VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY
INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

SLEDOVÁNÍ POHYBU OBJEKTŮ POMOCÍ POHYBLIVÉ KAMERY

TRACKING THE MOVEMENT OF OBJECTS USING A MOVING CAMERA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Andrea Duráníková
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Králík
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadaní bakalářské práce

Ústav:	Ústav mechaniky těles, mechatroniky a biomechaniky
Studentka:	**Andrea Duráníková**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Mechatronika
Vedoucí práce:	**Ing. Jan Králík**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Sledování pohybu objektů pomocí pohyblivé kamery

Stručná charakteristika problematiky úkolu:

S nástupem chytrých systémů začíná být nevýhodné mít pro některé aplikace pouze statické sledování. Jedná se především o prostory, kde je předpokládána menší hustota pohybu, nebo je sledování zapotřebí hlavně až je pohyb malý (skladovací systémy, zabezpečení prostoru, chytré parky apod.) U těchto případů je více než vhodné, aby jedna kamera dokázala obsáhnout větší prostor, ale při zachycení pohybu rozpoznat zda jde o hledaný objekt (přepravní vozík ve skladišti či chodec v parku) a následně dokázala hlídaný objekt sledovat a podle toho aktivovat další systémy.

Cíle bakalářské práce:

1) Porovnání systémů na zpracování obrazu.
2) Rozpoznání pohybu v obraze.
3) Rozpoznání předdefinovaných tvarů.
4) Sledování vybraného objektu ve vymezeném prostoru.
5) Reálné ověření systému.

Seznam doporučené literatury:

BLANCHET, G., CHARBIT, M.: Digital signal and image processing using Matlab. NewportBeach, CA: ISTE, 2006. ISBN 978-1-905209-13-2.

HLAVÁČ, V., SEDLÁČEK, M.: Zpracování signálů a obrazů. Vydavatelství ČVUT, Praha, 2000, ISBN 80-01-02114-9.

KOTEK, Z., MAŘÍK, V., HLAVÁČ, V., PSUTKA, J., ZDRÁHAL, Z.: Metody rozpoznávání a jejich aplikace, Academia, Praha, 1993, ISBN 80-200-0297-9.

PETROU, M., BOSDOGIANNI, P.: Image Processing – The Fundamentals. John Wiley & Sons, New York, 1999, ISBN 0-471-99883-4.

GOLLAPUDI, S.: Learn Computer Vision Using OpenCV, Apress, 2019, ISBN: 9781484242612.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

prof. Ing. Jindřich Petruška, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Bakalářská práce se zabývá softwarem pro dynamické kamery s vyhodnocením objektů. První teoretická část porovnává programy pro ovládání sestavy a vyhodnocení dat, dále způsobem zpracování pohybu, možnostmi zjištění pohybu a objektů. Praktická část realizuje algoritmus s funkcí sledování objektů, jejich rozlišení do skupin a záznamu sledování za použití programu MATLAB.

KLÍČOVÁ SLOVA

zpracování obrazu, detekce objektu, detekce pohybu

DURÁNÍKOVÁ, Andrea. Sledování pohybu objektů pomocí pohyblivé kamery. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/132910. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav mechaniky těles, mechatroniky a biomechaniky. Vedoucí práce Jan Králík.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že jsem bakalářskou práci na téma „Sledování pohybu objektů pomocí pohyblivé kamery“ vypracovala samostatně pod vedením Ing. Jana Králíka s využitím odborné literatury a zdrojů uvedených v seznamu, který je přílohou této práce.

V Brně dne

Andrea Duráníková

PODĚKOVÁNÍ

Děkuji vedoucímu bakalářské práce Ing. Janu Králíkovi za pedagogickou a odbornou pomoc při zpracování této práce.

OBSAH

1 Úvod

S rozvojem chytrých systémů a potřeby hlídání majetku se rozvíjí programy pro sledovací kamery. Obvyklé a jednodušší bývají statické kamery, které monitorují pouze prostor vymezený danou jednotkou. V případě, že potřebujeme sledovat méně frekventované místo nebo hledáme pouze předdefinovaný objekt je mnohdy výhodnější použít jednu dynamickou kameru s možností pohybu. Tímto pohybem jsme schopni monitorovat větší prostor než ve statickém případě.

Dynamická kamera dokáže nejen obsáhnout větší prostor, ale při vhodně zvoleném programu lze najít objekt zájmu, následovat ho, nebo aspoň určit směr jeho pohybu. Těchto znalostí můžeme využít při tvorbě chytrých osvětlení v parcích či halách s menší frekvencí pohybu. Pomocí nalezeného objektu a jeho pohybu můžeme predikovat jeho směr a dle toho upravit nastavení světel.

Dynamickou kameru a její specifické nastavení lze použít i v soukromém sektoru chytrých domácností. Takové kamery nám mohou nejen postupně rozsvítit či zhasnout světla, jak bylo již zmíněno výše, dále je na systém možné připojit otevírání garážových vrat s rozeznáním registrační značky a například barvy automobilu nebo odemykání vchodových dveří, vypnutí a zapnutí alarmů a dalších poplašných jednotek.

V následujících částech této práce jsou porovnané možné programy na ovládání kamery, způsoby vyhodnocení obrazu a jeho následného zpracování za účelem vyhledání objektů, jejich sledování a následný posun sledovaného prostoru za objektem.

2 Zpracování obrazu

2.1 Software

K tvorbě softwaru lze využít několik programovacích prostředí. Různá prostředí mají své knihovny, práce s nimi, jejich výhody a nevýhody jsou popsány níže.

2.1.1 Java

Jazyk Java je objektově zaměřený. Pro lepší využití tohoto jazyka a jeho vyšší kompatibilitu slouží interpretační metoda, na které celá Java funguje. Tato metoda je založena na přepisu kódu *Bytecode*, který je kompatibilní se všemi systémy. Java využívá interpretační metodu, kdy je zdrojový soubor přeložen pomocí překladače *javac*, pro výpočet je využíváno *Java Virtual Machine*, což je součástí balíčku *Java Runtime Environment*. Rámci programu a jeho výpočtů je využito řady knihoven *Java Core API*. Celkové výhody tohoto jazyka jsou vysoký stupeň zabezpečení, objektová orientace s možností využít procedurálního programování, portabilita a podobnost s jazykem C. Při samotném programování je využíváno blokové struktury. V programu se využívá citlivosti na velká a malá písmena, tímto se rozšiřuje množství možných proměnných [1].

Knihovny pro zpracování obrazu lze využít dvě, a to buď *Java Advanced Imaging (JAI)* nebo *Open CV*, která je tvořena v jazyce C.

JAI je rozsáhlou knihovnou pro zpracování obrazu s novými aktualizacemi. Knihovna čerpá i z jiných podpůrných knihoven jako je *Java 2D API*.

Java 2D API má pro zpracování obrazu různá rozhraní a třídy. Mezi rozhraní patří *BufferedImageOp* ke zpracování objektů na vstupu a výstupu a jejich definice. K tomu v tomto rozhraní slouží třídy *AffineTransformOp*, která je abstraktní třídou používající transformaci pro převedení lineárních souřadnic ze vstupního 2D (dvoudimenzionálního) obrázku na výstupní, *ConvolveOp*, jenž využívá konvoluci k přepočtu vstupních pixelů na výstupní, *BandCombineOp*, což je třída provádějící úpravu daného pásu rastru pomocí specifické matice, a poslední *LookupOp* pro přesun vyhledávání ze zdroje do cíle. *ImageObserver* je rozhraním řešícím notifikace o vytvořeném obraze. *ImageConsumer* se používá současně s dalším rozhraním *ImageProducer* zpracovávajícím informace pro každý obrázek, řeší například změnu výšky či šířky obrázku. Pro přístup ke specifickému kódu obsahuje rozhraní *ImagingLib*. *WriteableRenderedImage* jako rozhraní zajišťuje úpravu dat a jejich přepisování. V rámci této knihovny lze využít ještě další rozhraní a třídy s funkcemi na úpravu obrázku jakou může být barva, velikost, filtry a podobně [2].

Pro programování v *JAI* jsou čtyři základní části, jádro, rozšířené operace, hlavní operátory a zobrazení. K dispozici jsou pokročilejší možnosti úprav obrazu jako filtry,

tvorba histogramů, vyhledávání kritických míst dle zadání. Jsou k dispozici Fourierovy transformace, hledání maxima, minima či srovnání obrazů. Pro orientaci na obraze jsou k dispozici funkce pro tvorbu vektorů a soustavy os x a y. Mezi další možnosti zpracování patří logické operace, pro které jsou k dispozici speciální třídy. Třídy jsou rozděleny dle formátu vkládaného obrazu například formát PNG, JPEG a podobné. Mezi zajímavé možnosti získání vstupů patří i třída zaměřující se na přístup přes URL adresu daného obrázku. Při zpracování videa je potřeba si pro tuto knihovnu z něj udělat sérii obrazů a pak s nimi dále pracovat [2].

Celkově tedy je možné zpracovávat obraz ve dvou i třech dimenzích, tvořit histogramy, filtry, detekce hran, je možné vytvořit zpracování obrazů v čase [3].

2.1.2 Python

Tento jazyk je velmi rozšířeným a kompatibilním s propojením s dalšími programovacími jazyky a jejich knihovnami. Vhodnou výhodou je rychlejší zpracování skriptu a jeho zjednodušení funkcemi tohoto jazyku. Python zvládá vysokoúrovňové datové typy a šetří na nich čas. Toto prostředí má vyšší odolnost vůči chybám programátora a umožňuje programovat interaktivně s testováním skriptu. Nabízí možnosti tvorby uživatelských aplikací a jejich testování se svižnou odezvou. Mnohdy je používán jako testovací verze před finálním převodem například do jazyka C nebo C++ [4].

Pro zpracování obrazu na různých úrovních slouží *scikit-image*, *NumPy*, *SciPy*, *PIL* s nástupcem *Pillow*, *Mahotas*, *SimpleITK*, *pgmagick*, *Pycairo*, *SimpleCV* a *OpenCV*. Jednotlivé knihovny a jejich použití je uvedeno níže. *SimpleITK, SimpleCV* a *OpenCV* jsou základem pro jazyk C, a proto budou popsány v podkapitole týkající se tohoto jazyka.

Scikit-image je rozšiřujícím Python balíčkem pro práci s *NumPy*. Umožňuje pokročilé analýzy obrazu. Balíček podporuje ovládání expozice a barev obrazu jako tvorbu a srovnávání histogramů, popřípadě jejich převod na prostorový (3D) graf, převod barevného obrazu na černobílý a podobně. Lze používat funkce pro hledání linií a hran pro detekci objektů v obraze. Dále nabízí geometrické transformace a registry, obrazové registry, filtry a možnosti obnovy, segmentace obrazu a základní funkce pro detekci objektů. Ke všem těmto částem je k dispozici podpora v příkladech přímo od tvůrců balíčku [5].

Numpy je souborem základních matematických funkcí. Pro zpracování dat do maticového tvaru, interpolaci či analýzu. Slouží jako základní výchozí knihovna pro další jako je *Pillow*, *SciPy* nebo *TensorFlow* [6]. Balíček *TensorFlow* obsahuje funkce pro zpracování obrazu a audia, proto je vhodným rozšířením pro zpracování obrazu z video záznamu. Rozšíření *TensorFlow* je možné použít v lehce omezené míře v jazyku Java [7].

SciPy rozšiřuje možnosti matematických analýz balíčku *Numpy,* další využití nalezne ve zpracování signálů. V rámci matematických výpočtů zahrnuje transformace, filtry a další potřebné funkce [8].

Knihovna *PIL* je původní, nyní už starou knihovnou. Z této knihovny vychází nástupce knihovny *Pillow*. Tato knihovna má v základu svižný přístup k datům a mnoho funkcí pro zpracování obrazu. *Pillow* dokáže zpracovávat velké množství formátů vstupů, filtry, práce s barvami, histogramy, statistické analýzy, předělání formátu obrazu jako je změna velikosti, poměrů stran, rotace a možná tvorba transformací [9].

Mahotas je balíčkem počítačového vidění v Pythonu. Kompatibilita použití je s jazykem C nebo v Pythonu s použitím *Numpy*. Funkce obsahují ovládání barev RGB, hledání dvojic a odlišností v binárním popisu obrazu, morfologické zpracování obrazu, pro práci s obrazem je možné využít Eulerovo číslo, histogramy, Gaussův filtr jak pro jednodimenzionální, tak pro multidimenzionální problém, Laplaceův filtr pro 2D obraz, velké množství transformací a klasických úprav obrazu jako změna velikosti, názvu a podobně [10].

Pgmagick patří do programu *GraphicsMagick*. Program je kompatibilní napříč programovacími jazyky. Balíček obsahuje opět spoustu funkcí na úpravu velikosti, barev, natočení, porovnání a popsání daného obrazu nebo jejich animace do formátu GIF. Formát a velikost vložených obrázků je rozsáhlá [11].

Pycairo je další možností pro práci s obrazem. Dostupné funkce jsou hlavně pro kreslení vektorů v daném prostoru, což je odlišné zpracování, než řeší tato práce. Náleží do programu *Cairo* [12].

2.1.3 C a C++

Tento jazyk je velmi často využívaný pro mnoho různých aplikací. Syntaxe jazyka je složitější, v některých případech nepřehledná chybou programátora. Syntaxe může být problémem při tvorbě vizualizací, ladění a přidávání nových algoritmů. Často se tedy využívá jako jazyk finálního programu, ale předchozí testování, kompilace a ladění jednotlivých částí provádíme v jiném prostředí [13].

Pro zpracování obrazu je v C a C++ vytvořena rozsáhlá knihovna *Open CV*. Knihovnu je možné použít i v jazycích Java nebo Python a Octave s generátorem rozhraní SWIG. Dále lze použít *SimpleCV* a *SimpleITK*. Ty jsou popsány níže.

OpenCV je velmi obsáhlou platformou fungující pro více jazyků. Plným názvem se jmenuje *Open Source Computer Vision* Library a proto obsahuje rozsáhlou nabídku algoritmů od základních úprav obrazů až po pokročilé funkce detekce, identifikace a porovnání objektů, a to i ve formátu videa. Dále nabízí možnosti sledování objektů, tvorbu prostorových modelů, vyhledávání v dostupné databázi. Všechny algoritmy vedou knihovnu k vysoké úrovni možností počítačového vidění, neurální sítě a práci

s obrazem. V rámci zpracování obrazu má knihovna *imgproc.* k dispozici filtry, kreslící funkce, transformace obrazu, práci s barvami, histogramy, analýzy struktury a popis tvaru, detekce objektů, tvarů pohybu a sledování objektů [14].

SimpleCV funguje jak pro C, tak pro Python. Knihovna je rovněž pro počítačové vidění, ale je méně rozsáhlá než předchozí *OpenCV*. Obsahuje možnosti tvorby jednotlivých segmentů v obraze, histogramy, zpracování barev, úpravy dle špiček odstínů. Podporuje detekci objektů. Dostupné jsou i algoritmy pro kreslení a psaní do obrazů [15].

Knihovna *SimleITK* je kompatibilní s Pythonem, C++ a mnoha dalšími. Knihovna je součástí platformy *ITK*, která je vytvořena ve spolupráci s vědeckou komunitou pro tvorbu široké škály obrazových analýz [16]. Umožňuje zpracování a segmentaci ve dvou, třech i více dimenzích. Knihovna obsahuje obsáhlou podporu hlavně pro medicínské využití, popřípadě pro analýzy struktur materiálů. Podporované provázání je na zpracování výstupů elektronového nebo fotonového mikroskopu, rentgenové tomografie [17].

2.1.4 Matlab

Syntaxe jazyka je odlišná od ostatních, ale jednoduše pochopitelná a snadná na naučení. Využíván je hlavně pro vědecké účely s využitím širokého spektra matematických operací. Program je založen na maticovém počítání, bez využívání matic během simulací může být pomalejší než ostatní výše uvedené možnosti. Pro svou přehlednost, snadné editování a úpravy kódu se často používá při studiu a praxi v různých odvětvích. Mezi nejčastější odvětví patří strojírenství, elektrotechnika a informační technologie. Postupným rozšiřováním a vylepšováním zvládá operace v reálném čase, popřípadě převod zapsaného kódu do C nebo jiných jazyků, pro snadnější kompatibilitu s cílovým zařízením. Vzhledem k možnostem převodů do jiného jazyka je možné využívat i jiné knihovny než přímo oficiální balíčky *Mathworks*, například výše zmíněná knihovna *OpenCV* nebo další prostředí kompatibilní s jazykem C, Java, Python a další. Pro některá řešení je vhodnější grafické programování, pro tuto možnost je Matlab plně kompatibilní s programem Simulink, kde je spousta vytvořených simulačních funkcí pro složitější procesy [18].

V tomto prostředí jsou jednotlivé funkce rozděleny mezi více balíčků. Pro zpracování obrazu se využívá hlavně *Image Processing Toolbox*. Při zpracování video záznamu je možné využít *Image Acquisition Toolbox*.

Image Processing Toolbox je vhodný pro zpracování obrazu, analýzu, vizualizaci a vývoj algoritmů s touto problematikou spojených. Balíček podporuje širokou škálu formátu vstupních dat pro snadnější práci s kamerami, senzory, mikroskopy, medicínskými zařízeními a dalšími vědeckými přístroji. V rámci přípravy obrazu lze použít funkce na odstranění šumu, úpravu barev či výběr oblastí. Vizualizace dat je možná ve 3D

na poskládání celých předmětů z dat a práce s nimi. Při zpracování obrazu detekcí hran nabízí mnoho metod vyhledávání. Mezi další funkce patří hledání určitých tvarů v obraze, segmentace obrazu a mnoho dalších. Pro fungování algoritmů lze využít běh simulace přímo na počítači nebo kompatibilitu s dalšími zařízeními s kódem v Matlabu nebo převedeném na C [19].

Image Acquisition Toolbox je balíčkem zaměřeným na rozšířené možnosti kompatibility zařízení s programem. Zahrnuje funkce pro nastaveni a vyhledání vhodných zařízení, možnosti převodu vstupních dat do souboru s příponou Matlabu, zapisování videa včetně možností jeho kvality. Dále obsahuje funkce pro zlepšení vědeckých funkcí ve specializovaných oborech jako medicína, astrologie a podobně. Mimo různých optických kamer je možné ovládat třeba termokamery, dále pomocí dalších vylepšení lze lépe algoritmus spustit na procesorech, aby program nepotřeboval prostředí přímo v počítači [20].

2.2 Metody zpracování obrazu

Dle potřeb určování objektů, pohybu a jeho sledování lze využít několik metod zpracování obrazu. Tyto metody budou námětem dalších podkapitol tříděných dle hledaného problému, a to buď pohybu, objektu nebo sledování vybraného objektu. V první podkapitole jsou však uvedeny základní úpravy obrazu.

2.2.1 Základní úpravy a pojmy

Pro základní úvahy o digitálním obraze je potřeba si obraz přestavovat rozdělený do jednotlivých pixelů poskládaných do čtvercové mřížky. Tato mřížka je dále zpracovávaná. Objekty hledané v mřížce jsou dosaženy úpravami neboli segmentací obrazu a po dokončení je rozeznatelný objekt a jeho pozadí. K rozlišení objektu od pozadí může sloužit hodnota jasu daného pixelu. Pro porovnání jasu pixelů slouží předem určená hodnota prahu, s tou je jas srovnán a dle parametrů vyhodnocen [21].

Dalším pojmem v oblasti digitálního obrazu jsou hranice, ty se uplatňují hlavně při metodě detekce hran. Tyto hranice tvoří okraje daných objektů neboli množin pixelů se stejnou hodnotou jasu. To lze nejlépe vysvětlit na černobílém obraze, kde je bílé pozadí a černé plné kolečko, kružnice na okraji kolečka tvoří hranici. Tato kružnice není samozřejmě ve čtvercové mřížce zcela hladká. Mezi další pojem může patřit díra v objektu, což je místo, které má stejnou barvu jako pozadí. Poté by výše zmíněný kroužek měl hned dvě hraniční kružnice, a to kolem díry a kolem objektu. Pokud by hranice byla dělena ještě na vnější a vnitřní, tak dokonce čtyři [21].

Dalším důležitým pojmem je hrana v digitálním obraze. Tato hrana vyjadřuje změnu v obraze, respektive funkci, kterou by obraz byl popsán. Hrana je dle

matematického popisu dána gradientem této funkce, což je první derivace. Z matematické definice tedy vyplývá její dvourozměrná velikost složená z hodnoty funkce a jejího směru. Témata hran budou pokračovat níže při metodě detekce hran [21].

Histogram patří mezi potřebné informace o obraze. Jedná se o histogram jasu, který má rozpětí jasu dle rozsahu řešeného snímku. Histogram lze použít jako ukázku hustoty pravděpodobnosti rozdělení jasu v obraze. Pro uživatele je pod histogramem viditelný sloupcový graf s celkovými informacemi o jasu obrazu. Dle histogramu můžeme provádět segmentaci obrazu na pozadí a objekt. Práce s histogramem je závislá na hodnotách jasu. Při konstantním jasu prostředí je histogram pro hledaný objekt obdobný i při pohybu tohoto objektu. Na základě hodnot histogramu mohou vznikat lokální minima či maxima, která v některých případech způsobují chyby a je vhodné je odstranit neboli filtrovat. Na tuto filtraci se hodí metoda klouzavého průměru, kde je brán ohled na okolí v daném histogramu [21].

Pro barevný obraz je nutné provést buď nějaké úpravy nebo počítat s více výsledky. Při předpokladu barevného obrazu jsou k dispozici jak funkce obrazu, tak hodnoty jasu pro jednotlivé složky barev. Barva je vlastností materiálu odrážet světlo o určité vlnové délce. Pro práci s digitálním obrazem se pracuje jen s odrazem některých barev, pro zpracování odrazu všech barev by bylo zpracování příliš složité. Digitální obraz tedy pracuje pouze jen s vybranými barvami tvořícími úzké části spektrálního pásma, a to pro každý sensor jedno pásmo. Dle využití jsou voleny počty sensorů. Nejčastěji používané je složení RGB, kde se využívá spektrálních složek pro vlnovou délku červené, zelené a modré barvy. Smícháním všech složek vznikne barevný obraz. Pro práci s tímto obrazem se výstupy jednotlivých barev berou zvlášť a jen v případě zjištění některých závislostí mohou být provázány funkcemi. Při zpracování je vhodné vzít v úvahu větší množství dat než při práci s černobílým obrazem a také podobnost těchto dat pro dané barvy [21].

Mezi dalšími úpravami obrazu můžou být transformace. Mezi geometrické transformace patří korekce chyb kamery, transformace má za cíl přepočet umístění pixelů do čtvercové mřížky z různých mnohoúhelníků či zakřiveného 3D obrazu. Transformace jasu naopak umožňují přizpůsobení stupnice jasu lepší práci a vyhlazení některých chybových hodnot. Transformace jasu vznikají i v závislosti na vhodném osvětlení scény zobrazené na snímku [21].

2.2.2 Rozpoznání pohybu v obraze

Na rozpoznání pohybu v obraze lze použít více různých metod, které se liší vhodností použití k jednotlivým kombinacím pohyblivosti obrazu a kamery. Během snímání obrazu software rozpoznává pohyb kamery a pohyb jednotlivých objektů scény. Mezi metody můžeme zařadit odečítání pozadí, porovnání snímků scény či optický tok. Každá z metod

vykazuje pro program určitou chybu dle zadaných parametrů, získaného obrazu a jeho změn.

Odečítání pozadí

Při použití této metody se předpokládá neměnné pozadí. Tohle pozadí se zaznamená do paměti. Následně získaný obraz je složený ze dvou částí, již dříve zjištěného pozadí a objektů navíc, které toto pozadí překrývají. Objekty v popředí jsou určeny k dalšímu zpracování obrazu [22].

Pro odečítání pozadí je ideální obraz získaný ve stupních šedi, pokud je obraz barevný, tak se pozadí odečítá pro každou barvu zvlášť jak bylo již výše uvedeno. Barevný RGB obraz lze pomocí příkazů programovacích jazyků převést na stupně šedi. Po odečtení pozadí od aktuálně řešeného snímku je získán binární obraz objektu či objektů v popředí. Tento získaný obraz je vhodný pro následné zařazení objektu do skupin nebo pro porovnání s následujícím snímkem pro zjištění směru pohybu objektu.

Při této metodě mohou chyby vznikat při výrazných změnách pozadí scény, hlavně z hlediska jasu, který je používán při stupních šedi. Metoda je poměrně jednoduchá a vhodná pro téměř neměnné oblasti jako jsou interiéry firem, kde se předpokládá provoz za umělého osvětlení. Při použití v exteriéru by bylo vhodné pozadí měnit v závislosti například na změnách počasí, či denní doby. Toto pozadí by nahradilo to původní a bylo by zaznamenáno během doby, kdy v prostoru nebyly žádné sledované objekty.

Porovnání snímků

Tato metoda slouží k porovnání snímku předchozího s následujícím. Během porovnání je zjištěn pohyb všech objektů daného snímku. Tento pohyb může být v rámci vyhodnocení chybný jak z hlediska pohybu kamery, tak pohybem nesledovaných objektů jako mohou být například větve stromu. Z hlediska pohybu kamery by bylo vhodné snímat vždy alespoň dva snímky při jedné pozici kamery ke zjištění pohybu v obraze [23].

Při porovnání lze srovnávat buď neupravené snímky nebo upravené. Ty upravené mohou být převedeny do odstínů šedi pro lepší následné zpracování nebo mohou být vyřazeny oblasti s předem definovaným jasem.

Metoda je odolná vůči změnám světla v dané scéně na rozdíl od metody odečítání pozadí za předpokladu, že se světlo nemění během snímání dvou porovnávacích snímků. Pokud je mezi porovnávanými snímky rozdíl světla, celé porovnání bude chybové. Nevýhodou dané metody může být zaznamenaný pohyb pro veškeré objekty snímané scény. Následné řazení objektů do skupin je složitější právě o pohyb nesledovaných předmětů.

Prahování

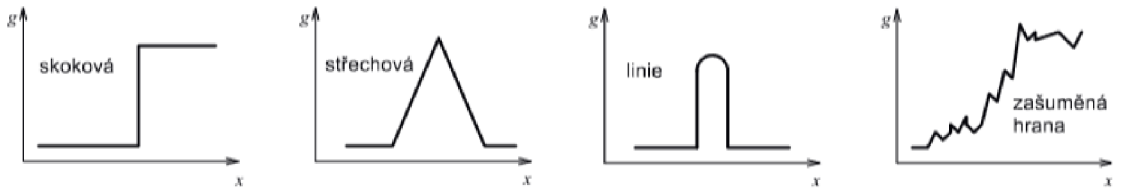
Tato metoda se používá nejčastěji pro obraz převedený na stupně šedi. Tento obraz se poté na úrovni této šedi srovnává s funkcí pro úpravu prahováním. Při porovnávání všech hodnot v matici obrazu s jednou hodnotou prahu se jedná o globální prahování, při porovnávání s více hodnotami dle umístění v obraze se jedná o adaptivní prahování. Výsledkem je binární obraz [24].

Zjišťování hodnoty pro porovnání je možné na testovaných obrazech se zjištěním stupně šedi v histogramu pro vybrané snímky.

Metoda je velmi vhodná pro detekování objektu na rozdílném podkladu. Pokud je objekt v kontrastu k tomuto podkladu, je metoda spolehlivá, ale při malém kontrastu může docházet k chybám. Pro zjištění pohybu v obraze je potřebné použít následně další metodu, například porovnání binárních snímků nebo může být pohyb řešen jako zjištění objektu na pozadí o stále stejné hodnotě jasu.

Detekce hran

Tato metoda vychází z hledání hran, které jsou definované derivací dané obrazové funkce, jak bylo psáno výše. Pro analýzu obrazu mohou být podstatné oba rozměry, častěji je to však velikost hrany. Velikost hrany je přirovnávána k velikosti definované pomocí všesměrového lineárního Laplaceova operátoru, který představuje druhé parciální derivace obrazové funkce. Hrany v obraze hledané pomocí gradientů pro předpokládaný homogenní jas celého objektu se spojují v hranice objektů [21].



Obrázek 1 – Nejčastější jasové profily hran [21]

Pro jednorozměrné jasové profily lze dle gradientu určit jejich profil. Nejčastější profily jsou vidět na obrázku 1. Přičemž jen posledního profilu lze reálně dosáhnout, první tři jsou idealizované do profilu skokové a střechové hrany nebo tenké linie [21].

Metody detekce hran lze využít i při potřebu vyšší ostrosti obrazu, respektive jeho přechodů. Ke zjištění strmosti hran je možné použít jak gradientů, tak Laplaciánu. Pro vyšší ostrost je potřeba zvýšit strmost nalezených hran. Při zostření obrazu dojde i k navýšení kontrastu, což může být potřebné pro zlepšené definování objektů [21].

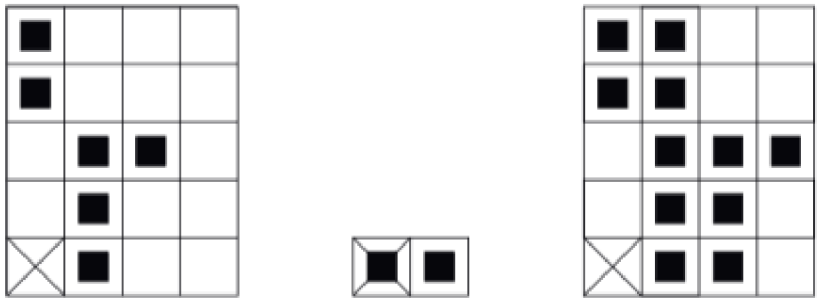
Jedním z nejznámějších detektorů hran je Cannyho detektor hran, který má jistá rozšíření a vylepšení oproti výše popsanému postupu. Základem detektoru je vyhledávání skokové hrany pomocí filtru. Funkce filtru je optimalizována dle tří základních kritérií, a to detekčního kritéria, které zajistí neopomenutí významných hran

a zároveň záznam hran jen s jednou odezvou, dále lokalizační kritérium, které zajišťuje minimální rozdíl mezi skutečnou a nalezenou polohou hrany, a poslední požadavek jedné odezvy, jenž slouží hlavně pro zašuměné nebo nehladké hrany. Při optimalizaci filtru je provedeno nahrazení za filtrací pomocí Gaussiánu. Tímto vzniká diferenciální operátor, který je schopen poskytnout informace i o orientaci hrany. Díky šumu v obraze je následně použito prahování, to však může zavinit vzniknutí nesouvislé hrany. Pro omezení šumu a zároveň zajištění souvislých hran je využito prahování s hysteresí. Posledním krokem detektoru je volba měřítka. Detektor používá vlastní syntézu na odezvy v různých měřítkách, proto je schopen leších výsledků [21].

Morfologické operátory

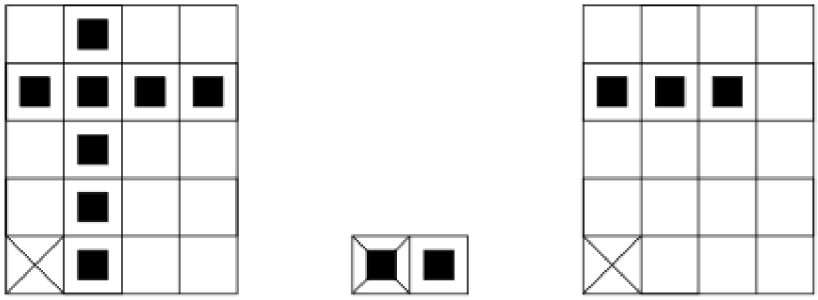
Pro práci s digitálním obrazem je předpoklad euklidovského prostoru a zpracování binárních obrazců. Pomocí operátorů je možné vyhladit šum a nepatrné pohyby, které jsou chybné pro vyhodnocení pohybu objektů. Po použití těchto operátorů vznikají nové obrazy vylepšené o odstranění častých chyb vyhodnocení [21].

Dilatace patří mezi první operátory. Zde dochází ke změně objektu na základě přiřazování jasu. Toto přiřazování je vázáno na jas sousedního pixelu, jedná se tedy o přiřazení nejvyššího jasu sousedního pixelu tomu, který řešíme. Názorný příklad dilatace je vidět na obrázku 2, kde vlevo je původní obrazec, vpravo poté rozšířený objekt. Z obrázku je vidět, že dilataci můžeme použít pro zaplnění menších děr či zálivů v objektu pro jeho lepší zpracování [21].



Obrázek 2 – Dilatace [21]

Dalším operátorem je eroze, která je duální operací k dilataci. Jedná se opět o úpravu na základě sousedních pixelů. Eroze však v obrazcích odstraňuje definovanou tloušťku a tím může složitější obrazce rozdělit na více jednodušších. Pomocí eroze lze snadno získat obraz objektu bez výplně (myšleno jen hranici) pomocí odečtení obrazce po erozi od obrazce před erozí. Příklad eroze je vidět na obrázku 3, kde původní objekt je vlevo a výsledný vpravo [21].



Obrázek 3 – Eroze [21]

Dále lze pomocí kombinace eroze a dilatace obraz otevřít či uzavřít. Výsledkem bude zjednodušený obraz bez některých detailů. Otevření je způsobeno provedením eroze a poté dilatace. Uzavření je potom dilatace následovaná erozí. Uzavření a otevření se využívá hlavně pro odmazání detailů menších, než je jeden element bez porušení celkového objektu. Na obrázcích níže lze vidět, že při otevření se objekt rozdělí na několik menších a při uzavření se naopak objekt stává jednolitým s možným vymizením drobných děr [21].

Obrázek 4 - Otevření (původní vlevo, výsledný vpravo) [21]

Obrázek 5 - Uzavření (původní vlevo, výsledný vpravo) [21]

2.2.3 Rozpoznání objektů

Pro zpracování a zařazení objektu do skupin použijeme segmentované snímky z předchozí části. Pro rozpoznání objektů slouží několik metod, všechny se ale shodují v předchozím uložení skupiny vzorových obrazců a pojmenování těchto skupin neboli

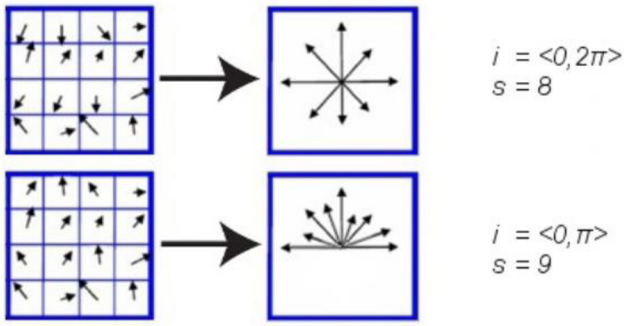
tříd. Vzory jsou poté použity pro porovnání buď celého tvaru, podobnosti částí nebo třeba jen některých významných prvků. Další možností je v dnešní době využívána tvorba neuronové sítě, kde za použití vzorových obrazců tvoříme učenlivý program na detekci objektů. Jednotlivé metody jsou popsány níže.

Metoda Histograms of Oriented Gradients (HOG)

Tato metoda vznikla hlavně z potřeby rozpoznávat lidské postavy v obraze. Vstupem může být, jak barevný obraz, tak obraz ve stupních šedi. Metoda je reprezentována gradienty s úvahou jejich směru i velikosti [25].

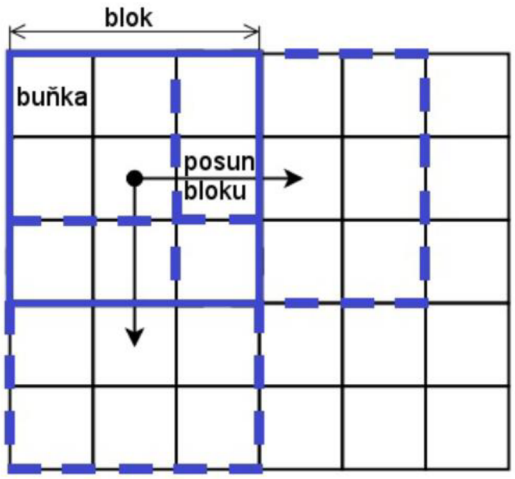
Pro určení gradientu byl použit Gaussovský filtr s různým parametrem a následně byly aplikovány masky. Jako nejlepší kombinace pro tuto metodu byla vybrána konvoluční Gaussova filtrace při parametru rovném nule a definovanou maskou [25].

Po získání obrazů pomocí Gaussova filtru s parametrem a maskou je vypočítána velikost a směr gradientu. Poté je vytvořen histogram orientací specifikovaný intervalem rozsahu směru *i* s nejčastější velikostí jedné kružnice nebo půlkružnice a počtem výsečí *s*, které rovnoměrně rozdělují interval rozsahu, což je vidět na obrázku 6. Přínos gradientů do jednotlivých částí histogramu je dán velikostí gradientu a zároveň bilineárním rozdělením mezi další části [25].



Obrázek 6 - Histogramy orientací pro různé hodnoty i a s [25]

Následně musí být histogram orientací normalizován z důvodů jasových transformací. Posledním krokem je získání deskriptoru této metody, a to rovnoměrným rozdělením vstupního obrazu do čtvercových polí o dané hraně z určitého počtu pixelů. Tyto pole jsou dále členěny do buněk o hranách menších než pro pole. Pro buňky jsou zjištěny normalizované histogramy orientací gradientů. Histogramy buněk jsou poté zprůměrovány a tvoří popis pro celé pole z těchto buněk. Po získání tohoto popisu se pole posune o daný počet pixelů a proces je zopakován. Závěrečný deskriptor této metody je složen zřetězením všech popisů polí uvnitř daného obrazu na vstupu. Dle dostupné literatury se nejčastěji pole či bloky zobrazují jako čtvercové, což je vidět s posunem na obrázku 7, výjimečně potom kruhové [25].

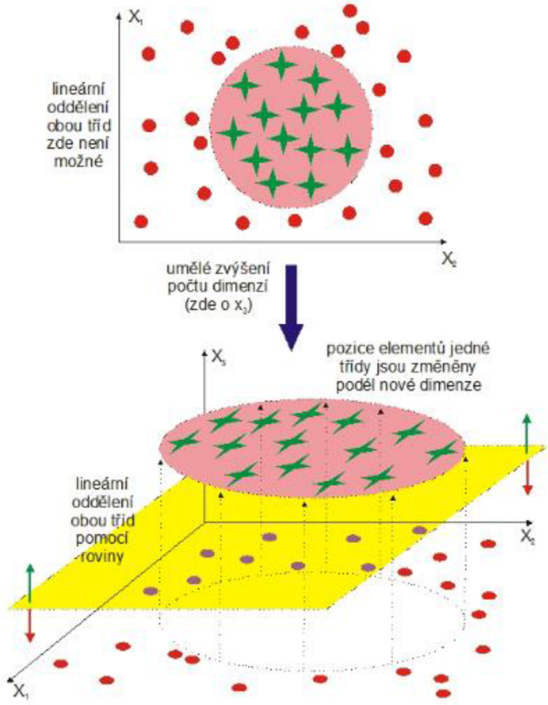


Obrázek 7 - Princip konstrukce deskriptoru HOG [25]

Algoritmy podpůrných vektorů

Metoda spadá do skupiny jádrových algoritmů ty se vyznačují především využitím výhod efektivních algoritmů pro vyhledání lineárních hranic a zároveň mohou reprezentovat značně nelineární rovnice. Základním smyslem metody je převést vstupní prostor do jiného (vícedimensionálního), kde je snazší dělit třídy lineárně.

Na obrázku 8 je vidět princip této metody přidáním dimenze, kdy v původní dvourozměrné dimenzi je rozdělení nelineární a přidáním dimenze můžeme přidat bodům další souřadnici a vytvořit pro ně novou rovinu posunutou po ose nové dimenze. Poté je potřeba určit umístění lineární hranice pro efektivní práci s dalšími daty a následná optimalizace této hranice. Metoda poskytuje přidáním dimenze vždy možnost získat lineární hranici separací do nadroviny. Protože při vysokém počtu dimenzí vznikají nepřesnosti stejně jako při prokládání bodů polynomem vysokého řádu, je potřeba hledat optimalizované řešení pro co největší mezeru mezi daty v rovinách. Vzniká tak optimální lineární oddělovač optimalizovaný kvadratickým programováním, kde lze efektivně najít globální maximum a skalární násobky dvojic tvořící definice podpůrných vektorů [26].



Obrázek 8 - Princip vzniku lineárního řešení přidáním dimenze [26]

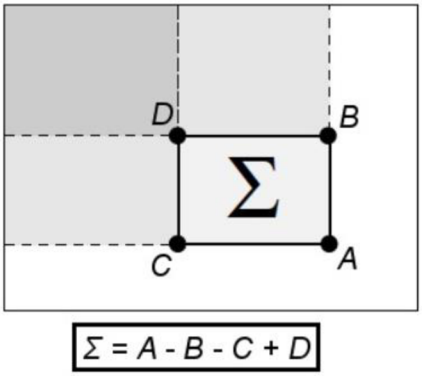
Zjištěné lineární oddělovače je možné zpětně vložit do původního prostoru, kde vytvoří libovolně zvlněné a nelineární hranice mezi body definovanými jako pozitivní (hledané) a negativní. Častým využitím může být aplikace na ručně psaný text s množstvím odlišností [26].

Metoda Speeded-Up Robust Features (SURF)

Hlavním záměrem této metody bylo vytvoření výpočetně rychlého a stabilního deskriptoru na detekci v reálném čase. Oproti jiným metodám je zrychlena sestavením měřítkově nezávislého reprezentování obrazu a redukcí velikosti výsledných deskriptorů [27].

Metoda využívá k detekci hran Gaussovy funkce, kde ale vznikají odezvy i mimo hrany, ty se následně odstraňují použitím Hessianu. Tyto kroky jsou sjednoceny a detekce probíhá přímo pomocí determinantu Hessianovy matice. Svižný výpočet determinantu je umožněn použitím integrálního obrazu [27].

Integrální obraz tvoří strukturu vybudovanou nad zadaným obrazem a umožňuje svižné součty hodnot jednotlivých částí. Výpočet hodnot je viditelný na obrázku 9, kde hodnoty písmen A až D jsou hodnoty integrálního obrazu na daných souřadnicích [27].

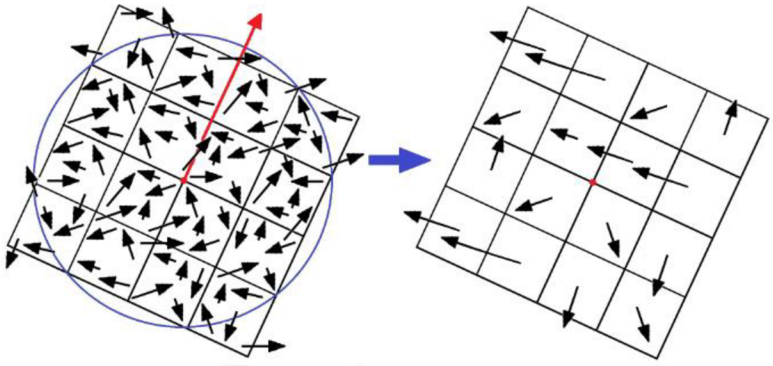


Obrázek 9 - Výpočet součtu hodnot pro danou oblast [27]

Pro výpočet Hessianovy matice jsou použity obdélníkové funkce, ty aproximují druhou derivaci v diskrétní formě, což je konvoluce obrazu s filtry. Odezvy na tyto obdélníkové filtry jsou normalizovány pro zachování váhových hodnot pro všechna měřítka. Váha je použita pro vhodnou aproximaci Gaussova jádra [27].

V rámci detekce jsou významné body hledány pomocí maxima. Pokud tedy má bod ve svém okolí maximum, je považován za významný a jeho lokace je vylepšena použitím Taylorova rozvoje, což způsobí určení sub-pixelové a sub-měřítkové polohy tohoto bodu [27].

Metoda SURF také určuje orientaci významného bodu, ale na rozdíl od jiných metod nevyužívá histogramu orientací, ale odezvu na Haarovu vlnku v kruhovém okolí. Pro navýšení rychlosti výpočtu je Haarova vlnka aproximována opět obdélníkovými filtry. Poté jsou získány informace o orientaci včetně hodnot pro odolnost vůči světelným změnám obrazu s využitím dominantního směru dle obrázku 10 [27].



Obrázek 10 - Odezvy na Haarovu vlnku (vlevo) a výsledný deskriptor (vpravo) [27]

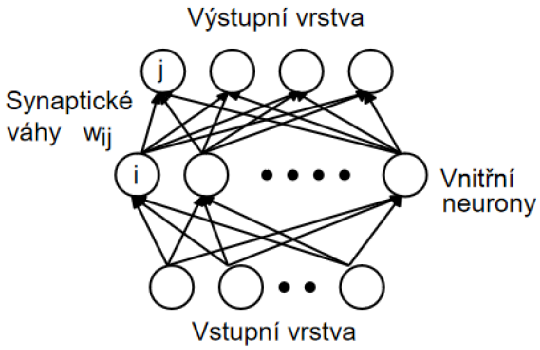
Neuronové sítě

Neuronové sítě se shodují ve snaze o napodobení chování nervových soustav živých organismů, které nepoužívají naprogramované algoritmy. Při učení se tedy vytváří vlastní algoritmus, který se postupem času mění a vyvíjí dle optimálního řešení dané

úlohy. Přenos těchto informací je pojat spíše jako globální, něž lokální. Pro vylepšení funkčnosti jsou posilovány vazby se správnými výsledky a vazby pro špatné výsledky jsou naopak oslabovány opakovaným trénováním sítě. Naučené vlastnosti těchto sítí jsou obdobné jako pro organismy, kdy výsledky vychází z předvolené trénovací skupiny [28].

Jeden z nejčastěji používaných modelů je perceptron. Tato metoda se dá vysvětlit jako postup neuronů mezi hladinami, při dosažení určité hodnoty (obvykle překročení prahové hodnoty) postoupí na vyšší hladinu v opačném případě klesne. Dle tohoto principu jsou v programu přiřazovány logické hodnoty. Matematicky je metoda popsána pomocí aktivační funkce například signum. Jedna ze známých metod pro adaptaci neuronu je Hebbovo pravidlo popsané několika kroky. Prvním krokem je inicializace vah a prahu, dalším je dodání vstupu a požadovaného výstupu trénovací množiny dat. Dalším postupem je stanovení skutečné odezvy a adaptace, společně s adaptací vah ovlivněnou faktorem učení, který ovlivňuje adaptaci. Naposled se do adaptace přidá chyba určení správného výsledku [28].

Dalším modelem neuronové sítě je vícevrstvá síť s metodou backpropagation. Metoda se zakládá na propojení perceptronů. Topologie propojení je vidět na obrázku 11. Z topologie je viditelné členění vrstev na vstupní, výstupní a vnitřní vrstvy, kterých může být několik dle složitosti neuronové sítě. Pro každý neuron nižší vrstvy platí, že má spojení ke všem neuronům vyšší vrstvy. K důležitým vstupům patří trénovací množina s danou problematikou pro definování synaptických vah ve vrstvách [28].



Obrázek 11 – Vícevrstvá neuronová síť [28]

2.2.4 Sledování objektů

Při sledování objektů je využito vždy nějaké funkce sloužící k detekci pohybu, popřípadě i funkce rozlišení objektů, aby byl sledován správný shluk bodů. Metody jsou vhodně zvoleny pro svižnou reakci systému. Pokud by byl program s metodami z předešlých kapitol výpočetně značně náročný, docházelo by ke zpoždění odezvy při sledování.

Způsob sledování se liší dle použitého hardwaru a požadavků na umístění sledovaného shluku bodů. Pokud je hardware statický bývá využito jen orámování daných bodů a posun tohoto rámu po zaznamenávané ploše obrazu. Pro dynamické

sledování je potřeba určit, zda k posunu kamery dochází v případě, že se objekt dostane k okraji zaznamenávaného obrazu, či chceme objekt udržet v určené zóně, ideálně kolem středu záběru.

Když je pohyb kamery vyvolán, až při kontaktu daného objektu s okrajem obrazu, může dojít ke ztrátě objektu. Tuto ztrátu by způsobila velmi pomalá pohybová odezva hybného pohonu kamery. V tomto případě je třeba brát v úvahu zpoždění součástek s vhodnou regulací dojezdu na definovanou pozici.

Při použití hlídané zóny vhodně umístěné v obraze, například zabírající polovinu snímku vycentrovanou na střed, se objekt neztratí z vyhodnocovaného snímku a hybný pohon kamery má čas na odezvu. Opět je potřeba tuto odezvu doplnit regulací polohy pro dojezd do zadané polohy. Tato metoda bude nevýhodná, pokud se bude systém snažit umisťovat objekt do středu snímku, ale objekt samotný se bude pohybovat tam a zpět přes hranici zóny vyšší rychlostí.

Pro obě metody je potřeba převést velikost pohybu pohonu na pohyb snímku. Tento pohyb je třeba definovat v programu a ovlivnit jej vhodnou regulací, například PI nebo PID regulátor. Regulátor bude tvořen programem nebo může být součástí pohonu pomocí daných elektronických součástek.

3 Řešení

V této kapitole budou zvoleny použité programy s knihovnami, metody rozpoznávání a sledování včetně vhodného testovacího hardwaru.

3.1 Vyhodnocení prostředí

Pro tuto práci bylo zvoleno prostředí Matlab z důvodu jednoduché syntaxe programu, snadného a přehledného editování kódu s následnou možností práce v reálném čase, či převodem kódu do jiného programovacího prostředí pro lepší kompatibilitu. K další výhodě patří možnost tvorby uživatelské aplikace pro jednodušší použití zapsaného algoritmu a snížení vzniku chyb z důvodu jeho nevhodného přepsání.

Prostředí nabízí spoustu vhodných knihoven zmíněných v předchozí kapitole. Je tedy možné nabídnout uživateli přijatelné ovládací prostředí včetně vhodných funkcí zpracování obrazu, či jeho uložení pro další pozdější využití.

V novějších verzích programu lze při tvorbě algoritmu využít kontroly zapsané syntaxe a funkcí po jednotlivých menších úsecích či editace při zapnutém algoritmu. Úpravy jsou tak mnohdy přehlednější a svižnější. Samotný Matlab svým uživatelům nabízí pomoc při opravě syntaxe a rozsáhlou podporu s mnoha příklady včetně poskytnutých funkčních kódů.

3.1.1 Použité knihovny

Některé vhodné knihovny byly popsány již výše. Obsahem této podkapitoly je výčet použitých knihoven s jejich využitím v algoritmu. Dále byly použity základní funkce pro cykly, které nejsou zahrnuty ve výčtu použitých knihoven. Výčet knihoven se vztahuje na specifické funkce a příkazy.

MATLAB Coder Support Package for NVIDIA Jetson and NVIDIA DRIVE Platforms

Knihovna zajišťuje provázání hardwaru se softwarem. V této práci byly využity příkazy pro tvorbu snímků. Tyto snímky jsou tvořeny daným příkazem Matlabu a uloženy ve Workspace ve vhodném přednastaveném formátu [29].

Data Import and Analysis

V této práci byla knihovna využita pro zpracování záznamu videa. Dále knihovna disponuje příkazy pro vizualizaci a zpracování dat [30].

Deep Learning Toolbox

Tato knihovna je specifická pro práci s pokročilým zpracováním obrazu. Balíček je určen pro tvorbu nových a využití již definovaných neuronových sítí. Nabízí využití více druhů sítí, v přednastavených jsou k dispozici hlavně konvoluční neuronové sítě. Nabídka přednastavených sítí v balíčku je provázána s jednotnou sadou obrázků pro trénink a testování sítě [31].

Pro tvorbu nových sítí jsou k dispozici přednastavené aplikace pro snazší orientaci v tréninku či přímo příkazy na načtení a klasifikaci objektů pomocí již vytvořených sítí [31].

Image Processing Toolbox

Tento toolbox slouží především pro zpracování obrazu, vizualizaci, tvorbu algoritmů. Obsahuje množství přednastavených funkcí v této oblasti pro snadnější použití. Úpravy zahrnují segmentaci, geometrické transformace či filtry. Umožňuje zpracování obrazu ve dvou a třídimenzionálním prostoru a možnost zpracování velkých obrázků.

Pro zpracování obrazu a videa nabízí interaktivní aplikace. Některé funkce jsou v rámci knihovny podporovány kódy v jazycích C/C++ a dále umožňuje práci s grafickými procesními jednotkami. Popis knihovny vychází ze zdroje [32].

V algoritmu byly využity funkce pro zobrazení snímků do uživatelské aplikace.

App Building

Tato skupina příkazů a funkcí obsahuje předdefinované možnosti pro tvorbu uživatelských aplikací. Tvorba uživatelských aplikací je rozdělena na dvě základní použitelná prostředí, a to *GUIDE* a *App Designer*. Tvorba v těchto prostředích je možná s využitím aplikace a následným automatickým generováním rozsáhlého kódu nebo ručním použitím jednotlivých funkcí specifických pro dané prostředí, což může významně zkrátit délku napsaného programu.

Pro uživatelskou aplikaci bylo využito ručně psaného kódu s prvky staršího prostředí *GUIDE*. Mezi použité prvky patří okno aplikace s definovanou velikostí a umístěním na obrazovce, interaktivním výběrovým menu pro určení hledané skupiny objektů, tlačítek s definovanou odezvou (zapnutí a vypnutí algoritmu) a okna pro zobrazení snímku s jeho pojmenováním [33].

3.2 Rozpoznání pohybu

Běžně se v algoritmech rozpoznává pohyb různými metodami zmíněnými v kapitole výše. Tato práce však využívá opačné logiky detekce objektů, a proto rozpoznání pohybu není přímo řešeno v algoritmu. Daný algoritmus prvotně rozpoznává objekt a následně jej sleduje při jeho pohybu. Pro tento algoritmus je předpokládáno, že pohyb není podstatný pro vyhledání objektu. Pohyb objektu je hlídán sledováním.

3.3 Rozpoznání tvaru

Rozpoznání tvaru bylo řešeno využitím neuronové sítě. Dostatečně velká síť je schopna rozeznávat obsáhlé množství skupin nebo prostředí a tím zvýší možnosti hledaných druhů objektů. Tyto sítě jdou využít pro velké množství případů nebo se dají definovat dle požadavků konkrétního použití, kde je možné vytvořit a naučit neuronovou síť žádané skupiny a specifické předměty, zvířata či lidé [34].

V rámci této práce bylo využito již vytvořené neuronové sítě pro prostředí Matlab. Síť je provázána s knihovnou *Deep Learning* Toolbox. Tato síť je schopná rozřazování do definovaných skupin, dle tréninkového setu, který je primárně využíván programem Matlab a to set snímků ve skupinách *ImageNet*. Tento set umožňuje neuronové síti tisíc různých zařazení objektů, a to v rámci pojmenování předmětů (ořezávátko, dalekohled, banán), zvířat (jednotlivé druhy, pro rozšířenější zvířata jako například pes jsou k dispozici jednotlivá plemena) či místa (druhy obchodů). Pro rozpoznání člověka je potřeba využít jiný set dat [34].

Použita byla přednastavená konvoluční neuronová síť *SqueezeNet*. Tato síť zpracovává vstupní snímek o velikosti 227x227 pixelů se třemi kanály pro barevné vstupy, má 18 vrstev a je tedy dostatečně rychlá pro práci v reálném čase. Při použití předtrénované sítě je možné využít příkazu *net* jako tomu bylo v této práci [34].

3.4 Sledování objektů

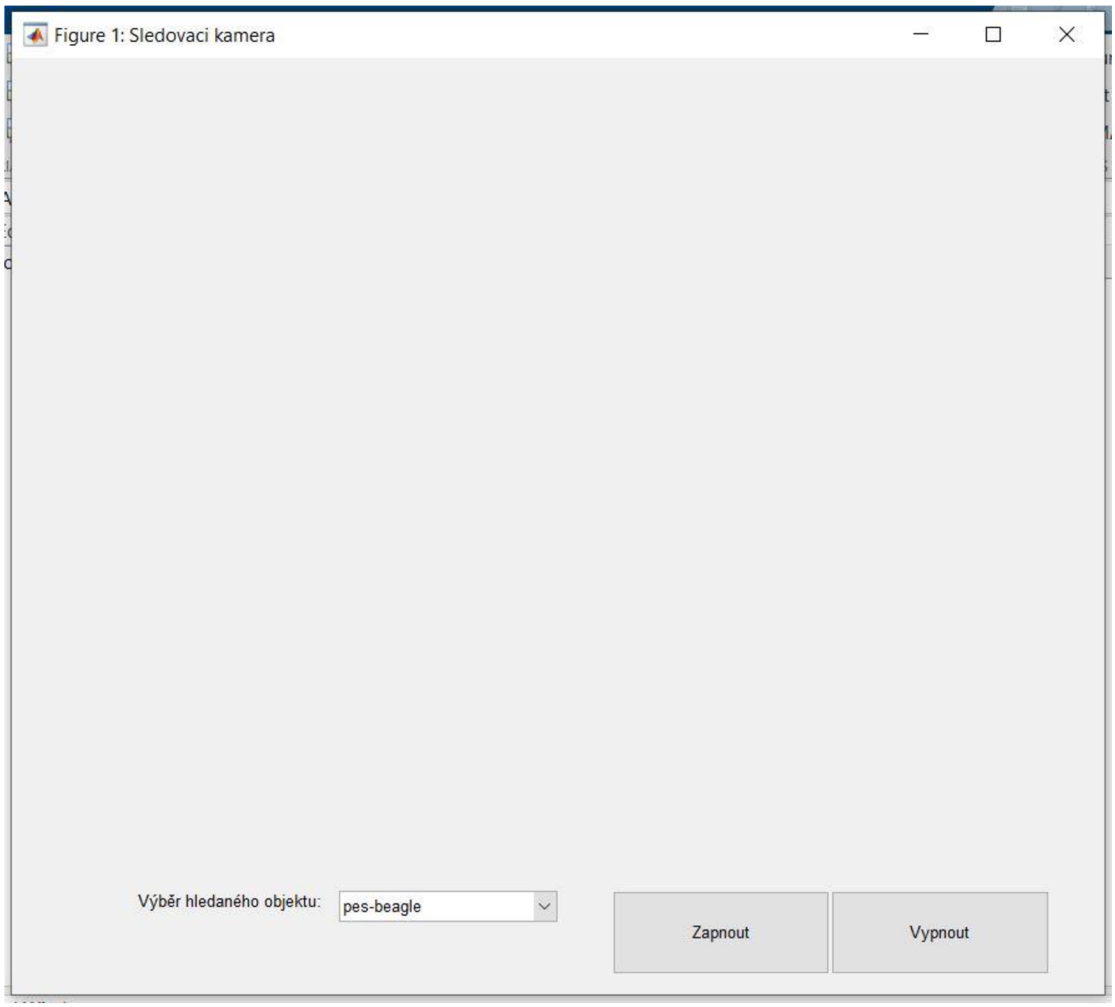
Pro sledování byla vytvořena zóna v obraze. Pokud je sledovaný objekt v této zóně, nevzniká požadavek na posun kamery, pokud se objekt začne posouvat mimo zónu, tak vzniká požadavek na posun. Vytvořená zóna na středu je přizpůsobena pevné velikosti snímku pro rozpoznání vybranou neuronovou sítí. Posun při kontaktu s hranicí zóny je v programu prezentován nápisem nad aktuálním snímkem kamery v uživatelské aplikaci, které poukazují na žádaný pohyb kamery. Požadavek na posunutí se nezobrazuje v případě nenalezení hledaného objektu na daném snímku.

V rámci této práce nebyla vytvořena žádná konstrukce. Aplikace byla spuštěna v reálném čase v programu Matlab při využití integrované webkamery. Integrovaná kamera měla rozlišení 480x640 pixelů, což bylo zohledněno při tvorbě hlídané zóny. Při použití jiného rozlišení je možné zkreslení obrazu při detekci či nezahrnutí vrchního okraje do sledovaného záběru. Sledování je tedy ovlivněno manuálním posunutím operátorem.

4 Reálné ověření programu

Při reálném ověření byl pomocí aplikace vyhledáván pes plemene beagle. Na obrázcích níže jsou vidět jednotlivé případy vyhledávání s odezvou aplikace a celkový vzhled této aplikace spolu s popisem jednotlivých funkčních částí.

Aplikace se spouští po uložení skriptu a propojení cesty ke skriptu v Matlabu pomocí názvu *sledovac* zadaného přímo do *Command Window* (příkazového řádku programu) nebo po otevření skriptu pomocí tlačítka *Run*. Při spuštění uživatelské aplikace se zobrazí vyskakovací okno. Tohle okno má definovanou polohu na monitoru a zadanou velikost v algoritmu. Vyskakovací okno je vidět na obrázku 12. V prostředí Matlab vytvoří při zobrazení vyskakovacího okna soubor ve své paměti o definovaných velikostech z podrobností o aplikaci uvedených ve skriptu v části definování podrobností pro interaktivní a neměnné části této aplikace.

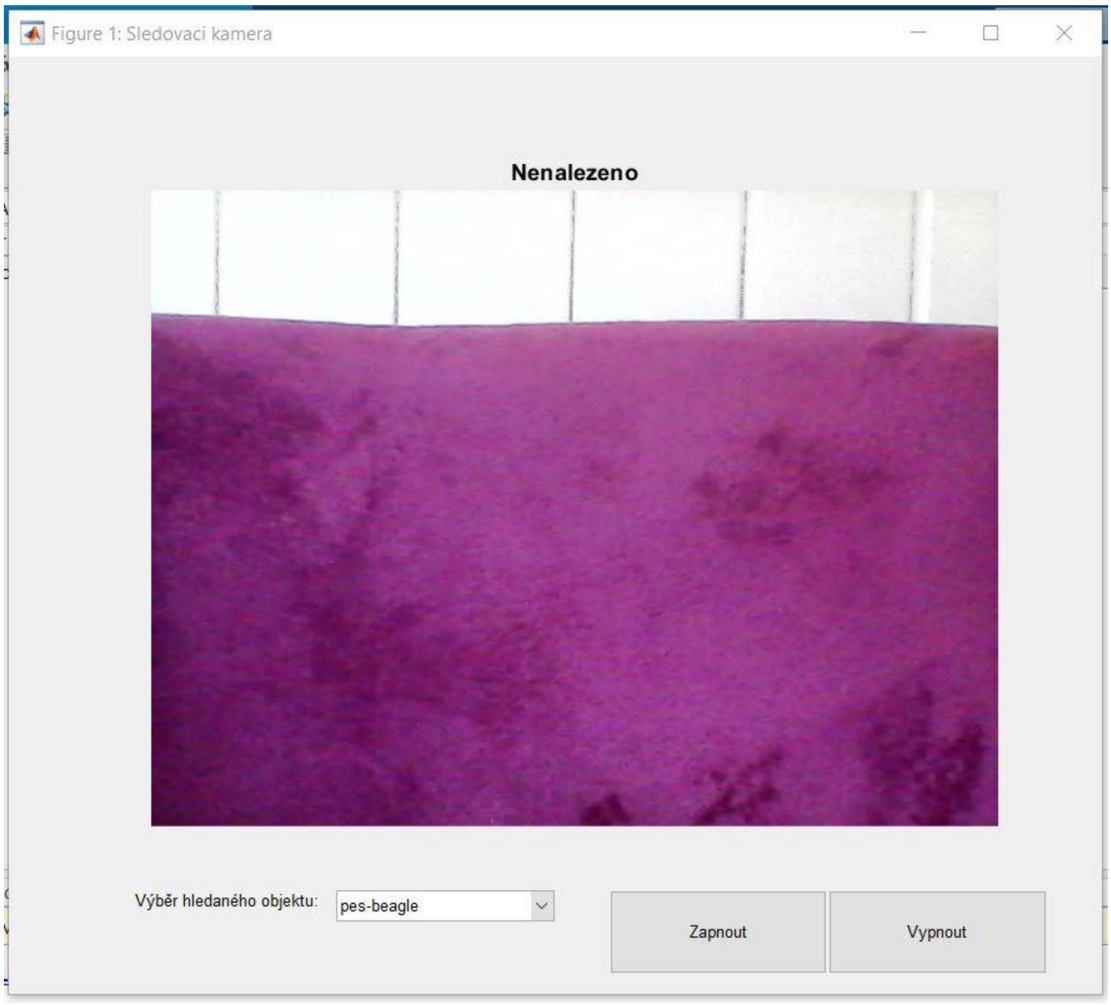


Obrázek 12 - Základní okno aplikace

V tomto základním okně je potřeba vybrat název hledaného objektu. Výběr je prováděn kliknutím na název objektu z výběrového menu. Aktuální verze aplikace

pro ilustraci nabízí tři skupiny hledaných předmětu a to pes-beagle, banán a sluneční brýle. Tyto skupiny lze v algoritmu jednoduše vyměnit nebo výběr rozšířit o další možnosti dle specifických tříd v trénovacím setu neuronové sítě. V případě nevybrání objektu kliknutím bude při spuštění algoritmu vybrán objekt pes-beagle. Hledané objekty je možné měnit při zapnutém algoritmu řešení.

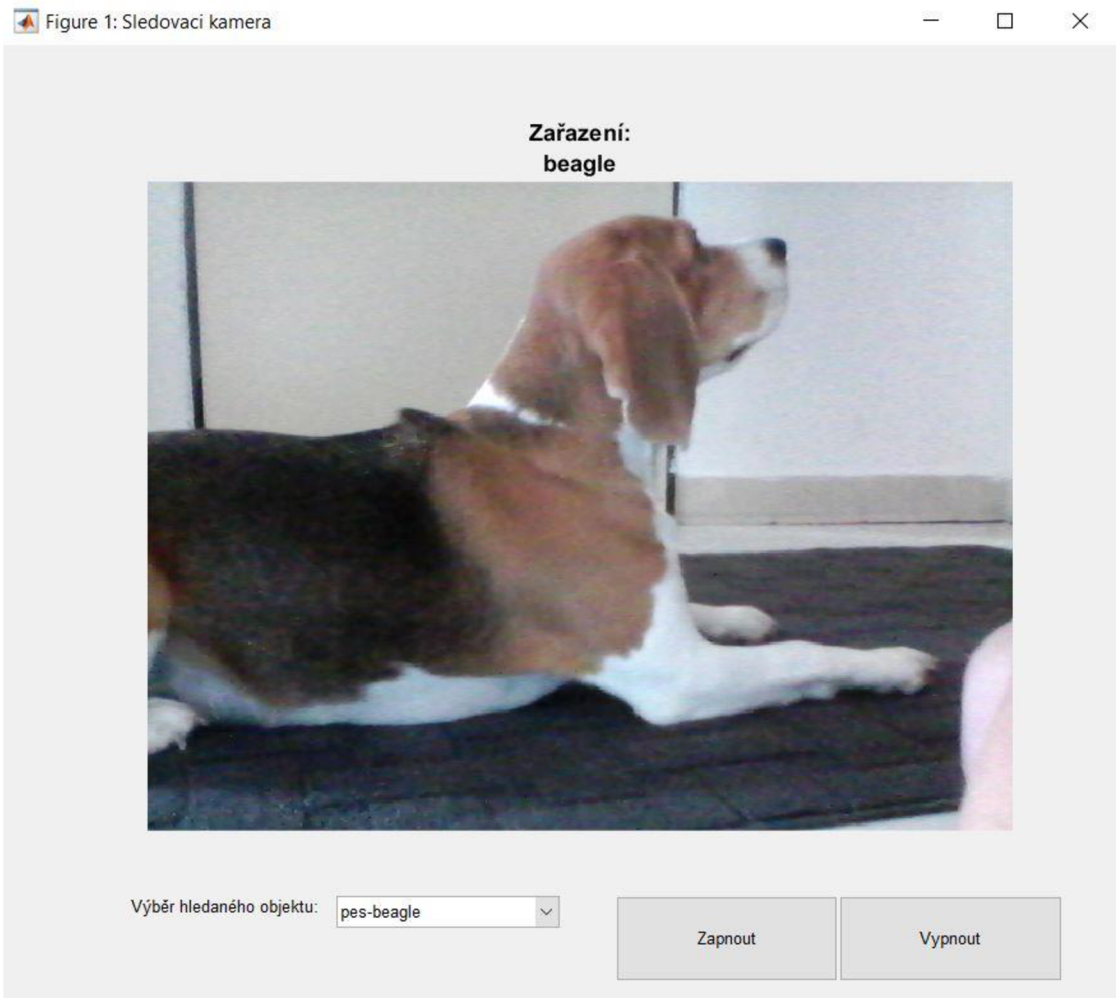
Po kliknutí na tlačítko *Zapnout* se spustí algoritmus a v aplikaci se zobrazí aktuální snímek z připojené kamery, což je vidět na obrázku 13. Na obrázku 13 nad aktuálním snímkem je vidět, že hledaný objekt nebyl nalezen, což je prezentováno nápisem *Nenalezeno* nad zobrazeným snímkem.



Obrázek 13 - Okno aplikace bez detekovaného objektu

Aplikace dále nad snímkem zobrazuje nápis *Zařazení:* a pod ním je viditelná klasifikace objektu na snímku, který je pro algoritmus označen jako sledovaná zóna, popřípadě jedna z okrajových stran a zároveň je to hledaný objekt. Název skupiny je uveden v anglickém jazyce, protože základní set obrázků je vytvořen v tomto jazyce a pro pojmenování v českém jazyce by bylo nutné přidat do algoritmu překlady jednotlivých tříd. U aktuálního setu obrázků by se jednalo o překlad či přejmenování

tisíce tříd. Při překladu by se tedy jednalo buď o novou tvorbu trénovacího setu, což nemusí vadit při tvorbě nové sady vhodnější k danému využití, nebo manuální dopsání překladu jednotlivých tříd setu, což by v tomto případě významně prodloužilo algoritmus.

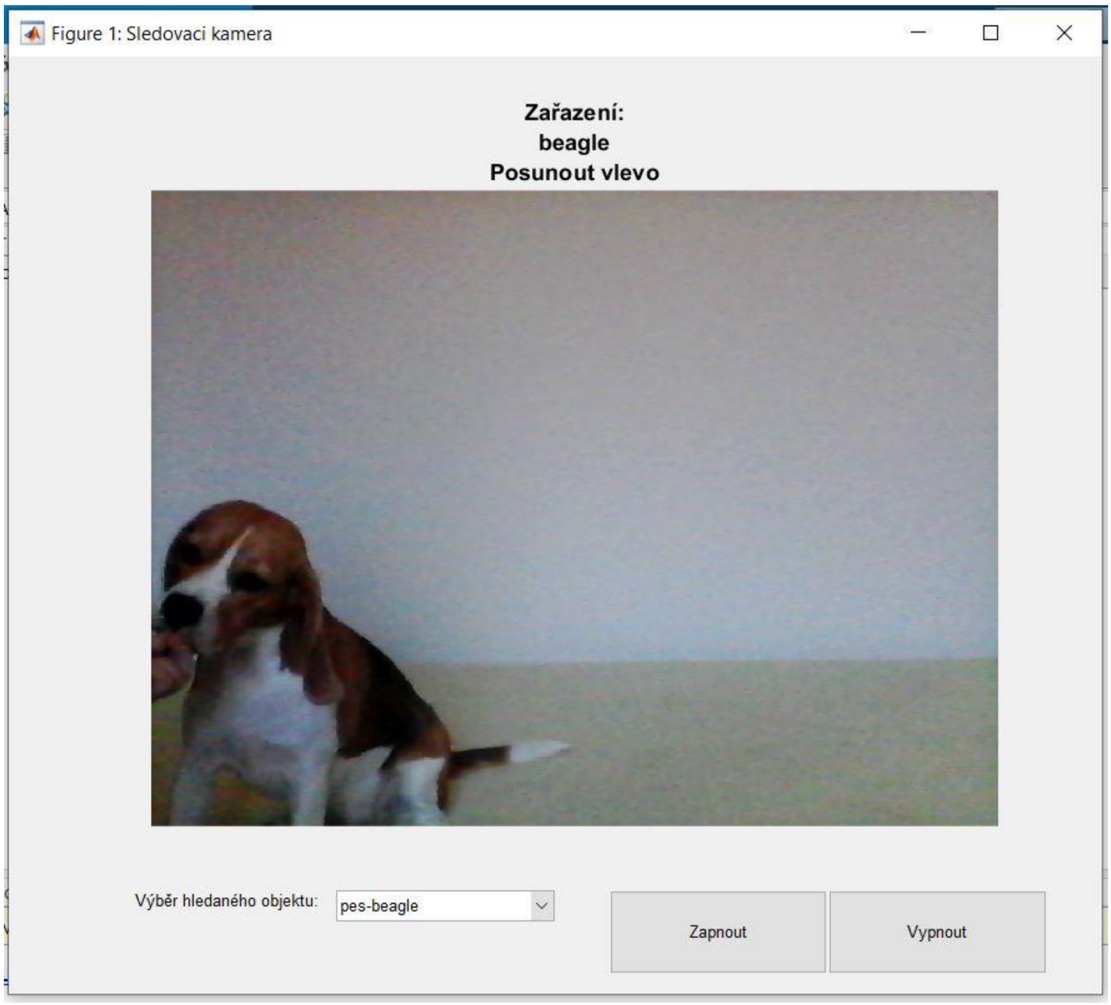


Obrázek 14 - Okno aplikace s hledaným objektem v zóně

Na obrázku 14 je vidět pes, který je správně detekován ve sledovací zóně algoritmu. Aplikace v horní části vypsala základní anglický název skupiny *beagle*. Řádek, na kterém byla dosud informace o nenalezení objektu, je zaměněn za informaci o zařazení.

Při možném vylepšení algoritmu a uživatelské aplikace by bylo možné nahradit aktuální menu pro výběr skupin polem pro výběr více možností najednou. Více možností by potom bylo definováno buď jako skupiny se stejnou důležitostí nebo by se musela přidat další část, kde by se definovala priorita pro hledání jednotlivých skupin objektů. Algoritmus by se tak mohl rozšířit na hledání několika objektů najednou. Po následném nalezení by se pomocí sledování již algoritmus přeladil na hlídání prvního nalezeného objektu, u kterého by poté následovaly kroky jako v aktuální aplikaci. Popřípadě by se

obraz mohl dále detekovat pro vyhledání objektu s vyšší prioritou. U tohoto vylepšení by však bylo nutné brát ohled na náročnost algoritmu, aby byl nadále schopen fungovat v reálném čase bez výrazného opoždění.

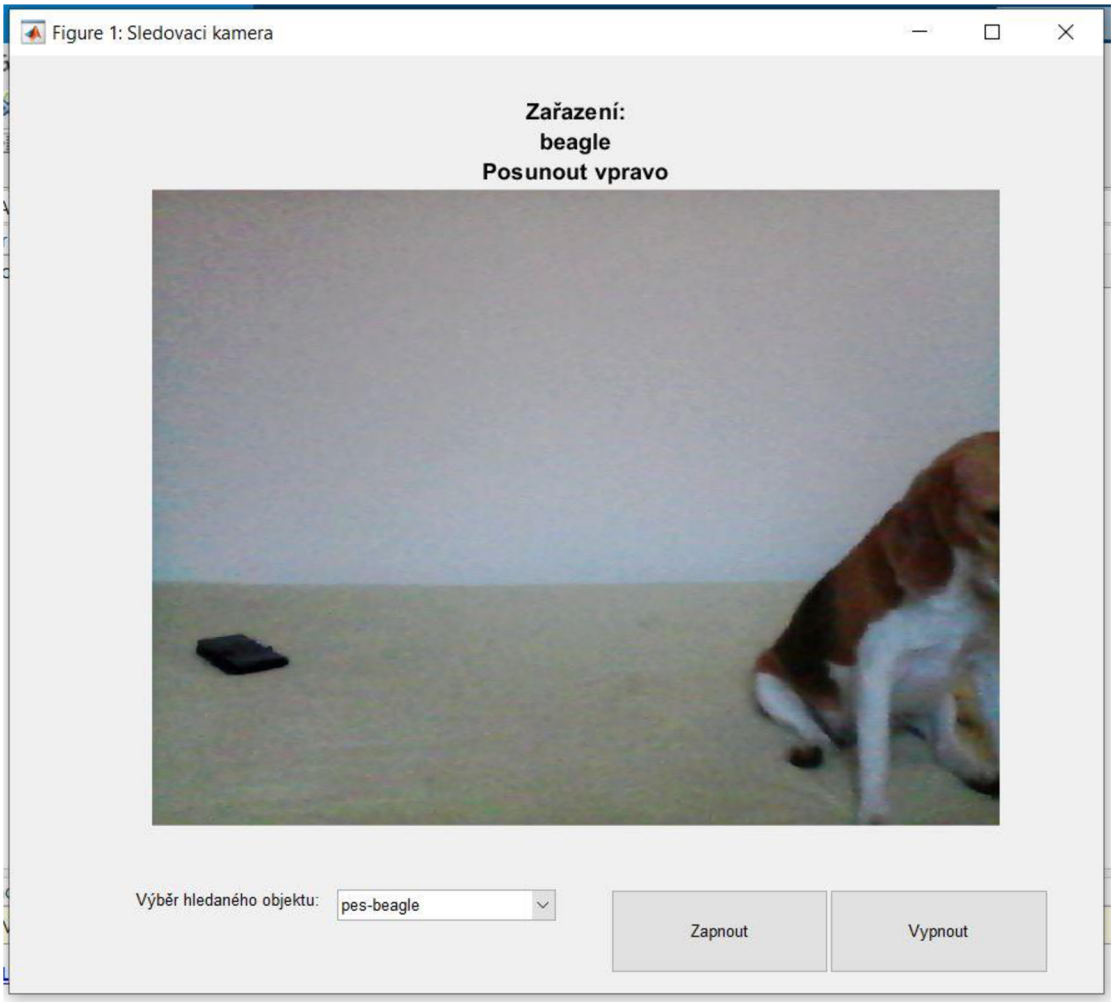


Obrázek 15 – Okno aplikace s objektem pro posun vlevo

Na předchozím obrázku 15 a následujícím obrázku 16 jsou vidět další funkce aplikace. Jedná se o požadavky na posun kamery. Tento posun je potřebné pro aktuální hardware provést ručně. V případě obrázku 15 se objekt přesunul z hlavní sledovací zóny na levý okraj snímku, a proto je nad aktuálním snímkem zobrazen řádek s nápisem *Posunout vlevo*. Aplikace klasifikuje objekt v levé okrajové části, proto je zobrazena třída objektu *beagle*. Podobně je tomu na obrázku 16, kde dochází k přesunu objektu k pravému okraji. Opět je požadavek na posun prezentován nápisem tentokrát *Posunout vpravo*.

V případě vylepšení by se jednalo o rozšíření aplikace na hardware, který je schopen samostatného posuvu. Pro posuv ze strany na stranu by bylo vhodné využít elektrického pohonu s vhodnou regulací polohy. Ideální regulací polohy by mohl být regulátor PID, který by měl správně zvolené jednotlivé složky, a to proporcionální,

integrační a derivační. Tento regulátor by mohl být naprogramován a přímo začleněn do algoritmu. Tento regulátor by řešil i možné chyby vzniknuté zákmity při kličkování objektu směrem ke kameře, kdy by aktuálně program mohl vyhodnocovat neustálé změny pohybu.



Obrázek 16 - Okno aplikace pro posun vpravo

Během zapnutého algoritmu je ukládán video záznam ve formátu *mp4* do stejné složky jako původní skript s názvem *sledovani.mp4*. Tento záznam se vypíná společně s algoritmem pomocí tlačítka *Vypnout* umístěném vedle tlačítka pro zapnutí algoritmu. Záznam není upraven, proto jeho rychlost odpovídá základnímu nastavení pro tvorbu video záznamu. Tato rychlost by v případě potřeby bylo možné vhodně upravit dle požadovaných parametrů.

Aplikaci lze opakovaně pomocí tlačítek zapínat a vypínat, záznam je však tvořen vždy poslední nahrávanou částí, protože se při stejném názvu soubor vždy přemaže z důvodu stejného názvu. Pro úplné vypnutí aplikace je nutné ji zavřít pomocí křížku v pravém horním rohu aplikace.

Pro potřebu ukládání všech záznamů by se algoritmus mohl rozšířit například o cyklus, který bude měnit pořadové číslo záznamu. V aplikaci by bylo poté nutné zaznamenat počet předpokládaných nahrávek. Další možností může být editovatelné pole kam uživatel vždy vypíše název dané nahrávky před zapnutím algoritmu a začátkem záznamu videa.

5 Závěr

Cílem této práce bylo vytvořit vhodný program pro dynamickou kameru s možností sledování definovaných tvarů, která by se dala využít k různým účelům jako například pro vylepšení funkcí chytrých domácností či obecních a firemních prostor.

V rámci práce se vyhodnocovala různá možná prostředí pro programování algoritmu. Mezi porovnávanými prostředími byl vybrán program Matlab. Tato volba byla zvolena s přihlédnutím na jednoduchost a přehlednost programování, uživatelsky intuitivní prostředí, které vhodně vyhodnocuje vzniklé chyby při tvorbě. Mezi další výhody patří zpracování dat pomocí matic, což při některých operacích může zjednodušit algoritmus. Poslední výhodou bylo prohloubení znalostí a časté využívání programu ke studijním účelům při tvorbě modelového chování nebo jiných aplikací.

V dalších krocích byly porovnávány jednotlivé metody pro detekci pohybu v obraze a rozlišení tvarů pro zařazení objektu. Pro detekci pohybu nebyla pro reálné řešení zvolena žádná metoda z důvodů úpravy priorit vyhledávání. V práci je prvotně řešeno rozlišení tvarů a jeho správné zařazení a detekce pohybu už je nahrazena přímým sledováním hlídaného objektu. Pro správné rozpoznání objektů na sledovaném snímku byla vybrána neuronová síť, to konkrétně konvoluční neuronová síť dostupná z rozšiřujících balíčků programu Matlab. Pro trénink této sítě byla použita sada dat rovněž vytvořená pro program. V případě možných specifických rozšíření může být vhodné síť použít s jiným setem dat přímo pro konkrétní problém.

V rámci sledování objektu byla zvolena metoda definující část snímku jako zónu bez nutnosti pohybu a okrajové části, které vyžadují pohyb snímací kamery. Jelikož pro práci bylo využito integrované kamery notebooku a pohyb byl zajišťován operátorem, byl pokyn k pohybu pouze zobrazen v uživatelské aplikaci. Pro možné rozšířené automatizované řešení by bylo vhodné použít externí kamery, elektropohonu a doplnění algoritmu o příkazy pro pohyb motoru s vhodnou polohovou regulací pro ideální umístění objektu vůči snímku.

Při reálném ověření algoritmu bylo řešení doplněno o jednoduchou uživatelskou aplikaci s nahráváním video záznamu z kamery. Reálné řešení spolu s porovnáním metod detekce a vhodného prostředí byly splněny zadané body této práce.

Seznam použité literatury

[1] *Programování 1 – Java* [online]. Fakulta elektrotechnická, České vysoké učení technické [cit. 2021-03-13].
Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a0b36pr1/lectures/01/36pr1-01_uvod_java.pdf

[2] *Programming in Java Advanced Imaging: Introduction to Java Advanced Imaging* [online]. Sun Microsystems, Inc., 1999 [cit. 2021-03-13].
Dostupné z: http://iihm.imag.fr/Docs/java/jai1_0guide/API-summary.doc.html

[3] *Java Media APIs* [online]. ORACLE, 2021 [cit. 2021-03-13].
Dostupné z: https://www.oracle.com/java/technologies/javase/media-apis.html

[4] ŠVEC, Jan. *Učebnice jazyka Python (aneb Létající cirkus)* [online]. PyCZ, 2002 [cit. 2021-03-17]. Dostupné z: https://docplayer.cz/300544-Ucebnice-jazyka-python-aneb-letajici-cirkus.html

[5] *scikit-image image processing in python* [online]. the scikit-image development team [cit. 2021-03-17]. Dostupné z: https://scikit-image.org/

[6] *NumPy* [online]. NumPy [cit. 2021-03-17]. Dostupné z: https://numpy.org/

[7] *SciPy documentation* [online]. The SciPy community [cit. 2021-03-17]. Dostupné z: http://scipy.github.io/devdocs/index.html

[8] *TensorFlow* [online]. TensorFlow [cit. 2021-03-17].
Dostupné z: https://www.tensorflow.org/

[9] *Overview* [online]. Alex Clark and Contributors [cit. 2021-03-17]. Dostupné z: https://pillow.readthedocs.io/en/stable/handbook/overview.html

[10] *Mahotas: Computer Vision in Python* [online]. Luis Pedro Coelho [cit. 2021-03-17]. Dostupné z: https://mahotas.readthedocs.io/en/latest/

[11] *GraphicsMagick Image Processing Toolbox* [online]. GraphicsMagick Group [cit. 2021-03-17]. Dostupné z: http://www.graphicsmagick.org/

[12] *cairo* [online]. 2014 [cit. 2021-03-17]. Dostupné z: https://cairographics.org/

[13] PHILLIPS, Dwayne. *Image processing in C*. Lawrence, 1994.
ISBN 01-310-4548-2.

[14] *OpenCV modules* [online]. doxygen, 2020 [cit. 2021-03-21]. Dostupné z: https://docs.opencv.org/4.5.1/

[15] *SimpleCV Tutorial* [online]. Sight Machine, 2013 [cit. 2021-03-21]. Dostupné z: http://tutorial.simplecv.org/en/latest/

[16] *ITK Overview* [online]. ITK [cit. 2021-03-30]. Dostupné z: https://itk.org/about/

[17] *About SimpleITK* [online]. SimpleITK [cit. 2021-03-30]. Dostupné z: https://simpleitk.org/about.html

[18] *MATLAB* [online]. The MathWorks, Inc. [cit. 2021-03-31]. Dostupné z: https://www.mathworks.com/products/matlab.html

[19] *Image Processing Toolbox* [online]. The MathWorks, Inc. [cit. 2021-03-31]. Dostupné z: https://www.mathworks.com/products/image.html

[20] *Image Aquisition Toolbox* [online]. The MathWorks, Inc. [cit. 2021-03-31]. Dostupné z: https://www.mathworks.com/products/image-acquisition.html

[21] HLAVÁČ, Václav, SEDLÁČEK, Miloš <{hlavac,sedlacem }@fel.cvut.cz> *Zpracování signálu a obrazu* [HTML dokument]. Elektrotechnická fakulta ČVUT v Praze, prosinec 1999 [cit. 2021-05-10]. Dostupné z: https://docplayer.cz/7024031-Zpracovani-signalu-a-obrazu-pracovni-verze-skripta-v-tisku-pro-studenty.html

[22] Background subtraction techniques: a review. In: *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)* [online]. 2004 [cit. 2021-05-08]. DOI: 10.1109/ICSMC.2004.1400815 ISSN:1062-922X. Dostupné z: https://ieeexplore-ieee-org.ezproxy.lib.vutbr.cz/document/1400815

[23] Moving target classification and tracking from real-time video. In: *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98 (Cat. No.98EX201)* [online]. 1998 [cit. 2021-05-08]. DOI: 10.1109/ACV.1998.732851 ISBN: 0-8186-8606-5.
Dostupné z: https://ieeexplore.ieee.org/document/732851

[24] VAŠIČKA, Aleš. Detekce a klasifikace objektů v obraze z kamery. Zlín, 2018. Dostupné z: https://vutbr-my.sharepoint.com/personal/162899_vutbr_cz/Documents/Soubory%20z%20chatu%20aplikace%20Microsoft%20Teams/va%C5%A1i%C4%8Dka_2018_dp.pdf. Diplomová práce. Univerzita Bati ve Zlíně, Fakulta aplikované informatiky.

[25] HRÚZ, Marek. *LBP, HoG* [online]. 2015 [cit. 2021-05-11]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/mpv/05/materialy05.pdf

[26] *Support vector machines* [online]. [cit. 2021-05-11]. Dostupné z: https://is.muni.cz/el/1433/podzim2006/PA034/09_SVM.pdf

[27] HRÚZ, Marek. *SIFT, SURF, MSER* [online]. 2015 [cit. 2021-05-11]. Dostupné z: http://www.kky.zcu.cz/uploads/courses/mpv/04/materialy04.pdf

[28] VONDRÁK, Ivo. *Neuronové sítě* [online]. Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky, duben 1994 [cit. 2021-05-11]. Dostupné z: http://vondrak.cs.vsb.cz/download/Neuronove_site.pdf

[29] *MATLAB Coder Support Package for NVIDIA Jetson and NVIDIA DRIVE Platforms* [online]. MathWorks, Inc. [cit. 2021-08-09]. Dostupné z: https://www.mathworks.com/help/supportpkg/nvidia/index.html?s_tid=CRUX_lftnav

[30] *Data Import and Analysis* [online]. MathWorks, Inc. [cit. 2021-08-09]. Dostupné z: https://www.mathworks.com/help/matlab/data-import-and-analysis.html?s_tid=srchbrcm

[31] *Deep Learning Toolbox* [online]. MathWorks, Inc. [cit. 2021-08-09]. Dostupné z: https://www.mathworks.com/help/deeplearning/index.html?searchHighlight=deep%20learning%20toolbox&s_tid=srchtitle

[32] *Image Processing Toolbox* [online]. MathWorks, Inc. [cit. 2021-08-09]. Dostupné z: https://www.mathworks.com/help/images/?s_tid=srchbrcm

[33] *App Building* [online]. MathWorks, Inc. [cit. 2021-08-09]. Dostupné z: https://www.mathworks.com/help/matlab/gui-development.html?s_tid=srchbrcm

[34] *squeezenet* [online]. MathWroks, Inc. [cit. 2021-08-09]. Dostupné z: https://www.mathworks.com/help/deeplearning/ref/squeezenet.html

Seznam obrázků

Seznam příloh

Elektronické přílohy:

PŘÍLOHA 1: Uživatelská aplikace na detekci objektů