



**MATEMATICKO-FYZIKÁLNÍ
FAKULTA**
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Vojtěch Bláha

Použití šablon nezávislých na rotaci pro rozpoznání vajíček na snímku

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Marta Vomlelová, Ph.D.

Studijní program: Informatika

Studijní obor: IPP5

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V dne

Podpis autora

Chtěl bych poděkovat vedoucí mé bakalářské práce Mgr. Martě Vomlelové, Ph.D. za odborné rady, trpělivost a čas strávený při konzultacích mé práce. Také bych chtěl poděkovat všem, kteří mě při studiu podporují, protože s jejich podporou se studium stává o mnoho snadnější.

Název práce: Použití šablon nezávislých na rotaci pro rozpoznání vajíček na snímku

Autor: Vojtěch Bláha

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Marta Vomlelová, Ph.D., Katedra teoretické informatiky a matematické logiky

Abstrakt: Tato bakalářská práce se zabývá rozpoznáváním vajíček na obrázku. Cílem bylo vytvořit skupinu programů, které nejdříve nasnímají obrazová data, poté v nich najdou vajíčka a nakonec zpřístupní výsledky v nějakém uživatelském prostředí. Postupně jsme otestovali různé klasifikační metody (porovnání se vzory, logistickou regresi a neuronové sítě). Také jsme vyzkoušeli různé reprezentace obrázku jako je maticová reprezentace a prstencová projekce. Podívali jsme se i na různá předzpracování obrazu před samotným hledáním, využili jsme šedotón, barevná spektra a hrany detekované horní propustí nebo Kirscheho detektorem. Po otestování všech metod jsme vybrali tu nejlepší a vytvořili jsem samotný klasifikační program. Nejúspěšnější metoda byla logistická regrese s prstencovou reprezentací.

Klíčová slova: prstencová projekce, vajíčka, porovnání se vzory, logistická regrese, neuronová síť, Kirschův hranový detektor

Title: Rotation-invariant pattern matching for egg recognition

Author: Vojtěch Bláha

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Marta Vomlelová, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: This bachelor's thesis follows up the recognition of eggs in the image. The goal was to create a group of programs that firstly captures image data, than finds eggs in them and finally shows the results in some user environment. We tested gradually different classification methods (template matching, logistic regression and neural network). We tried also different representations of the image such as matrix representation and ring projection, and various pre-processing of the image before the actual finding, we used grayscale, color spectra and edges detected by a high-pass or Kirsche detector. After testing all methods, we selected the best one and created the classification program itself. The most successful method was logistic regression with ring projection.

Keywords: ring projection, eggs, template matching, logistic regression, neural network, Kirsch's edge detector

Obsah

Úvod	3
1 Reprezentace obrázku	4
1.1 Maticová reprezentace	4
1.2 Prstencová projekce	5
2 Předzpracování obrázku	6
2.1 Barevné složky	6
2.2 Šedotón	7
2.2.1 Průměr spekter	7
2.2.2 Vážený průměr spekter	7
2.3 Detekce hran	8
2.3.1 Horní propust	8
2.3.2 Kirschův hranový detektor	10
3 Klasifikace obrázku	11
3.1 Porovnání se vzorem	11
3.1.1 Normovaná kvadratická chyba	11
3.1.2 Normovaná vzájemná korelace	11
3.1.3 Normované korelační koeficienty	12
3.2 Logistická regrese	13
3.3 Neuronové sítě	15
3.3.1 Trénování neuronové sítě	15
3.3.2 Použití neuronové sítě	16
4 Hledání komponent	17
5 Snímání obrázku	18
5.1 Hardware	18
5.2 Software	19
6 Webový server	20
6.1 Databáze	20
6.1.1 Nastavení kamer	20
6.1.2 Data kamer	21
6.2 Webové stránky	22
7 Klasifikační program	24
7.1 Příprava dat	24
7.2 Ohodnocení klasifikátoru	25
7.3 Porovnání se vzory	26
7.3.1 Výběr vzorů	26
7.3.2 Výběr chybové funkce	26
7.3.3 Výběr reprezentace obrázku	27
7.3.4 Hodnocení	28
7.4 Porovnání s prstencovými vzory	28

7.4.1	Výběr vzorů	28
7.4.2	Výběr chybové funkce	28
7.4.3	Výběr reprezentace obrázku	29
7.4.4	Hodnocení	30
7.5	Logistická regrese	30
7.5.1	Výběr reprezentace obrázku	30
7.5.2	Ohodnocení trénovacích dat	31
7.5.3	Kombinace reprezentací	31
7.5.4	Hodnocení	32
7.6	Neuronové sítě	33
7.6.1	Výběr reprezentace obrázku	33
7.6.2	Ohodnocení trénovacích dat	33
7.6.3	Kombinace reprezentací	34
7.6.4	Vrstvy neuronů	35
7.6.5	Hodnocení	35
7.7	Samotný klasifikační program	36
	Závěr	37
	Seznam použité literatury	38
	Seznam obrázků	39
	Seznam tabulek	41
A	Přílohy	42
A.1	Vývojová dokumentace	43
A.1.1	Snímač obrázků	44
A.1.2	Hledač vajíček	45
A.1.3	Webové rozhraní	47
A.1.4	C++ moduly	48
A.1.5	SQL databáze	48
A.2	Uživatelská dokumentace	49
A.2.1	SQL Databáze	49
A.2.2	Snímač obrázků	50
A.2.3	C++ moduly	58
A.2.4	Hledač vajíček	59
A.2.5	Webové rozhraní	61
A.3	Testování	67
A.3.1	Obrázky	67
A.3.2	Porovnání se vzory	68
A.3.3	Porovnání s prstencovými vzory	69
A.3.4	Hledání pomocí klasifikačních modelů	70

Úvod

Cílem této práce je vytvořit skupinu programů, které budou zpracovávat obrázky a hledat v nich vajíčka. První program se bude starat o snímání obrazových dat z místa, kam slepice pravidelně snáší vajíčka, a následně bude ony snímky odesílat na server, který bude zprostředkovávat komunikaci mezi jednotlivými programy a uživatelským rozhraním. Druhý program bude hledat vajíčka na snímcích, které jsou uloženy na zmíněném serveru. Poslední částí aplikace bude webové uživatelské prostředí, které bude zpřístupňovat nafocené snímky a pozice nalezených vajíček.

V 1. kapitole se podíváme na možné reprezentace obrázku, které bychom mohli použít jako příznaky pro klasifikaci. Uvedeme si základní maticovou reprezentaci a také rotačně nezávislou prstencovou projekci, která nám umožňuje reprezentovat dvě rotace téhož obrázku stejnými příznaky.

V 2. kapitole si rozebereme použitelné předzpracování obrázku, tedy možné způsoby jak z barevného tříspektrálního snímku dostat jednospektrální snímek pro následnou klasifikaci. Zmíníme použití jednotlivých barevných složek, šedotónové reprezentace, ale také hran detekovaných různými hranovými detektory. Z těchto detektorů použijeme horní propust a Kirschův hranový detektor.

V kapitole 3 se podíváme na jednotlivé klasifikační metody, které bychom mohli použít. Nejjednodušší z nich je porovnání se vzorem, u kterého si definujeme různé chybové funkce, podle kterých budeme počítat shodnost s daným vzorem. Dále si ukážeme složitější metody, jako je logistická regrese a neuronové sítě. U obou těchto metod se zmíníme o jejich trénování a následné klasifikaci dat.

Ve 4. kapitole se krátce podíváme na algoritmus, který budeme používat pro získání samotných pozic vajíček z oklasifikované matice. Použijeme k tomu mírně upravený algoritmus na hledání komponent souvislosti v grafu.

V kapitole 5 si rozebereme snímání obrazových dat. Podíváme se jak na hardware snímacího zařízení, tak na jeho software. K snímání použijeme minipočítač „Raspberry Pi Zero“ s připojenou kamerou.

V předposlední kapitole 6 se podíváme na webový server, který bude zprostředkovávat komunikaci mezi jednotlivými programy. Rozebereme si strukturu jednotlivých databázových tabulek a webových stránek, které budou zpřístupňovat výsledky uživatelům.

V poslední kapitole 7 otestujeme jednotlivé metody na hledání vajíček. Nejprve si připravíme trénovací a testovací data. Poté budeme ladit jednotlivé klasifikační parametry. A nakonec vybereme nejúspěšnější metodu a vytvoříme samotný klasifikační program.

1. Reprezentace obrázku

Obrazová data je možné reprezentovat mnoha způsoby. Jednotlivé reprezentace se liší v mnoha parametrech. Nejdůležitějším z nich je, zda jsou data reprezentována rastrově nebo vektově. My budeme uvažovat pouze rastrovou reprezentaci. Dále se můžeme zabírat rozlišením, bitovou hloubkou, barevnými spektry, kompresí a dalšími vlastnosti obrázku.

Nás ale bude především zajímat, jestli neexistují nějaké použitelné projekce, které by zrychlily či zlepšily vyhledávání vzorů v obrázku. V našem experimentu budeme testovat takzvanou prstencovou projekci (anglicky ring-projection). Výsledky vyhledávání s prstencovou projekcí poté porovnáme s výsledky s maticovou reprezentací obrázku.

1.1 Maticová reprezentace

Maticová reprezentace je u rastrových obrázků standartně používána. Obrázek je rozdělen na čtverečky, takzvané pixely, kde každý pixel má danou hodnotu v každém barevném spektru. Příklad maticové reprezentace můžete vidět na obrázku 1.1.

0	0	0	0	0	0
0	0	0	0	0	0
255	255	255	255	255	255
255	0	0	0	0	0
255	0	0	0	0	0
255	255	255	255	255	255
255	255	0	0	0	0
255	255	0	0	0	0
255	255	255	255	255	255
255	255	255	255	255	255
255	255	0	0	0	0
255	255	0	0	0	0
255	255	255	255	255	255
255	0	0	0	0	0
255	0	0	0	0	0
255	255	255	255	255	255
0	0	0	0	0	0
0	0	0	0	0	0

R
G
B

Obrázek 1.1: Příklad rastrového obrázku (vlajka).

1.2 Prstencová projekce

Prstencová projekce je dle článku Tsai a Chiang (2002) definována jako funkce transformující 2D šedotónový obrázek na rotačně nazávislou reprezentaci v 1D. Tato projekce byla inspirována už známými algoritmy (TANG a kol. (1991); Yuen a kol. (1998)).

Mějme tedy čtvercový obrázek $f(x,y)$ o velikosti $2N$. Nejprve obrázek $f(x,y)$ v karteziánských souřadnicích transformujeme do polárních souřadnic:

$$x = r \cdot \cos(\theta),$$

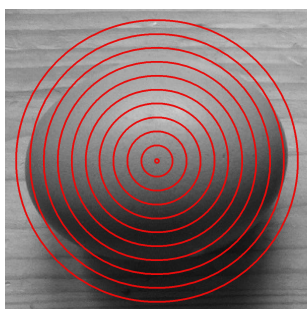
$$y = r \cdot \sin(\theta).$$

Prstencová projekce obrázku $f(x,y)$ na poloměru r , budeme ji značit $p(r)$, je definována jako průměr hodnot $f(r \cdot \cos(\theta), r \cdot \sin(\theta))$ přes všechny úhly θ . To jest:

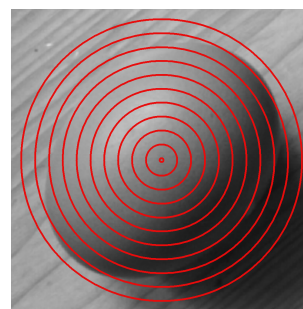
$$p(r) = \frac{1}{n_r} \sum_k f(r \cos(\theta_k), r \sin(\theta_k)), \quad (1.1)$$

kde n_r je celkový počet pixelů na kružnici o poloměru r a $r = 0,1,2,\dots,N$.

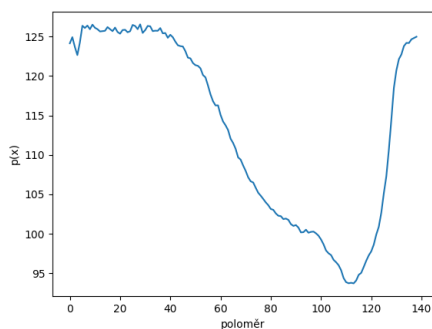
Protože projekce je počítána na soustředných kružnicích s rostoucím poloměrem, tak výsledná 1D prstencová projekce je nezávislá na otočení vstupního 2D obrázku. Mějme například pootočené vstupní obrázky 1.2 a 1.3. Pak je vidět, že jejich projekce 1.4 a 1.5 jsou až na zanedbatelné odchylky identické.



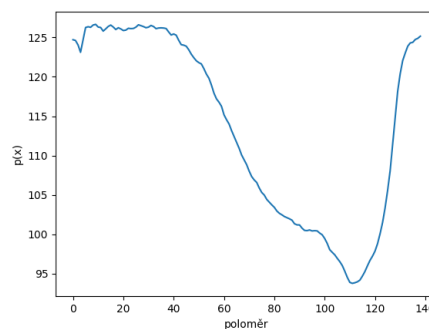
Obrázek 1.2: Vejde otočené 1.



Obrázek 1.3: Vejde otočené 2.



Obrázek 1.4: Projekce vejce 1.



Obrázek 1.5: Projekce vejce 2.

2. Předzpracování obrázku

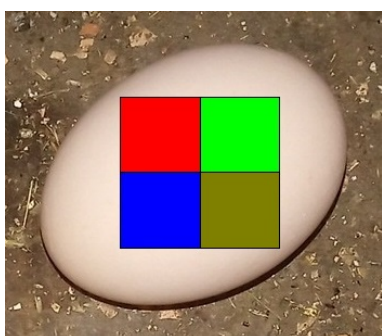
Před samotným hledáním vajíček v obrázku je nutné předzpracování obrazových dat. Data totiž dostáváme ve formě barevných tříspektrálních (RGB) fotek, zatímco algoritmy na hledání vajíček pracují s jednospektrální maticí. Proto použijeme nějaké ze známých algoritmů na transformaci vstupního obrázku na jednospektrální matici.

Jednodušší metody transformují obrázek nezávisle pixel po pixelu. Z nich můžeme použít například převedení barevného obrazu na šedotónový nebo extrahování pouze jednoho barevného spektra. Avšak tyto metody obrázku nepřidávají žádné nové informace, proto použijeme i komplikovanější hranové detektory.

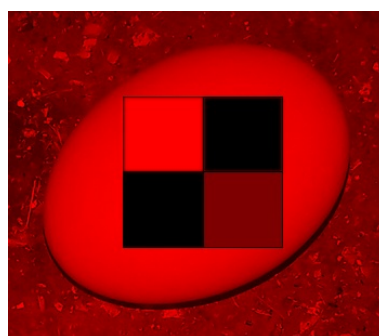
2.1 Barevné složky

Běžně dostáváme obrazová data ve formátu matice M typu $(m, n, 3)$, kde m je šířka obrázku a n jeho výška. Na převedení do tvaru (m, n) bychom mohli použít jednu ze tří podmatic $M[:, :, 0]$, $M[:, :, 1]$ nebo $M[:, :, 2]$. Tyto podmatice odpovídají barevným spektrům RGB daného obrázku.

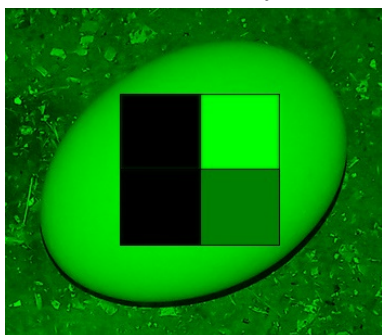
Použití barevných spekter by mohlo být docela účinné na hledání vajíček, protože vajíčka mají poměrně výraznou červenou složku a to by mohlo pomoci při klasifikaci obrázku. Příklad barevných spekter můžete vidět na obrázcích 2.2 - 2.4.



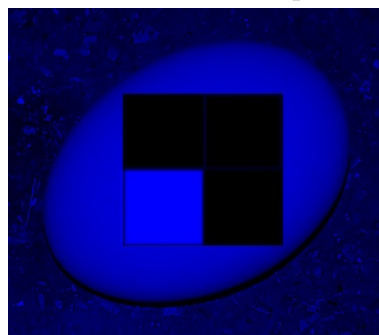
Obrázek 2.1: Barevný obrázek.



Obrázek 2.2: Červené spektrum.



Obrázek 2.3: Zelené spektrum.



Obrázek 2.4: Modré spektrum.

2.1 - 2.4: Ukázka barevných spekter obrázku.

2.2 Šedotón

Existuje mnoho způsobů, jak transformovat barevný obrázek s červeným, zeleným a modrým spektrem na šedotónový (Kanan a Cottrell (2012)), my si zde ukážeme pouze základní z nich. Porovnání níže uvedených metod můžete vidět na obrázcích 2.6 - 2.8.

2.2.1 Průměr spekter

Nejjednodušší metoda spočívá v průměrování všech spekter. Tedy hodnotu pixelu na pozici x, y , označenou $g(x, y)$, vypočítáme takto:

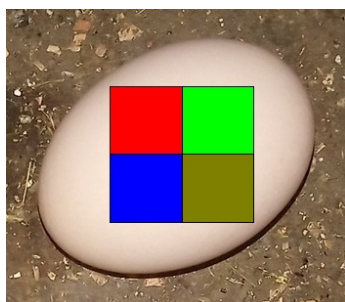
$$g(x, y) = \frac{1}{3}(R(x, y) + G(x, y) + B(x, y)), \quad (2.1)$$

kde $R(x, y)$ je hodnota v červeném spektru, $G(x, y)$ v zeleném spektru a $B(x, y)$ v modrém spektru.

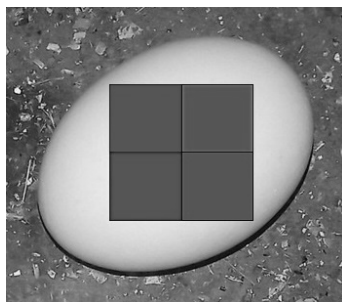
2.2.2 Vážený průměr spekter

Výše uvedená metoda ale neodpovídá tomu, jak barevné složky vnímá lidský mozek. Lidské oko totiž obsahuje 3 druhy buněk, které vnímají barvy (Wikipedie (2022)). Tyto buňky se nazývají čípky a detekují modrofialovou, zelenou a oranžovou barvu. Poměr počtů těchto buněk určuje, jakým poměrem bychom měli vážit barevné složky (RGB), aby vznikl šedotónový obrázek, který odpovídá lidskému vnímání. Proto hodnotu šedotónu vypočítáme jako vážený průměr barevných spekter s danými konstantami:

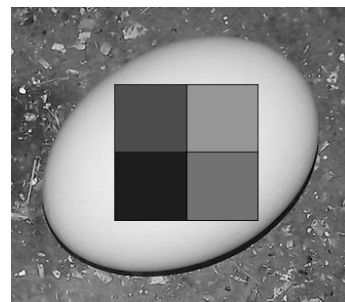
$$g(x, y) = 0.2126 \cdot R(x, y) + 0.7152 \cdot G(x, y) + 0.0722 \cdot B(x, y). \quad (2.2)$$



Obrázek 2.6: Barevný.



Obrázek 2.7: Šedotón (2.1).



Obrázek 2.8: Šedotón (2.2).

2.6 - 2.8: Porovnání transformací na šedotón.

2.3 Detekce hran

Na detekování hran bychom mohli použít horní propust (angl. high pass filter) nebo hranové detektory, které k detekci používají konvoluční filtry.

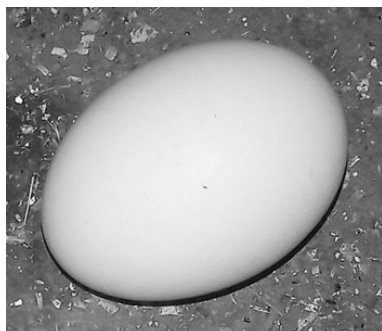
2.3.1 Horní propust

Detekce hran pomocí horní propusti spočívá v tom, že z obrázku vyfiltrujeme nízké frekvence, které většinou odpovídají souvislým plochám, a ponecháme vysoké frekvence, které odpovídají právě hranám.

Abychom získali horní propust obrázku $f(x, y)$ musíme nejprve spočítat diskétní fourierovu transformaci $F(x, y)$ danou vzorcem:

$$F(x, y) = \sum_{n=0}^{W-1} \left\{ \omega_W^{x \cdot n} \sum_{m=0}^{H-1} \omega_H^{y \cdot m} \cdot f(x, y) \right\},$$
$$\omega_W = e^{\frac{-i2\pi}{W}},$$
$$\omega_H = e^{\frac{-i2\pi}{H}},$$

kde W je šířka obrázku, H jeho výška. Na obrázcích 2.10, 2.11 můžete vidět vstupní obrázek $f(x, y)$ a amplitudu fourierovy transformace $F(x, y)$.



Obrázek 2.10: Vstupní obrázek.



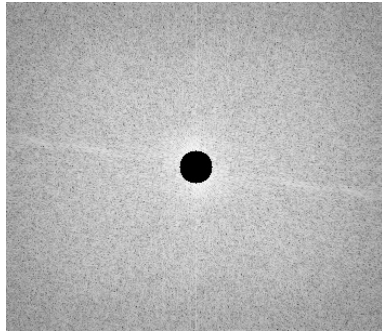
Obrázek 2.11: Furierova transformace.

Následně vynulujeme hodnoty fourierovy transformace v nízkých frekvencích, tím získáme vyfiltrované vysoké frekvence $G(x, y)$ (viz obrázek 2.12).

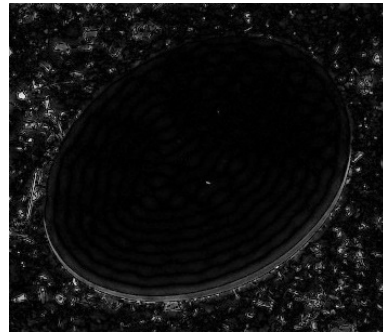
Poté provedeme inverzní fourierovu transformaci z $G(x, y)$ a získáme horní propust $g(x, y)$ (viz obrázek 2.13).

Inverzní fourierova transformace je daná vzorcem:

$$g(x, y) = \frac{1}{W \cdot H} \sum_{n=0}^{W-1} \left\{ \omega_W^{-x \cdot n} \sum_{m=0}^{H-1} \omega_H^{-y \cdot m} \cdot G(x, y) \right\}.$$



Obrázek 2.12: Oříznutá furierova transformace.



Obrázek 2.13: Horní propust obrázku.

2.3.2 Kirschův hranový detektor

Abychom mohli vysvětlit kirschův hranový detektor (Ghosal a kol. (2021)), budeme muset nejdříve definovat konvoluci. Konvoluce obrázku $f(x, y)$ s filtrem $g(x, y)$, označíme $(f * g)(x, y)$, je daná vzorcem:

$$(f * g)(x, y) = \sum_{i=-k}^k \left\{ \sum_{j=-k}^k f(x-i, y-j) \cdot g(i, j) \right\},$$

$$k = \left\lfloor \frac{N}{2} \right\rfloor,$$

kde N je rozměr konvolučního filtru, tedy $g(x, y)$ má rozměr $N \times N$.

Kirschův hranový detektor používá osm konvolučních filtrů na detekování hran v osmi různých směrech. Filtry vypadají následovně:

$$N = \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}, NW = \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}, W = \begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix},$$

$$SW = \begin{pmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{pmatrix}, S = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{pmatrix}, SE = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{pmatrix},$$

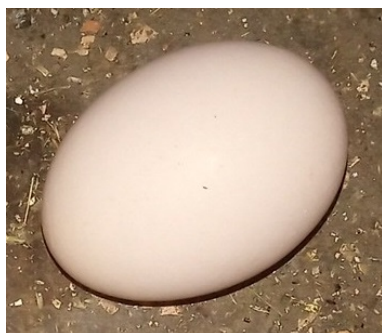
$$E = \begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix}, NE = \begin{pmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{pmatrix}.$$

Filtr N detekuje hrany jdoucí zdola nahoru, filtr NW hrany jdoucí z pravého dolního rohu do levého horního, a tak dále.

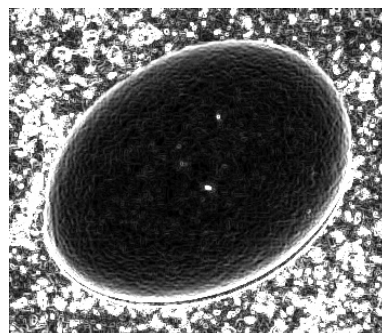
Poté co detektor vypočítá všechny konvoluce obrázku $f(x, y)$, tak výsledné hrany $h(x, y)$ spočítá jako maximum všech konvolucí v bodě x, y , tedy:

$$h(x, y) = \max \left\{ (f * N)(x, y), (f * NW)(x, y), (f * W)(x, y), (f * SW)(x, y), \right. \\ \left. (f * S)(x, y), (f * SE)(x, y), (f * E)(x, y), (f * NE)(x, y) \right\}$$

Výsledek kirscheho hranového detektoru můžete vidět na obrázcích 2.14, 2.15.



Obrázek 2.14: Vstupní obrázek.



Obrázek 2.15: Detekované hrany.

3. Klasifikace obrázku

Nyní se podíváme na metody, které bychom mohli použít ke klasifikaci obrázků. Výstupem těchto metod by mělo být rozhodnutí, zda na obrázku je nějaké vajíčko, nebo tam naopak žádné není.

Na klasifikaci použijeme jednoduché metody, které k pozitivní klasifikaci požadují pouze dostatečnou podobnost obrázků, ale i složitější metody jako lineární regrese nebo neuronové sítě.

Nyní si tedy přiblížíme jednotlivé klasifikační metody.

3.1 Porovnání se vzorem

Tato metoda porovná vstupní obrázek s předem zadaným vzorem. Pokud je vstupní obrázek dostatečně podobný danému vzoru, tak ho klasifikuje pozitivně. V opačném případě je obrázek klasifikován negativně.

K určení podobnosti dvou obrázků budeme potřebovat chybové funkce (viz OpenCV (2022)), které vrátí jak moc rozdílné obrázky jsou. Jednotlivé chybové funkce si ukážeme při použití na vstupním obrázku $I(x, y)$ (viz obrázek 3.1), v kterém budeme hledat vzor $T(x, y)$ s šířkou W_t a výškou H_t (viz obrázek 3.2).

3.1.1 Normovaná kvadratická chyba

Hodnotu normované kvadratické chyby, označíme $E(x, y)$, vypočítáme pro jeden podobrázek $I(x : x + W_t, y : y + H_t)$ následovně:

$$E(x, y) = \frac{\sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} \{T(i, j) - I(x + i, y + j)\}^2}{\sqrt{\sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} T(i, j)^2 \cdot \sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} I(x + i, y + j)^2}}.$$

Výsledek této metody za použití šedotónové reprezentace můžete vidět na obrázku 3.3.

3.1.2 Normovaná vzájemná korelace

Korelace udává, jak moc jsou obrázky podobné, tedy čím menší mají obrázky korelaci, tím větší je chyba.

Hodnotu normované vzájemné korelace, označíme $C(x, y)$, vypočítáme pro jeden podobrázek $I(x : x + W_t, y : y + H_t)$ následovně:

$$C(x, y) = \frac{\sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} \{T(i, j) \cdot I(x + i, y + j)\}^2}{\sqrt{\sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} T(i, j)^2 \cdot \sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} I(x + i, y + j)^2}}.$$

Výsledek této metody za použití šedotónové reprezentace můžete vidět na obrázku 3.4.

3.1.3 Normované korelační koeficienty

Korelační koeficienty vzniknou aplikováním vzájemné korelace na upravené vstupní obrázky. Upravené vstupní obrázky získáme odečtením průměru obrázku v každém pixelu.

Hodnotu normovaných korelačních koeficientů, označíme $C(x, y)$, vypočítáme pro jeden podobrázek $I(x : x + W_t, y : y + H_t)$ následovně:

$$C(x, y) = \frac{\sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} \{T'(i, j) \cdot I'(x + i, y + j)\}^2}{\sqrt{\sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} T'(i, j)^2 \cdot \sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} I'(x + i, y + j)^2}},$$

$$T'(x, y) = T(x, y) - \frac{1}{W_t \cdot H_t} \cdot \sum_{i=0}^{W_t-1} \sum_{j=0}^{H_t-1} T(x, y),$$

$$I'(x, y) = I(x, y) - \frac{1}{W_i \cdot H_i} \cdot \sum_{i=0}^{W_i-1} \sum_{j=0}^{H_i-1} I(x, y),$$

kde W_i je šířka vstupního obrázku a H_i jeho výška.

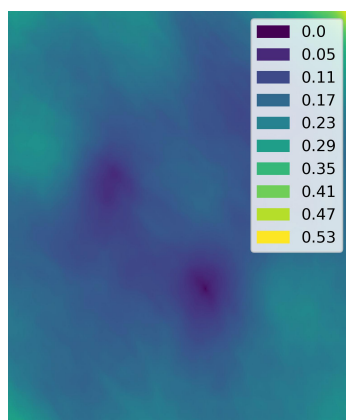
Výsledek této metody za použití šedotónové reprezentace můžete vidět na obrázku 3.5.



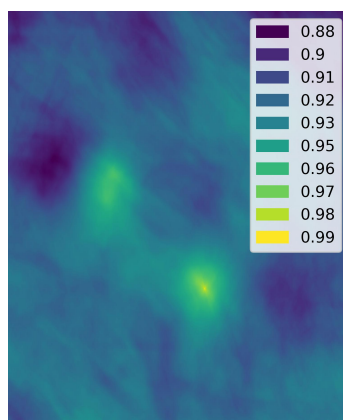
Obrázek 3.1: Vstupní obrázek.



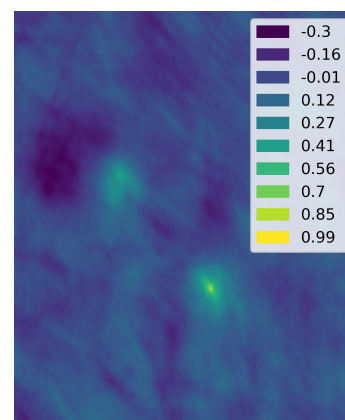
Obrázek 3.2: Vzor.



Obrázek 3.3: Kvadratická chyba.



Obrázek 3.4: Vzájemná korelace.



Obrázek 3.5: Korelační koeficienty.

3.2 Logistická regrese

Logistická regrese (viz Bishop (2006)) je algoritmus, který dokáže klasifikovat vstupní data do dvou či více tříd. Je založena na maximalizování věrohodnosti přes váhy regrese. Níže budeme vstupní obrázek označovat písmenem ϕ , a jeho skutečnou třídu písmenem t .

Pro daný obrázek ϕ a váhy modelu \mathbf{w} je pravděpodobnost, že obrázek patří do třídy C_1 , daná následujícím vzorcem:

$$p(C_1|\phi) = \hat{y}(\phi) = \sigma(\mathbf{w}^T \phi),$$
$$\sigma(a) = \frac{1}{1 + \exp(-a)},$$

kde \mathbf{w}^T jsou váhy logistické regrese velikosti jednoho obrázku a $\hat{y}(\phi)$ je predikce obrázku ϕ .

Věrohodnostní funkci p pro data $\{\phi_n, t_n\}$, kde $t_n \in \{0, 1\}$, pro $n = 1, \dots, N$ můžeme napsat následovně:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^N \hat{y}_n^{t_n} \{1 - \hat{y}_n\}^{1-t_n},$$
$$\mathbf{t} = (t_1, \dots, t_N),$$
$$\hat{y}_n = p(C_1|\phi_n).$$

Když už máme věrohodnostní funkci, můžeme si definovat chybovou funkci $E(\mathbf{w})$ a spočítat její gradient $\nabla E(\mathbf{w})$:

$$E(\mathbf{w}) = -\ln(p(\mathbf{t}|\mathbf{w})) = -\sum_{n=1}^N \{t_n \ln(\hat{y}_n) + (1 - t_n) \ln(1 - \hat{y}_n)\},$$
$$\nabla E(\mathbf{w}) = \sum_{n=1}^N (\hat{y}_n - t_n) \phi_n.$$

Nyní se můžeme podívat na samotný algoritmus logistické regrese. Algoritmus opakovaně upravuje váhy \mathbf{w} podle gradientu chybové funkce a Hessovy matice, dokud neklasifikuje všechna trénovací data korektně nebo pokud dosáhneme maximálního počtu iterací.

Nejdříve nastavíme všechny váhy na nulu a poté v každé iteraci upravím váhy takto:

$$\mathbf{w}^{(new)} = \mathbf{w}^{(old)} - H^{-1} \cdot \nabla E(\mathbf{w}),$$
$$H = \nabla \nabla E(\mathbf{w}) = \sum_{n=1}^N \phi_n \cdot \phi_n^T,$$

kde H je Hessova matice.

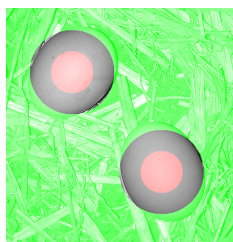
My ale použijeme zjednodušenou verzi logistické regrese a nahradíme Hessovu matici konstantou α následujícím způsobem:

$$\mathbf{w}^{(new)} = \mathbf{w}^{(old)} - \alpha \cdot \nabla E(\mathbf{w}),$$

kde α je míra trénování. Čím větší α je, tím rychleji se regrese trénuje, ale tím méně je přesná.

Použití logistické regrese si ukážeme na příkladu, kde budeme používat prstencovou reprezentaci obrázku. Nejdříve natrénujeme regresi na trénovacím obrázku (viz obrázek 3.6), kde červené pixely označují vajíčka, zelené pixely místa, kde vajíčka nejsou, a šedé pixely nepoužijeme, protože není jasné, jak by měli být klasifikováni.

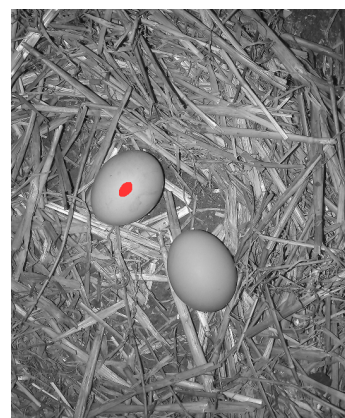
Poté predikujeme vajíčka pomocí již natrénované regrese. Predikci testovacího obrázku 3.7 můžete vidět na obrázku 3.8. Na výsledku je ale vidět, že regrese nefunguje úplně správně, nenašla totiž vajíčko, které bylo v trénovacím obrázku, a proto použijeme i jiné komplikovanější metody.



Obrázek 3.6: Trénovací obrázek.



Obrázek 3.7: Vstupní obrázek.



Obrázek 3.8: Predikce vajíček.

3.3 Neuronové sítě

Neuronová síť (viz Bishop (2006)) je klasifikační model založený na umělých neuronech. Tento neuron je uzel sítě, který podle hodnot na všech jeho vstupech, ohodnotí svůj výstup. My použijeme dopřednou vrstevnatou neuronovou síť (Multi Layer Perceptron), která se skládá z vrstev neuronů, které jsou mezi jednotlivými vrstvami propojeny.

Popišme si blíže chování j -tého neuronu jedné vrstvy. Mějme vstupní hodnoty z předchozí vrstvy, velikosti D , x_1, \dots, x_D , dále mějme váhy daného neuronu w_{j0}, \dots, w_{jD} . Vypočítejme lineární kombinaci vah a vstupních hodnot následujícím vzorcem:

$$a_j = \sum_{i=1}^D w_{ji} \cdot x_i + w_{j0},$$

kde a_j je lineární kombinace j -tého neuronu.

Abychom získaly výstupní hodnotu z_j daného neuronu transformujeme a_j aktivační funkcí h :

$$z_j = h(a_j).$$

My budeme používat dvě aktivační funkce. V poslední vrstvě použijeme sigmoid funkci $\sigma(a)$:

$$\sigma(a) = \frac{1}{1 + e^{-a}}.$$

A ve skrytých vrstvách si zvolíme *ReLU* funkci:

$$ReLU(a) = \max(0, a).$$

3.3.1 Trénování neuronové sítě

Když už víme jak vypočítat výsledek neuronové sítě $\mathbf{y}(\mathbf{x}, \mathbf{w})$, definujme si chybovou funkci $E(\mathbf{w})$, kterou budeme minimalizovat. Mějme vstupní vektory sítě \mathbf{x}_n , kde $n = 1, \dots, N$, a jejich odpovídající správné výstupní vektory \mathbf{t}_n , dále mějme váhy sítě \mathbf{w} , pak chybová funkce je daná vzorcem:

$$E(\mathbf{w}) = \frac{1}{2} \cdot \sum_{n=1}^N |\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{t}_n|^2.$$

Abychom dosáhli nejmenší hodnoty chybové funkce, budeme se snažit dosáhnout nulového gradientu $\nabla E(\mathbf{w})$. Protože neexistuje žádný známý vzorec na váhy w , který by odpovídal nulovému gradientu, tak váhy nejdříve nastavíme na malé náhodné hodnoty a poté je budeme iterativně upravovat následujícím způsobem:

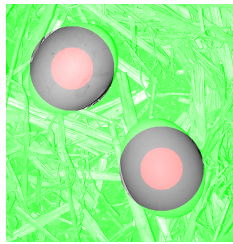
$$\mathbf{w}^{(new)} = \mathbf{w}^{(old)} - \alpha \cdot \nabla E(\mathbf{w})$$

kde α je míra trénování a $\nabla E(\mathbf{w})$ je gradient chybové funkce.

3.3.2 Použití neuronové sítě

Použití neuronové sítě si ukážeme stejně jako u logistické regrese na prstencové reprezentaci obrázku. Nejdříve síť natrénujeme na trénovacím obrázku (viz obrázek 3.9), kde červené pixely označují vajíčka, zelené pixely, kde vajíčka nejsou, a šedé pixely nepoužijeme, protože není jasné, jak by měli být klasifikováni.

Poté predikujeme vajíčka pomocí natrénované neuronové sítě. Predikci testovacího obrázku 3.10 můžete vidět na obrázku 3.11. Na výsledku je vidět, že neuronová síť sice nerozpoznala středy vajíček přesně, ale detekovala je obě správně, tudíž by mohla posloužit jako poměrně dobrý klasifikátor.



Obrázek 3.9: Trénovací obrázek.



Obrázek 3.10: Vstupní obrázek.



Obrázek 3.11: Predikce vajíček.

4. Hledání komponent

Nyní se podíváme na metodu, jak z matice klasifikovaných pixelů získat pozice jednotlivých vajíček. Použijeme k tomu algoritmus na hledání komponent souvislosti se specificky definovanou sousedností $S(p, q)$:

$$S(p, q) = 1 \iff \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} \leq MAX_DIST,$$

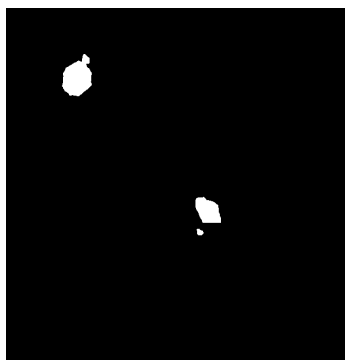
kde $p = (p_x, p_y)$ a $q = (q_x, q_y)$ jsou pixely, MAX_DIST je maximální vzdálenost pro sousednost.

Pseudokód tohoto algoritmu můžete vidět zde:

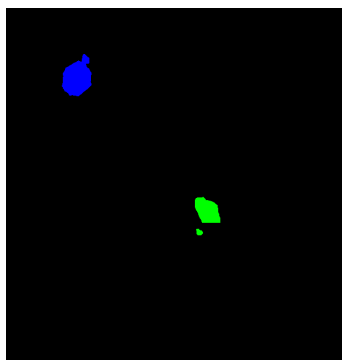
1. pro každý pozitivní pixel, který není v žádné komponentě
2. vytvoř novou komponentu a přidej do ní tento pixel
3. opakuj dokud komponenta roste
4. přidej do komponenty všechny pozitivní pixely,
které jsou od komponenty blíže než maximální vzdálenost
5. přidej novou komponentu do nalezených komponent

Mějme tedy matici klasifikovaných pixelů (viz obrázek 4.1). Z této matice získáme komponenty, tak že seskupíme pozitivně klasifikované pixely do shluků, které obsahují pouze dostatečně blízké pixely. Výsledné shluky můžete vidět na obrázku 4.2.

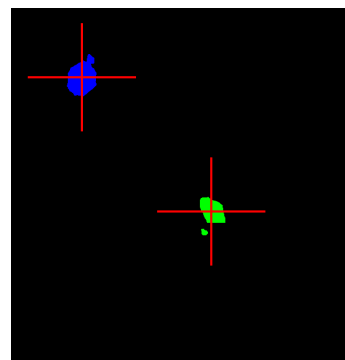
Poté jednotlivé pozice vajíček vypočítáme jako průměrný pixel v každém shluku. Pozice vajíček označené červeným křížkem můžete vidět na obrázku 4.3.



Obrázek 4.1: Matice klasifikovaných pixelů.



Obrázek 4.2: Klasifikované shluky.



Obrázek 4.3: Průměry shluků.

5. Snímání obrázku

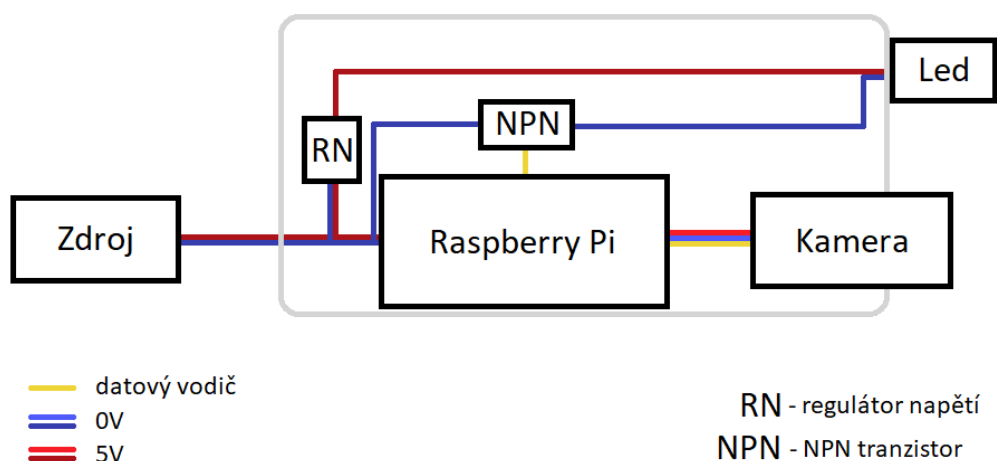
Nyní se podíváme na samotné získávání obrazových dat. Obrázky snímá minipočítač „Raspberry Pi Zero“ s připojenou kamerou. Následně je odesílá na server samotného rozpoznávače vajíček.

5.1 Hardware

Hardware zařízení na snímání obrázků oponuje poměrně výkonným minipočítačem „Raspberry Pi Zero 2W“ s připojenou kamerou „Raspberry Pi kamera V2“, a SD kartou o kapacitě 64 GB.

Co se týče osvětlení snímané plochy, tak je použita LED dioda, která je napájena pomocí regulátoru napětí a spínaná NPN tranzistorem. Celé zařízení je napájeno zdrojem s napětím 5 V a proudem 1 A.

Schéma zapojení můžete vidět na obrázku 5.1, a náhled snímacího zařízení na obrázku 5.2.



Obrázek 5.1: Schéma zapojení snímacího zařízení.



Obrázek 5.2: Náhled snímacího zařízení.

5.2 Software

Software snímacího zařízení je poměrně jednoduchá smyčka, která nejprve zapne LED diodu, poté vyfotografuje obrázek, a nakonec odešle obrázek na server do SQL databáze. Zde můžete vidět pseudo kód této smyčky:

1. opakuj
 2. zapni LED
 3. vyfoť fotku
 4. vypni LED
 5. odešli fotku na server

Obrázky jsou snímány ve formátu JPEG po dobu 10 ms. Kvalitu, rozlišení, interval snímání a další parametry lze nastavit ve webové aplikaci (viz kapitola 6.2).

6. Webový server

Webový server zprostředkovává komunikaci mezi uživatelem, snímacím zařízením a klasifikátorem vajíček.

Nejdříve snímací zařízení uloží obrázek do SQL databáze. Tento obrázek si nahraje klasifikátor, který najde vajíčka a výsledek uloží do této databáze. Následně si uživatel může zobrazit nalezená vajíčka na webových stránkách, které získávají data ze zmíněné databáze.

Na databázi a webové stránky jsem použil balíček XAMPP.

6.1 Databáze

Na ukládání dat jsem použil databázi typu MySQL, která obsahuje dvě tabulky. První tabulka obsahuje nastavení kamer, které jsou k aplikaci připojené, a druhá obsahuje obrázky a data z těchto kamer.

6.1.1 Nastavení kamer

Tabulka, v které je uloženo nastavení, obsahuje ID kamery a jednotlivé parametry snímání. Strukturu tabulky si můžete prohlédnout v tabulce 6.1.

Název	Popis	Typ
camera_id	ID kamery	integer
shooting_interval	Interval snímání v ms	integer
to_server	Zda se mají obrázky ukládat do databáze nebo do úložiště snímače	boolean
insert_new	Zda se mají vytvářet nové obrázky, nebo přepisovat ty staré	boolean
img_width	Šířka obrázku v pixelech	integer
img_height	Výška obrázku v pixelech	integer
quality	Kvalita JPEG obrázku	integer

Tabulka 6.1: Struktura tabulky nastavení kamer

S těmito daty pracují snímací zařízení, jejichž chování odpovídá aktuálnímu nastavení, a webové stránky, na kterých je možné nastavení změnit (viz kapitola 6.2).

6.1.2 Data kamer

Tabulka, v které jsou uložena data, obsahuje ID obrázku, ID kamery, vstupní obrázek, výstupní obrázek a další. Strukturu tabulky si můžete prohlédnout v tabulce 6.2.

Název	Popis	Typ
id	ID obrázku	integer
camera_id	ID kamery	integer
date_time	Datum a čas nahrání vstupního obrázku	datetime
image	Vstupní obrázek	longblob
processed	Zda už byl vstupní obrázek zpracován a vygenerován výstupní obrázek	boolean
result	Výstupní obrázek	longblob
egg_count	Počet nalezených vajíček	integer
result_date_time	Datum a čas nahrání výstupního obrázku	datetime
locations	Souřadnice nalezených vajíček	array

Tabulka 6.2: Struktura tabulky dat kamer

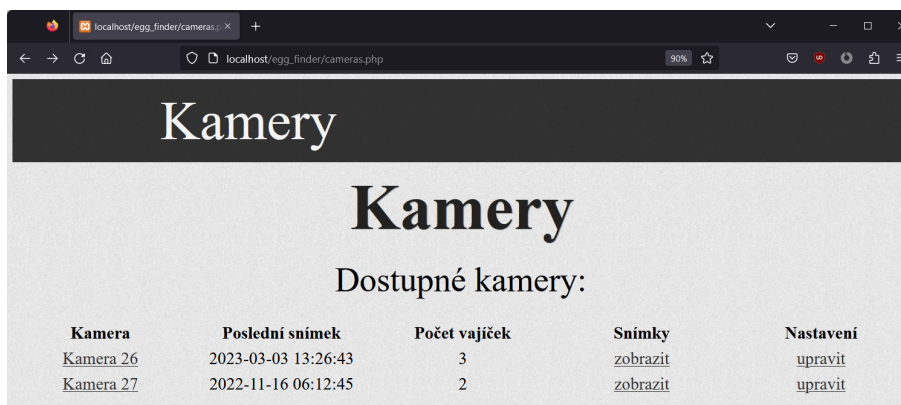
Vstupní obrázek odpovídá obrázku, který nahrálo snímací zařízení do databáze. A výstupní obrázek odpovídá vstupnímu obrázku, na kterém jsou označena nalezená vajíčka.

S těmito daty pracují snímací zařízení, které nahrávají vstupní obrázky, klasifikátor, který generuje výstupní data, a webové stránky, na kterých je možné si data zobrazit.

6.2 Webové stránky

Webové stránky slouží jako uživatelské prostředí. Je v nich možné měnit nastavení kamer, prohlížet si nafocené obrázky a kontrolovat počet vajíček na jednotlivých snímcích.

Na hlavní stránce (viz obrázek 6.1) je dostupný seznam kamer, které už nahrály nějakou fotku. U každé kamery je vidět identifikátor kamery, datum a čas posledního snímku, počet nalezených vajíček, odkaz na vyfocené snímky a odkaz na nastavení.



Kamera	Poslední snímek	Počet vajíček	Snímky	Nastavení
Kamera 26	2023-03-03 13:26:43	3	zobrazit	upravit
Kamera 27	2022-11-16 06:12:45	2	zobrazit	upravit

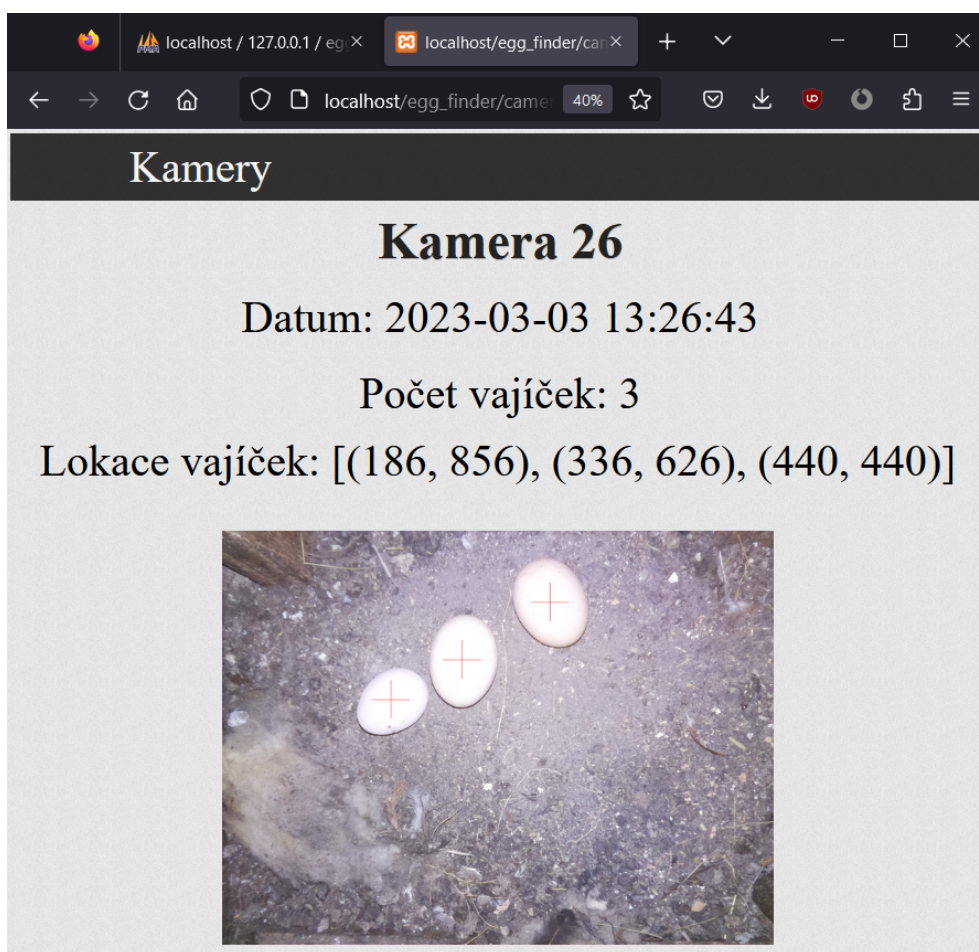
Obrázek 6.1: Hlavní stránka webových stránek.

Výstup dané kamery (viz obrázek 6.2) si lze prohlédnout kliknutím na odkaz „Kamera XX“, kde „XX“ označuje ID kamery.

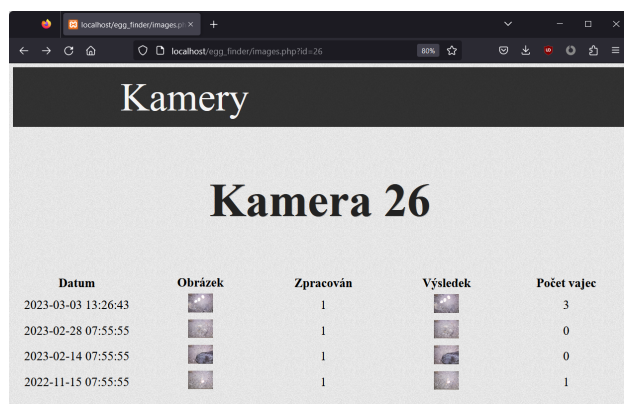
Všechny uložené snímky dané kamery (viz obrázek 6.3) si lze prohlédnout kliknutím na odkaz „zobrazit“ ve sloupci „Snímky“.

U každé kamery je možné změnit její nastavení kliknutím na odkaz „upravit“ ve sloupci „Nastavení“. Stránku nastavení můžete vidět na obrázku 6.4.

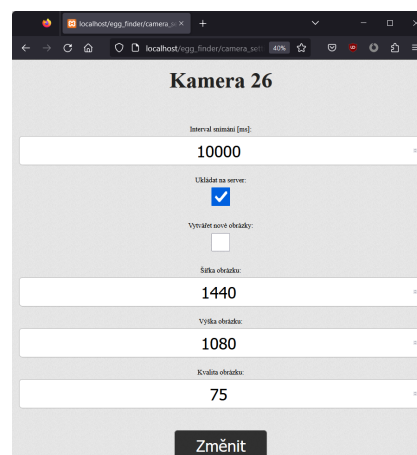
Podrobnější informace o nastavení můžete najít v manuálu.



Obrázek 6.2: Webové stránky - výstup kamery.



Obrázek 6.3: Webové stránky - snímky kamery.



Obrázek 6.4: Webové stránky - nastavení kamery.

7. Klasifikační program

V této kapitole vyzkoušíme různé metody na hledání vajíček. Nejprve si připravíme trénovací a testovací data, poté budeme tyto metody nastavovat a testovat. Nakonec vybereme nejúspěšnější metodu a vytvoříme samotný klasifikační program.

7.1 Příprava dat

Nejprve si nasnímáme obrazová data. Snímky budeme zachycovat každých 20 minut v jedné slepičárně po dobu asi jednoho měsíce. Fotit budeme pomocí zařízení Raspberry Pi s kamerou (viz kapitola 5). Ve výsledku nasnímáme 2921 snímků. Příkladový obrázek můžete vidět na obrázku 7.1.



Obrázek 7.1: Příkladový obrázek z kamery.

Takto nasímaná data ale obsahují mnoho duplicitních obrázků. Proto tyto obrázky před testováním odstraníme. Tím nám zbude 514 snímků, kde každý snímek je alespoň trochu odlišný od ostatních.

Když už máme připravené snímky, tak přichází řada na náhodné rozdělení snímků na trénovací, validační a testovací. Data rozdělíme v poměru 4:2:3.

Tím získáme 228 trénovacích, 144 validačních a 172 testovacích obrázků. Tyto obrázky dohromady obsahují 182 trénovacích, 97 validačních a 102 testovacích vajíček.

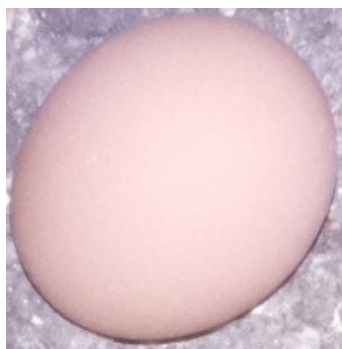
Abychom mohli testovat úspěšnost jednotlivých metod, musíme ručně na každém snímku označit pozice vajíček. To uděláme pomocí mého programu, který používá knihovnu `matplotlib`.

Označovat budeme dva druhy vajíček. První typ budou vajíčka, které klasifikační program musí najít, pokud je nenajde budou označeny jako falešně negativní výsledky. Tento typ odpovídá vajíčkům, která nejsou nijak překrytá (špinavá vajíčka se nepočítají jako překrytá) a jsou celá na obrázku. Druhý typ budou vajíčka, která program může rozpoznat, ale když je nerozpozná, nepočítají se jako falešně

negativní výsledky. Tento typ odpovídá vajíčkům, která jsou nečím překrytá nebo vyčnívají z obrázku, nebo skořápkám, které tvoří alespoň půlku povrchu vajíčka. Příklady prvního typu můžete vidět na obrázcích 7.2 - 7.4 a příklady druhého typu na obrázcích 7.5 - 7.7.



Obrázek 7.2: Příklad 1 vajíčka typu 1.



Obrázek 7.3: Příklad 2 vajíčka typu 1.



Obrázek 7.4: Příklad 3 vajíčka typu 1.



Obrázek 7.5: Příklad 1 vajíčka typu 2.



Obrázek 7.6: Příklad 2 vajíčka typu 2.



Obrázek 7.7: Příklad 3 vajíčka typu 2.

7.2 Ohodnocení klasifikátoru

Abychom mohli jednotlivé klasifikátory porovnávat a hodnotit, musíme si zadefinovat hodnotící kritérium. Jako hodnotící kritérium použijeme F-skóre, které vypočítáme z počtu pravdivě pozitivních (TP), falešně pozitivních (FP) a falešně negativních (FN) výsledků, podle následující rovnice:

$$F = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (7.1)$$

Pravdivě pozitivní bude případ, kdy klasifikátor klasifikuje vajíčko na pozici, která je dostatečně blízká pozici ručně označeného vajíčka prvního nebo druhého typu (viz kapitola 7.1).

Falešně pozitivní bude situace, kdy klasifikátor klasifikuje vajíčko v místě, kde žádné ručně označené vajíčko není.

A Falešně negativní bude případ, kdy klasifikátor neklasifikuje žádné vajíčko v pozici, kde je ručně označené vajíčko prvního typu.

7.3 Porovnání se vzory

Jako první metodu použijeme porovnávání se vzory (viz kapitola 3.1), která porovnává maticové reprezentace obrázku.

Nejprve extrahujeme vzory vajíček z trénovacích dat, tím získáme 173 obrázků reprezentující vajíčka prvního typu. Těchto vzorů je ale příliš mnoho, a proto z nich na klasifikaci budeme vybírat pouze některá.

Všechny následující testy poběží na notebooku s procesorem „Intel Core i5-10300H @ 2.50GHz“ a pamětí RAM velikosti 8 GB. Na tomto počítači každý test poběží rámcově jednotky hodin (2 až 15), a proto budeme většinou testovat pouze jeden parametr, aby nám testování netrvalo příliš dlouho.

7.3.1 Výběr vzorů

Vybírat vzory budeme pomocí metody grid search, která bude zkoušet jednotlivé kombinace vzorů a podle F-skóre validačních dat vrátí nejlepší kombinaci vzorů. Budeme zkoušet náhodné kombinace, můj výběr vzorů a výběr vzorů, vypočítaný pomocí níže popsaného algoritmu.

Algoritmus na hledání nejlepší množiny vzorů nejprve roztřídí všechny vzory pomocí Gaussian Mixture klastrování a poté najde medián v každém klastru. Výsledná množina vzorů odpovídá právě těmto mediánům.

Pojďme tedy otestovat jednotlivé kombinace vzorů velikosti 12, které budou obsahovat moji kombinaci, kombinaci vygenerovanou předchozím algoritmem a 25 náhodných kombinací. Testovat budeme na šedotónové reprezentaci obrázků a jako chybovou funkci použijeme normované korelační koeficienty.

V tabulce s nejlepšími výsledky pro všechna generování (viz tabulka 7.1) můžeme vidět, že ručně vybrané vzory dávají nejlepší výsledky. Vzory generované zmíněným algoritmem mají také poměrně dobrý výsledek. A co se týče náhodně generovaných vzorů, tak vzhledem k počtu generací je jejich nejlepší výsledek také velice dobrý.

Generování vzorů	TP	FP	FN	F-skóre
Ručně vybrané	88	3	14	0,912
Generované algoritmem	84	7	14	0,889
Náhodně generované	88	5	15	0,898

Tabulka 7.1: Výsledky výběru vzorů

Vzhledem k těmto výsledkům si jako vzory pro následující testování zvolíme ručně vybrané vzory.

7.3.2 Výběr chybové funkce

Když už máme vybrané vzory, tak přichází řada na výběr chybové funkce. Tyto funkce budeme vybírat stejným způsobem jako vzory, tedy pomocí metody grid search, která vyzkouší jednotlivé chybové funkce a porovná jejich F-skóre.

Vyzkoušíme 3 chybové funkce, které jsou zmíněné v kapitolách 3.1.1 - 3.1.3, tedy normovanou kvadratickou chybu (SQDIFF), normovanou vzájemnou korelací (CCORR) a normované korelační koeficienty (CCOEFF). Pojďme tedy otestovat jednotlivé chyby na šedotónové reprezentaci obrázku.

V tabulce s výsledky pro jednotlivé chybové funkce (viz tabulka 7.2) můžeme vidět, že normované korelační koeficienty mají velice dobrý výsledek. Naopak normovaná kvadratická chyba a normovaná vzájemná korelace mají výsledky velmi špatné.

Chybová funkce	TP	FP	FN	F-skóre
SQDIFF	59	36	40	0,608
CCORR	65	50	34	0,607
CCOEFF	88	3	14	0,912

Tabulka 7.2: Výsledky výběru chybové funkce

Na základě těchto výsledků si jako chybovou funkci pro další klasifikaci zvolíme normované korelační koeficienty.

7.3.3 Výběr reprezentace obrázku

Nyní přichází řada na výběr reprezentace obrázku a příslušných hranic, kdy je pixel klasifikován pozitivně. Znovu využijeme metodu grid search pro jednotlivé parametry. Hranicí je myšlena minimální hodnota normovaných korelačních koeficientů, která je potřeba pro pozitivní klasifikaci.

Testovat budeme reprezentace zmíněné v kapitole 2. Tedy šedotónovou reprezentaci vypočítanou jako vážený průměr, jednotlivé barevné složky (červená, zelená, modrá), hrany detekované horní propustí a hrany detekované Kirschem detektorem. U každé reprezentace otestujeme sedm hranic, které se pohybují okolo hodnot, které vrací porovnání s danou reprezentací.

V tabulce s výsledky pro jednotlivé reprezentace a nejlepší hranice (viz tabulka 7.3) můžeme vidět, že modré spektrum a horní propust mají výsledky velmi špatné. Zatímco šedotón, červené a zelené spektrum a hrany detekované Kirschem mají výsledky poměrně dobré.

Reprezentace	Hranice	TP	FP	FN	F-skóre
Šedotón	0,70	88	3	14	0,912
Červená	0,70	90	6	12	0,909
Zelená	0,68	92	7	11	0,911
Modrá	0,68	79	13	21	0,823
Horní propust	0,33	77	20	26	0,770
Kirsch	0,46	90	3	13	0,918

Tabulka 7.3: Výsledky výběru reprezentace obrázku

Vzhledem k těmto výsledkům si jako reprezentaci obrázku zvolíme hrany detekované Kirschem detektorem a jako hranici si zvolíme hodnotu 0,46.

7.3.4 Hodnocení

Nyní pustíme hledání vajíček s nejlepšími parametry na testovací množinu. Tím získáme výsledek, kde počet pravdivě pozitivních výsledků je 103, počet falešně pozitivních 1, počet falešně negativních 11 a doba hledání vajíček v jednom obrázku je 1,157 sekund. Z toho vypočítáme F-skóre, které se rovná 0,945.

Tento výsledek je poměrně dobrý, ale zkusme ještě využít prstencovou reprezentaci obrázku místo maticové.

7.4 Porovnání s prstencovými vzory

Nyní budeme testovat druhou metodu, která také používá porovnání se vzory (viz kapitola 3.1), ale porovnává prstencovou reprezentaci obrázku (viz kapitola 1.2) a ne maticovou. Jako vzory znovu použijeme vajíčka prvního typu z trénovacích dat. Prstencovou reprezentaci budeme počítat v kružnicích o poloměrech v intervalu [1, 25].

7.4.1 Výběr vzorů

Na výběr vzorů znovu použijeme metodu grid search. Budeme zkoušet náhodné kombinace, můj výběr vzorů a výběr vzorů, vypočítaný pomocí algoritmu pospaného v kapitole 7.3.1.

Pojďme tedy otestovat jednotlivé kombinace vzorů velikosti 12 (moji kombinace, kombinace vygenerovaná algoritmem a 25 náhodných kombinací). Testovat budeme na šedotónové reprezentaci obrázků a jako chybovou funkci použijeme kvadratickou chybu.

V tabulce s nejlepšími výsledky pro všechna generování (viz tabulka 7.4) můžeme vidět, vzory vygenerované zmíněným algoritmem jsou velice špatné. Naopak ručně vybrané vzory i nejlepší náhodné vzory mají výsledky poměrně dobré.

Generování vzorů	TP	FP	FN	F-skóre
Ručně vybrané	92	19	16	0,840
Generované algoritmem	80	24	23	0,773
Náhodně generované	93	20	15	0,842

Tabulka 7.4: Výsledky výběru vzorů

Vzhledem k těmto výsledkům si jako vzory zvolíme nejlepší kombinaci vzorů, která byla náhodně vygenerována.

7.4.2 Výběr chybové funkce

Nyní budeme vybírat chybovou funkci. Využijeme metodu grid search, které bude testovat následující chybové funkce: kvadratickou chybu (SQDIFF), vzájemnou korelaci (CCORR), korelační koeficienty (CCOEFF) a jejich normované varianty (SQDIFF_NORMED, CCORR_NORMED, CCOEFF_NORMED). Testovat budeme klasifikaci, která porovnává šedotónové reprezentace obrázků.

V tabulce s výsledky pro jednotlivé chybové funkce (viz tabulka 7.5) můžeme vidět, že jednoznačně nejlepší jsou obě varianty kvadratických chyb. Normovaná varianta je o něco lepší, a proto si ji zvolíme jako chybovou funkci pro další testování.

Chybová funkce	TP	FP	FN	F-skóre
SQDIFF	93	20	15	0,842
CCORR	8	6	91	0,142
CCOEFF	32	38	76	0,360
SQDIFF_NORMED	89	16	17	0,844
CCORR_NORMED	15	6	84	0,250
CCOEFF_NORMED	47	20	60	0,540

Tabulka 7.5: Výsledky výběru chybové funkce

7.4.3 Výběr reprezentace obrázku

Nyní otestujeme jednotlivé reprezentace obrázku a příslušné hranice pomocí metody grid search.

Testovat budeme reprezentace zmíněné v kapitole 2. Tedy šedotónovou, jednotlivé barevné složky (červená, zelená, modrá), hrany detekované horní propustí a hrany detekované Kirscheho detektorem. U každé reprezentace otestujeme pět hranic, které se pohybují okolo hodnot, které vrací porovnání s danou reprezentací.

V tabulce s výsledky pro jednotlivé reprezentace a nejlepší hranice (viz tabulka 7.6) můžeme vidět, že nejlépe je na tom šedotónová, červená a Kirschova reprezentace. Ale vzhledem k tomu, že jsem očekával daleko lepší výsledek u Kirschovy hranové reprezentace, tak ještě jednou zkusíme najít nejlepší vzory pro tuto reprezentaci.

Reprezentace	Hranice	TP	FP	FN	F-skóre
Šedotón	$8 \cdot 10^{-05}$	92	16	16	0,852
Červená	$7 \cdot 10^{-05}$	84	6	22	0,857
Zelená	$8 \cdot 10^{-05}$	90	22	18	0,818
Modrá	$6 \cdot 10^{-05}$	86	39	19	0,748
Horní propust	0,004	66	16	35	0,721
Kirsch	0,0019	97	10	20	0,853

Tabulka 7.6: Výsledky výběru reprezentace obrázku

Po testování ručně vybraných, algoritmem generovaných a náhodných vzorů jsme našli kombinaci vzorů s 87 pravdivě pozitivními případy, 5 falešně pozitivními a 20 falešně negativními. To odpovídá F-skóre rovno 0,874. Vzhledem k tomuto skóre pro další testování zvolíme právě tyto vzory a jako reprezentaci zvolíme hrany detekované Kirscheho detektorem.

7.4.4 Hodnocení

Nyní pustíme hledání vajíček s nejlepšími parametry na testovací množinu. Tím získáme výsledek, kde počet pravdivě pozitivních výsledků je 103, počet falešně pozitivních 5, počet falešně negativních 15 a doba hledání vajíček v jednom obrázku je 4,970 sekund. Z toho vypočítáme F-skóre, které se rovná 0,912.

Tento výsledek je horší než u maticové reprezentace (0,945). Je možné, že je to tím, že máme málo vzorů. Ale kdybychom jejich počet zvýšili, tak hledání v jednom obrázku by trvalo ještě delší dobu. Proto bychom mohli zkusit nejdříve natrénovat nějaký model (logistická regrese nebo neuronové sítě) na všech trénovacích datech a poté pomocí tohoto modelu vajíčka predikovat.

7.5 Logistická regrese

Nyní jako klasifikátor vyzkoušíme logistickou regresi (viz kapitola 3.2) na prstencových reprezentacích obrázku. Budeme hledat její nejlepší parametry pomocí metody grid search.

7.5.1 Výběr reprezentace obrázku

Nejdříve najdeme nejlepší reprezentaci obrázků a hranici, kdy je pixel klasifikován pozitivně. U logistické regrese je hranice hodnota od 0 do 1, která udává, jak moc musí být pravděpodobné, že je na pixelu vajíčko, abychom tento pixel klasifikovaly pozitivně.

V tabulce s výsledky pro jednotlivé reprezentace a nejlepší hranice (viz tabulka 7.7) můžeme vidět, že modré spektrum má velice špatný výsledek. Červené a zelené spektrum a horní propust mají výsledky poměrně dobré. A nejlépe na tom je šedotón a hrany detekované Kirscheho detektorem.

Reprezentace	Hranice	TP	FP	FN	F-skóre
Šedotón	0,93	113	21	4	0,900
Červená	0,87	104	27	11	0,846
Zelená	0,70	112	36	5	0,845
Modrá	0,77	79	36	34	0,693
Horní propust	0,81	88	13	19	0,846
Kirsch	0,98	90	1	18	0,905

Tabulka 7.7: Výsledky výběru reprezentace obrázku

Vzhledem k těmto výsledkům si pro další testování zvolíme Kirschovu reprezentaci.

7.5.2 Ohodnocení trénovacích dat

Nyní si zvolíme jakým způsobem ohodnotíme trénovací data před samotný trénováním. Budeme nastavovat a testovat dva parametry. Prvním bude poloměr pozitivně klasifikovaného kruhu okolo středu vajíčka prvního typu. A druhý bude poloměr neutrálního kruhu, jehož pixely nebudou použity jako trénovací data, okolo středu vajíčka obou typů.

Pojďme tedy otestovat jednotlivé parametry pomocí metody grid search, která otestuje jednotlivé klasifikační hranice a vybere tu nejlepší.

V tabulce s výsledky pro jednotlivé parametry a příslušné hranice (viz tabulka 7.8) můžeme vidět, že nejlépe je na tom pozitivní poloměr hodnoty 5 a neutrální poloměr hodnoty 20, které dohromady dosahují na validačních datech F-skóre dokonce 0,931. Zvolíme si je tedy jako poloměry pro další testování.

Pozitivní r	Neutrální r	Hranice	TP	FP	FN	F-skóre
5	0	0,03	93	1	16	0,916
5	10	0,09	96	2	13	0,928
5	20	0,47	95	0	14	0,931
5	30	0,67	94	1	14	0,926
5	40	0,85	90	0	17	0,914
10	0	0,13	97	4	13	0,919
10	30	0,95	90	0	17	0,914
10	40	0,97	90	0	17	0,914
20	0	0,83	90	1	18	0,905
20	30	0,98	90	1	18	0,905
20	40	0,98	90	1	18	0,905
30	0	0,91	89	5	18	0,886
30	40	0,97	88	3	19	0,889

Tabulka 7.8: Výsledky výběru ohodnocení trénovacích dat

7.5.3 Kombinace reprezentací

Vzhledem k tomu, že výsledky na validačních datech nejsou až tak dobré, tak zkusíme jako příznaky obrázku použít kombinaci více reprezentací. Tedy místo vektoru příznaků, který obsahuje jednu prstencovou reprezentaci, použijeme vektor, který je složený z více reprezentací.

Budeme testovat různé kombinace červeného (R), zeleného (G) a modrého (B) spektra, šedotónu (S), horní propusti (H) a hran detekovaných Kirscheho detektorem (K). Ke každé kombinaci najdeme také nejlepší klasifikační hranici.

V tabulce s výsledky pro jednotlivé kombinace a nejlepší hranice (viz tabulka 7.9) můžeme vidět, že nejlépe je na tom kombinace Kirscheho hran, horní propusti a všech barevných složek. I přestože doba hledání jednoho obrázku

je zhruba pětinasobná oproti hledání pomocí jedné reprezentace, tak si tuto kombinaci zvolíme pro další testování. Dlouhá doba hledání nám víceméně nevadí z toho důvodu, že scéna, kde jsou vajíčka, je hodně statická.

Reprezentace	Hranice	TP	FP	FN	F-skóre
K	0,47	95	0	14	0,931
K + S	0,97	96	1	14	0,928
K + R	0,93	100	7	10	0,922
K + G	0,97	95	1	15	0,922
K + RG	0,97	99	6	12	0,917
K + RGB	0,93	105	6	7	0,942
H + RGB	0,93	101	7	8	0,931
K + H + RGB	0,94	105	5	7	0,946
K + H + RGB + S	0,96	104	4	8	0,945

Tabulka 7.9: Výsledky výběru kombinací reprezentací

7.5.4 Hodnocení

Nyní přichází řada na otestování nalezených parametrů. Natrénujeme si tedy logistickou regresi na všech trénovacích i validačních datech, a poté klasifikujeme obrázky z testovací množiny.

Tím získáme výsledek, kde počet pravdivě pozitivních výsledku je 116, počet falešně pozitivních 1, počet falešně negativních 2, doba trénování je 1 hodina a 46 minut a doba hledání vajíček v jednom obrázku je 11,906 sekund. Z toho vypočítáme F-skóre, které se rovná 0,987.

Tento výsledek je až na dobu hledání velice dobrý. Falešně byla klasifikováno pouze jedna část opeření slepice (viz obrázek 7.8) a klasifikovány nebyly dvě špičková vajíčka (viz obrázky 7.9 a 7.10).



Obrázek 7.8: Falešně pozitivní výsledek.



Obrázek 7.9: Falešně negativní výsledek.



Obrázek 7.10: Falešně negativní výsledek.

7.6 Neuronové sítě

Pojďme otestovat ještě vícevrstevnatou perceptronovou síť (viz kapitola 3.3) na prstencových reprezentacích obrázku. Znovu na nalezení nejlepších parametrů použijeme metodu grid search.

V předchozím testování na zkušebních datech z jiné kamery nám nejlépe vycházela neuronová síť se dvěma skrytými vrstvami, kde každá vrstva měla 100 neuronů. A proto budeme zpočátku používat tuto síť, poté otestujeme ještě jiné rozložení skrytých vrstev.

7.6.1 Výběr reprezentace obrázku

Nejdříve najdeme nejlepší reprezentaci obrázků a příslušné hranice stejně jako u logistické regrese.

V tabulce s výsledky pro jednotlivé reprezentace a nejlepší hranice (viz tabulka 7.10) můžeme vidět, že nejlépe je na tom horní propust a Kirscheho hrany. Ostatní reprezentace mají výsledky poměrně špatné, proto si jako reprezentaci pro další testování zvolíme hrany detekované Kirscheho detektorem.

Reprezentace	Hranice	TP	FP	FN	F-skóre
Šedotón	0,21	85	12	26	0,817
Červená	0,07	87	23	24	0,787
Zelená	0,17	77	12	32	0,778
Modrá	0,05	65	23	40	0,674
Horní propust	0,11	86	3	19	0,887
Kirsch	0,67	88	1	18	0,903

Tabulka 7.10: Výsledky výběru reprezentace obrázku

7.6.2 Ohodnocení trénovacích dat

Nyní si zvolíme poloměry pozitivně a neutrálně klasifikovaných kruhů okolo středů vajíček.

Nejprve se podíváme na poloměr pozitivního kruhu, tedy kruhu, v kterém klasifikujeme pixely pozitivně. V tabulce s výsledky (viz tabulka 7.11) můžeme vidět, že nejlepší F-skóre má poloměr rovný 20, tudíž si tento poloměr zvolíme.

Pozitivní r	Hranice	TP	FP	FN	F-skóre
5	0,67	88	1	18	0,903
10	0,93	88	0	18	0,907
20	0,89	89	0	18	0,908
30	0,97	88	3	19	0,890

Tabulka 7.11: Výsledky výběru pozitivního poloměru

Poté si vybereme nejlepší poloměr neutrálního kruhu, tedy kruhu jehož pixely nejsou do trénování vůbec zapojeny. Opět můžeme v tabulce s výsledky (viz tabulka 7.12) vidět, že nejlépe je na tom poloměr velikosti 5.

Neutrální r	Hranice	TP	FP	FN	F-skóre
0	0,85	89	1	18	0,904
5	0,77	91	1	17	0,910
10	0,73	90	3	17	0,900
20	0,89	89	0	18	0,908
30	0,85	97	9	13	0,898

Tabulka 7.12: Výsledky výběru neutrálního poloměru

Vzhledem k těmto výsledkům si tedy jako pozitivní poloměr zvolíme hodnotu 20 a jako neutrální poloměr hodnotu 5.

7.6.3 Kombinace reprezentací

Pojďme ještě otestovat různé kombinace reprezentací stejně jako u logistické regrese.

Znovu budeme testovat různé kombinace červeného (R), zeleného (G) a modrého (B) spektra, šedotónu (S), horní propusti (H) a hran detekovaných Kirscheho detektorem (K). Ke každé kombinaci najdeme také nejlepší klasifikační hranici.

V tabulce s výsledky pro jednotlivé kombinace a nejlepší hranice (viz tabulka 7.13) můžeme vidět, že nejlépe je na tom kombinace všech reprezentací. Ale protože kombinace Kirsche a červeného a zeleného spektra má velmi podobné F-skóre a doba jednoho hledání je zhruba poloviční, tak si pro další testování zvolíme právě tuto kombinaci.

Reprezentace	Hranice	TP	FP	FN	F-skóre
K	0,77	91	1	17	0,910
K + S	0,53	99	2	14	0,925
K + R	0,81	97	1	14	0,928
K + G	0,63	96	1	14	0,928
K + RG	0,81	98	1	14	0,929
K + RGB	0,83	97	1	14	0,928
H + RGB	0,55	95	1	16	0,918
K + H + RGB	0,71	107	9	8	0,926
K + H + RGB + S	0,55	110	10	6	0,932

Tabulka 7.13: Výsledky výběru kombinací reprezentací

7.6.4 Vrstvy neuronů

Nyní otestujeme, jak se budou měnit výsledky vícevrstevnatého perceptronu, když budeme měnit velikosti a počet skrytých vrstev.

V tabulce s výsledky (viz tabulka 7.14) můžeme vidět, že nejlepší výsledek dosahuje perceptron se dvěma vrstevami o velikosti 100. Má nejlepší F-skóre a nejkratší dobu hledání v jednom obrázku. A z toho důvodu ho následně použijeme.

Skryté vrstvy	Hranice	TP	FP	FN	F-skóre	doba
(100)	0,97	98	2	14	0,925	9,5 s
(200)	0,45	98	2	14	0,925	11,5 s
(300)	0,77	98	1	14	0,929	13,5 s
(500)	0,89	97	1	14	0,928	17,5 s
(1000)	0,43	98	2	14	0,925	30,5 s
(100, 100)	0,81	98	1	14	0,929	10,0 s
(200, 200)	0,59	98	1	14	0,929	13,0 s
(300, 300)	0,59	98	1	14	0,929	15,0 s

Tabulka 7.14: Výsledky výběru skrytých vrstev

7.6.5 Hodnocení

Nyní přichází řada na otestování nalezených parametrů. Natrénujeme si tedy neuronovou síť na všech trénovacích i validačních datech, a poté klasifikujeme obrázky z testovací množiny.

Tím získáme výsledek, kde počet pravdivě pozitivních výsledku je 105, počet falešně pozitivních 1, počet falešně negativních 11, doba trénování je 1 hodina a 14 minut a doba hledání vajíček v jednom obrázku je 8,980 sekund. Z toho vypočítáme F-skóre, které se rovná 0,946.

Tento výsledek až zas tak dobrý není. Je možné, že to je způsobené malým množstvím trénovacích dat. Možná kdybychom na trénování použily desetitisíce snímků a ne tři sta, tak by výsledky byly daleko lepší. Falešně pozitivně byla klasifikována zase část opeření slepice a falešně negativní byla špinavá vajíčka.

7.7 Samotný klasifikační program

Vzhledem k výsledkům jednotlivých metod (viz tabulka 7.15) a tomu, že scéna je statická, a tudíž nepotřebujeme rychlé hledání vajíček, tak si pro samotný klasifikační program zvolíme jako klasifikátor logistickou regresi.

Metoda	Doba trén.	Doba hledání	TP	FP	FN	F-skóre
Matic. por.	0 s	1,157 s	103	1	11	0,945
Prsten. por	0 s	4,970 s	103	5	15	0,912
Log. reg.	1 h 46 m	11,906 s	116	1	2	0,987
Neur. síť	1 h 14 m	8,980 s	105	1	11	0,946

Tabulka 7.15: Výsledky klasifikačních metod

Klasifikační program poběží na jakémkoliv počítači, který má přístup k serveru, kde jsou uložená data.

Princip klasifikačního programu spočívá v tom, že bude průběžně kontrolovat, jestli na serveru není nějaký obrázek určený ke klasifikaci. V případě, že takový obrázek existuje, program si ho stáhne, najde v něm vajíčka a uloží pozice vajíček a výsledný obrázek s označenými vajíčky zpátky na server. Následně si uživatel může prohlédnout výsledky na webových stránkách.

Zde můžete vidět pseudokód takového programu:

1. opakuj
 2. pokud na serveru je nezpracovaný obrázek
 3. stáhni obrázek
 4. najdi vajíčka na obrázku
 5. nahraj výsledek zpátky na server

Závěr

Během této bakalářské práce jsme vytvořili poměrně úspěšnou aplikaci, která rozpoznává vajíčka s F-skóre 0,987. Tato aplikace se skládá ze tří nezávislých programů. První program se stará o snímání obrazových dat z místa, kam slepice snáší vajíčka, následně tyto snímky odesílá na server. Druhý program hledá na snímcích vajíčka, kde jednotlivé pixely jsou klasifikovány pomocí logistické regrese. Poslední částí aplikace je webové uživatelské prostředí, které zpřístupňuje nafocené snímky a pozice nalezených vajíček.

V 1. kapitole jsme se podívali na možné reprezentace obrázku, které bychom mohli použít jako příznaky pro klasifikaci. Uvedli jsme si základní maticovou reprezentaci a také rotačně nezávislou prstencovou projekci, která umožňuje reprezentovat dvě rotace téhož obrázku stejnými příznaky.

V 2. kapitole jsme si rozebrali použitelné předzpracování obrázku, tedy možné způsoby jak z barevného tříspektrálního snímku dostat jednospektrální snímek pro následnou klasifikaci. Zmínili jsme použití jednotlivých barevných složek, šedotónové reprezentace, ale také hran detekovaných různými hranovými detektory. Z těchto detektorů jsme použili horní propust a Kirschův hranový detektor.

V kapitole 3 jsme se podívali na jednotlivé klasifikační metody, které bychom mohli použít. Nejjednodušší z nich bylo porovnání se vzorem, u kterého jsme si definovali různé chybové funkce, podle kterých se počítala shodnost s daným vzorem. Dále jsme si ukázali složitější metody, jako je logistická regrese a neuronové sítě. U obou těchto metod jsme se zmínili o jejich trénování a následné klasifikaci dat.

Ve 4. kapitole jsme se krátce podívali na algoritmus, který používáme pro získávání samotných pozic vajíček z oklasifikované matice. Použili jsme k tomu mírně upravený algoritmus na hledání komponent souvislosti v grafu.

V kapitole 5 jsme si rozebrali snímání obrazových dat. Podívali jsme se jak na hardware snímacího zařízení, tak na jeho software. K snímání jsme používali minipočítač „Raspberry Pi Zero“ s připojenou kamerou.

V předposlední kapitole 6 jsme se podívali na webový server, který zprostředkovává komunikaci mezi jednotlivými programy. Rozebrali jsme strukturu jednotlivých databázových tabulek a webových stránek, které zpřístupňují výsledky uživatelům.

V poslední kapitole 7 jsme otestovali jednotlivé metody na hledání vajíček. Nejprve jsme si připravili trénovací a testovací data. Poté jsme ladili jednotlivé klasifikační parametry. A nakonec jsme vybrali nejúspěšnější metodu a vytvořili samotný klasifikační program. Nejlepších výsledků dosáhla logistická regrese, která jako příznaky použila prstencové projekce všech barevných spekter, horní propusti a hran detekovaných Kirscheho detektorem. Tato metoda detekovala všechna vajíčka na jednom snímku v průměru za necelých 12s, což není málo, ale vzhledem k tomu, že focená scéna je poměrně dost statická, tak tato rychlost hledání je podle mého názoru dostačující.

Seznam použité literatury

- BISHOP, C. (2006). *Pattern Recognition and Machine Learning*. Springer. URL <https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/>.
- GHOSAL, S. K., CHATTERJEE, A. a SARKAR, R. (2021). Image steganography based on kirsch edge detection. *Multimedia Systems*, **27**(1), 73–87.
- KANAN, C. a COTTRELL, G. W. (2012). Color-to-grayscale: does the method matter in image recognition? *PloS one*, **7**(1), e29740.
- OPENCV (2022). Object detection in image processing. URL https://docs.opencv.org/4.x/df/dfb/group__imgproc__object.html#gga3a7850640f1fe1f58fe91a2d7583695dac5babb7dfda59544e3e31ea928f8cb16.
- TANG, Y. Y., CHENG, H.-D. a SUEN, C. Y. (1991). Transformation-ring-projection (trp) algorithm and its vlsi implementation. *International Journal of Pattern Recognition and Artificial Intelligence*, **5**(01n02), 25–56.
- TSAI, D.-M. a CHIANG, C.-H. (2002). Rotation-invariant pattern matching using wavelet decomposition. *Pattern Recognition Letters*, **23**(1), 191–201. ISSN 0167-8655. doi: [https://doi.org/10.1016/S0167-8655\(01\)00099-X](https://doi.org/10.1016/S0167-8655(01)00099-X). URL <https://www.sciencedirect.com/science/article/pii/S016786550100099X>.
- WIKIPEDIE (2022). Čípek (oko) — wikipedie: Otevřená encyklopedie. URL [https://cs.wikipedia.org/w/index.php?title=%C4%8C%C3%ADpek_\(oko\)&oldid=21612037](https://cs.wikipedia.org/w/index.php?title=%C4%8C%C3%ADpek_(oko)&oldid=21612037). [Online; navštíveno 17. 11. 2022].
- YUEN, P. C., FENG, G.-C. a TANG, Y. Y. (1998). Printed chinese character similarity measurement using ring projection and distance transform. In *Advances In Oriental Document Analysis And Recognition Techniques*, pages 209–221. World Scientific.

Seznam obrázků

1.1	Příklad rastrového obrázku (vlajka).	4
1.2	Vejde otočené 1.	5
1.3	Vejde otočené 2.	5
1.4	Projekce vejce 1.	5
1.5	Projekce vejce 2.	5
2.1	Barevný obrázek.	6
2.2	Červené spektrum.	6
2.3	Zelené spektrum.	6
2.4	Modré spektrum.	6
2.5	2.1 - 2.4: Ukázka barevných spekter obrázku.	6
2.6	Barevný.	7
2.7	Šedotón (2.1).	7
2.8	Šedotón (2.2).	7
2.9	2.6 - 2.8: Porovnání transformací na šedotón.	7
2.10	Vstupní obrázek.	8
2.11	Furierova transformace.	8
2.12	Oříznutá furierova transformace.	9
2.13	Horní propust obrázku.	9
2.14	Vstupní obrázek.	10
2.15	Detekované hrany.	10
3.1	Vstupní obrázek.	12
3.2	Vzor.	12
3.3	Kvadratická chyba.	12
3.4	Vzájemná korelace.	12
3.5	Korelační koeficienty.	12
3.6	Trénovací obrázek.	14
3.7	Vstupní obrázek.	14
3.8	Predikce vajíček.	14
3.9	Trénovací obrázek.	16
3.10	Vstupní obrázek.	16
3.11	Predikce vajíček.	16
4.1	Matice klasifikovaných pixelů.	17
4.2	Klasifikované shluky.	17
4.3	Průměry shluků.	17
5.1	Schéma zapojení snímacího zařízení.	18
5.2	Náhled snímacího zařízení.	18
6.1	Hlavní stránka webových stránek.	22
6.2	Webové stránky - výstup kamery.	23
6.3	Webové stránky - snímky kamery.	23
6.4	Webové stránky - nastavení kamery.	23
7.1	Příkladový obrázek z kamery.	24

7.2	Příklad 1 vajíčka typu 1.	25
7.3	Příklad 2 vajíčka typu 1.	25
7.4	Příklad 3 vajíčka typu 1.	25
7.5	Příklad 1 vajíčka typu 2.	25
7.6	Příklad 2 vajíčka typu 2.	25
7.7	Příklad 3 vajíčka typu 2.	25
7.8	Falešně pozitivní výsledek.	32
7.9	Falešně negativní výsledek.	32
7.10	Falešně negativní výsledek.	32
A.1	Schéma celé aplikace.	43
A.2	Schéma snímače obrázků.	44
A.3	Schéma hledače vajíček.	45
A.4	Schéma webového rozhraní.	47
A.5	Schéma hardwaru snímače.	50
A.6	Pi Imager - Úvodní okno.	51
A.7	Pi Imager - Výběr OS.	51
A.8	Pi Imager - Nastavení OS.	52
A.9	Instalace kamery - příkaz.	53
A.10	Instalace kamery - Úvodní okno.	53
A.11	Instalace kamery - Nastavení rozhraní.	54
A.12	Instalace kamery - Potvrzení.	54
A.13	Instalace kamery - Ověření.	54
A.14	Instalace kamery - Ukončení.	55
A.15	Instalace kamery - Restartování.	55
A.16	Instalace snímače - Příkaz scp.	56
A.17	Instalace snímače - Knihovny.	56
A.18	Instalace snímače - rc.local.	56
A.19	Ovládání snímače - Nastavení.	57
A.20	Instalace C++ modulů - Instalace prostředí.	58
A.21	Instalace C++ modulů - Instalace modulů.	58
A.22	Instalace hledače - Instalace prostředí.	59
A.23	Web - Uvítací obrazovka.	61
A.24	Web - Seznam kamer.	62
A.25	Web - Nalezená vajíčka.	63
A.26	Web - Snímky kamery.	64
A.27	Web - Snímek kamery.	65
A.28	Web - Nastavení kamery.	66

Seznam tabulek

6.1	Struktura tabulky nastavení kamer	20
6.2	Struktura tabulky dat kamer	21
7.1	Výsledky výběru vzorů	26
7.2	Výsledky výběru chybové funkce	27
7.3	Výsledky výběru reprezentace obrázku	27
7.4	Výsledky výběru vzorů	28
7.5	Výsledky výběru chybové funkce	29
7.6	Výsledky výběru reprezentace obrázku	29
7.7	Výsledky výběru reprezentace obrázku	30
7.8	Výsledky výběru ohodnocení trénovacích dat	31
7.9	Výsledky výběru kombinací reprezentací	32
7.10	Výsledky výběru reprezentace obrázku	33
7.11	Výsledky výběru pozitivního poloměru	33
7.12	Výsledky výběru neutrálního poloměru	34
7.13	Výsledky výběru kombinací reprezentací	34
7.14	Výsledky výběru skrytých vrstev	35
7.15	Výsledky klasifikačních metod	36

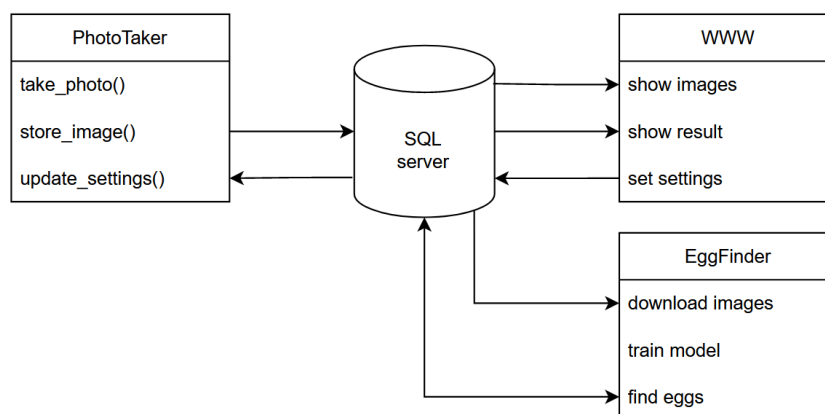
A. Přílohy

A.1 Vývojová dokumentace

V této kapitole si popíšeme strukturu celé aplikace. Nejprve se podíváme na komunikaci mezi jednotlivými podprogramy a poté si tyto podprogramy popíšeme podrobněji.

Celá aplikace se skládá ze tří programů. První program pomocí Raspberry Pi snímá obrázky, druhý v obrázcích hledá vajíčka a třetí je webová aplikace, která zprostředkovává komunikaci s uživatelem. Všechny tyto programy komunikují přes SQL server.

Schéma těchto programů můžeme vidět na obrázku A.1:

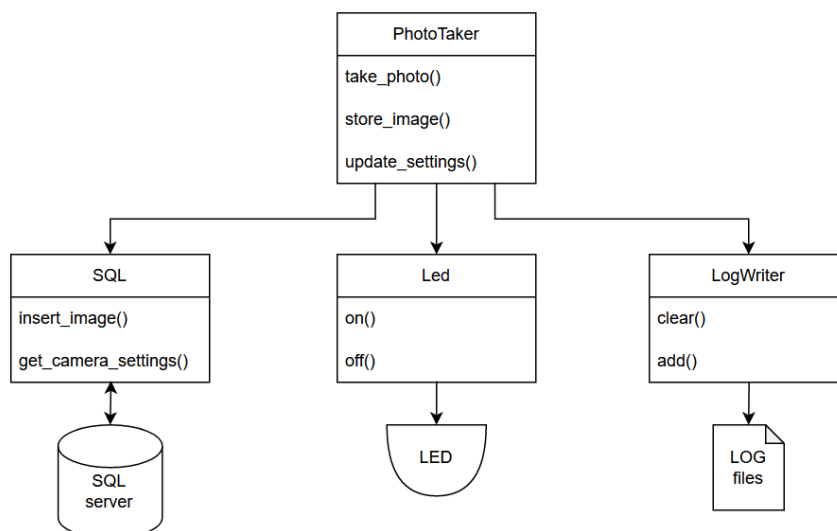


Obrázek A.1: Schéma celé aplikace.

Program „PhotoTaker“ reprezentuje Raspberry Pi, umí snímat obrázky, ukládat je na server a aktualizovat ze serveru svoje nastavení. Program „EggFinder“ se stará o hledání vajíček, je schopen stáhnout všechny snímky ze serveru, natrénovat klasifikační model a hledat vajíčka na snímcích uložených na serveru. Webová aplikace „WWW“ obsahuje stránky, které umí zobrazit všechny obrázky na serveru, zobrazit nalezená vajíčka nebo změnit nastavení kamer.

A.1.1 Snímač obrázků

Obrázky snímá program napsaný v Pythonu běžící na zařízení „Raspberry Pi Zero“. Tento program používá knihovny „RPi.GPIO“ (přístup k pinům) a „mysql.connector“ (komunikace s SQL serverem). Schéma snímače můžeme vidět na obrázku A.2:



Obrázek A.2: Schéma snímače obrázků.

Hlavní třída programu je „PhotoTaker“, která používá pomocné třídy „SQL“, „Led“ a „LogWriter“.

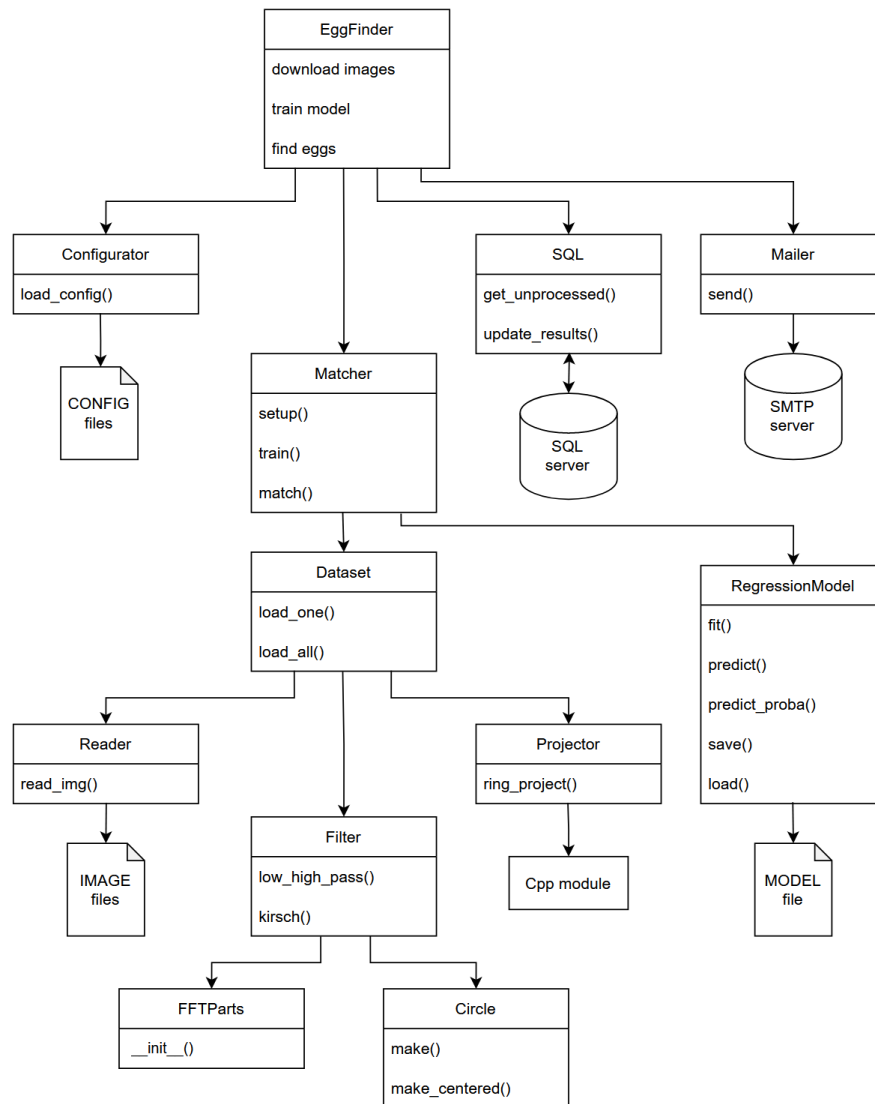
Třída „SQL“ slouží ke komunikaci s SQL databází. Umí se dotazovat na obecné dotazy, ale také na specifické dotazy jako například vložení obrázku na server nebo získání nastavení kamery. Přístupové údaje k databázi jsou předávány v konstruktoru.

O obsluhu LED žárovky se stará třída „Led“. Je to velice jednoduchá třída, která se připojí k danému pinu na Raspberry a pomocí metod „on“ a „off“ je schopná žárovku zapínat a vypínat.

Poslední třída „LogWriter“ zapisuje logovací hlášky do příslušných souborů. Disponuje metodami „clear“, která vymaže celý logovací soubor, a metodou „add“, která připiše logovací hlášku na konec souboru.

A.1.2 Hledač vajíček

Hledání vajíček běží v Pythonu na jakémkoliv zařízení. V případě hodně připojených kamer je možné mít připojeno také více hledačů, aby hledání netrvalo příliš dlouho. Hledač vajíček používá knihovny „mysql.connector“ (komunikace s SQL serverem), „scikit-learn“ (implementace klasifikátorů), „opencv“ (práce s obrázky), „smtplib“ a „email“ (odesílání mailů), „pickle“ a „lzma“ (ukládání modelů do souboru). Schéma hledače vajíček můžeme vidět na obrázku A.3:



Obrázek A.3: Schéma hledače vajíček.

Tento program můžeme spustit ve třech režimech. První stáhne všechny obrázky ze serveru, druhý natrénuje klasifikační model a třetí hledá vajíčka na obrázcích ze serveru. Všechny tyto režimy používají pomocné třídy „Configurator“, „SQL“, „Mailer“ a „Matcher“.

Třída „Configurator“ slouží k načítání konfiguračních souborů. Obsahuje jen jednu metodu „load_config“, která najde konfigurační soubor, přečte ho a vrátí slovník reprezentující načtenou konfiguraci.

Komunikaci s SQL databází obsluhuje třída „SQL“. Umí se dotazovat na

obecné dotazy, ale také na specifické dotazy jako například získání dosud nezpracovaných obrázků nebo vložení nalezených vajíček do databáze. Přístupové údaje k databázi jsou předávány v konstruktoru.

Třída „Mailer“ souží k odesílání mailů na SMTP server v případě, že bylo nalezené nějaké vajíčko. Je to jednoduchá třída s jednou metodou „send“, která odešle mail z předem dané mailové adresy na adresu příjemce, která je zapsána v konfiguračním souboru.

K samotnému hledání vajíček slouží třída „Matcher“. Má metody „setup“ na nastavení parametrů hledání, „train“ na natrénování klasifikačního modelu a metodu „match“ na nalezení vajíček na snímku. Tato třída používá pomocné třídy „Dataset“ a „RegressionModel“.

Třída „RegressionModel“ reprezentuje nějaký regresní model. Podporuje vícevrstevnatou perceptronovou síť a logistickou regresi. Má metody „fit“ na natrénování modelu, „predict“ na predikování výsledků, „predict_proba“ na predikování pravděpodobností výsledků, „save“ na uložení modelu do souboru a metodu „load“ na načtení uloženého modelu ze souboru.

Třída „Dataset“ reprezentuje datovou sadu, která se skládá z příznaků a cílů nějakých dat. Obsahuje metody „load_one“, která načte data z jednoho obrázku a metodu „load_all“, která načte data ze všech obrázků ve složce. Tato třída používá pomocné třídy „Reader“, „Projector“ a „Filter“.

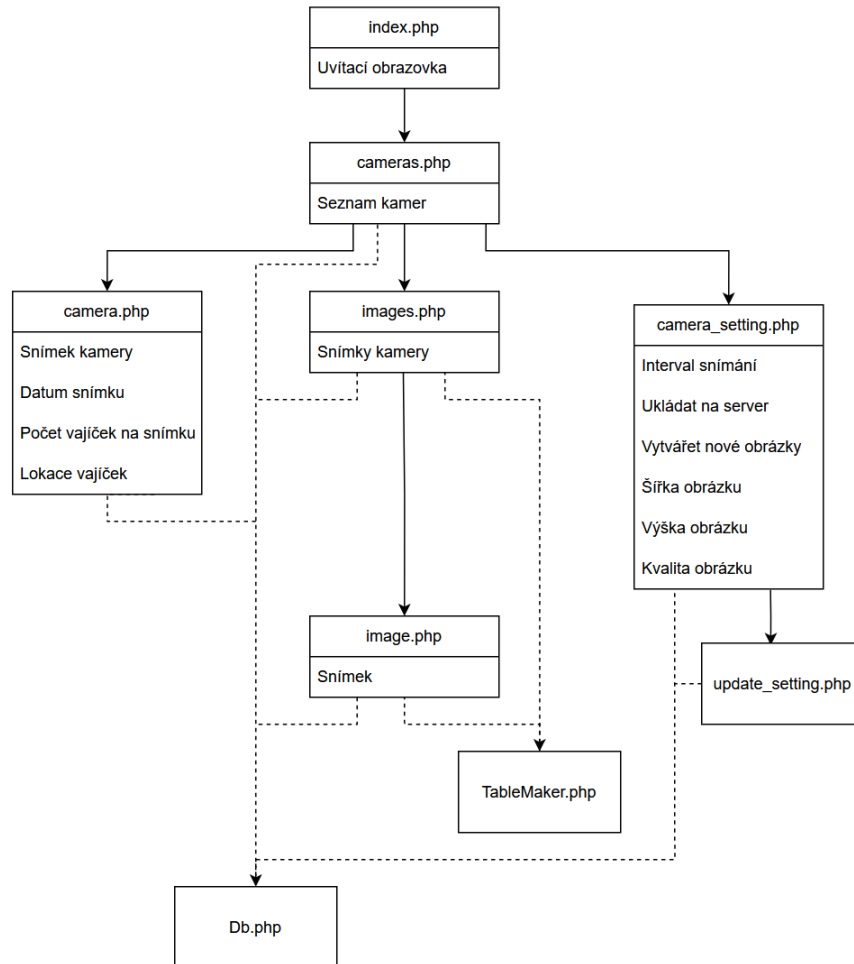
Na načtení obrazových dat slouží třída „Reader“. Obsahuje jedinou metodu „read_img“, která vrací matici daných obrazových dat. Při načítání obrázku je možné převést obrázek do šedotónové reprezentace.

Třída „Projector“ transformuje maticovou reprezentaci obrázku na prstencovou. Její jediná metoda „ring_project“ vrací prstencovou reprezentaci vstupního obrázku, která se počítá za pomoci vlastního Cpp modulu.

Na různá předzpracování obrazu slouží třída „Filter“. Tato třída má metody „low_high_pass“, která počítá horní nebo dolní propust obrázku, a metodu „kirsch“, která vrací hrany detekované Kirscheho hranových detektorem. K počítání daných předzpracování jsou používány třídy „FFTParts“, která počítá Fourierovu transformaci obrázku, a třída „Circle“, která vrací kruhové binární masky.

A.1.3 Webové rozhraní

Jako uživatelské rozhraní slouží sada webových stránek napsaných v jazyce PHP. Na těchto stránkách je možné měnit nastavení kamer, prohlížet si nasnímané obrázky a kontrolovat nalezená vajíčka. Schéma jednotlivé stránky můžeme vidět na obrázku A.3:



Obrázek A.4: Schéma webového rozhraní.

V tomto schématu můžeme vidět jednotlivé stránky a jejich obsah. Šipky s plnou čarou označují odkazy mezi stránkami a šipky s přerušovanou čarou znamenají použití PHP modulu.

Třída „Db“ slouží ke komunikaci s SQL databází a třída „TableMaker“ vytváří tabulky daného formátu.

A.1.4 C++ moduly

Program, který hledá vajíčka, používá na hledání dva C++ moduly. Hledání pomocí C++ modulů je zhruba tisíckrát rychlejší než při použití samotného Pythonu.

První modul počítá prstencovou reprezentaci obrázku z maticové reprezentace. Má funkci „project“, která vrací matici prstencových reprezentací daného jednospektrálního obrázku. Vstupem to této funkce je daný obrázek a poloměry v kterých se má reprezentace počítat.

Druhý modul hledá středy komponent v binárním obrázku. Obsahuje funkci „find_components“, která vrací matici čísel, kde číslo v dané pozici označuje velikost komponenty se středem v této pozici (0 - žádný střed komponenty). Tento modul k výpočtu používá algoritmus popsáný v kapitole 4.

A.1.5 SQL databáze

Jako SQL databáze je použita MySQL databáze běžící na aplikaci „XAMPP“. Struktura jednotlivých tabulek je popsána v kapitole 6.1.

A.2 Uživatelská dokumentace

A.2.1 SQL Databáze

Zde si popíšeme jak vytvořit SQL databázi, která zprostředkovává komunikaci mezi jednotlivými programy v aplikaci.

Instalace

1. Nainstalujte SQL databázi, kterou chcete použít (například mySQL databázi od aplikace „XAMPP“).
2. Spusťte SQL dotaz, který vytvoří potřebnou databázi. Text dotazu naleznete v souboru „WWW/install/create_database.txt“.
3. Přidejte potřebná privilegia pro zařízení, které k databázi budou přistupovat:
 - Text dotazu naleznete v souboru „WWW/install/add_user.txt“.
 - Přidejte privilegia pro webové rozhraní a pro všechny kamery a hledače vajíček.
 - V dotazu nahraďte všechny výskyty „{USER_NAME}“ za uživatelská jména (například „'photo_taker'“).
 - V dotazu nahraďte všechny výskyty „{USER_IP}“ za adresy uživatele (například „'192.168.1.10'“).

A.2.2 Snímač obrázků

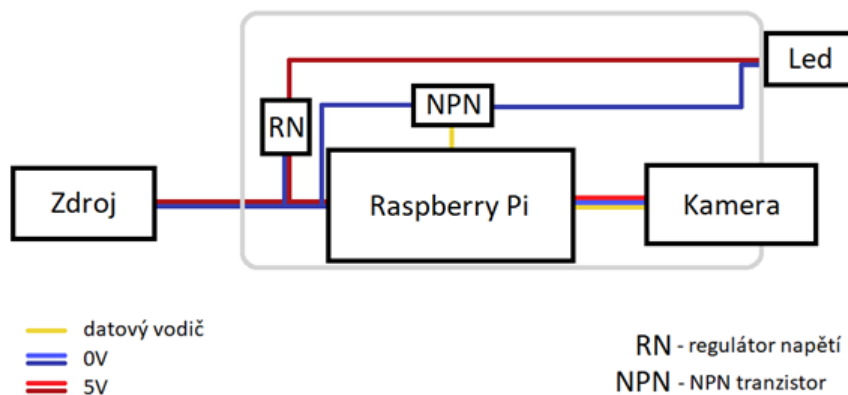
Zde si popíšeme hardware, instalaci a používání snímače obrázků.

Hardware

Snímač obrázků běží na minipočítači „Raspberry Pi“. K němu je připojen zdroj napětí, kamera a LED žárovka. Níže je seznam potřebných komponent:

1. CPU - Raspberry Pi Zero 2 W
2. Kamera - Raspberry Pi kamera V2
3. SD karta - 64 GB (lze použít i 32 GB)
4. LED na osvětlení fotografované plochy - LED 1 W, bílá, 140 lm, 120°
5. Chladič LED - Šestiúhelníkový chladič GT- P03W54101140
6. Regulátor napětí - Lineární stabilizátor napětí, THT, 3,3 V, 0,5 A, TO220 LF33CV
7. Tranzistor - Bipolární tranzistor, NPN, THT, 80 V, 1,5 A, 8 W, TO126 BD139-16

Schéma zapojení těchto komponent můžeme vidět na obrázku A.5:



Obrázek A.5: Schéma hardwaru snímače.

Instalace OS

Nejdříve je nutné si nainstalovat „Pi Imager“, který na SD kartu zapíše operační systém Raspbian:

1. Stáhněte si Pi Imager na adrese: „<https://www.raspberrypi.com/software>“.
2. Nainstalujte Imager.
3. Spusťte Imager, mělo by se objevit následující okno (viz obrázek A.6):



Obrázek A.6: Pi Imager - Úvodní okno.

4. Vyberte operační systém „Raspberry Pi OS (32-BIT)“ (viz obrázek A.7):



Obrázek A.7: Pi Imager - Výběr OS.

5. Vyberte SD kartu, kam chcete OS nainstalovat.
6. Klikněte na ikonku nastavení a nastavte potřebné parametry (viz obrázek A.8):
 - Povolte SSH.
 - Nastavte uživatelské jméno a heslo.
 - Nastavte jméno a heslo WiFi, ke které se má Raspberry připojit.

The screenshot shows the 'Advanced options' window in Pi Imager. At the top, there is a title bar with 'Advanced options' and a close button 'X'. Below the title bar, there is a section 'Image customization options' with a dropdown menu set to 'to always use'. The main content area contains several configuration options:

- Set hostname: `raspberrypi` .local
- Enable SSH
 - Use password authentication
 - Allow public-key authentication only
 - Set authorized_keys for 'pi': _____
- Set username and password
 - Username: `pi`
 - Password: `••••••••`
- Configure wireless LAN
 - SSID: `jmeno_wifi`
 - Hidden SSID
 - Password: `heslo_wifi`
 - Show password

At the bottom center of the window is a red 'SAVE' button.

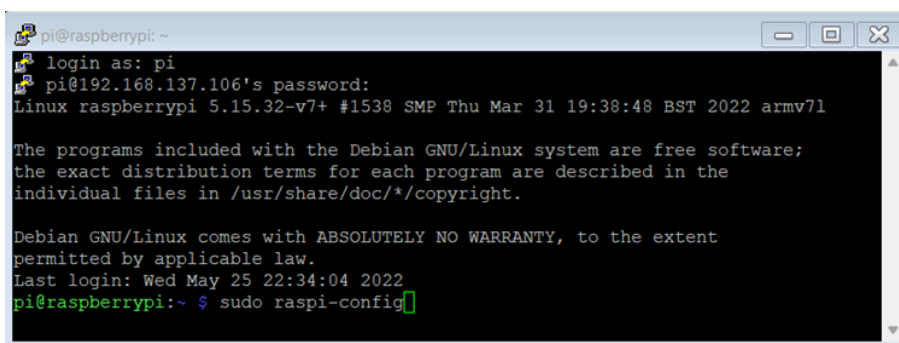
Obrázek A.8: Pi Imager - Nastavení OS.

7. Nastavení uložte a klikněte na tlačítko „WRITE“ pro zapsání operačního systému.

Instalace kamery

Po nainstalování operačního systému na SD kartu, můžeme nastavit kameru:

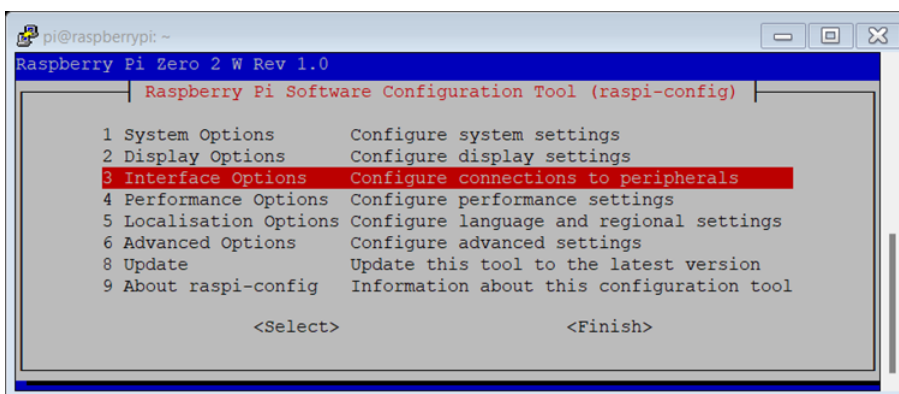
1. Vložte SD kartu do Raspberry a zapojte napájení Raspberry.
 - Raspberry by se mělo naboootovat a připojit k nastavené WiFi.
2. Připojte se počítačem k Raspberry přes SSH.
 - Můžete se připojit pomocí programů: PuTTY, Visual Studio Code, ...
 - Počítač musí být připojen ke stejné WiFi jako Raspberry.
 - K připojení použijte IP adresu Raspberry, kterou naleznete v seznamu připojených zařízení v nastavení WiFi.
 - Před samotným propojením budete vyzváni k napsání uživatelského jména a hesla, které jste si zvolili v nastavení OS v Imageru.
3. Spusťte nastavení Raspberry tak, že do konzole napíšete příkaz „sudo raspi-config“ (viz obrázek A.9):



```
pi@raspberrypi: ~  
login as: pi  
pi@192.168.137.106's password:  
Linux raspberrypi 5.15.32-v7+ #1538 SMP Thu Mar 31 19:38:48 BST 2022 armv7l  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Wed May 25 22:34:04 2022  
pi@raspberrypi:~ $ sudo raspi-config
```

Obrázek A.9: Instalace kamery - příkaz.

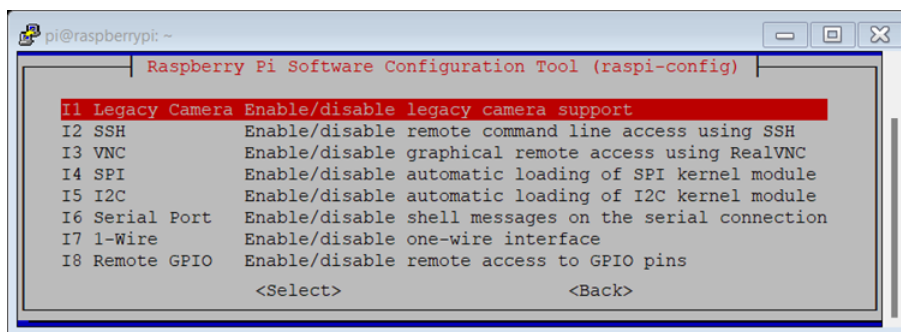
4. Objeví se interaktivní okno, v něm vyberte „3 Interface Options“ a zmáčkněte „Enter“ (viz obrázek A.10):



```
Raspberry Pi Zero 2 W Rev 1.0  
Raspberry Pi Software Configuration Tool (raspi-config)  
  
1 System Options          Configure system settings  
2 Display Options        Configure display settings  
3 Interface Options       Configure connections to peripherals  
4 Performance Options     Configure performance settings  
5 Localisation Options   Configure language and regional settings  
6 Advanced Options       Configure advanced settings  
8 Update                  Update this tool to the latest version  
9 About raspi-config     Information about this configuration tool  
  
<Select>                  <Finish>
```

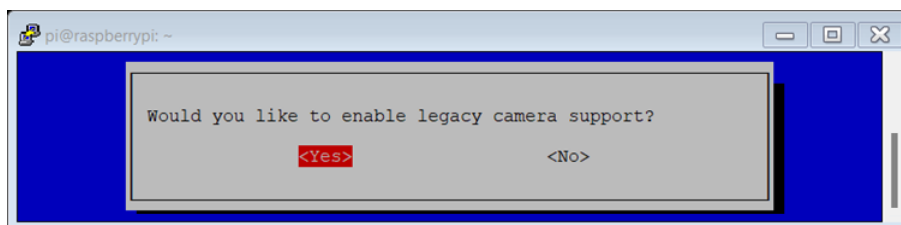
Obrázek A.10: Instalace kamery - Úvodní okno.

5. Zvolte nastavení kamery, tedy „I1 Legacy Camera“ (viz obrázek A.11):



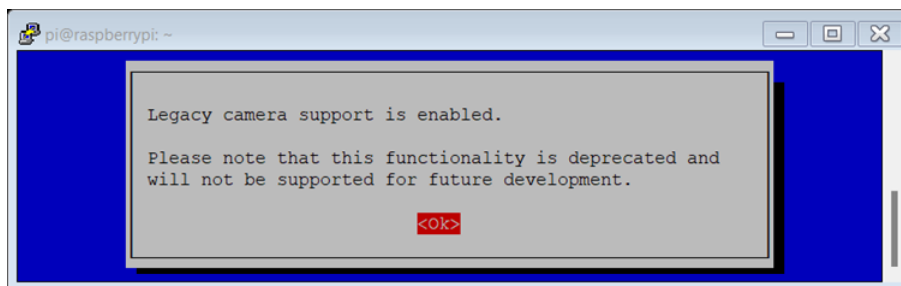
Obrázek A.11: Instalace kamery - Nastavení rozhraní.

6. Potvrďte povolení kamery zvolením „Yes“ (viz obrázek A.12):



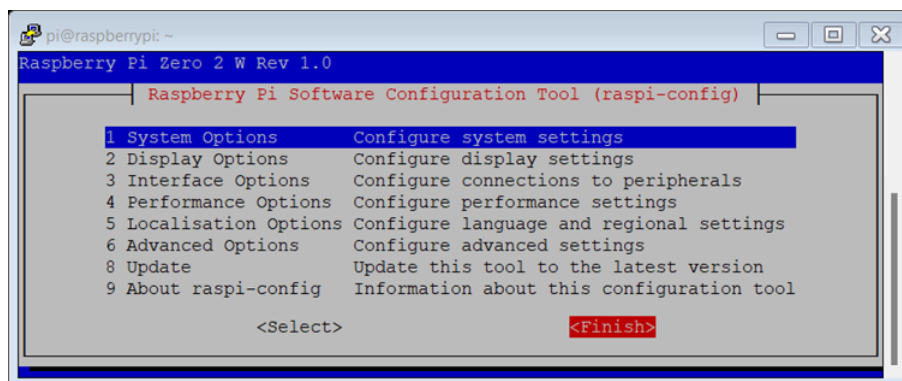
Obrázek A.12: Instalace kamery - Potvrzení.

7. Pokud se podařilo kameru nastavit, tak se zobrazí okno s hláškou, že kamera je povolená, zvolte „Ok“ (viz obrázek A.13):



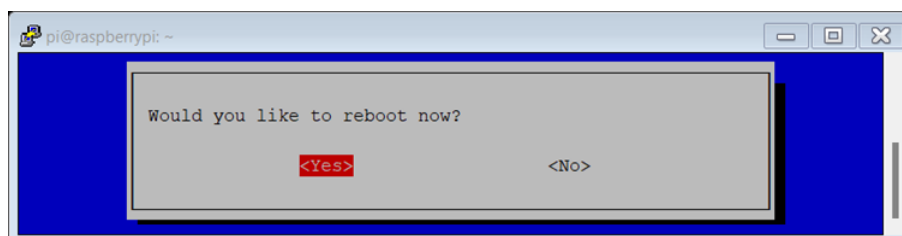
Obrázek A.13: Instalace kamery - Ověření.

8. Zvolte „Finish“ pro ukončení nastavení (viz obrázek A.14):



Obrázek A.14: Instalace kamery - Ukončení.

9. Potvrďte restartování zařízení kliknutím na „Yes“ (viz obrázek A.15):



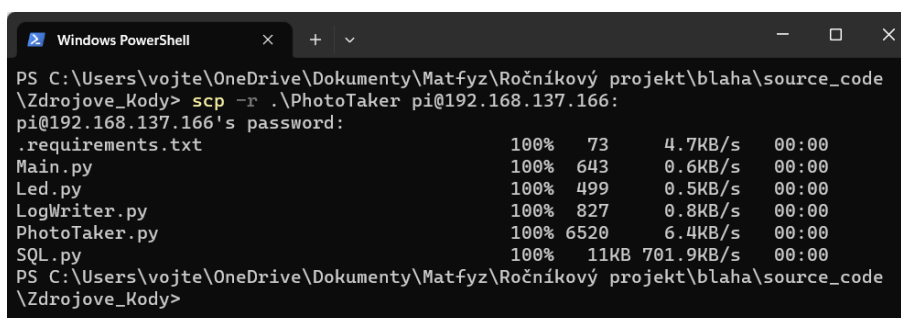
Obrázek A.15: Instalace kamery - Restartování.

10. Znovu se k Raspberry připojte pomocí SSH.

Instalace snímače

Nejprve přesuneme zdrojové kódy snímače do Raspberry a poté nainstalujeme běhové prostředí:

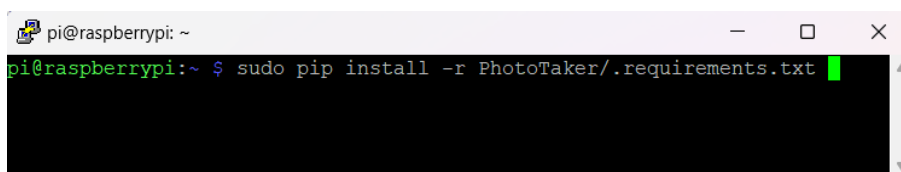
1. Přesuňte složku „PhotoTaker“ se zdrojovými kódy z počítače do Raspberry (viz obrázek A.16):
 - Je možné k tomu použít příkaz „scp“, který spustíte na počítači ve složce se všemi zdrojovými kódy.
 - Příkaz: „scp -r .\PhotoTaker pi@{ip_adresa}:“, kde {ip_adresa} nahradíte IP adresou Raspberry.
 - Příkaz vás vyzve k zadání hesla pro Raspberry.



```
Windows PowerShell
PS C:\Users\vojte\OneDrive\Dokumenty\Matfyz\Ročníkový projekt\blaha\source_code\Zdrojove_Kody> scp -r .\PhotoTaker pi@192.168.137.166:
pi@192.168.137.166's password:
.requirements.txt          100%  73      4.7KB/s   00:00
Main.py                   100% 643      0.6KB/s   00:00
Led.py                    100% 499      0.5KB/s   00:00
LogWriter.py              100% 827      0.8KB/s   00:00
PhotoTaker.py             100% 6520     6.4KB/s   00:00
SQL.py                    100% 11KB    701.9KB/s 00:00
PS C:\Users\vojte\OneDrive\Dokumenty\Matfyz\Ročníkový projekt\blaha\source_code\Zdrojove_Kody>
```

Obrázek A.16: Instalace snímače - Příkaz scp.

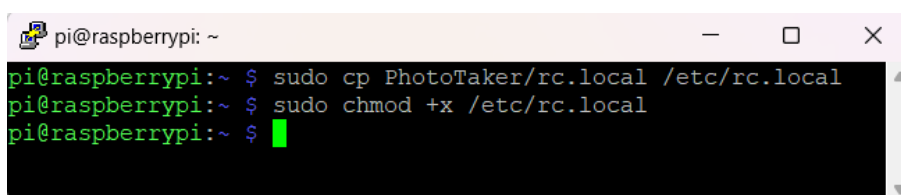
2. Nainstalujte potřebné python knihovny na Raspberry (viz obrázek A.17):
 - Zadejte příkaz: „sudo pip install -r PhotoTaker/.requirements.txt“



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo pip install -r PhotoTaker/.requirements.txt
```

Obrázek A.17: Instalace snímače - Knihovny.

3. Nastavte automatické spouštění snímače (viz obrázek A.18):
 - Zkopírujte skript: „sudo cp PhotoTaker/rc.local /etc/rc.local“
 - Nastavte skript: „sudo chmod +x /etc/rc.local“



```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo cp PhotoTaker/rc.local /etc/rc.local
pi@raspberrypi:~ $ sudo chmod +x /etc/rc.local
pi@raspberrypi:~ $
```

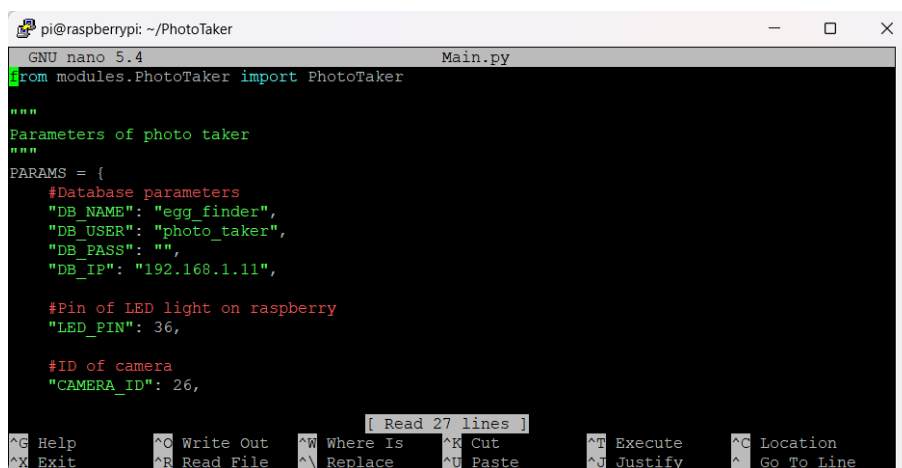
Obrázek A.18: Instalace snímače - rc.local.

4. Restartujte Raspberry příkazem: „sudo reboot“.

Ovládání

1. Nastavení snímače (viz obrázek A.19):

- Snímač lze nastavit upravením souboru „PhotoTaker/Main.py“.
- V tomto souboru nastavte přístupové údaje k databázi (jméno, uživatel, heslo uživatele a IP adresu databáze), pin, ke kterému je připojena LED žárovka, a identifikátor kamery (každý snímač by měl mít unikátní identifikátor).



```
pi@raspberrypi: ~/PhotoTaker
GNU nano 5.4 Main.py
from modules.PhotoTaker import PhotoTaker

"""
Parameters of photo taker
"""
PARAMS = {
    #Database parameters
    "DB_NAME": "egg_finder",
    "DB_USER": "photo_taker",
    "DB_PASS": "",
    "DB_IP": "192.168.1.11",

    #Pin of LED light on raspberry
    "LED_PIN": 36,

    #ID of camera
    "CAMERA_ID": 26,
}

[ Read 27 lines ]
^E Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Obrázek A.19: Ovládání snímače - Nastavení.

2. Spouštění snímače:

- Snímač se spustí automaticky po naběhnutí operačního systému.
- Pro explicitní spuštění zadejte: „sudo python3 PhotoTaker/Main.py“

3. Logovací hlášky:

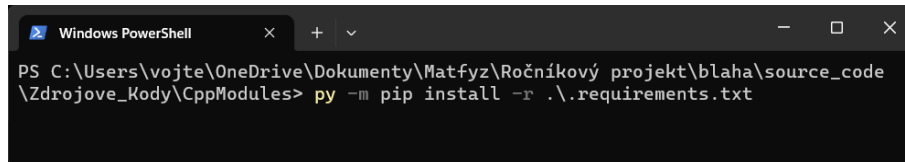
- Logovací hlášky o běhu programu se zapisují do souborů „LOG.txt“ a „LOG_run.txt“, které naleznete ve složce „PhotoTaker/log“.
- Soubor „LOG.txt“ obsahuje pouze nejdůležitější hlášky (změny nastavení, nedostupnost serveru, ...).
- Soubor „LOG_run.txt“ obsahuje všechny běhové hlášky (fotografování snímku, ukládání na server, ...).

A.2.3 C++ moduly

Zde si popíšeme instalaci C++ modulů.

Instalace

1. Nainstalujte prostředí příkazem „`py -m pip install -r .\requirements.txt`“ v adresáři se zdrojovými kódy ve složce „CppModules“ (viz obrázek A.20):

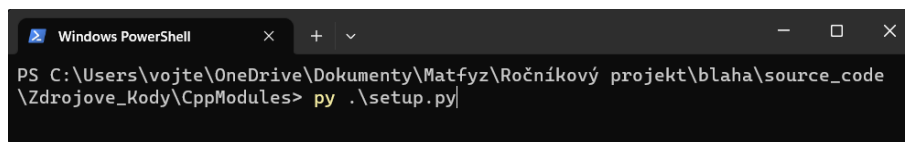


```
Windows PowerShell
PS C:\Users\vojte\OneDrive\Dokumenty\Matfyz\Ročníkový projekt\blaha\source_code\Zdrojove_Kody\CppModules> py -m pip install -r .\requirements.txt
```

Obrázek A.20: Instalace C++ modulů - Instalace prostředí.

2. Nainstalujte samotné C++ moduly:

- Bud příkazem: „`py .\setup.py`“, který nainstaluje moduly automaticky (viz obrázek A.21):



```
Windows PowerShell
PS C:\Users\vojte\OneDrive\Dokumenty\Matfyz\Ročníkový projekt\blaha\source_code\Zdrojove_Kody\CppModules> py .\setup.py
```

Obrázek A.21: Instalace C++ modulů - Instalace modulů.

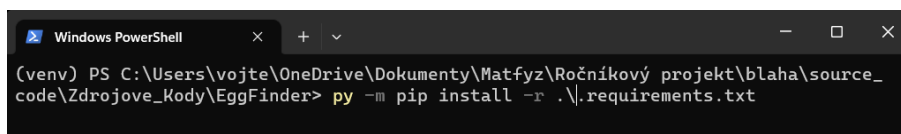
- Nebo sadou příkazů:
 - (a) „`py setups/setup_projector.py bdist_wheel`“
 - (b) „`py setups/setup_componenter.py bdist_wheel`“
 - (c) „`py -m pip install .\dist\matcher-1.0.0-cp310-cp310-win_amd64.whl`“
 - (d) „`py -m pip install .\dist\componenter-1.0.0-cp310-cp310-win_amd64.whl`“

A.2.4 Hledač vajíček

Zde si popíšeme instalaci a použití hledače vajíček.

Instalace

1. Nainstalujte C++ moduly, jak je popsáno v kapitole A.2.3.
2. Nainstalujte prostředí příkazem „`py -m pip install -r .\requirements.txt`“ v adresáři se zdrojovými kódy ve složce „EggFinder“ (viz obrázek A.22):



```
Windows PowerShell
(venv) PS C:\Users\vojte\OneDrive\Dokumenty\Matfyz\Ročníkový projekt\blaha\source_code\Zdrojove_Kody\EggFinder> py -m pip install -r .\requirements.txt
```

Obrázek A.22: Instalace hledače - Instalace prostředí.

Nastavení

- Hledač lze nastavit upravením souboru „EggFinder/config.json“.
- V tomto souboru můžete nastavit následující parametry:
 1. zda je použita prstencová reprezentace nebo maticová („ring“)
 2. množina používaných reprezentací („preprocessing“)
 3. typ klasifikačního modelu („model_type“)
 4. parametry klasifikátoru („C“, „solver“, „max_iter“, „hidden_layers“)
 5. pravděpodobnostní hranice pro pozitivní klasifikaci („threshold“)
 6. parametry prstencové projekce („r_min“, „r_max“, „r_step“)
 7. poloměry vajec („egg_radius“, „nothing_radius“)
 8. parametry komponent („min_dist“, „max_dist“)
 9. maximální velikost trénovacích dat („max_data_size“)
 10. poloměr horní propusti („highpass_radius“)
 11. parametry odesílání mailů („mail“, „mail_subject“)
 12. přístupové údaje k databázi („DB_IP“, „DB_NAME“, „DB_USER“, „DB_PASS“)

Stahování obrázků

Zde si popíšeme jakým způsobem stáhnout všechny obrázky z databáze, které poté využijeme ke trénování.

1. Spustíte stahovač obrázků příkazem „`py .\main_download.py`“.
2. Stažené obrázky najdete ve složce „EggFinder/images/download“.

Označení obrázků

Nyní si popíšeme způsob, jak ručně označit vajíčka na trénovacích snímcích.

1. Přesuňte obrázky k označení do složky „EggFinder/images/train/color“
2. Přesuňte se do složky „EggFinder/images/train“
3. Spusťte označovač příkazem „py .\marker.py“.
4. Označte všechny vajíčka na obrázku a ukončete okno s obrázkem. Toto opakujte dokud neoznačíte vajíčka na všech snímcích. Levým kliknutím myši na střed vajíčka se označí pozitivně klasifikované vajíčko ke trénování, pravým kliknutím označíte neutrálně klasifikované vajíčko (tedy vajíčko, které se ke trénování vůbec nevyužije).
5. Označené obrázky najdete ve složce „EggFinder/images/train/marked“.

Trénování modelu

Zde si popíšeme jak natrénovat klasifikační model.

1. Přesuňte trénovací obrázky do složky „EggFinder/images/train“. Neoznačené obrázky do složky „color“ a označené do složky „marked“.
2. Přesuňte se do hlavní složky hledače „EggFinder“.
3. Spusťte trénování příkazem „py .\main_train.py“.

Spuštění hledače

Nyní si popíšeme jak spustit natrénovaný model, který bude hledat vajíčka ve snímcích uložených na serveru a výsledky bude ukládat zase zpátky na server.

1. Přesuňte se do hlavní složky hledače „EggFinder“.
2. Spusťte hledání příkazem „py .\main_match.py“. To spustí nekonečný hledač vajíček, který ukončíte klávesami „Ctrl + C“.

A.2.5 Webové rozhraní

Nyní si popíšeme jak vytvořit a používat webové rozhraní aplikace.

Instalace

1. Nainstalujte server, na kterém poběží webové rozhraní. Tento server musí podporovat PHP skripty (například Apache server od aplikace „XAMPP“).
2. Vytvořte na serveru nový adresář „egg_finder“.
3. Přesuňte všechny soubory ze složky „WWW/files“ do adresáře serveru „egg_finder“.

Ovládání

Webové rozhraní se skládá z navzájem propojených webových stránek, které si popíšeme v této kapitole.

1. Uvítací obrazovka (viz obrázek A.23)
 - Adresa: „{IP adresa}/egg_finder/“
 - Na této stránce můžete vidět uvítací obrazku s menu.
 - Kliknutím na položku „Kamery“ v menu se dostanete na stránku, kde jsou zobrazeny informace o jednotlivých kamerách.



Obrázek A.23: Web - Uvítací obrazovka.

2. Seznam kamer (viz obrázek A.24)

- Adresa: „{IP adresa}/egg_finder/cameras.php“
- Na této stránce můžete vidět seznam všech kamer, které na server už odeslaly nějaká data.
- Kliknutím na odkaz „Kamera {ID kamery}“ si můžete zobrazit aktuálně nalezená vajíčka na dané kameře.
- Kliknutím na odkaz „zobrazit“ ve sloupci „Snímky“ si můžete zobrazit všechny snímky dané kamery.
- Kliknutím na odkaz „upravit“ ve sloupci „Nastavení“ můžete změnit nastavení dané kamery.



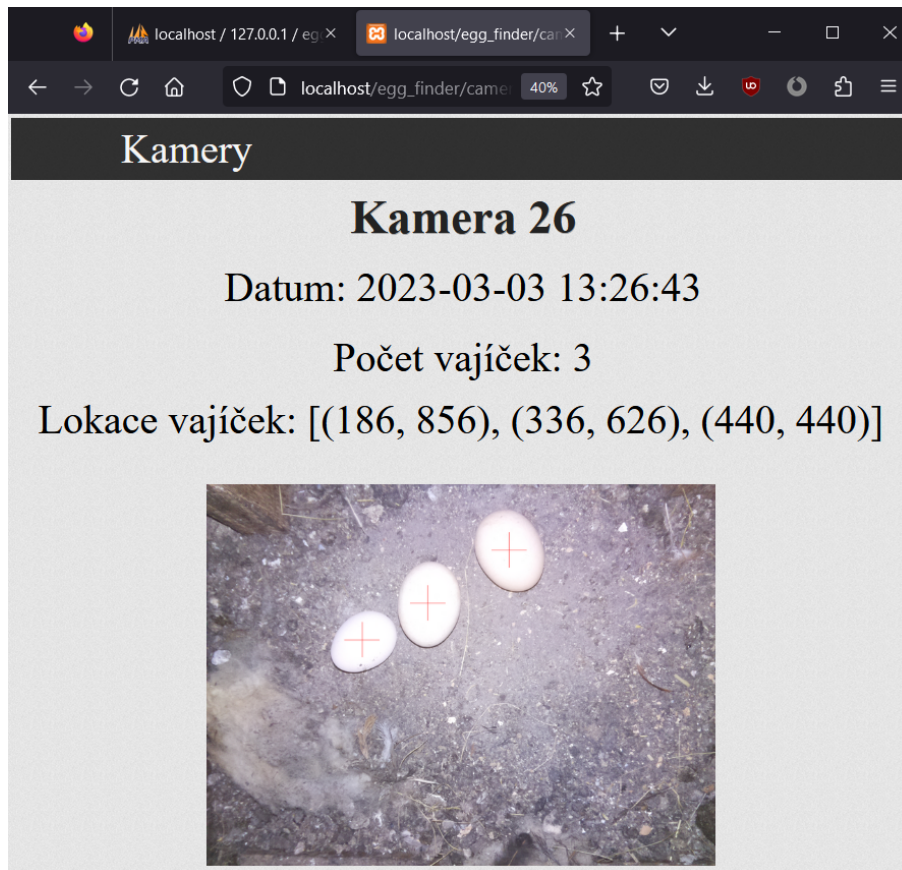
The screenshot shows a web browser window with the address bar displaying 'localhost/egg_finder/cameras.php'. The page content includes a dark header with the word 'Kamery' in white, followed by a light gray section with the title 'Kamery' and the subtitle 'Dostupné kamery:'. Below this is a table with five columns: 'Kamera', 'Poslední snímek', 'Počet vajíček', 'Snímky', and 'Nastavení'. The table lists two cameras: 'Kamera 26' and 'Kamera 27', each with its last image timestamp, egg count, and links for viewing images and settings.

Kamera	Poslední snímek	Počet vajíček	Snímky	Nastavení
Kamera 26	2023-03-03 13:26:43	3	zobrazit	upravit
Kamera 27	2022-11-16 06:12:45	2	zobrazit	upravit

Obrázek A.24: Web - Seznam kamer.

3. Nalezená vajíčka (viz obrázek A.25)

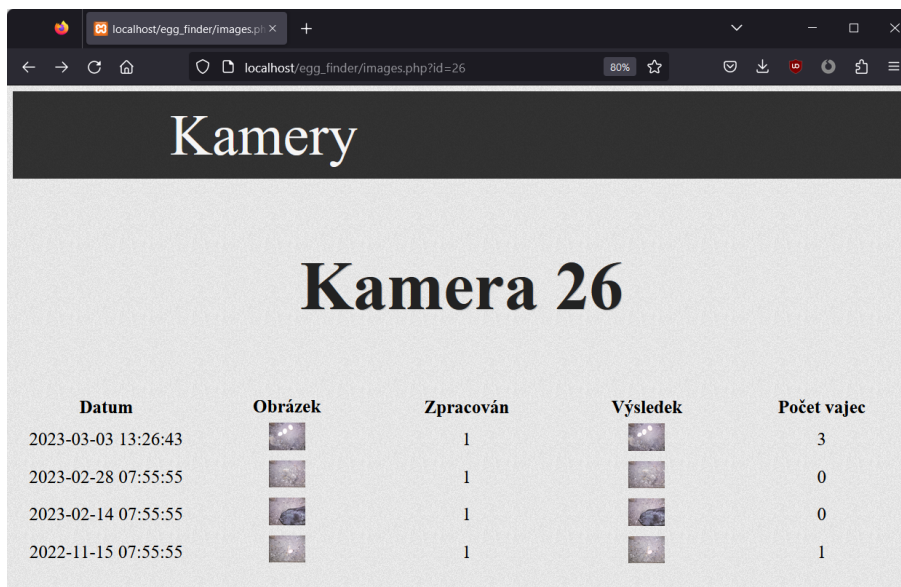
- Adresa: „{IP adresa}/egg_finder/camera.php?id={ID kamery}“
- Na této stránce můžete vidět nalezená vajíčka na posledním snímku dané kamery.
- Stránka obsahuje datum pořízení snímku, počet vajíček na snímku, lokace jednotlivých vajíček na obrázku a samotný obrázek s označenými vajíčky.











Obrázek A.25: Web - Nalezená vajíčka.

4. Snímky kamery (viz obrázek A.26)

- Adresa: „{IP adresa}/egg_finder/images.php?id={ID kamery}“
- Na této stránce si můžete prohlédnout všechny snímky dané kamery.
- Každý snímek obsahuje následující informace: datum pořízení snímku, vyfocený obrázek, zda už byl tento snímek zpracován hledačem vajíček, výsledný obrázek s označenými vajíčky a počet nalezených vajíček na snímku.
- Kliknutím na jakýkoliv obrázek si můžete tento obrázek prohlédnout detailněji nebo si ho stáhnout.

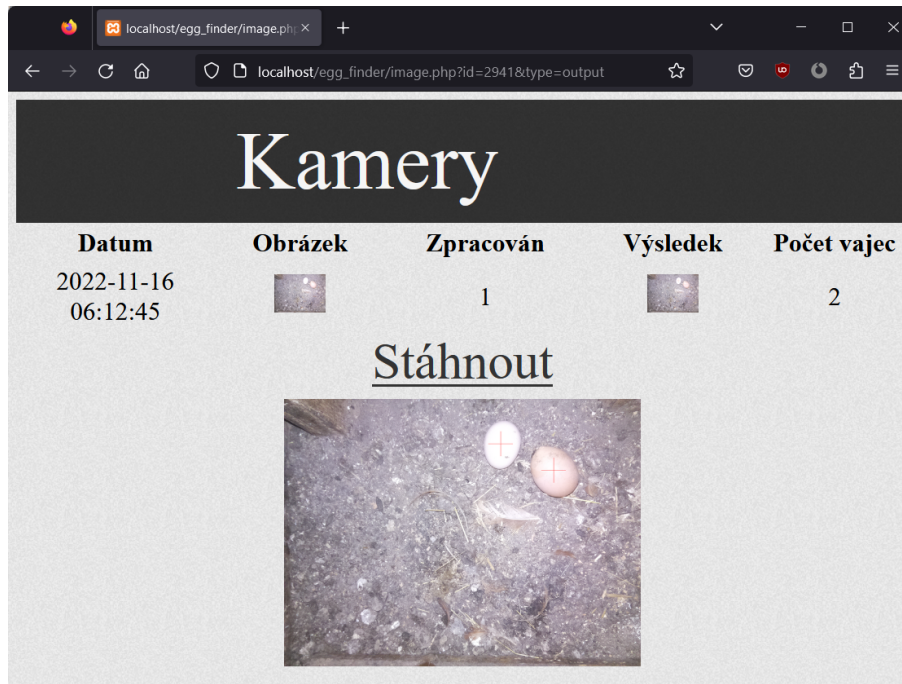


Datum	Obrázek	Zpracován	Výsledek	Počet vajec
2023-03-03 13:26:43		1		3
2023-02-28 07:55:55		1		0
2023-02-14 07:55:55		1		0
2022-11-15 07:55:55		1		1

Obrázek A.26: Web - Snímky kamery.

5. Snímek kamery (viz obrázek A.27)

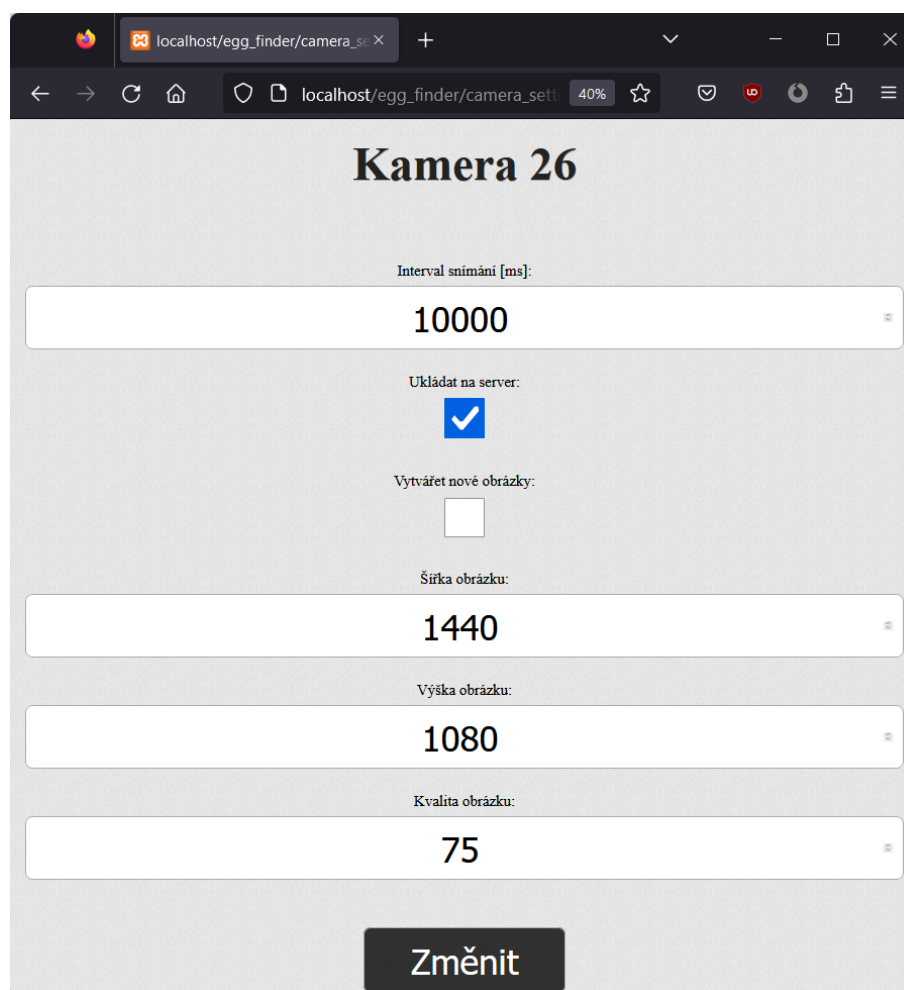
- Adresa: „{IP adresa}/egg_finder/image.php?id={ID obrázku}“
- Na této stránce si můžete prohlédnout konkrétní snímek kamery.
- Kliknutím na odkaz „Stáhnout“ si můžete tento obrázek stáhnout na disk.



Obrázek A.27: Web - Snímek kamery.

6. Nastavení kamery (viz obrázek A.28)

- Adresa:
„{IP adresa}/egg_finder/camera_setting.php?id={ID kamery}“
- Na této stránce je možné nastavit jednotlivé parametry snímání dané kamery:
 - Interval snímání - prodleva mezi snímky v ms
 - Ukládat na server - ukládání na server/lokálně
 - Vytvářet nové obrázky - zda ponechávat/přepisovat minulé snímky
 - Šířka obrázku
 - Výška obrázku
 - Kvalita obrázku - kvalita JPEG obrázku



The screenshot shows a web browser window with the URL `localhost/egg_finder/camera_setting.php?id={ID kamery}`. The page title is "Kamera 26". The settings are as follows:

Parameter	Value
Interval snímání [ms]	10000
Ukládat na server	<input checked="" type="checkbox"/>
Vytvářet nové obrázky	<input type="checkbox"/>
Šířka obrázku	1440
Výška obrázku	1080
Kvalita obrázku	75

At the bottom of the form is a button labeled "Změnit".

Obrázek A.28: Web - Nastavení kamery.

A.3 Testování

Zde si popíšeme jakým způsobem jsme testovali jednotlivé algoritmy na hledání vajíček.

A.3.1 Obrázky

Všechny obrázky, které jsme použili k testování jsou uloženy ve složce „Testovani/images“. Tato složka obsahuje jednotlivé podsložky:

- template - vzorová vajíčka z trénovacích dat
- template_cut - oříznutá vzorová vajíčka z trénovacích dat
- test - obrázky určené na testování
- test_marked - označené obrázky určené na testování
- train_clear - obrázky určené na trénování před validací
- train_clear_marked - označené obrázky určené na trénování před validací
- train_all - obrázky určené na trénování před testováním
- train_all_marked - označené obrázky určené na trénování před testováním
- validation - obrázky určené na validaci
- validation_marked - označené obrázky určené na validaci

A.3.2 Porovnání se vzory

Zde si popíšeme jakým způsobem otestovat metodu, která hledá vajíčka na základě porovnání se vzory.

1. Přesuňte se do složky „Testovani/01_template_matching“.
2. Přesuňte potřebné obrázky do příslušných složek:
 - templates - ořezané vzory ze složky „template_cut“
 - test_marked_pictures - označené testovací obrázky („test_marked“)
 - test_pictures - testovací obrázky („test“)
 - validation_marked_pictures - označené validační obrázky („validation_marked“)
 - validation_pictures - validační obrázky („validation“)
3. V souboru „main.py“ odkomentujte test, který chcete provést:
 - find_templates() - porovná různé kombinace vzorů
 - find_error_function() - porovná jednotlivé chybové funkce
 - find_preprocessing() - porovná jednotlivé předzpracování obrazu
 - test_results() - vyzkouší nejlepší nalezené parametry na testovacích datech
4. Spusťte testování spuštěním souboru „main.py“.
5. Výsledky najdete v souborech „val_scores.txt“ a „scores.txt“.

A.3.3 Porovnání s prstencovými vzory

Nyní si popíšeme jakým způsobem otestovat metodu, která hledá vajíčka na základě porovnání prstencových reprezentací.

1. Přesuňte se do složky „Testovani/02_ring_template_matching“.
2. Přesuňte potřebné obrázky do příslušných složek:
 - templates - neořezané vzory ze složky „template“
 - test_marked_pictures - označené testovací obrázky („test_marked“)
 - test_pictures - testovací obrázky („test“)
 - validation_marked_pictures - označené validační obrázky („validation_marked“)
 - validation_pictures - validační obrázky („validation“)
3. V souboru „main.py“ odkomentujte test, který chcete provést:
 - find_templates() - porovná různé kombinace vzorů
 - find_error_function() - porovná jednotlivé chybové funkce
 - find_preprocessing() - porovná jednotlivé předzpracování obrazu
 - test_results() - vyzkouší nejlepší nalezené parametry na testovacích datech
4. Spusťte testování spuštěním souboru „main.py“.
5. Výsledky najdete v souborech „val_scores.txt“ a „scores.txt“.

A.3.4 Hledání pomocí klasifikačních modelů

Nyní si popíšeme jakým způsobem otestovat metodu, která hledá vajíčka pomocí klasifikačních modelů, jako je logistická regrese nebo vícevrstevnatá perceptronová síť.

1. Přesuňte se do složky „Testovani/03_ring_feature_learning“.
2. Přesuňte potřebné obrázky do příslušných složek ve složce „images“:
 - test_marked - označené testovací obrázky („test_marked“)
 - test - testovací obrázky („test“)
 - train_marked - označené trénovací obrázky pro validaci („train_clear_marked“)
 - train - trénovací obrázky pro validaci („train_clear“)
 - train_all_marked - označené trénovací obrázky pro testování („train_all_marked“)
 - train_all - trénovací obrázky pro testování („train_all“)
 - validation_marked - označené validační obrázky („validation_marked“)
 - validation - validační obrázky („validation“)
3. V souboru „main.py“ odkomentujte test, který chcete provést:
 - find_preprocessing() - porovná jednotlivé předzpracování obrazu u logistické regrese
 - find_radia() - porovná různé klasifikační poloměry u logistické regrese
 - find_distances() - porovná různé vzdálenosti komponent u logistické regrese
 - find_regression_params() - porovná různé parametry klasifikátoru u logistické regrese
 - find_combination() - porovná různé kombinace předzpracování obrazu u logistické regrese
 - test_regression() - vyzkouší nejlepší nalezené parametry na testovacích datech u logistické regrese
 - find_mlp_preprocessing() - porovná jednotlivé předzpracování obrazu u MLP
 - find_mlp_radia() - porovná různé klasifikační poloměry u MLP
 - find_mlp_distances() - porovná různé vzdálenosti komponent u MLP
 - find_mlp_combination() - porovná různé kombinace předzpracování obrazu u MLP
 - find_mlp_params() - porovná různé parametry klasifikátoru MLP
 - test_mlp() - vyzkouší nejlepší nalezené parametry na testovacích datech u MLP
4. Spusťte testování spuštěním souboru „main.py“.
5. Výsledky najdete v souborech „val_scores.txt“ a „scores.txt“.