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 HP xy-plotter provided by Harvard University. While it is already possible to communicate with the plotter using the program called "Termite"(ref), there is a need for software that integrates tracking, plotter communication, statistics and a GUI. 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 the IT University of Copenhagen and therefore has additional emphasis on the international collaboration and developement process. Section XX 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 regular ants 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 (INDSÆT REF) framework to implement our tracking. We will return to this framework in Section XX 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 (INDSÆT MODEL) XY-plotter by mail from our councillor at Harvard. Additionally we received a small ant farm via mail and we bought some brushes and acryllic paint for the ants. 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, indsæt refs.

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 INDSÆT MODEL 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. A graphical user interface containing two modes:
2. - Calibration mode. Must contain a direct feed from the lower camera, a proccessed feed and sliders to adjust the processing.
3. - Tracking mode. Must contain a direct feed from the lower camera, a direct feed from the overhead camera and statistics.
4. The ability to save the collected statistics.
5. Teh ability to adjust the tracking parameteres before and during the tracking.

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.

Data + billedbehandling + tracking i c/c++ tracking termit/myre markeret med farver (mellem andre termitter) flytte plotteren sammen med tracking

Statestik: 1. Position i skaalen over tid (marker dens rute). + indstilling for hvor lang tid tilbage ruten skal vises. 1. Gennemsnitshastighed/tid (siden start), +indstilling for running average. 2. Heatmap over hvor myren opholder sig mest i skaalen, (groft firkantet grid er fint til at starte med). 3. Heatmap over hvor myren holder pauser i skaalen, +indstilling for hvor lang en 'pause' er, hvor lav den gennemsnitshastighed skal vaere. 3. Tael andre myrer den moeder/tid. +Indstilling for afstand af et 'moede'. 4. Hvor lang tid der gaar imellem den moeder andre/tid (siden start). 5. Hvor meget tid den pauser naer andre myrer/tid. +Indstilling for afstand+pause laengde 6. Hvor meget af skaalen den har undersoegt (Additivt) over tid. +Indstilling for radius af hovedet, (hvor langt fra center af hovedet den kan undersoege af gangen) 7. Hvor meget nyt areal den daekker hvert interval over tid. +instilling for interval: f.eks. 30s/1min/2min 8. Mean free path. (maal hvor lang tid der gaar mellem pauser. Pauserne defineres af lav gennemsnitshastighed)

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*.

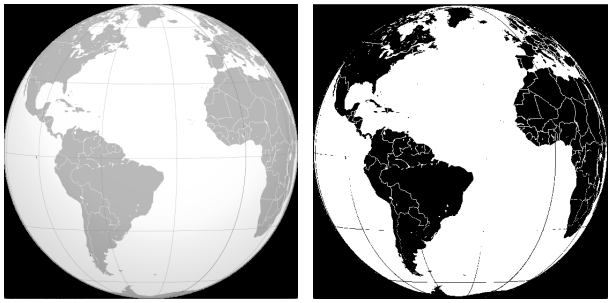
Thresholding

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,

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

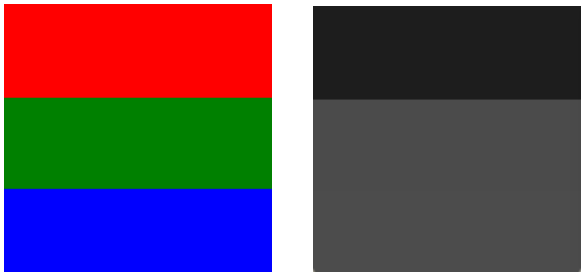
Most threshold operations are applied on grayscale images, where all pixel values range between 0 (black) and 255 (white). Sometimes these values are normalized to range between 0 and 1 instead. However for the rest of this report we will assume that grayscale images use the former convention. Soon we 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.

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) Grayscale image *(b) Thresholded image*

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



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

Fig. 2: Example of grayscaling a color image

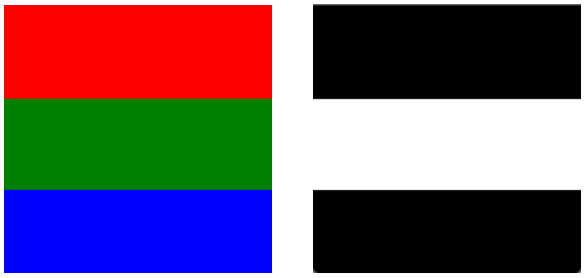
A step in the right direction would be to define the threshold value T as a scalar consisting of three values - one for each color channel. We will denote this threshold scalar as S and define it as shown in Equation 3.

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

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*

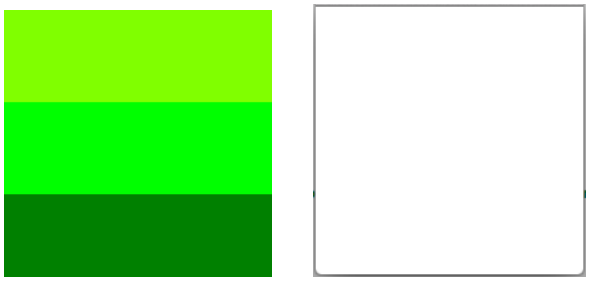
Fig. 3: Example of thresholding a color image with Equation 4 using the scalar S=(0,100,0)

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.

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 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)?

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

Hvad var process planen? Hvordan kommunikerede vi med Harvard + vejleder? Hvordan gik det (eval)? Havde vi nogen problemer med det? (tidsforskel, kultur forskel) Hvilke værktøjer brugte vi? Virkede de efter hensigten? Hvilke kommunikations kanaler brugte vi? Virkede de efter hensigten? ville vi gerne have haft mere eller mindre kommunikation? Ville vi gøre noget anderledes hvis vi skulle gøre det igen/hvad har vi lært?

Vi har modtaget video først for at kunne starte før vi fik plotteren. Vi har sagt i god tid når vi har været på ferie og haft eksamen.

Halls has defined cultures as being either high-context or low-context, which states the degree to which their culture is expressed. As an example, a low-context culture will have a lot of implicit actions, where a high-context culture will act more explicitly. - Olson, J. & G. Surprises in Remote Software Development Teams. p. 58

J. & G. Olson, Surprises in Remote Software Development Teams from lectures - http://www.itu.dk/people/nh/BAAAP/2011/Olson,%20Culture%20surprises%20in%20Remote%20Software%20Development%20Teams.pdf

http://delivery.acm.org/10.1145/970000/966804/olson.pdf?ip=130.226.133.117&id=966804&acc=OPEN&key=F9B7F4BB951339C12170E67C436BC2820AE1841B53DC2B42D8707E9E1A8C0378363661&CFTOKEN=86967028&__acm__=1384268141_38f7c1e2b1f339624b7349b5d33a1fc1

http://dl.acm.org/citation.cfm?id=966804

8 Reflection

TODO

Known bug:
når man lukker java GUI'en kan den godt crashe (det har ingen reel effekt)

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

Høj lyd / hastighed kan skræmme myrerne til at flytte sig hurtigere/mere. TODO

10.1 Threats to internal validity

Internal Validity is concerned with confirming that the correlation between the treatment and the outcome is indeed casual, and not accidental, or caused by some third variable that has not been observed. For example we may discover that all programmers in C were faster than programmers in Java, but forget that all the programmers in Java took the experiment very late at night, when they were tired.

10.2 Threats to external validity

External validity discusses how far the results are generalizable, or in other words how representative the sample of subjects and the circumstances of the experiment were, to be able to draw general conclusion. Do you expect the same results to be confirmed in somewhat modified conditions?

11 Related Work

TODO

12 Future Work

More statistics Bias mode Support for multiple plotters Plotter control as a library Tests with real termites Developer terminal/log (print was is sent to the plotter and the answer + the hard tracking coordinate data)

13 Conclusion

TODO

14 Defition of terms

TODO

References

[1] TestTitle - TestPublisher - 1999

A First Appendix