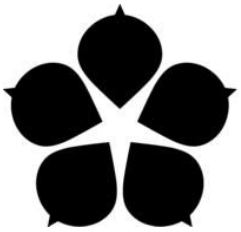


Přírodovědecká fakulta
Faculty of Science

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích

Přírodovědecká fakulta

Katedra Aplikované informatiky

Mobilní aplikace pro včelaře

Bakalářská práce

Vypracoval: Nikola Kraus

Školitel: Ing. Miroslav Skrbek, Ph.D.

Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

ZADÁVACÍ PROTOKOL BAKALÁŘSKÉ PRÁCE

Student: Nikola Kraus
(jméno, příjmení, tituly)

Obor – zaměření studia: 1801R001 / Aplikovaná informatika

Katedra: Ústav aplikované informatiky

Školitel: Ing. Miroslav Skrbek, Ph.D.
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Garant z PřF: .....
(jméno, příjmení, tituly, katedra – jen v případě externího školitele)

Školitel – specialista, konzultant: .....
(jméno, příjmení, tituly, u externího š. název a adresa pracoviště, telefon, fax, e-mail)

Téma bakalářské práce: Mobilní aplikace pro včelaře

Cíle práce:

Seznamte se s existujícími aplikacemi pro identifikaci a vyhodnocení napadení včelstev roztočem a identifikaci matky na včelím plástu. Najděte slabá místa aplikací a navrhněte a implementujte vlastní aplikaci vhodnou pro české včelařské prostředí. Aplikaci implementujte pro mobilní telefon pod operačním systémem Android. Detekci roztoče Varroa destruktor a detekci včelí matky převezměte z předcházejících závěrečných prací a portujte je do mobilní aplikace a případně upravte tak, aby svými nároky odpovídala výkonnostním možnostem mobilních zařízení. Uživatelské rozhraní navrhněte tak, aby bylo pro uživatele maximálně intuitivní a bez nutnosti nastavování většího množství parametrů. Uživatelské rozhraní podrobte uživatelskému testu. Rozsah práce stanovte po dohodě s vedoucím práce.

Základní doporučená literatura: literaturu dodá vedoucí práce v průběhu řešení bakalářské práce.
Financování práce : práce nemá finanční nároky

Vedoucí práce : Ing. Miroslav Skrbek, Ph.D. .....podpis : .....

U externích vedoucích fakultní garant práce.....podpis : .....

Garant oboru bak.. studia (nepožaduje se u zaměření „příprava na mag. studium biologie)
..... podpis : .....

Vedoucí katedry ..... podpis .....

Případný souhlas vedoucího ústavu AV .....podpis : .....

V Českých Budějovicích dne .....

Převzal/a dne 1. 3. 2019 podpis : Kraus

Prohlášení

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě, elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích

dne :.....

…….……

Nikola Kraus

Bibliografické údaje

KRAUS, Nikola. Mobilní aplikace pro včelaře. [Mobile application for beekeeper. Bc. Thesis, in Czech. 45], České Budějovice, Czech Republic, 2020. Bakalářská práce. Faculty of Science, University of South Bohemia.

Mobilní aplikace pro včelaře

Anotace

Tato bakalářská práce se zabývá problematikou implementace a sdružení již dříve vyvinuté aplikace pro detekci a sečtení jedinců *Varroa destructor* na lepové podložce vyjmuté ze dna včelího úlu a aplikace využívajících metod počítačového vidění a konvolučních neuronových sítí za účelem detekce a lokalizace včelí matky. Výše uvedené aplikace byly vyvinuty v počítačovém prostředí a následně proběhla jejich implementace do mobilní aplikace, čímž by mělo dojít k jednoduchému zpřístupnění této technologie pro koncového uživatele.

Klíčová slova: *Varroa destructor*, včela medonosná, neuronové sítě, analýza obrazu, mobilní aplikace

Bibliographic information

KRAUS, Nikola. Mobilní aplikace pro včelaře. [Mobile application for beekeeper. Bc. Thesis, in Czech. 45], České Budějovice, Czech Republic, 2020. Bakalářská práce. Faculty of Science, University of South Bohemia.

Mobile app for beekeeper

Annotation

This bachelor thesis deals with the problematics of implementation and merging of two applications: the first and previously already developed application serves for the purposes of detection and counting of Varroa destructor individuals on a glue pad removed from the bottom of a beehive, the second application uses computer vision methods and convolutional neural networks to detect and locate the queen bee. The applications mentioned above were developed in a computer environment and subsequently implemented into a joint mobile application, which should make this technology easily accessible for the end user.

Keywords: *Varroa destructor*, honeybee, neural network, image analysis, mobile application

Poděkování

Rád bych touto cestou poděkoval panu Ing. Miroslavu Skrbkovi, Ph.D za ochotu a systematické vedení mé bakalářské práce. Dále bych také chtěl velmi poděkovat Ing. Václavu Krištůfkovi, CSc. za cenné rady a vstřícný přístup během konzultací bakalářské práce. Bez výše uvedených by bylo mnohem těžší, aby tato práce vznikla. V neposlední řadě bych rád poděkoval své rodině a přátelům za podporu, kdykoli jsem ji potřeboval.

Obsah

1. ÚVOD

Rychlost vývoje technologií v jednadvacátém století jde neúprosnou rychlostí vpřed, jejich užívání je v dnešní době rozšířeno takovou měrou, že prakticky v moderní společnosti neexistuje odvětví, kde by se moderní technologie neužívaly. Pokud mohou existovat aplikace do mobilních telefonů rozpoznávající druhy rostlin nebo hub, tak se nabízí využít tyto technologie i v jiných odvětvích, a to nejen v rostlinné říši, ale i v živočišné, v tomto případě konkrétně u společenského typu hmyzu, kterým jsou včely. Ač je včelařství minoritním odvětvím potravinářské produkce, přesto je význam včel vyšší. Dnes již nejspíš nejsou včelaři, kteří by neměli chytrý telefon, proto by se tato práce měla zabývat implementováním již existujících aplikací pro analýzu obrazu k rozpoznání včelí matky a aplikace k zhodnocení míry napadení včelstva *Varroa destructor.*

Tato práce vznikla v návaznosti na předchozí dílčí práce vycházející z původního projektu Inteligentní včelí úl (IVÚ), který probíhá v rámci projektu Strategie AV 21, Rozmanitost života a zachování ekosystémů (ROZE) na Biologickém centru AV ČR, v. v. i. v Č. Budějovicích. Projekt se snaží vytvořit ideální skelet včelího úlu vhodného pro sběr dat za pomoci senzorů sledujících vlhkost a teplotu uvnitř úlu, dále měnící se hmotnost úlu, a zvukové záznamy včel. To vše by mělo vést k vytvoření softwaru pro ukládání a následné zpracování dat, a tím přispět k celkovému pochopení problematiky.

V jednotlivých částech této práce se autor zabývá nejprve popisem problematiky výskytu *Varroa destructor* a popisem včelího společenstva. Dále se autor v bakalářské práci zabývá problematikou implementace a sdružení již dříve vyvinutých aplikací vhodných pro detekci a počítání roztočů *Varroa destructor* a také využívajících metod počítačového vidění za účelem detekce a lokalizace včelí matky. Výše uvedené aplikace byly již dříve vyvinuty a nyní by následně v rámci této práce měla proběhnout jejich implementace do mobilní aplikace.

1.1. Motivace

Včelařství k naší společnosti neodmyslitelně patří a sehrává v ní nezastupitelnou roli. Ať v zemědělství, potravinářství, medicíně nebo kosmetice[10], [32], [22]. Včely jakožto blanokřídlí sociální druh hmyzu jsou chovány v úlech. Jedna včelí kolonie (včelstvo) může čítat až 50 000 jedinců, kteří se dělí do tří kast. Nejpočetněji jsou v úlu zastoupeny dělnice, ty se vyskytují ve včelstvu po celý rok. Dále se vyskytují ve včelstvu samci, tak zvaní trubci, ti jsou však v úlu pouze sezoně. Posledním členem tohoto společenství je matka, která se ve včelstvu, mimo rojení, vyskytuje pouze jedna[6].

V současné době je jednou z nejčastějších chorob včel napadení roztočem kleštíkem včelím (*Varroa destructor*). Pro správnou léčbu a prevenci tohoto onemocnění je velmi důležitá včasná detekce, kterou se zabývá i tato bc. práce [12], [24]. Informace o počtu jedinců *Varroa destructor* jsou stěžejní pro zvolení správných léčebných opatření[10].

Dále se tato práce zabývá dalším typem analýzy obrazu, která by měla usnadnit nalezení matky na plástu mezi ostatními včelami během práce včelaře v úlu. Během běžné práce včelaře hrozí umačkání některých jedinců. Pokud jde o rozmáčknutí trubce nebo dělnice, tak se jedná o přípustné ztráty, avšak pokud by zamáčklou obětí byla matka, jednalo by se o nenahraditelnou ztrátu[38]. Z tohoto důvodu včelaři po dobu práce v úlu matku odchytávají do krabičky a nechávají ji stranou. Pro snazší nalezení matky je možnost si ji na hrudi barevně označit, avšak tato metoda není v praxi běžně používána [6]. A právě nalezení neoznačené matky, by měla tato aplikace usnadnit.

1.2. Cíle práce

Cílem práce je prostudovat možnost použití mobilní aplikace na používaných klasických neuronových sítích, které by umožnily jednoduché, přesné a rychlé zpracování snímku měli z úlu a snímku rámku se včelami. Tato analýza obrazu by měla umožnit rozpoznání množství roztoče *Varroa destructor* na spadové desce, a tím ne přímo určit míru napadení včelstva. Dále by tato aplikace měla být schopna během snímání včelího rámku se včelami rozpoznat přítomnost včelí matky na daném snímku a případně ji označit. V případě úspěšné implementace by bylo možné používat mobilní aplikaci k analýze obrazu v praxi, a tím usnadnit včelaři vyhodnocení míry napadení roztočem a zvolení správného postupu léčby. Zároveň by tato aplikace mohla usnadnit hledání včelí matky na rámečku a zvýšit tak celkový komfort při práci v úlu. Pro splnění výše uvedeného byly stanoveny dílčí cíle této práce, které jsou v bodech rozepsány níže:

- Seznámit se s existujícími aplikacemi pro identifikaci roztoče *Varroa destructor* a identifikace matky na včelím plástu.
- Najít slabá místa aplikací, navrhnout a implementovat vlastní aplikaci.
- Implementovat aplikaci do systému Android.
- Vytvořit mobilní aplikaci za použití předcházejících závěrečných prací, která by odpovídala výkonnostním možnostem mobilních zařízení.
- Navrhnout intuitivní uživatelské rozhraní a podrobit ho uživatelskému testu.

2. Popis problematiky

2.1. Roztoč kleštík včelí (*Varroa destructor*)

Jednou z největších včelařských hrozeb současnosti je *V. destructor* (obrázek 1). Žádný jiný druh patogenu nemá tak silný dopad na tak velkou část populace včel v celosvětovém měřítku a takové míře. Je to způsobeno mimo jiné i tím, že včelaři nemají dostatečné množství zkušeností s tímto roztočem, jelikož se jedná o jeho relativně nedávnou expanzi do Evropy [11][23].

Obrázek 1: *Varroa destructor* [16]

2.1.1. Historie *Varroa destructor*

Roztoč *Varroa destructor* byl poprvé zjištěn na včele indické (*Apis cerana*) na ostrově Jáva entomologem E. Jacobsonem roku 1904. Na včele medonosné byla nákaza *Varroa destructor* objevena poprvé v Číně roku 1958. Z Asie se roztoč rozšířil do Evropy. Roku 1976 se dostal do Maďarska. Roku 1977 byl roztoč zjištěn v Německu, kam byl nešťastně přemístěn importem včelích matek z Pákistánu. Roku 1982 je objeven ve Francii. Na území bývalé Československé socialistické republiky pronikl roztoč *Varroa destructor* přirozenou cestou přes hranice z území Zakarpatské Ukrajiny roku 1978 [21]. Byl také nalezen na východním pobřeží bývalého SSSR (1952), v Pákistánu (1955), Japonsku (1958), Číně (1959), Bulharsku (1967), Jižní Americe (Paraguay-1971), Německu (1977) a USA (1987). Dnes je téměř kosmopolitní. Nebyl zatím nalezen v Austrálii[29].

2.1.2. Biologie *Varroa destructor*

U roztoče *V. destructor* je viditelný značný sexuální dimorfismus s mnoha morfologickými adaptacemi na hostitele[1]. Charakteristika stejná pro obě pohlaví je rozdělení těla na dvě části: gnathosoma a idiosoma. Idiosoma zahrnuje větší část a jeden hřbetní štít a různé břišní štíty[30]. Samičí idiosoma je širší než delší, zploštělá a elipsoidní 1,1 – 1,5 mm dlouhé a široké 1,5 – 1,9 mm s červenohnědým až hnědě lesklým zbarvením [38][9][21]. Končetiny samice jsou krátké a silné se strukturami, díky nimž se lépe přichytí k hostiteli. Hřbetní a břišní štíty jsou sklerotizované s červeno-hnědou barvou. Samci jsou pouze málo sklerotizováni a jsou menší než samice ve všech vývojových stádiích. Vzhledem k tělu jsou nohy samce delší než nohy samice [29][30].

Životní cyklus roztoče v laboratorních podmínkách může vykonat až 7 generačních cyklů. V přírodě jsou to ovšem 2 až 3 cykly. U včely *A. mellifera* probíhá reprodukce u trubčího plodu i u plodu dělnic. Určité procento samic roztoče nenaklade vajíčka. U evropských včel je 5 až 20 % roztočů neplodných, avšak důvod neplodnosti není znám [29][6]. Hlavní obětí roztoče *Varroa destructor* je stádium kukly. Roztoč napadá většinou trubčí populaci, a to během fáze kukly, čímž snižuje očekávanou délku života a schopnost letu. Míra napadení roztočem negativně souvisí s velikostí kolonie a s počtem dospělých trubců [8]. Pří nízkém napadení roje roztoči neparazitují na matečnících. Při větším zamoření včelstva, která obsahuje menší potomstvo dělnic a trubců, napadají roztoči i matečníky. Pro roztoče není žádná evoluční výhoda rozmnožovat se v matečnících, jelikož v nich mají méně dní pro vývoj než v trubčím plodu [15].

Detekce roztoče může být provedena různými způsoby. Jedním z nich je pozorování plodu. Dalším způsobem je technika, kdy se do sklenice vloží 100 - 200 včel a startovací roztok. Poté se se sklenicí třese a válí. Roztoči zůstanou na sklenici a mohou se snadno spočítat. Poslední nejvíce efektivní a používanou metodou jsou lepové destičky. Ty se vloží na dno úlu, který se vykouří. Poté se destička vyjme a sečte se množství roztočů, kteří na ní ulpěli. Alternativně se může destička pouze ponechat 1-3 dny v úlu [31], tato včasná diagnostika roztoče ve včelstvu je klíčová, k včasnému zastavení infekčního tlaku[6]. Vzhledem k časové náročnosti pro přepočítání ulpělých roztočů na lepové desce, se jeví

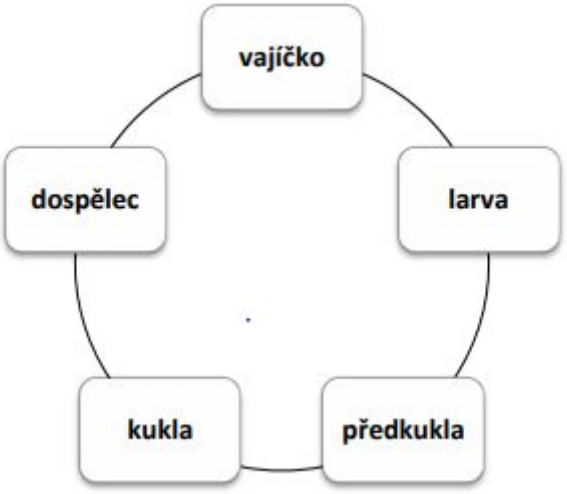
použití aplikace pro přesné lokalizování a sumarizování počtu ulpělých roztočů jako velmi vhodná. Nehledě na to, že včelaři nikdy nemají pouze jedno včelstvo, ale počty úlu se pohybují v řádech desítek.

2.2. Včela medonosná *(Apis mellifer*a)

Včely jsou nejdůležitější komerční opylovači plodin, jejichž reprodukce závisí na živočišném opylení, proto jsou důležité pro ekonomické a udržitelné zemědělství. Navíc opylují velké množství divokých rostlin, a tak přispívají k biodiverzitě [14]. V USA jsou včely primárním opylovačem a jejich vliv na kvalitu a výnos zemědělských plodin byl v roce 2000 odhadován na 14,6 miliard dolarů. Včely mají také medicinální vliv [7].

2.2.1. Biologie *Apis mellifera*

Včela medonosná (*A. mellifera*) je sociální hmyz žijící v koloniích. Každá kolonie obsahuje dvě samičí kasty: matku a dělnice, jejichž počet na kolonii se pohybuje od 20 000 do 50 000. Život v tomto společenstvu je řízen mnoha feromony vydávanými primárně včelí matkou. Včelstvo obsahuje také samčí složku, trubci, kteří slouží primárně k oplození matky[31].



Obrázek 2: Vývoj včely medonosné. Převzato z[31]:

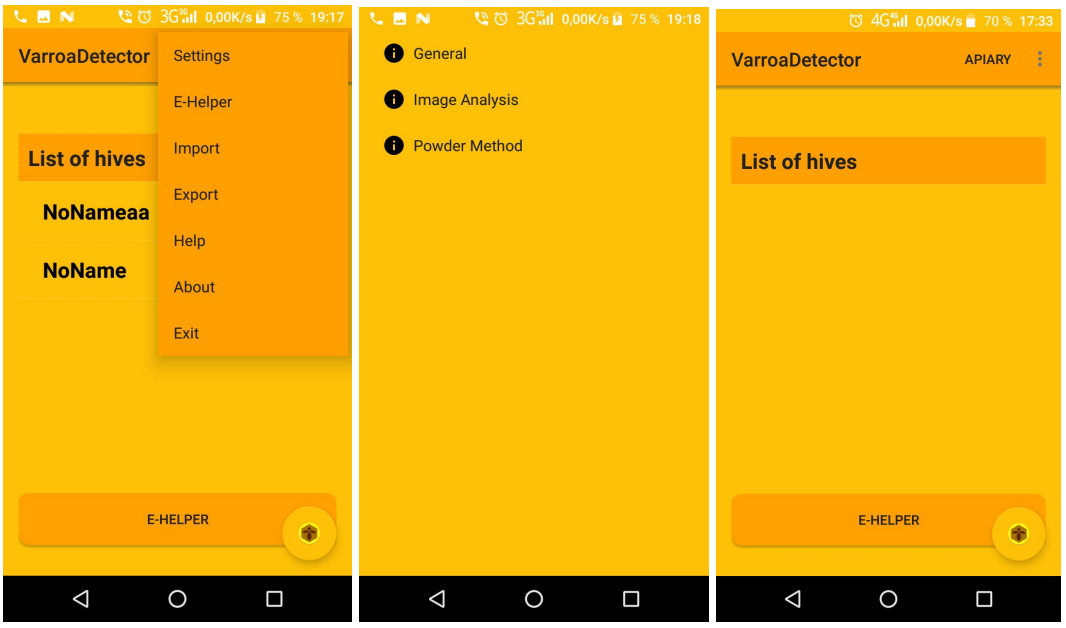
Včelí kolonie obsahuje několik kast a stádií vývoje včel (obrázek 2). Mladé dělnice v kolonii pečují o matku a potomstvo, starší dělnice vylétají pro pyl, nektar, vodu a ochraňují kolonii. Dalším typem včely je matka, jejíž vývoj trvá 16 dní a od ostatních včel se zjevně liší svojí velikostí, dosahuje 20-25 mm, dělnice se vyvijí 21 dní a trubci 24 dní, dosahují velikosti 15-17 mm. Matka dokáže přežít několik let, zatímco dělnice a trubci žijí pouze několik týdnů [31]. U dělnic se v mírném pásmu projevuje rozdíl v délce života, krátce žijící letní (krátkověké) a dlouho žijící zimní včely (dlouhověké)[4]. Morfologie a barevnost matky je oproti ostatním typům včel odlišná, matka má méně výrazné až neznatelné rozlišení světlých a tmavých pruhů na zadečku, na hrudi má tmavší štít a nohy jsou znatelně světlejší a delší než u ostatních včel [6].

Na základě výše uvedených rozdílů matky od dělnic by mělo být možné vytvořit kombinovanou aplikaci schopnu rozpoznat včelí matku na včelím dílu, a zároveň tuto aplikaci propojit s aplikací hodnotící míru napadení včelstva *Varroa destructor* z analýzy obrazu desky ze dna úlu.

3. Existující aplikace

V rámci této práce byly dohledány a otestovány některé již existující mobilní aplikace, jejich výčet a hodnocení je uvedeno níže. V současné době k dispozici jiné než tyto aplikace nejsou, z daného zkoumání vyplývá, že níže uvedené aplikace nejsou plně intuitivní a jejich obslužnost není tím pádem komfortní. Dále při studiu problematiky mobilních aplikací bylo dosaženo zjištění, že již jsou vyvíjeny další mobilní aplikace zabývající se podobnou problematikou. Například aplikace Pollenity by měla být schopna rozpoznat napadení roztočem a identifikovat včelí matku na včelím díle mezi ostatními typy včel. [35]

3.1. Varroa Detektor



Obrázek 3: úvodní plocha do aplikace Varroa Detektor a následně plochy umožňující různé nastavení citlivosti aplikace [37].

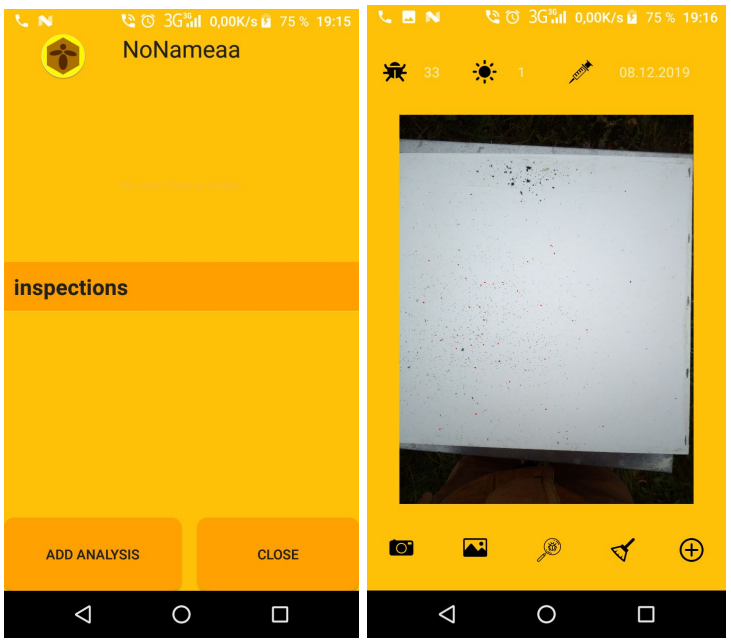
Po spuštění aplikace se objeví prázdná plocha, kde v budoucnu bude seznam včelínů a úlu majitele aplikace. Kliknutím na apiary se dostaneme do sekce přidávání včelínů a úlů, kde lze přidat nebo odebrat název včelínu, nebo úlů. V případě, že si přejeme nějaký přidaný úl, případně včelín, odebrat ze seznamu, podržíme displej delší dobu a zobrazí se možnost smazání vybrané položky (obrázek 3).

Nastavení: V nastavení máme možnost nastavovat obecné parametry, pro analýzu obrazu při různých způsobech sběru roztoče, například prachovou metodu, kde se mimo jiné zadává i váha naprášeného cukru.

General (obecné parametry): Zde se zadává druh zaléčení, typ úlu, a logování roztočů.

Image analysis: Zde se dá volit typ detektoru, typ hran, černobílé hrany, chytré hrany, chytrý faktor.

3.1.1. Analýza pomocí kontur



Obrázek 4: Analýza obrazu v aplikace Varroa Detektor [37].

Kliknutím na jméno úlu přicházíme k možnosti analýzy obrázku (obrázek 4). Při analýze counter detector (která trvala pár vteřin i při nastavení maxima – 99 bodů) byl nalezen u některých testovacích fotografií pouze 1 kleštík, jindy žádný. V případě, že nejsou označeni všichni kleštíci na snímku, je možné je kliknutím dodatečně označit, v případě chybného označení je možné toto označení odebírat. Pokud dojde k záměně roztoče s nečistotou, aplikace umožňuje toto chybné označení zrušit tak, aby byl konečný počet správný. Následně se po přidání výsledné analýzy dostáváme zpět do nabídky pro úl, kde je již přidaná hodnota a graf vyjadřující množství kleštíků.

3.1.2. Analýza přes neuronovou síť

V rámci aplikace je možno v nastavení přepnout z analýzy založené na technice rozpoznávání pomocí kontur obrazu, na analýzu založenou na neuronové síti. Pří změně na tento typ metody byla zaznamenána vyšší úspěšnost, avšak stále nedostatečná. Nález činil pouze 0 – 11 roztočů z konečného počtu který se pohyboval v intervalu 50 – 60 jedinců na dané ploše. Dále bylo provedeno nastavení v image z metody analýzy okraje na typ analýzy

canny, u výsledku takto změněného nastavení bylo zaznamenáno opět zlepšení, ale stále se pohybujeme velmi daleko od skutečného počtu roztočů. U nastavení typu oats docházelo ke znatelně lepšímu rozpoznání jednotlivých těl roztoče, než u typu canny. U některých testů v rámci opakovaného testování, výsledek nabýval hodnot 30 jedinců na zkoumaný snímek, to však stále činilo necelých 60% ze skutečného počtu.

3.1.3. Zhodnocení aplikace Varroa Detektor

Pro analýzu je v popisu dané aplikace uvedeno používání neuronové sítě tensorflow. V ramci zkoumané aplikace je pro uživatele až příliš mnoho parametrů k nastavování. Pro obyčejného uživatele, který se nezabývá touto technikou, může být nastavování parametrů příliš složité a zmatečné. Kladnou kompenzací nedostatečné schopnosti rozpoznávat jednotlivé roztoče pomocí neuronové sítě, je možnost manuálního lokalizování zaměřených roztočů.

3.2. Varroa Counter

Obrázek 5: Úvodní plocha do aplikace Varroa Counter [36].

Jako další aplikace, kterou je možno stáhnout přes Google play na českém trhu je aplikace Varroa Counter (obrázek 5). Tato aplikace pro mobilní telefon však oproti již výše

hodnocené aplikaci roztoče prakticky nerozpozná, z tohoto důvodu její popis není zpracován tak podrobně, jako u aplikace Varroa Detektor.

Hned při prvním spuštění této aplikace zjišťujeme její první nedostatek - aplikace je pouze v německém jazyce. Dále nemá možnost načtení již vyfoceného obrázku z úložiště v mobilním zařízení. Pro tuto aplikaci je nutné vytvářet vždy nové snímky, u kterých je zapotřebí přednastavit dva parametry - šířka a výška focené plochy, jinak aplikace nefunguje pro nastavení tohoto parametru. Je zde v paměti místo pro 10 přednastavení.

Při testování aplikace bylo provedeno focení dvou variant ke zkoušce přesnosti aplikace. První varianta byla s větším výskytem jedinců roztočů, kde jejich počet byl i přes 60 jedinců. Aplikace poukazovala na 23 nalezených jedinců roztočů, zároveň zaměřila i jiné částice na snímku desky. Ukázalo se, že její přesnost není příliš vysoká. Druhý test byl zaměřen na přesný počet kleštíků, a to 20. Jedinců na snímku, v prvním opakování, aplikace rozpoznala 15. V druhém opakování nerozpoznala 3 jedince z 20 a ve třetím opakování bylo identifikováno 9 jedinců, a to při stejných světelných podmínkách a nastavení fotoaparátu. Z výše uvedených výsledků vyplývá, že aplikace má značné nedostatky a není vhodná k funkci, pro kterou je určena.

3.3. BeeScanning - diagnose bees health[5]

Obrázek 6: Úvodní plocha do aplikace BeeScanning - diagnose bees health [5] .

Tato aplikace se nepodařila spustit. Po startu aplikace bylo požadováno zadání emailové adresy pro ověření a přihlášení uživatele. Po zadání emailu pro přihlášení je po uživateli vyžadováno potvrzení, které bylo aplikací zasláno na email. I přes opakované potvrzení přes email se nepodařilo přihlásit do aplikace. Z tohoto důvodu nebylo možné dále tuto aplikace testovat. Dalším důvodem pro vyřazení aplikace z testu bylo špatné hodnocení na Google play a to, že aplikace by měla pracovat s fotografiemi celé plástve se včelami a popisovat jejich stav případně identifikovat matku a ne jen fotografovanou výseč plástve.

4. Hardware

Pro co nejlepší detekci roztoče *Varroa destructor* nebo včelí matky je zapotřebí vysokého rozlišení pořizovaného snímku, vysoké výpočetní schopnosti a velké operační

paměti zařízení, které bude daný snímek vyhodnocovat. Ve všech uvedených směrech jsou parametry u mobilního zařízení limitovány. Z toho důvodu je zapotřebí nalézt optimální řešení, pomocí kterého bude možné docílit dostatečně dobrých výsledků analýzy obrazu a následné detekce výše uvedených objektů zájmu i na mobilním zařízení.

Pro testování aplikace a pořizování snímků spadu kleštíků na desku bylo použito následující zařízení [13]

Název zařízení: EVOLVEO StrongPhone G4

Parametry: Čtyřjádrový4G/LTE Dual SIM telefon, IP68, 1.4 GHz, 3 GB RAM, 32 GB vnitřní paměť, WiFi/WiFi HotSpot, Full video, baterie 3 000 mAh, MIL-STD-810G:2008, Android 7.0, 13.0 Mpx (8.0 Mpx optické rozlišení).

5. Software

5.1. OpenCV

OpenCV je open sourceová knihovna pod licencí BSD, která slouží ke strojovému vidění a strojovému učení. Nativně je psána v C++. Zároveň lze využít v rozhraní Java, Python a MATLAB a podporuje operační systémy Windows, Linux, Android a Mac OS [27].

5.1.1. Implementace

Implementace OpenCV byla provedena podle návodu na stránkách OpenCV4Android. Importoval se modul OpenCV. Smazala se minimální verze v Manifestu. V souboru gradle se u app a u OpenCV nastavily stejné minimální a maximální verze. V project struct se v dependencies přidala opencvLib. Vytvořila se složka JIN, do které se zkopíroval zdrojový kód OpenCV (složka JIN slouží k zprostředkování nativního kódu C přes interface do Javy) [28][20].

5.2. Neuronové sítě - TFLite

Neuronová síť je soubor neuronů a jejich spojů. Neuron je výpočetní jednotka, která je propojena s jiným neuronem jednosměrným spojem, které jsou ohodnoceny vahami. Díky učení (přizpůsobení vah) se umožňuje realizovat kvalitativně novou funkci na základě

trénovacích vzorků. Nezbytnou vlastností neuronových sítí, kromě učení (nacházení závyslostí v trénovacích datech), je zevšeobecňování (generalizace) získaných poznatků. Tato schopnost znamená, že neuronová síť bude reagovat i na vstupy, se kterými dříve neuronová síť nepracovala[25][26].

Tensorflow je open sourceová platforma pro strojové učení vyvíjena společností Google. Tensorflow může běžet najednou na více procesorech a GPU s rozšířením CUDA a SYCL pro zpracování grafiky. Podporuje 64bitové systémy Linux, macOS, Windows a mobilní systémy Android a IOS. Výpočty Tensorflow jsou vyjádřeny přes stavové grafy datových toků. Tensorflow provádí výpočty na více úrovňových datových polích, které se nazývají tenzory [34]{TensorFlow.}

V této práci je využito Tensorflow Lite, což je balíček vytvořený přímo pro vývoj pro mobilní zařízení. Díky tomu je umožněno využít strojového učení i na zařízeních s pomalou odezvou a malou operační pamětí {TensorFlow.}

5.2.1. Model

Pro užití této neuronové sítě je potřeba užít modelu ve formátu tflite. Tento formát se nedá získat učením, ale konvertováním již naučeného modelu. Jedná se pouze o optimalizaci modelu. Pro konvertování se dají využít modely učené pomocí kerasu nebo pomocí tf.keras. Ke konverzi byl v této práci využit kód 1.[33]

```
import tensorflow as tf
kerasModelPath = 'varroa.model'
tfliteModelPath = 'model.tflite'

model = tf.keras.models.load_model(kerasModelPath, compile=False)
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tfmodel = converter.convert()
open (tfliteModelPath , "wb") .write(tfmodel)
```

Kód 1: convertování tflite

5.2.2. Implementace

Rozpoznání pomocí neuronové sítě je zprostředkováno pomocí knihovny tensorflow lite. Je zde využita poskytovaná tensorflow lite AAR knihovna poskytovaná přes JCenter, přidáním implementace do dependencies v souboru gradle a zároveň je snaha ušetření paměti pomocí abiFilteru v základním nastavení viz kód 2, 3. AbiFilter funguje, jak již název napovídá, jako filtr, kdy filtruje instrukční sady pro málo používaná CPU. Je zakázána komprese souborů s příponou tflite aby byly možné použít modely tohoto formátu viz. kód 4.

```
implementation 'org.tensorflow:tensorflow-lite:0.0.0-nightly'
implementation 'org.tensorflow:tensorflow-lite-gpu:0.0.0-nightly'
```

Kód 2: implementace tflite

```
ndk {
    abiFilters 'armeabi-v7a', 'arm64-v8a'
}
```

Kód 3: abiFilter

```
aaptOptions {
    noCompress "tflite"
}
```

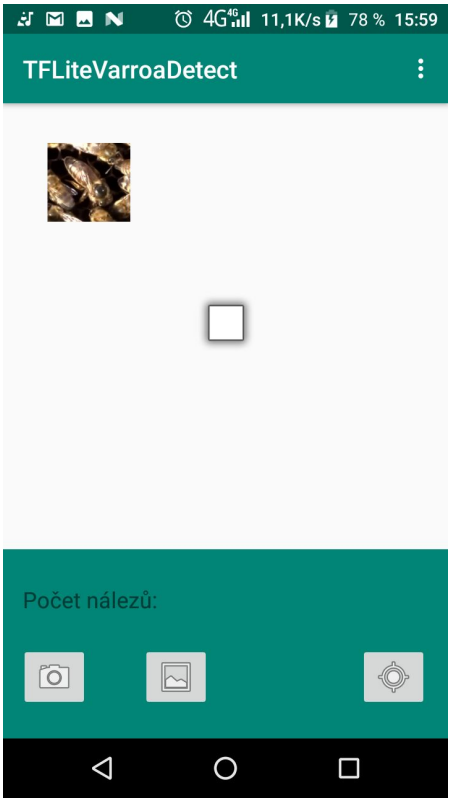
Kód 4: zákaz komprese

5.3. Android Studio

Android Studio je vývojové prostředí vytvořeno firmou Google za spolupráce JetBrains. Vývojové prostředí je založené na IntelliJ IDEA, což zjednodušuje a optimalizuje pracovní postup při vývoji v tomto prostředí. Pro psaní kódu bylo využito programovacího jazyku Java[3]

6. Prostředí

Při prvním spuštění aplikace je uživatel požádán o povolení práv přístupu k úložišti telefonu a práci s kamerou (kód 5). Po spuštění aplikace se uživatel dostane do hlavní části aplikace (obrázek 7). Zde si může nahrávat snímky kleštíků pro rozpoznání, přepínat mezi jednotlivými snímky zpracování, přepínat mezi modely pro rozpoznávání, posouvat snímek pro lepší náhled na výsledek a spouštět aktivitu pro nalezení včelí matky.



Obrázek 7: hlavní aktivita

```
private boolean hasCameraStoragePermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        return checkSelfPermission(PERMISSION_CAMERA) == PackageManager.PERMISSION_GRANTED &&
                checkSelfPermission(PERMISSION_READ_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED &&
                checkSelfPermission(PERMISSION_WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED;
    } else {
        return true;
    }
}

private void requestCameraStoragePermission() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        if (shouldShowRequestPermissionRationale(PERMISSION_CAMERA)) {
            Toast.makeText(
                    context: MainActivity.this,
                    text: "Camera permission is required for this app",
                    Toast.LENGTH_LONG)
                    .show();
        }
        requestPermissions(new String[] {PERMISSION_CAMERA,PERMISSION_READ_EXTERNAL_STORAGE,PERMISSION_WRITE_EXTERNAL_STORAGE}, PERMISSIONS_REQUEST);
    }
}
```

Kód 5: uživatelské povolení

6.1. Načítání snímků

Načtení snímku se dá provést dvěma způsoby, ke kterým slouží ikony v levém dolním rohu (obrázek 7).

První způsob je možný po kliknutí na ikonu fotoaparátu. Následně se zavolá metoda fullImgCapture pro nastavení parametrů pro kameru. Kamera se spustí a tou uživatel snímek vyfotí (kód 6).

Druhý způsob je formou načtení snímku z úložiště zařízení. Toho se dá docílit kliknutím na ikonku obrázku. Spustí se metoda startFindPathToPhotoFile, nastaví se spuštění průzkumníka a spustí se pro výběr snímku (kód 6).

```
private void startFindPathToPhotoFile(){
    Intent intentFilePath = new Intent(Intent.ACTION_GET_CONTENT);
    intentFilePath.setType("*/*");
    startActivityForResult(intentFilePath, REQUESTCODE_GET_PHOTO_CONTENT);
}

private void fullImgCapture () {
    ContentValues values = new ContentValues();
    values.put(MediaStore.Images.Media.TITLE,"newImg");
    values.put(MediaStore.Images.Media.DESCRIPTION,"capturetImg");
    imgUri = getContentResolver().insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,values);

    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT,imgUri);
    startActivityForResult(takePictureIntent,REQUESTCODE_IMAGE_PATH);
}
```

Kód 6: Načtení snímku uložiště/foto

Po získání snímku se zavolá metoda onActivityResult, kde se pomocí uri vyčte ze zařízení snímek. Snímek se hned po načtení pošle k rozpoznání kleštíků a výsledek ve formě obrázku, a počtu rozpoznaných kleštíků, se zobrazí na obrazovce (kód 7). Během průběhu rozpoznávání by bylo dobré vytvořit ukazatel průběhu rozpoznávání. Při velkém množství kontur totiž dochází k prodlevě, obrazovka zčerná a uživatel neví, kdy bude snímek ukázán.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (resultCode == RESULT_OK) {
        switch (requestCode){
            case REQUESTCODE_GET_PHOTO_CONTENT:
                imgUri= data.getData();
            case REQUESTCODE_IMAGE_PATH:
                try {
                    varroaBitmap = MediaStore.Images.Media.getBitmap(getContentResolver(),imgUri);
                    imgViewCamera.setImageBitmap(varroaBarvaPoznat.recognizeVarroaByColor(varroaBitmap));
                    textPointCount.setText(String.valueOf(varroaBarvaPoznat.getLastCountDetecVarr()));

                } catch (IOException e) {
                    e.printStackTrace();
                }
                break;
            case REQUESTCODE_GET_MODEL_CONTENT:

                break;
        }
    }
    resetImgPosition();
}
```

Kód 7: získané snímky

6.2. Posun snímku

Pro lepší náhled na obrázek byl přidán posun a možnost přiblížení. Na tuto funkcionalitu nebyla využita žádná cizí knihovna. To s sebou nese i některé nedostatky, které se nestihly v této práci dořešit, jako např. posun obrázku může být proveden do velké vzdálenosti, kdy může dojít ke ztrátě snímku v posunu. Tyto nedostatky byla snaha alespoň částečně odstranit formou přidání tlačítka, které nastaví obrázku základní pozici a velikost. Tlačítko je v pravém dolním rohu se znakem pozice (obrázek 7). V (kódu 8) je znázorněn postup procesu posunu a zvětšení. V metodě onToucheEvent jsou zaznamenávány veškeré aktivity dotyku spojené s obrazovkou. V případě, že dojde ke gestu pro změnu velikosti, provede se metoda onScale, kde se přepočítá měřítko velikosti a podle toho se velikost

nastaví. Pro posun je zde využito větvení, které pomůže při zjištění prvního kontaktu s obrazovkou (ACTION_DOWN) zaznamenat počáteční body xy a konečné puštění obrazovky (ACTION_UP), kdy je vypočítán posun a změna pozice snímku. Je zde přidáno ošetření, kdy v případě, že byla provedena změna přiblížení, nedojde k úpravě pozice snímku.[2]

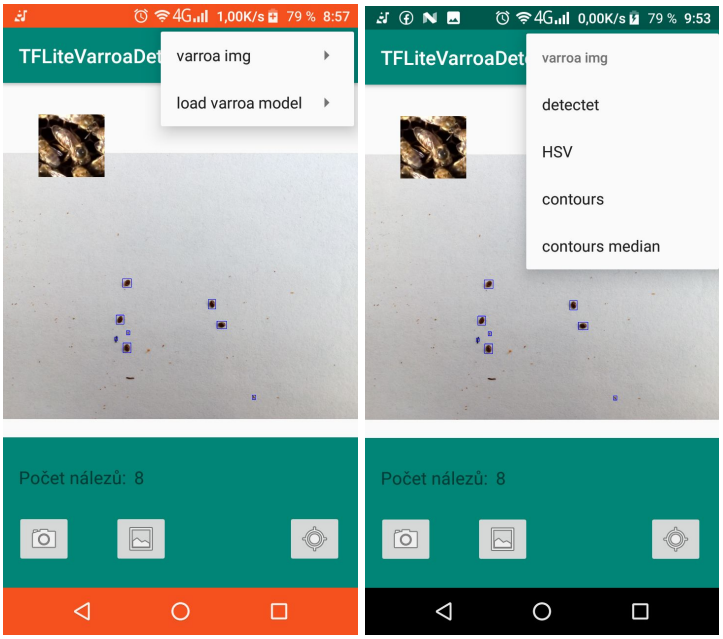
```
@Override
public boolean onTouchEvent(MotionEvent motionEvent) {

    float motionX = motionEvent.getX();
    float motionY = motionEvent.getY();

    float diffX = motionX-previousX;
    float diffY = motionY-previousY;

    xCounter += diffX;
    yCounter += diffY;
    scaleGestureDetector.onTouchEvent(motionEvent);
    if (prevScaleX == imgViewCamera.getScaleX())
        switch (motionEvent.getAction()){
            case MotionEvent.ACTION_DOWN:
                previousX = motionX;
                previousY = motionY;
                break;
            case MotionEvent.ACTION_UP:
                if(previousX!=0&&previousY!=0)
                    //todo osetreni posunu obrazku aby nezmizel do nenavratna
                    // todo v pripade nacteni noveho obrazku nastavit defoultni pozici (det nastavuje levy roh obrazku)
                    imgViewCamera.setX(imgViewCamera.getX()+motionX-previousX);
                imgViewCamera.setY(imgViewCamera.getY()+motionY-previousY);
                break;
        }
    prevScaleX = imgViewCamera.getScaleX();
    return true;
}

private class ScaleListener extends ScaleGestureDetector.SimpleOnScaleGestureListener {
    @Override
    public boolean onScale(ScaleGestureDetector scaleGestureDetector) {
        mScaleFactor *= scaleGestureDetector.getScaleFactor();
        mScaleFactor = Math.max(0.1f, Math.min(mScaleFactor, 10.0f));
        imgViewCamera.setScaleX(mScaleFactor);
        imgViewCamera.setScaleY(mScaleFactor);
        return true;
    }
}
```

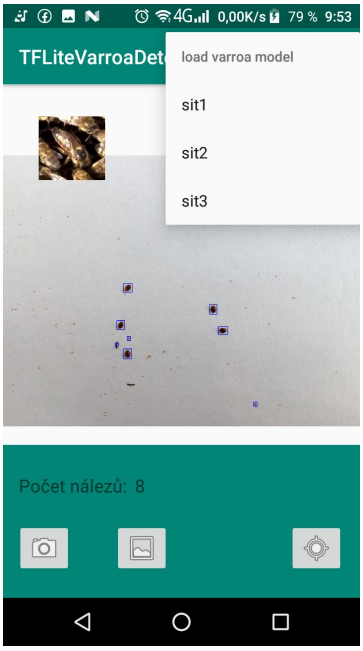
Kód 8: posun a zvětšení snímku

6.3. Menu

V pravém horním rohu je přidáno menu, kde je možné získat náhledy na obrázky vytvořené v průběhu rozpoznávání snímku nebo pro nahrání jiného modelu pro rozpoznávání kleštíků (obrázek 7, 8).

Menu je řešeno větvením. Po stisknutí volby obrázku se z instance třídy pro rozpoznání kleštíků navrátí buď zpracovaný obrázek, který se zobrazí v image view, nebo se navrátí hodnota null v případě, že instance dosud žádný snímek nezpracovávala. V případě, že se navrátila hodnota null, tak se zobrazí hláška že daný obrázek neexistuje. Při zvolení jiného modelu se vytvoří nová instance s novým modelem, a jelikož je prázdná je třeba ji nahrát nový snímek pro zpracování. Celé řešení viz kód 9.[2]





Obrázek 8: menu

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()){
        case R.id.mhsvimg:
            varroaBitmap = varroaBarvaPoznat.getImgHSV();
            if(varroaBitmap!=null)
                imgViewCamera.setImageBitmap(varroaBitmap);
            else
                makeTitle ( text: "Obrazek neexistuje");
            break;
        case R.id.mcontours:
            varroaBitmap = varroaBarvaPoznat.getImgConturs();
            if(varroaBitmap!=null)
                imgViewCamera.setImageBitmap(varroaBitmap);
            else
                makeTitle ( text: "Obrazek neexistuje");
            break;
        case R.id.mcontoursmedian:
            varroaBitmap = varroaBarvaPoznat.getImgContursMedian();
            if(varroaBitmap!=null)
                imgViewCamera.setImageBitmap(varroaBitmap);
            else
                makeTitle ( text: "Obrazek neexistuje");
            break;
        case R.id.mdetectet:
            varroaBitmap = varroaBarvaPoznat.getDetectedVarroa();
            if(varroaBitmap!=null)
                imgViewCamera.setImageBitmap(varroaBitmap);
            else
                makeTitle ( text: "Obrazek neexistuje");
            break;
        case R.id.sit1:
            loadVarroaDetect(LABELS,SIT1);
            break;
        case R.id.sit2:
            loadVarroaDetect(LABELS,SIT2);
            break;
        case R.id.sit3:
            loadVarroaDetect(LABELS,SIT3);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

Kód 9: řešení menu

6.4. Prostředí pro nalezení včelí matky

V levém horním rohu je tlačítko se včelí matkou kterým se spouští aktivita pro rozpoznání matky na plástu mezi včelami. Aktivita využívá knihovny openCV, přesněji implementuje třídu CvCameraViewListener2. To umožňuje přístup k výstupu kamery zařízení po jednotlivých framech. Toho je využíváno pro rozpoznání včelí královny, kdy se snímek pošle rozpoznat do klasifikátoru od tflite. Část řešení kódu k nahlédnutí viz kód 10.

Tato funkcionalita je zatim nedořešená z důvodu problému s modelem pro rozpoznání včelí matky. O této problematice více viz kapitola Proces rozpoznání včelí matky. Nicméně po spuštění této aktivity uživatel uvidí spuštěnou kameru, kde ve finální verzi by aplikace při pohledu na včelí matku měla dokázat rozeznat a označit.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_bee_detect);
    recognitions = new ArrayList<>();
    cameraView = (JavaCameraView) findViewById(R.id.cameraView);
    cameraView.setVisibility(SurfaceView.VISIBLE);
    cameraView.setCvCameraViewListener(BeeDetect.this);
    try{
        this.classifier = Classifier.create( activity: this, Classifier.Model.FLOAT, Classifier.Device.CPU, numThreads: 1, labelpath: "beeLabels.txt", modelpath: "advancedModel.tflite");
    }catch (Exception e){
        Log.d(TAG,e.getMessage());
    }
}

@Override
public void onCameraViewStarted(int width, int height) {
    bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
    outputMat = new Mat(width,height, CvType.CV_16UC4);
}

@Override
public void onCameraViewStopped() {

}

@Override
public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
    outputMat = inputFrame.rgba();
      Utils.matToBitmap(inputFrame.rgba(),bitmap);
      recognitions = classifier.recognizeImage(bitmap);
      Utils.bitmapToMat(bitmap,outputMat);
      for(Classifier.Recognition recognition: recognitions){
          if(recognition.getTitle().equals("queenbee"))
          Imgproc.rectangle(outputMat,new Point(recognition.getLocation().left,recognition.getLocation().bottom),new Point(recognition.getLocation().right, recognition.getLocation().top),new Scalar(0, 0, 255),3);
      }
    return outputMat;
}
```

Kód 10: Aktivita rozpoznání včelí matky

7. Řešení rozpoznávání

K rozpoznávání kleštíků a včelí matky je zde využíváno příkladu tříd od Tensor Flow (TF). Třídy od TF byly upraveny pro potřeby této aplikace. Pro možnost zaměňování modelů v aplikaci byly interní parametry nahrazeny parametry zadávanými při tvorbě instance Klasifikátou, a to parametry labels a model. Jsou zde vkládány v podobě textového řetězce, který slouží při načítání modelu a označení, jako cesta v assetech aplikace. Dále zde bylo přídáno zadávání velikosti pro úpravu snímků při rozpoznávání. Tedy v případě, že si přejeme, aby třída pracovala s obrázkem, tak jak byla navržena. Není potřeba zadávat žádné parametry velikosti snímku, pokud však se budou rozpoznávat snímky menší než je snímek z celé obrazovky. V opačném případě, kdy je potřeba rozpoznat menší obrázek na který byl vytvořen model, je potřeba upravit velikost snímku. To se udělá zadáním parametrů výšky a šířky požadované pro rozpoznávání viz kód 11.

```java
/**
 * Creates a classifier with the provided configuration.
 *
 * @param activity The current Activity.
 * @param model The model to use for classification.
 * @param device The device to use for classification.
 * @param numThreads The number of threads to use for classification.
 * @return A classifier with the desired configuration.
 */
public static Classifier create(Activity activity, Model model, Device device, int numThreads, String labelpath, String modelpath)
    throws IOException {
  if (model == Model.QUANTIZED) {
    return new ClassifierQuantizedMobileNet(activity, device, numThreads, labelpath, modelpath);
  } else {
    return new ClassifierFloatMobileNet(activity, device, numThreads, labelpath, modelpath);
  }
}

public static Classifier create(Activity activity, Model model, Device device, int numThreads , String labelpath, String modelpath, int modelImgH, int modelImgW)
        throws IOException {
  if (model == Model.QUANTIZED) {
    return new ClassifierQuantizedMobileNet(activity, device, numThreads, labelpath, modelpath, modelImgH, modelImgW);
  } else {
    return new ClassifierFloatMobileNet(activity, device, numThreads, labelpath, modelpath, modelImgH, modelImgW);
  }
}
```

Kód 11: vytvoření klasifikátoru.

Po takto vytvořené instanci se již vždy před rozpoznáním upraví každý vstupní obrázek na požadovanou velikost díky funkci resizeBitmap (kód 12) a díky tomu může být dále rozpoznán.

```
private Bitmap resizeBitmap(final Bitmap bitmap){
  Bitmap outBitmap = Bitmap.createBitmap(this.modelImgW, this.modelImgH, Bitmap.Config.ARGB_8888);

  float originalWidth = bitmap.getWidth();
  float originalHeight = bitmap.getHeight();

  Canvas canvas = new Canvas(outBitmap);

  float scale = this.modelImgW / originalWidth;

  float xTranslation = 0.0f;
  float yTranslation = (this.modelImgH - originalHeight * scale) / 2.0f;

  Matrix transformation = new Matrix();
  transformation.postTranslate(xTranslation, yTranslation);
  transformation.preScale(scale, scale);

  Paint paint = new Paint();
  paint.setFilterBitmap(true);

  canvas.drawBitmap(bitmap, transformation, paint);

  return outBitmap;
}

/** Runs inference and returns the classification results. */
public List<Recognition> recognizeImage(final Bitmap bitmap) {
  // Log this method so that it can be analyzed with systrace.
  Trace.beginSection( sectionName: "recognizeImage");
  Bitmap myBitmap;
  //provedeni resize v pripade varroa
  if(modelImgH!=DEFAULT_IMG_SIZE||modelImgW!=DEFAULT_IMG_SIZE){
    myBitmap = resizeBitmap( bitmap);
  }else{
    myBitmap = bitmap;
  }
  Trace.beginSection( sectionName: "preprocessBitmap");
  convertBitmapToByteBuffer(myBitmap);
  Trace.endSection();

  // Run the inference call.
  Trace.beginSection( sectionName: "runInference");
  long startTime = SystemClock.uptimeMillis();
  runInference();
  long endTime = SystemClock.uptimeMillis();
  Trace.endSection();
  //LOGGER.v("Timecost to run model inference: " + (endTime - startTime));
```

Kód 12: klasifikace

8. Proces rozpoznání kleštíků

Pro rozpoznání kleštíků byla vytvořena třída RecognizeVarroa. Kleštíci jsou rozpoznáváni pomocí metody popsané v práci pana Chlubny a byli k ní využity i jeho

vytvořené modely pro rozpoznávání, které byly překonvertovány pomocí kódu č. 1. Snímek je převeden z modelu RGB do modelu HSV, následně je pomocí povoleného rozsahu barev vytvořena maska, která vyznačuje podezřelé body na snímku. Pro omezení počtu těchto bodů je použitý medián, který vyčistí masku od příliš malých částic. A z výsledné masky se získají kontury viz kód 13.[18]

```
private  List<MatOfPoint> getContours(Mat inputMat){
    List<MatOfPoint>outputContours = new ArrayList<>();

    Imgproc.cvtColor(inputMat,imgHSV,Imgproc.COLOR_BGR2HSV);
    Core.inRange(imgHSV,lowerH,upperH,mask);
    Imgproc.medianBlur(mask,median, ksize: 9);
    Imgproc.findContours(median,outputContours,hierarchy,Imgproc.RETR_EXTERNAL,Imgproc.CHAIN_APPROX_SIMPLE);

    return outputContours;
}
```

Kód 13: získání podezřelých bodů

Tyto kontury (podezřelé body) jsou dále jedna po druhé rozpoznávány pomocí tflite klasifikace a souřadnice kontur, které jsou touto klasifikací rozpoznány jako kleštíci, jsou zaznamenány do Seznamu. Po vytvoření tohoto seznamu jsou veškeré pozice rozpoznaných kleštíků označeny rámečkem na vstupním snímku viz kód 14.

```
for (MatOfPoint c:
        contours) {
    detected = Imgproc.boundingRect(c);
    y = detected.y;
    x= detected.x;
    w = detected.width;
    h = detected.height;
    if ((x > 0) & (y > 0) & (w > 0) & (h > 0) ){
        //getting surroundings of object
        x -= 10;
        y -= 10;
        w += 20;
        h += 20;
        //getting the new object - obrazek nove varroy
        //* parametr oriznuti
        int inMatH = inputMat.height();
        int inMatW = inputMat.width();
        if(x+w > inMatW){
            x= inMatW-w-1;
        }
        if(y+h > inMatH){
            y= inMatH-h-1;
        }
        if(x<0){
            x=0;
        }
        if(y<0){
            y=0;
        }
        roi = new Rect(x, y, w, h);

        cropped = new Mat(inputMat, roi);
        imgVarroa = Bitmap.createBitmap(cropped.width(), cropped.height(), Bitmap.Config.ARGB_8888 );
        Utils.matToBitmap(cropped,imgVarroa);
        listRecognition = (ArrayList<Recognition>) this.classifier.recognizeImage(imgVarroa);

        //zjisteni varroy
        if(listRecognition.get(0).getTitle().equals("varroa")){
            detectedVarr.add(roi);
        }
    }
    detectedVarroaCount = detectedVarr.size();
    for (Rect place:detectedVarr
         ) {
        Imgproc.rectangle(inputMat,new Point(place.x,place.y),new Point( x: place.x + place.width,  y:
    }
```

Kód 14: rozpoznání kleštíků

9. Proces rozpoznání včelí matky

Pro rozpoznání včelí matky bylo zamýšleno využít modelů pro rozpoznávání pana Humpála, který měl navržené rozpoznávání pomocí dvou úrovní. První bylo zjištění, zda se

matka na snímku vyskytuje a druhá část byla její přesná lokalizace. Bohužel se tyto modely nepodařily překonvertovat. Proto byl zvolen vlastní vytvoření modelů pro rozpoznání včelí matky.[17]

K vytvoření modelu byly využity fotografie a roztřídění pana Humpála na včely, matku a negativní. Pro vytvoření bylo využito postupu na Colab. Byly vytvořeny tři modely přetrénováním již existujících modelů.[17][19]

První byl vytvořen jako základní model. Obrázky byly rozděleny na trénovací a testovací v poměru 9:1 a model prošel pěti epochy viz obr. 9.

U druhého byly obrázky rozděleny na trénovací, validační a testovací v poměru 8:1:1 a prošlo se pěti epochy viz obr. 10.

U třetího řešení došlo k rozdělení obrázků na trénovací validační a testovací opět v poměru 8:1:1. Tentokrát se však použije jiný prvotní model a projde se deseti epochy viz obr.. 11

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hub_keras_layer_v1v2 (HubKer (None, 1280)              3413024
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 3)                 3843
=================================================================
Total params: 3,416,867
Trainable params: 3,843
Non-trainable params: 3,413,024
_________________________________________________________________
None
INFO:tensorflow:Retraining the models...
INFO:tensorflow:Retraining the models...
Epoch 1/5
108/108 [==============================] - 9s 86ms/step - loss: 0.5650 - accuracy: 0.8626
Epoch 2/5
108/108 [==============================] - 9s 85ms/step - loss: 0.4489 - accuracy: 0.9306
Epoch 3/5
108/108 [==============================] - 9s 86ms/step - loss: 0.4243 - accuracy: 0.9424
Epoch 4/5
108/108 [==============================] - 9s 85ms/step - loss: 0.4237 - accuracy: 0.9485
Epoch 5/5
108/108 [==============================] - 9s 85ms/step - loss: 0.4107 - accuracy: 0.9560
```

Obrázek 9: Základní model

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hub_keras_layer_v1v2_1 (HubK (None, 1280)              3413024
_________________________________________________________________
dropout_1 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 3843
=================================================================
Total params: 3,416,867
Trainable params: 3,843
Non-trainable params: 3,413,024
_________________________________________________________________
None
INFO:tensorflow:Retraining the models...
INFO:tensorflow:Retraining the models...
Epoch 1/5
96/96 [==============================] - 11s 115ms/step - loss: 0.6172 - accuracy: 0.8551 - val_loss: 0.4418 - val_accuracy: 0.9297
Epoch 2/5
96/96 [==============================] - 10s 109ms/step - loss: 0.4585 - accuracy: 0.9245 - val_loss: 0.4189 - val_accuracy: 0.9427
Epoch 3/5
96/96 [==============================] - 11s 111ms/step - loss: 0.4420 - accuracy: 0.9339 - val_loss: 0.4099 - val_accuracy: 0.9531
Epoch 4/5
96/96 [==============================] - 11s 110ms/step - loss: 0.4248 - accuracy: 0.9417 - val_loss: 0.4073 - val_accuracy: 0.9609
Epoch 5/5
96/96 [==============================] - 11s 110ms/step - loss: 0.4194 - accuracy: 0.9473 - val_loss: 0.4009 - val_accuracy: 0.9531
```

Obrázek 10: Detailní model

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hub_keras_layer_v1v2_3 (HubK (None, 1280)              3413024
_________________________________________________________________
dropout_3 (Dropout)          (None, 1280)              0
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 3843
=================================================================
Total params: 3,416,867
Trainable params: 3,843
Non-trainable params: 3,413,024
_________________________________________________________________
None
INFO:tensorflow:Retraining the models...
INFO:tensorflow:Retraining the models...
Epoch 1/10
96/96 [==============================] - 11s 116ms/step - loss: 0.5885 - accuracy: 0.8545 - val_loss: 0.4435 - val_accuracy: 0.9323
Epoch 2/10
96/96 [==============================] - 10s 109ms/step - loss: 0.4525 - accuracy: 0.9277 - val_loss: 0.4149 - val_accuracy: 0.9531
Epoch 3/10
96/96 [==============================] - 11s 110ms/step - loss: 0.4357 - accuracy: 0.9391 - val_loss: 0.4076 - val_accuracy: 0.9635
Epoch 4/10
96/96 [==============================] - 11s 110ms/step - loss: 0.4217 - accuracy: 0.9502 - val_loss: 0.4050 - val_accuracy: 0.9557
Epoch 5/10
96/96 [==============================] - 10s 109ms/step - loss: 0.4167 - accuracy: 0.9479 - val_loss: 0.4002 - val_accuracy: 0.9635
Epoch 6/10
96/96 [==============================] - 11s 110ms/step - loss: 0.4126 - accuracy: 0.9535 - val_loss: 0.4007 - val_accuracy: 0.9609
Epoch 7/10
96/96 [==============================] - 11s 110ms/step - loss: 0.4087 - accuracy: 0.9541 - val_loss: 0.3963 - val_accuracy: 0.9661
Epoch 8/10
96/96 [==============================] - 10s 109ms/step - loss: 0.4024 - accuracy: 0.9629 - val_loss: 0.3952 - val_accuracy: 0.9583
Epoch 9/10
96/96 [==============================] - 11s 110ms/step - loss: 0.4054 - accuracy: 0.9567 - val_loss: 0.3930 - val_accuracy: 0.9557
Epoch 10/10
96/96 [==============================] - 11s 110ms/step - loss: 0.3991 - accuracy: 0.9580 - val_loss: 0.3911 - val_accuracy: 0.9635
```

Obrázek 11: Pokročilý model

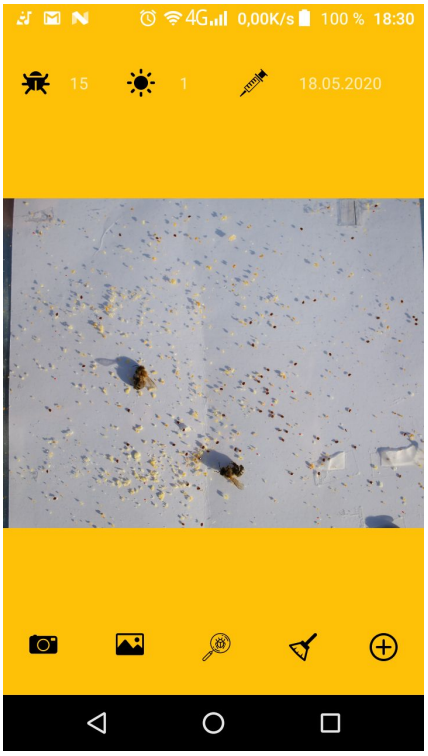
Bohužel, nebylo dostatek času, aby se modely do řešení aplikace implementovaly. Řešením pro nalezení matky by nejspíše bylo o určitém rozměru rozřezat získaný snímek pro rozeznání jednotlivých oblastí a oblast, kde by se rozeznala matka, by se označila.

10. Testování aplikace

Aplikace byla testována společně s konkurenční aplikacemi Varroa Detector zmíněné v kapitole Existující aplikace. Aplikace byla vyzkoušena na snímcích uložených v mobilním zařízení abychom mohli porovnat úspěšnost nálezů. Byla testována ještě aplikace Varroa Counter na stejné desce, ale nebylo možné využít stejných snímků, jelikož aplikace dokázala přijímat vzorky pouze pomocí fotoaparátu. Aplikace přesto nenalezla jediný pozitivní nález proto není dále porovnávána. Snímky byly pořízeny v podmínkách blízkých v jakých by se měly vyskytovat v praxi a prováděny na spadových deskách přímo z úlu.

U fotografie pana Chlubny byly roztoči rozpoznány velice správně s malým množstvím chyb viz obrázek 12. To konkurenční aplikace, nedokázala viz obr. 13. Dosáhla velmi nízkého počtu nálezů a většina nálezů byla chybných. Ovšem momentálně se jednalo o snímek pana Chlubny, na který byl jeho model naučen, tudíž měl jistou převahu nad konkurenční aplikací.

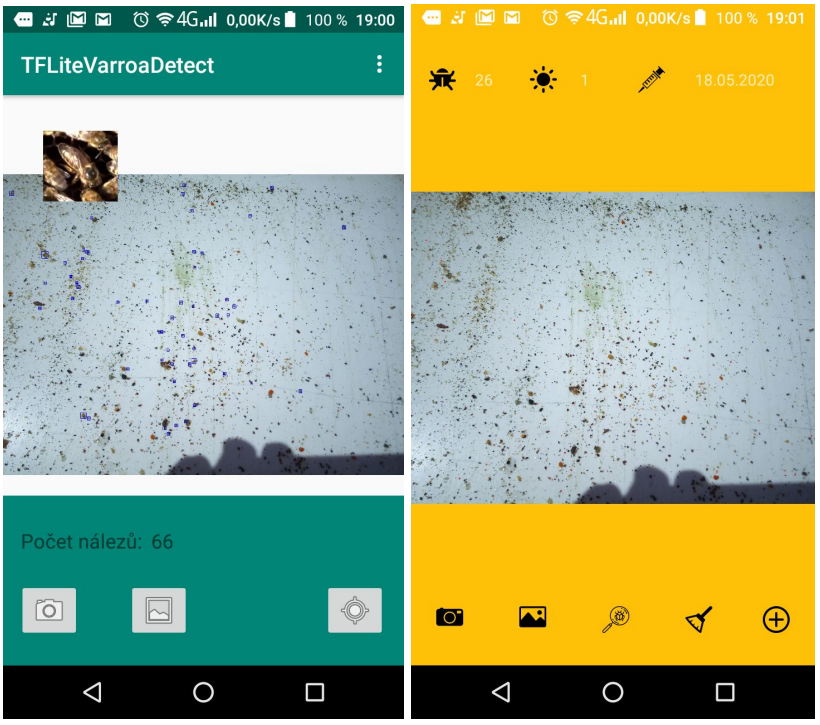
Obrázek 12: snímek pana Chlubny app této práce



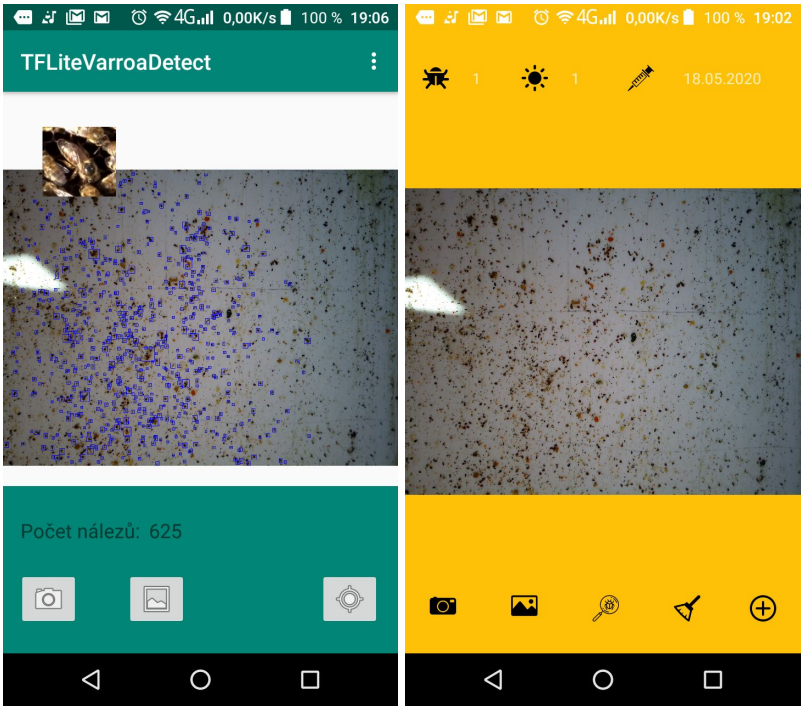
Obrázek 13: snímek pana Chlubny app VarroaDetector

Dále byly testovány snímky z terénu se spádovými deskami. Na těchto fotografiích bylo vidět, že kontury byly aplikací této práce nalezeny správně, ovšem už je model hůře rozeznával. Nalézají se ve výsledcích i pozitivní nálezy, ale velice zřídka. Porovnávaná aplikace na tom nebyla o nic lépe viz. obr. 14, 15.

Pro zlepšení rozpoznávání by mohlo pomoci vytvořit model, který by byl trénován přímo na fotografiích získaných z terénu mobilním zařízením. Tyto snímky by měly totiž menší rozlišení a různé venkovní světlo. Tím by se alespoň model trénoval na snímcích, které by sami poté museli být podle něj rozpoznávány.



Obrázek 14: porovnání aplikací 1



Obrázek 15: porovnání aplikací 2

11. Závěr

Tato práce se zabývala problematikou implementace a sdružení vyvinutých aplikací vhodných pro detekce a sečtení výskytu jedinců *Varroa destructor* a aplikace detekující včelí matku. Uvedené aplikace byly vyvinuty v počítačovém prostředí, následně měla proběhnout jejich implementace do mobilní aplikace.

V prvních částech této práce se autor zabývá popisem problematiky výskytu *Varroa destructor* a popisem včelího společenstva. Dále popisuje fungování již existujících aplikacemi Varroa Detektor, Varroa Counter pro identifikaci roztoče *Varroa destructor* a ↓aplikaci BeeScanning - diagnose bees health vhodný pro identifikace matky na včelím díle. V rámcí tohoto popisu autor zjistil a popsal fungovaní výše zmíněných aplikací a mimo jiné se snažil analyzovat slabá místa těchto aplikací.

Další částí této práce byl návrh a implementace vlastní aplikace do systému Android, za použití předcházejících závěrečných prací pro detekce a sumarizaci *Varroa destructor* a za účelem detekce a lokalizace včelý matky, která by odpovídala výkonnostním možnostem mobilních zařízení.

Podařilo se implementovat rozpoznávání roztoče *Varroa destructor* z předchozí práce. Bylo vytvořeno prostředí které podporuje bližší náhled na výsledky rozpoznávání. U tvorby prostředí byla vyvinuta snaha o intuitivnost ovládání.

Rozpoznávání včelí matky nebylo dokončeno z důvodů problému s modely pro rozpoznávání včelí matky. Byla však tato část rozpracována do fáze, kdy je třeba implementovat modely. Zároveň byly vytvořeny nové modely ze snímků předchozí práce pana Humpála, které by nejspíše bylo možné implementovat.

Na závěr bylo provedeno testování aplikace s porovnáním aplikace VarroaDetektor. Zároveň bylo popsáno, co by bylo dobré zlepšit, aby aplikace lépe fungovala.

12. Seznam použité literatury a internetových zdrojů

[1]Anderson, D. I., Trueman, J. W. H.: Varroa jacobsoni (Acari: Varroidae) is more than one species. Experimental and Applied Acarology, 2000.

[2]*Android Developers* [online]. google [cit. 2020-05-18]. Dostupné z: https://developer.android.com/

[3]Android Studio: An IDE built for Android [online]. 2013 [cit. 2019-12-01]. Dostupné z: https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html

[4]Amdam, G. V., Omholt, S. W.: The regulatory anatomy of honeybee lifespan. Journal of Theoretical Biology, 2002.

[5]BeeScanning - diagnose bees health - Apps on Google Play. [online]. Copyright ©2019 Google [cit. 18.02.2019]. Dostupné z: https://play.google.com/store/apps/details?id=com.beescanning.android

[6] BIENEFELD, Kaspar. Včelařství krok za krokem: pro milovníky krásného koníčka. 2. vyd. Přeložil Anna ŠTORKÁNOVÁ. Líbeznice: Víkend, 2010. ISBN 978-80-7433-023-0

[7]Bogaerts, A., Baggerman, G., Vierstraete, E., Schoofs, L., Verleyen, P.: The hemolymph proteome of the honeybee: Gel-based or gel-free? Proteomics, 2009.

[8]Büchler, R., Berg, S., Le Conte, Y.: Breeding for resistence to Varroa destructor in Europe. Apidologie, 2010.

[9]Čavojský, V. (1981). Včelárstvo. Bratislava, Príroda.

[10] ČERMÁK, K. a kol. Včelařství. České Budějovice: PSNV, 2016. ISBN 978-80- 260-9090-8.

[11]Dainat, B., Evans, J. D., Chen, Y. P., Gauthier, L., Neumann, P.: Predictive markers of honey bee colony collapse. PloS ONE, 2012.

[12]Dietemann, V., Pflugfelder, J., Anderson, D., Charrière, J. D., Chejanovsky, N., Dainat, B., de Miranda, J., Delaplane, K., Dillier, F. X., Fuch, S., Gallmann, P., Gauthier, L., Imdorf, A., Koeniger, N., Kralj, J., Meikle, W., Pettis, J., Rosenkranz, P., Sammataro, D., 86 Smith, D., Yanez, O., Neumann, P.: Varroa destructor: research avenues towards sustainable control. Journal of Apicultural Research, 2012.

[13]EVOLVEO StrongPhone G4 [online]. [cit. 2019-11-30]. Dostupné z: https://www.evolveo.com/cz/strongphone-g4-android-7

[14]Genersch, E.: Honey bee pathology: current threats to honey bees and beekeping. Applied Microbiology and Biotechnology, 2010.

[15]Harizanis, P. C.: Infestation of queen cells by the mite Varroa jacobsoni. Apidologie, 1991.

[16]Hubert, J., Bicianova, M., Ledvinka, O., Kamler, M., Lester, P., J., Nesvorna, M., Kopecky, J., Erban, T.Changes in the Bacteriome of Honey Bees Associated with the Parasite Varroa destructor, and Pathogens Nosema and Lotmaria passim. Microbial Ecology 73, (2017).

[17]HUMPÁL, Jan. Analýza pohybu vcelstva po plástu. [Analysis of movement of bees on honeycomb. Bc. Thesis, in Czech.]. Ceské Budejovice, Czech Republic, 2018. Bakalářská práce. Faculty of Science, University of South Bohemia.

[18]CHLUBNA, Marek. Analýza spadu na podložky na dně včelího úlu.

[Analysis of the fallout on the pads at the bottom of the beehive. Bc. Thesis, in Czech.] –

36p., České Budějovice, Czech Republic, 2018. Bakalářská práce. Faculty of Science,

University of South Bohemia.

[19]Image_classification.ipynb: Image classification with TensorFlow Lite Model Maker. *Colab.research.google.com* [online]. google [cit. 2020-05-18]. Dostupné z: https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/lite/g3doc/tutorials/model_maker_image_classification.ipynb

[20]Java Native Interface (JNI) [online]. [cit. 2019-12-02]. Dostupné z: https://docs.oracle.com/en/java/javase/11/docs/specs/jni/intro.html#java-native-interface-overview

[21]Knesplová, T. (2012). Léčba a tlumení varroázy v chovu včel a včelařství jako didaktické téma: diplomová práce. Praha: Univerzita Karlova v Praze. Pedagogická fakulta.

[22]Krabec, J.: Medu jíme poměrně málo, zřejmě na něj moc nemyslíme. Včelařství, 2005, roč. 58, č. 10, s. 3-4 (příloha). ISSN 0042-2924

[23]Le Conte, Y., Ellis, M., Ritter, W.: Varroa mites and honey bee health: can Varroa explain part of the colony losses? Apidologie, 2010.

[24]Milani, N.: The resistence of Varroa jacobsoni Oud. to acaricides. Apidologie, 1999.

[25]Molnár K, Úvod do problematiky umělých neuronových sítí, Elektrorevue, 2000

[26]Novák M. a kol., Umělé neuronové sítě, teorie a aplikace, Praha, 1998

[27]*OpenCV* [online]. [cit. 2019-11-30]. Dostupné z: https://opencv.org/about/

[28]OpenCV4Android SDK. *OpenCV* [online]. opencv dev team, 2011 [cit. 2020-05-18]. Dostupné z: https://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html

[29]Rosenkranz, P., Aumeier, P., Ziegelmann, B.: Biology and control of Varroa destructor. Journal of Invertebrate Pathology, 2010, vol. 103, s. 96-119.

[30]Ruijter, A. D., Kaas, J. P.: The anatomy of varroa mite. Rotterdam: Balkema,1983.

[31]Sammataro, D., Gerson, U., Needham, G.: Parasitic mites of honey bees: Life, history, implications and impact. Annual Review of Entomology, 2000.

[32]Tempír, Z.: Začátky sdružování včelařů. Včelařství, 2002, roč. 55, č. 1, s. 2 (obálka). ISSN 0042-2924 3

[33]TensorFlow [online]. [cit. 2019-12-01]. Dostupné z: https://www.tensorflow.org/

[34]TensorFlow, Google's Open Source AI, Signals Big Changes in Hardware Too [online]. 2015 [cit. 2019-12-01]. Dostupné z: https://www.wired.com/2015/11/googles-open-source-ai-tensorflow-signals-fast-changing-hardware-world/

[35]Pollenity [online]. [cit. 2019-12-08]. Dostupné z: https://pollenity.com/varroa-destructor-worldwide-scourge/

[36]Varroa Counter - Apps on Google Play. [online]. Copyright ©2019 Google [cit. 18.02.2019]. Dostupné z: https://play.google.com/store/apps/details?id=topLab.VarroaCounter

[37]Varroa Detector - Apps on Google Play. [online]. Copyright ©2019 Google [cit. 18.02.2019]. Dostupné z: https://play.google.com/store/apps/details?id=quaxi.org.varroacounter

[38]Veselý, V. a kol.: Včelařství. 2. vyd. Praha: Nakladatelství Brázda, 2003. 272 s. ISBN 80- 209-0320-8