LD videos have also been created. As the evaluation has shown, the cost of encoding was with approximately 23000 NPL reasonably low. The quality of the videos was evaluated thoroughly and were found to be suitable for the previously mentioned tracking algorithms. The file size went down to 0.336 times and 0.282 times the original size, which are around 40 TB of data. Moreover it was verified that the encoding process did not conclude in any quantitative data loss.

Work package 1 was completed successfully, which was reducing the overall file size by encoding legacy JPEG image data as HEVC videos. This was done without loosing data in quantity and quality. The amounts of data are reasonably large and might not yet be stored locally. This is because the requirements in video quality and resolution are high. Also an extra buffer was used in terms of quality, as the data is thought to be crucial for the project. This might improve in the future due to changes mentioned in section 7.1 but is not within the realm of possibility for now. The thesis has elaborated different compression schemes and found the presented compression the best semi-of-the-shelf solution. Improving compression even more might be possible using, for example, custom motion estimation algorithms in HEVC or discarding data in quality or quantity. The former seems possible, but is beyond the scope of this thesis. Later idea has been discarded as explained previously.

6.2. Live Video Encoding

The aim of live encoding was to improve recording quality while compressing data in real time, which is work package 2.

Work package 2.1 is advancing the recording software to generate HEVC videos on GPU. The live encoding is done according to the quality properties determined in work package 1. Application stability has been proven and data transfer is even more reliable than before. This is because the videos are much smaller in size, so a cache of the same size may be sufficient for a much longer period of time. Furthermore the acquisition application has

been improved to offer a simple interface to interact live with the recording data. This makes later additions to the acquisition easier and more reliable without elaborate tests. Live statistics have been done and found to be meaningful. Semi-real-time statistics are able to reveal changes in the setup and can indicate improvements or deteriorations in image quality.

The advanced lighting system developed according to work package 2.2 is not only cheaper, but also more space efficient and yields better quality in the recordings as there is no back light. However the accounted amount of lamps is not sufficient and building additional lights will be appropriate.

7. Concluding Remarks

7.1. Supercomputer Assisted Video Encoding

The task of encoding existing data was meant to be done only once. However there are uses for the modified observer script and verification scripts.

Quality Adaptation for a New Pipeline

In section 3.5 the quality evaluations have been done. The desired quality was based on how well the pipeline performs. Recently the Biorobotics Lab is working on a new pipeline which is entirely different in nature. The new pipeline uses deep convolutional neural networks (CNN) to localize and decode tags in two steps. It is a common pattern to sample the input down when using CNNs, since it is reducing complexity. For localizing, first tests have show that sampling down the image works very well. It is yet unknown how well the decoder will perform on the data. It might be the case that the CNN approach is able to perform well on less high resolution input. Though unlikely, the contrary might be the case as well. To evaluate this is definitely a task when the upcoming pipeline comes to use. The methods introduced in section 3.5 will not work for evaluating the quality just like that. The general approach however seems adaptable.

Observer Scripts

In his thesis Simon Wichmann pointed out, that extending the observer might be a future task. In fact one of the changes he named as future work have been done in this thesis. This is, establishing sequentiality in the images provided by the observer. In his thesis, he stated this might be useful for the pipeline and tracking tasks. This is still valid. A foundation for tasks like this has been made. The only job left to do is to enable the observer to extract videos instead of TAR archives. A video and frame list file can be treated as a TAR file equivalent.
Alternatively the video and file list tuple could be supplied to the pipeline tasks as a whole.

7.2. Live Video Encoding

There are some aspects of the live encoding which may be extended or improved in the future.

Additions to Calibration

The image statistics are currently only used to ensure constant quality during the recordings and to assist during calibration. The calibration process could be improved to not only assist, but partially takes over the calibration task. The camera lenses and aperture can only be manually adjusted. But parameters of the camera software can be automatically adjusted via software. So a possible extension to the software is implementing focusing and configuration adjustment algorithms similar to point-and-shoot cameras.
Using a different kind of lens or extending the available ones it may be possible to also adjust the aperture and focus via software.

Live- and Near-Live Streaming

The shared memory currently is a simple interface to grab the most recent image without obstructing the recordings in any way. This may be used to stream the images as a video live to other locations, not only to a local viewer. One could inspect or maintain the recordings live over the internet or do a public live stream. For instance this could be coupled with the pipeline to show entire tracks or partial tracking results live.

Exposing Buffers for Live Processing

In the same fashion as the most recent image is being exposed using shared memory, entire buffers could be exposed. This makes doing the localization and decoding live more feasible. The decoding task is very time extensive, but the only task which requires very high resolution data. If this can be done live, HD videos might not be necessary any more.

Appendices

A. Calculations and Tables

A.1. HEVC And AVC Compression Ratio For Beesbook Recordings

Rerabek et al have shown that HEVC compresses up to 50% better than AVC.(36) As the Beesbook recordings are of special nature, this ratio was verified sufficiently. Figure A.1 shows that HEVC in fact is performing significantly better than AVC, but does not reach 50%.

Threads	preset fast	preset superfast	preset ultrafast	codec
4	19.1s – 65.4MB	04.4s – 71.3MB	01.6s – 88.5MB	x264
4	36.4s – 42.5MB	31.0s – 50.8MB	16.8s – 50.1MB	x265
8	16.6s – 65.4MB	03.5s – 71.3MB	01.5s – 88.5MB	x264
8	34.9s – 42.5MB	30.1s – 50.8MB	16,4s – 50.1MB	x265
16	13.0s – 65.4MB	03.6s – 71.3MB	01.6s – 88.5MB	x264
16	34.9s – 42.5MB	30.3s – 50.8MB	16.5s – 50.1MB	x265

Figure A.1.: Comparison of encoding performance of x265 and x264 for Beesbook recordings. The sample images were a sequence of 30 images recorded from the beehive in raw format. CRF parameter was set to 15. The first column indicates the value of the thread parameter being used, the last one the codec. The fields show the respective performance according to the preset parameter in the headline. Although no real quality measure was taken apart from the CRF value, it can be seen that x265 performs better in terms of filesize. All encoding times are from a single test PC and are hence only comparable among each other. Note that the fastest possible encoding speed is 1.5s. Using a better compressing preset the fastest one is 3.5s. The file size savings range from 0.57 times the AVC encoded version to 0.71 times.

A.2. Lighting System Cost Calculation

In figures A.2 and A.3 the prices of the legacy and advanced lighting system have been calculated. The advanced system is 347.12€ or 0.268 times the price of the previous one.

A.3. Disk Space Cost Calculation

The Biorobotic Lab strives to save the recorded videos locally in the future. For this reason a cost estimation for hard disk space has been done. Currently the archive HDD

Product	Amount	Vendor	Cost
PCB board, 23x51, 8 days delivery	8	pcb-pool.com	62.77
MOSFET *AO4752*	16	digikey.com	4.53
LED OSRAM *SFH-4716S*	10	digikey.com	45.68
Signal Amplifier *LM358DT*	10	digikey.com	2.83
2.2Ohm Resistor *CRM2512-JW-2R2ELF*	64	digikey.com	10.30
10kOhm Resistor *RC0603JR-0710KL*	25	digikey.com	0.20
120Ohm Resistor *RC0603JR-07120RL*	10	digikey.com	0.11
2.2mF Capacitor *JMK107BJ225KA-T*	50	digikey.com	4.15
Hirose plug *HR25-7TP-8P*	4	digikey.com	89.84
Subtotal			220.41
Power supply, 500W, *LC500H-12*	2	conrad.de	79.98
Wire, black 10m, *SH1927*	1	conrad.de	5.95
Wire, red 10m, *SH1926*	1	conrad.de	5.95
Arduino Duemilanove	1	amazon.de	17.79
Losi Connector Set: Micro-T/B/DT	1	amazon.de	5.61
Cat 7 Wire, 2m	2	amazon.de	9.90
Male to Male Jumper Wires, Mixed Set	1	amazon.de	1.53
Total			**347.12**

Figure A.2.: Price calculation of the newly designed flash boards of the advanced system including equipment. All prices in Euro. The subtotal is the equipment which had top be bought by Biorobotics Lab. The second part was on-stock and did not have to be bought. All prices retrieved 21.06.2016.

Product	Amount	Vendor	Cost
IR Light ABUS TV6700	24	conrad.de	971.76
VOLTCRAFT USPS-600 Power Supply	24	conrad.de	323.76
Total			**1295.52**

Figure A.3.: Price calculation of the legacy lighting system. All prices in Euro. Instead of using 24 small power supplies, a single larger one may be used.

'ST8000AS0002' by Seagate comes at a good price performance of 248€[1] per 8 TB and is of feasible quality for this task. As high reliability is desired, a RAID 5 system on three disks is considered. Slightly rounded amounts of required disk space are used: 200 TB for image data and 40 TB for video data. Costs of management devices are not being taken into account.

$$HDD\ cost\ images = 160TB/8TB * 248€ * 1.5 = 7440€ \tag{A.1}$$

$$HDD\ cost\ videos = 40TB/8TB * 248€ * 1.5 = 1860€ \tag{A.2}$$

It can be seen in equation A.1 that storing the data as HEVC video files saves about 5580€ on hard drives per season.

[1] Price per amazon, retrieved 21.06.2016: `https://www.amazon.de/Seagate-Archive-interne-Festplatte-Cache/dp/B00QGFEQXU/ref=sr_1_sc_1?ie=UTF8&qid=1466504694&sr=8-1-spell&keywords=seagote+8TB+storage`

B. Glossary

AVC Advanced Video Coding. A coding standard. Also H.264/MPEG-4 AVC

CLI Command Line Interface.

CPU Central Processing Unit.

CRF Constant Rate Factor.

GND Chassis Ground (electronics).

GPU Graphics Processing Unit.

HD High Definition. Here: Videos of size 4000x3000 and high quality.

HEVC High Efficiency Video Coding. Also H.265 or MPEG-H Part 2

HLRN Norddeutscher Verbund zur Förderung des Hoch- und Höchstleistungsrechnens.

IO Input/Output

IR Infra red.

JAABA Janelia Automatic Animal Behaviour Annotator

JPEG Joint Photographic Experts Group.

JSON JavaScript Object Notation.

LD Low Definition. Here: Videos of size 2000x1500 and low quality.

LED Light-Emitting Diode.

MKV Matroska Video. A video container format.

MOLEX Molex Incorporated. A manufacturer of electronic systems.

MPEG4 Moving Picture Experts Group 4. A video coding standard.

NAS Network Attached Storage.

NPL Norddeutsche Parallelrechner-Leistungseinheit.

OS Operating System.

PSNR Perceived Signal to Noise Ratio.

PWM Pulse Width Modulation.

QT Derived from *cute*. A programming library.

RAID Redundant Array of Independent Disks.

SCP Secure Copy.

TAR Tape Archiver. Here: The format for storing archives.

Bibliography

[1] Anne Aaron. High Quality Video Encoding at Scale, 2015. URL http://techblog.netflix.com/2015/12/high-quality-video-encoding-at-scale.html. [Online, Accessed 2016-03-15].

[2] Wikipedia Authors. Comparison of video container formats, 2016. URL https://en.wikipedia.org/wiki/Comparison_of_video_container_formats. [Online, Accessed 2016-05-26].

[3] Wolfgang Baumann. HLRN Homepage, 2016. URL https://www.hlrn.de/. [Online, Accessed 2016-08-05].

[4] Fabrice Bellard. BPG Image format, 2015. URL http://bellard.org/bpg/. [Online, Accessed 2016-08-05].

[5] Frank Bossen, David Flynn, Karl Sharman, and Karsten Sühring. JCTVC-Software Manual. 2016. URL https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/trunk/doc/software-manual.pdf. [Online, Accessed 2016-03-22].

[6] Patrick E. Clark, Douglas E. Johnson, Mark A. Kniep, Phillip Jermann, Brad Huttash, Andrew Wood, Michael Johnson, Craig McGillivan, and Kevin Titus. An Advanced, Low-Cost, GPS-Based Animal Tracking System. *Rangeland Ecology & Management*, 59(3):334–340, may 2006. doi: 10.2111/05-162R.1.

[7] Security Center of Group ABUS Corporation. IR-Scheinwerfer. pages 1–60. URL https://firstmall.de/etc/pdf/TV6700_INT.pdf. [Online, Accessed 2016-08-05].

[8] X264 Developers. X264 Changelog, 2016. URL http://x264.nl/x264/changelog.txt. [Online, Accessed 2016-05-26].

[9] Ogle Player development Team. MPEG headers Quick Reference, 2004. URL http://dvd.sourceforge.net/dvdinfo/mpeghdrs.html. [Online, Accessed 2016-03-16].

[10] DivXLabs. DivX HEVC Community Encoder, 2014. URL http://labs.divx.com/divx265. [Online, Accessed 2016-08-05].

[11] George Douvos. OptimumCS-Pro Optical Science. URL www.georgedouvos.com/douvos/OptimumCS-Pro_Optical_Science.html. [Online, Accessed 2016-07-12].

[12] Mohd Awais Farooque and Jayant S Rohankar. Survey on Various Noises and Techniques for Denoising the Color Image. *International Journal of Application or Innovation in Engineering and Management*, 2(11):217–221, 2013.

[13] Mark Fiala. ARTag Rev2 Fiducial Marker System : Vision-based Tracking for AR. *Library*, 2005. URL `http://campar.in.tum.de/twiki/pub/ISMAR/IarAbstractARTag/IarDetailsFialaSlides.pdf`. [Online, Accessed 2016-08-05].

[14] Andrew Ford. x265 Command line options, 2014. URL `http://x265.readthedocs.io/en/default/cli.html#input-output-file-options`. [Online, Accessed 2016-05-26].

[15] John L. Furlani and Peter W. Osel. Modules – Software Environment Management, 2012. URL `http://modules.sourceforge.net/`. [Online, Accessed 2016-08-05].

[16] Google. Timelines API, 2016. URL `https://developers.google.com/chart/interactive/docs/gallery/timeline`. [Online, Accessed 2016-05-26].

[17] Sebastian Gross, Martin Schink, Thomas Stehle, Alexander Behrens, Jens Tischendorf, Christian Trautwein, and Til Aach. Echtzeitfähige extraktion scharfer standbilder in der video-koloskopie. In *Bildverarbeitung für die Medizin (BVM)*, page accepted, March 14–16 2010.

[18] Bernd Heinrich. *The Mechanisms and Energetics of Honeybee Swarm Temperature Regulation*, volume 91. 1981. doi: 10.1126/science.212.4494.565.

[19] Heinz Hofbauer, Christian Rathgeb, Johannes Wagner, Andreas Uhl, and Christoph Busch. Investigation of better portable graphics compression for iris biometric recognition. In *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, volume P-245, 2015. doi: 10.1109/BIOSIG.2015.7314609.

[20] MulticoreWare Inc. Preset Options — x265 documentation, 2014. URL `http://x265.readthedocs.org/`. [Online, Accessed 2016-08-05].

[21] MathWorks Incorporation. MATLAB VideoReader - MathWorks Deutschland, 2016. URL `http://de.mathworks.com/help/matlab/ref/videoreader.html`. [Online, Accessed 2016-05-26].

[22] Intel Corporation. Ultra HD Experiences Made Ultra Simple. (Vdi), 2016. URL `http://www.sigmasd.com/partner/wp-content/uploads/2015/09/Intel-Media-Server-Studio-2016.pdf`. [Online, Accessed 2016-08-05].

[23] Nanxiang Jin, Tim Landgraf, Simon Klein, and Randolf Menzel. Walking bumblebees memorize panorama and local cues in a laboratory test of navigation. *Animal Behaviour*, 97(November):13–23, 2014. doi: 10.1016/j.anbehav.2014.08.013.

[24] Brother John. Containerformate, 2012. URL http://encodingwissen.de/formatedschungel/container#fn6. [Online, Accessed 2016-03-16].

[25] Gnu.org Maintainers. Basic Tar Format, 1990. URL http://www.gnu.org/software/tar/manual/html_node/Standard.html. [Online, Accessed 2016-05-26].

[26] Randolf Menzel and M Blakers. Comparative Colour Receptors in the Bee Eye-Morphology and Spectral Sensitivity. *Journal of Comparative Physiology A*, 108(1): 11–33, 1976. doi: 10.1007/BF00625437.

[27] Danielle P Mersch, Alessandro Crespi, and Laurent Keller. Tracking individuals shows spatial fidelity is a key regulator of ant social organization. *Science Express*, 340(6136):1090–3, 2013. doi: 10.1126/science.1234316.

[28] N. Ng Kuang Chern, Poo Aun Neow, and M.H. Ang. Practical issues in pixel-based autofocusing for machine vision. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, volume 3, pages 2791–2796. IEEE, 2001. doi: 10.1109/ROBOT.2001.933045.

[29] NVidia Corporation. Nvidia Video Codec Sdk, 2013. URL https://developer.nvidia.com/nvidia-video-codec-sdk. [Online, Accessed 2016-08-05].

[30] NVidia Corporation. Nvidia video encoder 5.0. (November):1–13, 2014. URL http://developer.download.nvidia.com/compute/nvenc/v5.0_beta/NVENC_DA-06209-001_v06.pdf. [Online, Accessed 2016-08-05].

[31] NVidia Corporation. NVENC - NVIDIA Video Encoder API Reference Manual version 6.0. 2015.

[32] Osram. OSLON Black Series (850 nm) - Datasheet. 2012. URL http://www.osram-os.com/Graphics/XPic7/00204371_0.pdf/SFH%204716S.pdf. [Online, Accessed 2016-08-05].

[33] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi. Video coding with H.264/AVC: tools, performance, and complexity. *IEEE Circuits and Systems Magazine*, 4(1):7–28, jan 2004. doi: 10.1109/MCAS.2004.1286980.

[34] PointGrey. Flea3 USB 3.0 Digital Camera Technical Reference version 7.0. 2014. URL http://www.ptgrey.com/support/downloads/10120/. [Online, Accessed 2016-02-23].

[35] ITU-T Recommendations. ITU-T Recommendation database, 2003. URL http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=2820. [Online, Accessed 2016-04-29].

[36] Martin Rerabek, Philippe Hanhart, Pavel Korshunov, and Touradj Ebrahimi. Quality evaluation of HEVC and VP9 video compression in real-time applications. In *2015 Seventh International Workshop on Quality of Multimedia Experience (QoMEX)*, volume 2013, pages 1–6. IEEE, may 2015. doi: 10.1109/QoMEX.2015.7148088.

[37] Thomas D Seeley. The wisdom of the hive: the social physiology of honey bee colonies, 1997. ISSN 00280836.

[38] Peter Skorupski and Lars Chittka. Photoreceptor spectral sensitivity in the bumblebee, Bombus impatiens (Hymenoptera: Apidae). *PLoS ONE*, 5(8), 2010. doi: 10.1371/journal.pone.0012049.

[39] Jean-luc Starck, Fionn Murtagh, Emmanuel J Candès, and David L Donoho. Gray and Color Image Contrast Enhancement by the Curvelet Transform. *IEEE Transactions on Image Processing*, 12(6):706–717. doi: 10.1109/TIP.2003.813140.

[40] Gary Sullivan and Stephen Estrop. Recommended 8-Bit YUV Formats for Video Rendering (Windows), 2008. URL `http://msdn.microsoft.com/de-de/library/dd206750(v=vs.85).aspx`. [Online, Accessed 2016-08-05].

[41] Christian Tietz. Entwurf und Implementierung einer Speicherarchitektur für Bildmassendaten zur Verhaltensanalyse von Honigbienenkolonien. 2015.

[42] VideoLAN. Multicoreware x265 wiki. URL `https://bitbucket.org/multicoreware/x265/overview`. [Online, Accessed 2016-08-05].

[43] Marko Viitanen, Koivula Ari, and Lemmetti Ari. Ultra Video Joiner, 2010. URL `http://ultravideo.cs.tut.fi/#encoder`. [Online, Accessed 2016-08-05].

[44] Fernando Wario, Benjamin Wild, Margaret J. Couvillon, Raúl Rojas, and Tim Landgraf. Automatic methods for long-term tracking and the detection and decoding of communication dances in honeybees. *Frontiers in Ecology and Evolution*, 3 (September):1–14, 2015. doi: 10.3389/fevo.2015.00103.

[45] Simon Wichmann. Storage and Parallel Processing of Image Data Gathered for the Analysis of Social Structures in a Bee Colony, 2014.

[46] Thomas Wiegand, Heiko Schwarz, Anthony Joch, Faouzi Kossentini, and Gary J. Sullivan. Rate-constrained coder control and comparison of video coding standards. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):688–703, 2003. doi: 10.1109/TCSVT.2003.815168.

[47] Jing Xiao, Liang Liao, Jinhui Hu, Yu Chen, and Ruimin Hu. Exploiting global redundancy in big surveillance video data for efficient coding. *Cluster Computing*, pages 531–540, 2015. doi: 10.1007/s10586-015-0434-z.

[48] Mareike Ziese. Scene Labeling eines Bienenstocks. 2016.