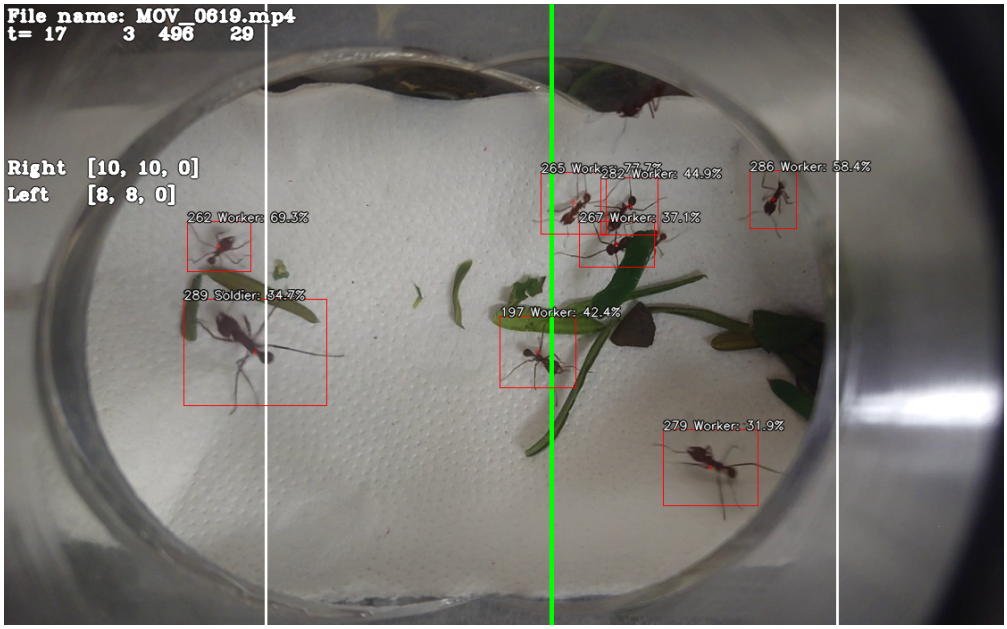
CHALMERS



Förbättring av ett program för att räkna bladskärarmyror

Examensarbete inom Högskoleingenjörsprogrammet i Datateknik

Oskar Hammargren
Robert Sjöberg

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2021

Uppsats för kandidatexamen 2021:NN

Förbättring av ett program för att räkna bladskärarmyror

Oskar Hammargren
Robert Sjöberg

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för data- och informationsteknik
Chalmers Tekniska Högskola
Göteborgs Universitet
Göteborg, Sverige 2021

Förbättring av ett program för att räkna bladskärarmyror

Oskar Hammargren
Robert Sjöberg

Examinator: Jonas Duregård

Institutionen för data- och informationsteknik
Chalmers tekniska högskola 412 96 Göteborg
Tel: 031-772 1000
Fax: 031-772 3663

Institutionen för data- och informationsteknik
Göteborg, Sverige 2021

Abstract

This project explored how to improve upon an existing program for counting leaf-cutting ants. The original program could count the number of ants but could not distinguish between the different types of leaf-cutting ants (soldier, worker and minis). The main goal of this project was to count and classify each of the different ant types. Both the original program and our expansion of it analysed videos of ants filmed from above to count ants.

A leaf-cutting ant colony at Universeum in Gothenburg was filmed and used as source material. Due to uncertainties if the mini ant type appeared in the source material, ants were only classified as worker or soldier.

The classification was done with two methods, one based on analysing the ant's size and the dominant colour, the other based on machine learning where three different models was used. Results from the different methods varied, with no method able to discern the soldier ants with great precision. The reason for the classification issues with soldier ants could be because of low film quality, low ant type variation or a lack of training data for the machine learning models.

Förord

Tack Cybercom och Henrik Lundqvist för möjligheten att utföra examensarbetet hos er. Det har varit en lärorik och rolig upplevelse. Vi tackar även Universeum som lät oss filma deras myror.

Innehåll

1 Inledning

Det här kapitlet ger en kort introduktion till arbetet och innehåller bakgrund och syfte.

1.1 Bakgrund

Bladskärarmyror lever i symbios med en typ av svamp. De skär blad som de använder för att odla svampen, som då producerar näringsknoppar vilka myrorna sedan äter. Myrornas aktivitet varierar under olika perioder och förhållanden, anledningen är delvis oklar.

Universeum har en koloni av dessa myror i terrarier, sammankopplade med genomskinliga rör. När myrornas aktivitet ökar passerar fler myror genom rören. Universeum vill undersöka vilka faktorer som bidrar till myrornas ökade aktivitet. I dag finns ett bildanalyseringsprogram [1][2] som räknar myror. Det kan dock inte urskilja de olika typer av bladskärarmyror som finns, vilket är av intresse för myrskötarna på universeum. Det här programmet kallar vi för ursprungsprogrammet.

Arbetet utfördes till en början på uppdrag av Universeum hos Cybercom Group i Göteborg. Men ställdes in av Universeum i slutet av mars 2020. Vi jobbade vidare hos Cybercom och valde att fortsätta med samma ämne, men med modifierat syfte.

Cybercom Group är ett internationellt IT-konsultföretag med kontor i bland annat Sverige, Polen och Indien.

1.1.1 Ursprungsprogrammet

Programmet som räknar myror är skrivet i en äldre version av programspråket python (python 2.7) och använder sig utav en gammal version av bildanalysbibliotek openCV (2.4.10). Detta gör programmet svårt att exekvera, speciellt på linux. Det finns en risk att inom en snar framtid kan programmet bli ännu svårare att använda, då python 2 inte längre utvecklas från och med 2020-01-01 [3].

Efter programmet räknat färdigt produceras en textfil som utdata. Textfilen beskriver antalet myror som passerat över tid, men är i ett format som gör det svårt att

hantera i andra applikationer. Ett vanligt format för dataöverföring mellan applikationer är t.ex. JSON [4].

1.2 Syfte

Syftet är att modifiera den ursprungsprogrammet till att:

- Använda en ny version av openCV och python 3
- Använda JSON som utdataformat
- Detektera de olika typerna av bladskärarmyror

Där ett stort fokus ligger på den sistnämnda punkten.

2 Teori/Teknisk bakgrund

Det här kapitlet tar upp den teorin och tekniska bakgrund som krävs för att följa rapporten.

2.1 Myror

Det finns fyra olika typer av bladskärarmyror i en myrkoloni. Drottning, mini, arbetare och soldat. Drottningen befinner sig i svampen och producerar fler myror. Minis är de minsta myrorna, deras huvudsyssla är att reglera svampens klimat. Arbetarna är de mellanstora myrorna, de skär blad och transporterar de till svampen. De största myrorna, soldaterna, förvarar kolonin från andra kolonier.

Hur aktiva myrorna är varierar under olika perioder och förhållanden. Det kan bero på temperatur, luftfuktighet eller något annat med miljön. Innan man tar reda på vilka faktorer som påverkar myraktiveteten behövs en metod för att mäta den.

Ett bra mått på aktivitet är att räkna antalet myror som passerar ett stråk. Detta är väldigt tidskrävande, jobbig och otillförlitlig uppgift för människor [2]. Ett program som noggrant räknar myror kan ge bättre underlag för forskningen inom myrornas aktivitet.

Beskrivningen av bladskärarmyror i den här rapporten är baserad på information given av myrskötarna på Universeum samt ur boken [5]. Det finns många arter av bladskärarmyror med både små och stora variationer mellan dem. En generell och förenklad beskrivning har givits här, som eventuellt inte stämmer helt med den arten Universeum äger.

2.1.1 Klassificeringsproblem

I de inspelade filmerna för myrräkning är vi osäkra om någon myra av typ mini finns med. Alla myror varierar lite i storlek och vi vet inte om de mindre myrorna i filmerna är minis, eller något mindre arbetsmyror. Vi har därför valt att endast klassificera myrorna som arbetsmyra eller soldatmyra, där storlekskillnaden är mycket mer markant.

2.2 Maskininlärning

Maskininlärning kan beskrivas som en teknik som förbättrar sig genom tidigare erfarenheter[6]. Det använder sig av neurala nätverk för att lösa problem som är svårlösta med ”traditionella” algoritmer. Ett exempel på ett område där maskinlärning ofta används är inom bildklassificering. Vid bildklassificering tränas en modell med förklassade bilder till att skapa kopplingar som kan användas för att klassificera okända bilder rätt.

Träningen görs iterativt på en träningmängd vilken är en delmängd av all data. Träningsdatan, som är förklassificerad, matas in i modellen som sedan utför sin klassning av indatan. Efteråt jämförs det riktiga svaret mot det som algoritmen kom fram till. När jämförelsen är gjord så används en så kallad förlustfunktion vilket försöker styra algoritmen mot det korrekta svaret för hela träningsmängden. Förhoppningsvis leder detta till att modellen kan gissa rätt på liknande indata som inte är en del av träningsmängden.

Ett viktigt koncept inom maskininlärning är överanpassning (overfitting). Överanpassning av nätverket kan uppstå ifall träningsdatan har för få variationer. T.ex. ett nätverk tränas för att upptäcka objekt där bakgrunden råkar vara en viss färg. Om bakgrundsfärgen ändras efter träningen kan det leda till att objekten inte längre detekteras. Då är nätverket överanpassat för en specifik bakgrundsfärg.

2.3 Objektdetektering

Objektdetektering utökar bildklassificering genom att upptäcka var i en bild ett, eller flera, objekt befinner sig. Objektdetekterare finns i två olika former, 1-stegsdetektorer och 2-stegsdetektorer [7]. 1-stegsdetektorer är generellt snabbare på kan upptäcka alla objekt i en bild än 2-stegsdetektorerna. Men 2-stegsdetektorerna har oftast högre precision för att kompensera för sin långsammare hastighet. Med högre precision menas att större andel av objekten hittades samt att placeringen av dessa var mer exakt.

2 stegs objektdetektorer delar in detekteringen i två delar, hitta objekt och klassificera objekt. För att hitta området där ett objekt befinner sig i använder 2 stegs objektdetektorer en sökalgoritm eller maskininlärning. Dessa områden med möjliga objekt analyseras sedan av ett nätverk för bildklassificering.

I 1 stegs objektdetektorer utförs allt på samma gång istället. Det kan göras genom att placera ett rutnät över bilden och enbart använda de delar med hög klassannolikhet (sannolikheten att det är en specifik klass i denna delen av rutnätet).

2.4 Bibliotek

OpenCV - Det här biblioteket är gjort för bildbearbetning. Det finns över 2500 optimerade algoritmer för att göra det mesta inom att spåra och hitta objekt mm. OpenCV står för Open source Computer Vision Library. Det används, bland annat, av stora företag och har gränssnitt till flera programmeringsspråk [8].

Tensorflow - Tensorflow är ett bibliotek skapat för utveckling av maskininlärning i applikationer. På grund av dess många funktioner som den stödjer, samt att Google bygger och använder biblioteket, har det blivit väldigt populärt.[9]

3 Metod

Det här kapitlet går igenom de metoder som användes för insamling av data och verifiering av programmets precision.

3.1 Datainsamling

För att testa våra ändringar av originalprogrammet krävdes film på bladskärarmyror. Universeum har en koloni med bladskärarmyror som vi filmade vid ett tillfälle. På Universeum lever bladskärarmyrorna i nio stycken terrarium sammankopplade med genomskinliga rör, se figur 3.1. Vi filmade myrorna uppifrån med en mobilkamera genom en öppning i ett av rören. Längs botten placerades en vit servett för att få hög kontrast mellan myror och bakgrunden, för att underlätta bildanalysen senare.

Flera filmer av längd 4-12 minuter spelades in för en totallängd på cirka 40 minuter. Eftersom arbetet med Universum avslutades tidigt kunde vi inte filma mer än de 40 minuterna vi fick från det första, och enda, filmtillfället.

Figur 3.1: Röret vi filmade igenom. Mobilkameran placerades över den svarta tejpen riktad nedåt.

3.2 Verifiering

Verifiering av programmets precision utfördes genom manuell räkning av myror på en filmsekvens uppspelad i reducerad hastighet. Det var svårt att räkna antal myror i båda riktningarna samtidigt, så för noggrannhetens skull räknades myrorna i en riktning åt gången. Varje riktning räknades minst två gånger per person (två personer). Medelvärdet antogs vara korrekt och programmets resultat jämfördes med detta.

För maskininlärningsmetoden hölls träningsdata och testdata separat.

3.3 Räkning och klassificering

Tre olika metoder användes för att räkna myror, maskininlärning, dominantafärg-metoden och den ursprungliga metoden. De två sistnämna spårar och detekterar myror genom att leta efter konturer. Den ursprungliga metoden klassificerar inte myror, utan räknar endast det totala antalet. Dominantafärg-metoden bygger på den ursprungliga metoden men klassificerar även myror baserat på storleken och den vanligaste färgen i varje kontur.

Makininlärningsmetoden använder sig av ett neuralt nätverk som tränats på uppmärkta bilder för att detektera och klassificera myror. Mer om alla dessa metoder i avsnitt 4.

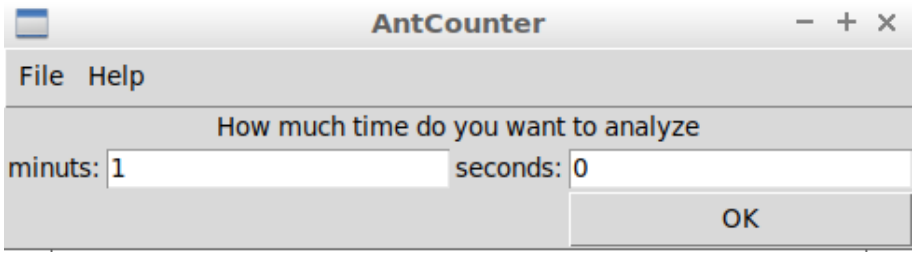
4 Systemdesign

Det här kapitel beskriver systemdesignen på ursprungsprogrammet och slutprogrammet med utökad funktionalitet.

4.1 Ursprungsprogram

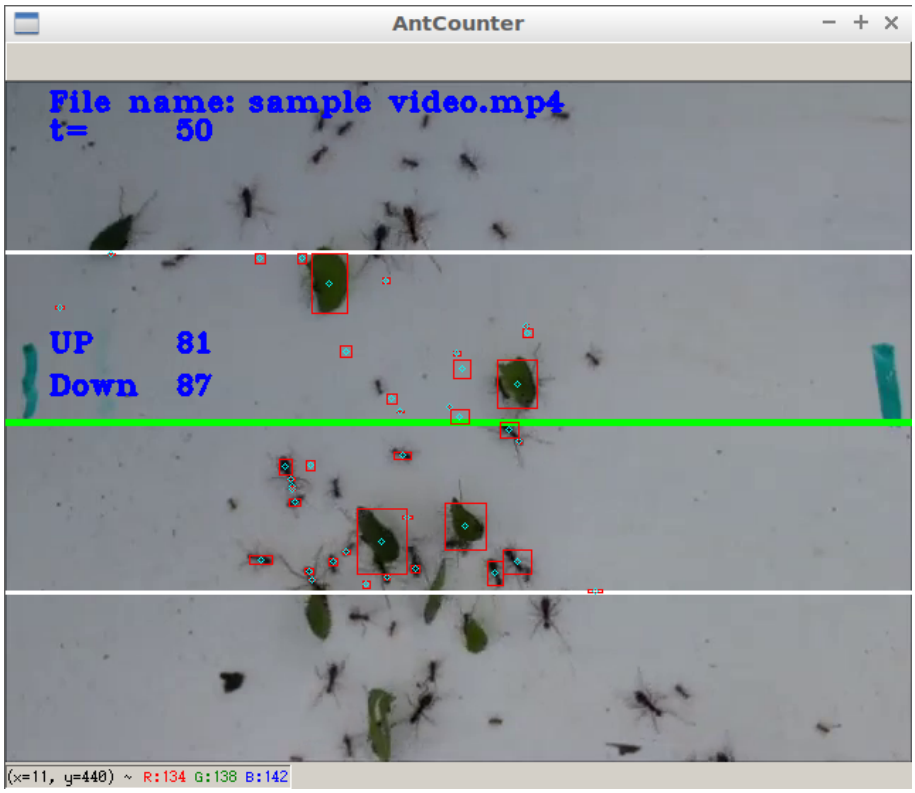
För att underlätta exekvering av ursprungsprogramet valde vi att skriva om det till att använda en nyare version av openCV (4.2) och python 3. Denna uppgradering var inte helt trivial då openCV 2.4 och senare versioners pythonbindingar skiljer sig på många sätt. Den nya versionen av openCV bytte namn och funktionalitet på flera funktioner och konstanter. Paradigmen ändrades också till att vara mer objektorienterad. Ursprunsprogrammet refererar hädanefter till den uppdaterade versionen av programmet, med python 3 och openCV 4.2.

Ursprungsprogrammet bestod av ett grafiskt användargränssnitt där en eller flera filmsekvenser valdes för analysering samt den tid att analysera dem i, se figur 4.1. Efter bekräftelse öppnades ett nytt fönster där den valda videon spelades upp och analyserades.



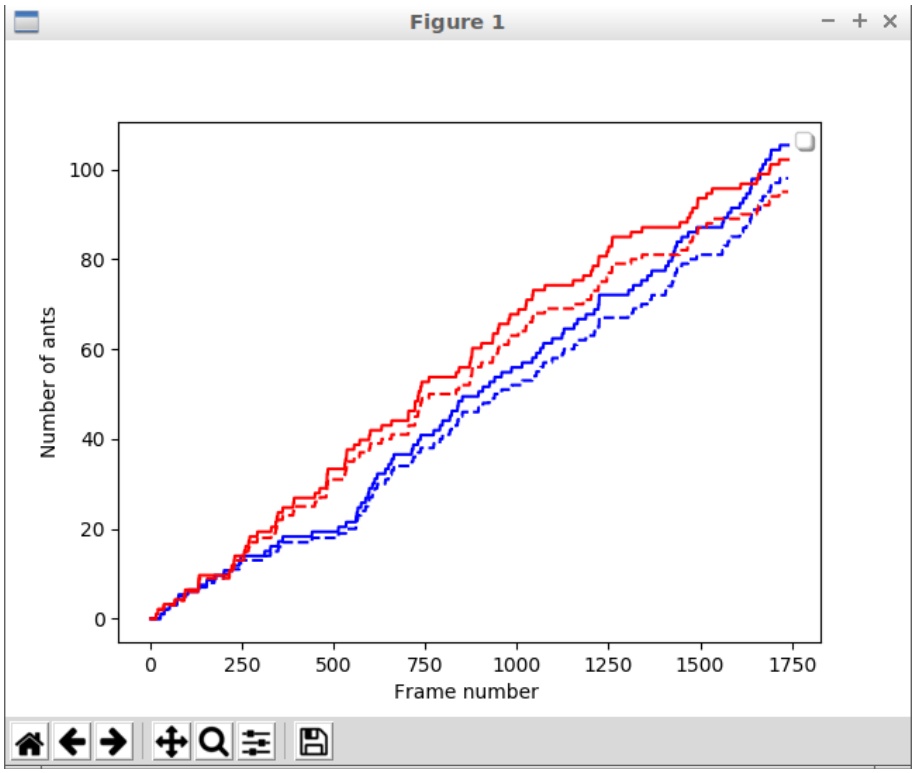
Figur 4.1: Det grafiska användargränssnittet

Analyseringsfönstret fungerade genom att analysera varje bild i en video och söka efter konturer. Varje kontur mellan de vita linjerna ringades in av en rektangel och tilldelas ett unikt id-nummer, se figur 4.2. Dessa konturer spårades mellan varje bildruta och om en kontur passerade mitten med riktning upp (eller ner) inkrementerades antalet myror upp (eller ner).



Figur 4.2: Analyseringsfönstret efter 50 sekunder. 81 myror har passerat uppåt, 87 neråt. Myrornas id-nummer är gömt för användaren

När den angivna tiden löpt ut slutade analysen av videon och ett nytt fönster skapades med en graf på antalet myror upp och ner över tid, se figur 4.3. Uppskattningsvis missade programmet cirka 7,5% av alla myror. För att ta hänsyn till det fanns även ett justerat värde för myror upp och ner.

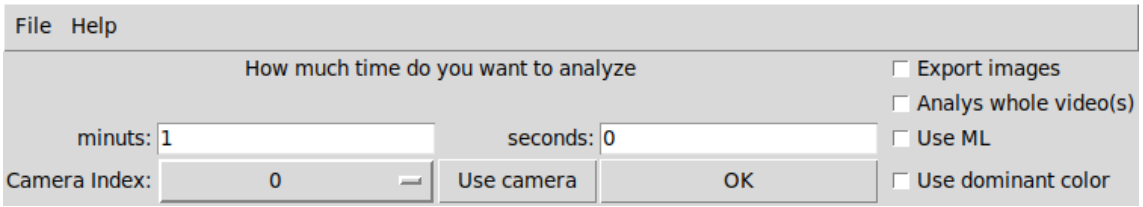


Figur 4.3: Graf på antal myror upp (blå streckad) och ner (röd streckad) över tid samt de justerade värdena (solid)

Slutligen genererades en textfil med antalet myror upp och ner samt de justerade värdena för varje bildruta av videon.

4.2 Slutprogrammet

Slutprogrammet är baserat på samma grafiska gränssnitt, fast utökat med flera funktioner, se figur 4.4. Till höger finns fyra nya alternativ, ”Export images”, ”Analyse whole videos(s)”, ”Use ML” och ”Use dominant color”. Längst ner är det möjligt att välja en kamera för att räkna myror i realtid, istället för att analysera en filmsekvens. Om flera kameror är anslutna till datorn kan man välja vilken kamera som ska filma i listan "Camera index".

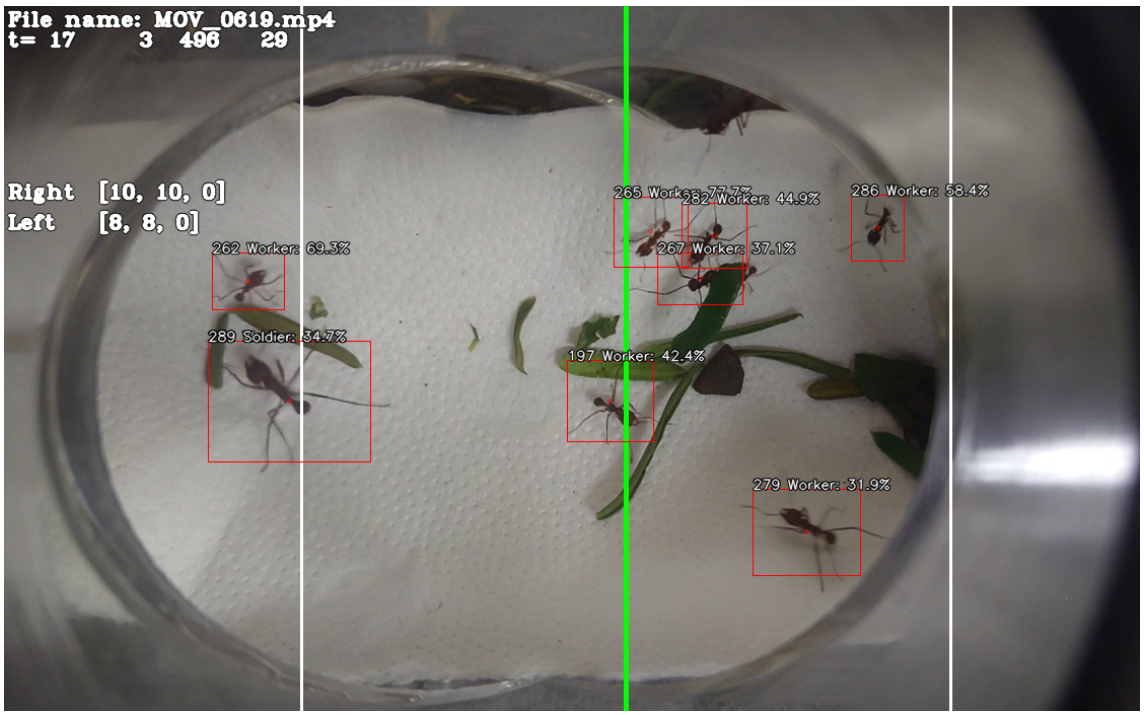


Figur 4.4: Det slutliga programmets gränssnitt. Fler alternativ för att analysera myror finns.

Utöver dessa inställningar finns även en JSON konfigurationsfil (se appendix A.1) för mer tekniska detaljinställningar som exempelvis tröskelvärden för konturdektektion.

4.2.1 Maskininlärningsmetoden

När ”Use ML” är aktivt används en förtränad modell för att detektera och klassificera myror. Precis som tidigare inkrementeras antalet myror när en myra passerar mittlinjen. Om modellen anser att myran är en arbetarmyra inkrementeras antalet arbetarmyror, samma gäller för soldaterna. Modellen beräknar sannolikheten att en myra är en viss typ. Typen med högst sannolikhet är det en myra räknas som när den korsar mitten. Se figur 4.5 för hur användningen av en maskininlärningsmodell kan se ut.



Figur 4.5: En skärmdump 17 sekunder in i analyseringen av en filmsekvens. Här syns flera arbetsmyror och en soldatmyra. Alla detekterade myror är inramade med lite text över. Texten innehåller myrans ID, vilken myrtyp modellen anser är mest trolig samt hur säker modellen är. För soldatmyran (den stora till vänster) är modellen inte så säker på att det är en soldatmyra. Klassificeringen hoppar här mellan arbetsmyra och soldatmyra. Oturligt nog var soldatmyran klassad som arbetsmyra när den passerade mittlinjen och räknades därför som arbetsmyra. Beroende på vilken modell som används kan klassificeringen vara ganska resurskrävande. Detta indikeras av info-texten längst upp till vänster, som säger att det för närvarande analyseras 3 bilder per sekund[1]. Värt att notera är att här räknas myrorna från vänster/höger istället för upp/ner. Detta kan ställas in i JSON konfigurationsfilen.

[1] På en Macbook Pro early 2011

4.2.2 Dominantafärg-metoden

Den dominantafärg-metoden gör en förutsägelse på sannolikheten att en rektangel innehåller en arbetsmyra eller soldatmyra. Dominantafärg-metoden bygger på ursprungsmetoden, att leta efter konturer, men lägger till klassificering baserat på

arean hos rektangeln (runt konturen). En soldatmyra är större än en arbetsmyra vilket leder till en större kontur och rektangel.

Men enbart storlek är inte en bra indikator för typen av myra. Problem uppstår när en arbetsmyra bär på ett blad. Då blir myrans kontur större vilket leder till en större rektangel som kan uppfattas som en soldatmyra. För att åtgärda detta granskar programmet varje rektangel som är stor nog för att innehålla en soldatmyra och kollar vilken färg som är vanligast. Om den vanligaste färgen är grön tolkar programmet det som en arbetsmyra bärandes på ett blad.

För att öka klassificeringsprecisionen analyseras rektanglar flera gånger och ett snitt mellan alla tidigare förutsägelser görs. Tidigare analyserade rektanglar analyseras igen efter att ett visst antal bildrutor (standard är fem bildrutor) har förflutit.

Bildintervallet på hur ofta en rektangel analyseras kan ändras i JSON konfigurationsfilen. Även minsta arean för en soldatmyra och färgintervallet (vad som tolkas som grönt) går att justera.

4.2.3 Andra inställningar

Om ”exportera bilder” är aktivt så exporteras en stillbild till en mapp (denna mapp kallas förvalt för ”img” och placeras i samma mapp som programmet kördes ifrån). Standard är att var 30:e bildruta exporteras, detta kan ändras i konfigurationsfilen. Vi använde denna funktion för att samla bilder till att träna modellerna för maskininlärning. Det kan också vara användbart om en annan användare av programmet vill träna en egen modell, om personen filmar med andra förhållanden som kan påverka klassificeringen av myror.

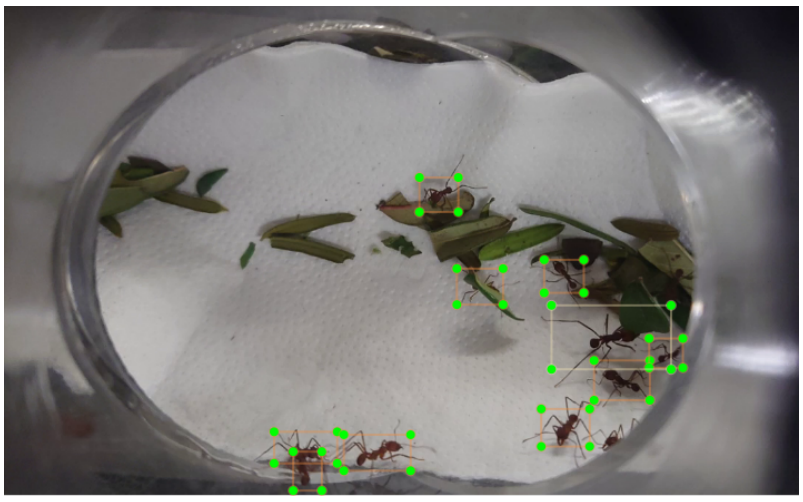
”Analyse whole video” är en inställning som överskrider analystiden som anges i de två textrutorna. Detta underlättar analysen av flera videor med olika längd. Till exempel, om man vill analysera tre hela filmsekvenser på 1, 2 respektive 3 minuter vardera i ett svep.

5 Maskinlärningsmodeller

Myrorna klassificerades med hjälp av maskininlärning. För att generera modeller för klassificering av myror användes förtränade modeller, tränade i generell objektdetektering, som utgångspunkt. Dessa modeller tränades sedan vidare med bilder på manuellt uppmärkta myror i skärmdumpar från myrvideorna. Detta resulterade i modeller som detekterar och klassificerar myror.

5.1 Träningsdata

Träningsdatan kom från exporterade stillbilder från filmsekvenserna. Stillbilderna exporterades med en sekunds intervall för att ge variation mellan bilderna. Dessa bilder lästes sedan in i bildklassificeringsprogrammet lableImg [10] där vi manuellt markerade alla myror med rektanglar. Vi tilldelade sedan rektanglarna typen arbetsmyra eller soldatmyra. Denna process, upprepad med många bilder, skapade träningsdatan åt alla maskinlärningsmodeller. Figur 5.1 ger ett exempel på hur markeringen av myror såg ut.



Figur 5.1: Exempel på markering av myror i programmet LabelImg. Till höger syns en soldatmyra med en aningens ljusare rektangel kring sig. Totalt markerades strax över 500 stillbilder.

5.2 Träning

Efter att bilderna har klassificerats skapas en så kallad record-fil. Record-filen är en binär fil som innehåller den data om bilderna som tensorflow behöver för att träna modellen.

När record-filen finns kan den användas för att träna vidare på förtränade modeller. Modellerna vi använde oss av var från tensorflows object detection Model Zoo [11] och var förtränade på det standardiserade datasetet COCO (common objects in context)[12]. Efter fortsatt träning på vårt dataset exporteras modellen till en fil för att kunna användas av andra program.

5.3 Modelltyper

Vi använde oss utav tre olika modeller för att detektera och klassificera myror, Resnet och Mobilenet version 1 och 2. Vi valde dessa modeller då de var bra rankade i Model Zoo och relativt snabba. Resnet är det nätverk som facebooks AI avdelning har skapat och är en av de bästa objektdetekeringsmodeller när det gäller precision och hastighet. Detta användes som en referens för alla ML modeller vi testade och det som förväntas ge bäst resultat. Då denna modell var bäst rankad av de modeller vi använde. Mobilenet version 1 och 2 användes då de inte är lika beräkningsmässigt krävande. Som indikerat av namnet är de menade till att användas i mobila system med låg beräkningskapacitet. Dessa används för att se skillnaden i precision mellan mobilenets och ett större nätverk som Resnet.

6 Resultat

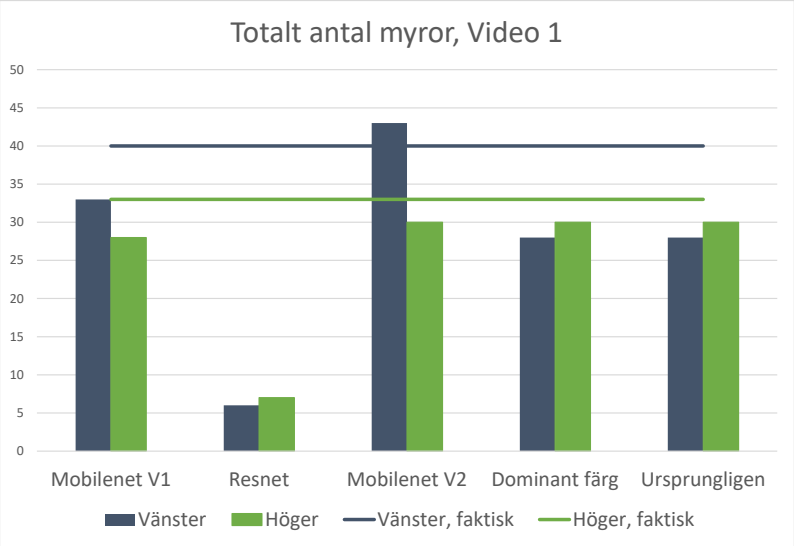
Programmet har uppdaterats och använder en nyare version av openCV (4.2) och python 3. Utdataformatet är ändrat till ett standardiserat JSON-format (se appendix A.2) som underlättar användning av utdatan i andra applikationer.

Detekteringen av de olika typerna av bladskärarmyror realiserades med två olika metoder, dominantafärg-metoden och med maskininlärning. För maskininlärningen skapades tre olika modeller: Mobilenet v1, Mobilenet v2 och Resnet. Detta ger totalt fyra olika resultat på antalet myror av varje typ.

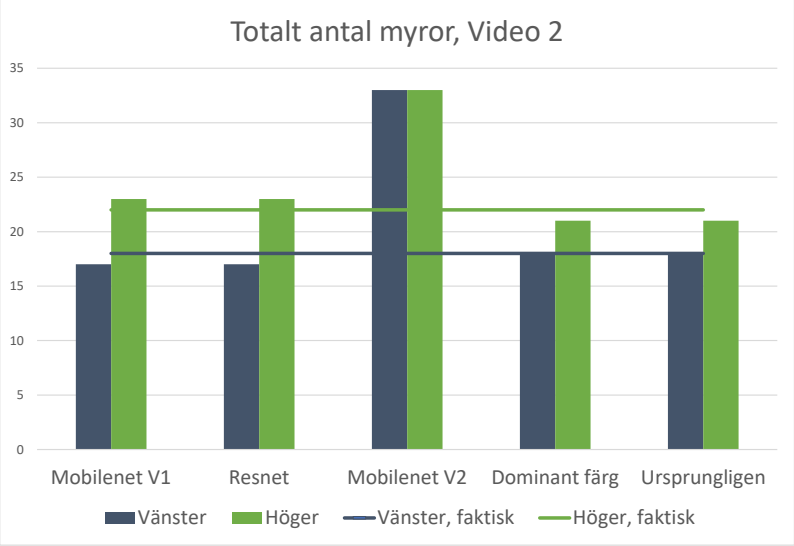
För att sätta dessa fyra reslutat i perspektiv har den ursprungliga metodens resultat också inkluderats i jämförelserna med totala antalet myror. Eftersom den ursprungliga metoden endast räknar det totala antalet myror, så har jämförelser med denna metod gjorts genom att addera antalet arbetsmyror och soldatmyror. Vidare inkluderas inte den ursprungliga metoden i jämförelserna med antalet av de olika myrtyperna, då den inte klassificerar myror.

För variationens skull analyserades två olika videor i en minut var, en med lite löv och en med mycket löv mot den vita bakgrunden. Nedan följer resultatet av det totala antalet myror, se figur 6.1 och 6.2. Resultaten i båda diagrammen är ganska jämna. Undantaget är mobilenet v2 som har något högre antal myror i båda filmsekvenserna och resnet i video 1 med väldigt få myror.

Dominantafärg och den ursprungliga metoden visar samma resultat i dessa grafer. Detta för att dominant färg har samma tillvägagångssätt som den ursprungliga metoden att detektera myror. Skillnaden är bara att den dominanta färgen och storleken på myran tas hänsyn till för att klassificera myran. Resultaten från dessa metoder kan möjligtvis förbättras genom att ändra t.ex. den gaussiska oskärpan eller gråskalans tröskelvärden (för att hitta konturer) i konfigurationsfilen.

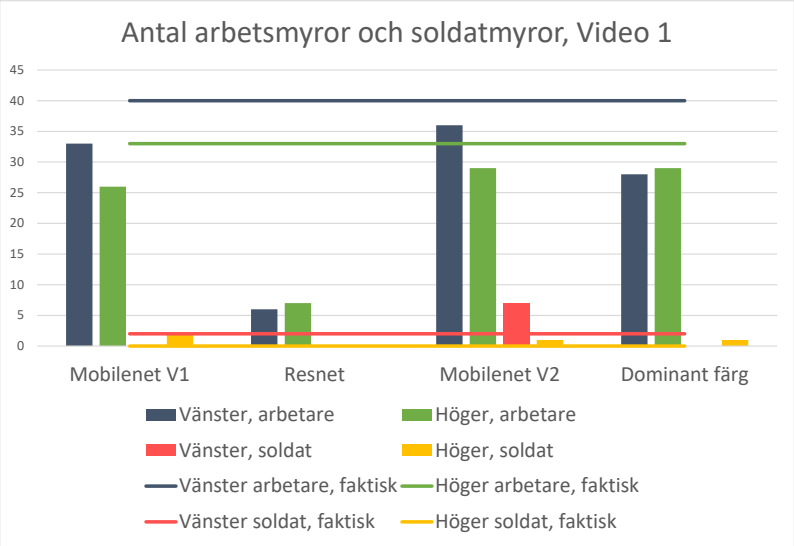


Figur 6.1: Resultat av de olika modellerna och metoderna för video 1. Horisontell linje avser det korrekta antalet myror. Video 1 har färre löv utspridda på underlaget.

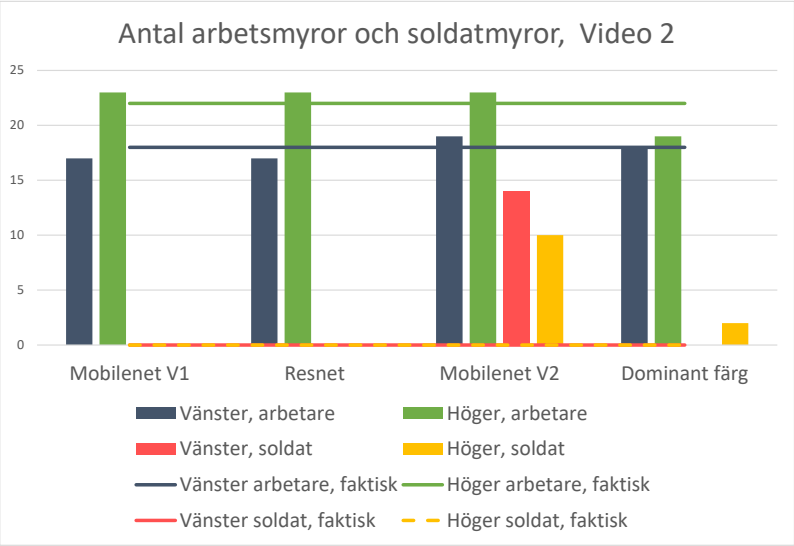


Figur 6.2: Resultat av de olika modellerna och metoderna för video 2. Horisontell linje avser det korrekta antalet myror. Video 2 har fler löv utspridda på underlaget.

Liknande resultat syns även vid analysen av antalet av de olika myrtyperna, se figur 6.3 och 6.4. Generellt är antalet arbetsmyror ungefär samma oberoende av metod och modell. Resnet är undantaget som ännu en gång som visar väldigt få arbetsmyror i video 1. Intressant att notera är att mobilenet v2 grovt överskattar antalet soldatmyror i video 2, som inte ens innehåller några soldatmyror.



Figur 6.3: Resultat av mobilenet v1, mobilenet v2, resnet och dominantafärg-metoden för video 1. Horisontell linje avser det korrekta antalet myror. Video 1 har färre löv utspridda på underlaget. Två soldatmyror passerade åt vänster i denna video, inga åt höger.



Figur 6.4: Resultat av mobilenet v1, mobilenet v2, resnet och dominantafärg-metoden för video 2. Horisontell linje avser det korrekta antalet myror. Video 2 har fler löv utspridda på underlaget. Inga soldatmyror i denna video.

7 Diskussion

Det här kapitlet diskuterar problem som uppstått under projektets gång och möjliga åtgärder för dessa. Målen att uppdatera programmet och ändra utdataformatet till JSON tog relativt kort tid att slutföra, med få saker att diskutera. Därför diskuteras bara aspekter relaterade till detektering av olika myrtyper här.
Utöver ovannämnda diskussionen kommer även etiska aspekter och hållbarhetsfrågor nämnas.

7.1 Filmproblem

Ett bekymmer vi stötte på var kvalitén på filmsekvenserna. Vår avsikt var att återvända till Universeums myror, därför var vi inte speciellt förberedda och filmade inte under optimala förhållanden. Till exempel användes en servett som bakgrund för att ge hög kontrast. Det var helt acceptabelt men den innehöll små gråa prickar som ibland kunde tolkas som en kontur av programmet, eller hjälpa till att förvränga en kontur. I bland kröp det myror under servetten och blev inte filmade.

Utöver det filmade vi där det var lättast. Det råkade vara i ett rör utanför det terrariet där blad placerades åt myrorna av myrskötarna. Problemet var att många myror flyttade bladen från terrariet till röret där vi filmade. Efterhand ökade mängden blad på servetten. Många av dessa blad var mörkgröna och gjorde det väldigt svårt att få någon kontrast mellan myrorna och bakgrunden. Många blad hamnade på mittlinjen, precis där myrorna räknas. Det ledde till ett typiskt scenario där en myra som spårades bra av programmet tills den kom till mittlinjen. Vid mittlinjen passerade den över ett blad, kontrasten försvann och myran blev inte räknad.

7.2 Maskininlärning

Våra modeller för att klassificera myror resulterade inte riktigt i så bra resultat som vi hade hoppats på. Många soldatmyror blev felaktigt klassificerade som arbetsmyror. Modellen lyckades ofta lista ut att en soldatmyra faktiskt var en soldatmyra, men försent. Det typiska händelseförloppet när en soldatmyra kröp in i bild var att den först klassades som en arbetsmyra. Detta med ganska hög sannolikhet, men med tiden lutar modellen mer åt soldathållet och när soldatmyran passerar mittlinjen är modellen osäker vilken typ det är. När soldaten sedan passerat mittlinjen har mo-

dellen blivit säker på att det är en soldat. Vilket, vid det laget, är försent.

En möjlig lösning på det problemet är att fortsätta ha koll på vilken typ en myra är efter den har passerat mittlinjen och retroaktivt uppdatera antalet av varje myrtyp, om den byter klass. En annan lösning kan vara att filma på ett lite längre avstånd, då får modellen mer tid på sig att korrekt klassificera myran. Det är också möjligt att en annan modell hade givit bättre och mer exakt resultat. Viktigt att tänka på med modeller är att mer exakta modeller är långsammare och tar mer datorkraft.

Ytterligare en faktor som troligen bidrog till att klassificeringen av soldatmyror inte var optimal är de få soldatmyror i videorna, jämfört med arbetsmyror. Ett försök till åtgärd var att selektivt välja fler stillbilder från videorna innehållande soldatmyror, för att motverka klassoblansen. Det är möjligt att detta kan ha överanpassat modellen genom det begränsade urvalet.

Ett annat bekymmer är att klassificering av myror med hjälp av maskininlärningsmetoden troligtvis fungerar sämre om man filmar med andra förhållanden. Om man ska räkna myror på ett annan plats med en annan kamera, filmavstånd, ljussättning, bakgrund, upplösning eller andra typer av blad, så behöver modellen tränas på det.

7.3 Etik

Då vi använder maskininlärning i detta projekt så är det av intresse att diskutera de potentiella etiska implikationerna. Etik i maskininlärning är ett omdebatterat område just nu[13].

Vi har bedömt att det inte finns några stora etiska dilemman med vårt program. Rent hypotetisk, då programmet spårar myror, skulle det med stora modifikationer och en annan träningsdata kunna användas att spåra människor. Människospårning är något som kan vara oetiskt. Dock skulle det innebära mycket kod-modifikation, en helt ny modell och en ny stor bildmängd på människor för att realiseras. Realistiskt är det troligtvis enklare att börja om från början än att utgå från vårt program om en applikation för människospårning ska utvecklas.

7.4 Hållbarhet

Detta program kan bidra till att ge en bättre förståelse om bladskärarmyrors beteende. Allmänt bättre kunskap om vår natur är viktigt om vi ska minska vår negativa påverkan på naturen[14].

8 Slutsats

Vi ser ett ganska spritt resultat från graferna i del 6. Antalet arbetsmyror var generellt ganska nära det faktiska värdet (oftast lite lägre). Beräkningen av soldatmyrorna är det svårare att dra några slutsatser ifrån då de var väldigt få i antal. Från våra tester presterade mobilenet v1 bäst av maskinlärningsmodellerna och gav liknade resultat jämfört med dominantafärgmetoden och ursprungsmetoden. Däremot är alla maskinlärningsmodellerna mycket mer resurskrävande än de två andra metoderna. Om en bästa metod för att räkna myror ska utses, är det också viktigt att ta hänsyn till faktorer som tillgänglig beräkningskapacitet och energikonsumptionsbegränsningar.

I nuläget ser vi inte någon större fördel med tillägget av detektering av olika myrtyper på grund av osäkerheten kring soldatmyrorna samt att mini-typen inte räknas alls. Om bättre maskinlärningsmodeller skapas är det möjligt att programmet kan tillföra något.

Däremot gör uppdateringen av programmet det lättkört nu, och framöver. Vidare kan JSON-formatet på utdatan underlätta användningen av resultatet i andra applikationer.

Litteratur

[1] S. B. Sanint. (n.d.). AntCounter, [Online]. Tillgänglig: https://sites.google.com/site/antcounter/ (hämtad 18 dec. 2020).

[2] S. Bustamante och A. Amarillo, "AntCounter Software: Counting Leaf-Cutting Ants Was never so Precise, fast and Easy", *Journal of Insect Behavior*, årg. 29, maj 2016. DOI: 10.1007/s10905-016-9558-0.

[3] Python Software Foundation. (n.d.). Sunsetting Python 2, [Online]. Tillgänglig: https://www.python.org/doc/sunset-python-2/#:~:text=We\%20have\%20decided\%20that\%20January,as\%20soon\%20as\%20you\%20can. (hämtad 10 april 2021).

[4] (n.d.). Introducing JSON, [Online]. Tillgänglig: https://www.json.org/json-en.html (hämtad 10 april 2021).

[5] B. Hölldobler och E. O. Wilson, *The Superorganism: The Beauty, Elegance, and Strangeness of Insect Societies*. W.W. Norton & Company Inc, 2009, kap. 9, s. 408–468.

[6] M. Mohri, A. Rostamizadeh och A. Talwalkar, *Foundations of machine learning*. MIT press, 2018, s. 1.

[7] P. Soviany och R. T. Ionescu, "Optimizing the Trade-Off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction", *2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, s. 209–214, 2018.

[8] OpenCV team. (n.d.). OpenCV, [Online]. Tillgänglig: https://opencv.org/about/ (hämtad 18 dec. 2020).

[9] Google Brain team. (n.d."). Tensorflow, [Online]. Tillgänglig: https://www.tensorflow.org/about (hämtad 18 dec. 2020).

[10] Qaprosoft. (n.d). LabelImg, [Online]. Tillgänglig: https://github.com/qaprosoft/labelImg (hämtad 21 dec. 2020).

[11] Google Brain team. (n.d.). Model Zoo, [Online]. Tillgänglig: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (hämtad 5 jan. 2021).

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár och C. L. Zitnick, "Microsoft COCO: Common Objects in Context", i *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele och T. Tuytelaars, Eds., Cham: Springer International Publishing, 2014, s. 740–755, ISBN: 978-3-319-10602-1.

[13] T. Hagendorff, "The Ethics of AI Ethics: An Evaluation of Guidelines", *Minds & Machines 30*, 99–120, 2020. DOI: https://doi.org/10.1007/s11023-020-09517-8.

[14] H. Bruyninckx. (2018). Environmental change: knowledge is key to mitigating impacts on people and nature, [Online]. Tillgänglig: https://www.eea.europa.eu/articles/environmental-change-knowledge-is-key (hämtad 10 april 2021).

A Appendix 1

A.1 JSON konfigureringsfil

```
{
    "ml": {
        "imageWidth": 257,
        "imageHeight": 270,
        "reevaluateAntFrameInterval": 15,
        "classes": [
            "Nothing",
            "Worker",
            "Solider"
        ],
        "modelName": "ant_model.h5"
    },
    "dominantColor": {
        "greenThreshold": [
            [0, 120], [28, 255], [0, 120]
        ],
        "reevaluateAntFrameInterval": 5
    },
    "paths": {
        "previousFiles": [
            "/home/robert/Documents/skola/Exjobb/Myr_videor/MOV_0619-r.mp4",
            "/home/robert/Documents/skola/Exjobb/Myr_videor/MOV_0621-r.mp4"
        ],
        "exportImagesRelative": "/img/roi"
    },
    "other": {
        "maxMovAnt": 42,
        "gaussianBlur": [19, 19],
        "greyThreshold": [60, 255],
        "minAntArea": 400,
        "soldierAntArea": 2500,
        "exportImagesFrameInterval": 30
    }
}
```

A.2 Urklipp ur resultatet från det standardiserade JSON dokument

För varje bildruta i filmen skapas ett objekt med antalet myror åt båda hållen. Detta urklipp är från bildruta 1740, efter nästan en minut.


```
"1740": {
            "Ants Right": {
                "Total": 7,
                "Total adjusted": 7.527799999999999,
                "Worker ant": 6,
                "Soldier ant": 1
            },
            "Ants Left": {
                "Total": 9,
                "Total adjusted": 9.6786,
                "Worker ant": 8,
                "Soldier ant": 1
            }
        }
```