TERMES: Termite tracking in collaboration with Harvard University

Nikolaj Aaes, Niklas Schalck Johansson, and Hildur Uffe Flemberg
{niaa, nsjo, hufl}@itu.dk

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S

Abstract. 10 linjer
problem stilling
evaluation
resultat

• State the problem - There is no existing software that allows tracking of termites using the HP plotter provided by harvard as well as being able to extract relevant statistics.
• Say why it's an interesting problem - It will enable biologists to make better analyses based on more empricial data
• Say what your solution achieves - A way to control the plotter that tracks an ant/termite in real time and collects data that can be extracted as statistics
• Say what follows from your solution - While this is just a prototype (in a lab enviroment) it is a step towards making new and better field equipment for biologists along wiht corresponding software.

Keywords: Computer Vision, Image Processing, Termite Tracking, Biology, Computer Science.

Table of Contents

1 Introduction

This report is written at the IT University of Copenhagen (ITU) in the fall term of 2013 as a project supervised by Kasper Støy from ITU and with additional guidance from Kirstin Petersen from Harvard University. The project was developed from 1. september to 16. december 2013 and was worth 15 ETCS points. The report is addressed to people interested in tracking ants and/or termites using image analysis.

The Self-Organizing System Research Lab at Harvard University has created a autonomous robot for tracking African termites in a lab environment. Working with and tracking these termites is cumbersome as they only thrive in environments that resemble their native environment and only when they are together with other termites. Therefore automatic tracking of specific termites is necessary, as tracking up until now has been done manually. Hardware has been created to handle this, however it lacks proper software support.

This project aims to develop software that is able to track both ants and termites in a natural environment using a Hewlett-Packard XY-plotter provided by Harvard University. While it is already possible to communicate with the plotter using the program called Termite [1], there is a need for software that integrates tracking, plotter communication, statistics and a graphical user interface. The end product is to be used by biologists in the field and this software will enable them to track ants/termites with more precision and with better data being produced.

A team at Harvard University previously experimented with using the plotter to track a singel ant/termite on a white background with some success. While the details of this effort were unknown to us we decided to start this project from scracth.

Since tracking of ants/termites is very hard to test systematically we have chosen to evaluate it by describing in what instances we expect the tracking to work as intended and in what instances we expect it not to.

TODO:
Skriv noget om outcome.
A summary (a “map”) of how the paper is organized.

2 Project Proposal

The purpose of this project is to develop software, such that the provided hardware can be used to track these ants/termites and analyze the output, as well as providing basic user input to the robot tracker.

There are three parts in this project:

1. Tracking termites using a low resolution camera. In order to track a termite, it is necessary to be able to determine its position and move the camera to its new position using image analysis on the camera output.
2. Interact with the tracking device and update the camera's position with the result from the tracking software.
3. Design and develop a user interface that can be used by biologists to retrieve statistical data from the tracking.

3 Background

This project was undertaken as a substitute for the "Global Software Development" course at ITU and therefore has additional emphasis on the international collaboration and developement process. Section 7 Process will describe the tools used as well as evaluate the overall process.

Our councillor at Harvard wanted the tracking and the interfacing with the tracking device to be implemented in C++ to make it easier for further developement. The graphical user interface was not restricted when choosing programming langauge and we ended up using C# and Windows Forms to implement it.

As mentioned in the project proposal the project was divided into three parts. The first part of the project was to be able to track termites. Since african termites are hard to come by and transport we settled for giant ants (Camponotus Ligniperdus), which are the largest ants found in Denmark [3], instead. While these are not as big as the termites, they behave in a similar way and the solution should be able to easily adapt to the termites. In the first part we started by implementing the tracking on a video. We received a video from Harvard of a single ant running around in a petri dish with a white background. The ant itself was painted red and green to make tracking easier. This was the simplest setup we could think of and acted as a good starting point. By recommendation we decided to use the OpenCV [2] framework to implement our tracking. We will return to this framework in Section 4.2 Framework.

The second part of the project interacting with our tracking device and conform the tracking code to work with the device. We received a Hewlett-Packard 7046a XY-plotter by mail from our councillor at Harvard. Additionally we received a small ant farm, containing one queen and seven worker ants (no soldier ants), via mail and we bought some brushes and acryllic paint for the ants. The ants did not proliferate during the project. While we discussed using infrared paint or fluorescend paint, both these ideas were discarded because both types of paint contains compounds that the termites and ants really like which makes them eat the painted ant. Along with the plotter we also received a small circuit board that plugged into the serial port of the plotter and had a USB cabel for

us to communicate with it. Lastly the package contained a collection of petri dishes for us to use.

In this part we had to paint the ants with the paint we bought and we want to give some friendly advice to anyone who wants to do any projects involving ants. If you put them 2-4 minutes in the freezer they become very docile and much easier to paint without harming the ants. When they are taken out they act completely normal after 2-3 minutes. The ants also had a tendency to crawl up the sides of the petri dish. We were recommended to try putting teflon tape (should be really slippery) on the sides without any luck. We were also recommended to try mineral oil but we were unable to get any. Our solution was to place a small petri dish in a bigger petri dish and fill the gap with water. The ants would quickly discover that there was water surronding them which prevented them from escaping the small petri dish.

The third part of the project involved constructing the graphical user interface, hooking it up to the tracking code and extracting statistics.

TODO: beskriv third part. Skriv at myrer godt kan svømme! plus de problemer vi havde med at holde myrene inden for petri skålen.

3.1 Assumptions

TODO

3.2 Scope

While the project lays the foundation for a fully useable field solution we have chosen to narrow our scope to the essential functionality. This also means that the project was designed and tested in a controlled lab environment where factors such as lighting, wind etc. are constant.

Because actual termites were out our reach and that our councillor at Harvard recommended it, we did all of our testing with normal ants instead of termites. We are confident that only minor adjustments are necesarry to make the solution work with termites since the main difference is the size.

Painting the ants in different colors can have some impact in the quality of the tracking. As a rule of thumb it is easier to track colors that contrast the background the most. In this project we chose to paint the ants white and bright green. We did not try any other colors because we found these to have a high degree of contrast which were sufficient.

As the HP 7046a plotter was the only hardware available to us we have restricted the project to only include this type of plotter. One could potentially

gain better results with a plotter with a finer granularity in coordinates and smoother movement. The cameras supplied with the plotter was also the only cameras available to us and we have therefore also restricted the project to only include these cameras. While one could use higher resolution cameras to obtain a better result, the weight of the camera is also important for the smoothness of the movement of the plotter and must be carefully considered.

While it could be intereseting to track several ants we have chosen to focus on tracking a single ant in this project due to time restrictions. However, we do not believe that there are any technical barriers against this and would simply require more time to implement.

TODO:
fortæl om hvordan vi IKKE tester

3.3 Requirements

Mandatory requirements The mandatory requirements of the project are listed below and must be fulfilled for the project to be considered a success.

1. Tracking of ants and/or termites using a low resolution camera using C/C++.
2. The ability to adjust the tracking parameteres before and during the tracking.
3. Moving the tracking hardware in corresponding to the tracking.
4. A graphical user interface containing two modes:
5. - Calibration mode. Must contain a direct feed from the lower camera, a proccessed feed and sliders to adjust the processing.
6. - Tracking mode. Must contain a direct feed from the lower camera, a direct feed from the overhead camera and statistics.
7. The ability to collect the following statistics:
8. - The route of the ant/termite over time.
9. - Heatmap of where in the petri dish the ant stay.
10. The ability to save the collected statistics.

Optional requirements The optional requirements are the so called "nice-to-have requirements". They are not mandatory for the project but features that could be desirable to implement if the time and resources permits it.

1. An additional third mode in the graphical user interface Bias mode: adds the ability to select a certain point which the plotter will try to lure the termites towards using food or pheromones.
2. The ability to choose certain areas which should be avoided during the tracking.
3. Collection of these additional statistics:
4. - Average speed of the ant/termite.

5. - The amount of ants/termites meet during the tracking. A way to adjust what defines a meeting.
6. - The amount of time between each meeting. A way to adjust when a meeting starts and ends.
7. - The duration of each meeting.
8. - The area of the petri dish covered by the ant/termite. A way to adjust how much area is covered by an ant/termite when stationary (also known as "headsize").
9. - How much area is covered over time.
10. - Mean free path (the amount of time between each "stay").

4 Tracking

This chapter is divided into three subparts. First we will introduce the reader to different image processesing techniques, and their uses. Following this we will introduce the reader to a framework that supports these techniques, and finish off by describing the implemened solution and the design choices.

4.1 Theory

This section will provide information about five common image processing techniques; thresholding, dilating and eroding, contrast, image segmentation and background filtering. The description of each technique will be backed up by examples. We will also use the *ternary if* statement notation in this section which looks like the one shown in Equation 1.

$$value = Boolean\text{-}Condition?True\text{-}Evaluation : False\text{-}Evaluation \quad (1)$$

It works just like you would expect from many programming languages. It is also known as an inline *if-statement*.

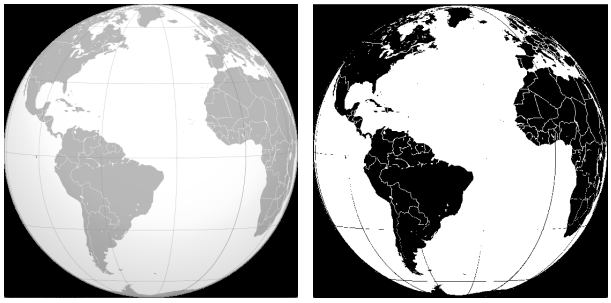
Thresholding

Thresholding is an image processing technique used to make a final decision about each pixel in an image. Either a pixel value is one that we are interested in or it is not. This is usually done by assigning a specific pixel value to the pixel we want, and another to those that we do not want. In general we compare the ith pixel of the source image, *src*, to the threshold value, T, and saves the result in a destination image, *dst*. For instance, to create a binary image where we are interested in all pixels above the threshold T, the equation would look like the one shown in Equation 2,

$$dst_i = src_i \geq T?255 : 0 \quad (2)$$

Most threshold operations are applied on grayscale images, where all pixel values range between 0 (black) and 255 (white). Sometimes these values are

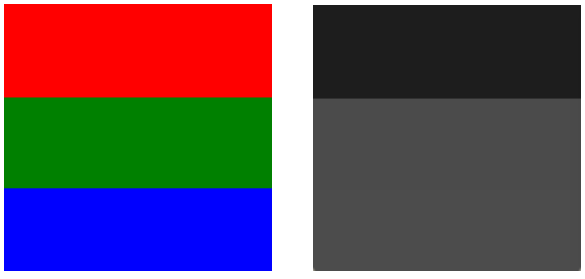
normalized to range between 0 and 1 instead. However for the rest of this report we will assume that grayscale images use the former convention. Soon we will argue how thresholding can be expanded to also cover thresholding an RGB image. Figure 1 show an example of applying threshold to a grayscale image.



(a) Grayscale image *(b) Thresholded image*

Fig. 1: An example of applying a threshold on a grayscale image. This example use the threshold value $T = 200$

Now what happens if what you are interested in is a color? To use standard thresholding we first need to convert it to a grayscale image. However doing so might result in an image that have lost important information as can be seen in figure 2. Unless you want to find the red color, you have no way of differentiating the blue and green color.



(a) RGB image *(b) Grayscale image*

Fig. 2: Example of grayscaling a color image

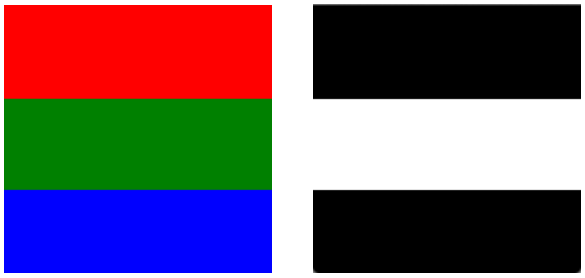
A step in the right direction would be to define the threshold value T as a scalar consisting of three values - one for each color channel. We will denote this threshold scalar as S and define it as shown in Equation 3.

$$S = \begin{pmatrix} S_R \\ S_G \\ S_B \end{pmatrix} \tag{3}$$

To apply it to an RGB image we need to change Equation 2 to the one shown in Equation 4.

$$dst_i = src_{iR} \geq S_R \wedge src_{iG} \geq S_G \wedge src_{iB} \geq S_B ? 255 : 0 \tag{4}$$

Trying it out makes it much easier to differentiate between colors. In Figure 3 a threshold attemp using this method directly on the RGB image is shown.



(a) RGB image *(b) Thresholded image*

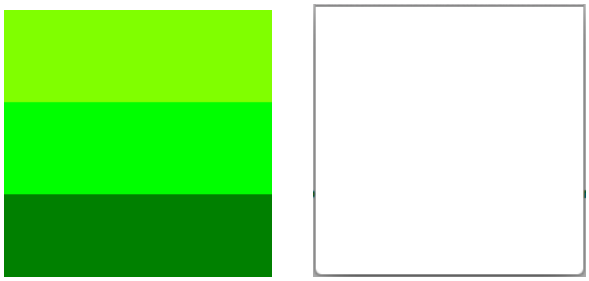
Fig. 3: Example of thresholding a color image with Equation 4 using the scalar S=(0,100,0)

However one issue remains - what if you want to find a color among similar colors? The problem is illustrated in Figure 4 using the same scalar as in Figure 3

The solution is to specify a *range* of acceptable values instead of just a threshold. We will specify two scalars; S Upper, *SU*, and S Lower, *SL*.

$$SL = \begin{pmatrix} SL_R \\ SL_G \\ SL_B \end{pmatrix} \tag{5}$$

$$SU = \begin{pmatrix} SU_R \\ SU_G \\ SU_B \end{pmatrix} \tag{6}$$



(a) RGB image *(b) Thresholded image*

Fig. 4: Example of thresholding a color image with Equation 4 using the scalar S=(0,100,0). Unlike before the result is unsatisfactory.

We will use the definitions in Equations 5 and 6 to update Equation 4. The changes can be seen in Equation 7.

$$dst_i = SU_R \geq src_{iR} \geq SL_R \wedge SU_G \geq src_{iG} \geq SL_G \wedge SU_B \geq src_{iB} \geq SL_B ? 255 : 0 \tag{7}$$

Using a range to specify a color instead will yield a much more satisfactory result as shown in Figure 5.

(a) RGB image *(b) Thresholded image*

Fig. 5: Example of thresholding a color image with Equation 7 using the range SL=(0,100,0) and SU=(1,150,10). We have sucessfully located the dark green color area.

Dilating and Eroding

Forklar at vi har en kernel, B, med størrelsen s, der bliver anvendt på hver pixel, p, i et billede A. Angiv at dilating finder local optima, og at eroding finder local minima. Angiv at de her metoder bliver brugt EFTER thresholding for at "rense" billedet.

(a) Grayscale image *(b) Eroded image* *(c) Dilated image*

Fig. 6: Eroding and dilating an image. It is clear that eroding expands dark areas and dilate expands bright areas. This example uses a 7x7 kernel.

Contrast
Describe how α and β is applied to each pixel in an image.

Image Segmentation
Describe how clustering algorithms is applied to create "pixel" blobs.

Background Filtering
Describe how computing the absolute difference in pixel values between two image filters out the background.

4.2 Framework

Describe OpenCV. Explain why it is preferable over other frameworks.

4.3 Realization

Describe the solution and design choices. E.g. why do we use thresholding but not Background Filtering. Something about blob detection?
Nævn Vi har modtaget video først for at kunne starte før vi fik plotteren.

5 Integration with camera

NOTE: Husk at nævn det ikke er super hensigtsmæssigt at maskineriet hakker så meget som det gør.

5.1 Practical information

5.2 Communication with the plotter

Hvilken plotter er det og hvilket udstyr sidder på den Hvordan er den sat til computeren hvad vil vi gerne have den til (nævn at det ville have været nice at have det cross platform) hvordan gør vi dette hvilke "services" ender vi med at udstille til resten af programmet (flyt til koordinat, current coordinate?) Hvordan er performance? kan videoen køre i real time? hvis nej hvad betyder det for os? skal vi skippe frames?

* 0x01 = send coordinate to plotter * 0x01 + 2 bytes for x + 2 bytes for y (up to 10 bits) up to 0x03FF (y probably only up to 01F0) * 0x02 in response = sucesss * 0xFE in response = failure

5.3 Moving the camera

Now that we know how to move the plotter, how do we move the camera when the ant moves? what about the volume of the movement sound? what about the speed of the movement?

6 Graphical user interface

What was the reqs for the GUI? How did we want it to look? What tasks do we expect the users to do (normal work flow)? Are we satisfied with the GUI (eval)? JNI skulle være cross platform men fungerede ikke med OpenCV. (nævn at det ville have været nice at have det cross platform) Vi har ikke noget der finder myren til at starte med. Hvordan kunne dette gøres?

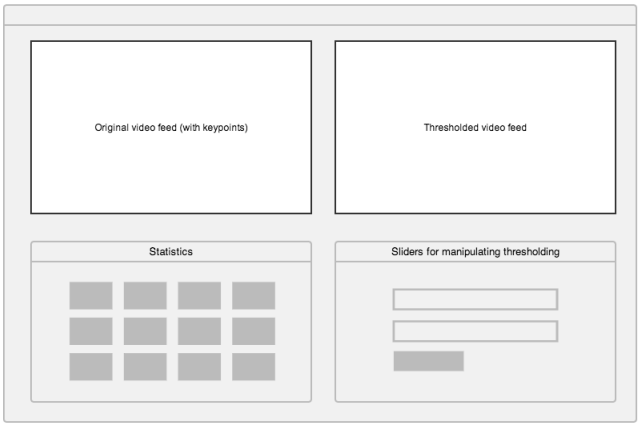


Fig. 7: Initial GUI draft

6.1 Statistics

What statistics do we extract? How do the users do this? How are they produced? Can they be improved (future work)?

7 Process

The developement process in this project required more planning than usual because of the long distance collaboration with Harvard University. This meant that we, at an early stage, sat down and agreed on a time plan, a preliminary table of contents for the report and which tools we would use to communicate and track our progress.

For communication we used regular email, skype and the Github [4] wiki and issue tracking system. Email allowed us to easily communicate with our councillor at Harvard even when either part was too busy for a skype meeting. It also helped to express the more formal questions about the project and the email correspondance was a good reference throughout the developement of the solution and writing of this report. We made sure to update our councillors weekly via email to inform them on our progress and problems.

Skype enabled us to have face-to-face meetings several times in the project. A summary of each meeting was created on the wiki both for own sake but also to make it easy for us, our councillor at ITU and our councillor at Harvard to track our agreements and progress. The skype meetings were more sparse than email correspondance because it was harder to agree on a time to hold them because of the time difference, and because they took more time which reduced the time spent coding. Since the Danish culture is a high-context culture (defined by Halls [5]) skype meetings was not greatly advantageous compared to pure email correspondance but it was nice to be able to discuss certain aspects and when a bundle of questions presented themselves at the same time it would save time to ask them in bulk. The fact that both we and our councillor at Harvard shared the same culture (specifically work culture) made the collaboration quite a lot smoother. This made the expectations of work hours, work load and final product align more easily along with avoiding any nasty confrontations as a result of suprising holdidays or customs etc.

The issue tracking system on Github were something we started out using but quickly abandoned. Since most of our work was done in the same location with all group members present the issue tracking systems primary function was to help our councillors track our progress easily. We quickly discovered that we were using a lot of time planning these issues and that our councillors were satisfied with the weekly email update we provided.

In general we were satisfied with the tools we used. The mix of verbal and written communication provided us with good channels to fulfill our needs and

we feel that all of our questions were answered in a satisfactory way. Additionally we feel that our tools have helped us to inform our councillors of our progress and problems along the process in a satisfactory way as well. The fact that email is an asynchronus form of communication also helped us deal with the time difference between Copenhagen and Boston. We did experience any problems with this.

The learning outcome of this project regarding the process has been mostly about when to change direction when one encounters a technical roadblock. When we ecountered problems with using JNI and communication with the circuitboard we should have been quicker to search for alternate solutions or abandon the cross platform idea. This could have given us more time to implement more features in the final solution instead of struggling with technical problems.

8 Reflection

TODO

9 Evaluation

vis hvornår det virker og hvornår det ikke virker. Se at det virker når vi regnede med at det virkede.

Hvad tester vi? Hvordan tester vi det? Virkede det efter hensigten? Hvorfor/hvorfor ikke? Er der nok testing? Hvordan kan man lave mere testing? Er der andre måder vi kunne have testet på? (fordele og ulemper ved det)

10 Threats to validity

10.1 Threats to internal validity

While we believe our results to be purely causal there are always some threats to the internal validity that questions which factor could have an impact on the results. We describe the important ones here:

Behavior of the ants This project had a development period spanning over half a year. In this time the season changed and so did the temperature. The ants could have altered behavioral patterns, movement patterns and speed during the project essentially making the ants we tested on at the last part of the project, completely different from the ants we started out with.

Painting We painted our first ant relatively early in the development process and the ants could have had some reaction to the paint. We did use acryllic paint as recommended by our councillor, and we did not observe any noticeable changes but it could still be present.

Hardware Getting the camera in the same position every time is not something that can be done with 100% accuray every time. The small margin of error during positioning might have had a miniscule impact on the each tracking session.

Lights While we did try to do our testing at the same time of day with the same types of light each time it is very hard to keep completely controlled. Since light does have a significant impact on the tracking this threat to validity might the greatest but it is also something that can be hard to control.

10.2 Threats to external validity

The outcome of this project is highly dependant on the environment around the hardware and the ants/termites. This makes it very hard to replicate the results even with slight changes in things like light or camera resolution. This if course has the reprocussion that while the software works in theory and in a controlled environment it is unlikely to work in the natural habitat of any ants or termites. The following list contains the environment factors we consider to be important for the results:

Light Different intensities of light may affect where the ants/termites wants to move towards. Additionally the tracking relies on thresholded images which can be distorted by different levels of light which makes the tracking more inaccurate and more likely not to find the ant/termite.

Reflections Keeping a lid on the petri dish or filling the outer rim of the petri dish with water, like described in section 3 Background, generates a lot of reflections. This is somewhat tied to the light factor and if the lights are positioned in such a way that the camera catches the reflections it can distort the thresholded image used for tracking.

Camera resolution The camera used had a faily low resolution of 640 times 480. While it could be nice to have a larger resolution, the low resolution enables us to process each frame fairly quickly. The threat to validity is mostly a performance one. If a better camera is chosen one might need to skip more frames or upgrade the connected PC since each frame will have a longer processing time.

Camera weight The camera attached to the plotter during our development and testing was very light. This made the plotter stutte less when it moves thereby generate less sound. Both sudden movements and loud noises can have an effect on the movements of the ants and if the camera is exchanged for a heavier camera it could impact the movement of the ants/termites.

Movement of the ants In our testing phase we observed that whenever we placed an ant in our petri dish the first thing it would do would be trying to escape. It would immediatly seek the edges of the dish regardsless of whether we had the rim filled with water or not. If the plotter was bigger or if the species of ants was different this might not have been the case.

Temperature, wind and humidity Ants, like any other animals, react to temperature, wind and humidity. If any of these factors are changed it could impact how the ants move and how they behave which in turn can impact the tracking.

Ants and termites Ants are not termites. This fact is something of a gap in the testing of the project. We did not have the opportuniy to test the software with real african termites and therefore settled for ants. While our councillor assured us that they behave in a very similar way we do not have any data supporting that this project will work with real termites, only a strong indication that it should.

All in all we are quite certain that the outcome of this project can be quite hard to replicate. This makes the project more useful as a theoretic base for further development than an actual tool to take into the field in its current state.

11 Related Work

TODO

12 Future Work

Because time was a limitid resource for our team there were features wi did not implement and ideas for additional functionality that could have been included. This section describes a short list of the possible extensions to the project we might have worked towrads given more time.

The first task would be to implement the optional requirements listed in section 3.3 Requirements. ER DER NOGEN VANSKELIGEHEDER VED DET ELLER ER DET LIGE UD AF LANDEVEJEN?

When all these requirements were fulfilled it could be beneficial to support multi types of XY-plotters. The plotter we used was a little dated and to be able to switch to any XY plotter could be very practical. Of course this would also imply testing with an array of different plotters to see whether or not the same result could be produced. To support mulitple plotters one could implement plotter control as a service library to make it easy for other applications to control the plotter. This could expose an communication interface so it would

be easy to switch the both the hardware and software of the plotter.

One of things we really wanted to do, but did not have the oppertunity to, was to test with real termites. The solution was design to easily be able to switch between different insects of different sizes and to test with real termites would be a strong indicator of how well this switch would work.

To help future development the implementation fo a developer console or log could be helpful. Developers could be able to manipulate the movement of the plotter through the terminal or interact with the tracking parameter. The log could save all the raw tracking coordinates to expose what data is recorded by the tracking and how the statistical data was created.

The graphical user interface could be enhanced by adding a color picker in addition to the RGB sliders that are currently present. Addtionally a third "Bias" mode could be added where an incentive (either food or pheromones) were attached to the camera, a specific point was selected via the GUI and the ant/termite would be led towards that point. This would open the possibility for even more statistics to be collected.

13 Conclusion

TODO

14 Defition of terms

TODO

References

[1] Termite: a simple RS232 terminal - `http://www.compuphase.com/software_termite.htm`
[2] OpenCV - `http://opencv.org/`
[3] Kæmpemyrer (Campotonus ligniperdus) - `http://www.fugleognatur.dk/artsbeskrivelse.asp?ArtsID=8063`
[4] Aaes/TermiteTracker - `https://github.com/Aaes/TermiteTracker`
[5] J. & G. Olson, Surprises in Remote Software Development Teams from lectures - `http://dl.acm.org/citation.cfm?id=966804`

A First Appendix