Diplomarbeit

BeeTube: A Management System for Digital Videos and the Visualization of Honeybee Trajectories

Verfasser:
Thomas Schwaier

Betreuer:
Prof. Dr. Raúl Rojas

March 16, 2010

Freie Universität Berlin
Institut für Informatik

Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, March 16, 2010_____

Acknowledgments

I would like to thank my family, here and abroad, for their continued love and support for I am truly blessed to have them in my life. I would especially like to thank my siblings - I wouldn't know what to do without them. Praise also is due to my closest friends, who encouraged and motivated me throughout my studies and hardships with their understanding, advice and the occasional kick in the right spot.

As this unexpectedly long journey comes to a close, the promise of greener pastures looms on the horizon.

I can't wait to get there!

Abstract

Bees communicate certain information by performing dances. However, researchers still do not know, which of the many signals, that are exhibited during the dance, are relevant for transmitting this information. Biologist track the dance patterns on recorded videos with the help of computer based analysis. The use of these methods has created a new predicament for researchers. The amount of recorded videos, tracked dances, and digital data has long surpassed the limit for manual methods of organization.

This thesis is about the design and development of a web-based media content management system, called BeeTube, to alleviate this situation. The system is intended to assist scientist in handling the vast amounts of data and provide them with a tool for systematic research and analysis. It also implements unprecedented features for the visualization and evaluation of the extracted dances, thereby allowing researchers to examine the existing data in new and improved ways.

List of Figures

2.1 **Round Dance** . . . 5
2.2 **Waggle Dance** . . . 6
2.3 **Waggle Dance Orientation** . . . 7
2.4 **Bee Robot 2.0** . . . 8
2.5 **Infrared Lighting** . . . 9
2.6 **Tracking Software** . . . 10
2.7 **Current Data Management Workflow** . . . 12
2.8 **Workflow Comparison** . . . 13

3.1 **Grails Technology Stack** . . . 23
3.2 **General Architecture BeeTube** . . . 25
3.3 **BeeTube Database Schema** . . . 27
3.4 **Screenshot Experiment Create View** . . . 28
3.5 **Screenshot Experiment Show View** . . . 29
3.6 **Screenshot Experiment List View** . . . 30
3.7 **Screenshots Video Views** . . . 31
3.8 **Screenshot Video List View** . . . 32
3.9 **Screenshot Video Show View** . . . 33
3.10 **Screenshots Trajectory Views** . . . 34
3.11 **Screenshot Search View** . . . 35

4.1 **Crashed Tracking Software** . . . 37
4.2 **SMIL Video Overlay Animation Example** . . . 39
4.3 **Flash/Flex Examples** . . . 40
4.4 **Silverlight Examples** . . . 41
4.5 **Java Applet Example** . . . 42
4.6 **JavaFX Examples** . . . 43
4.7 **Unsynchronized Visualization Example** . . . 47
4.8 **Semi Synchronized Keyframe Substition** . . . 48
4.9 **BeeTube Player UML Class Diagram** . . . 50
4.10 **Visualization Single Trajectory** . . . 52
4.11 **Visualization Multiple Trajectories** . . . 52

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
  - 1.1.1 Robobee . . . 1
- 1.2 Objective . . . 2
- 1.3 Thesis Structure . . . 3

2 Fundamentals **4**
- 2.1 Honeybee Research . . . 4
- 2.2 Bee Dance Language . . . 4
- 2.3 Bee Dance . . . 5
  - 2.3.1 The Round Dance . . . 5
  - 2.3.2 The Waggle Dance . . . 6
- 2.4 Project RoboBee . . . 8
  - 2.4.1 Video Tracking Software . . . 9
    - 2.4.1.1 Data Management Issues . . . 11
    - 2.4.1.2 Management Solution . . . 11
- 2.5 Information Management . . . 12
  - 2.5.1 Information Systems . . . 13
  - 2.5.2 Data and Information . . . 14
- 2.6 Content Management . . . 14
  - 2.6.1 Content . . . 14
  - 2.6.2 Management . . . 15
- 2.7 Content Management Systems . . . 15
  - 2.7.1 Web Content Management Systems . . . 16
  - 2.7.2 Digital Asset Management Systems . . . 16
- 2.8 BeeTube . . . 17

3 BeeTube **19**
- 3.1 Requirements Specification . . . 19
- 3.2 Technology Discussion . . . 19
  - 3.2.1 Customization vs Creation . . . 20
  - 3.2.2 Web Development Technology Frameworks . . . 21
  - 3.2.3 Grails . . . 22
- 3.3 Implementation . . . 24
  - 3.3.1 General Architecture . . . 24
  - 3.3.2 Controller . . . 24

3.3.3 Model . . . 26
3.3.4 View . . . 26

4 Visualization **36**
4.1 Video Format Fundamentals . . . 36
4.2 Visualization Technologies . . . 37
4.2.1 Requirements . . . 37
4.2.2 Technology Platforms . . . 38
4.2.3 SMIL . . . 38
4.3 Plug-In Technologies . . . 39
4.3.1 Flash/Flex . . . 40
4.3.2 Silverlight . . . 41
4.3.3 Java Applets/JavaFX . . . 42
4.3.4 Technology Evaluation . . . 43
4.4 Video Encoding . . . 44
4.4.1 Video Format Problem . . . 45
4.4.2 Unsynchronized Visualization . . . 46
4.4.3 Keyframe Substitution . . . 46
4.4.4 Semi Synchronized Keyframe Substition . . . 48
4.5 Implementation . . . 49
4.5.1 Structure . . . 49
4.5.2 Movie Controller . . . 49
4.5.3 Display Controller . . . 51
4.5.4 Graphics Controller . . . 51
4.6 BeeTube Player . . . 51

5 Summary **53**
5.1 Results . . . 53
5.2 Outlook . . . 53

1 Introduction

The first chapter gives an introduction to the thesis and is supposed to provide a first glance at the work as a whole. This includes the thesis' motivation, the objective, structure, and its place within the overall effort of the *Robobee* Project.

1.1 Motivation

Insects, and especially bees, play a vital role in our ecosystem as pollinators for plants. Honeybees (*Apis Mellifera*) are the most prominent pollinators, since, for several thousand years, human civilizations have literally been depending on the fruits of their labor. Today, domesticated and feral honeybees are still an integral part of the world's food production, whose pollinated crop total an annual global worth of US$ 217 billion. [1] Yet, the underlying mechanisms of how exactly bees encode and transmit information continue to elude scientists.

An important key to significantly advance our understanding of bees and how they communicate, is to understand the meaning of the signals that are used to communicate during the bee dance. According to Karl von Frisch's Nobel Prize awarded theory, published in 1946, bees communicate abstract information via their dance, thus dubbing it the "bee dance language". They encode the location of a food source, such as distance, direction and angle from the sun, by varying their dance pattern correspondingly in duration, speed, and orientation. [2] This idea was first challenged in 1967, by Adrian Wenner, who concluded, that bees, like all other insects, communicated using odors. The ensuing debate has remained unresolved for the past 40 years despite the fact that scientists have learned a great deal about honeybees during that period. [3] However, the question of how exactly information is encoded in the different signals, and which significance each of them have for transmitting this information during the dance, still remains unanswered.

1.1.1 Robobee

The Artificial Intelligence study group at the Department of Computer Science under Professor Raúl Rojas[1] and the Neurobiology study group under Professor Randolf Menzel[2] joined efforts in 2005 to form the *Robobee* Project.[3] The intention was to investigate the bee dance's unresolved questions. The Robobee project encompasses several projects

[1] http://www.inf.fu-berlin.de/groups/ag-ki/index.html
[2] http://www.neurobiologie.fu-berlin.de/menzel/menzel.html
[3] http://www.robobee.eu

to achieve this. The central effort is focused on the design and development of an artificial biomimetic bee robot. The robot is designed to be able to communicate with other bees by mimicking their dance. It can simulate the signals that are involved during communication and is thereby supposed to help determine their relevance. If successful, it would greatly help to understand the mechanism of their communication.

A necessary prerequisite for creating artificial bee dances is to first extract the key features of bee dances. The Artificial Intelligence study group introduced computer based video analysis to support the systematic study of the bee dances in 2007. These were originally recorded using a high speed camera. A tracking software was developed which allowed researchers to track the bees and their exact dance trajectories. Using this system, the team was able to extract and analyze important features of the bee dance. This helped tremendously in creating artificial bee dances for the biomimetic bee robot, but also allowed to gain new insights into the nature of the dance. [4]

Yet the wealth of new data that has been gathered by the experimenters, presents in itself a new predicament. The steady increase of video data is approaching the point where it is becoming unmanageable. The volume of high speed videos which have been recorded over the past three years is about to reach 300 Gigabytes and keeps growing continuously. Sighting, ordering, and analyzing the abundance of recorded information as well as tracking the bee dances by software has become a task of its own. [5]

In order to watch a once captured trajectory and video file, a user has to play them back simultaneously, using the aforementioned tracking software. Unfortunately, the tracking software's user interface has not been designed with laymen in mind. The software is also unable to display multiple trajectories within a video and is unstable, which, at times, causes it to crash during playback.

The analysis of the gathered material is further impeded by the fact that information concerning the videos, i.e. experiment meta data and trajectories, are scattered over several computer systems, files, and formats. This makes it tedious and time consuming for scientists to keep track of the information of a particular video and distracts them from the actual task at hand.

1.2 Objective

The objective of this thesis is twofold. First, a web-based media content management system (MCMS), called *BeeTube*, is proposed and implemented to efficiently store, process, and analyze the videos with their extracted bee trajectories. The system is to be used to contextualize videos with the experiment meta data via a user friendly interface. Second, a visualization plug-in is designed to display the recorded videos and their corresponding trajectories simultaneously. The plug-in will be realized as an integrated part of the media content management system. Together, the MCMS BeeTube is supposed to support scientists in organizing the collected experiment data by unifying the access to it. By alleviating the problem of scattered resources researchers will then be free to fully concentrate their time and energy on the systematic analysis of the accumulated material. All data will be accessible through a central repository that allows uniform

management and controlled access.

The system's target audience is mainly comprised of biologists, who on average tend to be computer laymen, so an intuitive and easily accessible user interface is required. Furthermore, it is also supposed to be highly available - meaning at any time and anywhere - without requiring preinstalled special software[4]. The system should also provide means for the efficient persistence and backup of data, as well as advanced search functionality within it. It should be easy to deploy, maintain, and extend for possible additions, such as advanced automated analysis methods, other types of data, etcetera. A web-based system provides the desired platform independence, and requires only software that is packaged as part of all available operating system, i.e. a web browser. It provides a central entry point for controlling access, meets the requirement of high availability and its usage is so wide spread, that it is safe to assume that computer laymen should not have trouble using it.

The use of a database in the back-end guarantees the persistence and backup requirement. A web-based system also allows the easy integration of video playback and visualization by the use of specific plug-ins. The appropriate choice of technology combined with a modular design will ensure the system's flexibility and extensibility.

1.3 Thesis Structure

The first chapter has served as an introduction to the overall subject and objective of this thesis.

The second chapter is going to examine the biological aspects of the bee dance research. The chapter also lays the foundation for the following sections by giving an introduction to information systems and media content management systems in particular.

The third chapter starts with the requirements and design of the application. The design phase is followed by a brief discussion of the used technology and a description of the implementation.

The fourth chapter deals with the visualization plug-in for the extracted bee dance trajectories. It is very similar to the third chapter in its general structure. It defines the plug-in's requirements, has a technology discussion, and introduces the technology used. The implementation covers the challenges of the task as well as the implementation itself.

The implementation chapters are concluded with the fifth section which will be a discussion of whether the original objective was successfully met and how. The discussion is concluded with an outlook about possible enhancements and extensions for the future research and analysis.

[4] i.e. tracking, word processor, or spreadsheet software

2 Fundamentals

This chapter provides background information on the current research on honeybee behavior (*Apis Meliferra*) and summarizes the development of information systems, in particular web-based media content management system. It is supposed to give the reader the necessary understanding of the biological aspects and technological developments involved, in order to be able to evaluate the proposed solutions. For a broader overview about the objective, layout and general purpose of this thesis, the gentle reader may refer to the 1st chapter.

2.1 Honeybee Research

Honeybees are an integral part of our ecosystem, due to their role as pollinators. Throughout history, bees were often recognized as being indispensable for agriculture and were worshipped accordingly by many civilizations. In ancient Greece, for example, bees were considered the messengers of the Gods and for Egyptians honey was of religious significance. [6]

Today, about a third of the world's food production, equaling US$ 217 billion a year, still depends on the pollination by domesticated and feral honey bees. [1] So obviously humans have been depending on bees for a very long time and thereby always had a strong interest to understand bees and their behavior. Better knowledge of bees helped to solve problems in the past and enables humans today to make better use of their capabilities. This is especially true since the recent appearance of the mysterious colony collapse disorder (CCD), which keeps scientists guessing as to what is causing it. CCD has ravaged bee populations around the world and may become a serious threat to food production in the future. [7]

2.2 Bee Dance Language

Besides the bees' ecological and economical importance to human civilization, the extraordinary form of communication among them is what makes bees a fascinating research subject for scientists. Bees can be observed to perform particular dance patterns after they have returned from foraging food, while other bees follow the dancing forager bee attentively. Scientists believe that bees communicate the location of food sources to each other by performing these dances. Aristotle was the first to record them, but it was not until 1946, when Karl von Frisch discovered that information about the location of a food source is encoded and transmitted via the dance. He believed that bees

communicated information by dancing and thus called it the "bee dance language". [2] No other species of insects is known to possess such a sophisticated form of symbolic communication. For one part, this would make bees quite extraordinary in the world of insects, but it also evokes justified criticism, questioning if the bee dance language really is a language. Adrian Wenner was among the most vocal opponents to Frisch's theory. He challenged the notion of information transmission and instead pointed out that all other insects communicated using odors. The resulting "bee dance language vs odor plume" controversy has been debated ever since.[3] Hence, now even the bee research community has a stake in solving this question that transcends mere scientific curiosity.

2.3 Bee Dance

2.3.1 The Round Dance

Now since so much debate surrounds the bee dance, the next section will explain what the dance actually is. Biologist distinguish two types of dances, the *round* and the *waggle* dance. The round dance is used by forager bees if the food source is in close proximity to the hive. Whether any further information about the food source's location is given is still subject to debate. [8] [9] The definition of what is in close proximity and what is further away varies with the bee race.[10]

Figure 2.1: **Round Dance**

2.3.2 The Waggle Dance

The waggle dance is the more complex dance and is at the core of the bee dance research. Unlike the round dance, the waggle dance contains information about the location of the food source. It consists of two distinct movement patterns, the 'waggle run' and 'return phase'. It begins with the 'waggle run', during which the bee performs a trembling movement, or waggle, while moving in a straight line on the comb's surface. At the end of the straight waggle run, the bee alternatingly turns left or right to move back to the starting point in a semi circle. This section is therefore called the 'return phase'. These movements are repeated several times and are only interrupted by the dancer bee offering food samples to follower bees (trophallaxis). Altogether, the different dance phases form a path whose idealized outline resembles the shape of the figure "eight". The path a bee dances is called the bee dance's trajectory, and will be referred to as such for the remainder of this thesis. The follower bees that pursue the dancing bee,stay in physical contact with the dancer by repeatedly feeling the dancers tail with their antennae. [11]

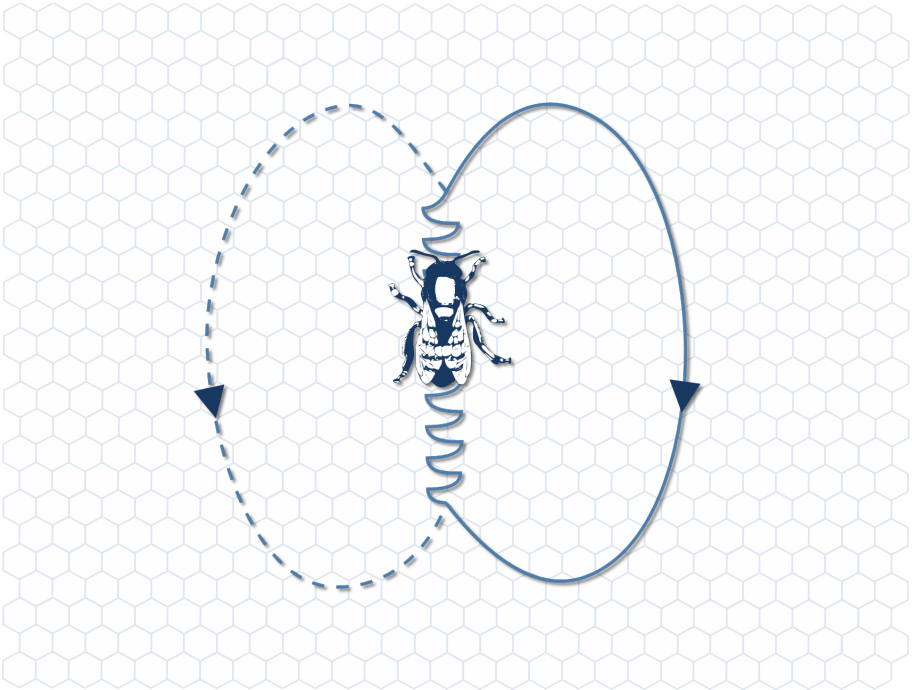

Figure 2.2: **Waggle Dance**

The dance seems to be a reenactment of the flight to the promoted food source. [12] The variations of the dance's length and duration correspond with the distance from the hive to the food source. This is also true for its orientation towards the sun. The dance is performed on the vertical surface of the comb and the angle of the dance depends on the orientation of the food source from the hive and its angle from the sun. Figure 2.3 illustrates this effect on the dance. If the food source is in direct line with the sun, the

waggle run is performed straight up and down parallel gravity's direction. But if the food source's location is not in line with the sun, the whole dance is rotated in the same angle as the food source's angle towards the sun, called the azimuth.

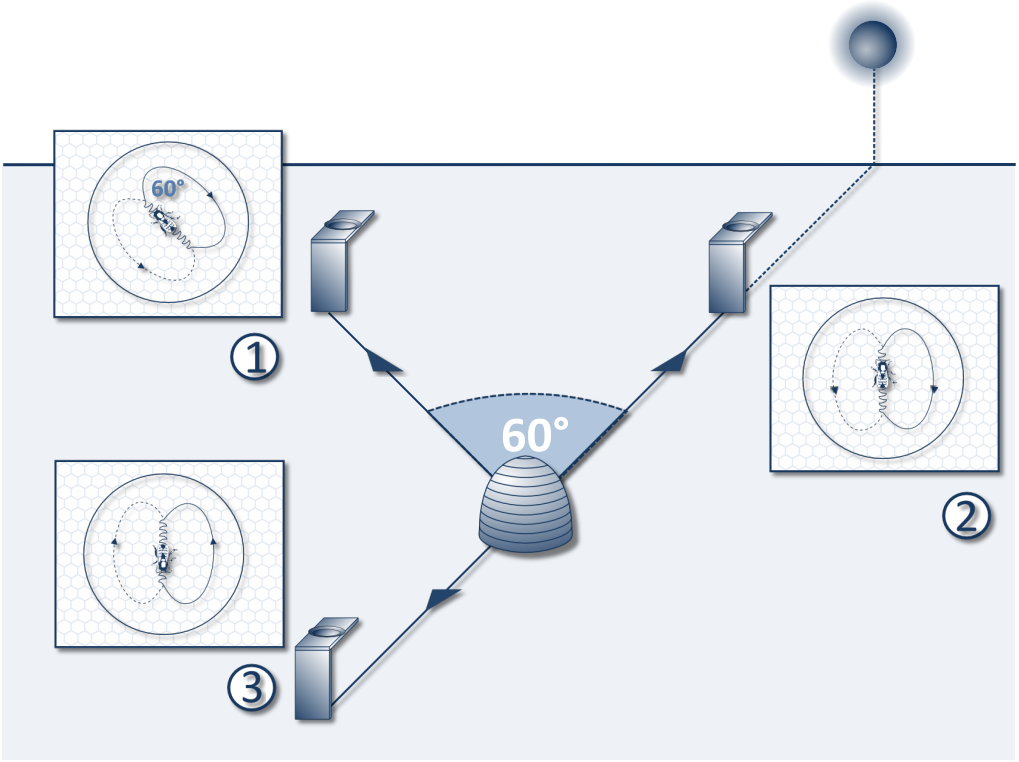

Figure 2.3: **Waggle Dance Orientation**

1) The food source's location spans a 60° angle from the hive to the sun, thereby rotating the dance's center axis accordingly. 2) The food source's location is in line with the sun and hive and the dance's center axis remains unchanged. 3) The food source is inline with the hive and sun, but on the opposite side, therefore reversing the dance's direction.

The information encapsulated in the bee dances have been deciphered by biologists through correlating dances with the known location of the food sources they were describing. Afterwards, the varying parameters were extracted using statistical methods on this data. Even though the content of bee dances could be decoded, the real question occupying scientists, is which signals are involved in transmitting the information. The dances are performed in the total darkness of the bee hive so optical signals cannot have much relevance. Many other, nonvisual signals have been identified, yet their meaning and significance still needs to be further explored. The main signal being researched are acoustic near fields [13], jet streams [14], electrical fields [research group Menzel], pheromones [15], trophallaxis [15], [16] and body contacts[17]. Biologists believe that a combination of these stimuli is necessary for communication, but are unclear exactly

which ones. [18]

2.4 Project RoboBee

The project RoboBee[1] is aimed at researching the bee dance and the significance of the different stimuli that are used for communication during the bee dance. It is the result of the combined interdisciplinary efforts by the Artificial Intelligence study group at the Department of Computer Science under Professor Raúl Rojas[2] and the Neurobiology study group under Professor Randolf Menzel[3]. The project was created in 2005 and is centered around the development of a biomimetic bee robot, hence the name Robobee, to help isolate the determining aspects of communication via the bee dance.

The idea was to develop a state of the art robotic bee, that could mimic the bee dance behavior. The idea itself is not new though. Michelsen [19] first developed a robotic bee in 1990, but the results were not scientifically recognized, since the amount of recruited bees was not statistically significant. Figure 2.4 shows the second and most recent version of the robotic bee. This time the robot has been equipped with capabilities that can imitate most of the signals that are potentially relevant to the communication of bees.

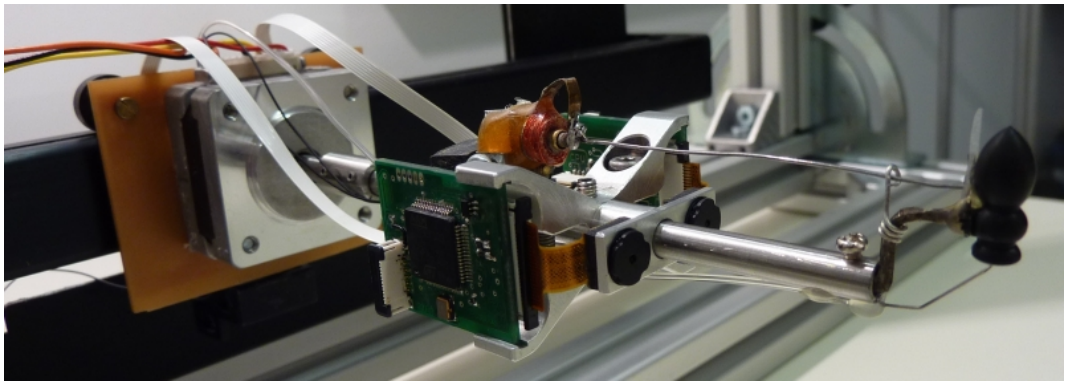

Figure 2.4: **Bee Robot 2.0**

The second version of the Robobee, complete with optical sensors for collision detection and artificial, motorized wings to create jet streams.

The robot provides researchers with a tool to isolate the important signals by subsequently deactivating the robot's different stimuli. This will enable them to analyze the signals and their roles in the communication from dancer to follower bees. The aim for the computer controlled bee robot is to mimic realistic behavior movements to increase the acceptance by real bees. Many of the signals suspected to be relevant for the communication are simulated by the robot. It can "dance" a given trajectory and avoid

[1] http://www.robobee.eu
[2] http://www.inf.fu-berlin.de/groups/ag-ki/index.html
[3] http://www.neurobiologie.fu-berlin.de/menzel/menzel.html

other bees by using visual sensors for collision detection. [20] It has fixtures for providing food samples to other bees (Trophallaxis) and can induce electric fields. Lastly, it will be outfitted with artificial wings to create jet streams that are also part of a bee's dance. The emanation of different scents or pheromones will be achieved by directly applying them on the robot. The capabilities of the robot almost match the whole list of communication signals that were considered to be relevant in the previous section. The signals that are currently not implemented will be in the robots next version.

2.4.1 Video Tracking Software

The bee movement is the integral component of bee dance and robotic bee's capabilities. The robots artificial generated bee dance trajectories are the result of the preliminary work that would later result in the Robobee project. The goal of this precursory project was to track the waggle dance trajectories on high speed infrared video recordings. These were necessary since the bees' waggle movement are executed about fifteen times per second in the darkness of the hive.[4] Once the tracking software was available, the trajectories could be captured and analyzed. Their analysis enabled researches to extract the parameters of the dance motion. These parameters then served as the basis from which a controller software could dynamically extrapolate entire dances.

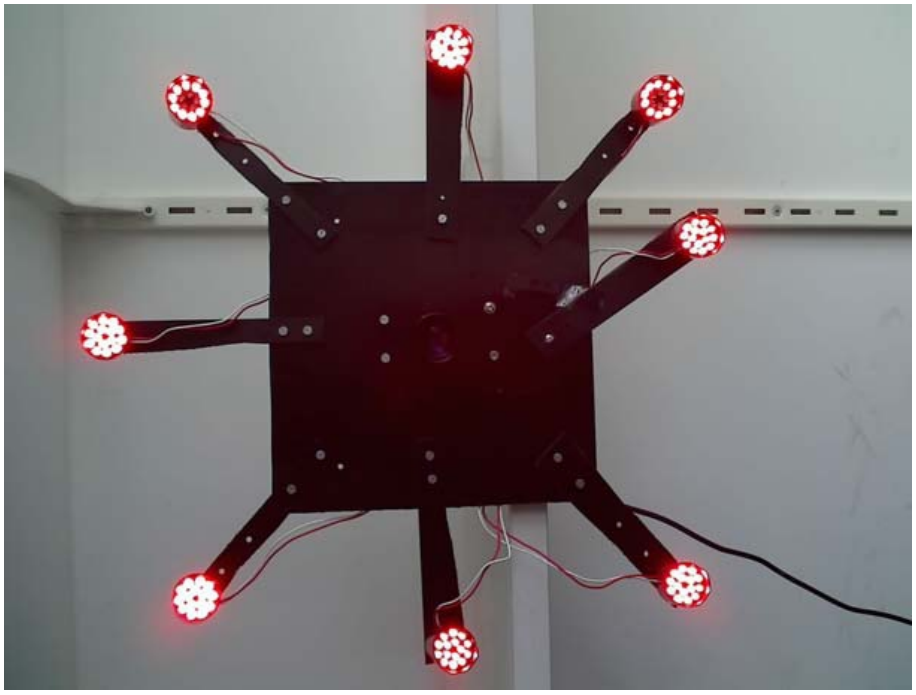

Figure 2.5: **Infrared Lighting**

Array of infrared diodes used for lighting the hive during the recording of high speed videos. (Source: [4])

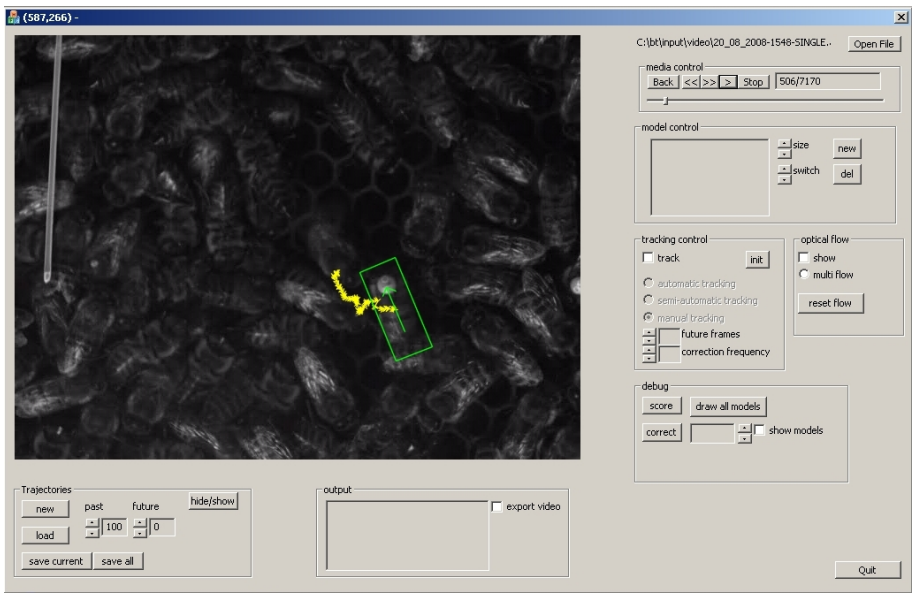

Figure 2.6: **Tracking Software**

Image of the software used to track the bees' waggle dances in the high speed videos. (Source: [4])

The videos were used to obtain parameters for creating artificial bee dances, but they represent a valuable resource beyond this purpose. The wealth of collected data in form of video recordings is enormous. They not only contain the waggle dances, but many other events and behaviors. These range from the trajectories of the follower bees, to trophallaxis, the round dance, and to the effects of the bee robot on real bees.

The majority of the collected material hasn't even been sighted yet, let alone properly contextualized and analyzed. Unfortunately, there is no suitable system in place that could support researchers in handling this task. The amount of data has long surpassed the limit for handling it via the currently used workflow. It consists of an unhandy mixture of different software applications and simple folder structures. Unsurprisingly, this creates the need for a system that automates the majority of data handling and allows new means to contextualize, organize, and search the different kinds of existing data.

At this point, the needs of bee dance researchers and the advantages of computer based data management converge to form the goal of this thesis – the implementation of a content management system. It also underscores again the advantages of interdisciplinary research efforts. Computer science can provide researchers with tools that allow them to work more efficiently thereby opening up new vistas that were not possible before.

The problems that have arisen in the course of introducing computer based video analysis have only been briefly touched. The following section will first detail on the specific ramifications of the problems and then continue to line out the elements of a

software based management solution.

2.4.1.1 Data Management Issues

All of the files which have been created during the video recording experiments are currently stored and ordered using a simple file system and basic folder structure. Folder in combination with file names are used to organize and label videos and trajectories. Videos and their trajectories are only related by matching file names. There is nothing like a managed status property indicating if it contains any trajectories and whether they have all been tracked or not.

Any kind of metadata of the experiment, during which a video was recorded, is stored in separate Excel spread sheets. This data consists of the date, time, location, and detailed weather conditions, and is related to videos by matching the experiment's date with the one in the videos file name. There is no reliable mechanism to relate video and trajectory files with their experiment metadata.

To view the already tracked videos for further analysis, one has to use the unstable, operating system dependent, tracking software. This software is unable to display multiple trajectories simultaneously, thereby excluding the option of comparative analysis.

Also, the entire collection of videos is stored on a single network attached storage unit (NAS) and can only be accessed by directly connecting to a small, private local area network (LAN).

As a result of keeping experiment and video data separate, search and order capabilities are inefficient and extremely limited. Already processed content can also not be easily browsed, searched, or ordered according to certain attributes.

While such an organizational workflow may work fine for very small amounts of files without any associated metadata or relations between them, it is clearly inadequate for the magnitude[4] of this instance. [21, page 1]

The resulting need to handle different programs for analysis and the lack of automatized file organization has many negative implications. Manual management of digital resources is very time consuming and cumbersome. This usually results in an error prone workflow. The lack of tools to efficiently relate, search, sort and give an overview of data in its entire context is a severe impediment to proper evaluation and analysis. The unreliable software to playback videos and trajectories aggravates the situation even more. Additionally, the limitations on the direct access of data preclude any time saving collaborative efforts. The above mentioned problems distract greatly from sighting and analyzing data, and in the worst case severely impede this task. Scientists are forced to waste a large amount of energy on organizational aspects rather than actual research.

2.4.1.2 Management Solution

The evident shortcoming and complications can be alleviated by introducing an integrated system that manages the data and workflows to significantly improve the analysis process. The system should free scientist of the laborious organization of data and allow

[4] The volume of collected data is about to surpass 300 Gigabytes.

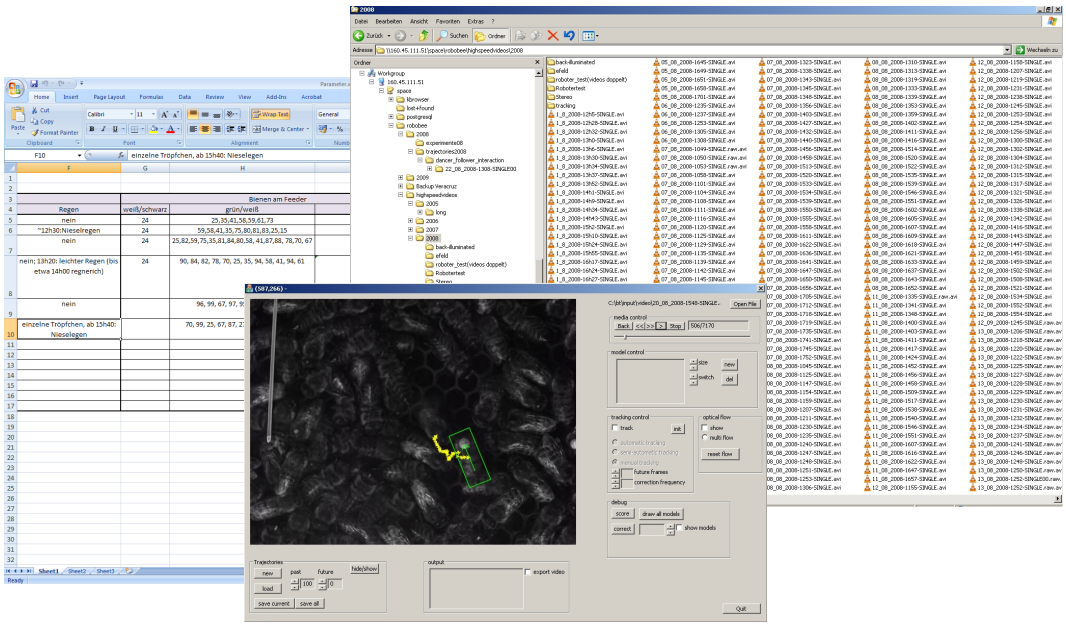

Figure 2.7: **Current Data Management Workflow**

The different programs are used to manage the 300 GBytes of data: A file system, a spreadsheet software and the custom built tracking software.

a uniform, controlled access. It should provide one platform to relate data with each other and to contextualize it with experiment data. File handling and storage have to be automated and integrated with mechanisms for annotating and managing metadata. The system should also offer functionality for searching, ordering, and browsing through the collected data. Other features include the display of correlating information in an appropriate form as well as a reliable and functional playback of videos with multiple trajectories.

The outlined requirements call for an integrated system, that allows the consolidation of the existing workflows into a coherent whole. It should also provide means that go beyond improved file management with metadata, but also provide extended functionality to allow systematic and efficient analysis of the existing data. The demands on such a system correspond with the profile of a web-based media content management system. Figure 2.8 illustrates the complexity and dependencies of the current workflow and is contrasted with the advantages of a web-based solution.

In the following technical section the underlying concepts of theses system will be introduced as well as the reason why they are suitable for this particular setting.

2.5 Information Management

Where the introduction on the biological aspect of the bee dance research has provided a background on what and why exactly the bee dance is being investigated, the tech-

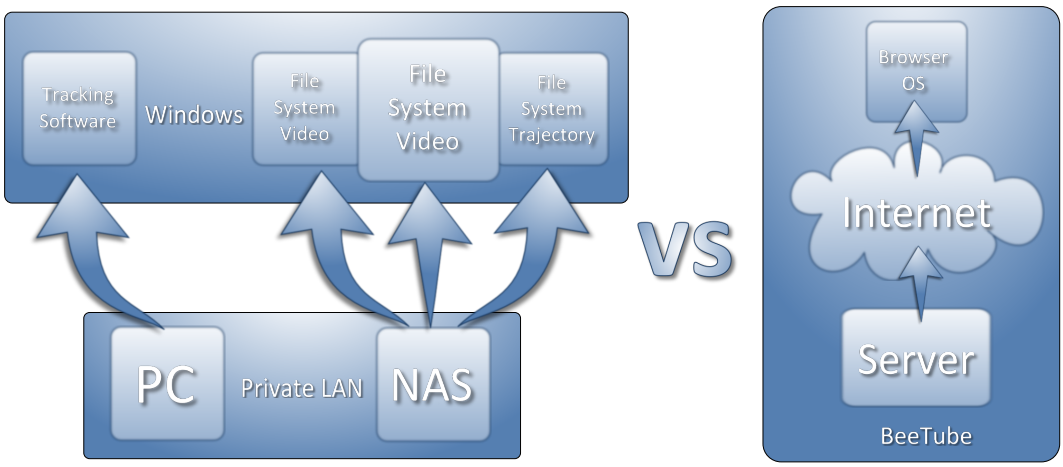

Figure 2.8: **Workflow Comparison**

The structure and dependencies of the current workflow are juxtaposed to the advantageous simplicity and oversight of a integrated web-based system.

nical introduction will show how computer science can help solve the problems that have emerged during this endeavor. In the preceding sections, the research behind the Robobee project as well as the resulting data management issues were covered. This section will give an overview of the basic concepts that underlie the technologies that will be used to realize the needed data management solution. It will cover the theoretical background of the system to be developed, whereas Chapter 3 will focus on the actual implementation.

The proposed system, called BeeTube, will be a web-based media content management system (MCMS). In order to explain what a MCMS is, and where to position it in the overall body of information science, it is important to point out some inherent difficulties, when discussing this topic. The information systems discipline is a very broad field, and there exists a plethora of different definitions and understandings for the same terms. Before dwelling deeper into the subject, it is first necessary to specify which category of information systems will be covered. It is further necessary to clarify which definition of a term or concept is used. So, to avoid any ambiguities, each term will be explained upon its first usage. This thesis is about the design, implementation and use of a software system. It therefore only considers information systems in the context of computer science and defines all used terms in that respect.

2.5.1 Information Systems

Computers were invented for the purpose of processing data. Their success is due to their ability process huge amounts of data at a speed and precision, that are magnitudes beyond human capabilities. In computer science, ISs are said to be used for the computer based collection, storage, management, exchange and display of information. Today, with the pervading presence of computers in every aspect of life, equally many areas of

application for computer based data processing have emerged. Yet, despite the diversity of these data processing systems, they are still all considered information systems.

So, the initially mentioned MCMS can be broadly categorized as an information system (IS) as well. Other classically related categories for ISs include database and data-warehouse systems as well as knowledge management and expert systems. [22, pages 153-157] [23, pages 6,85,155] The purposes of these systems all differ in varying degrees and, depending on what they are processing, so do the definitions of data or information. For a further discussion on the specifics of information systems, and especially content management systems, we need to clarify their individual constituent parts. The next sections will define and detail on the terms that represent the basic components of content management systems.

2.5.2 Data and Information

All information systems have in common that they manage *data* in one form or another. In our particular case, *data* is defined as the simplest form of *information* from which any context or meaning has been removed. Examples of such data are raw numbers or facts, i.e. "8" or "Berlin".

These examples seem arbitrary and useless, unless there is additional data available that is related to this data to give it meaning. [23, pages 4-6] For the aforementioned examples additional data like "8°C" or "Berlin, Germany" contextualizes the formerly meaningless data. This processed data, that has been contextualized is called *information*. [24] Information can range from these simple examples to entire texts, sounds, images, and videos.

Databases, for example, only store data, but the data's metadata and the relationships among the data embodied by the database scheme, altogether implicitly represent information. Data-warehouses have the same basic functionality as databases, but their data has already been preprocessed to better extract conclusions based on the prior aggregated information.

2.6 Content Management

BeeTube is a specific type of *content management system (CMS)*, which is the practical realization of content management (CM). So, the two questions arise, what encompasses content management and what exactly qualifies as content? The answers, of course, vary with the particular area of application and intended functionality of a CMS.

2.6.1 Content

BeeTube is primarily about managing videos and their metadata. A historically important application domain for media related CM and CMS with a focus on videos is the broadcasting industry. Therefore, this domain context will serve as starting point for the first definition for what content is. In the past several broadcasting institutions have

defined the nature of content for this specific field. According to them, content consists of *essence* and *metadata*. Essence is the actual message of content in form of a text, image, sound or video, free from its physical carrier. Metadata, on the other hand, is the information that describes the actual manifestations of essence. Metadata is further divided in content-, material-, and location-related metadata, which range from descriptions of the essence's content to details on the format, encoding parameters, number of copies and type of physical medium. [22, page 4]

This clear distinction is useful in the context of professional, collaborative media production for items like news clips, TV shows or movies, but doesn't quite fit the purpose of the intended system. The definition is too domain specific and detailed. All material already exists in digital form, which renders the concept of essence unnecessary. The same is true for the subdivision of metadata, because in this case the digital media simply does not require such an elaborate categorization.

Consequently, a better fitting definition of content is necessary. Boiko defines content as the combination of different information into a coherent whole that is given a purpose. [23, pages 4-9] Applied to the BeeTube use case, the contextualization of videos and their trajectories with the corresponding experimental data equates to the combination of information into a coherent whole. The intention behind this contextualization is to help scientists in their analysis, which gives the newly combined information a purpose, thus creating content.

2.6.2 Management

After having defined what content is in the context of this thesis, the task of defining content management (CM) is much more straight forward.

Content management is defined as the process that manages the content's lifecycle, which spans from its creation, review, and publishing to its final storage. [25] In the case of BeeTube all data and content is digital. The resulting content management system is thereby completely realized as a software system. This distinction is necessary, since in the initially mentioned broadcasting industry a CMS would entail the handling of physical and digital forms of content, such as physical tapes and digital video files.

2.7 Content Management Systems

A content management system (CMS) represents the actual implementation of the different processes that constitute content management. Translated in practical computer science terms, this means that a software based CMS is used to collect, store, manage, and publish digital content. [25]

Its functionality enables a group of users to contribute and work collaboratively, by offering tools for creating, editing, and reviewing content. It is a centralized mechanism that provides the easy access to and manipulation of data. It also controls this access and modification by offering different user roles, that define what a particular user can view

and edit. Furthermore, it can feature version control and, because of the centralization of resources, a CMS can help to reduce redundant structures.

Depending on the application domain, the breadth of CMS can vary significantly. A CMS can be a standalone application that manages the content of an entire news agency or can be interfaced with other systems, like the billing system of an e-commerce website. It can also be just an integrated component of a much larger enterprise content management (ECM).[5] In case where content in a CMS is associated intellectual property rights (IPR), the system usually also supports digital rights management (DRM). [23, pages 8-10]

2.7.1 Web Content Management Systems

Today, the term CMS usually implies a web-based content management system (WCMS), since it is the most prevalent. WCMS are used for all sorts of content and purposes. Dynamic sites that deal with news, e-commerce, encyclopedia, dictionaries, and many types of media, like music and video, are *all* considered to be a WMCS. The content generation, review and publishing does usually not require expert knowledge, such as programming skills. It can therefore be easily employed in a lot of different domains and levels.

The simplest WCMS are based on a 3-Tier architecture. The browser to view the CMS, represents the first tier, the CMS, to generate content and to contain the necessary workflows, the second, and a database, in which the CMS stores its data, the third and final tier. [25] Larger CMS consist of even more tiers due to sophisticated scalability or heterogenous data sources. [25, pages 62-67]

WCMS can further be categorized according to the content that they manage. This thesis focuses on the category of *media content management systems (MCMS)* whose content is primarily composed of digital media, such as images, sound and especially video. The pioneer of publicly accessible MCMS and one of the world's most popular video content delivering websites is undoubtedly YouTube. [6] It is also the obvious inspiration behind the naming of the MCMS BeeTube that is developed over the course of this thesis.

2.7.2 Digital Asset Management Systems

When discussing media related MCMS, it is also necessary to consider and compare *digital asset management systems (DAMS)* with them. DAMS focus on the handling of digital media and in many ways resembles the characteristics of a MCMS. A DAMS can be used for a number of tasks.

Generally, a DAMS supports and facilitates the workflows, that are necessary to effectively create digital media content. Its main function is to order, version and manage all available digital assets, that can be used in digital media production. Its applica-

[5] ECM includes the entirety of business processes, work flows and digital resources of a company.
[6] http://en.wikipedia.org/wiki/YouTube

tion purposes range from the aforementioned broad casting industry to game and music production and well beyond that.

Digital media becomes an asset if it has rights that can be commercially exploited. A DAMS is not only responsible for managing the digital asset by creating, editing and storing it, but also to manage its commercial exploitation by ensuring the proper digital rights management. [22, page 5-7] [21, page 2]

DAMS are specialized for particular types of media and allow advanced features, such as real time transcoding into different formats.[7] They are heavily focused on managing the assets metadata and its distribution. DAMSs are used by professional providers of digital media content and, hence, their scope are usually well beyond that of a web-based MCMS.[8] At the same time they are mainly used within an organization and not so much for the actual publication of created content, which is a key aspect of a web-based MCMS. [21, page 3-5]

BeeTube's purpose is the management of digital media, such as videos, texts and files. But its focus is more on the contextualization, ordering and publishing of thereby created content, rather than the sole organization of digital assets. [23, page 195] The degree of the DAMS' specialization on the videos metadata is simply not required. DAMS are also not intended for the integration and association of the experimental metadata for publication purposes. WCMS are usually format agnostic when it comes to storing data and content, but they do support certain formats to provide optimal support. The focus on a more general, content publication functionality fits the requirements of BeeTube better than the specialization of a DAMS. [22, page 10]

2.8 BeeTube

The preceding sections have provided the background and motivation behind creating the web-based content management system BeeTube. After detailing on the data management problems that have emerged during the research efforts of the Robobee project, a possible solution was developed. This solution is comprised of a set of requirements for a data management system. In the next step, the suitable theoretical context was given to contextualize the computer science aspects of the system. Yet, before the next chapter will describe the actual implementation, the systems requirements are briefly summarized.

BeeTube will be implemented as a web-based content management system, with some alterations specific to its use case. These customizations include the omission of a version control and an explicit review process before publication. These features are usually part of WCMS that are focused on articles and news, and these capabilities are not required in this case.

BeeTube has to provide a central repository for all files and data gathered during the video recording experiments. The data has to be easily contextualized by this system, meaning that the videos, trajectories and experiment metadata can be easily correlated

[7] http://www.cmswatch.com/Feature/124-DAM-vs.-DM

[8] http://blog.widen.com/blog/image-storage/0/0/dam-vs-wcm–understanding-the-differences-

by computer laymen. The thereby created content has to be easily displayed, edited, and searched. The system must provide different roles for viewing and modifying the data. It must also be accessible for authorized users over the Internet via a browser. Furthermore, the system should be bandwidth and storage efficient.

The goal is not only to replace the existing workflow of handling scattered data on different files and formats with a uniform access, but to provide enhanced ordering, search and contextualization capabilities, that create a new and more efficient analysis workflow. BeeTube should provide researchers with a comprehensive tool that actually allows a new level of systematic research and thereby representing additional value.

The next chapter will describe which technology will be used and detail on the actual implementation of BeeTube.

3 BeeTube

The following chapter is about the implementation of the web-based content management system BeeTube. First, it briefly recapitulates the specific requirements of the system that were developed in the previous chapter, and then it discusses the possible technology choices for implementing BeeTube. The technology discussion covers the various alternatives and explains which ones are preferable in this particular context and why. After having explained and introduced the basics of the chosen technology, the implementation section will detail on the systems architecture and the specifics of its realization.

3.1 Requirements Specification

The motivation and problems behind creating BeeTube have been described in the preceding chapter. Its general properties have also been laid out by formulating a set of requirements the system should fulfill. Since the possibilities to implement such a system are quite numerous, it is further necessary to combine the system's requirements to a coherent profile. The created profile will then be serve as the basis for choosing a suitable technology.

BeeTube is a web-based content management system. It needs to be accessible via a browser, and all content manipulation, such as creating, editing, viewing and deleting, must be performed by using website typical presentation, complete with navigation and input forms. It must provide functionality for adding, i.e.uploading, videos, their trajectories and other relevant files. The system must then offer the ability to assign related files to each other and annotate experiment and file specific meta data. It has to feature the possibility to add all experimental data and offer sort and search functionality for the system's content. Videos must have a preview function that allows to watch bandwidth efficient proxy copies of the original videos, including any trajectories. The specifics of the system video playback and trajectory visualization will be covered in detail in chapter 4. Finally, BeeTube must also control access by offering different user roles for viewing and managing content.

3.2 Technology Discussion

The question which technology to use for implementing BeeTube requires some discussion. First, it has to be decided whether BeeTube will be implemented by customizing an already existing, generic WCMS or if the system should be developed from scratch

using a web development framework[1] Second, it has to be decided, which generic CMS or technology framework will be used and why.

3.2.1 Customization vs Creation

There are many different ways to realize a system such as BeeTube. The first decision for implementation is whether to customize an existing system or to create it oneself. In any case, only free available, i.e. open source compliant, systems and technologies are considered. Commercial solutions are costly, usually not customizable and a lot of the available free alternatives are as mature and professional as the commercial ones.

The advantages of customizing an existing system are numerous. The basic functionality of a WCMS is already there, and depending on the WCMS only the structure and style of the different content views has to be customized. Other aspects such as administration, user roles, security, backend functionality and integration has all been taken care of. But all these aspects also have a downside. A WCMS may provide much more functionality than is actually needed, but can lack the ability to fulfil special requirements at the same time. If the given use case can not be mapped to the workflows implemented in the WCMS, it is very difficult to modify them accordingly, because of their complexity. Another drawback can be a mandatory revision update of a given WCMS, that can lead to incompatibilities or loss of the customization. This would make modifications or, in the worst case, even re-implementation necessary. Most WCMS are designed with creating some form of text based content in mind, such as news and articles. BeeTube and its focus on video and experiment data management represent a special use case that few WCMS are able to handle.

Creating a WCMS from scratch can be considered the complimentary approach to modifying an existing CMS. There are scenarios where such an approach is advantageous. If, for example, the workflow of the proposed system is not mappable, or inadequate to the ones available in an existing WCMS, a custom development might be the only choice. Also if the systems are oversized or their scope is well beyond what is actually needed, the effort of creating a slim system that meets all needs can also make sense. Another benefit is that a self developed system can be customized and extended as desired, provided its design allows it. The obvious downside lies the necessity to provide all the needed features oneself and thereby creating potentially redundant work. Further, specialized aspects such as security, safety, and scalability all have to considered and properly taken care of. This is usually already considered in existing systems – albeit to varying degrees.

The downside of having to create a WCMS from the very beginning can be mitigated by the capabilities of modern frameworks. The amount of work necessary can be greatly reduced by frameworks, that offer automatized functionality and prepared components to handle a lot of mundane and time consuming development tasks. By employing such a framework, the endeavor of developing an own WCMS becomes much more feasible and economic.

[1] This is also known as a make-or-buy decision in economics.

In the case of BeeTube, a custom realization is favored over customizing an existing solution. There are several reasons to support this decision. BeeTube does not require a full fledged WCMS, since features like versioning and reviewing are not required. Most CMS focus on the publication of coherent textual content and images, whereas the management of videos and experiment data requires a different kind of workflow.

The customization of such systems tends to be tedious and complicated, and the required work to adapt it almost equals the effort to create a custom implementation, based on a modern framework. A newly designed system can be developed to exactly match the lined out requirements, while the self-contained nature of the system allow the use of frameworks which will significantly reduce the development effort.

The next step consists of choosing a suitable, efficient framework. The next section will briefly introduce the different types of frameworks and highlight their benefits and shortcomings. The specification for BeeTube, as defined by the 3.1 Requirements Section, will be the basis on which a framework will be chosen.

3.2.2 Web Development Technology Frameworks

There exists a plethora of web development frameworks, based on different languages and technologies whose sophistication, functionality and purpose vary greatly. So, a thorough discussion of available frameworks would go well beyond the scope of this thesis. That is why, only the general characteristics of the web development frameworks will be briefly covered. They all have in common that they were created to simplify development, by providing libraries, interfaces or automatization that reduce the work developers have to do themselves. Frameworks are either specialized on particular aspects of web technologies such as front-end, business logic, or back-end components or they provide all of the aforementioned in a single package. In case of the latter, a lot of the prominent ones adhere to convention over configuration (CoC) principle.

The abundance of frameworks has resulted in the situation, that today a lot of capable and equally powerful frameworks exist for almost any language or technology that has any significance for web development. [2] So the decision which framework to chose is not unnecessarily influenced by the language on which it is based, but the quality, scope and power of the framework. Since a WCMS consist of many distinct components, there exist equally many frameworks for each component. Of course the scope of frameworks varies, but usually one has to combine and integrate several frameworks with each other. If the technologies are compatible, meaning designed for combined usage with another particular or family of frameworks, this is usually not much of a problem. However, the integration usually requires some extra configuration and overhead.

The CoC was created to tackle this problem of configurational overhead and complexity. The idea is to limit configuration by defining certain conventions, which in turn allow the much faster development and deployment for the majority of typical use cases. These frameworks usually encompass the 3-Tiers of web applications of front-end, middle-ware and back-end, respectively the web view, controlling business logic, and database.

[2] http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

Though the data volume to be managed by the system is rather large, the system's scope is not too extensive due to the comparatively small amount of individuals using it. Therefore, BeeTube does not require an architecture based on frameworks that are intended for large scale applications, but instead can be realized using smaller frameworks that are easier to handle. BeeTube is also a self-contained system, that does not have to integrate legacy systems or interface with other preexisting systems. Frameworks using the CoC approach and covering all 3-Tiers of web application development are the ideal match for such scenarios. They usually consist of the combination of several frameworks that have been seamlessly integrated and allow the quick and fast development of fully functional CRUD[3] applications. These can be extended to provide the desired functionality of a custom WCMS.

There exist quite a few CoC frameworks that offer the all-in-one 3-Tier functionality for quickly creating CRUD applications. The most prominent among these is Ruby on Rails.[4] It greatly influenced the creation of similar frameworks for other languages and technologies, like CakePHP[5], Pylons[6], and Grails[7], to just name a few. All these technologies are well suited for implementing BeeTube. Hence, the decision for a particular technology is made based on additional factors.

Grails is one of the more recent additions to the field of frameworks. It integrates the two very popular and powerful frameworks Spring[8] and Hibernate[9] by using a Java based programming language Groovy[10]. Its central components as well as language are mature and popular frameworks all based on Java. It can thereby access the wealth of the Java API, as well as freely available libraries and containers of the Java Community. Java's ,i.e. Groovy's, platform independence advantage is matched by the majority of the other framework's languages, like ruby, PHP or Python. But Java's abundance of available libraries, documentation and technologies is unsurpassed.

Overall, Grails characteristics are well suited for the development of BeeTube. Additionally, the existing deployment infrastructure is also based on Java and the authors primary background is in Java.

The next section will briefly introduce the concepts and technologies behind the Grails framework, for a better understanding of the implementation details of the developed application.

3.2.3 Grails

Grails is an open source web application framework and has been developed following the convention over configuration paradigm. [26, page 1] It encompasses all of the three

[3] CRUD is an acronym for create, read, update and delete operations on database content.
[4] http://rubyonrails.org/
[5] http://cakephp.org/
[6] http://www.pylonshq.com/
[7] http://www.grails.org/
[8] http://www.springsource.org/
[9] https://www.hibernate.org/
[10] http://groovy.codehaus.org/

fundamental tiers of a web application. Figure 3.1 illustrates the technology stack of Grails.

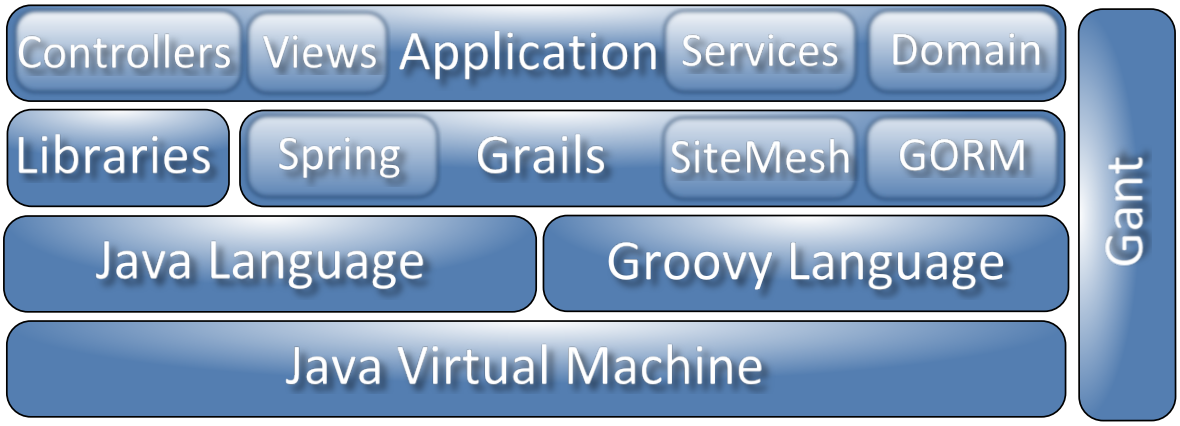

Figure 3.1: **Grails Technology Stack**

The middle-tier is comprised of the object-relational mapper (ORM) Hibernate and Spring, which is an inversion of control (IoC) container and wrapper for Java. Together these frameworks allow the storage and retrieval of Java or Groovy objects into a database. Hibernate introduces an abstraction layer, that decouples the application from the specifics of the used database system. It thereby allows an easy migration between database systems. Spring supports Hibernate, by wrapping the complexities of the Hibernate usage in a standardized API. In combination with default configuration and automatization mechanisms, both are used by the Grails platform to offer simple, configuration reduced interfaces for persisting data. GORM is an extension on top of that, which offers mechanism for simplified queries on the database.

The frontend consists of Groovy Server Pages (GSP), that are rendered via SiteMesh. This web page layout framework transforms the different GSPs into coherent HTML websites for display in a browser. All of these different components are integrated via the Java based Groovy language. It has full access to the Java API, but has a reduced syntax, which is oriented on scripting languages like ruby. [26, page 68] [27, page 3]

Besides providing an infrastructure for all necessary components of a web application, Grails also features all the necessary tools for development. It is furnished with a tool for the projects build process called GANT, a standard web container, i.e. application server, named jetty and the lightweight database HSQLDB.

A Grail application is by default designed according to the MVC paradigm. It has domain classes, representing the model, controller classes that are responsible for managing the page and workflows of the application, and a view in form of a GSP based frontend. [26, page 69]

All of these elements of the Grails platform will be used in creating BeeTube. The

next section will highlight the how and where the different components were used during implementation.

3.3 Implementation

3.3.1 General Architecture

The BeeTube web application consists of several components to process the different workflows. The basic architecture follows the Model-View-Controller (MVC) paradigm as seen in figure 3.2. In the figure all the distinct classes and GSPs of the model, view, controller and others have been abstracted to a general component to clarify the underlying architecture.

In Grails the model consists of domain classes, which are modeled according to the different elements of the application. In BeeTube's case the most relevant elements are experiment, video and trajectory. All of these elements have specific attributes, such as date and file names and are translated via Grails into corresponding database entities.

Each of the domain classes has a corresponding controller, which is responsible for any manipulation of an domain class instance via the view. The controller manages all of the users interactions via the view. This includes the aforementioned CRUD actions of a model or domain instance as well as page navigation, searching, browsing, and up- and download of files.

For each relevant domain class exists a view, that allows the user to modify a particular instance. The action invoked by the user is then handed from the view to the controller which actually performs the requested action.

3.3.2 Controller

There exist generally four different types of actions that the controllers perform or delegate. Any modifications of a model instance is delegated to the proper domain class which propagates the change via Spring and Hibernate to the corresponding database entity. The database only holds the metadata of the different domain class instances. Any files, such as videos, trajectories or attachments to an experiment are stored and managed by the file system.

Any view modifications, such as page navigation, searches or browsing are processed and returned to the view, which renders the response for display in the users browser. The view is transformed from the server side GSPs to plain HTML coded websites, that, in case of viewing a video, contain the Flash/Flex based visualization plugin developed in Chapter 4 Visualization for playback.

The central task of BeeTube is to provide the easy access and viewing of the video stock. The system stores the uploaded original video and creates a smaller, lower quality proxy copy. This copy is then embedded in an HTML site for convenient viewing via the web. In order to create this copy, BeeTube has a converter, which transforms not only the video but also any uploaded trajectories so that these can be combined for playback

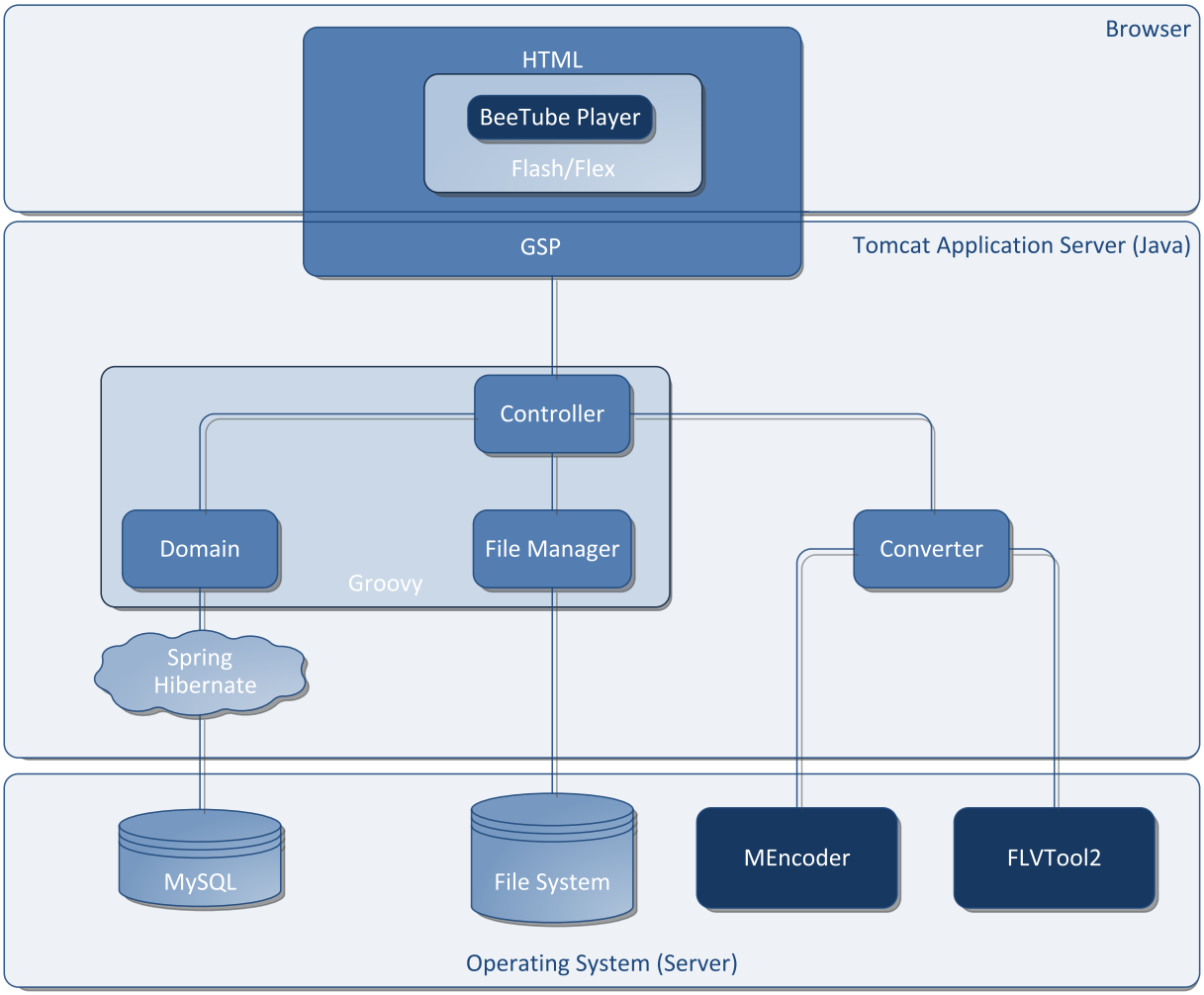

Figure 3.2: **General Architecture BeeTube**

The generalized architecture of the entire application with all its components on all three tiers.

via the web. Trajectories are converted via Java classes, but Video conversion requires the use of stand alone tools. These are handled by the converter element, which itself consist of several Java classes.

The up- as well as download, view, conversion, and deletion of various files creates the necessity for fourth and last component of the application, the file manager. It provides the functionality for any file system action necessary by the controller.

3.3.3 Model

BeeTube's model is created from the application's domain classes, its properties and and their relations as depicted by figure 3.3. The main entities include experiment, video and trajectory. A video may or may not belong to an experiment, which can may or may not have one or multiple videos. Each video in turn may or may not have one or multiple trajectories attached to it. Theses entities further provide attributes that contextualizes them such as the experiments metadata about, time, place and weather conditions. They also have entity specific metadata, such as duration or file name.

Aside from the already outlined relations between experiment, video and trajectory there are the entities of location, attachment, bee, and event. These are the location of the experiment, the number designator(s) of the bee(s) participating in the experiment, and any type and number of events that occur during a video, e.g. trophallaxis.

BeeTube also provides a limited form of access control, by distinguishing between three user roles. These are none, guest and administrator. Without any authorization a user is completely denied access to the system. A guest can only search, browse, list, view and download experiments, videos and trajectories. The administrator has full access to all available information and can create, modify and delete any content of any entity.

Since a personalization of user roles is currently not required and therefore omitted there also exist no sperate database entity. The user management is currently by the application, but can be easily augmented by a more sophisticated mechanism in the future.

3.3.4 View

Figures 3.4 through 3.11 show the three main entities of BeeTube and exemplarily their main pages for managing them.

After having implemented a system that complies with requirements of the previous chapter and therefore provides a management tool for the accumulated digital data, the next chapter will focus on completing this system. The visualization of the trajectories is an integral part of the requirements for BeeTube. It will help improve the scientists workflow almost as much as the CMS itself and allow a unprecedented and more efficient analysis of the recorded videos and their trajectories. The chapter 4 will cover the details of creating a plug-in which will allow for the watching of the video proxy copies and their trajectories via the web.

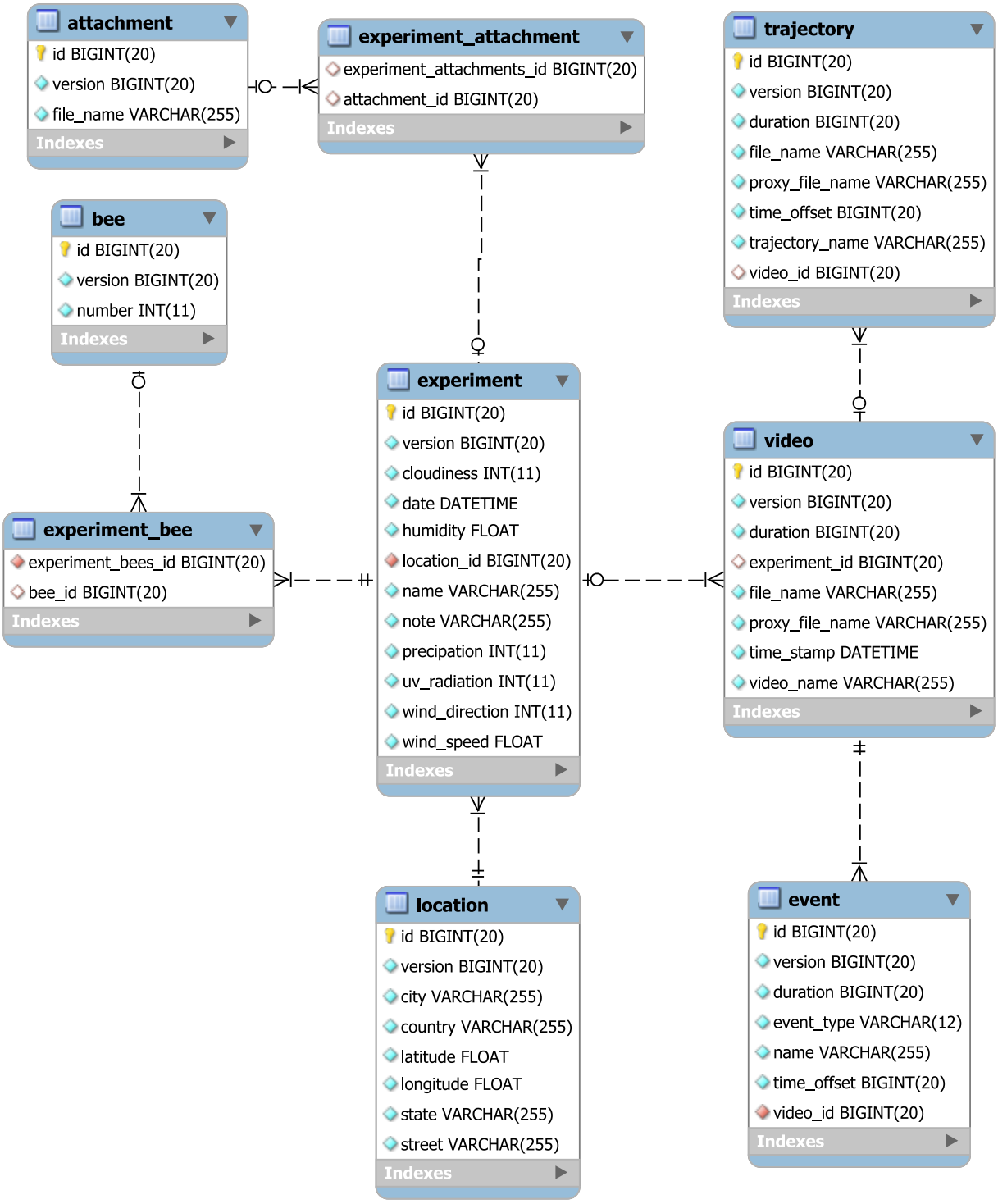

Figure 3.3: **BeeTube Database Schema**

BeeTube's database schema as created by Grails from the domain classes properties and relations.

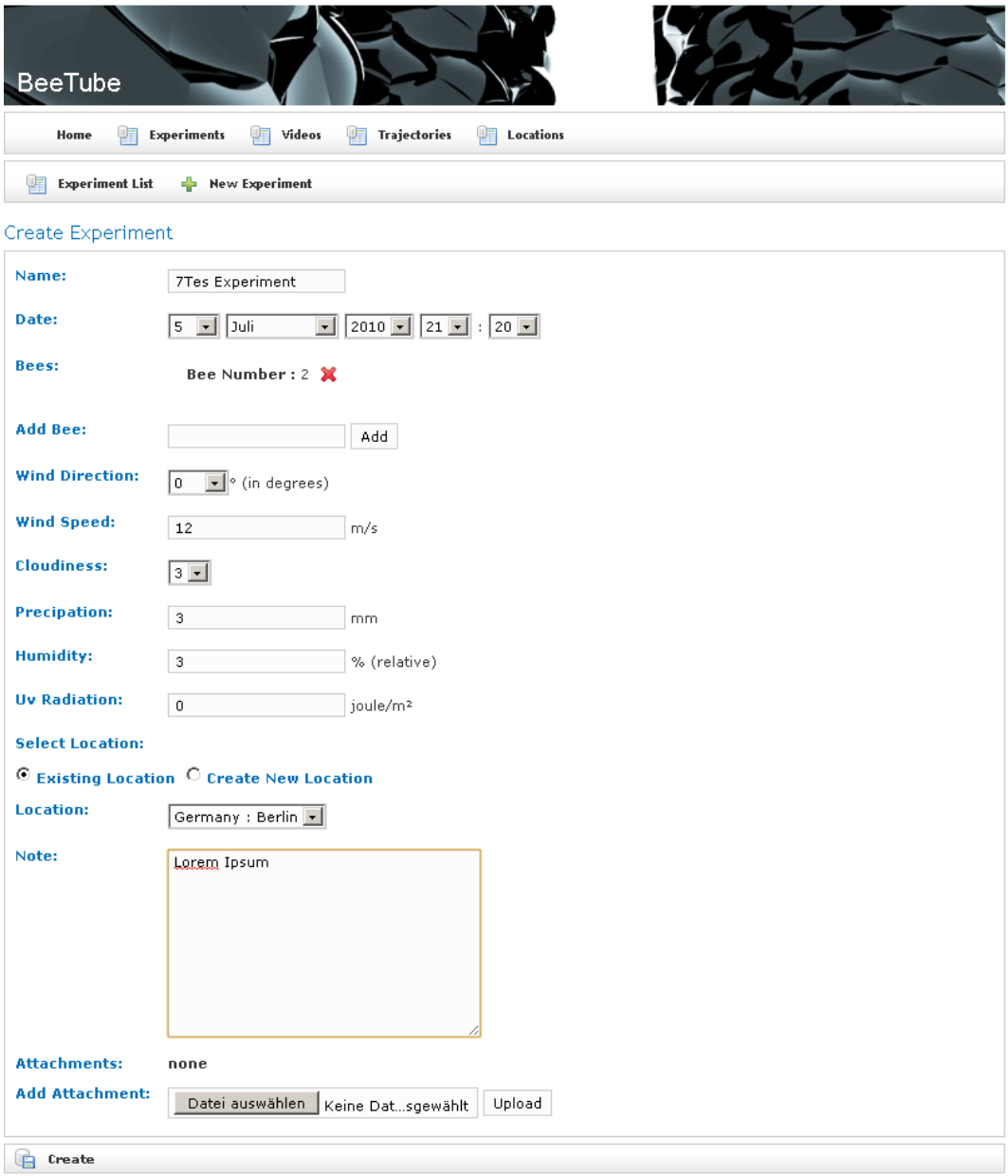

Figure 3.4: **Screenshot Experiment Create View**

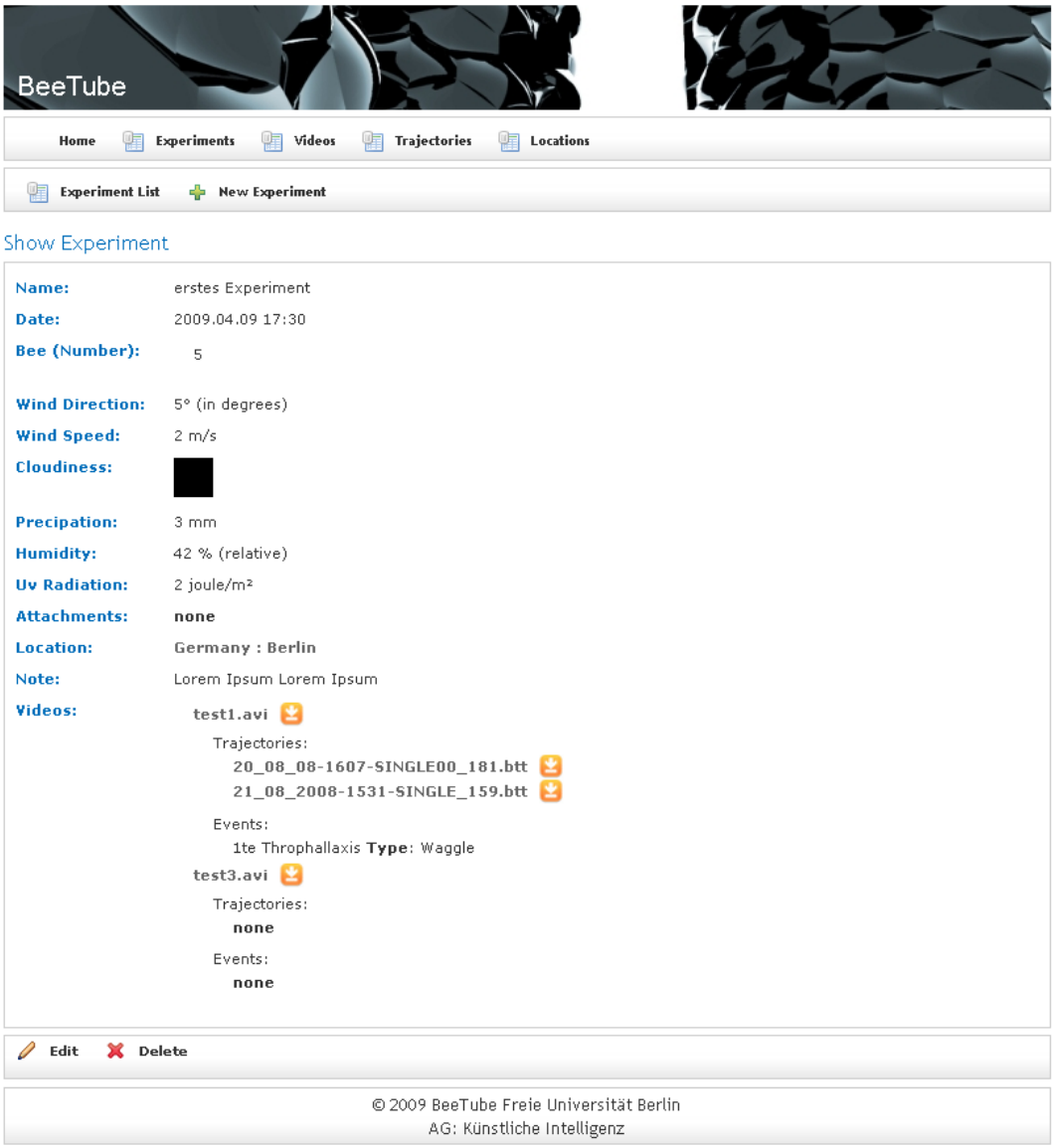

Figure 3.5: **Screenshot Experiment Show View**

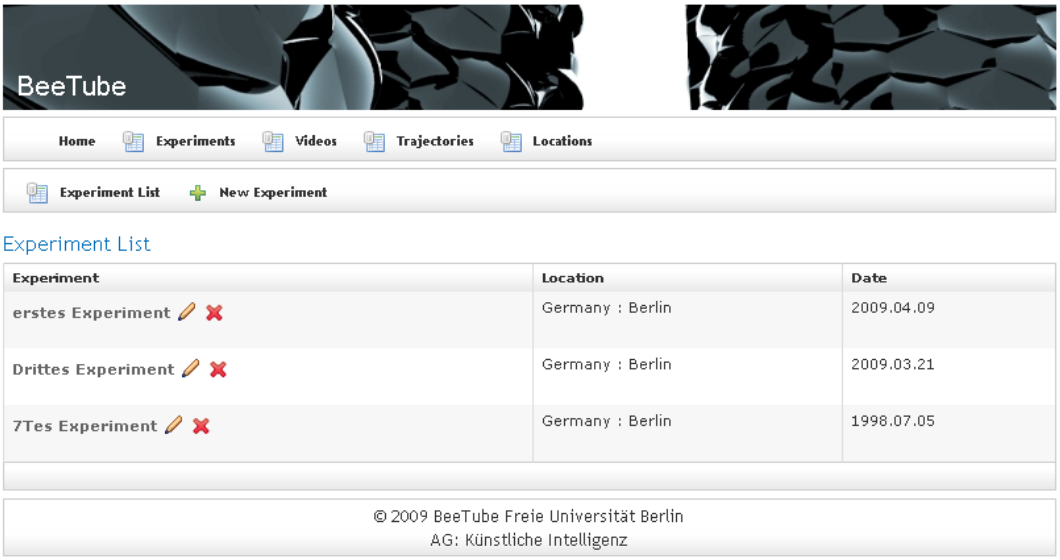

Figure 3.6: **Screenshot Experiment List View**

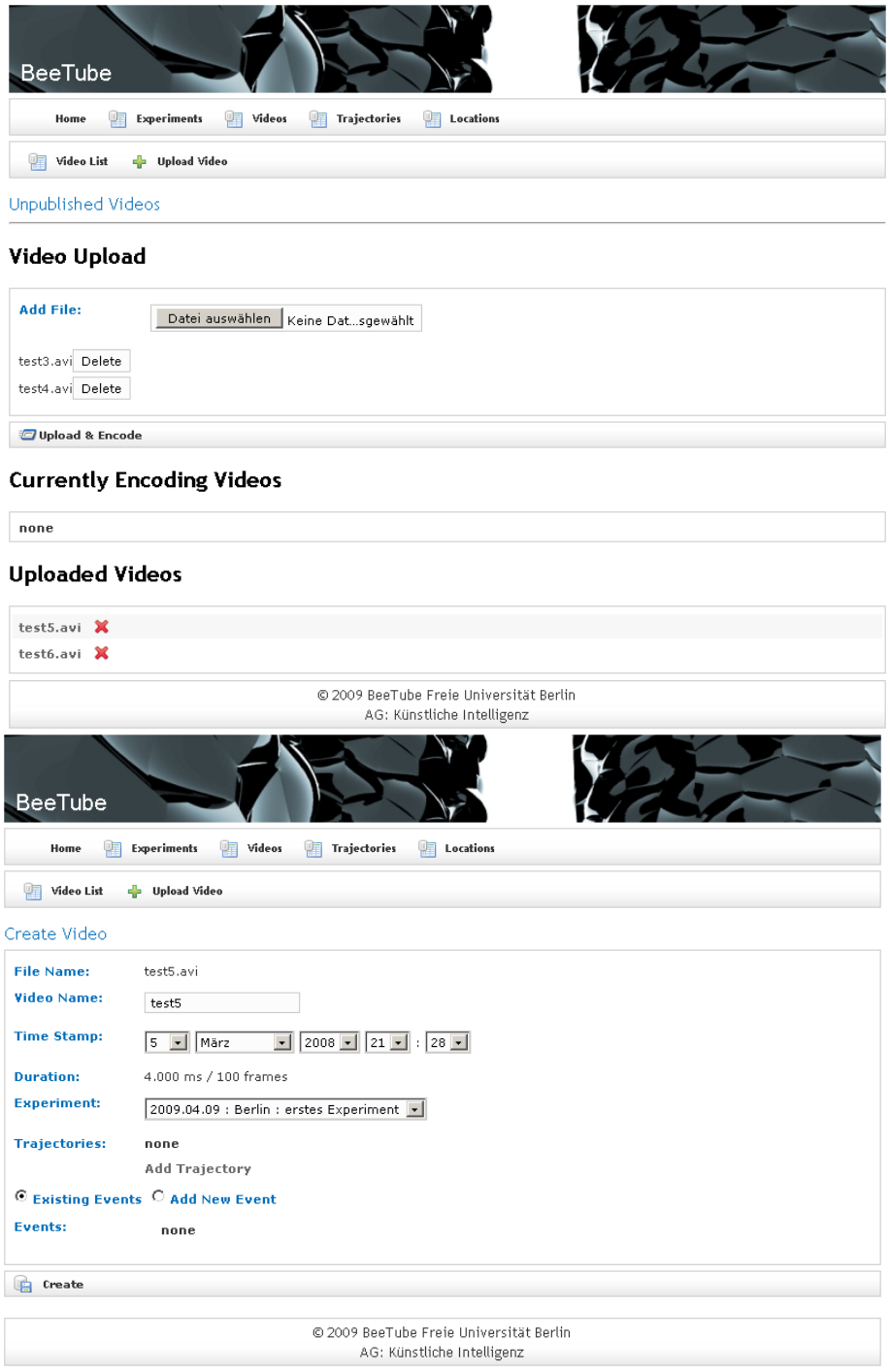

Figure 3.7: **Screenshots Video Views**

Upload, create, and list and upload view for videos.

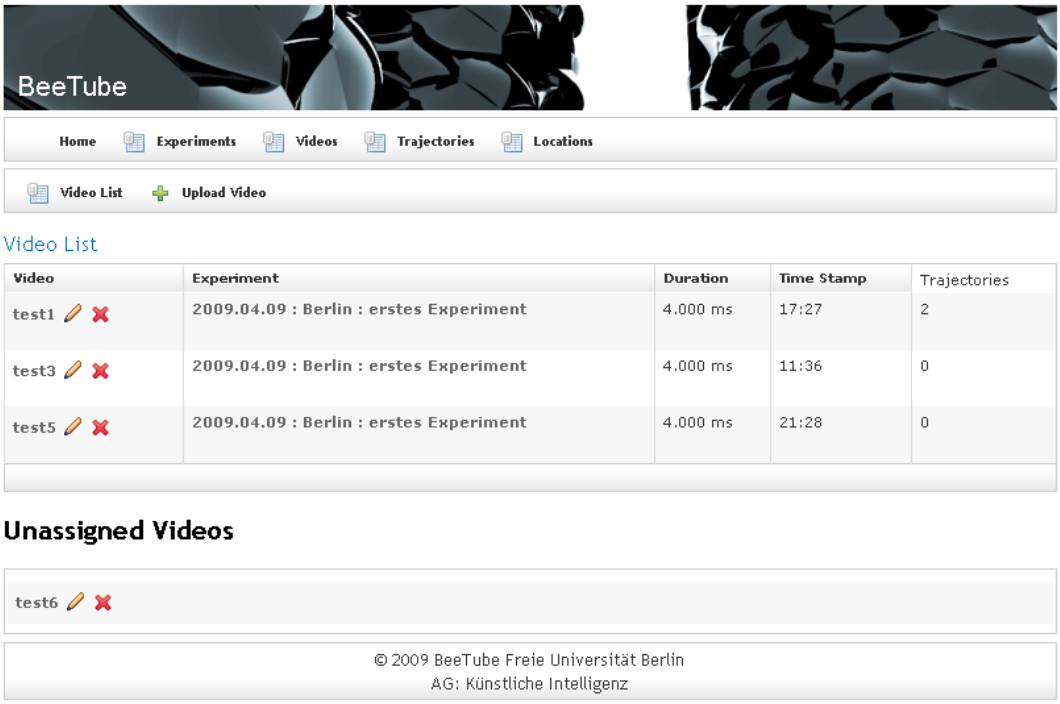

Figure 3.8: **Screenshot Video List View**

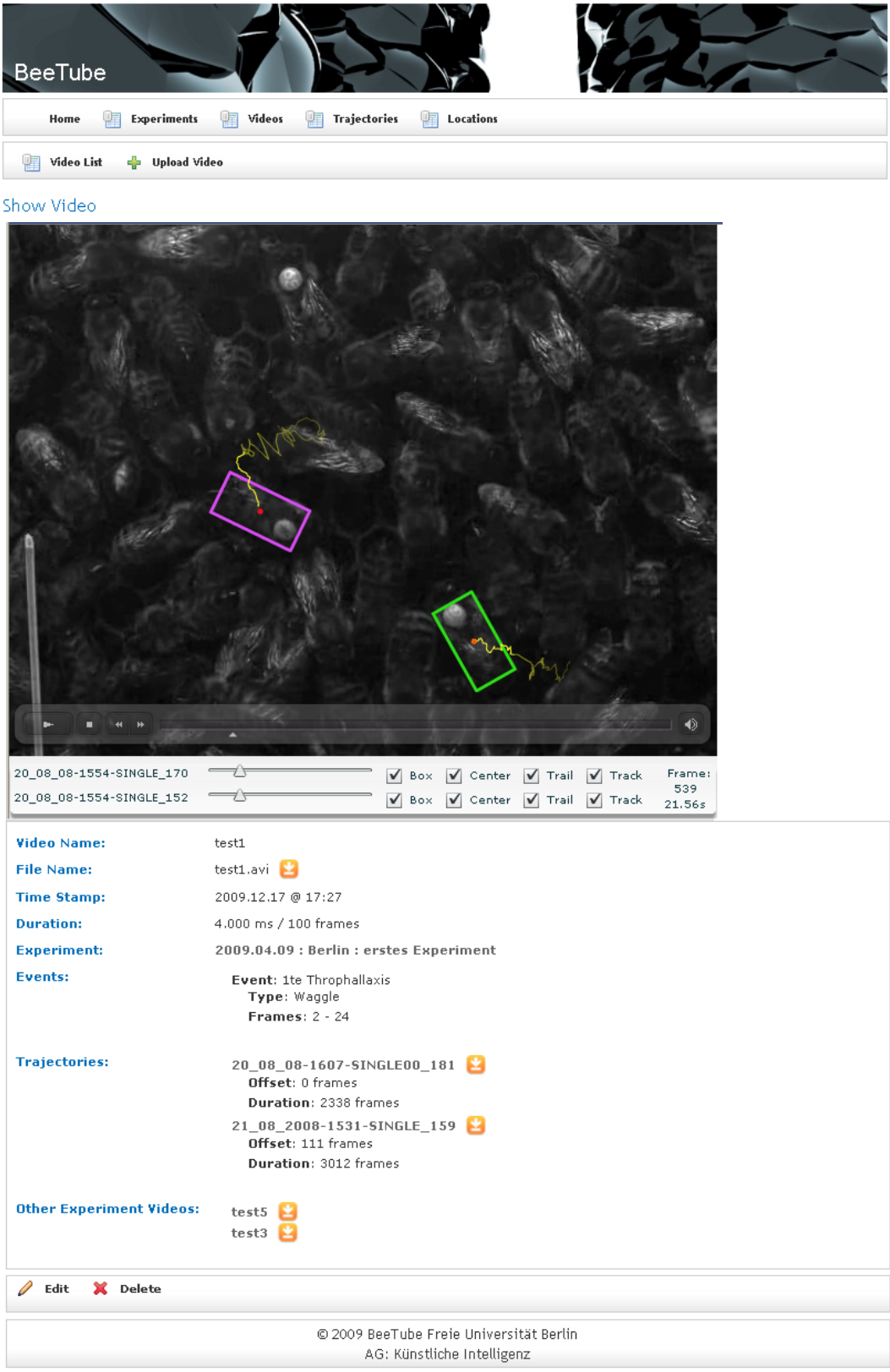

Figure 3.9: **Screenshot Video Show View**

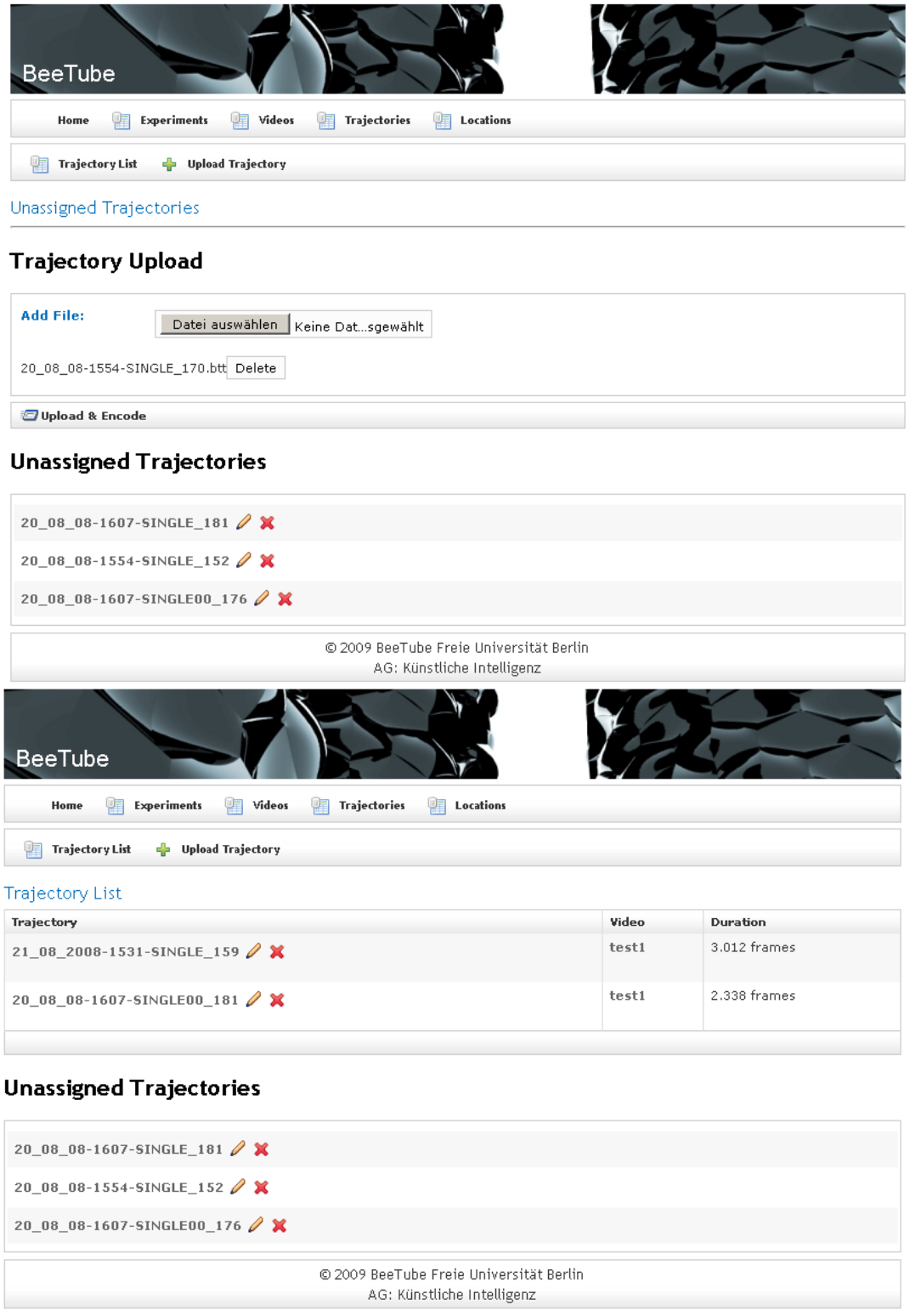

Figure 3.10: **Screenshots Trajectory Views**

Upload and list and view for Trajectories.

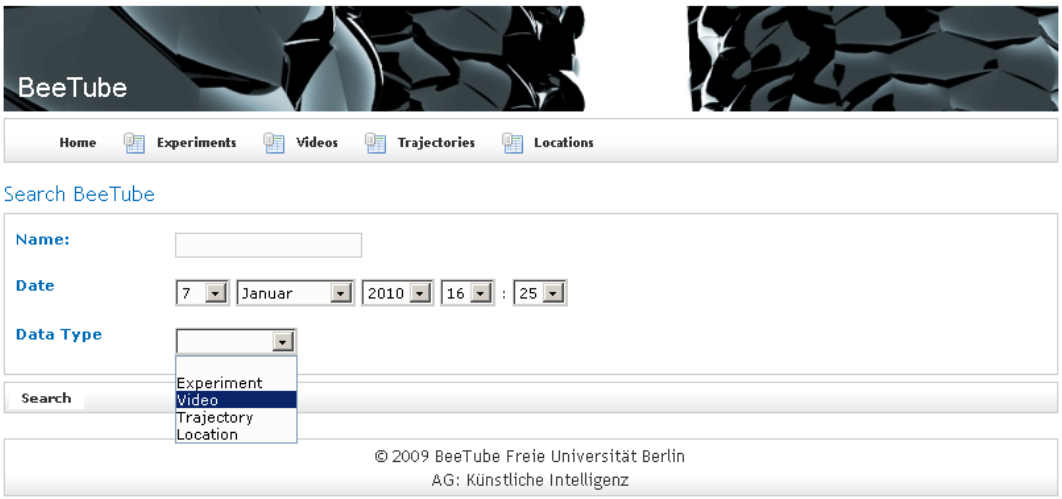

Figure 3.11: **Screenshot Search View**

Default search form view.

4 Visualization

This chapter deals with the visualization of bee dance trajectories during video playback. It is centered around the design and implementation of a plug-in, that allows the simultaneous display of multiple trajectories. The first section gives a brief introduction on the background of the visualization by describing the functionality and problems of the original video tracking software. This is followed by an outline of the requirements of a web-based visualization plug-in. After identifying the requirements, the different potentially suitable technologies for implementation are described, followed by an evaluation to determine the proper platform. The evaluation is the prerequisite for the actual implementation. The implementation section is twofold. In the first part, the technical challenges and the steps to their solution are discussed. The second part consists of details on the implementation's software structure.

4.1 Video Format Fundamentals

The bee video tracking software constitutes the foundation for the computer based video analysis. It tracks the identifying features of a bee within a bounding box around the its circumference. The bounding box highlights the bee's position and the position of the box's center determines the coordinates of the trajectory, as shown in figure **??**. The trajectories are stored in a simple text format, that saves the coordinates for each frame in which tracking occurred. The trajectory information consist of the frame number, the (x,y)-coordinates and the angle of the bounding box in degrees. [4]

To simply replay the video and its trajectories the user has to load the video and trajectory, into the video tracking software. The software uses Microsoft's DirectShow framework and therefore only runs under Microsoft Windows. It requires preinstalled codecs and additional *dynamically linked libraries (dlls)* that cement this dependency even more. Unfortunately, the software is not very robust and tends to crash abruptly as can be seen in figure 4.1.

The user interface offers numerous options, and does not provide different modes for tracking and simple playback. Additionally, the software can not simultaneously display multiple trajectories in one video. The overall installation and handling is tedious and complicates the task of analyzing and interpreting the video's content. The improvement of software stability and handling to increase the efficiency of analysis are the primary goal of the visualization plug-in. The specific requirements are outlined in the following chapter.

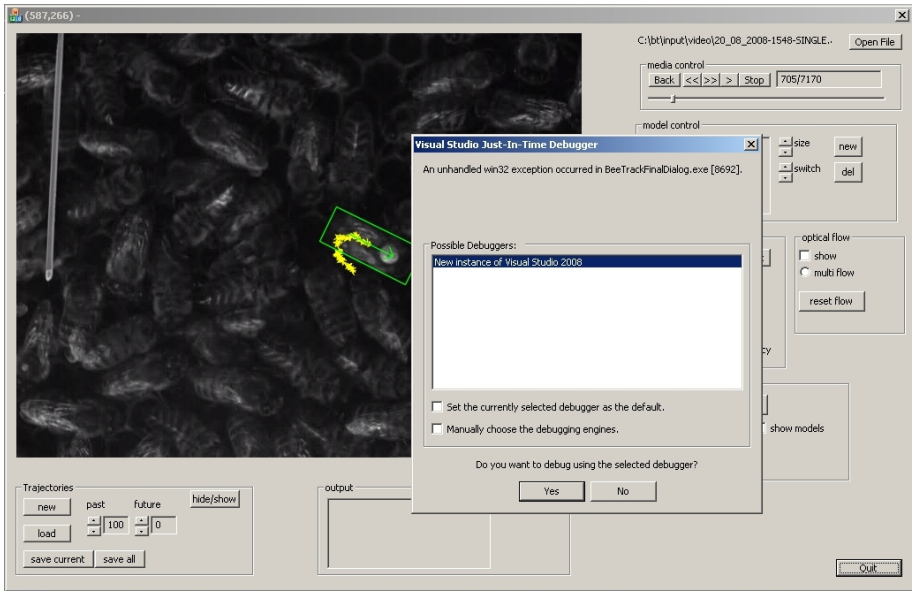

Figure 4.1: **Crashed Tracking Software**

Image of the tracking software [4] which has crashed during tracking.

4.2 Visualization Technologies

4.2.1 Requirements

In order to determine which technologies are suitable for visualizing the bee dance trajectories, it is necessary to define what exactly the requirements are for such a visualization. This visualization software, henceforth called the *player*, should offer at least the same playback functionality as the tracking software. It has to provide an easy to use interface for controlling the video playback and trajectory visualization. The viewer should be able to dynamically overlay the video playback with the frame exact drawing of the trajectory via the user interface. Since a video can contain several dances, one can extract several dance trajectories per video. Hence, the player has to be able to dynamically show and hide multiple trajectories, possibly simultaneously, and provide appropriate controls for modifying their respective display. This requirement exceeds the capabilities of the existing tracking software and thereby creates an unprecedented, additional value to the whole system.

Technically the trajectory visualization consists of several drawn 2D vector elements and shapes including the bounding box and the path the bess has danced. The box is used to highlight the body's position and angle among the other bees on the comb. The bee's center is highlighted by a small circle. The dance path is visualized by a slowly fading trace line with an adjustable length, so that the viewing area is not clogged up by numerous overlapping paths.

The player serves as the playback and visualization device for the proxy copies of the original videos. Consequently, the player as well as the video format have to be band-

width efficient, due to their distribution via the web-based media content management system (MCMS) BeeTube. Other important aspects are the requirements for actual playback, including computational resources as well as preinstalled software. The player should not require any, or at least a very small, preinstalled runtime environment amd should preferably be displayable within a browser. However, if an extra runtime environment is indispensable, the environment of choice should preferably be very common in use. The emphasis on a popular platform ensures that potential technical barriers are reduced to a minimum, while usability is maximized.

4.2.2 Technology Platforms

The standard preinstalled video playback software of all relevant end-user operating systems, i.e. Windows, Linux, Mac OS, are not capable of the required dynamic layered video visualization. Though they support bandwidth efficient video streaming formats, and can even be integrated in their respective standard webbrowsers, by themselves they all lack the capabilities to visualize trajectories. In recognition of the necessity for such multimedia capabilities, the *World Wide Web Consortium (W3C)* released the first draft for a standardized format to blend with other forms of media and time based animation in the *World Wide Web (WWW)*.[1] The specification was called *Synchronized Multimedia Integration Language (SMIL)*[2] and was designed as a platform independent XML format.

4.2.3 SMIL

SMIL is used in combination with the W3C's standard for 2D vector drawing *Scalable Vector Graphics (SVG)*. Together, these technologies can overlay video with animation. SMIL could theoretically be used draw the bee dance trajectories on top of bee videos.

Yet this approach is unfeasible, due to numerous problems. SMIL and SVG, for example, do not offer libraries for building user interfaces[3]. Though customized user interfaces could probably be built, their functionality would be rather crude. However, customizing the players user interfaces is not an option, since they are closed source for the most part. This approach would also contradict the notion of low technical barriers by platform independence. Further difficulties arise due to the very uneven supported of SMIL and SVG by the major video player vendors. Different players support SMIL to varying degrees or have proprietary extensions and formats.[4] As a result SMIL and SVG do not have a noteworthy significance in this particular field. This leads to the conclusion that "classical" desktop video player applications are at their limit and cannot be used for the outlined purpose.

[1] http://www.w3.org/AudioVideo/Activity.html
[2] http://www.w3.org/AudioVideo/
[3] http://www.w3.org/TR/2008/REC-SMIL3-20081201/smil-animation.html
[4] http://soap.stanford.edu/presentations/smil/

Figure 4.2: **SMIL Video Overlay Animation Example**

This example shows the theoretical capabilities of SMIL of video playback in combination with vector graphics and timed text. (Source: [28])

4.3 Plug-In Technologies

There is a group of web technologies available that are very well suited for the implementation of the proposed player. Much like SMIL, the underlying motivation for these technologies is to blend multimedia content, such as videos and animated graphics, with user interactivity[5]. In the wake of the massive expansion of high bandwidth connections larger, more complex applications within the WWW have emerged. These so called *Rich Internet Applications (RIA)* [29] [6] allow the combination of multimedia content and user interactivity on a level, that rivals the capabilities of classic desktop applications.

Where SMIL failed to establish itself as a foundation, these technologies have become the defacto standard on which most of today's RIAs are built. They are all different, proprietary technologies, but by and large offer the same range of functionality. They provide multiple, independent layers that can be combined to animate static with dynamic elements. The resulting multilayered, animated content can then be customized with programmed event listeners that respond to user interaction. Apart from equivalent capabilities, these technologies are all based on the same general architecture. They consist of a runtime environment that is installed in the client software as a plug-in - usually a browser. The environment contains a virtual machine that executes all the code of the RIA. The virtual machine allows these technologies to achieve a, albeit varying, platform independence.

In the following sections, the technical details of each candidate platform and their advantages as well as shortcomings are further explored. The presentation of each candidate platform is concluded by an evaluating discussion to determine the most appropriate

[5] http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf
[6] http://www.w3.org/2006/rwc/Activity.html

technology choice for the later implementation of the proposed player.

4.3.1 Flash/Flex

Adobe Flash is a technology that was developed for the presentation of multimedia content by primarily integrating graphics, animation, sound and video. Since its introduction, its original focus on animation significantly expanded to the integration of user interactive multimedia content, especially video, and RIAs into websites.[7]

Flash applications are written in *ActionScript*. The code is executed by the *ActionScript Virtual Machine* (*AVM*) which is part of the *Flash client runtime environment*, and available for various operating systems and platforms, i.e. desktop (*Adobe Air*), webbrowser (*Flash Player*), and mobile (*Flash Lite*). The current language version, *ActionScript 3.0*, is a fully object oriented language based on the ECMA4 Standard. [30]

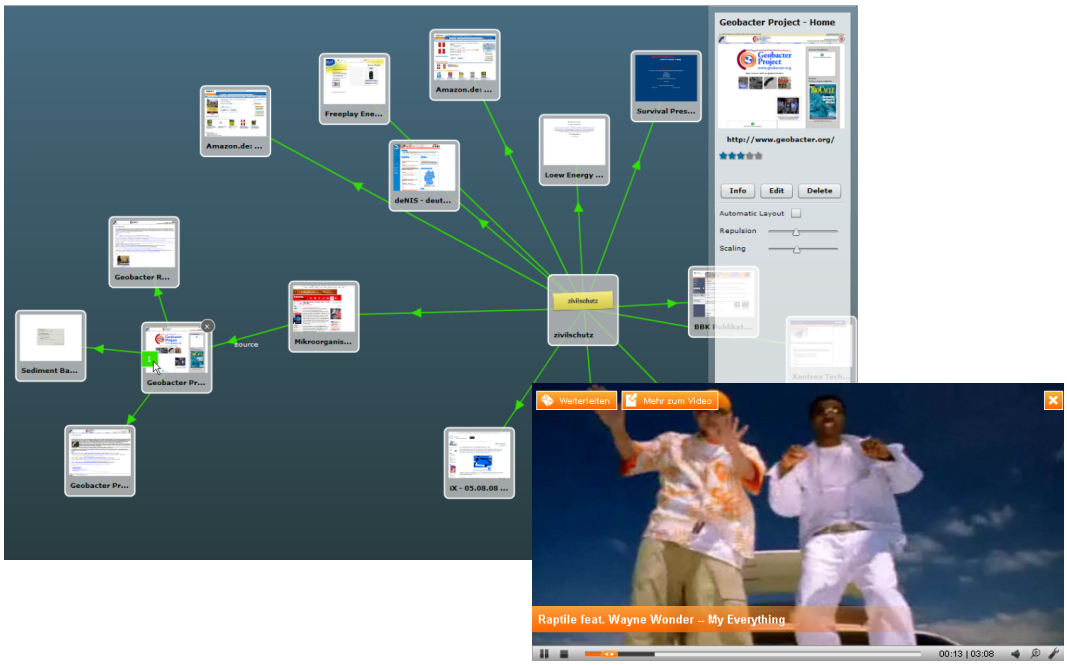

Figure 4.3: **Flash/Flex Examples**

Examples of a RIA Flex client and a customized Flash video player. (Sources: [31] [32])

The Flex framework extends the Flash API and provides standardized components for building RIAs.[8] Flex user interfaces are written in MXML, which is used to declaratively describe the elements of the user interface. These elements are Flex components that can also be annotated with ActionScript event listeners to add programmatic functionality and interactivity. The simple combination and programmatic connection of predefined,

[7] http://www.adobe.com/flashplatform/
[8] http://www.adobe.com/products/flex

standard components enhances and accelerates the development process. Additional functionality, i.e. business logic, must be implemented in ActionScript. [30]

Flex libraries are written entirely in ActionScript and included in the application, since they are not a standard component of the Flash runtime environment. This results in larger file sizes for the application, but also reduces the size of the plug-in installation, so that they do not have to be frequently updated if the framework changes.

4.3.2 Silverlight

Silverlight has been recently developed by Microsoft as a direct competitor to Adobe's Flash platform and therefore is quite similar in its capabilities.[9] The runtime environment is Mircosoft's own .NET framework and its *Common Language Runtime (CLR)* virtual machine, which is mainly supported by Windows. Nonetheless, Silverlight implementations have been ported to Mac OS and Novel has created a Linux implementation in cooperation with Microsoft, called Moonlight. [10] The Silverlight platform has access to a subset of the .NET library and thereby offers a wide, but not the entire, range of functionality. Because of this, it still is a lightweight platform with a comparatively small plug-in installation. [11]

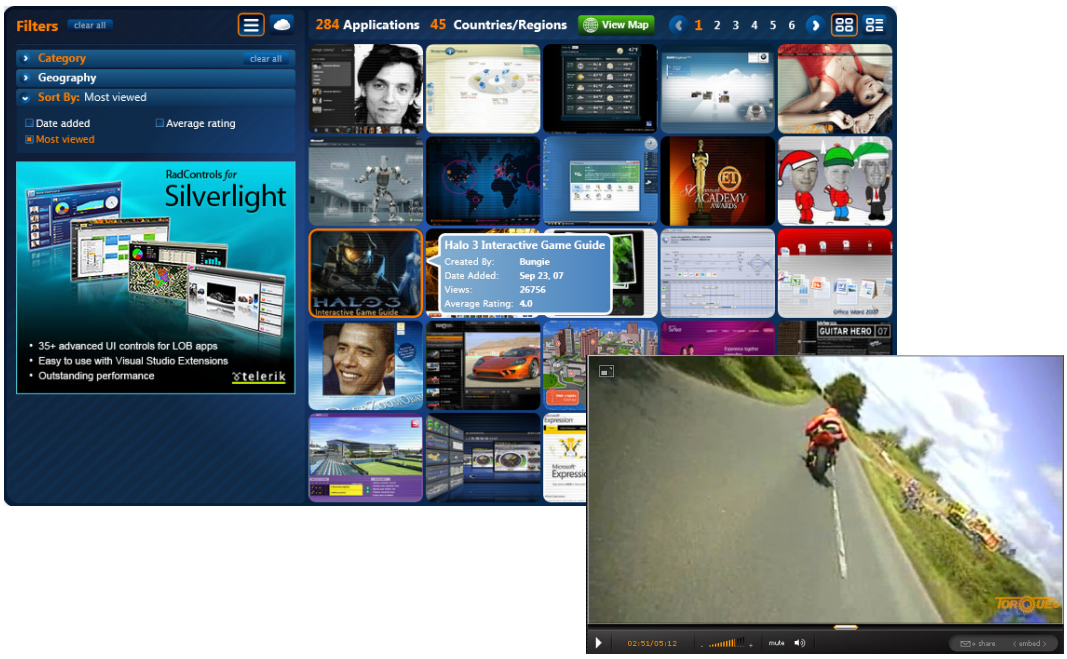

Figure 4.4: **Silverlight Examples**

Examples of a RIA Silverlight client and a customized Silverlight video player. (Sources: [33] [34])

[9] http://silverlight.net/
[10] http://www.mono-project.com/Moonlight
[11] http://msdn.microsoft.com/en-us/library/bb404700%28VS.95%29.aspx

Silverlight application user interfaces are declared in a similar fashion to Adobe's Flex, by using *Extensible Application Markup Language (XAML)*. The program logic is created and connected to the user interface via the .NET framework. The reliance on .NET also allows an unparalleled freedom of choice for the programming language. The application can be programmed in any .NET language, ranging from C# to Visual Basic and IronRuby to name a few. Silverlight 3 is the latest version and supports hardware acceleration for certain graphical effects to increase its performance. [35]

4.3.3 Java Applets/JavaFX

Java Applets were the first self-contained applications that could be integrated into websites and viewed in a browser. They require a regular *Java Runtime Environment (JRE)* to be executed. Thereby, Applets provide full access to the Java API and offer its full functionality.[12] Since RIAs and today's heavy focus on integrating multimedia content did not exist, when Applet technology was first developed, they are not very adept handling these tasks. Their design and programming is oriented at the classic desktop applications of the time. This is why Applets lack support for handling multimedia in combination with animation. They also do not offer a simplified declarative meta language, such as XML, to easily create user interfaces. Instead Applets rely on Java's Swing libraries to programm the interface. [13] The programming is rather tedious and the Applets usually suffer from long loading times. Java Applets are mainly used today for specific purposes, but their formerly prominent role has been reduced to a niche existence.

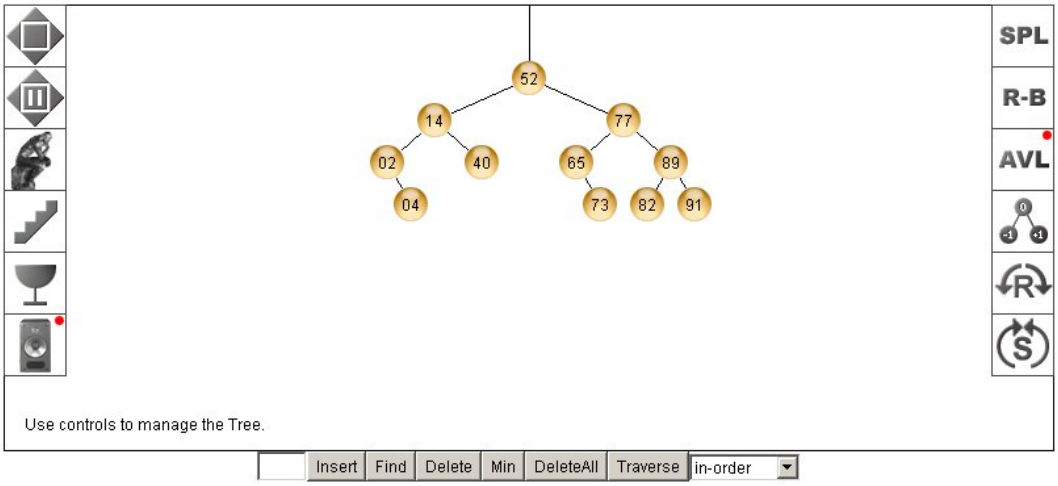

Figure 4.5: **Java Applet Example**

Example of a AVL Tree visualization implemented as a Java Applet. (Source: [36])

[12] http://java.sun.com/applets/
[13] http://java.sun.com/applets/

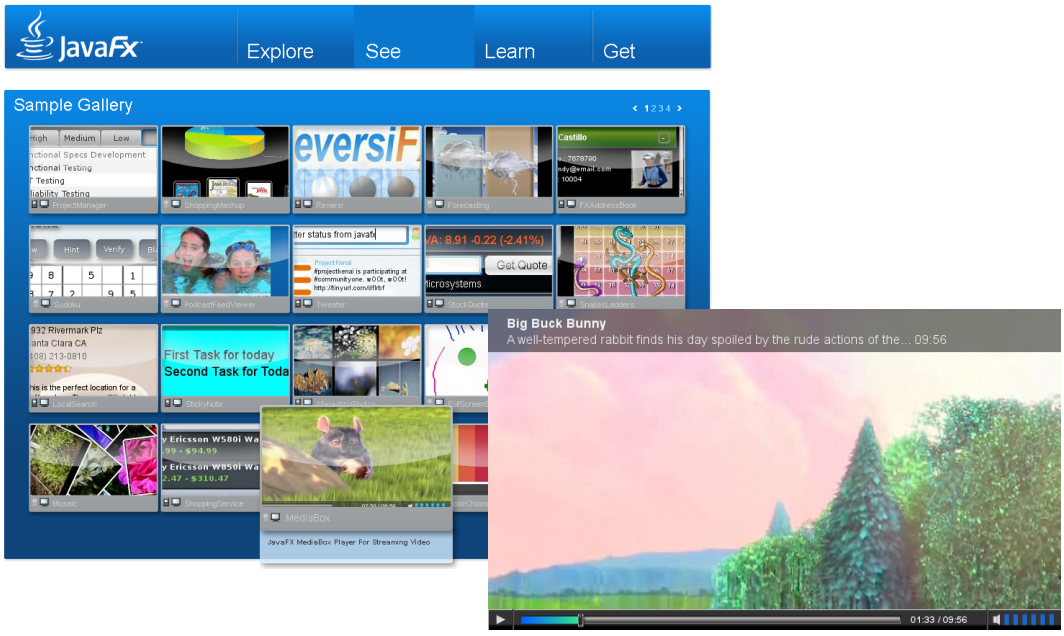

Figure 4.6: **JavaFX Examples**

Examples of a RIA JavaFX client and a customized JavaFX video player. (Source: [37])

However, in the wake of the recent emergence of RIAs and the new technologies, SUN has introduced JavaFX as a new attempt to provide a powerful, lightweight, modern day platform that can compete with the latest frameworks of other companies. JavaFX applications are written in a new language, called *JavaFX Script*, and are build on top of the Java Applet technology which they use as a container. This means that JavaFX applications only require an up to date JRE to be executed.[14]

User interfaces are defined using *JavaFX Script*, which is not declarative like XML, but resembles more of a scripting language like ActionScript. Program logic is implemented using JavaFX Script as well. JavaFX code is translated into *java bytecode* and executed on the *Java Virtual Machine (JVM)*. The translation into bytecode and common use of the JVM enables JavaFX applications to use compiled Java classes and libraries, such as Swing. Since JavaFX is based on Java and its standard API, the applications result in small file sizes, but in turn require larger runtime installations.

4.3.4 Technology Evaluation

All of the briefly introduced technologies are potentially capable to fulfil the task outlined in section 4.2.1 Requirements. They are also similar in their approaches. They offer a declarative language to create user interfaces and a high level language to program logic, and are then executed on the virtual machine of their respective runtime environment. Yet, to make an optimal choice, additional factors have to be considered. These are not

[14] http://www.javafx.com/about/at-a-glance.jsp

necessarily related to the technological capabilities of the frameworks, but are just as important.

The most crucial aspect of the player, besides the actual trajectory visualization, is a high accessibility. This means that using the player should impose no technical barriers, or these barriers should be reduced to a minimum. Since every single one of the candidate technologies relies on prior installed browser plugins in order to be used, the dissemination and availability for many different platform becomes the decisive factor.

Flash is the most widely spread plug-in[15] and has also been established for several years. This has the advantage that there are plug-ins available for all major browser and operating system with very few compatibility issues. Additional merits include a large community, freely available libraries, a vast amount of information and support documents by the community and Adobe. It is also backwards compatible with older browsers and operating systems.

Silverlight and JavaFX both suffer in this respect from their novelty. Silverlight and especially JavaFX are not very widespread. Silverlight has the advantage of the .NET framework with its functionality and freedom of .NET language choice, but still needs an extra plug-in installation. JavaFX can build on already installed JREs and the full Java API, but requires an up to date version of the JRE to work properly. Silverlight offers built-in hardware acceleration and a good documentation by Microsoft, but has compatibility issues with certain browser and operating systems.[16] The question of continued, strong support for the future plug-in development for operating systems and browsers other than Microsoft's own has also not been answered conclusively. JavaFX has some interoperability issues with different browser and does not fully support all major operating systems.[17] It is the latest and least mature framework, with quite a few issues left, which is why it is currently not very popular and has almost no relevance for creating for RIAs.[18]

Consequently, the choice is made for Flash and the Flex framework, because of its broad body of existing resources, the lowest technical access limitations of all presented technologies and large community support.

4.4 Video Encoding

The video proxy copies are transmitted over the internet via progressive HTTP download and not streamed. The merits of streaming videos are minor in this application, since there are no live feeds or potential quality of service issues. The bandwidth saving advantages of video streaming, by transmitting only the actually viewed video segments, nonetheless do apply to this scenario. Yet the savings are marginal. This is because the number of users is expected to be so low, that the potential traffic reduction doesn't

[15] Millard Brown Survey December 2009 http://www.adobe.com/products/player_census/flashplayer/tech_breakdown.ht
[16] http://www.microsoft.com/silverlight/get-started/install/default.aspx?reason=unsupportedplatform#sysreq
[17] http://www.javafx.com/faq/
[18] http://www.oreillynet.com/onjava/blog/2008/04/how_to_make_applets_and_javafx.html

outweigh the complexities involved in setting up and integrating a videostreaming environment.[19]

In order for the recorded videos to be playable by the Flash Player they have to be converted into the proper format. The current Flash Platform (10) offers two container formats, *FLV (Flash Video)* and the newer F4V. F4V is based on the *h.264* codec and intended for high-definition (HD) video. FLV is the older container format which supports the *On2 VP6* codec since Flash Player 8, and *Sorensen Spark* since Flash Player 7. All of these codecs use lossy compression, but offer good quality and compression ratios, which is essential for distributed via the web.[20]

The high speed videos were recorded with a Basler A602f camera that allows recording videos at VGA (640x480) resolution with 100 frames per second (fps) and uses the proprietary Indeo video 5.10 codec.[21] The resulting videos were slowed down to 25 fps to enable the human eye to be able to follow the bee's waggle dance movements. [4] This resulted in a video bit rate of about 6300 kbps for the original high speed videos. For the distribution of the videos via the web, the file size, and thus bitrate, has to be significantly reduced, while maintaining a good image quality. Since the VGA resolution is not considered high-definition, the On2 VP6, instead of .h264, codec is adequate for this task. The videos should not be reduced in resolution, because any further decrease in size would make it harder to recognize the videos content. The trajectory coordinates are also based on the original VGA resolution. In respect to the precondition of maintaining the original resolution, the best trade off between image quality and file size was reached at a bitrate of about 750kpbs at 25 fps. The new codec in combination with the reduced bitrate yields videos that are about 12% of their uncompressed counterparts.

The videos are transformed by BeeTube's converter component which uses MEncoder.[22] After the transformation of the high speed videos into the proper format for playback on the web, the next step is to visualize the bee dance trajectories in a layer on top of the video. This is achieved by employing the various 2D drawing components that the ActionScript 3.0 API offers. But first, the trajectories have to be transformed from their original custom format to XML in order to be easily read- and loadable for the player application. The trajectories are transformed by an XML wrapper of the BeeTube's converter component. After the successful video and trajectory format transformation, the biggest challenge still remains: The trajectory's frame exact drawing during video playback.

4.4.1 Video Format Problem

The bee dance trajectories are recorded frame by frame, with x- and y-coordinates representing the center of the bounding box and the angle of the box's direction in degrees. Hence, the drawing of the bounding box must be synchronous to the frames in the video.

[19] http://www.adobe.com/devnet/flash/learning_guide/video/part02.html An open source alternative to Adobe's proprietary and expensive solution(s) is Red5 http://www.red5.org

[20] http://www.adobe.com/devnet/flash/learning_guide/video/part01.html

[21] http://ligos.com/index.php/home/products/indeo/

[22] www.mplayerhq.hu

The chosen video codec On2 VP6 uses keyframes, but, unfortunately, the ActionScript 3.0 API does not allow direct access to a video stream. The `fl.video.VideoPlayback` class responsible for decoding the video, does not contain any methods or functions by which one can get a hold of individual frames. Neither does the actually used class, `fl.video.FLVPlayback` provide any such methods. `FLVPlayback` wraps and enhances `VideoPlayback` with network capabilities and user interactivity. It offers mostly generic, top-level functions for playback modification.[23] Also, no events could be captured that would allow an indirect access of frames, by indicating that a "keyframe event" had just been triggered. Instead, it suggests, that all decoding is computed "under the hood". This means that there are special routines and libraries within the Flash Player runtime environment, that execute the decoding of video material to ensure a smooth and efficient playback.

4.4.2 Unsynchronized Visualization

To draw the bounding box properly, one needs to synchronize it with the current frame. Unfortunately, the AS3 API does not provide any method or event that would allow for such a direct synchronization mechanism. A naive approach would be to draw the bounding box unsynchronized at the same speed that the video is being played. The speed of the video is known and fixed at 25 fps. This in turn means that one could trigger a drawing event every 40 ms (25 fps * 40 ms = 1000 ms = 1s) to draw the trajectory simultaneously with the video playback. But preliminary tests with unsynchronized drawing soon revealed, that the result is highly dependent on the speed of the computer used and its current workload. The video playback and frame drawing soon diverge out of sync, especially if the workload varies due to other processes. The Flash Player's event model requires computational time, that varies with the speed of the machine on which the application is executed. The drawing events have to be triggered and caught, followed by the actual drawing by the corresponding event handlers. All these operations take their time and this extra time accumulates as the playback of the video progresses. The result is an ever growing gap between the frame count of the video playback and the drawn trajectory. The Flash Player also tends to drop frames when it is lagging behind in playback. On different machines with varying workloads these events might cancel each other out, or aggravate the divergence.[24] In any case, it obviously renders the unsynchronized drawing approach unsuitable for the intended purpose. This is why a synchronization mechanism is indispensable for a frame exact visualization.

4.4.3 Keyframe Substitution

The FLV Video format allows to annotate meta data into videos in form of cue points. There are three different kinds of cue points, navigation, event and ActionScript. Navigation and event cue points are embedded during encoding, whereas ActionScript cue

[23] http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/
[24] http://www.kaourantin.net/2006/05/frame-rates-in-flash-player.html

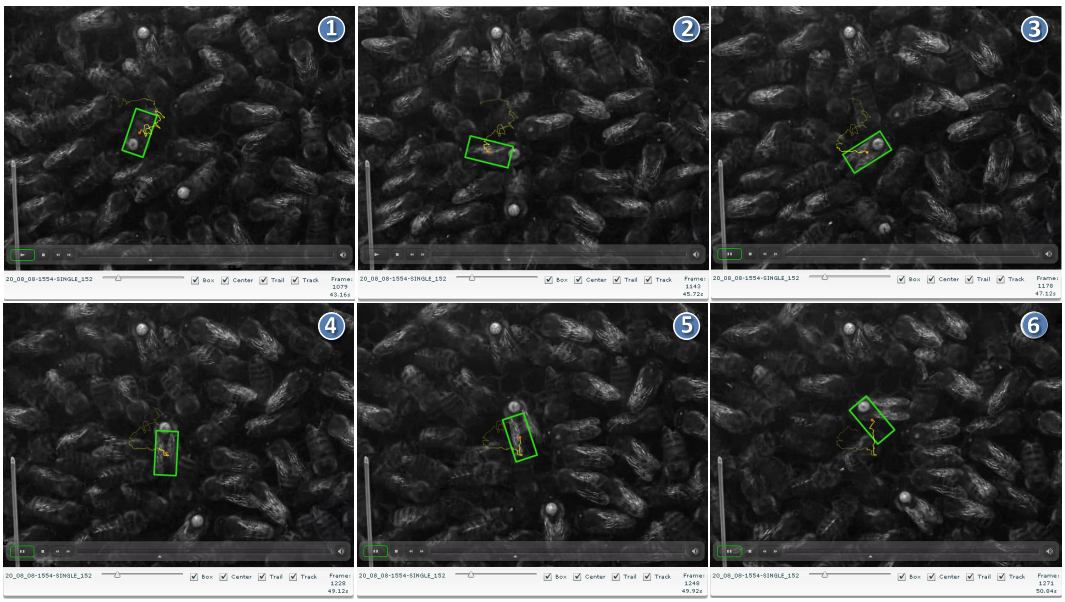

Figure 4.7: **Unsynchronized Visualization Example**

The visualization, i.e. bounding box. progressively shifts out of sync with the bee's body from images 1 through 6

points are added programmatically. The event cue points are designed to trigger events during the playback of a video that allow its synchronization with other flash elements, such as slides, and images.[25] This mechanism lends itself to the idea to use them as a substitute for individual frames by inserting as many cue points per second as there are frames in a second, i.e. 25 cue points per second. The FLV format allows such a dense amount of cue points in a FLV video. A prototypical implementation delivered a synchronized drawing of the visualization, but with a significant downside. Unfortunately, about 200ms elapse in between two subsequent drawing events, instead of the required 40ms. This results in a precise, but choppy drawing of the bounding box, since only every 5th frame (200ms/40ms = 5) is actually drawn. The reason for this is rooted in the complexity of the event model, which encompasses the encounter of an event cue point, the triggering and capturing the corresponding event and finally the drawing of the frame. ActionScript cue points also turned out to be unprecise, triggering two or more events simultaneously, while skipping others. Though event cue points have proven to be the better, more accurate choice, the large triggering delay requires a modified approach. This approach has to combine the smoothness of the unsynchronized drawing with the timely precision of the embedded event cue points.

[25] http://www.adobe.com/devnet/flash/learning_guide/video/part05.html

4.4.4 Semi Synchronized Keyframe Substition

The semi synchronized keyframe substitution combines the advantages of the two preceding approaches to achieve the exact and smooth drawing of trajectories on top of videos. First, event cue points are embedded in five frame intervals[26] to provide a programmatically accessible synchronization mechanism.[27] Figure 4.8 illustrates this synchronization mechanism.

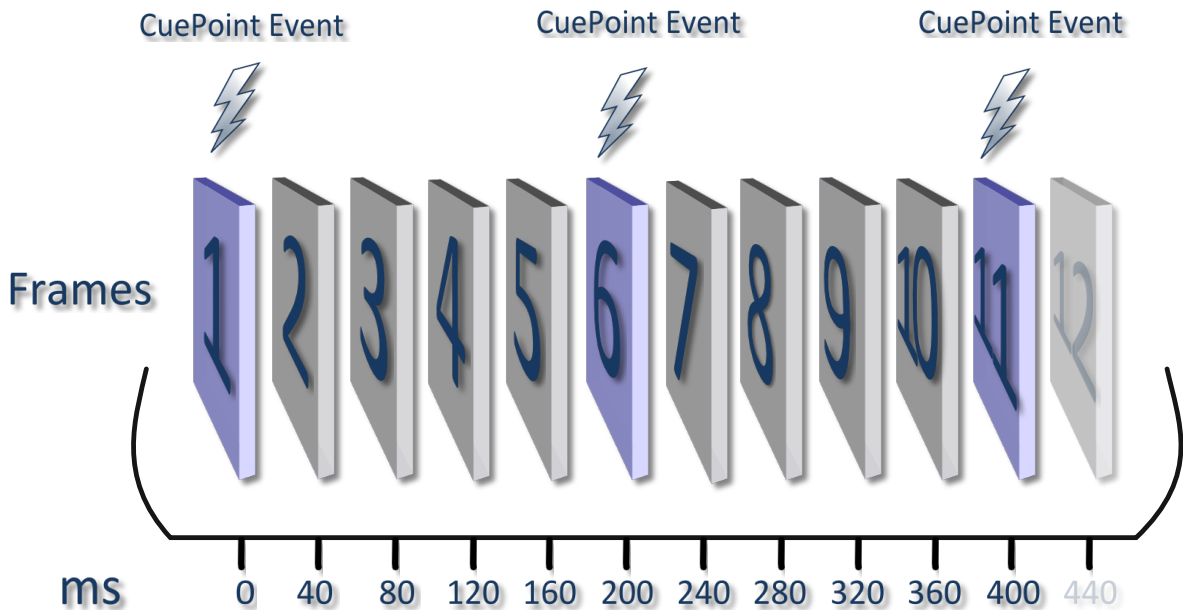

Figure 4.8: **Semi Synchronized Keyframe Substition**

A cue point event is embedded in every fifth frame of the video, to serve as a synchronization mechanism.

In between these keyframe substitutes, unsynchronized drawing is used to draw the trajectories. A timer object is generated at the start of a video and triggers an event every 40ms. This timer event increases an internal trajectory frame counter that determines which frame of the trajectory is to be drawn on top of the video. The embedded event cue points provide the exact synchronization mechanism. They are used by an internal keyframe counter to determine at each triggered cue point event, which frame of the video is currently supposed to be displayed and compares it with the internal trajectory frame counter. If they are equal, the simultaneous display of video and trajectory are in synch, otherwise the trajectory frame counter is adjusted to the cue point counter. This ensures a smooth as well as precise playback of both the video and trajectory. In case of video playback modifications, such as pause, stop and seeking, the counters

[26] The five frame interval was determined experimentally and is the direct result of aforementioned 200ms event handling delay.

[27] A modified version of the FLVtool2 is used to inject the event cue points.

are modified accordingly. The proposed approach does in fact yield the desired result. It performs robustly in respect to different computer speeds, operating systems and varying workloads. The following section will detail on the actual implementation of the approach.

4.5 Implementation

The Flex framework has a large library of user interface components. While Flash is more adapt to creating animation and media playback. Interactivity for components is added by defining event handlers in ActionScript, that respond appropriately to user input. In this case both is required, which is why components of each technology will be used. The user interface to control the dynamic showing and hiding of the bee dance trajectories will be implemented using Flex and its standard components. Since Flex's library only provides a very rudimental playback component, Flash's standard ActionScript 3.0 playback component `fl.video.FLVPlayback` will be used to display and control the actual video playback. The other elements of the player were built around it.

4.5.1 Structure

The BeeTube player plug-in essentially consists of three different, event-driven components for video playback, trajectory visualization, and user interaction. All of components are managed by their respective controllers. Figure 4.9 shows the UML diagram of the whole implementation. The controllers are located in the `de.beetube.controller` package. They are initialized by a bootstrap class, simply called `Controller`. The `Controller` also delegates the initial loading of available trajectories to the `XMLTrajectoryParser` and passes them to the `GraphicsController` for drawing.

4.5.2 Movie Controller

The `MovieController` loads the Flash video passed during creation and serves as a wrapper for the `BeeTubePlayer`, which is responsible for instantiating the `FLVPlayback` component. `BeeTubePlayer` contains the event handlers for interaction with the `FLVPlayback` controls such as play, pause, stop and seeking. It is responsible for the frame exact drawing of the trajectories and implements the synchronization mechanism developed in section 4.4.4. The `BeeTubePlayer` calls the `GraphicsController` on every frame, i.e. timer event, to draw the different shapes on top of the video. The shapes visibility and trail length can be modified using the controls underneath the video. These controls are handled by the `DisplayController`.

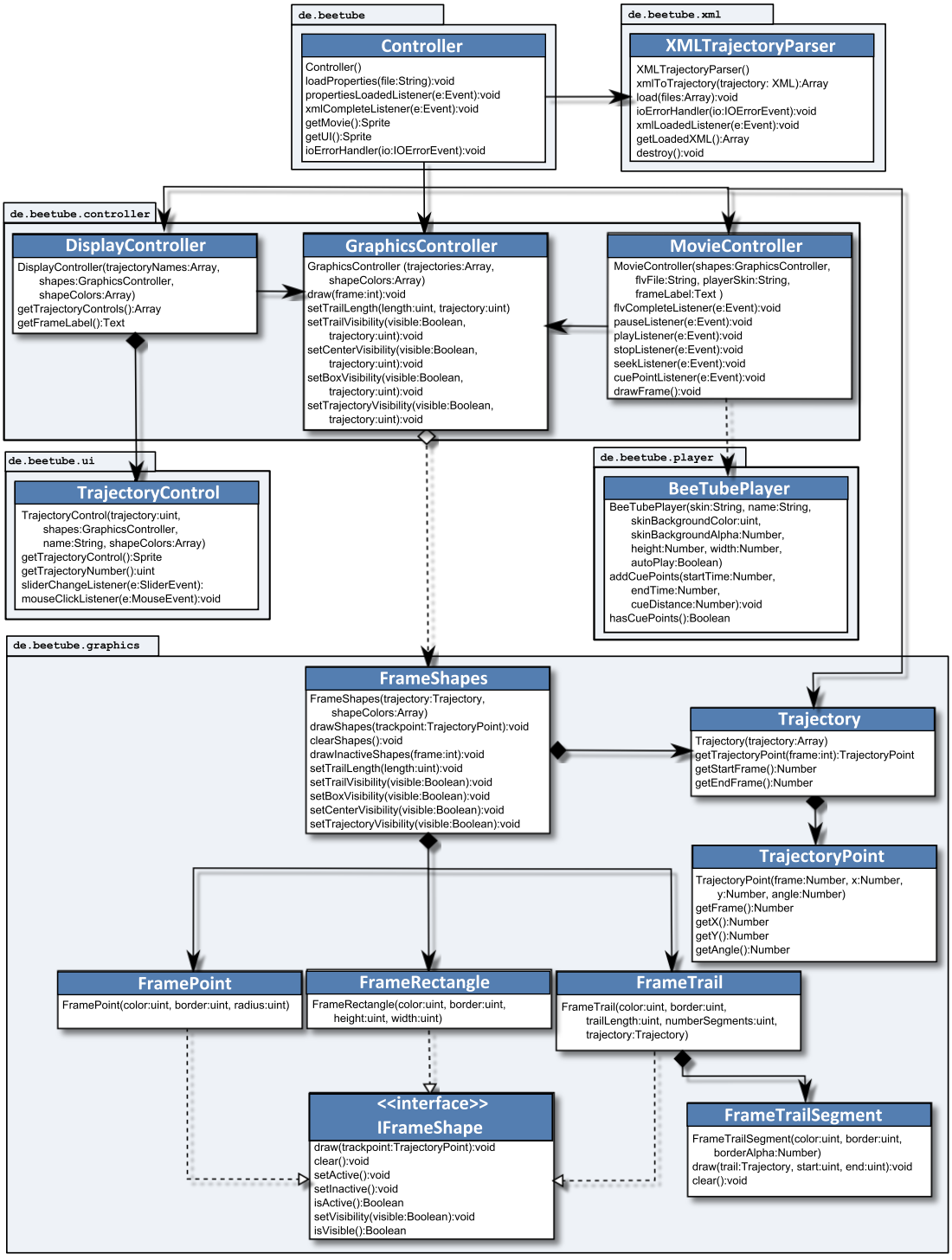

Figure 4.9: **BeeTube Player UML Class Diagram**

The structure of the new players ActionScript 3.0 classes as an UML diagram.

4.5.3 Display Controller

The `DisplayController` is responsible for creating the controls with which the user can modify the display of the trajectories. It creates as many `TrajectoryControl` instances as there trajectories to be displayed to given video. The `TrajectoryControl` also handles all the events that are triggered by a user when adjusting the display of a said trajectory. It passes the changed settings to the `GraphicsController` which is responsible for the actual drawing of the trajectory elements.

4.5.4 Graphics Controller

The `GraphicsController` manages the three different shapes of each trajectory and propagates the drawing calls from the `BeeTubePlayer` to the `FrameShapes` class which draws the respective shapes of the trajectory. `FrameShapes` also holds a representation of each trajectory through a corresponding instance of the `Trajectory` class. Once the draw method has been called and the number of the frame passed, the `FrameShapes` class then draws each trajectory's shapes. It also adjusts the visibility and trail length as passed on by the `DisplayController` to the corresponding trajectory shape.

4.6 BeeTube Player

After having covered the implementation of the BeeTube player visualization plug-in, its functionality and controls will be explained in the following section. Figure 4.10 shows a single trajectory and is an example of all the available controls. The video controls for modifying playback, such as play, pause, stop, and seek, can be found in the lower section of the video. The sound adjustment button has no functionality since the videos were recorded without an audio track. The display controls are located underneath the video and allow the adjustment of the drawn trajectories. The trail length can be changed as well as the visibility of the individual shape elements. The controls section also displays the current drawn frame and time index of the video in milliseconds.

Figure 4.11 shows the visualization of multiple trajectories. In this case only two trajectories are shown, but the player is capable of displaying up to six different trajectories.

The next and last chapter will summarize and evaluate the developments of the created system and its individual components. The visualization is the key component for BeeTube. It allows researches the unprecedented ability to display, compare and analyze multiple trajectories with each other.

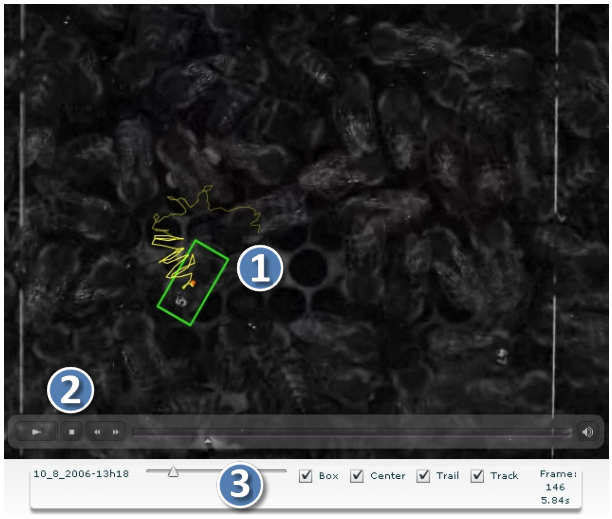

Figure 4.10: **Visualization Single Trajectory**

1) The trajectory visualization, consisting of the bounding box, center, and trail. 2) Playback controls featuring a play, pause, and stop as well as a seekbar. The sound button has no functionality. 3) Visualization controls for adjusting the trajectories display. The elements can be partially or completely deactivated, trail length modified. On the right is a frame counter and time index.

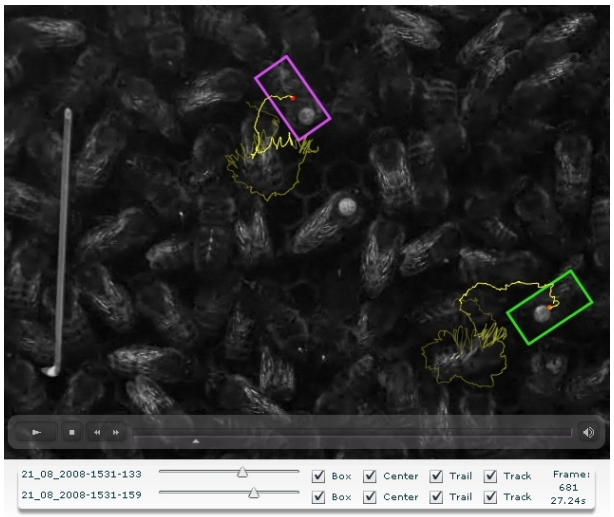

Figure 4.11: **Visualization Multiple Trajectories**

The new player visualizing multiple trajectories with complete controls for their display modification.

5 Summary

This final chapter will summarize and evaluate the results of developing the WCMS BeeTube. It will briefly recapitulate the motivation for creating the system and evaluate its worth in light of the original objective. In conclusion an outlook is given for further additions and improvements of the developed system.

5.1 Results

The thesis' objective was to create a system that would assist researchers in the management and analysis of the recorded high speed videos from their experiments. In order to accomplish this goal, two challenges had to be met. First, a system had to be conceived and implemented that would alleviate the problem of scattered digital data and allow a uniform access, management, correlation and contextualization. Second, a new visualization had to be created that would allow the easier and improved sighting and analysis of the existing data. Both of these goals, and thereby the overall objective of this thesis, could be achieved.

The web-based media content management system (MCMS) BeeTube, provides a platform that enables scientist to correlate experimental data, videos and trajectories in a stable and robust manner, that did not exist before. The ability to easily search, group and modify all of this data from practically anywhere, frees scientist from the mundane tasks of manually searching, ordering and sighting the existing data. It significantly reduces the possibility for errors in the workflow and allows them to focus on the actual analysis of the gathered data. The visualization plug-in further eliminates the manual setup for playing back videos and trajectories, while offering greater stability. On top of that, it considerably extends the old software's capabilities by being able to display multiple trajectories simultaneously.

All of these features combined, create a tool that not only dramatically improves the existing workflows, but opens up entirely new means of analysis and and research perspectives for the massive amounts of accumulated data.

In that respect, it can be said that the original goals, that were set out at the beginning of this thesis, have been achieved.

5.2 Outlook

Of course, as with most things in life, there is always room for improvement. In the case of BeeTube, both the usability of the system as well as the visualization plug-in can be improved.

The MCMS could be enhanced by implementing more AJAX functionality that allows a smoother and more informative user experience. Apart from simple navigation, in-place editing as well as improved form validation and upload functionality would benefit the application. A more sophisticated user management could also have advantages, in case personalization of information is deemed necessary.

The visualization plug-in could be augmented with further means of automatation. This includes automated analysis and evaluation of trajectory properties, as well as different modes for displaying them, such as two dimensional cartesian graphs. Another improvement could be the implementation of visual user interfaces, that allow the dynamic modification and analysis of trajectories and their constituting elements, such as the division in waggle and return phases.

In any case, all of BeeTube's components, including the visualization, have been implemented with extensibility in mind, to create a platform that is suited to meet future challenges.

Bibliography

[1] N. Gallai, J.-M. Salles, J. Settele, and B. E. Vaissière. Economic valuation of the vulnerability of world agriculture confronted with pollinator decline. *Ecological Economics*, (68):810–821, 2008.

[2] Karl von Frisch. Die tänze der bienen. *Österreichische Zoologische Zeitschrift*, (1):1–48, 1946.

[3] Adrian H. Wenner. The elusive honey bee "language" hypothesis. *Journal of Insect Behavior*, (15):839–858, 2003.

[4] Tim Landgraf and Raúl Rojas. Tracking honey bee dances from sparse optical flow fields. Technical report, Berlin, Germany, June 2007.

[5] Ashok Veeraraghavan and Rama Chellappa. Tracking bees in a hive. Technical report, Dept. of ECE, University Of Maryland, Maryland, USA, 2005.

[6] Hilda M. Ransome. *The Sacred Bee in Ancient Times and Folklore*. 2004.

[7] Jamie Ellis. Colony collapse disorder (CCD) in honey bees, 2007.

[8] I.L. Jensen, A Michelsen, and M Lindauer. On the directional indications in the round dances of honeybees. *Naturwissenschaften*, (84):452–454, 1997.

[9] W.H. Kirchner and et al. Honeybee dance communication: acoustical indication of direction in round dances. *Naturwissenschaften*, (75):629–630, 1988.

[10] R. Boch. Rassenmässige unterschiede bei den tänzen der honigbiene (Apis mellifica l.). *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 40(3):289–320, 1957.

[11] J. Clark, J. Cham, S. Bailey, E. Froehlich, P. Nahata, R. Full, and M. Cutkosky. Biomimetic design and fabrication of a hexapedal running robot. volume 4, 2001.

[12] Karl Von Frisch, Leigh Chadwick, and Thomas D. Seeley. *The Dance Language and Orientation of Bees*. Harvard University Press, 1967.

[13] Seiya Tsujiuchi, Elena Sivan-Loukianova, Daniel F. Eberl, Yasuo Kitagawa, and Kadowaki Tatsuhiko. Dynamic range compression in the honey bee auditory system toward waggle dance sounds. *PLoS ONE*, (2):234, 2007.

[14] Michelsen. Signals and flexibility in the dance communication of honeybees. *ournal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, (3):165–174, 2003.

[15] Walter M. Farina, Christoph Grüter, and Paula C. Diaz. Social learning of floral odours inside the honeybee hive. *Proceedings of the Royal Society B: Biological Sciences*, (1575):1923–1928, 2005.

[16] Mariana Gil and Rodrigo J. De Marco. Olfactory learning by means of trophallaxis in apis mellifera. *Journal of Experimental Biology*, (4):671–680, 2005.

[17] Timothy M. Judd. The waggle dance of the honey bee: Which bees following a dancer successfully acquire the information? *Journal of Insect Behavior*, (3):343–354, 1994.

[18] Rodrigo J. De Marco, Juan M. Gurevitz, and Randolf Menzel. Variability in the encoding of spatial information by dancing bees. *Journal of Experimental Biology*, (10):1635–1644, 2008.

[19] Axel Michelsen, Bent Bach Andersen, Jesper Storm, Wolfgang H. Kirchner, and Martin Lindauer. How honeybees perceive communication dances, studied by means of a mechanical model. *Behavioral Ecology and Sociobiology*, (3):143–150, 1992.

[20] Daniel Riehl. Visueller sensor für die kollisionserkennung eines biomimetischen bienenroboters, 2009.

[21] Jens Jacobsen, Tilman Schlenker, and Lisa Edwards. *Implementing a Digital Asset Management System - For animation, computer games, and web development.* Elsevier Inc., 2005.

[22] Andreas Mauthe and Peter Thomas. *Professional Content Management Systems - Handling Digital Media Assets.* Jon Wiley & Sons, Inc., 2004.

[23] Bob Boiko. *Content Management Bible.* Wiley Publishing, Inc., 2nd edition edition, 2005.

[24] Maryam Alavi and Dorothy E. Leidner. Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS Quarterly*, 25(1):107 – 136, March 2001.

[25] Arron Ferguson. *Creating Content Management Systems in Java.* Charles River Media, 2007.

[26] Christop M. Judd, Joseph Faisal Nusairat, and James Shingler. *Beginning Groovy and Grails.* Apress, 2008.

[27] Graeme Rocher and Jeff Broen. *The Definitive Guide to Grails.* Apress, New York, 2009.

[28] Wikipedia. Synchronized multimedia integration language – wikipedia. http://de.wikipedia.org/w/index.php?title=Synchronized_Multimedia_Integration_Language&oldi 2010.

[29] J. C. Preciado, M. L. Trigueros, F. Sanchez, and S. Comai. Necessity of methodologies to model rich internet applications. In *Proc. of the 7th IEEE International Workshop on Web Site Evolution*, pages 7–13, Budapest, Hungary, 2005.

[30] Colin Moock. *Essential ActionScript 3.0.* O'Reilly Media, Inc., Sebastopol, CA 95472, 2007.

[31] Tilman Walther. Konzeptionierung und implementierung eines blogsystems auf basis von wissensnetzen, 2008.

[32] MyVideo Broadband S.R.L. MyVideo. http://www.myvideo.de/.

[33] Microsoft. Showcase : The official microsoft silverlight site. http://silverlight.net/showcase/, 2009.

[34] Torque.TV. http://www.torque.tv/.

[35] Andreas Kuehnel. *Visual C# 2008. Das umfassende Handbuch.* Galileo Computing, 2008.

[36] Arsen Gogeshvili. AVL tree applet. http://webpages.ull.es/users/jriera/Docencia/AVL/AVL%20tr 2002.

[37] Sun Microsystems. JavaFX example code and project gallery. http://javafx.com/samples/, 2009.