TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Návrh vestaveného varovacího systému pro dopravní prostředky

Bakalářská práce

Studijní program: B0715A270008 Strojírenství
Autor práce: **Filip Řezníček**
Vedoucí práce: Ing. Andrii Shynkarenko, Ph.D.
Katedra výrobních systémů a automatizace

Liberec 2022

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta strojní

Zadání bakalářské práce

Návrh vestaveného varovacího systému pro dopravní prostředky

Jméno a příjmení: **Filip Řezníček**
Osobní číslo: S19000107
Studijní program: B0715A270008 Strojírenství
Zadávající katedra: Katedra výrobních systémů a automatizace
Akademický rok: **2021/2022**

Zásady pro vypracování:

Cílem bakalářské práce je návrh autonomního systému pro detekci a rozpoznávaní dopravní značky „Zákaz vjezdu vozidel, jejichž výška přesahuje vyznačenou mez“ pomocí umělé inteligence. Téma má aplikační potenciál v automobilech, které nejsou tímto systémem vybaveny.

1. Proveďte rešerši dané problematiky.
2. Seznamte se s technologií rozpoznávaní objektů pomocí umělé inteligence.
3. Na základě studie navrhněte vhodnou technologii a model pro realizaci.
4. Postavte vhodný vestavěný systém pro realizaci vybraného návrhu.
5. Realizujte návrh a otestujte vyrobené zařízení.

Rozsah grafických prací:	dle potřeby
Rozsah pracovní zprávy:	cca 40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] KAEHLER, Adrian a Gary R. BRADSKI. *Learning OpenCV 3: computer vision in C++ with the OpenCV library*. First edition, Second release. Sebastopol, CA: O’Reilly Media, 2017. ISBN 978-1-4919-3799-0.
[2] MÜLLER, Andreas C. a Sarah GUIDO. *Introduction to machine learning with Python: a guide for data scientists*. First edition. Sebastopol, CA: O’Reilly Media, Inc, 2016. ISBN 978-1-4493-6941-5.
[3] RICE, Stephen V., George NAGY a Thomas A NARTKER. *Optical Character Recognition: An Illustrated Guide to the Frontier*. ISBN 9781461550211
[4] CHAKRABORTY, Shouvik a Kalyani MALI, ed. *Applications of advanced machine intelligence in computer vision and object recognition: emerging research and opportunities*. Hershey, PA: IGI Global/Engineering Science Reference, 2020. ISBN 978-1-79982-736-8.
[5] MOORE, Alan D. *Python GUI Programming with Tkinter: Develop responsive and powerful GUI applications with Tkinter.* [online]. Birmingham: Packt Publishing, 2018 [vid. 2021-10-13]. ISBN 978-1-78883-568-8.
[6] LUTZ, Mark. *Programming Python*. Fourth edition. Beijing Boston Farnham Sebastopol Tokyo: O’Reilly, 2019. ISBN 978-0-596-15810-1.

Vedoucí práce:	Ing. Andrii Shynkarenko, Ph.D. Katedra výrobních systémů a automatizace
Datum zadání práce:	15. listopadu 2021
Předpokládaný termín odevzdání:	15. května 2023

L.S.

prof. Dr. Ing. Petr Lenfeld
děkan

Ing. Petr Zelený, Ph.D.
vedoucí katedry

V Liberci dne 15. listopadu 2021

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

21. března 2022 Filip Řezníček

Návrh vestaveného varovacího systému pro dopravní prostředky

Abstrakt

Tato práce řeší problematiku detekcí dopravních značek. U vozidel vyšších, než je výška mostů či tunelů, by mohlo docházet ke kolizi. Cílem této práce je návrh a realizace zařízení pro detekci dopravní značky B16 - výšky mostu, které se klasifikuje jako palubní kamera. Úkolem zařízení je sledovat značky v reálném čase a signalizovat řidiči dopravního prostředku o příliš nízkém mostu či tunelu. Na trhu je dostupná řada systémů, která řeší danou problematiku jiným způsobem, a to umístěním statické kamery či senzorů před kritickým místem. Jedná se o stacionární umístění systému. Výstupem této práce je návrh kompaktního vestavěného zařízení a aplikace do vozidla. V teorii je popsán princip a metody detekce objektů, počítačového vidění včetně metod učení klasifikátoru a umělých neuronových sítích, na kterých vlastní detekce pracují. Následuje rešeržní část pro zjištění dostupných alternativ detekce a vlastní realizace včetně testování funkčnosti.

Klíčová slova: detekce objektů, optické rozpoznávání textu, Raspberry Pi, Teserract OCR, OpenCV, Haar, detekce převýšení vozidla

Design of an embedded warning system for vehicles

Abstract

This thesis solves the issue of traffic sign detection. For vehicles higher than the height of bridges or tunnels, a collision could occur. The aim of this work is the design and implementation of a device for detecting the traffic sign B16 - the height of the bridge, which is classified as an on-board camera. The task of the device is to monitor the sign in real time and signal the driver of a vehicle with a bridge or tunnel that is too low. There are a number of systems available on the market that address the issue in a different way, by placing a static camera or sensors in front of a critical point. This is a stationary system location. The output of this work is the design of a compact built-in device and application to the vehicle. The theory describes the principle and methods of object detection, computer vision, including methods of classifier learning and artificial neural networks, on which the actual detection works. The research part follows to find out the available detection alternatives and the actual implementation, including functionality testing.

Keywords: objects detection, optical character recognition, Raspberry Pi, Tesseract OCR, OpenCV, Haar, overheight vehicle detection

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Andrii Shynkarenkovi, Ph. D., za odborný přístup, cenné rady a vstřícnost při konzultacích a zpracování bakalářské práce. Mým rodičům vděčím za možnost studia a vzdělání. A v neposlední řadě chci poděkovat kolegům – kamarádům za pomoc a spolupráci v průběhu studia.

Obsah

Seznam obrázků

Seznam tabulek

Seznam zdrojových kódů

Seznam zkratek

AI	Artificial intelligence, Umělá inteligence
ML	Machine learning, Strojové učení
DL	Deep learning, Hluboké učení
ANN	Artificial neural networks, Umělé neuronové sítě
P	Perceptron
FFNN	Feedforward neural networks, Dopředné neuronové sítě
DFFNN	Deep feedforward neural networks, Hluboké dopředné neuronové sítě
MLP	Multilayer perceptron, Vícevrstvý perceptron
RNN	Recurrent neural networks, Rekurentní neuronové sítě
LSTM	Long short-term memory, Síť s dlouhou krátkodobou pamětí
CNN	Convolutional neural networks, Konvoluční neuronové sítě
DCNN	Deep convolutional neural networks, Hluboké konvoluční neuronové sítě
RGB	Red-green-blue, Červená-zelená-modrá
CV	Computer vision, Počítačové vidění
SPZ	Statní poznávací značka
OCR	Optical character recognition, Optické rozpoznávání znaků
SVM	Support Vector Machines, Metoda podpůrných vektorů
HOG	Histogram of Oriented Gradients, Histogram orientovaných gradientů
SIFT	Scale-Invariant Feature Transform, Měřítko-invariantní transformace prvků
ORB	Oriented FAST and Rotated BRIEF, Orientovaný FAST a otočený BRIEF
SSD	Single Shot MultiBox Detector
R-CNN	Region Based Convolutional Neural Networks, Rekurentní neuronová síť založená na regionech
YOLO	You Only Look Once, Podíváš se jen jednou
ROI	Region of Interest, Oblast zájmu
ICR	Intelligent Character Recognition, Inteligentní rozpoznávání znaků
API	Application Programming Interface, Programovací rozhraní aplikace
GUI	Graphic user interface, Grafické uživatelské rozhraní

LS	Limit Switch, Koncový spínač
LLDR	Laser & Light Dependence Rezistor, Rezistor závislý na světle a laseru
LED	Light-Emitting Diode, Světelná dioda
HDMI	High-Definition Multimedia Interface, Multimediální rozhraní s vysokým rozlišením
USB	Universal Serial Bus, Univerzální sériová sběrnice
Wi-Fi	Family of wireless network protocols, Rodina bezdrátových síťových protokolů
ARM	Architectures for computer processors, Označení architektury procesorů
GPIO	General-purpose input/output, Univerzální vstupní/výstupní pin
NAS	Network Attached Storage, Datové uložiště
RF	Radio Frequency, Rádiová frekvence nebo Radiofrekvenční
RAM	Random Access Memory, Paměť s náhodným přístupem
HQ	High Quality, Vysoce kvalitní
TFT	Thin Film Transistor, Tenký tranzistorový film
CPU	Central processing unit, Centrální procesorová jednotka
GPU	Graphics processing unit, Grafická procesorová jednotka
HD	High Definition, Vysoké rozlišení
Full HD	Full High Definition, Plné vysoké rozlišení
FPS	Frames Per Second, Snímky za sekundu
CSI	Camera Serial Interface, Sériové rozhraní kamery
PC	Personal computer, Osobní počítač
FDM	Fused deposition modeling, Modelování tavené depozice
GB	Gigabyte, Gigabajt
GHz	Gigahertz, Gigahertz

Úvod

Činnosti spojené s umělou inteligencí nebo se strojovým učením jsou velmi populární a stávají se nedílnou součástí našich životů. Výstupem každého zkušeného fotografa, který se zaměřuje na fotografování sportu, resp. osob v pohybu, je správně vybalancovaný a především ostrý snímek. Pro takové případy jsou moderní fotoaparáty vybaveny umělou inteligencí, která detekuje obličeje nebo oči lidí. Snahou je ostření přávě na oblast očí, díky čemuž jsou fotky pěkně ostré.

Nehody jsou důsledky lidských chyb, např. nedostatek spánku, vysoká rychlost, přetížení kamionů nebo nedostatky v infrastruktuře. Cílem veškerých vyvíjených systémů je snaha zábránit podobným konfliktům a zvýšit bezpečnost. Snahou této práce je taktéž přispět pro zajištění bezpečnosti. Kdo ví, třeba se jednou tento systém stane součástí běžných automobilů a kamionů. Nedocházelo by nadále ke kritickým situacím, které by vyžadovaly ztráty na lidských životech.

Postup práce je rozdělen do jednotlivých kapitol. Níže je pro čtenáře uveden seznam obsahů kapitol.

Kapitola **1 Umělá inteligence, strojové učení a hluboké učení** vysvětluje základní pojmy, které jsou klíčové pro porozumění dané problematiky. Umělá inteligence se v poslední době stává každodenní záležitostí a nedílnou součástí našich životů.

Kapitola **2 Umělá neuronová síť** popisuje fungování neuronových sítích, které jsou klíčové při rozpoznávání znaků. Jsou zde vypsány vybrané architektury, které se běžně využívají. U některých z nich je také připojen příklad užití, aby si čtenář propojil teorii s praxí.

Kapitola **3 Počítačové vidění a optické rozpoznávání znaků** je jednou z nejdůležitějších úseků teoretické částí této práce. Zabývá se detekcí objektů a jejich ohraničením. V této kapitole jsou vypsány nástroje, které se používají pro počíta-

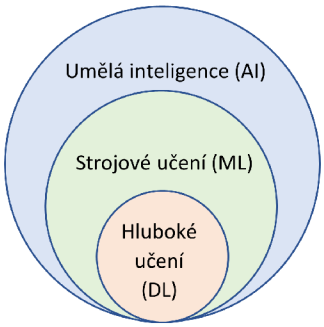
čové vidění. Součástí je také vysvětlení principu OCR včetně zahrnutí dostupných metod.

Kapitola **4 Realizace** je klíčovou sekcí práce. Kapitola začíná rešerží zařízení a systémů, které jsou na trhu dostupné. Zabývá se samotnou realizací, která zahrnuje výběr nástrojů, zařízení, tvorbu kódu, implementaci a samotnou montáž zařízení a testování. Na úplném konci je vložena diskuze a shrnutí kapitoly.

1 Umělá inteligence, strojové učení a hluboké učení

Tato kapitola se zaobírá tvorbou modelů, které se používají pro samotnou detekci objektů. Je nezbytné začít s popisem metod učení a postupně se dostat k pointě, resp. nejdůležitější části práce. Přirovnat se to dá např. k vaření bez receptu.

Nejčastější chybou je zámněna významů pojmů jako jsou **umělá inteligence**, **strojové** nebo **hluboké učení**. Vztah mezi nimi je znázorněn na obrázku 1.1 a jejich stručný význam popsán níže.



Obrázek 1.1: Vztah umělé inteligence, strojového a hlubokého učení

1.1 Umělá inteligence

Umělá inteligence (artificial intelligence, zkratka **AI**) je obor informatiky zabývající se tvorbou systémů, které dokáží napodobit lidský intelekt a chování. Dlouhodobým cílem je právě vznik tzv. obecné umělé inteligence, která by byla schopna se učit, zcela samostatně rozhodovat a umět řešit různé problémy. [1]

Mezi produkty umělé inteligence se řadí virtuální asistenty, např. Siri (od společnosti Apple), Google nebo Alexa (od společnosti Amazon), které v rámci rozpoznávání hlasu dokáží provádět operace typu vyhledávání na internetu, spuštění hudby nebo

zadávání schůzky do kalendáře.

Termín umělá inteligence byl poprvé použit v roce 1956, tedy v roce založení tohoto oboru. Použil ho jeden ze zakladatelů – John McCarthy, [2, s.17] který AI definuje následně: „*Je to věda a technika tvorby inteligentních strojů, zejména inteligentních počítačových programů.*“[1] [3, s.2]

1.2 Strojové učení

Strojové učení (Machine Learning, zkratka **ML**) je podmnožinou AI. Jedná se o proces analýzy dat, pomocí které se počítač sám učí. Využívá algoritmy pro identifikaci vzorů obsažených ve vstupních datech pro vytvoření datového modelu. S větším množstvím dat jsou výsledky přesnější. Stejně tomu je u lidí, kteří se zlepšují díky praxi. Na základě nových dat se ML dokáže rozšiřovat i bez lidské pomoci. O to se stará umělá inteligence, která získané znalosti aplikuje do praxe. Strojové učení je tedy jejím nástrojem. [4] [5]

Algoritmy strojového učení se dělí na tři základní druhy:

- Učení s učitelem (Supervised Learning)
- Učení bez učitele (Unsupervised Learning)
- Zpětnovazební učení (Reinforcement Learning)

Někdy se uvádí čtvrtý typ, který kombinuje učení se s učitelem a bez učitele (semi-supervised learning).

1.2.1 Učení s učitelem

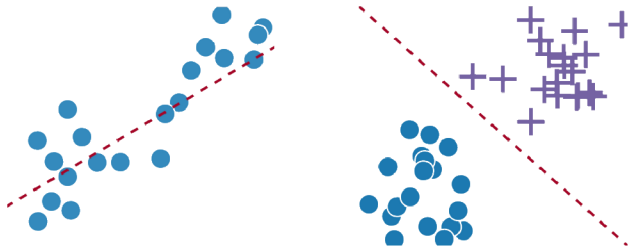
Na základě sady označených příkladů, které systému učitel poskytne, algoritmy vytváří predikce, které porovnává s očekávaným výsledkem. Na základě velikosti chyby (rozdílu) je model upraven tak, aby poskytoval přesnější předpovědi, tedy minimalizoval svou chybu. Postup se opakuje tak dlouho, dokud celková chyba neklesne pod určitou hodnotu. [4] [5]

Například bude-li cílem zjistit, kolik bude mít konkrétní město počet obyvatel za čtyři roky. Výsledek z datové sady bude obsahovat údaje o počtu obyvatel měst v každém roce za dobu 100 let, využívá popisky, které již v sadě existují: počet obyvatel,

[1] It is the science and engineering of making intelligent machines, especially intelligent computer programs.

město a rok. [5]

Zkrátka zdali víme, jak by měl výsledek vypadat, je toto učení ideální. Je ovšem důležité připravit tzv. množinu trénovacích dat, resp. snímků. Každý prvek takového souboru zahrnuje vstupní data a očekávaný výstup. Tato množina se dále dělí na dvě či tři podmnožiny.



(a) Regrese **(b)** Klasifikace

Obrázek 1.2: Grafické znázornění regrese a klasifikace (zdroj: [6])

První z nich je trénovací množina (training dataset), která spočívá v samotném učení modelu. Další je validační (ověřovací) množina (validation dataset), která kontroluje průběh učení na základě úspěšnosti. Zvýší-li se hodnota předpovědi trénovací množiny, a naopak klesne pro ověřovací množinu, dochází k tzv. přeučení. Znamená to, že si model pouze pamatuje trénovací data. Testovací množina (test dataset) vyjadřuje přesnost modelu po samotném dokončení učení. [7]

Metoda učení řeší dva typy úloh:

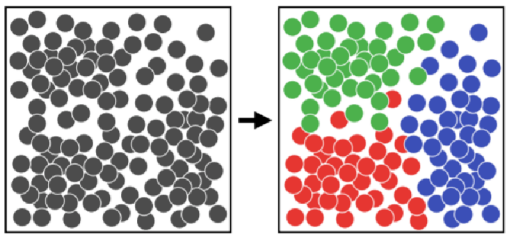
- Klasifikace (classification), kdy dojde k rozpoznání objektu a rozdělení do tříd, např. červená barva – obrázek 1.2a,
- Regrese (regression), kdy se na základě vstupu odhaduje výstup. Hodnotou je reálné číslo, které může vyjadřovat např. cenu. Oproti klasifikaci jde o rychlejší proces, protože nedochází k žádným výběrům (viz 3.2.2 Konvoluční neuronová síť založená na regionech) – obrázek 1.2b .

Příprava množiny dat s popisy se připravuje ručně nebo jen částečně automaticky. Trénované učení vždy umí pouze to, čemu ho člověk naučí. Systém nedokáže provádět činnosti, které nebyly předem definovány, resp. dokáže, ale s příliš velkou chybou.

1.2.2 Učení bez učitele

Při učení bez učitele nejsou označené datové body. Máme tedy surová data a algoritmus je musí sám označit např. uspořádáním nebo popisem jejich struktury.

Technika je užitečná, pokud nevíme, jak by měl výstup vypadat. Vše se tudíž děje bez cizí pomoci. [4] [5]



Obrázek 1.3: Grafické znázornění shlukování (zdroj: [7])

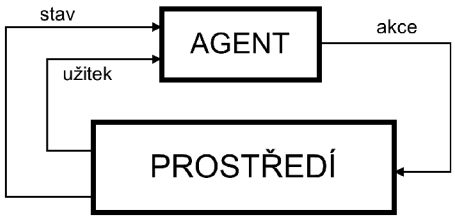
Metoda učení řeší dva typy úloh:

- Shlukování (clustering), kdy se data, bez znalosti obsahu, rozřazují do skupin s podobnými vlastnostmi,
- Asociace (association), kdy se mezi daty vytváří souvislosti, které se skládají do uceleného konceptu.

Učení využívá zmíněné shlukování dat k vytvoření komplexnějších informací. Například platforma YouTube Music sleduje a ukládá historii přehraných skladeb uživatelů. Následně jim zobrazuje tipy na podobný seznam. Metody učení, bez učitele a s učitelem, lze kombinovat.

1.2.3 Zpětnovazební (posilující) učení

Zpětnovazební učení je nejčastější forma ML, jenž využívá algoritmy, které se učí z předchozích výsledků. Následně se rozhodují, jakou akci provést příště. Na základě provedené akce algoritmus obdrží zpětnou vazbu o správnosti rozhodnutí – správná, neutrální nebo nesprávná volba. Technika se používá pro automatizované systémy, které musí provádět velké množství rozhodnutí bez lidského dozoru. [4] [5]



Obrázek 1.4: Princip zpětnovazebního učení (zdroj: převzato z [8])

Příkladem může být autonomní vůz, který má dodržovat předpisy a zajišťovat bezpečnost lidí. V rámci získaných zkušeností a historii zpětné vazby, se systém naučí,

jak se udržet v jízdním pruhu, brzdit před chodci a nepřekračovat povolené rychlosti. [4] [5]

1.3 Hluboké učení

Hluboké učení (Deep Learning, zkratka **DL**) je typ strojového učení, od kterého se liší metodami, kterými se učí a typem dat, se kterými pracuje. Lze řešit složitější úlohy a dosahovat lepších výsledků, ovšem na úkor nárůstu výpočetního výkonu. DL v úkolech, jako je detekce objektů, překlad jazyků, rozpoznávání řeči apod., využívá vícevrstvé umělé neuronové sítě. Pro dosažení nejlepších výsledků se vyžaduje mnoho dat a výkonné počítače. [9]

Algoritmy hlubokého učení jsou neuvěřitelně složité a existuje celá řada neuronových sítí, mezi které patří např. Konvoluční neuronové sítě (viz. 2.3.4 Konvoluční neuronová síť) nebo Rekurentní neuronové sítě (viz. 2.3.3 Rekurentní neuronové sítě). Mezi nejběžněji používané architektury se řadí např. Tensorflow nebo PyTorch. [9]

1.4 Shrnutí kapitoly

Jelikož byl cílem návrh vestavěného systému, byl kladen velký důraz na kompaktnost, cenu, využitelnost dostupných komponent. Zvolila se jednoduchá a ověřená metoda založená na natrénovaném klasifikátoru. Později v této práci bude upřesněno, proč se pro detekci objektů zvolila metoda Viola – Jones. Nicméně tento způsob detekce je založen na strojovém učení, konkr. na učení pod dohledem. Toto učení oproti hlubokému nevyžaduje příliš trénovaích dat a vysoký výpočetní výkon.

2 Umělá neuronová síť

Pro zpracování obrazu – klasifikace, detekce objektů, je zapotřebí znalost dostupných umělých neuronových sítí a jejich práci s daty. Níže jsou popsány známé a běžně používané sítě.

Umělá neuronová síť (Artificial Neural Networks, zkratka **ANN**) je matematický model inspirovaný biologickými neuronovými sítěmi, které tvoří lidský mozek. Je tvořen sériovým a paralelním spojením umělých neuronů. Jinými slovy se jedná o soubor propojených jednotek – umělých neuronů, které zpracovávají data, navzájem spolu komunikují a paralelně pracují. Následující tabulka 2.1 vysvětluje používané symboly. [7]

Tabulka 2.1: Význam symbolů použitých v následujících architekturách (zdroj: převzato z [10])

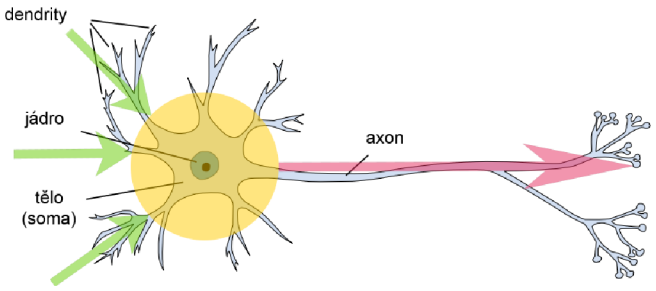
Schem. značka	Původní název	Český překlad
	Kernel	Kernel
	Recurrent Cell	Vracející se buňka
	Convolution or Pool	Konvoluce nebo sdružování
	Memory Cell	Paměťová buňka
	Hidden Cell	Skrytá buňka
	Input Cell	Vstupní buňka
	Output Cell	Výstupní buňka
	Backfed Input Cell	Zpětnovazební vstupní buňka
	Match Input Output Cell	— vstupní/výstupní buňka

2.1 Biologický neuron

Biologické neurony jsou buňky primárně určené pro přenos a zpracování signálů. Základem neuronů je tělo označované jako **soma**, ze kterého vycházejí výběžky tzv.

dendrity. Ty slouží jako vstupní přenosové kanály, zatímco **axon** reprezentuje výstupní přenosový kanál. Díky rozvětvenému axonu, může být připojen k dendritům jiných neuronů. [7]

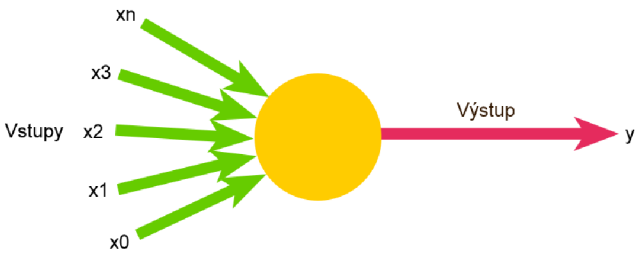
Na obrázku 2.2 je patrné porovnání biologického a umělého neuronu.



Obrázek 2.1: Znázornění biologického neuronu (zdroj: převzato z [7])

2.2 Umělý neuron

Umělý neuron je prostá výpočetní jednotka, která zpracovává vstupní signály, komunikuje a spolupracuje s jinými jednotkami. První umělý neuron vynalezli v roce 1943 neurofyziolog Warren McCulloch a logik Walter Pits. [11]



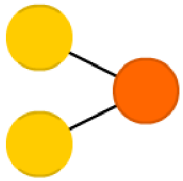
Obrázek 2.2: Základní struktura umělého neuronu (zdroj: převzato z [12])

Tento neuron přijímá jeden nebo více vstupů, které sečte a vytvoří výstup. Nejjednodušší NN se nazývá **perceptron**, jenž je složen z jednoho nebo několika modelů biologických neuronů.

2.3 Typy umělých neuronových sítí

Existuje řada různých typů sítí, kde každá je vhodná pro jiné aplikace s hlubokým učením. Několik příkladů běžně používaných neuronových sítí je vypsáno níže.

2.3.1 Perceptron



Obrázek 2.3: Perceptron (zdroj: [10])

Perceptron (zkratka **P**) je algoritmus pro řízené učení binárních klasifikátorů, kde binární klasifikátor je funkce, která může rozhodovat, zdali vstup stávající se z vektoru čísel patří do konkrétní třídy nebo ne. [13] Rovněž se jedná o typ lineárního klasifikátoru, který na základě funkce lineárního prediktoru kombinující sadu vah (tj. číselná hodnota, která závisí na modelu) s příznakovým vektorem provádí predikce. [7]

2.3.2 Dopředné neuronové sítě

(a) Dopředná neuronová síť **(b)** Hluboká dopředná neuronová síť

Obrázek 2.4: Dopředné neuronové sítě (zdroj: [10])

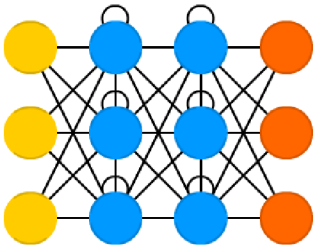
Dopředné neuronové sítě (feedforward neural networks, zkratka **FFNN**) patří mezi první a nejjednodušší NN. V této síti se informace pohybují pouze jedním směrem, a to od vstupní (input layer) přes skryté (angl. hidden layer) až po výstupní (angl. output layer) buňky. Neobsahují žádné cykly ani smyčky. Na obrázku 2.4a je znázorněna jednoduchá FFNN, kterou tvoří dvě vstupní a skryté buňky a jedna výstupní buňka. [14] [7]

Dopředné neuronové sítě obsahující více skrytých vrstev jsou označovány jako **hluboké dopředné neuronové sítě** (deep feedforward neural networks, zkratka **DFFNN**), které jsou zobrazeny na obrázku 2.4b. Bývají také nazývány jako vícevrstvé perceptrony (multilayer perceptron, zkratka MLP). [14] [7]

Vícevrstvé sítě využívají různé metody učení, kde mezi nejoblíbenější patří zpětnovazební (viz 1.2.3 Zpětnovazební (posilující) učení). Bývají součástí jiných architektur neuronových sítí, např. konvolučních neuronových sítí (viz 2.3.4 Konvoluční neuronová síť).

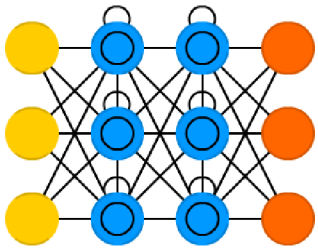
2.3.3 Rekurentní neuronové sítě

Rekurentní neuronová síť (Recurrent Neural Networks, zkratka **RNN**) mají oproti dopředným neuronovým sítím vnitřní stav (paměť). Skryté neurony dostávají informace jak z předchozí vrstvy, tak od sebe samých, na jejichž základě dokáží předpovídat, co přijde dál. Jinak řečeno, RNN predikují výsledky v sekvenčních datech, které jiné algoritmy nedokážou. [7] [15]



Obrázek 2.5: Rekurentní neuronové sítě (zdroj: [10])

Vzhledem k jednoduché paměti jsou použitelné pro sekvenční data, jako jsou například texty. Jejich běžné využití zahrnuje např. předpovídání slov (automatické dokončování), hlasové vyhledávání nebo aplikaci Google Translate. [15]



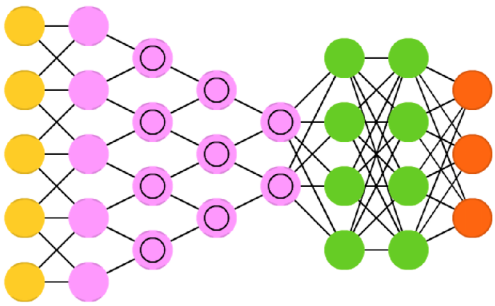
Obrázek 2.6: Long short-term memory (zdroj: [10])

„*Jedním z vážných omezení rekurentních neuronových sítí je neschopnost zachytit dlouhodobé závislosti v sekvenci (např. Existuje závislost mezi dnešní cenou a cenou před 2 týdny?).*“ [16] Jednou variantou, jak tento problém obejít, je RNN s **Long short-term memory**, zkratka **LSTM**. Do češtiny se překládá jako síť s dlouhou

krátkodobou pamětí. Cílem chování je zapamatovat si informace po dlouhou dobu (viz obrázek 2.6). [7]

2.3.4 Konvoluční neuronová síť

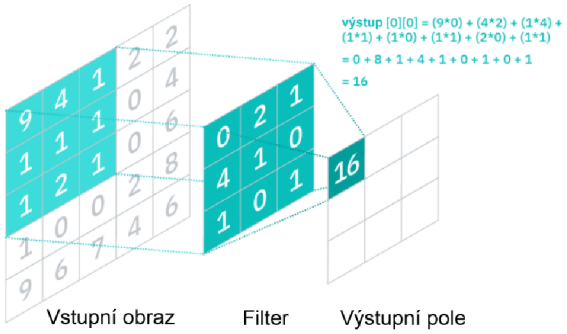
Konvoluční neuronová síť (Convolutional Neural Network, zkratka **CNN**, nebo deep convolutional neural networks, DCNN) je využívána především při zpracování obrazu např. ke klasifikaci obrazu nebo detekci objektů (viz 3.1 Klasifikace obrazu a 3.2 Detekce objektů), ale mohou se použít i pro jiné typy vstupu např. pro zvuk. Obraz je interpretován jako matice obsahující hodnoty, které představují intenzitu pixelů. Pro černobílý obraz je pouze jedna matice, kde jsou hodnoty v rozmezí 0 až 255, tedy od černé po bílou. Pro barevné (RGB) obrazy budou tedy matice tři a její hodnoty budou prezentovány stejně, tedy od 0 do 255 pro každou barvu. [7] [17]



Obrázek 2.7: Konvoluční neuronové sítě(zdroj: [10])

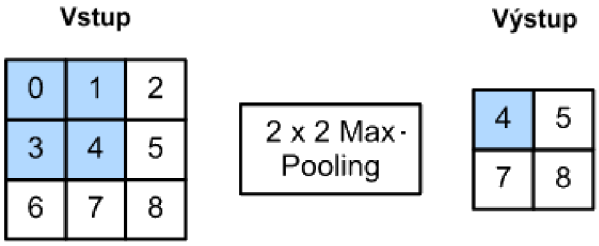
Metodou CNN je operace tzv. **konvoluce**, která po malých částech postupně čte obraz. Velikosti těchto částí jsou závislé na rozměru matice známé jako **kernel** (konvoluční jádro nebo filtr). Filtr (kernel) se v rámci obrazu posouvá a dochází k násobení jeho hodnot s hodnotami, které zakrývá, resp. které leží pod ním. Hodnota matice na výstupu konvoluční vrstvy je tvořena součtem těchto násobků. Operace se opakuje, dokud filtr neprojede celou šířkou obrazu. Následně pokračuje od začátku dalšího řádku do doby, než nepřečte celý obraz. Princip konvoluce je popsán na obrázku 2.8. [7] [17]

Důležité je, aby se CNN skládaly z více konvolučních vrstev. Každá z nich zachycuje důležité vlastnosti obrázku, od jednoduchých rysů (jako jsou hrany nebo barvy) přes textury a vzory až po komplexnější objekty ve vrstvách. Je běžné mezi po sobě jdoucí konvoluční vrstvy vkládat vrstvu sdružovací. Její funkcí je progresivně zmenšovat prostorovou velikost reprezentace, a tím omezit množství parametrů a výpočtů v síti. Metoda se nazývá **pooling** (sdružování) a dělí se na max-pooling a avg-pooling.



Obrázek 2.8: Princip fungování konvoluce (zdroj: [17])

U max-pooling dochází k výpočtu maximální hodnoty prvků v okně, zatímco u avg--pooling k výpočtu průměrné hodnoty. Princip sdružování je vidět na obrázku 2.9. [7] [17]



Obrázek 2.9: Metoda max-pooling (zdroj: převzato z [18])

Sdružováním se sníží nároky na výpočetní výkon, a také dochází k extrakci významných rysů. Skládáním konvolučních a sdružovacích vrstev lze získat dostatečně malou reprezentaci původního obrazu.

Mezi používané architektury CNN patří LeNet, AlexNet, GoogLeNet, VGGNet nebo ZFNet.

2.4 Shrnutí kapitoly

Metoda Viola – Jones je založena na strojovém učení, což z ní činí jednoduchý, přesný a rychlý způsob detekce objektů. Pro řešení danného úkolu se ovšem dají použít algortimy založené na CNN, nicméně se u nich vyžaduje větší výpočetní výkon zařízení (viz kapitola 4.5 Shrnutí a diskuze). Optické rozpoznávání znaků (konkr. Tesseract OCR), které bude vysvětleno v další kapitole, je založeno na LSTM rekurentní neuronové síti, čímž dosahuje lepších výsledků oproti ostatním algoritmům založených na jiném typu sítě.

3 Počítačové vidění a optické rozpoznávání znaků

Tato kapitola je jednou z nejdůležitějších a pojednává o zpracování obrazu včetně dostupných postupů. Pro detekci je nezbytné mít připravená trénovací data a z nich naučený model, a metodu pro detekci objektů. Dá se říct, že počítačové vidění je nadmnožinou dosud zmíněné teorie.

Počítačové vidění (Computer Vision, zkratka **CV**) je obor počítačové vědy, který umožňuje počítačům a systémům vidět, identifikovat a zpracovávat vizuální data v podobě digitálních snímků, videí a dalších obrazových vstupů stejným způsobem jako lidské vidění a poskytovat adekvátní výstup. [19]

Cílem je trénink strojů k vykonávání těchto funkcí za daleko kratší čas než lidský zrak. Systémy dokáží analyzovat tisíce objektů nebo procesů za jednotku času a zaznamenávat vzniklé problémy či vady. [19]

Obrázek 3.1: Počítačové vidění (zdroj: [20])

Počítačové vidění poskytuje data pro detekci a rozpoznávání znaků. Ty vyžadují velké množství dat, ze kterých provádí analýzy. Proces trvá tak dlouho, dokud systém nenajde rozdíly a snímky nerozpozná. Například pro rozpoznávání květin je nutné vložit veliké množství dat s květinami. Počítač je analyzuje, identifikuje vzory, které

jsou všem obrázkům podobné. Na konci procesu vytvoří jejich model, díky kterému bude zařízení schopné přesně detekovat, zda je na konkrétním obrázku daný objekt nebo ne. [19] [21]

Mezi nejoblíbenější aplikace a využití CV patří klasifikace vozidel, detekce vytíženosti parkovišť, rozpoznávání SPZ, analýza dopravního toku nebo počítání lidí.

Nyní něco málo k historii. První experimenty se uskutečnily v 50. letech minulého století. Ty využívaly některé z neuronových sítí k detekci hran nebo k dělení objektů do kategorií. První komerční využití počítačového vidění přišlo v 70. letech, kdy se pomocí OCR (viz 3.3 Optické rozpoznávání znaků) rozpoznával tištěný nebo ručně psaný text. Uplatnění se našlo při interpretaci psaného textu pro nevidomé, nebo ve zpracování faktur, dokumentů, rozpoznávání SPZ a dalších aplikací. [19]

V 90. letech došlo k vývoji internetu, a tím zpřístupnění online souborů pro analýzu. Začaly se vytvářet programy pro rozpoznávání obličejů, které díky velkému množství dat dokázaly identifikovat lidi na obrázcích a videích. Díky vývoji se výpočetní výkon, zařízení pro CV a analýzu stal mnohem dostupnější. Za méně než deset let se míra přesnosti klasifikace objektů zvýšila z 50 na 99 procent. V současnosti jsou systémy přesnější v detekci a reakci na obraz než lidé. [22]

V rámci CV často dochází k záměně významů pojmů, jako jsou rozpoznávání objektů, klasifikace obrazu nebo detekce objektů. Rozpoznávání objektů je obecný výraz, který popisuje soubor úloh počítačového vidění. [23] Níže jsou popsány zbylé metody.

3.1 Klasifikace obrazu

Metoda klasifikace obrazu se používá ke kategorizaci neboli určení třídy, resp. tříd rozpoznaných objektů, a k přiřazení štítků skupinám pixelů nebo vektorů reprezentující objekty na obrázku. Vstupní obraz je počítačem vnímán jako pole matic o velikosti rozlišení obrazu a pomocí algoritmů je prováděna analýza těchto dat. Vstupem do procesu je např. snímek s jedním nebo více objekty a výstupem je určení třídy, resp. tříd rozpoznaných objektů. [24]

Pro klasifikaci obrazu lze použít **metoda podpůrných vektorů** (Support Vector Machines, zkratka **SVM**), která je jednou z algoritmů strojového učení založená na učení s učitelem (viz 1.2.1 Učení s učitelem). Jednotlivé třídy (kategorie) jsou od sebe odděleny pomocí tzv. nadroviny, kterou ve 2D reprezentuje lineární funkce.

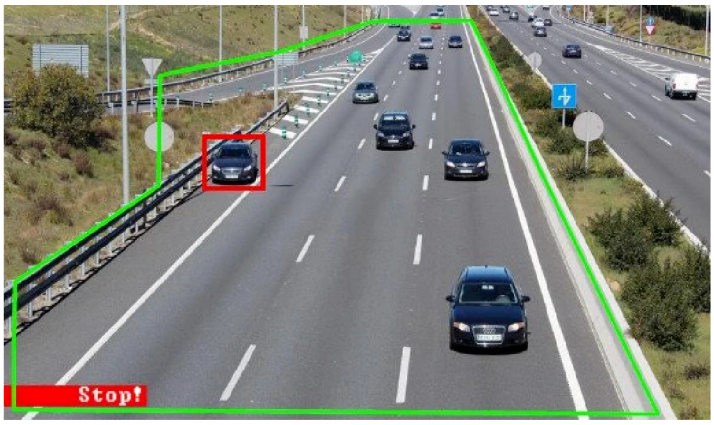
Přímka slouží jako rozhodovací hranice při rozřazování do kategorií. Síla a přesnost algoritmu je závislá na použité funkci kernel. [25] [26]

Další je metoda **rozhodovacích stromů** (Decision Trees), kterou tvoří stromová datová struktura, jenž u vybraných funkcí využívá příkazy if/else neboli pokud/jinak. Zkrátka se vstupní objekty třídí na základě jejich vlastností do kategorií. Příkladem může být: *Je venku hezké počasí?* Na základě odpovědi Ano/Ne se provede další činnost, např. *půjdu se projet na kole*, nebo *zůstanu doma a budu hrát hry*. Tímto tříděním dojde k určení třídy.

Mezi další klasifikační metody dále patří Umělé neuronové sítě (viz 2 Umělá neuronová síť) a Konvoluční neuronové sítě (viz 2.3.4 Konvoluční neuronová síť).

3.2 Detekce objektů

Cílem detekce je na daném vstupním obrazu lokalizovat polohy detekovaných objektů, přiřadit jim ohraničující rámečky a typy (kategorie) těchto výskytů. Techniky detekce se používají ve skutečném světě např. při detekci chodců, vozidel, dopravních značek nebo obličeje, u kterého lze rozlišit oči, nos, rty a rysy, jako jsou barva pleti nebo vzdálenost očí. [27]



Obrázek 3.2: Příklad detekce stojících vozidel (zdroj: [28])

Metody detekce objektů se řadí do dvou přístupů:

Tradiční přístupy, u kterých se pomocí jedné z následujících metod určí rysy a provede klasifikace (viz 3.1 Klasifikace obrazu), patří mezi starší metody, a to před vznikem hlubokého učení – před rokem 2014. [27]

- Histogram orientovaných gradientů (HOG)

- Scale-invariant feature transform (SIFT, příp. alternativa ORB)
- Viola-Jones založená na funkcích Haar

Druhým způsobem jsou neuronové sítě, u kterých je možné provádět detekci bez nadefinovaných funkcí a bývají často založeny na konvolučních neuronových sítích (viz 2.3.4 Konvoluční neuronová síť). Ty vznikly po zavedení DL – po roce 2014. [27]

- Single Shot MultiBox Detector (SSD)
- Návrhy regionů – R-CNN, Fast R-CNN, Faster R-CNN
- You Only Look Once (YOLO)

Detektory objektů založených na neuronových sítích plní dva úkoly:

1. Dle vysoké pravděpodobnosti výskytu objektů vybrat skupiny oblasti zájmu (viz 3.2.2 Konvoluční neuronová síť založená na regionech) a
2. Na základě aplikace CNN každý výskyt v obrazu klasifikovat, odhadnout jeho velikost a tu ohraničit.

Procesy lze rozdělit do dvou fází – jednostupňové a dvoustupňové detektory.

Jednostupňové detektory předvídají ohraničení objektů na snímcích bez návrhu regionu, čímž nedochází k velké spotřebě času. Algoritmy jsou jednodušší než dvoustupňových a jsou vhodné pro aplikace v reálném čase. Na druhou stranu dochází k problémům při rozpoznávání skupin malých objektů nebo tvarově složitých tvarů. [27]

Mezi nejoblíbenější z nich patří detektory YOLO, SSD a RetinaNet.

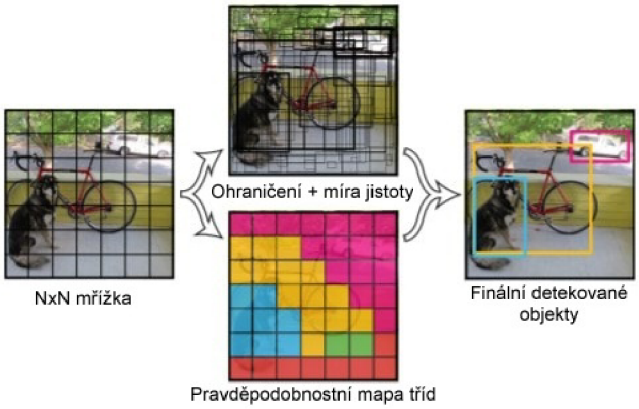
Pomocí hlubokých znaků jsou přibližně navrženy oblasti detekovaných objektů a to předtím, než jsou tyto znaky použity ke klasifikaci. Těmito úkoly se dosahuje vysoké přesnosti detekce a snížení rychlosti procesu. [27]

Mezi dvoustupňové metody se řadí konvoluční neuronová síť založená na regionech (R-CNN) a její novější a rychlejší alternativy, jako jsou Faster R-CNN nebo Mask R-CNN. Několik metod je stručně popsaných v následujících kapitolách. [27]

3.2.1 You Only Look Once

Cílem metody **You Only Look Once** (zkratka **YOLO**), která je založená na hlubokém učení, je rozdělit vstupní obraz na mřížku NxN mřížek (např. mřížka 3x3),

kde na každou oblast je aplikována detekce objektů. Algoritmus využívá konvoluční neuronové sítě (viz 2.3.4 Konvoluční neuronová síť) a využívá se zejména pro detekci v reálném čase. [29]

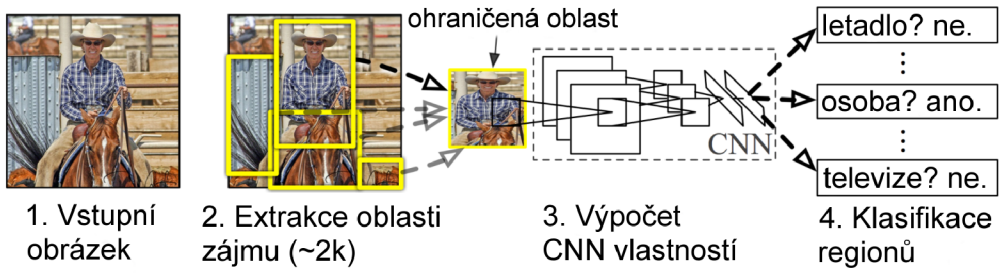


Obrázek 3.3: Princip YOLO metody (zdroj: [29], překlad: autor)

Technika využívá tři fáze (viz obrázek 3.3) – rozdělení na NxN oblasti, ohraničení detekovaných objektů a stanovení jejich jistoty a rozdělení do kategorií.

3.2.2 Konvoluční neuronová síť založená na regionech

Vědec Ross Girshick [30] navrhnul metodu **CNN založenou na regionech** (Region-based convolutional neural networks, zkratka **R-CNN**), která využívá selektivní vyhledávání pro extrakci pouhých 2000 oblastí, které nazval oblast zájmu (Region of Interest, zkratka ROI). Každá oblast prochází CNN a jsou vytvářeny výstupní funkce, díky kterým SVM (viz 3.1 Klasifikace obrazu) klasifikuje jednotlivé segmenty. [31] [32]



Obrázek 3.4: Princip R-CNN metody (zdroj: [32], překlad: autor)

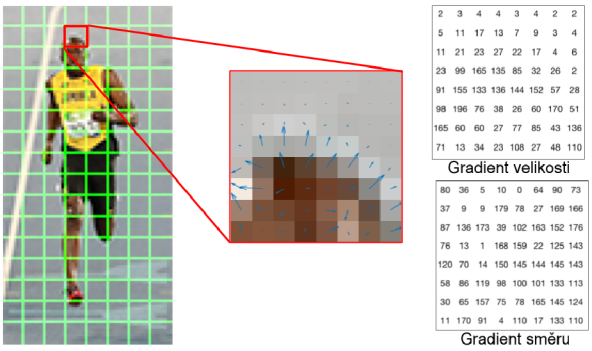
Kvůli obrovskému množství regionů je celý proces pomalý a nelze implementovat v reálném čase. Každý trénovací snímek zabere zhruba 47 sekund a je neefektiv-

ní. Byly proto vyvinuty alternativy této metody, např. Fast R-CNN nebo Faster R-CNN, které řeší některé tyto problémy. Fungují na opačném principu. Vstupní obraz nejprve prochází CNN, kde se vytvoří konvoluční mapa rysů, ze které se navrhují regiony. Konvoluce se tedy provádí pouze jednou v rámci snímku. [31]

3.2.3 Histogram orientovaných gradientů

Histogram orientovaných gradientů (Histogram of Oriented Gradients, zkratka **HOG**) se zaměřuje na tvar a strukturu objektu. K výpočtu prvků používá velikost a úhel gradientů, ze kterých generuje histogram. Jedná se o jednoduchou metodu pro detekci objektů. [33] [34]

Na obrázku 3.5 je ukázán princip metody HOG. Uprostřed snímku je vidět oblast pokrytou šipkami ukazující přechod. Vpravo jsou zobrazeny nezpracovaná čísla v rozsahu 0 až 180°, které představují přechody. [35]



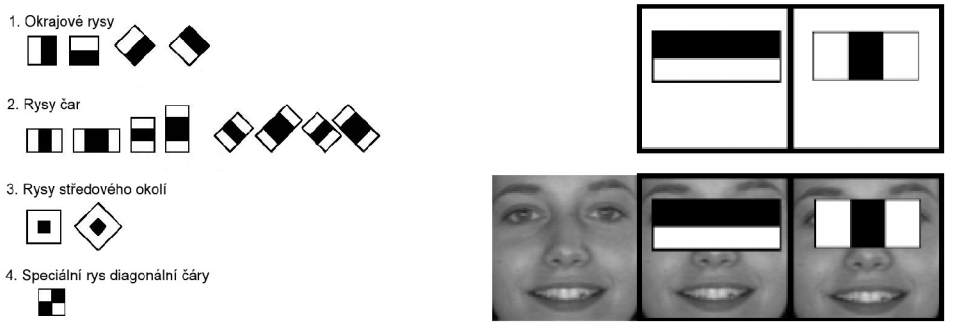
Obrázek 3.5: Princip HOG metody (zdroj: [35], překlad: autor)

3.2.4 Viola – Jones

Algoritmus Viola – Jones je založený na klasifikátorech Haar Cascades, který využívá „*hodnot intenzity pixelů a změny hodnot kontrastu mezi obdélníkovými skupinami pixelů, které jsou blízko u sebe. Rozdíl kontrastu mezi skupinami pixelů určuje bílé a černé oblasti.*“ [36] Dojde tedy k vytvoření obdélníků a odkazů na pole pro každou sekci. Haarovi rysy jsou vidět na obrázku 3.6.

Následně se pomocí **Adaboost** vybírají nejlepší funkce, ze kterých se trénují klasifikátory. Naopak ty nepotřebné se odstraní a dále s nimi nepracuje. Dojde k predikci oblastí, které jsou rozřazovány (klasifikovány) do pozitivní a negativní třídy. Negativně označené objekty jsou co nejrychleji odmítány. [37] [38]

Viola – Jones používá mnoho malých, tzv. **slabých** klasifikátorů, které se zaměřují na odlišnou část obrazu. Slabší klasifikátory jsou méně přesné a výstupem jsou false – positive detekce. Výsledky každých klasifikátorů se zkombinují a vznikne **silný** klasifikátor, který s vyšší pravděpodobností dokáže správně detekovat daný objekt.



(a) Haarovy rysy **(b)** Haarovy rysy aplikované na obličej

Obrázek 3.6: Ukázka typických rysů metody Haar včetně ukázky aplikace na obličej (zdroj: [38], překlad autor)

Metoda se používá pro detekci objektů. Vyžaduje delší dobu učení, ovšem dosahuje velmi rychlých a přesných detekcí.

3.3 Optické rozpoznávání znaků

Dalším krokem po detekovaném objektu je optické rozpoznávání znaků. Na základě této informace je možné vykonávat další činnosti, např. zvukovou signalizaci řidiči vozidla o nízkém mostu.

Optické rozpoznávání znaků (Optical Character Recognition, zkratka **OCR**) je způsob umožňující digitalizaci ručně psaných nebo tištěných textů, jako jsou fotky dopravních značek nebo faktur, článků a dalších. Existuje přes 160 jazyků pro extrakci tištěného textu a devět pro ručně psaný text. [39]

Program buď převádí obraz automaticky, nebo se musí nejprve naučit rozpoznávat znaky. Téměř vždy je nutné text podrobit korekci, docílí se tím správného překladu znaků. Pro snadné rozpoznávání byly vyvinuty strojově čitelné fonty, např. OCR-A či používanější OCR-B. Při správné detekci je možné dosáhnout 95% přesnosti. [39]

Rozpoznávání strojově tištěného textu je pouze jednou z mála možných řešení. Data lze extrahovat v různých formátech, jako je ručně tištěný text (Intelligent Character Recognition, zkratka ICR), zaškrtávací políčka (Optical mark reader, zkratka

OMR), čárové kódy atd. ICR je pokročilé optické rozpoznávání znaků, který umožňuje zpracování naučených písem včetně různých stylů rukopisu. Docílí se tím zvýšení přesnosti a úrovně rozpoznávání. K rozpoznávání fotografií, které obsahují jeden znak se běžně používá konvoluční neuronová síť (viz 2.3.4 Konvoluční neuronová síť). Pro texty libovolné délky se používá RNN (viz 2.3.3 Rekurentní neuronové sítě). [40]

Využití technologie OCR ve firmách pro sběr dat má vliv na snížení nákladů, urychlení procesů a zvýšení produktivity, snížení chybovosti a automatické zpracování obsahu dokumentů.

Tabulka 3.1: Porovnání metod OCR (zdroj: [41], překlad: autor)

Název metod	Rok založení	Online	Linux	Programovací jazyk
Tesseract OCR	1985	Ne	Ano	C/C++
Asprise OCR	1998	Ano	Ano	Java, C/C++
OCRopus	2007	Ne	Ano	Python
OCRFeeder	2009	Ne	Ano	Python
ABBYY FineReader	1989	Ano	Ne	C/C++

Tabulka 3.1 uvádí několik dostupných metod včetně informací o online užití, programovacích jazycích nebo kompatibilitu se systémem Linux. V následující kapitole bude uveden podrobnější popis některých z nich.

3.4 Dostupné knihovny pro počítačové vidění a optické rozpoznávání znaků

Mezi nejběžněji používané knihovny pro CV se řadí OpenCV, SimpleCV, Microsoft Computer Vision API, Google Cloud Vision API a další spíše komerční knihovny. Pro použití metody OCR také hraje roli dostupnost a rozšířenost. Níže jsou uvedeny některé z metod včetně stručného popisu. Je důležité se s nimi seznámit a mít přehled o dostupných a nejvíce rozšířených knihoven.

3.4.1 OpenCV

OpenCV je volně přístupná (open-source), která obsahuje přes stovky algoritmů počítačového vidění. V oblasti zpracování obrazu se jedná o velmi populární nástroj,

pomocí kterého lze zpracovávat snímky a videa, v rámci kterých je možné identifikovat objekty, rozpoznávat tváře, sledovat pohybující se objekty, vyhledávat podobné obrázky z databáze nebo sledovat pohyby očí. V současné době se uplatnění nachází v aplikacích CV v reálném čase.

OpenCV je k dispozici na platformách včetně Windows, OS X, Linux, Android a iOS, podporuje řadu programovacích jazyků, jako jsou C++, Python nebo Java. [20]

OpenCV byl založen roku 1999 ve společnosti Intel vědcem Garym Bradskim [20]. První verze této knihovny byla vydána v roce 2000, kdy se ke Garymu přidal Vadim Pisarevsky. [20] Cílem bylo řídit ruský softwarový tým Intelu. OpenCV byl v roce 2005 použit na vozidle Stanley [42], které bylo vytvořeno závodním týmem Standfordské univerzity. Téhož roku vyhrálo cenu v soutěži bez řidiče DARPA Grand Challenge. Na vývoji vozidla se pokračovalo za podpory Willow Garage s Garym Bradskim a vedoucím projektu Vadimem Pisarevskym.

Obrázek 3.7: Stanley - vozidlo bez řidiče (zdroj: [42])

3.4.2 SimpleCV

SimpleCV je jednodušší variantou k OpenCV zaměřenou na snadném užívání. Taktéž se jedná o open-source knihovnu, která byla napsána v Pythonu a lze jí nainstalovat na Linux, Windows či MacOS. SimpleCV umožňuje práci se vstupními daty a ulehčuje jejich pochopení. Používá se pro vyzkoušení prvotních projektů, a navíc nedosahuje vysokého výkonu a efektivity. Knihovna působí jako nadále nevyvíjený projekt.

3.4.3 Tesseract OCR

Tesseract OCR je open-source software, který byl vyvinut v letech 1985 až 1994 společností Hewlett-Packard [43]. V roce 2005 byl vydán jako program s otevřeným

zdrojovým kódem a od roku 2006 se vývoje ujala společnost Google. Touto metodou lze dobře zpracovávat text psaný zprava doleva. Tesseract nemá vestavěné GUI, ale je možné si některý stáhnout v rámci aplikací třetích stran. Běžným příkladem je OCRFeeder. Metoda je dostupná pro Linux, Windows nebo MacOS. [43]

3.4.4 OCRopus

Další ukázkou je OCRopus, který je bezplatným systémem pro rozpoznávání znaků a analýzu dokumentů. Cílem byla integrace v projektech digitalizace knih, jako jsou Google Books nebo knihovny. Použít se ovšem dá i pro kancelářské aplikace, nebo dokonce pro předčítání nevidomým. [44]

3.4.5 Asprise OCR

Asprise OCR je komerční knihovna určená pro OCR a rozpoznávání čárových kódů. Od roku 1997 se aktivně pokračuje ve vývoji. „*Paweł Łupkowski a Mariusz Urbanski z Univerzity Adama Mickiewicze v Poznani používají Asprise OCR verze 4 a ABBYY FineReader k rozpoznávání CAPTCHA. Adil Farooq z University of Engineering and Technology v Taxile implementoval systém rozhraní založený na řeči pro osoby se zrakovým postižením.*“ [45]

3.5 Shrnutí kapitoly

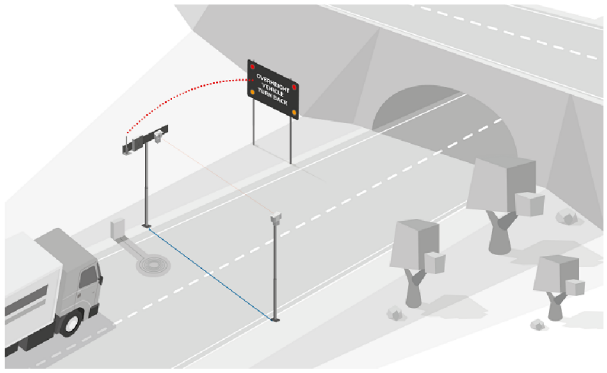
Pro řešení práce byla zvolena metoda Viola – Jones, resp. Haar klasifikátor, která společně s knihovnou OpenCV poskytuje velmi rychlou a přesnou detekci objektů v rámci počítačového vidění. Oproti hlubokému učení nevyžaduje velké množství dat a výpočetní výkon. Pro rozpoznávání znaků byl vybrán Tesseract OCR. Důvod výběru těchto metod a knihoven je popsán v následující kapitole (4 Realizace).

4 Realizace

Tato kapitola se skládá z rešeržní části, použitými zařízeními, knihovnami a hlavně samotnou realizací práce od výběru trénovacích dat, tvorby klasifikátoru, soupisu kódu, až po nahrání kódu do zařízení a vlastní montáž. V závěru se nachází samotné testování a shrnutí.

4.1 Rešerže problematiky

Na trhu existuje několik způsobů, jak kontrolovat výšku vozidel a přecházet kolizím. Všechny jsou závislé na typech senzorů, se kterými pracují. Patří sem koncový spínač (Limit Switch, zkratka LS), rezistor závislý na laseru a světle (Laser & light dependence resistor, zkratka LLDR), a kamera. První dvě jsou mechanické a optické senzory, které snímají a kontrolují specifickou výšku jedoucích vozidel. Zatímco kamera dokáže určit výšku vozidel, které přesahují výšku mostu či tunelu. Aby se předešlo nehododám, dochází ke světelné (LED tabule) nebo zvukové signalizaci. Světelná tabule umožňuje zobrazit zprávu buď o otočení vozidla nebo odklonění vozidla na jinou silnici a vyhnutí se kritickému úseku. [46]



Obrázek 4.1: Princip OVD způsobu měření (zdroj: [47])

Existuje několik firem a společností, které se těmito systémy zabývají, např. SWARCO nebo Coeval. Nevýhodou je, že by s každým, nedostatečně vysokým, mostem či

tunelem musel být nainstalován takový systém. Což z ekonomického hlediska není oproti vestavěnému systému výhodné.

Podobný princip detekce výšky mostů nebo tunelů přímo z vozidla jsem nezaznamenal. Nicméně součástí nových a moderních vozidel jsou systémy, které si např. hlídají jízdný pruh a udržují se v něm nebo dokáží měřit a dodržovat určitou vzdálenost od vozidla. Zkrátka také dokáží detekovat dopravní značky, především maximální povolenou rychlost. Tyto systémy bývají na tolik výkonné, že není problém do nich implementovat detekci dopravní značky – výška mostu.

Obrázek 4.2: Aplikace My Guard Camera (zdroj: [48])

Jednou ze zajímavostí a další alternativou, na kterou jsem narazil, je aplikace My Guard Camera [48], která promění telefon na palubní kameru a umožňuje nahrávat obraz během jízdy. Do aplikace je možné nahrát vlastní klasifikátor a detekovat vlastní předměty. Tato varianta a výstupní zařízení této práce si jsou velmi podobné, nicméně aplikace neumožňuje rozpoznávání textu. A navíc použití telefonu namísto vestavěného systému by mohl mít za následek náhlá rušení v podobě příchozích zpráv či hovorů, a docházelo k přerušení hlavní činnosti – detekce, OCR a možným nehodám.

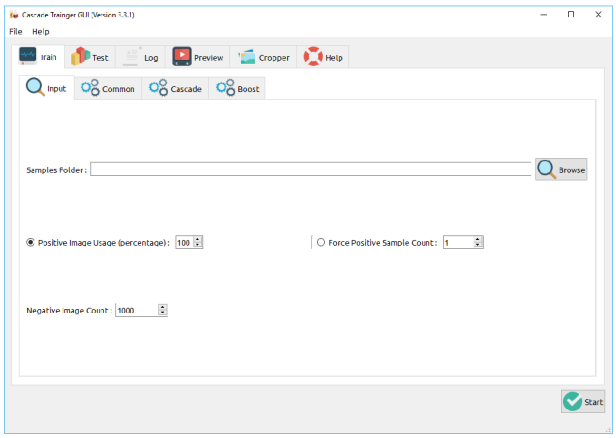
Hlavními požadacky na navrhnované zařízení je kompatibilita, uživatelsky přívětivé rozhraní (interface) a hlavně přenosnost, což je hlavní výhoda oproti stacionárně umístěným senzorům před mostem.

4.2 Použitá zařízení, knihovny a nástroje

Tato sekce sdružuje použitá zařízení, knihovny, grafická rozhraní a programovací jazyk, které byly pro realizaci použity.

4.2.1 Algoritmus Viola – Jones & Cascade Trainer GUI

Jak již bylo zmíněno, byl použit algoritmus Viola – Jones. Jedná se o starší metodu strojového učení, konkr. učení s učitelem, má ovšem značné výhody od modernějších přístupů hlubokého učení. Důvodem volby byla jednoduchost, přesnost a především rychlost detekce. Výkon zařízení Raspberry Pi byl pro zpracování obrazu a následnou detekci dostačující. Pro chod nebylo nutné velké množství trénovacích dat. Jednou z nevýhod byla doba učení modelu, která zabrala relativně dost čas. Nicméně se fáze tréninku odehrála ve fázi předzpracování (preprocessing), tedy před nahráním souborů do zařízení. S použitím funkce *detectMultiScale* v rámci knihovny OpenCV (viz 3.4.1 OpenCV) bylo možné detekovat objekty v reálném čase.



Obrázek 4.3: Ukázka grafického rozhraní pro tvorbu Haar klasifikátoru (zdroj: [49])

Snímek 4.3 reprezentuje ukázku rozhraní pro tvorbu klasifikátoru. Je zde vidět řada záložek pro trénink, testování včetně zobrazení aktuálního stavu procesu (log) nebo možnosti podpory. Konkrétně sekce učení se skládá z dalších podzáložek pro nastavení chodu činnosti.

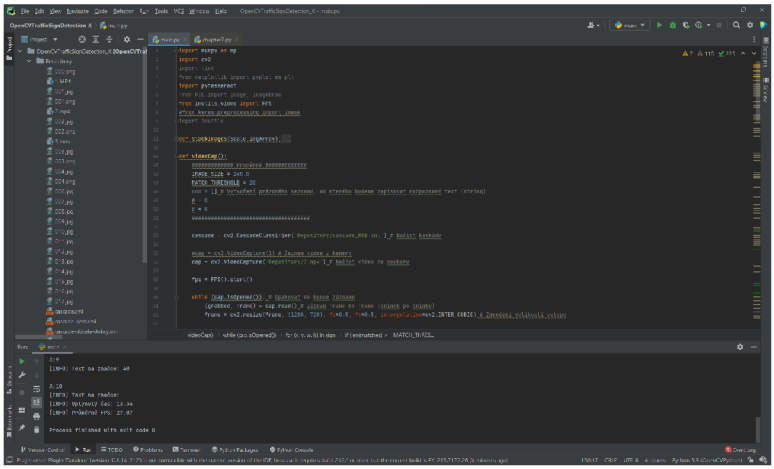
Cílem nebylo vytvoření kódu pro učení, tudíž se použil již vytvořený kód včetně grafického rozhraní Cascade Trainer GUI (dále jen GUI). Konkrétně toto rozhraní vytvořil íránský autor a vývojář Amin Ahmad a umožňuje také trénink např. HOG modelů či testování klasifikátoru na snímcích či videích. [49]

4.2.2 Nasazení OpenCV

V práci byla použita knihovna OpenCV 4.5.4.60. Je to velmi silný a populární nástroj, který je navíc rychlý a nezabere velké množství výpočetního času. Patří mezi nejoblíbenější přístupy počítačového vidění, i když se pro začínající uživatel může jevit jako složitý. OpenCV je velmi rozšířená knihovna, existuje pro ni komunita nadšenců, která vytváří návody a radí ostatním lidem, jak postupovat při chybách apod. Co víc si může člověk přát.

4.2.3 Python a PyCharm

Mezi nejpopulárnější programovací jazyky se řadí zejména Python, C/C++ a Java. Ke strojovému učení a obecně k počítačovému vidění se nejvíce používá jazyk Python, který je v porovnání s C/C++ výrazně snazší pro užívání. Python může být zkompilován do bajtového kódu pro vytváření velkých aplikací. Poskytuje také vysoké dynamické datové typy a podporuje dynamickou kontrolu typu. Lze jej snad integrovat s C, C++, COM, ActiveX, CORBA a Java. Jelikož je Raspberry Pi založené na linuxovém jádře, byla volba právě jazyku Python na místě. Nedocházelo ke komplikacím s kompatibilitou jazyků.



Obrázek 4.4: Grafické prostředí PyCharm dostupné pro Windows

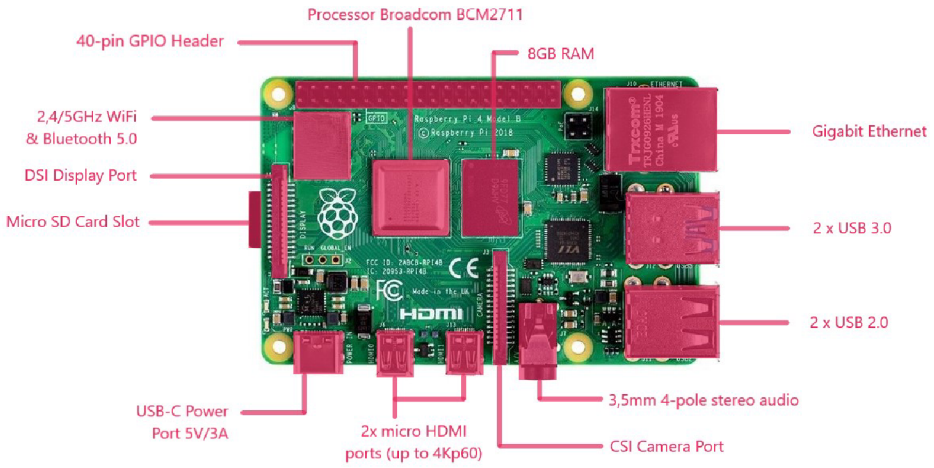
V souvislosti s tvorbou kódu existuje celá řada programovacích prostředí, která jsou na trhu dostupná. Patří sem např. Visual Studio Code, Atom, Microsoft Visual Studio, PyCharm, Anaconda a spousta obdobných programů. Jedná se pouze o nástroj, který např. zvýrazňuje klíčové funkce kódu pro přehlednost, umožňuje odladění chyb programu. Zkrátka se tyto programy používají pro lepší kontrolu nad programová-

ním. Použití softwaru PyCharm (konkr. verze Professional 2021.3) v této práci nemá žádný vliv na fungování výsledného kódu.

Na obrázku 4.4 je program PyCharm dostupný pro Windows. Jak vypadá výchozí prostředí systému Linux, které je implementované v operačním systému, je uvedeno v příloze K.

4.2.4 Nasazení Raspberry Pi

Raspberry Pi je jednodeskový počítač o velikosti zhruba platební karty, kde hlavním operačním systémem je Raspbian (Linux). Výkonem se dá srovnat se (slabším) stolním počítačem. Obsahuje vstupy pro monitor (HDMI), USB porty, zařízení se dá připojit k bezdrátové (Wi-Fi) či drátové (Ethernet) síti. S ohledem na výkon, množství RAM a zamýšleného použití bylo vyvinuto několik generací tohoto počítače. Mikroprocesor z rodiny ARM, jenž je na desce osazený, je srovnatelný s běžným chytrým telefonem. Na desce lze také nalézt řadu GPIO pinů a vstupů pro připojení kamery či displeje. [53]



Obrázek 4.5: Počítač Raspberry Pi model 4B (zdroj: [54])

Raspberry Pi je oproti platformě Arduino možné použít k vývoji aplikací, užít se dá jako multimediální stanice pro procházení souborů nebo přehrávání hudby a videí a s trochou zručnosti si lze postavit vlastní NAS server. Pro desky byla vytvořena řada rozšiřujících desek. Společně s nimi mohou najít uplatnění při řízení DC motoru, komunikaci po RS232 nebo pomocí RF modulů, nebo pouze jako záložní zdroj napájení. [53]

V práci byl použit počítač Raspberry Pi a byl vybrán z několika důvodů. K provozu, tedy k detekci objektů a rozpoznávání textu, se vyžaduje větší výpočetní výkon než u jiných desek. Byla proto použita verze s 8 GB RAM pro dosažení lepších výsledků. Na druhou stranu bylo nezbytné poskytnout nucené chlazení v kombinaci s přirozeným, aby nedošlo k přehřátí a tím ke snížení účinnosti. Také se tím zvýšila životnost zařízení. V tabulce 4.1 jsou vidět teploty při práci zařízení.

Tabulka 4.1: Porovnání hodnot teploty s, nebo bez chlazení (zdroj: [55], překlad: autor)

Typ práce	N/A [°C]	Pouze chladič [°C]	Chladič + ventilátor [°C]
Nečinný	38	37	35
Psaní kódu (konzole)	45	40	35
Plocha	62	52	43
Python skript	55	47	38

Mezi další, poněkud méně známé, platformy se řadí např. Nucleo, Odroid, Banana Pi nebo Intel Edison.

4.2.5 Nasazení Tesseract OCR

Hlavním kritériem při výběru OCR metody bylo především online užití. Musel být zvolen software, který ke svému procesu nevyžaduje připojení k internetu a běží na systému Linux. Pokrytí internetu není všude dostupné a není dobré se na to spoléhat. V kapitole 4.2.4 Nasazení Raspberry Pi bylo popsáno zařízení, které běží na systému Linux. Bylo tedy nutné, aby byly spolu kompatibilní. V neposlední řadě byl programovací jazyk – Python.

Metoda je ideální pro skenování textů na bílém pozadí, jako jsou knihy, dokumenty nebo dopravní značky. Protože je OCR založené na hlubokém učení, je možné dosáhnout vysoké přesnosti a variability písem. S přesným rozpoznáváním jsou spjaté vysoké nároky na výkon použitého zařízení. Pro realizaci byla použita verze pytesseract 0.3.9.

4.2.6 Kamera a objektiv

Pro získávání obrazu byla použita HQ kamera (viz obrázek 4.6a), která nabízí rozlišení 12 Mpx a díky většímu čipu má vyšší citlivost a umožňuje dosáhnout lepších výsledků za šera. Ke kameře byl připojen objektiv s bajonetem typu CS (viz obrázek

4.6b), který nabízí vysokou světelnost f/1.2 a ohniskovou vzdálenost 6 mm. mbox63° zorné pole plní roli širokoúhlého objektivu a umožňuje zachytit větší počet objektů. Tento objektiv má manuální ostření, tím pádem oproti kamerám s automatickým ostřením nedochází k přeostřování a vzniku neostrostí.

(a) Kamera (zdroj: [56])

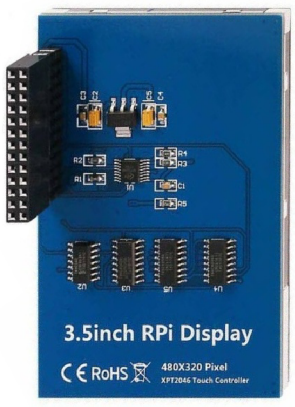
(b) Objektiv (zdroj: [57])

Obrázek 4.6: HQ kamera s CS objektivem

4.2.7 Použitý displej

Pro zadání výšky vozidla a vypisování informací byl použit 3.5” TFT displej s rozlišením 320x480 px a odporovým dotykovým ovládáním (obr. 4.7). Komunikace probíhá skrz sériové SPI (Serial Peripheral Interface) rozhraní a umožňuje přímou komunikaci s procesorem. Displej je kompatibilní s Raspberry Pi 4 pro modely A a B.

(a) Přední strana displeje



(b) Zadní strana displeje

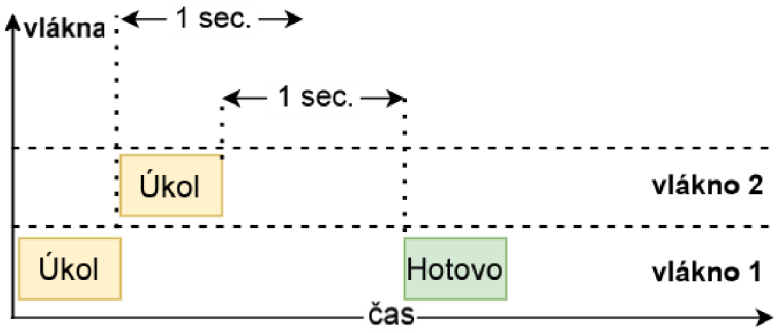
Obrázek 4.7: 3.5” TFT dotykový displej

Kamera je připojena pomocí 15 pinového konektoru k Raspberry Pi desce, díky čemuž umožňuje přímou komunikaci s GPU. Nedochází tedy k vytěžování CPU, což má za následek zvýšení výpočetního výkonu pro jinné činnosti.

4.2.8 Multithreading

Vlákno (thread) je samostatný tok operací. Vláknování (threading) umožňuje souběžné vykonávání více operací. Jedna a ta samá činnost může být spuštěna na více vláknem současně. Soubor *main.py* běží na jednom samostaném vláknu. Princip vláknování je znázorněno na obrázku 4.8.

Naneštěstí dochází k problému, kdy se více vláken snaží získat přístup ke stejným proměnným současně. Do proměnné by se uložila hodnota z posledního aktivního vlákna, tudíž by se přepisovali hodnoty zapsané jinými vlákny. K tomu se využívá funkce *Lock()*, díky které dojde k uzamčení sdílené proměnné. [58]



Obrázek 4.8: Multithreading – princip (zdroj: [58], překlad: autor)

Počet vláken je závislý na počtu jader procesoru zařízení. Jelikož má Raspberry Pi čtyři fyzická jádra, je možné vytvořit jedno vlákno na jedno jádro, tedy čtyři vlákna celkem. Jelikož hlavní program pracuje na samostatném vláknu, je možné využít už jen tři.

4.3 Příprava klasifikátoru a tvorba kódu

V této kapitole je popsán postup učení Haar klasifikátoru a tvorba zdrojového kódu včetně popisu klíčových částí.

4.3.1 Tvorba klasifikátoru Haar

Prvním a také klíčovým bodem realizace byla tvorba Haar klasifikátoru pro detekci objektů. Při výběru a shromažďování fotografií bylo důležité dávat si pozor na správně zvolené negativní snímky. Pokud cílem práce bylo rozpoznávání konkrétní dopravní značky, nebylo vhodné pořizovat, resp. shromažďovat fotky budov, zvířat, lidí, aut apod. Výsledkem detekce by byly spousty jiných objektů než požadovaná značka – běžně označovaná jako false – positive detekce.

(a) Pozitivní snímky **(b)** Negativní snímky

Obrázek 4.9: Ukázky pozitivních a negativních trénovacích dat pro Haar klasifikátor

Pro pozitivní kategorii – druh objektů, jenž bylo cílem detekovat, byly nasbírány snímky dopravní značky B16, tedy výšky mostu. Ostatní byly uloženy do negativní třídy. Na obrázku 4.9a je vidět soubor pozitivních snímků, zatímco na obrázku 4.9b fotek negativních.

Po seskupení „správných“ snímků se v GUI muselo nastavit několik parametrů pro správný chod učení např. počet negativních snímků ve složce, počet fází (iterací), hodnoty výšky a šířky pozitivních snímků, počet přiřazených RAM atd. V tabulce 4.3 je možné vidět nastavené hodnoty v rámci GUI. V přílohách A až D jsou uvedeny originální snímky s nastavenými hodnotami parametrů.

Nutno podotknout, že se také testovala velikost pozitivních snímků. Bylo vytvořeno několik variant velikostí, kde u příliš velikého rozlišení byla detekce pomalá ovšem na úkor kvality. Naopak u malého rozlišení byla detekce rychlá, ale nedocházelo k detekci všech dopravních značek na obrázku či videu. Nejideálnější nalezenou variantou bylo rozlišení 120px.

Tabulka 4.2: Srovnávací tabulka barevných a černobílých trénovacích vzorů

Typ obrazu	Snímky		Video	
	Počet správných detekcí	Počet nesprávných detekcí	Počet správných detekcí	Počet nesprávných detekcí
Barevný	18/18	0/18	19	0
Černobílý	17/18	0/18	20	2

Důležitou roli v učení zastupoval barevný profil. Z tabulky 4.2 je patrné, že pro černobílé trénovací vzory se v rámci detekce objektů na snímcích nasbíralo 17 správných z 18 testovaných, zatímco pro RGB snímky byla úspěšnost 100 %. Pro detekci ze záznamu bylo pro černobílé vzory správně nalezeno 20 objektů a další 2 nesprávné, kdežto při RGB 19 správných a žádné nesprávné. Je tedy zřejmé, že na trénování měl vliv i odstín barev.

Tabulka 4.3: Tabulka s nastavenými hodnotami parametrů pro trénink klasifikátoru

Záložka: Vstup (Input)	
Nastavovaný parametr	**Hodnota parametru**
Využití pozitivních snímků	100 %
Počet negativních snímků	280
Záložka: Obecné (Common)	
Nastavovaný parametr	**Hodnota parametru**
Počet fází	20
Vyhrazený počet RAM	4096 MB
Počet vláken	5
Záložka: Kaskáda (Cascade)	
Nastavovaný parametr	**Hodnota parametru**
Šířka vzorků	24 px
Výška vzorků	24 px
Typ funkce	HAAR
Trénovací proces	ALL - využívá svislých vzorů a jejich 45° natočení
Záložka: Boost	
Nastavovaný parametr	**Hodnota parametru**
Adaboost - typ	GAB - výchozí nastavení

Po přípravě následovala časově nejnáročnější fáze, a to učení. Natrénovat klasifikátor pro 180 pozitivních a 280 negativních vzorů trvalo cca 26 min. Záleželo na použitém počtu snímků, velikostí a metodách vstupních dat.

4.3.2 Popis zdrojového kódu

Další částí práce byla detekce objektů, se kterou byla spjatá tvorba kódu. Hned ze začátku bylo nutné si ujasnit, zda program psát na počítači se systémem Windows nebo přímo v Raspberry Pi. Důvodem volby PC s Windows bylo snadné testování a rychlost zpracování obrazu. Ve fázi realizace pouze stačilo odladit drobné nedostatky a přenést soubory do zařízení.

V prvé řadě se musel vytvořit soubor *main.py*, stáhnout a nadefinovat klíčové knihovny, které byly nezbytné pro volání funkcí kódu. Použité knihovny včetně jejich popisu jsou sepsány v tabulce 4.4.

Tabulka 4.4: Použité knihovny včetně jejich popisů

Název knihovny	Název knihovny pro Raspberry Pi	Popis knihovny
OpenCV	opencv-contrib-python (4.5.4.60)	Nástroj pro CV a real-time detekci
Numerical Python	numpy (1.19.5)	Python knihovna pro práci s poli (arrays) a maticemi
Imutils	imutils (0.5.4)	Nástroj pro měření FPS
Tesseract OCR	pytesseract (0.3.9)	Knihovna pro rozpoznávání znaků
RegExr	regex (2022.4.24)	Nástroj pro práci s regresivními výrazy (regular expression)
Threading	thread6 (0.2.0)	Nástroj pro práci s více nezávislými vlákny
Tkinter	tkinter (8.6)	Knihovna pro tvorbu GUI

Následovalo definování proměnných, cesty k vytvořenému klasifikátoru a způsob přijímání vstupního obrazu. V rámci optimalizace a zefektivnění chodu se před samotnou smyčkou spustilo hledání klíčových bodů vzorového (template) snímku vytvoření deskriptoru. Následně se vstupní obraz zmenšil na velikost HD, tedy 1080 x 720 px. Důvod a výsledky rozlišení je popsán v kapitole 4.3.3 Vliv parametrů na detekci.

O něco důležitější částí kódu je detekce objektů v záběru za použití *detectMultiScale*. Funkce přijímá několik vstupních parametrů, které jsou popsány v kódu 4.1. Důvod výběru hodnot parametrů je popsán v kapitole 4.3.3 Vliv parametrů na detekci. Ukázka definování proměnných ad. je zřejmá z kódu 4.1.

```
1  # Promenne
2  global odd
3  global oddOut
4  global threads
5  global sound_alert
6  IMAGE_SIZE = 240.0
7  MATCH_THRESHOLD = 20
8  A = 0
9  B = 0
10 cascade = cv2.CascadeClassifier('cascade_RGB.xml') # Nacist Haar kaskadu
11 cap = cv2.VideoCapture('/dev/video0') # Zaznam videa z kamery
12 fps = FPS().start()     # Start FPS pocitadla
13 # ORB a BFMatcher inicializace
14 orb = cv2.ORB_create(nfeatures=800, scaleFactor = 1.2, scoreType = cv2.ORB_FAST_SCORE)
15 # Brute-Force Matcher - porovnavac deskriptoru, ktery srovnava dve sady deskriptoru bodu a
       generuje vysledek ve forme seznamu shod
16 bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)
17 roadsign = cv2.imread('002.png', 0)     # Nacteni porovnavaciho snimku
18 kp1, des1 = orb.detectAndCompute(roadsign, None)    # Najit klicove body a vypocitat
       deskriptory
19 sign = cascade.detectMultiScale( # detekce objektu ruznych velikosti zjistene objekty jsou
       vraceny v seznamu obdelníku
20   frame,  # Vstupni obraz
21   scaleFactor=1.2,  # Meritko urcujici o kolik se zmensi velikost obrazu
22   minNeighbors=3,  # Pocet sousedu, ktery by mel mit kazdy kandidatsky obdelnik
23   minSize=(30, 30))  # Minimalni velikost objektu; mensi objekty jsou ignorovany
```

Zdrojový kód 4.1: Import knihoven, definování proměnných a tvorba deskriptorů

Při detekci dopravní značky se spustí smyčka *for* – 1. řádek kódu 4.2, kde vstupní proměnnou je *sign*. Ta obsahuje informace o pozicích a velikostech detekovaných objektů. Následně musí být provedena filtrace chybných kandidátů pomocí ORB. Principem je hledání a porovnávání klíčových bodů detekovaných značek se vzorovým (template) snímkem. Podle počtu nalezených bodů a hodnoty prahové hodnoty

MATCH_THRESHOLD dojde k ořezu a zaměření se pouze na text – řádky 10 - 12.

Následně je nutné provést převod do odstínů šedi a následně aplikovat funkci *adaptiveThreshold*, která automaticky převede oříznutý objekt do binární podoby (bílá a černá, 0 a 1). Na takto upravený snímek je možné již aplikovat OCR. Bohužel samotná rychlost zpracování znaků je pomalejší oproti detekci objektů. Dostupným řešením je multithreading, který byl popsán v kapitole 4.2.8 Multithreading. Vytvořením nezávislé činnosti se proces urychlí a bude probíhat nezávisle na hlavním kódu.

Definuje se vlákno s odkazem na tzv. definici funkce (function definition), která bude provádět nezávislou činnost. Funkce přijímá dvě vstupní proměnné, kde *object* obsahuje aktuální detekovaný objekt a *lock* definuje proces zámku. Následně se spustí vlákno, a s ním i proces rozpoznávání znaků (řádky 17 - 20), který je umístěn až po detekci dopravní značky. Je nelogické aplikovat OCR na každý snímek. Docházelo by k rozpoznávání znaků na celém obrazu, a tím výraznému snížení rychlosti zpracování a zpomalení systému.

```
1  for (x, y, w, h) in sign:       # prochazet vsemi detekovanymi objekty
2    obj = frame[y:y + h, x:x + w] # ziskat objekt ze snimku ulice
3    ratio = IMAGE_SIZE / obj.shape[1]
4    obj = cv2.resize(obj, (int(IMAGE_SIZE), int(obj.shape[0] * ratio)))
5    kp2, des2 = orb.detectAndCompute(obj, None)   # najit klicove body a deskriptory pro
         objekt
6    if len(kp2) == 0 or len(des2) == None: continue
7    matches = bf.match(des1, des2)        # vytvorit deskriptor shod
8    matches = sorted(matches, key=lambda x:x.distance)    # seradit shody podle vzdalenosti
9    if (len(matches) >= MATCH_THRESHOLD): # je-li splnena prahova hodnota...
10     obj = cv2.resize(obj, (240, 240))
11     w, h, _ = obj.shape
12     obj = obj[int(w / 4):int(w * 3 / 4), int(h / 4):int(h * 3 / 4)]     # ... oriznuti a
           zamereni se pouze na text
13     obj = cv2.cvtColor(obj, cv2.COLOR_RGB2GRAY) # Prevedeni do odstinu sedi
14     obj = cv2.adaptiveThreshold(obj, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY,
           199, 5)    # Automaticky urci prahovou hodnotu na zaklade intenzity pixelu
15     A = A + 1
16     if (A > 1): # Eliminace nahodneho sumu (detekci)
17         if (threads > 0 and threads < 4 and B <= 120):  # Definovani max. mozneho poctu
               aktivnich vlaken
```

```
18          t = threading.Thread(target=ocr, args=(lock, obj))      # Vytvoreni cilove definice
                a vstupnich promennych
19          t.start()       # Spusteni vlakna
20          threads = threads + 1
```

Zdrojový kód 4.2: Vytvoření deskriptoru a klíčových bodů detekovaného objektu

V kódu 4.3 na 1. řádku vstupuje oříznutý objekt – *obj* do procesu OCR. Veškeré nastavení je uloženo v proměnné *config*, kde *--oem 1*- definuje režim (engine) rozpoznávání. Číslice 1 využívá LSTM neuronovou síť (viz 2.3.3 Rekurentní neuronové sítě). Další argument *--psm 8*, definuje rozdělení snímku na řádky textu nebo jen slova. Konkrétně číslice 8 rozděluje obraz na slova. Posledním argumentem je filtrace nebo-li omezení znaků, které se mají rozpoznávat. Je patrné, že se na seznamu povolených znaků jsou pouze číslice od 0 do 9. Znaky „.,-:“ jsou kvůli velikosti obtížně čitelné a nemusely by být detekovány. Je lepší zajistit následné vyhodnocení než se spoléhat na malou přesnost těchto znaků. Zároveň nebude docházet k rozpoznávání těch chybných.

Následně dojde k vyvolání funkce *image_to_string* pro zahájení převodu znaků. Výstupní data se zapisují do proměnné *data*, na které je aplikován filtr. Pomocí regresivních výrazů (řádek 6) dojde k odstranění veškerých mezer, odsazení, zalomení apod. Pokud by se znaky nerozpoznaly nebo by číslice byla trojciferná nebo vyššího řádu, dojde k jejich ignorování. Dále dojde k uzamčení a zápisu hodnoty do listu, ze které se provede četnost hodnot. Proces se odemkne a ukončí, čímž se uvolní vlákno pro další detekovaný objekt (řádky 11 - 15).

```
1  def ocr (lock, obj):
2      global oddOut       # Nactení globalnich promennych
3      global odd
4      global threads
5      global hlimit
6      global sound_alert
7      global buzzer
8      config = r'--oem 1 --psm 8 -c tessedit_char_whitelist=0123456789'  # nastaveni a
           definovani OCR pouze na cisla
9      data = pytesseract.image_to_string(obj, lang='eng', config=config)  # rozpoznávání textu
            z obrázku
```

```
10      match = re.sub(r'\s', r'', data)  # vrati retezec bez nalezenych znaku '\s'
11      if match:
12          if (len(match) > 0 and len(match) < 3):  # eliminace trojcisli a vyssich
13              if (int(match) < 10):
14                  match = int(match) * 10
15               lock.acquire()  # zamceni procesu
16              odd.append(str(match))  # zapis do listu
17              oddOut = max(set(odd), key=odd.count)
18               threads = threads - 1
19              if (hlimit >= int(oddOut) and len(odd) > 2 and sound_alert == False):
20                sound_alert = True # Prepsani hodnoty pro spusteni zvukove signalizace
21                GPIO.output(buzzer, sound_alert)   # Spusteni bzucaku
22              lock.release()  # odemceni procesu
```

Zdrojový kód 4.3: Definice funkce pro rozpoznávání znaků

Poslední záložkou při tvorbě skriptu je vytvoření grafického rozhraní, které se spustí při zapnutí Raspberry Pi. Nejdříve se inicializuje název a velikost okna. Dále dojde k rozmístění textu, tlačítek a vstupního pole, nastavení barev pozadí, velikosti textu ad. Nakonec se program spustí.

Uživatel pomocí tlačítek *Increase* a *Decrease* zvýší či sníží výšku svého vozidla. Se stisknutím *Start* dojde ke spuštění hlavní programu a zápisu hodnoty do proměnné. Tímto parametrem se ověřuje, zda vozidlo nepřekročilo výšku mostu. Ukázka kódu 4.4 vysvětluje princip vytvoření takového rozhraní.

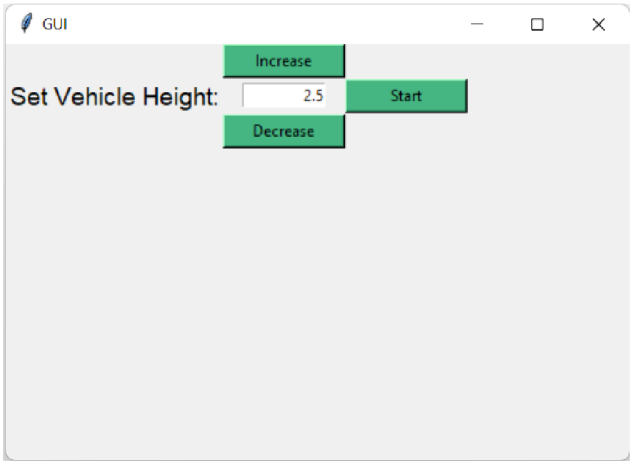
```
1  window = Tk()
2  window.title("GUI")  # Nazev okna
3  window.geometry('480x320')  # Velikost okna
4  window.attributes('-fullscreen', True)  # Roztahne okno na celou plochu displeje
5  def guiDesign():
6      SetLabel = Label(window, text="Set Vehicle Height:", font=("Arial", 14))  # Vlozeni
           textu
7      SetLabel.grid(column=0, row=2)  # Pozice textu
8      global inputField
9      inputField = Entry(window, width=10, justify='right')  # Definovani vstupniho pole
10     inputField.insert(1, "2.5") # Nastaveni vychozi hodnoty
```

```
11      inputField.grid(column=1, row=2)  # Pozice vstupniho pole
12      UpBtn = Button(window, text="Increase", bg="#46B382", width=12, command=incHight)  #
            Vytvoreni tlacitka UpBtn
13      UpBtn.grid(column=1, row=1) # Umisteni tlacitka UpBtn - 1. sloupec, 1 radek
14      DownBtn = Button(window, text="Decrease", bg="#46B382", width=12, command=decHight) #
            Vytvoreni tlacitka DownBtn
15      DownBtn.grid(column=1, row=3)  # Umisteni tlacitka UpBTN - 1. sloupec, 3. řádek
16      StartBtn = Button(window, text="Start", bg="#46B382", width=12, command=videoCap)  #
            Vytvoreni tlacitka StartBtn
17      StartBtn.grid(column=2, row=2)  # Umisteni tlacitka StartBtn
18  guiDesign()
19  window.mainloop()  # Spusteni GUI
```

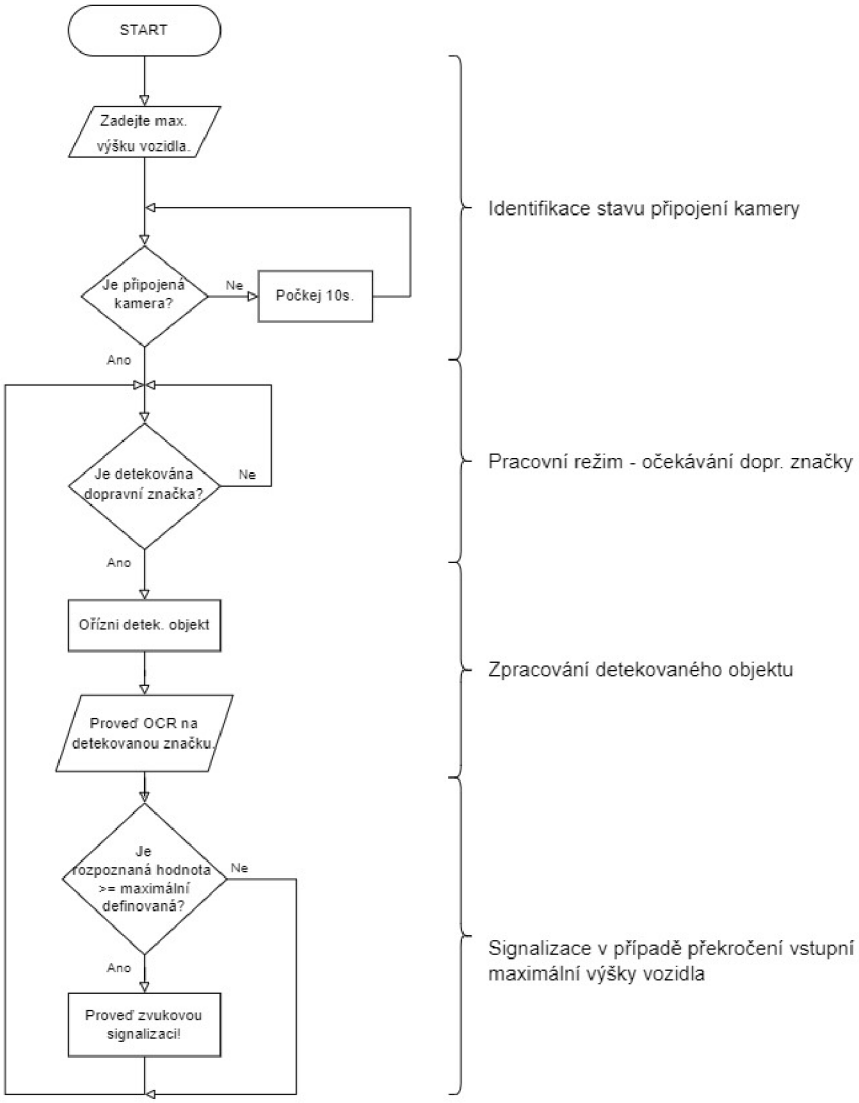
Zdrojový kód 4.4: Vytvoření grafického prostředí

Části zdrojových kódů jsou pouze pro ukázku a neobsahují úplné formátování (zarovnání) a další nezmíněné řádky kódu. V příloze X této práce je uveden úplný zdrojový kód. Pro přiblížení funce kódu byl vytvořen vývojový diagram 4.11.

Snímek 4.10 reprezentuje celkový pohled na vytvořené okno grafického rozhraní. Při návrhu byl kladen důraz na kompatibilitu s dotykovým displejem. Jedná se o rozpracovaný návrh. GUI je plně funkční, na pohled nepůsobí rušivě a na přímém slunci je vše čitelné. Jsou zde vidět 3 tlačítka, text a vstupní buňka.



Obrázek 4.10: Okno vytvořeného grafického rozhraní



Obrázek 4.11: Vývojový diagram fungování kódu

4.3.3 Vliv parametrů na detekci

Mezi klíčové parametry funkce *detectMultiScale*, které hrají významnou roli při false – positive detekci, se řadí *minNeighbors* a *scaleFactor*. Bylo provedeno několik měření, na jejichž základě byla vytvořena tabulka. Kvůli její velikosti musela být rozdělena do dvou – tabulka 4.5 a tabulka 4.6.

Z dat vyplývá optimální a zvolené nastavení těchto dvou parametrů. Z rozsahu hodnot byla zvolena střední cesta, tedy hodnoty 1.2 pro *scaleFactor* a 3 pro *minNeighbors*. Vstupní velikost obrazu se podle počtu správně detekovaných objektů a průměrných hodnot FPS změnšil (škáloval) z formátu Full HD na HD rozliše-

ní. Docílilo se tím zvýšení FPS na úkor relativně vysoké přesnosti detekce. Nutno podotknout, že se testovalo na počítači se systémem Windows a hodnoty nebudou identické s použitým zařízením (Raspberry Pi). Navíc se testovalo na video záznamu, tudíž se hodnoty FPS budou lišit pro živý záznam z kamery.

Tabulka 4.5: Porovnání hodnot parametrů funkce detectMultiScale včetně průměrných hodnot FPS - 1. část

detectMultiScale		OK | Všechny detek. objekty		Průměrné hodnoty FPS	
scaleFactor	minNeighbors	HD	Full HD	HD + detekce	Full HD + detekce
1,10	5	10 | 10	17 | 17	28,29	16,09
1,10	3	21 | 22	49 | 50	28,43	16,50
1,20	5	1 | 1	0 | 0	46,73	29,02
1,20	3	10 | 10	15 | 16	47,10	28,86
1,30	5	0 | 0	0 | 0	50,91	32,96
1,30	3	4 | 4	3 | 3	48,20	32,79
1,40	5	0 | 0	0 | 0	58,93	40,81
1,40	3	3 | 3	2 | 2	57,11	40,54
1,10 až 1,30	3	Optimální nastavení			
1,20	3	Zvolené hodnoty			

Tabulka 4.6: Porovnání hodnot parametrů funkce detectMultiScale včetně průměrných hodnot FPS - 2. část

detectMultiScale		OK | Všechny detek. objekty		Průměrné hodnoty FPS	
scaleFactor	minNeighbors	HD	Full HD	HD + OCR	Full HD + OCR
1,10	5	10 | 10	17 | 17	27,75	16,71
1,10	3	21 | 22	49 | 50	27,76	15,55
1,20	5	1 | 1	0 | 0	44,05	28,47
1,20	3	10 | 10	15 | 16	44,53	27,42
1,30	5	0 | 0	0 | 0	49,31	32,60
1,30	3	4 | 4	3 | 3	48,58	31,88
1,40	5	0 | 0	0 | 0	57,09	40,62
1,40	3	3 | 3	2 | 2	57,05	40,05
1,10 až 1,30	3	Optimální nastavení			
1,20	3	Zvolené hodnoty			

4.3.4 Konstrukce a montáž zařízení

Po sepsání kódu a odladění hodnot parametrů následovala implementace a realizace zařízení. Nejdříve se nainstaloval nejnovější operační systém – Raspberry Pi OS 64-bit (2022-04-04). K tomu se využilo prostředí nainstalované v systému Windows, jenž výrobce poskytuje. OS se nahrál na SD kartu, která se vložila do Raspberry Pi.

K zařízení, konkr. na procesor, se připevnil chladič a připojil 5V ventilátor (KR 30 mm) pro lepší odvod teplého vzduchu (viz 4.2.4 Nasazení Raspberry Pi). Ovšem namísto 5V byl připojen k 3,3V, čímž se snížila jeho hlučnost, spotřeba, ale také účinnost. Pro ladění a komunikaci se zařízením se připojily periferie – klávesnice a myš. Kamera se připojila skrz sériové rozhraní CSI (Camera Serial Interface) a displej se zapojil do pinů na Raspberry Pi jako rozšiřující deska. Lepší pohled na připojené komponenty je možné si prohlédnout z přílohy E.

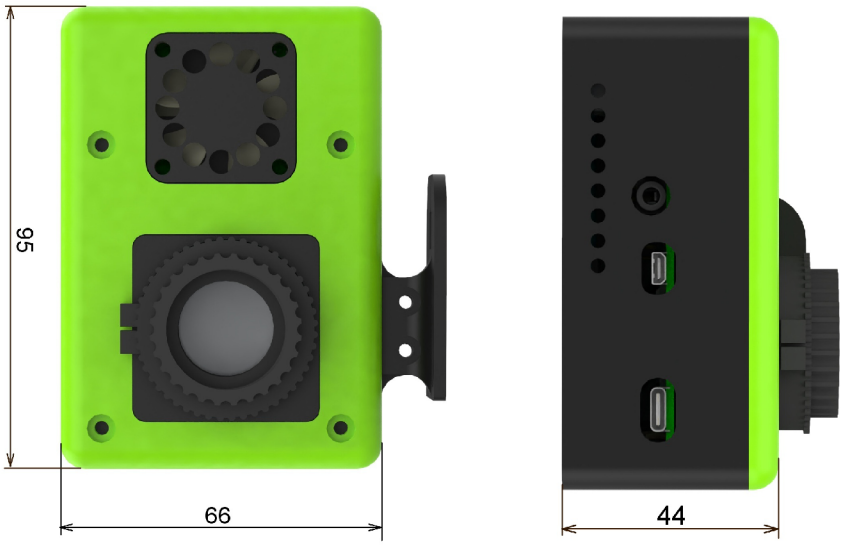
Po spuštění zařízení se systém z karty nainstaloval. V nastavení bylo nutné povolit komunikaci s kamerou, provést aktualizace interních knihoven a programovacího jazyka Python. Následovala stejná operace jako při tvorbě kódu, a to stažení použitých knihoven, jejichž seznam byl uveden v tabulce 4.4. Vytvořený kód se ze systému Windows přesunul do zařízení.

Deska s připojenými komponenty musela být ucelena a seskupena do kompaktního celku, aby nepůsobila rušivě a nemohlo dojít k závadě na zařízení. Navrhl se tedy model pouzdra, který by zakryl veškeré kabely, piny, porty a zároveň aby se dala připevnit kamera a ventilátor. Celá konstrukce se skládá ze tří hlavních dílů a umožňuje demontáž. Jelikož byl k zařízení připojen displej, nebylo nutné využívat jiný než napájecí USB-C konektor. Ovšem pro případ budoucích a náhlých situací, které by vyžadovaly ladění, se nechaly odkryté některé konektory – 3,5mm jack, micro HDMI a USB-C.

Velký důraz byl kladen na chlazení a způsob odvodu vzduchu. Ventilátor byl zapojen, aby sál vzduch z vnitřního prostoru a odváděl ho ven, podobně jako je tomu v PC skříních. Z tohoto důvodu byly v pouzdru vytvořeny průduchy.

Nedílnou součástí byl způsob uchycení zařízení, kamery a ventilátoru vůči pouzdru. Byly vytvořeny distanční sloupce pro vymezení polohy. Model a rámcové rozměry sestavy zařízení jsou patrné na snímku 4.12. Další ukázky jsou v přílohách F až H. Z důvodu umístění systému primárně za čelní sklo vozidla, byl pro 3D tisk vybrán materiál PET-G. Jedná se o plast vysoce odolný proti nárazům, je méně křehký,

má lepší mechanické a tepelné vlastnosti než ABS, je zdravotně nezávadný a lze jej recyklovat. Na druhou stranu ho lze snadno poškrábat a při tisku vyžaduje vyšší teplotu trysky oproti materiálům PLA nebo ABS.

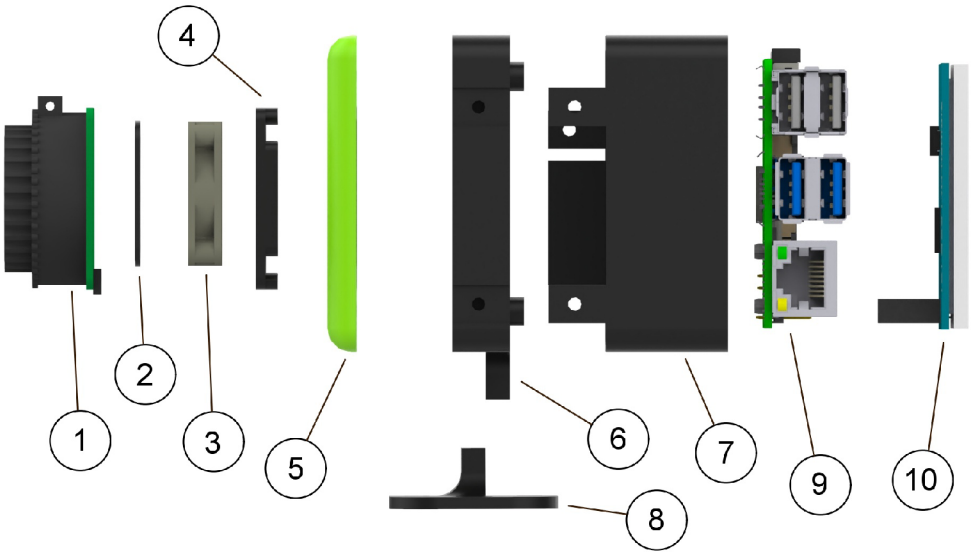


Obrázek 4.12: Model zařízení včetně základních rozměrů

Jelikož se pouzdro tisknulo na FDM 3D tiskárně, bylo nutné zajistit, aby veškeré díly splňovaly požadavky této metody. Důsledkem vrstvení materiálu se musel brát zřetel na samotný návrh, aby u funkčních částí nedocházelo v průběhu montáže k porušení. Další klíčovou záležitostí bylo zvolení ideálně velkého přídavku, resp. vůlí pro díry, zahloubené díry pro zajištění matic apod. Konkrétně se zvolila hodnota 0,4 mm. V tabulce 4.7 jsou uvedeny nejdůležitější z nich včetně jejich hodnot.

Tabulka 4.7: Nastavení parametrů pro 3D tisk

Název parametru	Hodnota parametru	Význam parametru
Hustota výplně	20 %	„Plnost“ vnitřku součásti
Vzor výplně	mřížka (grid)	Struktura a tvar výplně dílu
Teplota trysky	240 °C	Teplota nahřívání materiálu
Teplota podložky	65 °C	Teplota vyhřívané podložky
Podpůrné struktury	Ne	Generování podpor pro převislé konstrukce



Obrázek 4.13: Průřez použitými díly zařízení

Všechny díly navrženého zařízení, kromě šroubů, distančních sloupců, matic ad., jsou patrné ze snímku 4.13. Názvy položek jednotlivých pozic jsou vypsány v tabulce 4.8. Vše se nakonec smontovalo dohromady, čímž vzniklo ucelené a estetické zařízení, které se připevnilo k držáku s přísavkou na sklo. Následovalo testování systému.

Tabulka 4.8: Vysvětlivka pozic pro snímek 4.13

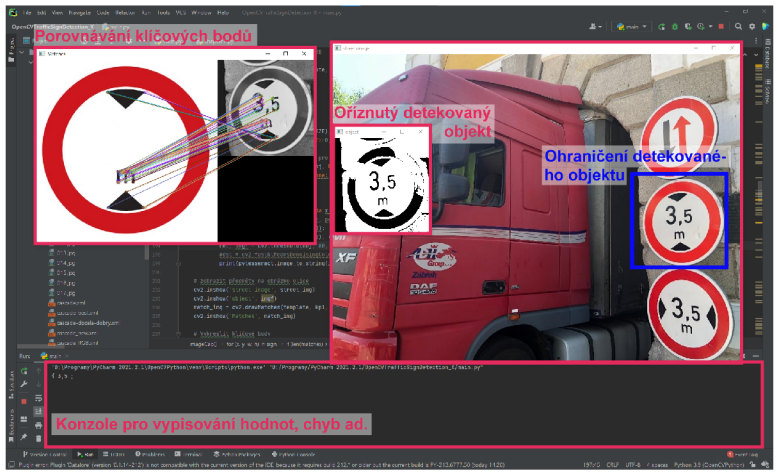
Číslo pozice	Název dílu	Číslo pozice	Název dílu
1	Kamera	6	Hlavní díl - propojka
2	Mřížka ventilátoru	7	Kryt desky a čelní díl displeje
3	Ventilátor	8	Základna (mount) pro připevnění
4	Krytka kamery	9	Raspberry Pi 4B
5	Čelo - pohledová strana	10	Dotykový displej

4.4 Testování a ladění

Testování funkčnosti detekce, rozpoznávání znaků na značce, důležitých částí kódu jakožto kontrola připojené kamery, vláknování ad. probíhalo nejprve na PC se systémem Windows. Kód nahraný do Raspberry Pi byl téměř identický. V rámci

optimalizace a zrychlení výpočtu se nevykreslovalo real-time okno, nedocházelo k vypisování hodnot do konzole pro ladění apod. Všechny další části kódu byly stejné. Komponenty typu displej a zvuková signalizace se musely otestovat až po vlastním připojení k zařízení.

Testování probíhalo v několika etapách. Nejprve se systém zkoušel na nasbíraných snímcích. Na ukázce 4.14 je zobrazen program PyCharm s detekcí dopravní značky a procesu OCR. Bylo také vytvořeno kontrolní okno pro náhled při porovnávání klíčových bodů se vzorem.



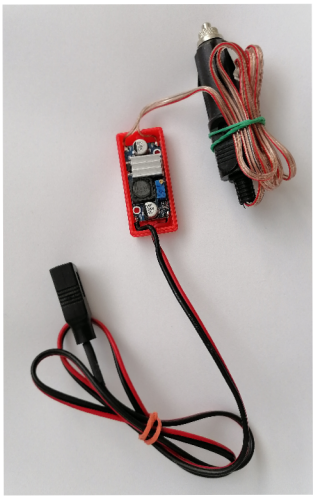
Obrázek 4.14: Testování systému na snímcích

Přílohy I a J ukazují testování systému na video záznamu získaného z palubní kamery umístěného za sklem osobního automobilu. Je zde patrné okno pro real-time sledování detekce objektů, okno detekované značky a konzole, ve které jsou vypsány čtyři rozpoznaná dvojčíslí, řádek se spuštěním zvukové signalizace při překročení max. povolené výšky vozidla a také průměrná hodnota FPS.

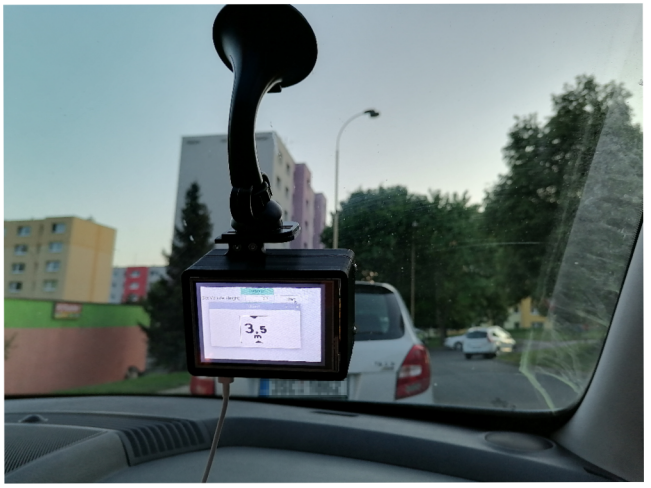
Třetím typem testovaného formátu byl přímý výstup z kamery. Výsledný snímek byl identický jako u video záznamu.

Funkční, ucelené a estetické zařízení se také nasadilo do reálného testování za sklem osobního automobilu. Napájení pro desku se přivedlo ze zásuvky vozu. Kvůli rozdílným hodnotám napětí – 5V pro zařízení a 12V pro zásuvku, se musel použít snižující měnič napětí. Připojené zařízení k napájení vozidla, je možné vidět na snímku 4.15a, zatímco obrázek 4.15 představuje nasazení zařízení do provozu. V příloze X je dostupný větší počet ukázek.

Chod reálného zařízení probíhal identicky jako na PC. Na vstupní obraz z kamery byla aplikována detekce objektů. Rozpoznaná značka spustila proces OCR a při překročení max. výšky vozidla došlo ke zvukové signalizaci.



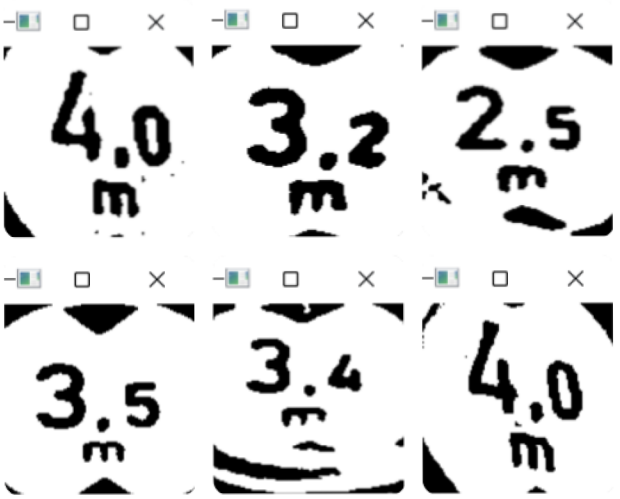
(a) Napájení pro Raspberry Pi do vozidla



(b) Navržené zařízení za sklem osobního automobilu

Obrázek 4.15: Ukázka reálného zařízení v praxi

Na obrázku 4.16 je vidět soubor detekovaných značek, které byly oříznuty a protaženy procesem OCR. Na snímcích byla aplikována funkce *adaptiveThreshold*, která snímek převedla do dvou barev – černá a bílá. To eliminovalo okolní šum, čímž se zvýšila pravděpodobnost na rozpoznání znaků.



Obrázek 4.16: Soubor oříznutých detekovaných značek

Pro přiblížení se při samotné detekci dopravních značek docílilo rychlosti 15 FPS. Po spuštění vláken a procesu OCR klesla rychlost na 5 FPS. Rychlost od detekce značky až po zapnutí signalizace trvá dvě až tři vteřiny. Ovšem přesnost detekce zůstala vysoká.

Celý proces vyžadoval příliš mnoho času na zpracování, během kterého by mohlo dojít ke kolizi. Je proto nutné zařízení optimalizovat, aby pracovalo rychleji. Dostačující hodnotou je 30 snímků za vteřinu, tudíž dvojnásobek rychlosti než dokáže navržený systém. O možnostech zlepšení a optimalizace zařízení bude řeč v následující kapitole 4.5 Shrnutí a diskuze.

4.5 Shrnutí a diskuze

Pro realizaci navrhovaného zařízení se nejdříve vybraly metody pro detekci objektů, rozpoznávání znaků a vhodná zařízení pro realizaci. Následovala klíčová část – tvorba kódu. Vytvořily se funkční kód, kterým se systém řídí. Dále došlo k montáži komponent k zařízení, a modelaci a tvorbě pouzdra pro zakrytí, ochranu dílů a zajištění chlazení. Pro ověření funkčnosti se celek nakonec otestoval v praxi.

Takto vytvořený systém splňuje všechny cíle, které byly kladeny. Systém je schopen detekovat dopravní značku B16 – výška mostu. Na základě oříznutého objektu dokáže rozpoznat text na značce a provést zvukovou signalizaci řidiči vozidla. Navrhnuté zařízení je rovněž funkční, kompaktní, vzhledově jednoduchý, barevně sladěný.

Samotná rychlost detekce byla velmi dobrá, dá se říct, že nijak neovlivňuje zpracování obrazu. Činnost OCR byla pomalejší a brzdila plynulost celého procesu. Na úkor zpomalení se docílilo přesně rozpoznaných číslic. Použité zařízení Raspberry Pi i navzdory vyššímu výkonu, ve srovnání s ostatními dostupnými deskami, nestíhalo provádět veškeré činnosti dostatečně rychle, a to i přes řadu optimalizací. Už při zpracování HD obrazu se nedosahovalo takových hodnot FPS, jak se očekávalo.

Možností, jak zlepšit systém, je celá řada. Jeho zabudováním např. do vozidel Tesla model S. Jelikož se jedná o elektromobily, obsahují celou řadu elektroniky a především řídící jednotku. Ta je pravděpodobně na tolik moderní a výkonná, že by na ní mohl běžet vytvéřený systém této práce. Mohlo by dojít ke zrychlení a příp. i zvětšení velikosti vstupního obrazu.

Určitě není od věci zkusit nahradit Raspberry Pi za např. desku Khadas VIM4. Jedná se o novou desku, která je osazena osmijádrovým procesorem s frekvencí 2

GHz (o 0,5 GHz více než u Raspberry Pi 4B), 8 GB RAM, má taktéž piny, a jedná se o konkurenta k Raspberry Pi deskám. Je ovšem nutné zvážit cenu, a také zdali by došlo k navýšení výkonu, urychlení činností. S ohlednem na veškerá navýšení a zlepšení by cena vzrostla dvojnásobně. Zkrátka je nutné zvážit, zdali se takové zařízení vyplatí.

Deska Raspberry Pi byla ideální volbou z pohledu poměru mezi cenou a výkonem. Její tvůrci vyvíjí novější model konkr. verzi 5, ktera by měla poskytnout větší výkon, větší počet paměti nebo např. lepší chlazení v podobě měděných vodičů. Zkrátka se jedná o výkonější řadu, která by mohla vést k řešení zmíněných nedostatků.

Samotná detekce objektů založená na metodě Viola – Jones je jednoduchá, přesná a rychlá. Nejprve dojde k detekci, poté k rozpoznání znaků. Alternativou je volba jiné metody, která by byla založená např. na konvoluční neuronové síti. Cílem by bylo detekovat značky z natrénovaného modelu a napřímo je klasifikovat, čímž by se eliminoval proces OCR. Výstupem by rovnou byla data o výšce. Nicméně shromáždit velké množství snímků dopravních značek s různými výškami je dosti pracné a je nutné zvážit takovou variantu.

Rozdělení činnosti OCR mezi vlákna byla také provedena za účelem zvýšení rychlosti. Při více než čtyřech vláknech by byly hodnoty zrychlení znatelnější. Zvážení jiné varianty rozpoznávání znaků by také mohlo přinést snížení výpočetního času.

Použitá kamera s objektivem měla celkem vysoké rozlišení, úhel záběru a umožňovala manuální regulaci clony a vzdálenost, na kterou zaostřila. Jediným nedostatkem bylo neúplné proostření obrazu.

Zařízení je elegantí, provozu schopné, akorát vyžaduje péči. Možností pro vývoj a zlepšení je dost. V rámci diplomové práce by se zařízení optimalizovalo, vybraly by se vhodnější postupy, zařízení a dovedlo k dokonalosti.

Závěr

Práce se zabývala strojovým učením, umělou inteligencí, počítačovým viděním, rozpoznáváním znaků a jejich aplikací v praxi, a to především pro detekci objektů a rozpoznávání znaků.

V rámci této práce byla vytvořena datová sada dopravních značek B16 – výška mostu, které byly shromážděny na území České republiky. Sadu lze rozšířit o zahraniční typy dopravních značek a může fungovat mimo ČR. Pro natrénování modelu se použilo učení pod dohledem, které nevyžadovalo velké množství trénovacích dat. Model sloužil jako vzor pro detekci značky.

Knihovna Tesseract OCR umožnila rozpoznávání číslic z detekované značky. Spolu s údajem o výšce vozidla, kterou uživatel nastavil před jízdou, došlo k vyhodnocení. Pokud výška vozidla byla vyšší než výška mostu, proběhla zvuková signalizace, která upozornila řidiče.

V závěru kapitoly, která se věnovala realizaci, byla řeč o shrnutí realizace a diskuzi, ve které byly uvedeny pozitiva a negativa navrženého systému. Byly také uvedeny možné způsoby, jak omezit nedostatky a zajistit rychlejší a plynulejší chod.

Vytvořený kód je dostupný v příloze X a je možné si ho stáhnout a nahrát do vlastního zařízení. Hlavními částmi kódu jsou detekce objektů, vláknování určené pro rozpoznávání znaků a vytvoření vlastního jednoduchého grafického rozhraní pro zadávání výšky vozidla.

Literatura

[1] What is artificial intelligence (ai)? In: *IBM* [online]. 2020 [cit. 2022-04-18]. Dostupné z: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence.

[2] RUSELL, Stuart J., a Peter, NORVIG. *Artificial Intelligence: a modern approach.* 3rd ed. Upper Saddle River, New Jersey 07458: Prentice Hall, 2010. Prentice Hall series in artificial intelligence. ISBN: 978-0-13-604259-4.

[3] MCCARTHY, John. *What is artificial intelligence?* [online]. Stanford, CA 94305: 2007. [cit. 2022-04-18]. Dostupné z: http://jmc.stanford.edu/articles/whatisai/whatisai.pdf.

[4] What is machine learning? In: *IBM* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://www.ibm.com/cloud/learn/machine-learning.

[5] *Algoritmy strojového učení* [online]. 2020 [cit. 2022-04-19]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/machine-learning-algorithms.

[6] SONI, Devin. Supervised vs. unsupervised learning. In: *Towards Data Science* [online]. 2018 [cit. 2022-04-19]. Dostupné z: https://towardsdatascience.com/supervised-vs-unsupervised-learning--14f68e32ea8d.

[7] ŠTOL, Bc Jan. *Strojové učení s využitím metody transfer learning.* Hradec Králové, 2019. Diplomová práce. Univerzita Hradec Králové. Fakulta informatiky a managementu. Katedra informacních technologií.

[8] DAŇHELOVÁ, Jana. *Zpětnovazební učení pro řešení herních algoritmů.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií.

[9] What is deep learning? In: *IBM* [online]. 2020 [cit. 2022-04-19]. Dostupné z: https://www.ibm.com/cloud/learn/deep-learning.

[10] VEEN, Fjodor Van. *The Neural Network Zoo* [online]. 2022 [cit. 2022-04-19].

Dostupné z: https://www.asimovinstitute.org/neural-network-zoo.

[11] KHOSROWPOUR, Mehdi. *Encyclopedia of information science and technology.* 2nd ed. Hershey, PA: Information Science Reference, 2009. ISBN: 978-1-60566--026-4.

[12] ABDUH, Muhammad. *A History of Triggering Artificial Neuron.* In: *Towards Data Science* [online]. 2018 [cit. 2022-04-19]. Dostupné z: https://towardsdatascience.com/a-history-of-triggering-artificial-neuron--d1d9853d9fdc.

[13] FREUND, Yoav a Robert E., SCHAPIRE. Large margin classification using the perceptron algorithm. In: *Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98* [online]. New York, USA: ACM Press, 1998, s. 209–217. ISBN: 978-1-58113-057-7. Dostupné z: doi: 10.1145/279943.279985.

[14] What are neural networks? In: *IBM* [online]. 2020 [cit. 2022-04-19]. Dostupné z: https://www.ibm.com/cloud/learn/neural-networks.

[15] DONGES, Niklas. Recurrent neural networks and LSTM networks. In: *Built In* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://builtin.com/data--science/recurrent-neural-networks-and-lstm.

[16] VALKOV, Venelin. Cryptocurrency price prediction using lstms | tensorflow for hackers (part iii). In: *Towards Data Science* [online]. 2019 [cit. 2022-04-19]. Dostupné z: https://towardsdatascience.com/cryptocurrency-price-prediction--using-lstms-tensorflow-for-hackers-part-iii-264fcdbccd3f.

[17] What are convolutional neural networks? In: *IBM* [online]. 2020 [cit. 2022--04-19]. Dostupné z: https://www.ibm.com/cloud/learn/convolutional-neural--networks.

[18] ZHANG, Aston, Zachary C., LIPTON Mu, LI aj. Convolutional neural networks. In: *Dive into Deep Learning* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://www.d2l.ai/chapter_convolutional-neural-networks/index.html.

[19] *Počítačové zpracování obrazu* [online]. 2022 [cit. 2022-04-19]. Dostupné z: https://azure.microsoft.com/cs-cz/services/cognitive-services/computer-vision.

[20] KARGIN, Kerem. Computer vision fundamentals and OpenCV overview. In: *Medium* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://medium.com/mlearning-ai/computer-vision-fundamentals-and-opencv-

-overview-9a30fe94f0ce.

[21] What is computer vision? In: *IBM* [online]. 2022 [cit. 2022-04-19]. Dostupné z: https://www.ibm.com/topics/computer-vision.

[22] *Computer Vision: What it is and why it matters* [online]. 2022 [cit. 2022--04-19]. Dostupné z: https://www.sas.com/cs_cz/insights/analytics/computer--vision.html.

[23] *Object Recognition* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://www.mathworks.com/solutions/image-video-processing/object-recognition.html.

[24] BOESCH, Gaudenz. A complete guide to image classification in 2022. In: *viso.ai* [online]. 2018 [cit. 2022-04-19]. Dostupné z: https://viso.ai/computer--vision/image-classification.

[25] STECANELLA, Bruno. Support vector machines (SVM) algorithm explained. In: *MonkeyLearn* [online]. 2017 [cit. 2022-04-19]. Dostupné z: https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm.

[26] JAIN, Taru. Basics of machine learning image classification techniques. In: *OpenGenus IQ* [online]. 2019 [cit. 2022-04-19]. Dostupné z: https://iq.opengenus.org/basics-of-machine-learning-image-classification--techniques.

[27] BOESCH, Gaudenz. Object detection in 2022: The definitive guide. In: *viso.ai* [online]. 2018 [cit. 2022-04-19]. Dostupné z: https://viso.ai/deep-learning/object-detection.

[28] MEEL, Vidushi. Video analytics: Ultimate overview in 2022. In: *viso.ai* [online]. 2018 [cit. 2022-04-19]. Dostupné z: https://viso.ai/computer-vision/video-analytics-ultimate-overview.

[29] SINGH, Aastha. Yolo: You only look once. In: *Medium* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://medium.com/nerd-for-tech/yolo-you-only-look-once-65ea86104c51.

[30] GIRSHICK, Ross, Jeff, DONAHUE Trevor, DARRELL aj. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv* [online]. 2013, 1311. Dostupné z: http://arxiv.org/abs/1311.2524.

[31] GANDHI, Rohith. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms. In: *Towards Data Science* [online]. 2018 [cit. 2022-04-19]. Dostup-

né z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object--detection-algorithms-36d53571365e.

[32] VERMA, Yugesh. R-cnn vs fast r-cnn vs faster r-cnn - a comparative guide. In: *Analytics India Magazine* [online]. 2021 [cit. 2022-04-19]. Dostupné z: https://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide.

[33] TYAGI, Mrinal. Hog (histogram of oriented gradients). In: *Towards Data Science* [online]. 2021. [cit. 2022-04-19]. Dostupné z: https://towardsdatascience.com/hog-histogram-of-oriented-gradients--67ecd887675f.

[34] SINGH, Aishwarya. Feature descriptor | hog descriptor tutorial. In: *Analytics India Magazine* [online]. 2019 [cit. 2022-04-19]. Dostupné z: https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor.

[35] MALLICK, Satya. Histogram of oriented gradients explained using opencv. In: *LearnOpenCV* [online]. 2016 [cit. 2022-04-19]. Dostupné z: https://learnopencv.com/histogram-of-oriented-gradients.

[36] WINARNO, Edy, Wiwien, HADIKURNIAWATI, Dahlan, ABDULLAH aj. Multi-view faces detection using viola-jones method. *Journal of Physics: Conference Series* [online]. 2018, 1144 [cit. 2022-04-20]. ISSN 1742-6588. Dostupné z: doi: 10.1088/1742-6596/1114/1/012068.

[37] MITTAL, Aditya. Haar cascades, explained. In: *Analytics Vidhya* [online]. 2020 [cit. 2022-04-20]. Dostupné z: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d.

[38] Om Rastogi. Haar cascade classifiers. In: *Data Driven Investor* [online]. 2020 [2022-05-10]. Dostupné z: https://medium.datadriveninvestor.com/haar-cascade-classifiers-237c9193746b.

[39] What is optical character recognition (OCR)? In: *IBM* [online]. 2022 [cit. 2022-04-20]. Dostupné z: https://www.ibm.com/cloud/blog/optical-character--recognition.

[40] *What is Optical Character Recognition OCR Technology?* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://www.hyland.com/en/resources/terminology/data-capture/what-is-

optical-character-recognition-ocr.

[41] Comparison of optical character recognition software. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2022 [cit. 2022-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Comparison_of_optical_character_recognition_software&oldid=1082049488.

[42] Stanley (vehicle). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2022 [cit. 2022-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Stanley_(vehicle)&oldid=1070923225.

[43] Tesseract (software). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2022 [cit. 2022-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Tesseract_(software)&oldid=1071915981.

[44] OCRopus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2022-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=OCRopus&oldid=1043556103.

[45] Asprise OCR. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2022-04-20]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Asprise_OCR&oldid=1004316401.

[46] MASSOUD, M. A. Over-height vehicle detection system in egypt. In: *Proceedings of the World Congress on Engineering 2013 Vol II* [online]. London, U.K.: WCE, 2013, July 3-5, 2013, s. 4 [cit. 2022-04-30]. ISBN 978-988-19252-8-2. ISSN 2078-0966. Dostupné z: http://www.iaeng.org/publication/WCE2013/WCE2013_pp1010-1013.pdf.

[47] *Overheight Vehicle Detection* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://www.swarco.com/products/electronic-signs/vehicle-activated-safety-signs/overheight-vehicle-detection.

[48] *Traffic Sign Detection & Car Protection App* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://www.myguardcam.com.

[49] AHMADI, Amin. *Amin* [online]. 2022 [cit. 2022-04-20]. Dostupné z: https://amin-ahmadi.com.

[50] ROSSUM, Guido van. *Guido van Rossum - Resume* [online]. 2021 [cit. 2022-

-04-20]. Dostupné z: https://gvanrossum.github.io/Resume.html.

[51] DEVIREDDY, Pravallika. Python introduction. In: *Learning Python programming language* [online]. 2020 [cit. 2022-04-20]. Dostupné z: https://medium.com/learning-python-programming-language/python-introduction-bab4aa994ce0.

[52] C (programovací jazyk). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2021 [cit. 2022-04-20]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=C_(programovac%C3%AD_jazyk)&oldid=20666056.

[53] Raspberry pi. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2022 [cit. 2022-04-20]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Raspberry_Pi&oldid=21086520.

[54] *Raspberry Pi 4 Model B (Raspberry-PI-4-8GB)* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://www.zbozi.cz/vyrobek/raspberry-pi-4-model-b-raspberry-pi-4-8gb.

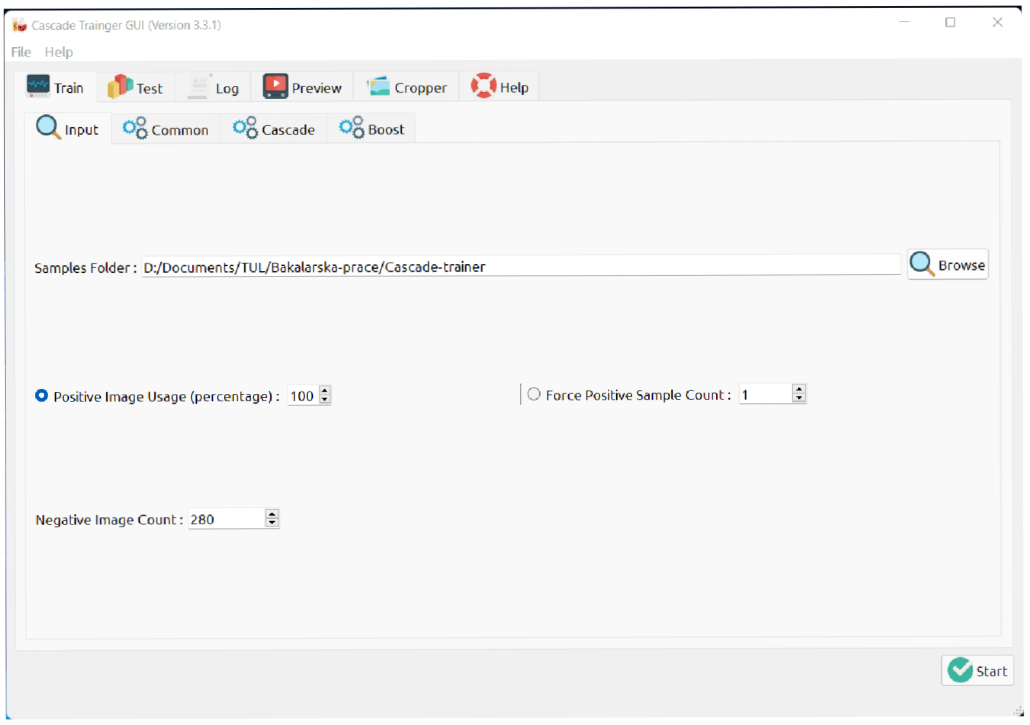
[55] How to keep your raspberry pi 4 from overheating | raspberry pi. In: *Maker Pro* [online]. 2019 [cit. 2022-04-20]. Dostupné z: https://maker.pro/raspberry-pi/tutorial/how-to-prevent-your-raspberry-pi-4-from-overheating.

[56] *Raspberry Pi HQ kamera* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://rpishop.cz/mipi-kamerove-moduly/2458-raspberry-pi-hq-kamera-0633696492738.html.

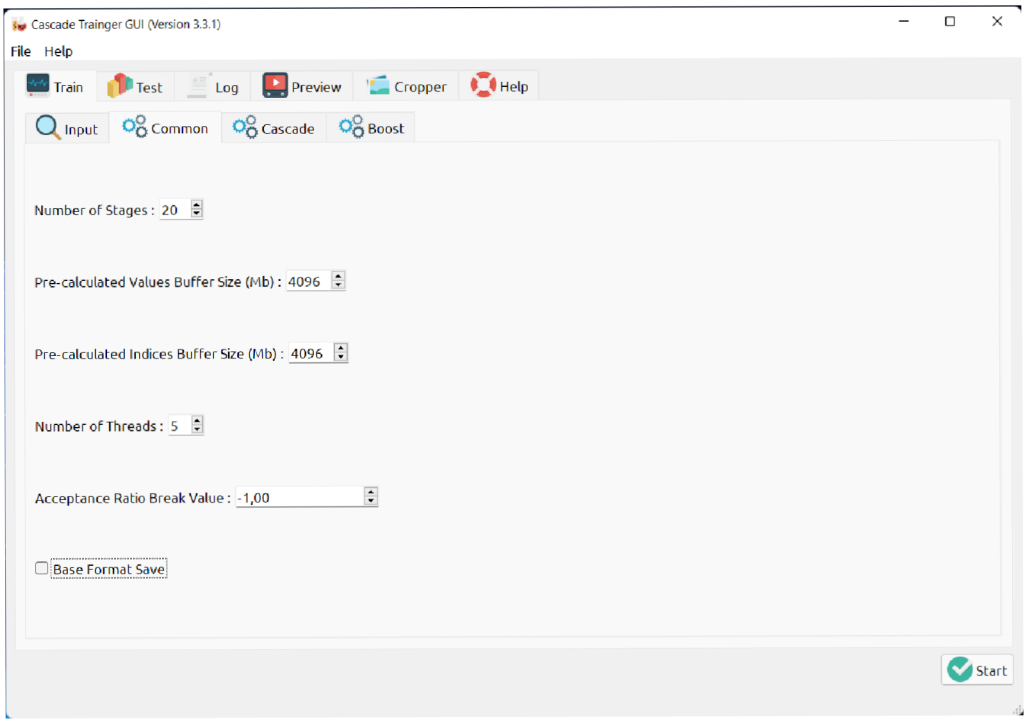
[57] *Raspberry Pi 6mm f/1.2 CS objektiv* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://rpishop.cz/normalni-objektivy/2460-raspberry-pi-6mm-sirokouhly-objektiv.html.

[58] A practical guide to python threading by examples. In: *Python Tutorial - Master Python Programming For Beginners from Scratch* [online]. 2022 [cit. 2022-05-01]. Dostupné z: https://www.pythontutorial.net/advanced-python/-python-threading.

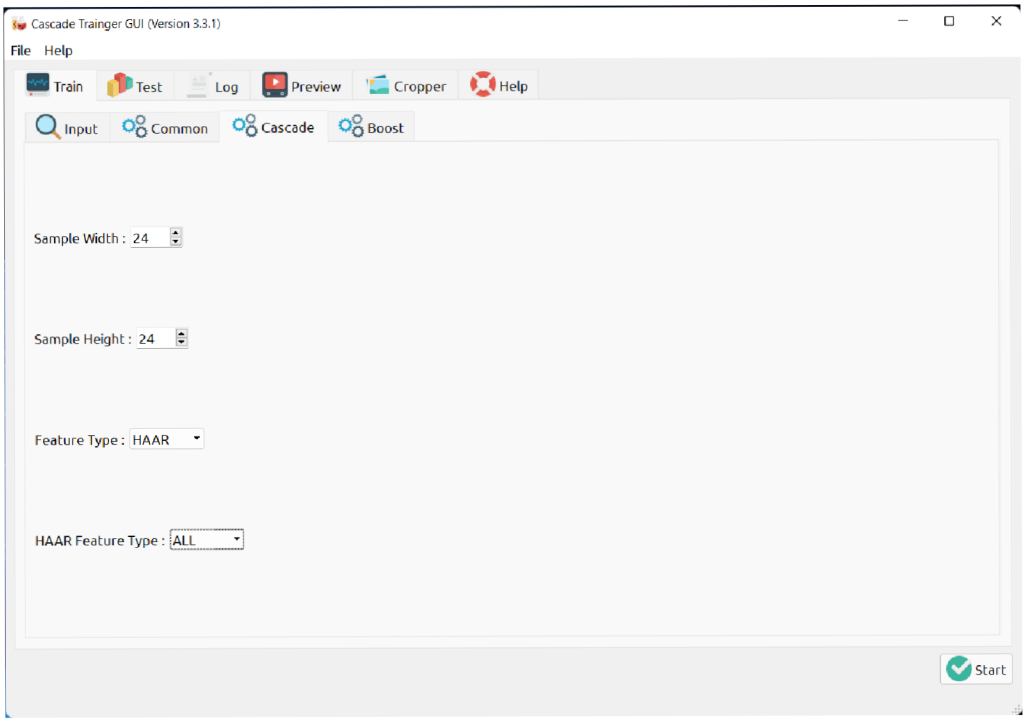
Seznam příloh



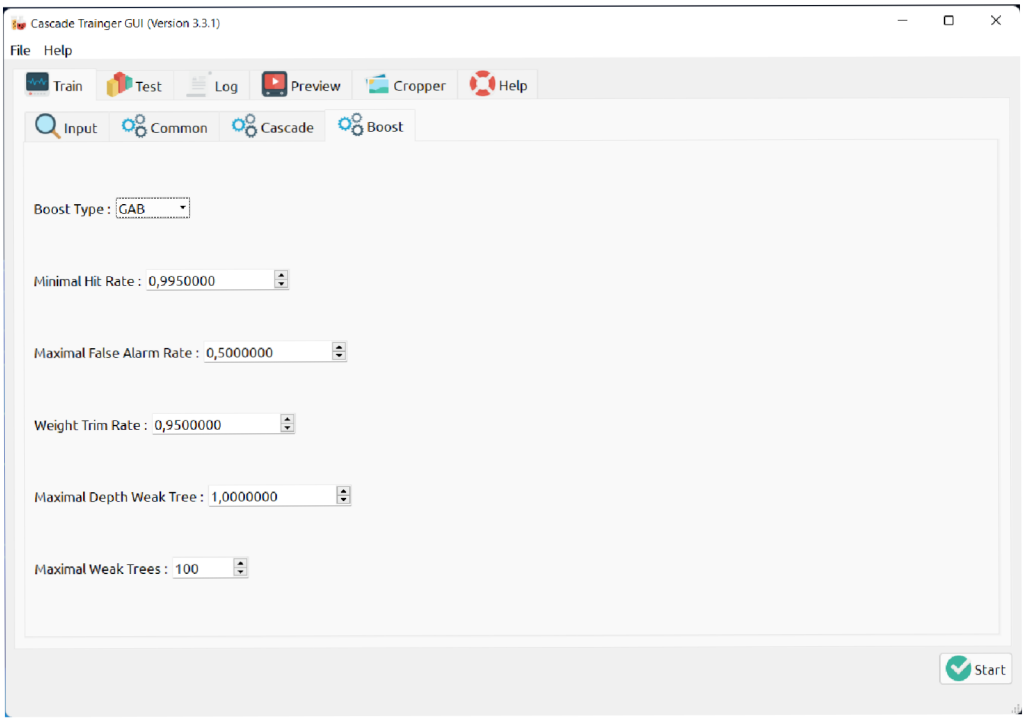
Příloha A: Nastavené hodnoty parametrů pro Haar klasifikátor - Vstup (Input)



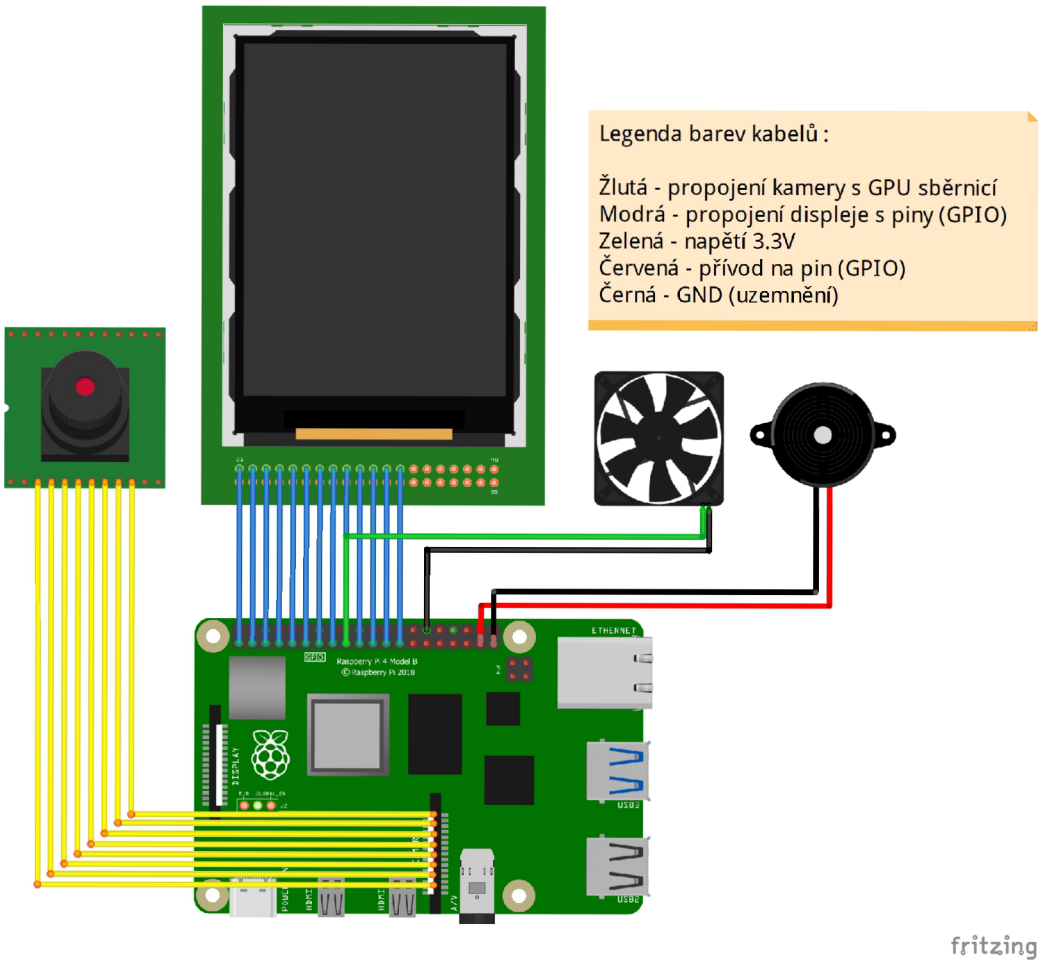
Příloha B: Nastavené hodnoty parametrů pro Haar klasifikátor - Obecné (Common)



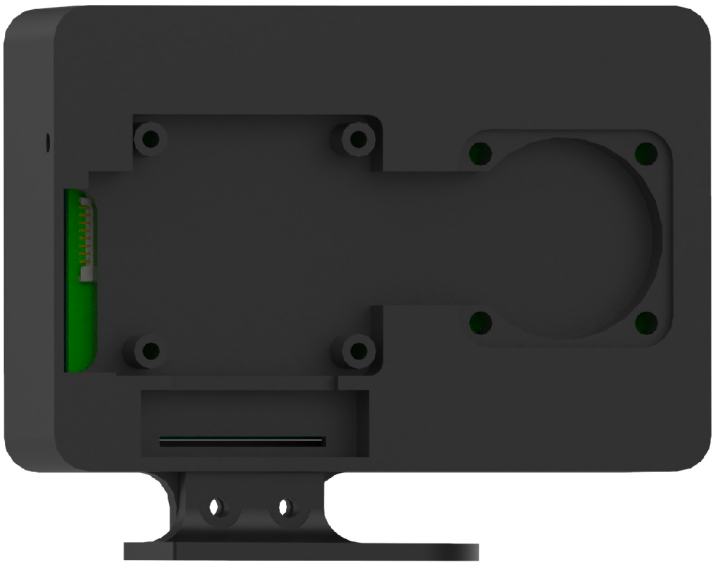
Příloha C: Nastavené hodnoty parametrů pro Haar klasifikátor - Kaskáda (Cascade)



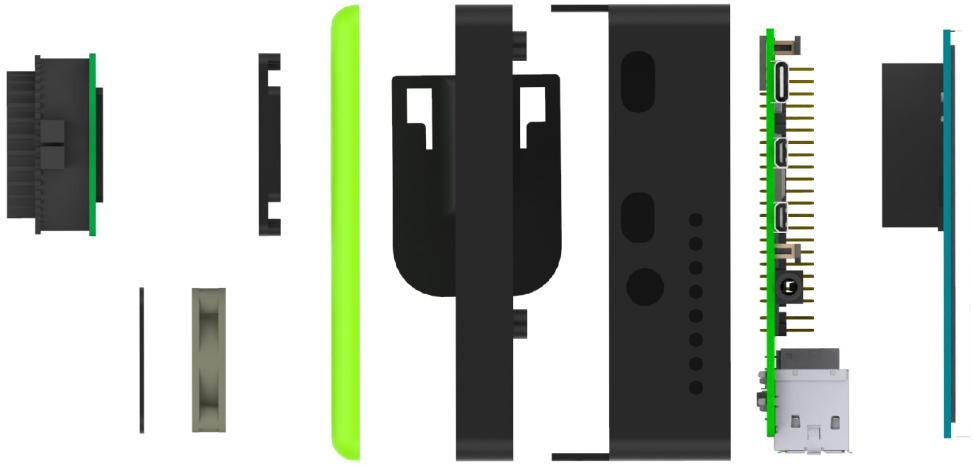
Příloha D: Nastavené hodnoty parametrů pro Haar klasifikátor - Boost



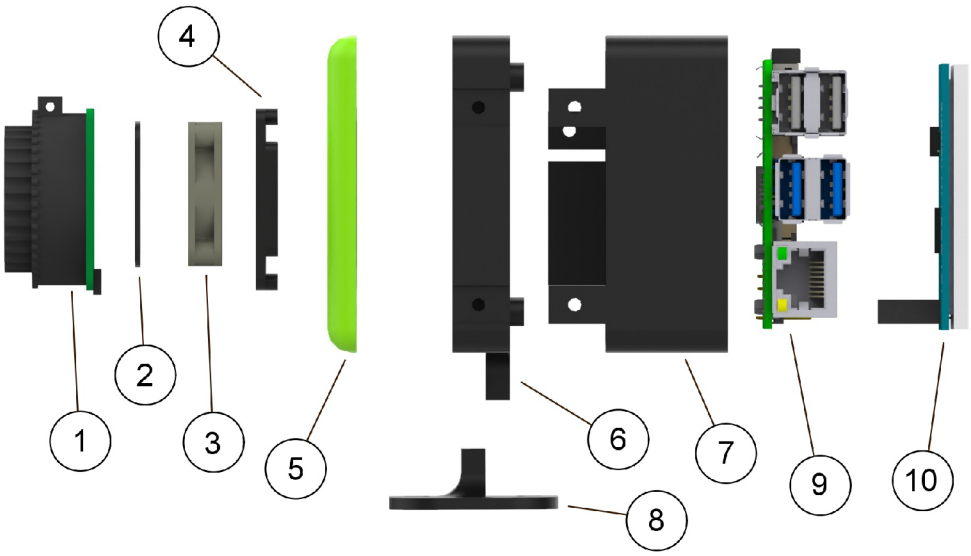
Příloha E: Schéma zapojení komponent k Raspberry Pi



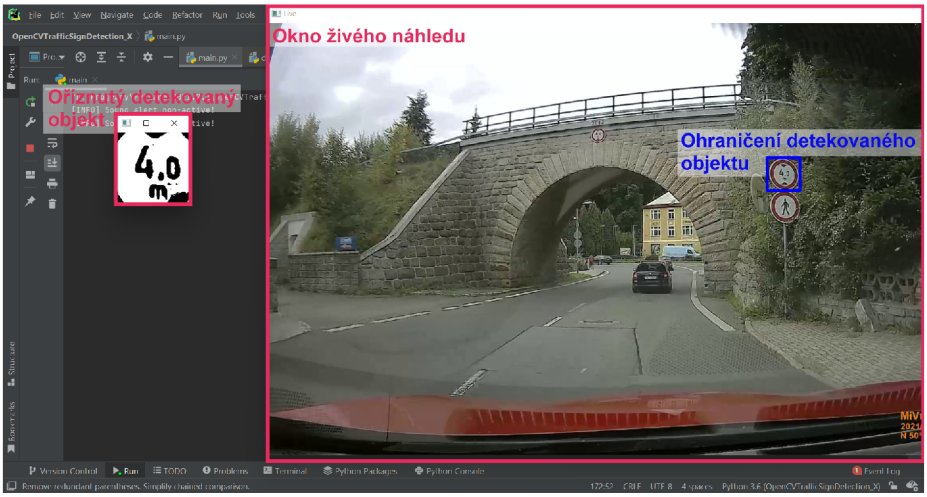
Příloha F: Kanál pro sání vzduchu z pouzdra



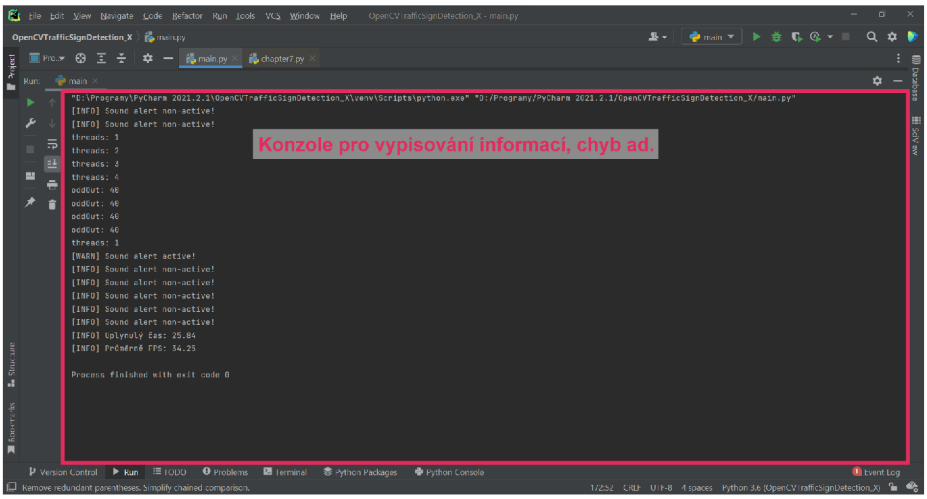
Příloha G: Horní, resp. spodní pohled na rozebranou sestavu zařízení (bez spojovacích dílů, kabelů apod.)



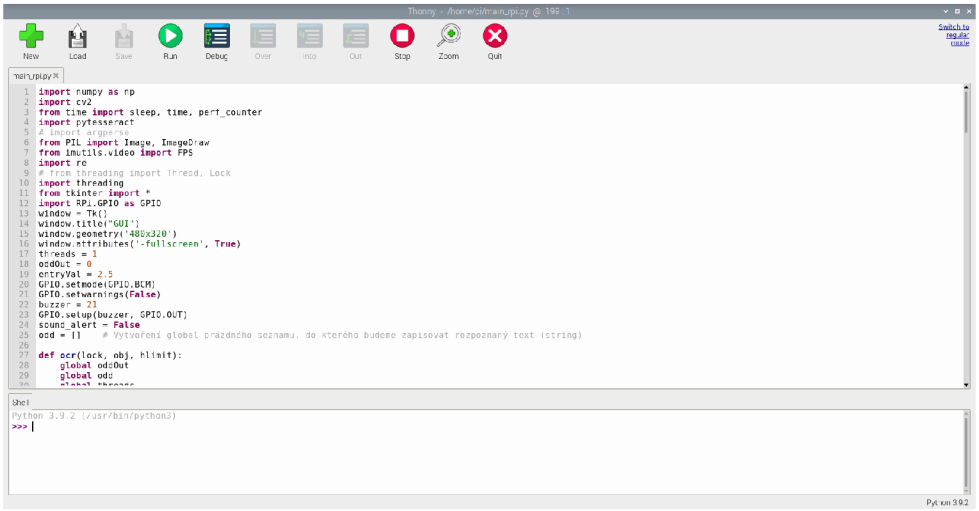
Příloha H: Boční pohled na rozebranou sestavu zařízení (bez spojovacích dílů, kabelů apod.)



Příloha I: Testování funkčnosti systému na PC se systémem Windows - 1. část



Příloha J: Testování funkčnosti systému na PC se systémem Windows - 2. část



Příloha K: Grafické prostředí dostupné pro Linux