VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV VÝROBNÍCH STROJŮ, SYSTÉMŮ A ROBOTIKY
INSTITUTE OF PRODUCTION MACHINES, SYSTEMS AND ROBOTICS

VYUŽITÍ OPENSOURCOVÝCH KNIHOVEN PRO POČÍTAČOVÉ VIDĚNÍ
OPEN SOURCE LIBRARIES USE FOR COMPUTER VISION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
David Strejček

VEDOUCÍ PRÁCE
SUPERVISOR
Ing. Tomáš Marada, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav výrobních strojů, systémů a robotiky
Student:	**David Strejček**
Studijní program:	Strojírenství
Studijní obor:	Základy strojního inženýrství
Vedoucí práce:	**Ing. Tomáš Marada, Ph.D.**
Akademický rok:	2021/22

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:

Využití opensourcových knihoven pro počítačové vidění

Stručná charakteristika problematiky úkolu:

Cílem práce je seznámení se s opensourcovými knihovnami pro počítačové vidění (OpenCV, Dlib, GIL, VXL atd.) a provést jejich srovnání. Dalším cílem je zvolit jednu knihovnu a vypracovat vzorové příklady do výuky.

Cíle bakalářské práce:

Rešerše na počítačové vidění.
Seznámení se s knihovnami pro počítačové vidění.
Vytvoření vzorových úloh využitelných ve výuce.
Doporučení a zdůvodnění volby vybrané knihovny.
Zhodnocení výsledků práce.

Seznam doporučené literatury:

The 3-Clause BSD License. Open Source Initiative [online]. 1998 [cit. 2018-05-02]. Dostupné z: https://opensource.org/licenses/BSD-3-Clause

Python [online]. Python Software Foundation, 2001 [cit. 2018-05-02]. Dostupné z: https://www.python.org

OpenCV library [online]. Santa Clara: Intel Corporation, Willow Garage, Itseez, 2000 [cit. 2018-05-02]. Dostupné z: https://opencv.org

NumPy [online]. 2006 [cit. 2018-05-02]. Dostupné z: http://www.numpy.org

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku.

V Brně, dne

L. S.

doc. Ing. Petr Blecha, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Cílem této bakalářské práce je seznámit čtenáře s východisky vybraných opensourcových knihoven a s jejich implementací v oblasti počítačového vidění. Úvod práce pojednává o teoretickém základu počítačového vidění a možnostmi jeho praktického využití. V rešeršní části je představen elementární přehled opensourcových licencí včetně vybraných opensourcových knihoven. Tato část je obohacena o srovnání vybraných opensourcových knihoven dle stanovených kritérií. V praktické části je aplikována zvolená openosourcová knihovna v rámci vzorové úlohy, zabývající se detekcí pozic laserového paprsku.

ABSTRACT

The aim of this bachelor's thesis is to introduce readers to basis of selected open-source libraries and their implementation in the field of computer vision. The thesis's introduction deals with theoretical basis of computer vision and possibilities of its practical use. The research part presents an elementary overview of open-source licenses including selected open source libraries. This part is enriched with a comparison of selected open-source libraries according to selected criteria. In the practical part, the selected open-source library was applied within the sample task, dealing with the detection of laser beam positions.

KLÍČOVÁ SLOVA

počítačové vidění, opensourcová knihovna, opensourcová licence, OpenCV, Dlib, VXL, VLFeat, srovnání opensourcových knihoven

KEYWORDS

computer vision, open-source library, open-source license, OpenCV, Dlib, VXL, VLFeat, comparison of open-source libraries

BIBLIOGRAFICKÁ CITACE

STREJČEK, D. *Využití opensourcových knihoven pro počítačové vidění,* Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství. 2021, 48 s., Vedoucí diplomové/bakalářské práce Ing. Tomáš Marada, Ph.D.

PODĚKOVÁNÍ

Rád bych touto cestou vyjádřil vřelé poděkování vedoucímu práce Ing. Tomáši Maradovi, PhD. za trpělivost, shovívavost a podnětné připomínky. Dále bych chtěl poděkovat své milé Elišce, která mě po celou dobu studia emočně podporovala.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Tomáše Marady, PhD. a s použitím literatury uvedené v seznamu.

V Brně dne 13.8.2021 .....

Strejček David

OBSAH

1 ÚVOD

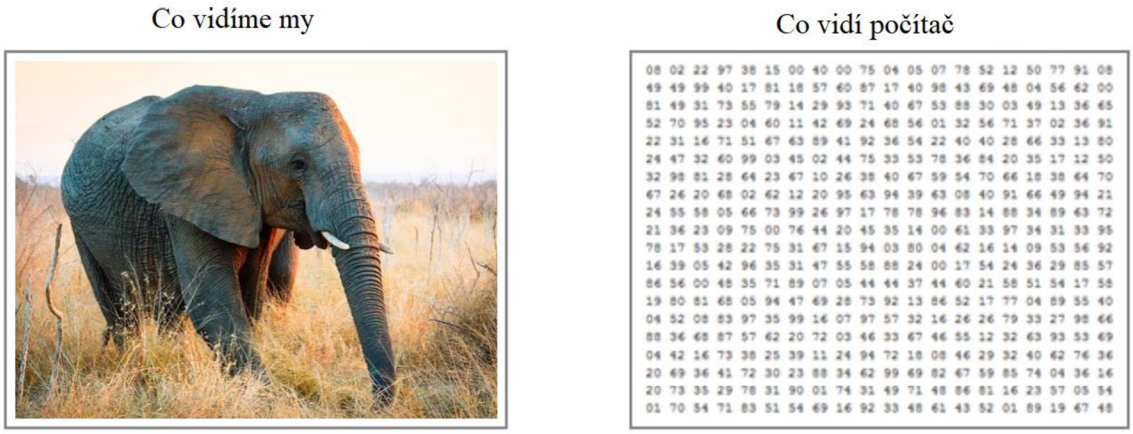
V posledních dvou desetiletích je rozmanitá škála každodenních aktivit hojně doprovázena výpočetní technikou jako takovou. Počítačové vidění usiluje o imitaci lidského vidění, a to snímáním obrazu elektrickými zařízeními a porozuměním jejich obsahu. Klíčovou úlohou interdisciplinárního odvětví je rozeznání zájmového objektu, jehož výsledný obraz je převedením části trojrozměrného do dvourozměrného prostředí.

Cílem této bakalářské práce je seznámení čtenáře s možnostmi a funkčními omezeními implementace vybraných opensourcových knihoven při počítačovém vidění. V úvodní části práce jsou představeny elementární poznatky o počítačovém vidění a možnostech jeho využití napříč různými obory. Rešeršní část práce se zaměřuje na charakteristiku základních opensourcových licencí s ohledem na možnosti využívání a distribuce. Tato sekce je doplněna o přehled nejvíce užívaných opensourcových knihoven včetně vzájemné komparace z hlediska podporovaných programovacích jazyků, podoby dokumentace a podkladů, rychlosti a možnostmi jejich využití. Na závěr je vybraná knihovna aplikována v rámci dvou vzorových úloh. Zvolenými úlohami jsou detekce laserového paprsku a detekce kruhovitých objektů o určité velikosti.

2 POČÍTAČOVÉ VIDĚNÍ

Jakožto lidé využíváme náš zrak dennodenně jako jeden z nejzákladnějších zdrojů informací o okolním světě. Dokážeme snadno a bez problémů rozeznávat nejrůznější objekty jak v našem světě, tak i na fotografiích. Například v Obr. 1 dokáže i malé dítě snadno identifikovat slona chodícího ve vysokém travnatém porostu. Jedním z mnoha problémů může tedy být, jak počítač pozná, zdali se v obraze slon nachází či nikoli. A právě tímto a dalším nespočtem důležitějších i méně důležitějších úkolů se zabývá počítačové vidění. [3]

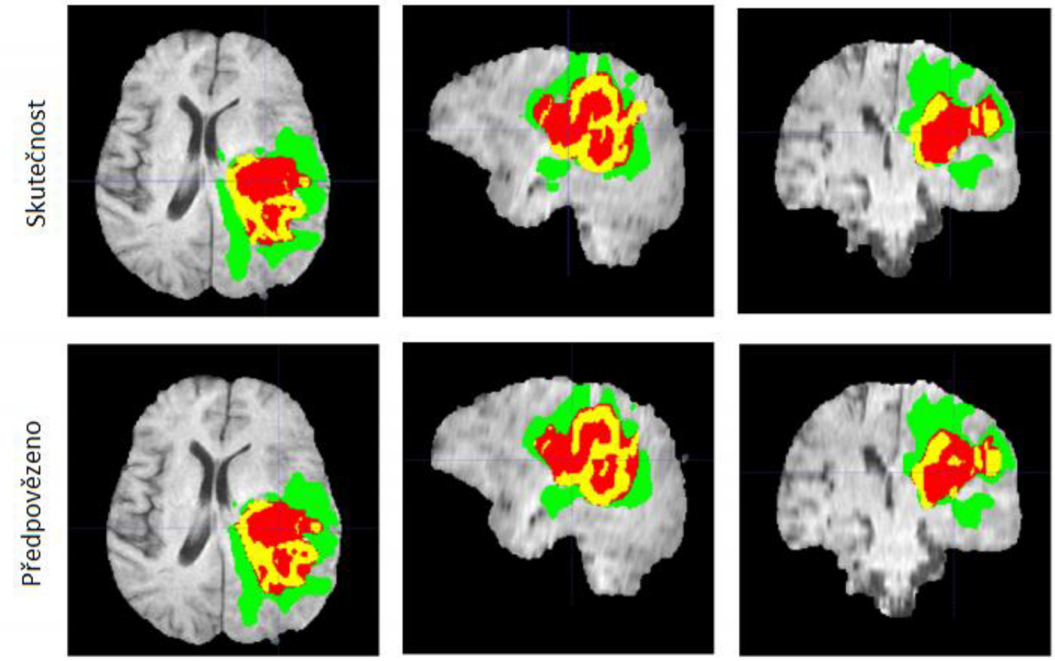
Počítačové vidění (angl. *computer vision*, zkráceně CV) je interdisciplinárním odvětvím zabývající se možnostmi, jak umožnit počítači získávat data z obrazu. Pokud se my, jako lidé, podíváme na fotografii vidíme různé barvy, stíny, osvětlení, hrany objektů apod., díky kterým snadno dokážeme snadno tuto fotografii popsat. Dáme-li však stejnou fotografii počítači, tak uvidí pouze nějaké uskupení čísel, které popisují intenzity jednotlivých barevných složek obrazu a počítač z něj dokáže vyčíst potřebné informace za použití nejrůznějších algoritmů a umělé inteligence. [1]



Obr. 1 Příklad rozdílu reprezentace slona pro člověka a pro počítač – upraveno [4]

2.1 Využití

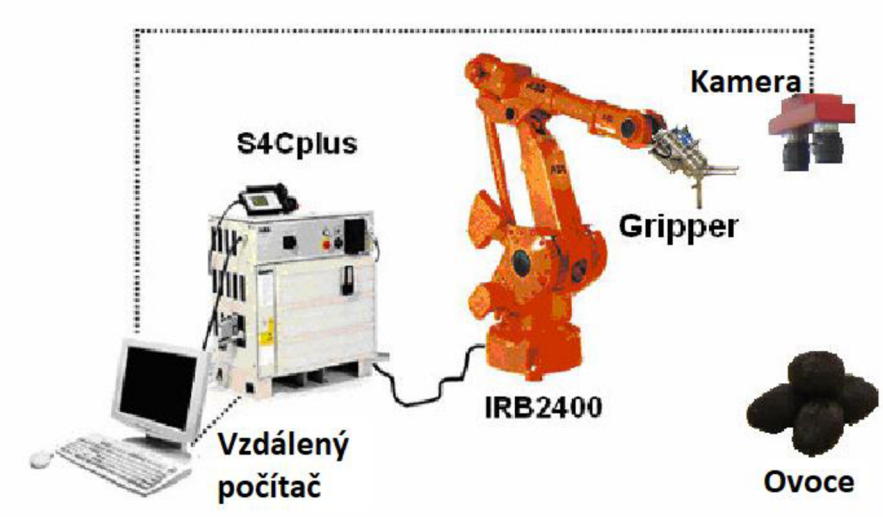
S počítačovým viděním se v dnešním světě můžeme setkat kdekoliv. Ve zdravotnictví pomáhá lékařům diagnostikovat problémy lidského těla, jako rozpoznání zhoubných a nezhoubných nádorů z dat poskytnutých magnetickou rezonancí, detekce koronaviru COVID-19 za pomocí rentgenu hrudi a mnoho dalšího. [11]



Obr. 2 Segmentace mozkového nádoru s využitím umělé inteligence – upraveno [12]

V zemědělství pak začíná hrát velmi důležitou roli, kdy pomáhá zemědělcům monitorovat nepatrné změny v plodinách způsobené podvýživou a umožnit tak včasné zareagování na tyto změny. Dále automatizace v zemědělství, kdy za použití nejmodernějších zemědělských strojů využívajících počítačového vidění a umělé inteligence je sklizeň mnohem efektivnější a cenově méně nákladná. [11]

V neposlední řadě je pak počítačové vidění velice hojně využíváno ve výrobním průmyslu, kdy se řadí do nedílné součásti v dnešním trendu automatizace a nástupu Průmyslu 4.0. Nejčastějším užitím v tomto oboru je vizuální navigace robotů, kdy kamera dokáže přesně lokalizovat objekt (výrobek, nástroj apod.) a tyto souřadnice poslat robotu. Dále pak detekce anomálií, kdy systém analyzuje nová obrazová data a porovnává je s již existujícím daty a nachází anomálie, které mohou vést k nebezpečným situací. O dalších aplikacích počítačového vidění ve výrobním průmyslu se dá najít například na webových stránkách společnosti Visionify [16] zabývající se počítačovým viděním. [15]



Obr. 3 Komponenty pro systém vizuální navigace robotu - upraveno

Za zmínku také stojí webová stránka profesora Univerzity Britské Kolumbie Davida Lowe [10], kde Lowe uvádí příklady společností zabývající se počítačovým viděním v nejrůznějších oblastech. Stránka byla naposledy aktualizována v roce 2015, jelikož jak sám autor uvedl: *„Od té doby [založení stránky v roce 1990] zaznamenala oblast počítačového vidění obrovský úspěch ve vývoji užitečných aplikací a je pro mě již nadále nemožné být v obraze s neustálým růstem společností zabývající se touto problematikou.“* (vlastní překlad). Každopádně seznam je stále velice dobrým přehledem. [10]

2.2 Teorie

Pro získání potřebných dat z obrazu je zapotřebí provést několik operací. Jako první krok je kalibrace kamery. Dále pak je třeba provést předzpracování obrazu, do čehož se řadí např. korekce jasu, filtrace obrazu, segmentace. Následuje samotné nasazení počítačového vidění, kdy za pomocí nejrůznějších algoritmů a využití neuronových sítí získáme z obrazu požadovaná data.

2.2.1 Reprezentace obrazu

Jak bylo řečeno v kapitole 2, počítač vidí pouze jakési uskupení intenzit pixelů. Obraz je tedy v počítači reprezentován funkcí jejíž vstupní parametry jsou pozice x a y a výstupem je buďto intenzita určitého barevného spektra v tomto bodě nebo vektor intenzit všech barevných spekter v tomto bodě.

Nejčastěji se můžeme setkat s maticí o rozměrech $m \times n$, kde m je šířka a n výška obrázku v pixelech, nebo s vektorem o délce $m \cdot n$, ke kterému je pak definována šířka obrázku m.

Výchozí hodnoty pixelů jsou reprezentovány 8bitovými hodnotami (rozmezí celých čísel 0-255). Tyto hodnoty jsou často převáděny do 32bitových hodnot s pohyblivou řádovou čárkou (rozmezí $1{,}17\times10^{-38}$- $3{,}40\times10^{38}$), díky čemuž budou výpočty dosahovat mnohem větších přesností.

2.2.2 Zpracování obrazu

Zpracováním obrazu rozumíme transformace hodnot jednotlivých pixelů tak, aby byl obraz lépe zpracovatelný vyšší úrovní. Zpracování obrazu zpravidla strategicky manipuluje s hodnotami pixelů tak, aby se vyšší úroveň počítačového vidění dosáhla co nejlepších výsledků.

Mezi zpracování obrazu se řadí například redukce šumu Gaussovským filtrem, vyrovnání histogramu pro zvýšení kontrastu, geometrické transformace (translace, rotace, modifikace perspektivy aj.) apod. [1]

2.2.3 Vyšší úroveň počítačového vidění

Jako vyšší úroveň počítačového vidění můžeme definovat všechny postupy, které již ze zpracovaného snímku dokážou dostat požadovaná data. Může se jednat o detekci a rozpoznání obličejů, trasování objektů, tvorba 3D modelů z obrazu objektu pod několika úhly, aj.

3 OPENSOURCOVÉ LICENCE

Za opensourcová řešení se považují takové programy, jejichž zdrojový kód je volně dostupný a koncoví uživatelé si jej mohou dle libosti upravovat, distribuovat a provádět další úkony za dodržení určitých podmínek na základě dané opensourcové licence. [5]

Jako hlavní zastřešovatel OS licencí se dá považovat nezisková organizace Open Source Initiative, zkráceně OSI, která prověřuje nejrůznější OS licence a vydává seznam schválených licencí. V současné době tento seznam čítá 114 licencí, včetně již v současnosti nepoužívaných, či nahrazených novějšími verzemi. [6]

3.1 2bodová BSD

Tato licence nese tento název, jelikož je definována třemi základními body:

1. V redistribuci zdrojového kódu musí být zachována uvedená autorská práva, seznam podmínek a níže uvedené odmítnutí odpovědnosti
2. Redistribuce v binární formě musí obsahovat uvedená autorská práva, seznam podmínek a níže uvedené odmítnutí odpovědnosti v dokumentaci anebo jiných materiálech poskytnutých s distribucí.

Celé znění odmítnutí odpovědnosti je uvedeno na webových stránkách licence [13].

3.2 3bodová BSD

Licence je totožná s 2bodovou BSD, jen obsahuje jeden bod navíc:

1. Jména držitelů autorských práv ani jména jejich přispěvatelů nesmí být použita pro podporování a promování produktů vycházejících z tohoto softwaru bez předešlého písemného souhlasu

3.3 Licence Apache 2.0

Tato licence dává uživateli všechny práva k použití, reprodukci, tvorbě odvozených projektů, veřejnému vystavování a využívání, sublicencování a distribuování po přijetí pravidel a podmínek definovaných touto licencí. [17]

3.4 Boost software license 1.0

Obdobně jako licence Apacahe dává Boost software license uživateli všechna práva k softwaru, za podmínek, že musí zanechat celou původní licenci v původním stavu. [33]

3.5 Zlib/libpng licence

Licence dává uživateli volně používat, šířit a redistribuovat software za předpokladu následujících podmínek: [34]

1. Původ softwaru nesmí být zkreslen; uživatel nesmí tvrdit, že napsal původní software
2. Všechny změny ve zdroji musí být jasně označeny a nebýt označeny jako součást původního softwaru
3. Toto oznámení nesmí být odstraněno ani změněno v žádné formě distribuce

4 VYBRANÉ OPENSOURCOVÉ KNIHOVNY

Jak již bylo řečeno, počítačové vidění je velice komplexní obor. Existuje nespočet algoritmů pro širokou škálu úloh, které dokáže počítačové vidění řešit. Využívá se také umělé inteligence, zejména pro detekci a rozpoznávání lidí, objektů apod. Pro jejich efektivní využití je nezbytné těmto algoritmům a neuronovým sítím porozumět. Avšak jejich implantace je o dost náročnější. Abychom mohli efektivně využívat všech možností, které počítačové vidění nabízí, je více než vhodné použít nějakou již vytvořenou a ověřenou knihovnu. Tyto knihovny pak obsahují již předvytvořené a optimalizované výše zmíněné algoritmy a implementace umělé inteligence.

V dnešní době existuje spoustu open sourcových knihoven zabývající se problematikou počítačového vidění. Zde však bude čtenář seznámen jen s těmi nejpoužívanějšími a nejověřenějšími.

4.1 OpenCV

OpenCV (celý název *Open Source Computer Vision Library)* je knihovna psána a používána v programovacím jazyce C++, obsahující wrapper pro Python, Javu a MATLAB. Obsahuje přes 2 500 algoritmů pro zpracování obrazu a strojového učení, které lze využít pro rozpoznávání obličejů, identifikování objektů, extrakci 3D modelů ze scény, produkci 3D mračna bodů atd. Vyniká zejména v *real-time* aplikacích. [7]

OpenCV zaštituje nezisková společnost OpenCV Foundation a v současnosti je touto a dalšími společnostmi vyvíjena. Původně vznikala jako výzkumný projekt společnosti Intel. Jedná se o jednu z nejvíce využívaných knihoven pro počítačové vidění, kterou používají mimo jiné i mnohé světové korporace (IBM, Google, Microsoft, Honda, Toyota aj.). [7]

OpenCV verze 4.4.0 a nižší je krytá 3bodovou BSD licenci a verze 4.5.0 a vyšší licenci Apache 2.0. [8]

Knihovna má modulární strukturu, což znamená, že se vlastně jedná o několik sdílených nebo statických knihoven mezi které patří:

- core (*core functionality* – základní funkčnost) – definuje základní data struktury jako již výše zmíněná třída Mat a základní funkce užívané všemi dalšími moduly
- imgproc (*image processing* – zpracování obrazu) – modul pro zpracování obrazu obsahující např. lineární a nelineární filtraci obrazu, geometrické transformace (změna velikosti, prespektivní zakřivení atd.)

- video (*video analysis – analýza videa*) – modul pro analýzu videa obsahující odhad pohybu, odstranění pozadí a algoritmy pro sledování objektů
- calib3d (*camera calibration and 3D reconstruction* – kalibrace kamery a 3D rekonstrukce) – základní algoritmy pro geometrii více pohledů, kalibrace samostatné kamery a stereo kamer, odhad pozice objektu a elementy 3D rekonstrukce
- features2d (*2D feature framework*) – detektory a deskriptory klíčových bodů
- objdetect (*object detection* – detekce objektů)
- highgui (*high-level GUI* – vysokoúrovňové grafické rozhraní)
- videio (*video I/O* – video input/output). [9]

4.2 Dlib

Dlib je multiplatformní knihovna napsána v programovacím jazyce C++, avšak kromě podpory tohoto jazyka ji lze využít i v Pythonu. Je to víceúčelová knihovna, která kromě funkcí pro počítačové vidění, neuronové sítě a strojové učení obsahuje také spoustu nezávislých softwarových komponent, které se zabývají problémy počítačových sítí, grafických rozhraní, datovými strukturami, lineární algebrou a mnohým dalším. [8]

Vznikla v roce 2002 a je primárně vyvíjena svým strůjcem Davisem Kingem. Původně se jednalo především o právě již zmíněnou víceúčelovou knihovnu zaměřenou na kontraktové programování, v posledních letech je však vývoj zaměřen hlavně na tvorbě širokého spektra statistických nástrojů pro strojové učení. Základní myšlenkou pro vývoj knihovny zůstává zejména portabilita a jednoduchost nasazení a použití. Knihovna je krytá licencí Boost Software License – version 1.0. [8]

Knihovna opět není jednotná, ale jedná se o kolekci všech svých komponent. Na rozdíl od OpenCV tyto komponenty nejsou tak široce obsáhlé, ale jsou více úzce definované. Mezi jednotlivými komponenty jsou například array2d (základní reprezentace obrazu pro dlib), image_processing (algoritmy pro počítačové vidění), image_transforms (předzpracování obrazu) atd. [8]

4.3 VXL

VXL sama o sobě není knihovna, ale jedná se o kolekci 6 základních a několika dodatečných knihoven. Je napsaná v jazyce C++ a navrhnutá tak, aby byla velice přenosná. Vznikla ze dvou velkých systémů Image Understanding Enviroment (IUE) a TargetJr, jakožto lehčí, rychlejší a konzistentnější alternativa. Je distribuována pod licencí libpng.

VXL je zkratka pro vision-*something*-library, kde právě slovo *something* je nahrazeno počátečními písmeny dané knihovny. Každá z následujících základních knihoven může být použita samostatně, jsou tedy nezávislé na ostatních. Dodatečné knihovny však jsou již na nějaké základní knihovně závislé. Základními knihovnami tedy jsou:

- VNL (*numerics* – číslice) – číselné kontejnery a algoritmy, např. matice, vektory, optimizátory
- VIL (*imaging* – zobrazování) – nahrávání, ukládání a manipulování s obrázky
- VGL (*geometry* – geometrie): geometrie bodů, křivek a dalších elementárních objektů
- VSL (*streaming I/O* – přenos videa)
- VBL (*basic templates* – základní šablony)
- VUL (*utilities* – užitečné nástroje).

4.4 VLFeat

VLFeat je open sourcová, přenosná knihovna zaměřující se především na rychlé prototypování a opakovatelný výzkum pro vědce a studenty oboru počítačového vidění. Obsahuje pečlivé implementace základních stavebních kamenů počítačového vidění, jakým jsou například detektory a extraktory klíčových bodů, k-mean shlukování atd. [14]

Knihovna je psána v programovacím jazyce C a určena pro přímou implenetaci v softwaru MATLAB [14]. Neobsahuje algoritmy pro zpracování obrazu a jde tedy o knihovnu zabývající se čistě vyšší úrovní počítačového vidění.

Je distribuována pod 2bodovou BSD licencí. [14]

5 SROVNÁNÍ VYBRANÝCH KNIHOVEN

5.1 Podporované programovací jazyky

Přehled podporovaných vybraných programovacích jazyků jednotlivými knihovnami je popsán v tabulce 1. Pro spoustu těchto kombinací však je třeba použití softwaru třetích stran. Zde dochází k jisté nejistotě zaručené funkčnosti a optimalizaci, je proto třeba tento software dobře prověřit.

Tab. 1 Přehled programovacích jazyků podporovaných jednotlivými knihovnami (1 - pouze starší verze, 2 - za použití softwaru třetích stran, 3 - nejsou implementovány všechny funkce)

Jazyk / Knihovna	C	C++	Python	C#	MATLAB	Java
OpenCV	Ano[1]	Ano	Ano	Ano[2]	Ano	Ano
Dlib	Ne	Ano	Ano[3]	Ano[2]	Ne	Ano[2]
VXL	Ne	Ano	Ano[2]	Ne	Ne	Ne
VLFeat	Ano	Ano	Ano[2,3]	Ne	Ano	Ne
BoofCV	Ne	Ano[2,3]	Ano[3]	Ne	Ne	Ano

5.2 Dokumentace

OpenCV má velice obsáhlou dokumentaci, ve které je velice snadná a intuitivní orientace. Každý modul je detailně popsán a spoustu funkcí obsahuje i popis toho, jak fungují. V dokumentaci jsou i tutoriály pro jazyky C++, Python a JavaScript, které pokrývají jak teoretickou stránku tématu, tak i její implenetaci v OpenCV. [19]

Dokumentace VXL knihoven je také velice obsáhlá a každá knihovna má svoji dokumentaci zvlášť. Všechny knihovny jsou opět detailně popsány a funkce taktéž. Tutoriály samotná dokumentace neobsahuje, avšak VXL má k dispozici svou online knihu [21], která čtenáře seznámí s funkčností všech hlavních a některých vedlejších knihoven, avšak v ní už nenalezne teorii. [20]

Dlib nemá vytvořenou dokumentaci jako takovou, ale vlastně webová stránka knihovny slouží jako dokumentace. V případě této knihovny je opravdu každá funkce a třída velice jasně popsána v komentářích přímo v kódu. Tyto komentáře mají danou strukturu, která je popsána na podstránce *Introduction* [8].

Vlfeat má velice jednoduchou a přehlednou dokumentaci, vzhledem k tomu, že knihovna není tak obsáhlá jako ostatní srovnávané. U každé funkce je opět detailní popis včetně vstupních parametrů. [26]

5.3 Podklady

Pro OpenCV existuje spoustu knížek a článků seznamující čtenáře s touto knihovnou. Kniha *Learning OpenCV: Computer vision with the OpenCV library* od autorů samotné knihovny, Gary Bradski a Adrian Kaehler, kombinuje teoretickou část počítačového vidění s její implementací právě v OpenCV. Dle autorů má dále kniha podat čtenáři intuitivní porozumění algoritmů počítačového vidění a kdy a jaké algoritmy použít. [2] Dále stojí za zmínku kniha *Practical OpenCV* od Samarth Brahmbhatt, která má podobný účel, ale více se zaměřuje na praktické využití spíš než na teoretickou stránku věci. [22] Mezi články pak můžeme zařadit *A brief introduction to OpenCV* od autorů Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo a Mario Cifrek ze Záhřebské univerzity, který si dává za cíl stručně seznámit čtenáře s knihovnou bez nutnosti čtení zdlouhavých knih a referencí. Jedná se o opravdu stručné seznámení, a tak zde nejsou zmíněny pokročilá témata, jako GPU akcelerace. Jak už bylo zmíněno v kapitole 5.2.1, tutoriály lze nalézt i v samotné dokumentaci knihovny. [23]

Pro žádnou další knihovnu neexistují tak rozsáhle podklady, jako právě pro OpenCV. Knihovny VXL mají jako podklad online knihu zmíněnou v kapitole 5.2.1. Kniha již počítá s tím, že čtenář je seznámen se teoretickými základy počítačového vidění, a tak se spíše věnuje popisu implementace jednotlivých funkcí a tříd v knihovnách [21]

Dlib má jen jeden článek, který se věnuje seznámení se s knihovnou, konkrétně s její částí pro strojové učení. [24] Na webových stránkách má však autor uvedeny příklady různých aplikací pro programovací jazyky C++ a Python, které slouží jako solidní základ pro seznámení se s touto knihovnou. [8]

VLFeat využívá také online tutoriál, které opět čtenáře seznamují s funkčností knihovny a implementacemi funkcí. Některé tutoriály se povrchově dotýkají i teoretické stránce. [25]

Z těchto informací lze vyvodit jeden závěr a to sice, že pro začátečníka v oblasti počítačového vidění je OpenCV nejlepší volbou vzhledem k širokému množství elektronických i fyzických podkladů, které kombinují teorii počítačového vidění s jeho implementací právě v OpenCV.

5.4 Využití

Využití je u všech knihoven velice široké. OpenCV, jak již bylo řečeno v kapitole 4.1, používají jedny z největších firem na světě zabývající se informačními technologiemi, jako jsou IBM, Google, Honda apod. [7] Nikde není blíže specifikováno, k čemu tyto firmy knihovnu používají, avšak se dá předpokládat, že automobilky Honda a Toyota ji používají v automobilech pro nejrůznější asistenční systémy, jako jsou hlídání jízdního pruhu (Lane Assist), parkovací senzory aj.

Dlib má na webových stránkách SourceForge [29] vypsány všechny projekty, které knihovnu používají. Jedna z nejprestižnějších technických univerzit na světě, Massachusettský technologický institut (MIT), založila na této knihovně svou vlastní knihovnu pod názvem MITIE (MIT Information Extraction) sloužící pro extrakci informací z psaného textu. [27] Dále pak nástroj pro analýzu kódu Rose Compiler [28], implementaci knihovny pro operační systém Android dlib-android [30] a mnoho dalších. V oblasti počítačového vidění je knihovna OpenFace založena na dlib. OpenFace je implenetace rozpoznávaní obličejů s použitím hlubokých neurálních sítí založené na modelech z knihoven dlib a OpenCV [31]. Česká firma vyvíjející bezpečnostní technologie, 7 Marsyas, využívala implementace enkódování, SVM (Single Vector Machine) a Bayesovských sítí. [29] Další využití nachází knihovna ve firmě Cubic Motion zabývající se automatizací animací obličejů v herním průmyslu a je v tomto oboru lídrem. [32] Využívá knihovny dlib k prozkoumávání nových možností optimalizace v obličejové analýze a animaci. [29]

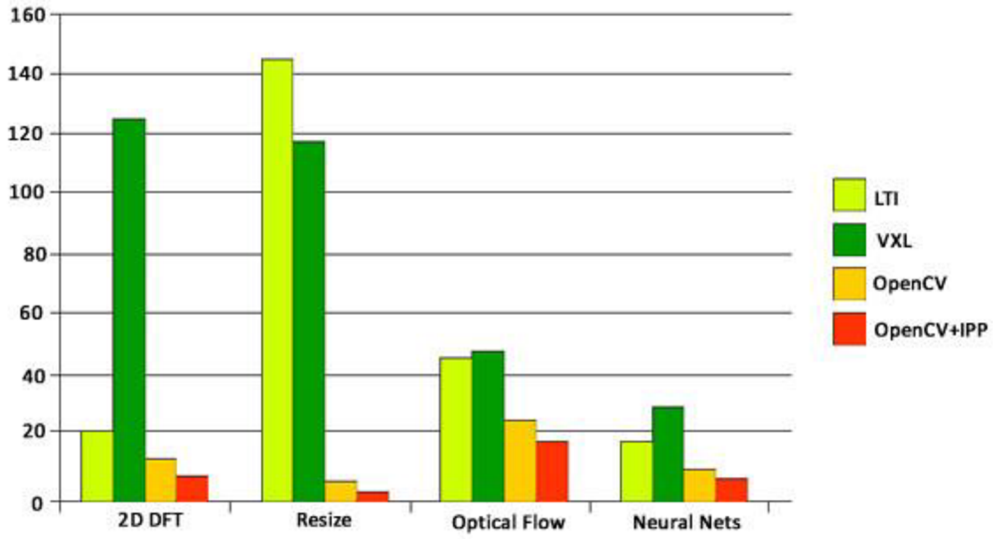
Knihovna VLFeat slouží především k výzkumným pracím. Proto nenajdeme žádné její využití v praxi. Co se týká využití VXL knihoven v praxi, tak neexistuje žádný záznam, který by toto pokrýval.

5.5 Porovnání rychlosti

Rychlost knihoven je velice důležitá zejména pro využití v reálném čase, například při monitorování prostředí. Těchto porovnání však neexistuje mnoho.

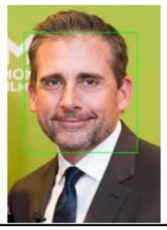
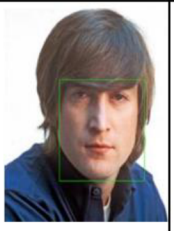
Jedno porovnání se nachází v knize *Learning OpenCV: Computer Vision with the OpenCV Library* od autorů Gary Bradski a Adrian Kaehler , kde je na 4 různých operacích porovnán rychlost knihoven OpenCV (ver. 1.0pre), LTI (ver. 1.9.14) a VXL (ver. 1.4.0). Testovací stanice běžela na procesoru Intel Pentium M o výkonu 1.7 GHz. Úlohy byli následující: [2]

- 2D DFT – Dopředná Fourierova transformace obrázku o rozměrech 515x512 pixelů
- Resize – Změna velikosti 8bitového tříkanálového obrázku z 512x512 na 384x384 s použitím bilineární interpolace
- Optical Flow – Sledováno 520 bodů s oknem o velikosti 41x41 pixelů
- Neural Nets – *mushroom benchmark* z knihovny FANN



Obr. 4 Porovnání rychlostí knihoven LTI, VXL, OpenCV a OpenCV s optimalizační knihovnou IPP. Na ose y je vyobrazen čas [2]

Další porovnání provedli Nataliya Boyko, Oleg Basystiuk a Nataliya Shakhovska ze Lvovské polytechnické národní univerzity ve své práci *Performance Evaluation and Comparison of Software for Face Recognition, based on Dlib and Opencv Library*. V tomto výzkumu byla opět rychlejší knihovna OpenCV, avšak byly použity pouze funkce založeny na algoritmu HOG (histogram of gradients). V případě použití jiných algoritmů, jako jsou například Haarovy kaskády, které jsou sice pomalejší, ale detailnější, se mohou výsledky lišit. Výsledky jsou na obrázku 5. Použitý hardware není specifikován. [18]

	Experiment č. 1	Experiment č. 2	Experiment č. 3
OpenCV			
Doba rozpoznání (sec.)	0. 5652	0.0934	0.6121
dlib			
Doba rozpoznání (sec.)	1.6584	0.2163	1.6889

Obr. 5 Porovnání výsledků a času potřebného pro nalezení obličejů s použitím OpenCV a dlib knihoven – upraveno [18]

6 VYTVOŘENÍ VZOROVÝCH ÚLOH

Pro vytvoření vzorových úloh byla vybrána knihovna OpenCV. Byla vybrána na základě široké funkčnosti, jednoduchosti instalace, velkému množství podkladových materiálů a internetových příspěvků, které velice usnadní práci s jakoukoliv knihovnou. Knihovna je v některých aplikacích i rychlejší než VXL a dlib, jak je zmíněno v kapitole 5.5.

Vzhledem k výběru knihovny OpenCV byli možnosti volby jazyka velké. Vybrán byl však jazyk Python, díky jeho přehlednosti, portabilitě a jednoduchosti nasazení. Je také vhodnou volbou do budoucna, jelikož se v dnešní době využívá jako jeden z předních jazyků pro umělou inteligenci, která k počítačovému vidění jednoznačně patří.

6.1 Detekce pozice laserového paprsku

Detekce pozice laserového paprsku je velice jednoduchá. Bude se jednat o detekci červeného paprsku na bílém pozadí. Pro detekci se použijí jen nejzákladnější funkce knihovny a základní numerické operace

6.1.1 Načtení videa z webkamery

Jako webkamera se používá mobilní telefon s operačním systémem Android. Obraz je přenášen přes lokální wi-fi síť za pomoci aplikace DroidCam.

```
cam = cv.VideoCapture(1)
while True:
    _, frame = cam.read()
    cv.imshow("frame", frame)

    if cv.waitKey(1) == ord('q'):
        break
```

Obr. 6 Ukázka kódu pro zobrazení videa z webkamery

Nejdříve je třeba vytvořit instanci třídy *VideoCapture(index)*, která bude číst data z webkamery, jejíž index se definuje při vytvoření instance. Poté se bude volat funkce *read*, která má 2 návratové hodnoty: *retval* a *image*. *Retval* je pouze boolean hodnota, která nabývá hodnot *true*, pokud je nějaký snímek detekován, v opačném případě je *false*. *Image* už je požadovaný obrázek. Obrázek je potom zobrazen do okna pomocí funkce *imshow(jmeno_okna, obrazek)*. Posledním prvkem je klauzule *if*, kde se kontroluje za pomoci funkce *waitKey(cas)*, zdali uživatel nestisknul požadovanou klávesu. Pokud stisknul, program se ukončí.

6.1.2 Detekce pozice

```
red = frame[:, :, 2]
red = cv.GaussianBlur(red, (5,5), 4)

_, threshRed = cv.threshold(red, 240, 255, cv.THRESH_BINARY)

laserPos = np.where(threshRed == 255)
```

Obr. 7 Kód pro jednoduchou detekci paprsku

Barva paprsku je červená, nemusíme tedy převádět barevný prostor, jelikož obrázek je reprezentován v BGR barvách (blue, green, red). Ze snímku je tedy extrahován červený kanál do proměnné *red*. Následuje funkce *GaussianBlur(obrazek, velikost_kernelu, sigma)*, která slouží k redukci šumu. Složka je pak funkcí *threshold(obrazek, minimalni_hodnota_intenzity, hodnota_intenzity_po_funkci, typ_prahu)* prahována. Funkce pomocí vstupů z obrázku odstraní hodnoty, které jsou nižší než *minimalni_hodnota_intenzity* a vyšší hodnoty nahradí hodnotami *hodnota_intenzity_po_funkci* a vznikne bitový obrázek. Následně funkce *np.where(threshRed == 255)* najde v matici pozice všech hodnot rovných 255. Tyto pozice jsou pak předány funkci *find_center(matice)*, která najde střed hodnot.

```
def find_center(matrix):
    if not len(matrix[0]) == 0:
        maxY = matrix[0][len(matrix[0])-1]
        minY = matrix[0][0]
        y = int((maxY+minY)/2)

        maxX = matrix[1][len(matrix[1])-1]
        minX = matrix[1][0]
        x = int((maxX+minX)/2)

        return x, y
    else:
        return None, None
```

Obr. 8 Funkce find_center(matrix)

Funkce *find_center(matrix)* po zavolání zkontroluje, zdali je matice prázdná. Pokud není prázdná, nalezne největší a nejmenší hodnoty pro x a y a tyto hodnoty zprůměruje a vrátí střed se souřadnicemi (x, y). Pokud matice je prázdná, tak vrátí prázdné hodnoty.

```
if xT:
    cv.drawMarker(frame, (xT, yT), (0, 255, 0))
    cv.putText(frame, "[" + str(xT) + ", " + str(yT) + "]", (0, 30), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0))
else:
    cv.putText(frame, "Laser nenalezen", (0, 30), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0))

cv.imshow("frame", frame)
```

Obr. 9 Kód pro vložení textu a popisovače do obrázku

Pokud tedy existuje souřadnice xT, funkce *drawMarker(obrazek, pozice, barva)* zakreslí do *obrazek* křížek na definované souřadnice o definované barvě. Funkce *putText(obrazek, text, pozice, font, meritko, barva)* vloží do obrázku text obsahující souřadnice středu. Pokud však je souřadnice xT prázdná, funkce pouze vypíše, že laser nebyl nalezen.

Obr. 10 Ukázka výstupu ze skriptu. Vlevo je výstup z kamery se zakresleným textem a křížkem. Vpravo je zpracovaný bitový obraz

6.2 Detekce kruhových objektů dle velikosti plochy

Druhou ukázkou je detekce kruhů podle zadané velikosti. Jako záznamové zařízení bude opět použit mobilní telefon.

```python
def find_circles(image, minArea, maxArea): # najde kruhy v obraze
    # image - vstupní obraz
    # minArea - minimální plocha požadovaného kruhu
    # maxArea - maxiimální plocha požadovaného kruhu

    gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)

    invGray = cv.bitwise_not(gray)
    _, invGray = cv.threshold(invGray, 100, 255, cv.THRESH_BINARY)
    contour_list = []
    centers = []

    contours, hier = cv.findContours(invGray, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)
    for contour in contours:
        if (cv.contourArea(contour) > minArea) & (cv.contourArea(contour) < maxArea):
            if check_if_circle(contour):
                contour_list.append(contour)
                centers.append(find_center(contour))

    cv.drawContours(image, contour_list, -1, (0, 255, 0), thickness=2)
    for center in centers:
        cv.drawMarker(image, (center[0], center[1]), (0, 255, 0), thickness=2)

    return centers, contour_list, image
```

Obr. 11 Funkce pro hledání kruhů v obrázku

Funkce má na vstupu *image, minArea* a *maxArea*. Jako výstup jsou nalezené středy kruhy, kontury kruhů a obrázek s nakreslenými kruhy a jejich středy. Nejprve je obraz převeden do černobílého spektra. Poté je kvůli následnému použití prahování potřeba obraz pomoci funkce *bitwise_not(obraz)* invertovat. Invertovaný obraz je poté prahován. Následuje nalezení kontur. Zde se využívá funkce *findCountours(obraz, mod, metoda)*, ze které je výstup pole kontur, kdy každá kontura je jedna položka v poli a jejich hiearchie. Následně se pomocí funkce *contourArea(kontura)* zkontroluje, zda-li plocha kontury odpovídá zadanému intervalu, následně se kontroluje, jestli kontura přibližně odpovídá kruhu pomocí funkce *check_if_circle(kontura)*. Kruhové kontury odpovídající velikosti jsou následně uloženy do proměnné, jsou nalezeny jejich středy a na dalších řádcích vykresleny do obrázku.

```python
def check_if_circle(contour): # zjistí zda kontura odpovídá kruhu
    x = []
    y = []
    for point in contour:
        x.append(point[0][0])
        y.append(point[0][1])

    xArea = pi * ((max(x) - min(x)) / 2) ** 2
    yArea = pi * ((max(y) - min(y)) / 2) ** 2
    area = abs(cv.contourArea(contour, True))

    if (abs(xArea-area)/area <= 0.05) & (abs(yArea-area)/area <= 0.05):
        return True
    else:
        return False
```

Obr. 12 Funkce, která kontroluje, jestli kontura odpovídá kruhu

Funkce *check_if_circle(kontura)* nejprve vytvoří pole souřadnic x a y. Následně z rozdílu maximální a minimální hodnoty pozice vypočítá plochy podle x a y. Tyto plochy nakonec srovná se zabudovanou funkcí OpenCV *contourArea(kontura, uzavrena_krivka?)* a pokud se liší o maximálně 5 %, vrátí True, v opačném případě False.

Obr. 13 Výstup z druhého ukázkového skriptu

7 ZÁVĚR

Hlavním úkole této bakalářské práce bylo seznámení se s vybranými opensourcovými počítačovými knihovnami. První část byla věnována pro seznámení s počítačovým viděním, co to vlastně je. V další části byl čtenář seznámen s definicí opensourcového softwaru a některých licencí. Licence byly vybrány na základě použitých knihoven.

V kapitole 3 proběhlo představení jednotlivých knihoven. I když knihoven existuje mnohem více, pro tyto existuje spousty materiálů a podkladů, rozsáhla dokumentace a podpora, proto byly vybrány právě tyto. Všechny knihovny mají širokou škálu použití, zejména pak dlib, který slouží i jako všeobecná knihovna obsahující mnoho jiných užití mimo počítačové vidění. Nejvíce používanou však jednoznačně zůstává OpenCV, vzhledem k velkému počtu podkladů a využití.

Následovalo srovnání těchto knihoven. Nejdříve byly srovnány možnosti využití programovacích jazyků. Zde je možné vidět, že OpenCV je nejuniverzálnější knihovnou, pro kterou existuje podpora pro všechny zmíněné jazyky. Na druhé straně stojí VXL, který je jen a pouze pro jazyk C++, s využitím softwaru třetích stran pak i pro Python. C++ je však velice rozšířený jazyk, který najde uplatnění zejména při nízkoúrovňovém programování. Python je sice pomalejší, ale jeho nasazení je mnohem rychlejší a psaní kódu taktéž. V další části srovnání se srovnávala dokumentace a dostupné podklady. Zde opět vede OpenCV vzhledem k nepřebernému množství materiálů, ať už fyzických či elektronických. V přehledu použití knihoven, pak naopak má dlib navrch, jelikož sami autoři spravují seznam známých uživatelů v praxi. Nakonec proběhlo srovnání výkonu. Pro srovnání výkonu neexistuje mnoho prací a pro tyto 4 knihovny se povedlo najít pouze jednu práci a jeden test z knihy. Každopádně v obou případech je OpenCV jednoznačně rychlejší než ostatní srovnávané knihovny. Jedná se ale o pouhé 2 testy, výsledky v jiných oblastech se mohou výrazně lišit.

V poslední kapitole proběhlo vypracování dvou vzorových úloh. První úloha je velice jednoduchá a slouží jako takový úvod do syntaxe knihovny. Bylo zde seznámeno s načtením obrázku z webkamery, použití základních funkcí, jako Gaussovské rozmazání, prahování a vkládání textu do obrázku a následně jeho zobrazení. Ve druhé úloze pak došlo ještě k bližšímu seznámení, kdy se za pomocí kontur a trochu matematiky z obrazu extrahoval kruh o požadované velikosti.

V případech praktických ukázek lze narazit na několik úskalí. Prvním z nich je špatné osvětlení nebo heterogenní pozadí. Zde je potřeba správně nastavit parametry Gaussovského rozmazání,

prahování případně různých detektorů hran. Toto nastavení je opravdu důležité a může znamenat rozdíl mezi funkčním a nefunkčním programem. Dalším problémem byla právě ve druhé úloze detekce kruhových objektů. OpenCV sice obsahuje funkci *HoughCircles()* právě pro detekci kruhových objektů, avšak její zprovoznění bylo neúspěšné. Kruhovitost se tedy testuje vlastní funkcí *check_if_circle()*. Funkce změří průměr horizontálně a vertikálně a z těchto hodnot vypočítá pro každý průměr obsah plochy, kterou porovná s výpočtem obsahu implementovanou funkcí *contourArea()*. Rozdíl 5 % se může změnit, avšak tato hodnota se zatím osvědčila

8 SEZNAM POUŽITÝCH ZDROJŮ

[1] SZELISKI, Richard. *Computer Vision: Algorithms and Applications*. London: Springer, 2010, xx, 812 s. : il. ISBN 978-1-84882-934-3.

[2] BRADSKI, Gary R a KAEHLER, Adrian. *Learning OpenCV*. Sebastopol: O'Reilly, 2008, xvii, 555 s. : ill. ISBN 978-0-596-51613-0.

[3] LEVIN, Golan. Image Processing and Computer Vision. *ofBook - Image Processing and Computer Vision* [online]. [cit. 2021-02-15]. Dostupné z: https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html

[4] THOMPSON, Mike. Please Enable Cookies. *StackPath* [online]. 4 May 2015. [cit. 2021-02-15]. Dostupné z: https://www.electronicdesign.com/technologies/embedded-revolution/article/21800754/conquer-the-challenge-of-integrating-efficient-embedded-vision

[5] About the Open Source Initiative | Open Source Initiative. *Open Source Initiative* [online]. [cit. 2021-05-13]. Dostupné z: https://opensource.org/about

[6] About - OpenCV. *OpenCV* [online]. Dostupné z: https://opencv.org/about/

[7] License - OpenCV. *OpenCV* [online]. Dostupné z: https://opencv.org/license/

[8] Dlib C++ Library – Introduction. *Dlib* [online]. 28.10.2015 [cit. 15.05.2021]. Dostupné z: http://dlib.net/intro.html

[9] OpenCV: Introduction. *OpenCV* [online]. Dostupné z: https://docs.opencv.org/master/d1/dfb/intro.html

[10] LOWE, David. The Computer Vision Industry. *The University of British Columbia* [online]. 2015. Dostupné z: https://www.cs.ubc.ca/~lowe/vision.html

[11] MEEL, Vidushi. 56 Most Popular Computer Vision Applications in 2021 | viso.ai. *Computer Vision Application Platform | viso.ai* [online]. Copyright © 2021 viso.ai [cit. 23.07.2021]. Dostupné z: https://viso.ai/applications/computer-vision-applications/

[12] ALARCON, Nefi. Automatically Segmenting Brain Tumors with AI | NVIDIA Developer Blog. *NVIDIA Developer* [online]. Copyright © 2021 NVIDIA Corporation [cit. 23.07.2021]. Dostupné z: https://developer.nvidia.com/blog/automatically-segmenting-brain-tumors-with-ai/

[13] The 3-Clause BSD License | Open Source Initiative. *Open Source Initiative* [online]. Dostupné z: https://opensource.org/licenses/BSD-3-Clause

[14] VEDALDI, Andrea a Brian FULKERSON. Vlfeat. In: *Proceedings of the international conference on Multimedia - MM '10* [online]. New York, New York, USA: ACM Press, 2010, 2010, s. 1469- [cit. 2021-7-24]. ISBN 9781605589336. Dostupné z: doi:10.1145/1873951.1874249

[15] BOKHAN, Kostiantyn. Computer vision in manufacturing: What, Why, and How?. *Eastern European Software Development Outsourcing Provider* [online]. Dostupné z: https://www.n-ix.com/computer-vision-manufacturing/

[16] Top Use-Cases for Computer Vision in Manufacturing - Visionify. *Custom Computer Vision & AI Solutions Provider | Visionify* [online]. Copyright © 2021 [cit. 24.07.2021]. Dostupné z: https://visionify.ai/top-use-cases-computer-vision-in-manufacturing/

[17] Apache License, Version 2.0 | Open Source Initiative. *Open Source Initiative* [online]. Dostupné z: https://opensource.org/licenses/Apache-2.0

[18] BOYKO, Nataliya, Oleg BASYSTIUK a Nataliya SHAKHOVSKA. Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library. In: *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)* [online]. IEEE, 2018, 2018, s. 478-482 [cit. 2021-7-25]. ISBN 978-1-5386-2874-4. Dostupné z: doi:10.1109/DSMP.2018.8478556

[19] OpenCV: OpenCV modules. *OpenCV documentation index* [online]. Dostupné z: https://docs.opencv.org/4.5.3/

[20] VXL Documentation. *VXL - C++ Libraries for Computer Vision* [online]. Dostupné z: https://vxl.github.io/doc/release/index.html

[21] VXL: . *VXL - C++ Libraries for Computer Vision* [online]. Dostupné z: https://vxl.github.io/doc/release/books/core/book.html

[22] BAHMBHATT, Samarth. *Practical OpenCV*. Apress, 2013. ISBN 978-1-4302-6080-6.

[23] I. Culjak, D. Abram, T. Pribanic, H. Dzapo and M. Cifrek, "A brief introduction to OpenCV," 2012 Proceedings of the 35th International Convention MIPRO, 2012, pp. 1725-1730.

[24] KING, Davis E. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research 10*, 2009, s. 1755-1758

[25] VLFeat - Tutorials. *VLFeat* [online]. Copyright © 2007 [cit. 25.07.2021]. Dostupné z: https://www.vlfeat.org/overview/tut.html

[26] VLFeat - Documentation. *VLFeat* [online]. Copyright © 2007 [cit. 25.07.2021]. Dostupné z: https://www.vlfeat.org/doc.html

[27] GitHub - mit-nlp/MITIE: MITIE: library and tools for information extraction. *GitHub* [online]. Copyright © 2021 GitHub, Inc. [cit. 29.07.2021]. Dostupné z: https://github.com/mit-nlp/MITIE

[28] Rose Compiler – Program Analysis and Transformation. Rose Compiler – Program Analysis and Transformation [online]. Copyright ©2021 Rose Compiler [cit. 29.07.2021]. Dostupné z: http://rosecompiler.org/

[29] dlib C++ Library / Wiki / Known_users. *SourceForge - Download, Develop and Publish Free Open Source Software* [online]. Copyright © 2021 Slashdot Media. All Rights [cit. 29.07.2021]. Dostupné z: https://sourceforge.net/p/dclib/wiki/Known_users/

[30] GitHub - tzutalin/dlib-android: Port dlib to Android. *GitHub: Where the world builds software · GitHub* [online]. Copyright © 2021 GitHub, Inc. [cit. 01.08.2021]. Dostupné z: https://github.com/tzutalin/dlib-android

[31] BRANDON, Amos, LUDWICZUK, Bartosz a SATYANARAYANAN, Mahadev. *OpenFace: A general-purpose face recognition library with mobile applications*. CMU-CS-16-118, CMU School of Computer Science. 2016

[32] About - Cubic Motion. *Cubic Motion: The World's Definitive Solution for Facial Animation* [online]. Copyright © 2011 [cit. 01.08.2021]. Dostupné z: https://cubicmotion.com/about/

[33] Boost Software License 1.0 (BSL-1.0) | Open Source Initiative. *Open Source Initiative* [online]. Dostupné z: https://opensource.org/licenses/BSL-1.0

[34] The zlib/libpng License (Zlib) | Open Source Initiative. *Open Source Initiative* [online]. Dostupné z: https://opensource.org/licenses/Zlib

9 SEZNAM ZKRATEK, SYMBOLŮ, OBRÁZKŮ A TABULEK

9.1 Seznam tabulek

9.2 Seznam obrázků

10 SEZNAM PŘÍLOH

- ukazka_1.py
- ukazka_2.py