FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Bakalářská práce

On-line systém pro počítačové vidění

Studijní program:	B0613A140005 Informační technologie
Studijní obor:	Inteligentní systémy
Autor práce:	**Matěj Navrátil**
Vedoucí práce:	doc. Ing. Josef Chaloupka, Ph.D. Ústav informačních technologií a elektroniky

Liberec 2023

FAKULTA MECHATRONIKY,
INFORMATIKY A MEZIOBOROVÝCH
STUDIÍ TUL

Zadání bakalářské práce

On-line systém pro počítačové vidění

Jméno a příjmení:	**Matěj Navrátil**
Osobní číslo:	M19000031
Studijní program:	B0613A140005 Informační technologie
Specializace:	Inteligentní systémy
Zadávající katedra:	Ústav informačních technologií a elektroniky
Akademický rok:	2022/2023

Zásady pro vypracování:

1. Seznamte se s problematikou zpracování a rozpoznávání obrazu.
2. Navrhněte a realizujte systém (v Pythonu) pro poloautomatické zpracování a rozpoznávání obrazu.
3. Tento systém by měl obsahovat přívětivé prostředí, ve kterém budou použity jednotlivé algoritmy pro zpracování a rozpoznávání obrazu z knihovny OpenCV. Celý systém musí být navržen tak, aby byl ”otevřený”, tj. aby se v budoucnu daly do programu přidávat nové moduly.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] Šonka, M., Hlaváč, V., Boyle, R.: Image processing, analysis, and machine vision. Fourth Edition. Australia: Cengage Learning, ISBN 978-1-133-59369-0, 2015.

[2] Gonzales, R. C., Woods, R: Digital image processing. Global edition. New York: Pearson, ISBN 978-1-292-22304-9, 2017.

[3] Hlaváč, V., Sedláček, M.: Zpracování signálů a obrazů. 2. přeprac. vyd. Praha: ČVUT, 255 s. ISBN 978-80-01-03110-0, 2007.

[4] Liu, Y.: Deep Learning Based Image Processing: Recent Advances and Future Trends. In Eliva Press, 2022. ISBN 978-9994982554.

Vedoucí práce:	doc. Ing. Josef Chaloupka, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	24. října 2022
Předpokládaný termín odevzdání:	22. května 2023

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 24. října 2022

Prohlášení

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

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

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

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

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

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

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

22. května 2023 Matěj Navrátil

On-line systém pro počítačové vidění

Abstrakt

V rámci této bakalářské práce je navržen a zhotoven systém s uživatelským rozhraním, který umožňuje použití algoritmů z oblasti zpracování a rozpoznání obrazu. Systém používá existující realizace algoritmů převážně knihovnou OpenCV, v menším zastoupení pak knihovnami FLANN a dlib. Přínosem práce je zprostředkování těchto algoritmů přívětivým grafickým uživatelským rozhraním s ovládacími prvky parametrů algoritmu a aktualizací výstupu v reálném čase. Další vlastnosti a funkce systému i výčet a představení algoritmů jsou podrobně rozebrány v odpovídajících kapitolách.

Kapitola *Návrh systému* se zabývá požadavky a postupem návrhu systému pro zpracování a rozpoznání obrazu. Je zde rozebrán prototyp, na kterém byl návrh ozkoušen před zhotovením skutečného systému.

Kapitola *Realizace systému* je věnována specifikům uskutečnění systému podstatným pro koncového uživatele systému. Je zde popsáno rozhraní a uživatelovy možnosti práce s programem.

Kapitola *Dokumentace tříd systému* rozebírá konkrétní implementaci systému. Pozornost je zde věnována obzvláště třídám, kterými je realizovaná budoucí rozšiřitelnost systému.

Kapitola *Použité algoritmy práce s obrazem* popisuje konkrétní algoritmy, které systém nabízí. K vybraným algoritmům jsou uvedeny ukázky vytvořené právě v systému zhotoveném v rámci práce.

Klíčová slova: Strojové vidění, zpracování obrazu, rozpoznání objektů, detekování objektů, OpenCV, FLANN

On-line system for computer vision

Abstract

Within the scope of this bachelor thesis, a system featuring a user interface, which enables the use of various algorithms used in image processing and reognition, is designed and created. System uses existing implementations of algorithms primarily by the open source library OpenCV, to a lesser extent libraries FLANN and dlib. This work's contribution is then facilitating these algorithms via a user friendly graphical interface with widgets to control parameters of algorithms and real-time output updating. Other properties and functions of the system, as well as the introduction to specific image processing algorithms used, is found in respective chapters.

The chapter *System design* deals with system requirements specification and methods used to design the system for image processing and recognition. A prototype is analyzed, which was used to test design decisions.

Chapter *System implementation* is dedicated to system specifics significant for the end user. The interface and use cases are described here.

Chapter *Class documentation* analyzes the implementation of the system. Attention is given to classes which facilitate future extensibility of the system.

Chapter *Used image processing algorithms* describes specific algorithms, which the system offers. Select algorithms are accompanied with demonstration figures, which were produced exclusively using the system created in this bachelor thesis.

Keywords: Computer vision, image processing, object recognition, object detection OpenCV, FLANN

Poděkování

Rád bych zde poděkoval vedoucímu práce doc. Ing. Josefu Chaloupkovi, Ph.D. za čas, který mi věnoval při zpracování projektu nejen v podobě konzultací, věcné rady, užitečné zdroje, směřování práce správným směrem a trpělivost.

Obsah

Seznam zkratek

ANN	Approximate nearest neighbor
BRIEF	Binary Robust Independent Elementary Features
CenSurE	Center-Surround Extrema
FAST	Features from Accelerated Segment Test
FLANN	Fast Library for Approximate Nearest Neighbors
LMedS	Least median of squares
LSH	Locality-sensitive hashing
OpenCV	Open Source Computer Vision Library
ORB	Oriented FAST and rotated BRIEF
RANSAC	Random sample consensus
SIFT	Scale Invariant Feature Transform
SURF	Speeded-Up Robust Features

1 Úvod

Algoritmy zpracování a rozpoznání obrazu jsou nezbytnou součástí mnoha moderních automatických a poloautomatických systémů. Dovednost vytáhnout chtěnou informaci z běžného obrazu nachází uplatnění i v poměrně běžných situacích: Kdokoli vlastnící digitální fotoaparát nebo mobilní telefon s fotoaparátem například pořizuje komprimované snímky ve formátu JPG, definovaném posloupností operací nad obrazem. Z nich poté může sestavit panoramatický snímek stehováním obrazu, které využívá algoritmy hledání shody. Rekonstrukce vadných oblastí těchto snímků, zohlednění světelných podmínek nebo odstranění šumu jsou dalšími příklady.

Mnoho využití ale zpracování a rozpoznání obrazu nachází v úlohách typu detekce a klasifikace, kdy je z velkého množství dat v podobě obrazových bodů vytažena odpověď na otázky v praktičtější podobě. Příkladem takové úlohy je klasifikace pacienta dle lékařského snímku do kategorií zdravý nebo nemocný. V kombinaci se zpracováním neobrazových signálů je zpracování obrazu užitečným zdrojem informací pro úlohu analýzy silniční dopravy samořídícími vozy. Zpracování obrazové informace pro autonomní navigaci také využívají vozítka na Marsu, jejichž vzdálenost znemožňuje ruční ovládání v reálném čase. Šíře a důležitost uplatnění dává prostor pro vznik knihoven, které tyto algoritmy poskytují vývojářům pro rychlé a snadné použití ve specializovaných sytémech.

Systém navržený a zhotovený v rámci této práce si klade za cíl zprostředkování těchto algoritmů uživateli v podobě grafického rozhraní, ve kterém je možné rychle ozkoušet jakoukoli posloupnost algoritmů a hýbat s parametry a sledovat změny v reálném čase, ať už za účelem prototypování, předzpracování, nebo budování intuice.

Program je zhotoven v programovacím jazyce Python, který nabízí širokou řadu rychlých knihoven napsaných v jazycích C a C++. Všechny využité knihovny algoritmů zpracování obrazu, tj. OpenCV, FLANN a dlib, jsou knihovny v jazyce C++, které poskytují propojení (language binding) mimo jiné s jazykem Python.

OpenCV je rozsáhlá knihovna, která ke dnešnímu datu obsahuje přes 2500 algoritmů zpracování obrazu, a stále obdržuje aktualizace. Obsahuje podmnožinu NumPy funkcí optimalizovaných na 8-bitová čísla a dvojrozměrné, resp. tříkanálové obrazy za cenu újmy na obecnosti a algoritmy pokrývající celé spektrum složitosti, od globálního prahování po komplexní metody parametrizace obrazu příznaky.

Knihovna FLANN je zaměřená na rychlé algoritmy hledání přibližného nejbližšího souseda, které nachází uplatnění při hledání shody mezi příznaky dvou obrazů. dlib obsahuje algoritmy a naučené modely z oblasti strojového učení, které jsou užity pro úlohy nalezení a rozpoznání obličeje.

2 Návrh systému

2.1 Požadavky na systém

Systém byl tak, aby uživateli umožnil:

- Použít libovolný ze specifikovaných algoritmů na vstupní obraz.
- Měnit parametry algoritmu přívětivými ovládacími prvky.
- Zobrazit srovnání vstupního a výstupního obrazu.
- Zobrazit další informace o obrazech v závislosti na prostoru.
- Automaticky odhalit chyby šptaného prostoru obrazu nebo neplatné kombinace parametrů.
- Automaticky aktualizovat zobrazený výstup s ohledem na změny parametrů v reálném čase.
- Tvořit sekvenci algoritmů, kterými je obraz zpracován postupně.
- Načíst a uložit sekvenci použitých algoritmů spolu s jejich parametry.
- Dávkově zpracovat soubory aktuální sekvencí algoritmů.

2.2 Prototyp

Realizace požadavků byla nejdříve vyzkoušena na jednoduchém prototypu programu. Jako uživatelské rozhraní prototyp poskytoval POSIX-kompatibilní argumenty pro volání z příkazové řádky.

2.2.1 Návrh prototypu

Návrh prototypu bral ohled na budoucí rozšiřitelnost programu a zahrnoval abstrakci algoritmů zpracovnání obrazu jakožto třídu obsahující funkci o dvou parametrech, obraz a seznam parametrů, a informaci o tom, jaké parametry přijímá. Platnost parametrů byla hlídána v těle funkce.

Jedno volání programu z příkazové řádky opdovídá jednomu algoritmu zpracování obrazu. Výstup může být zapsán do souboru, nebo předán dalšímu volání programu.

Požadavek	Řešení v prototypu	Řešení v programu
Použití algoritmu	Abstraktní třídou	Řešení zachováno
Zobrazení	Uložením souboru	V uživatelském rozhraní
Formát parametrů	Parsované ve funkci	Typ určen ovládacím prvkem
Ošetření parametrů	V těle funkce	Zamezení ovládacím prvkem
Řetězení algoritmů	Bash operátor roury	Seznam v rámci programu
Ošetření prostoru	V těle funkce	Dle polí třídy algoritmu
Uložení posloupnosti	Formát bash skriptu	Formát csv

Tabulka 2.1: Srovnání prototypu s výsledným programem.

2.2.2 Realizace prototypu

Rozhraní prototypu tvoří čtyři POSIX-kompatibilní argumenty:

- `-f`: Funkce, kterou bude vstupní obraz zpracován
- `-p`: Seznam parametrů zvolené funkce
- `-i`: Vstupní soubor, jinak očekáván na standardním vstupním proudu
- `-o`: Výstupní soubor, jinak vytisknut na standardní výstupní proud

Vynechání parametrů vstupního a výstupního souboru umožňuje předání mezivýsledného obrazu bash operátorem roury v podobě base64 kódování nekomprimovaných binárních dat obrazu.

Prototyp poskytoval drobnou podskupinu požadovaných algoritmů výsledného programu, jmenovitě Gaussovo rozmazání, (adaptivní) prahování, zachování rozsahu hodnot, a změnu barevného prostoru.

2.2.3 Závěry z fáze prototypu

Srovnání realizace požadavků v prototypu a výsledném programu je vidět v tabulce 2.1. Struktura abstrakce algoritmů zpracování obrazu byla zachována a rozšířena o další vlastnosti, jako vstupní a výstupní obrazový prostor a podrobnější informace o parametrech. Vytažení zpracování parametrů z funkce do abstraktní třídy algoritmu se váže na abstrakci ovládacích prvků parametrů.

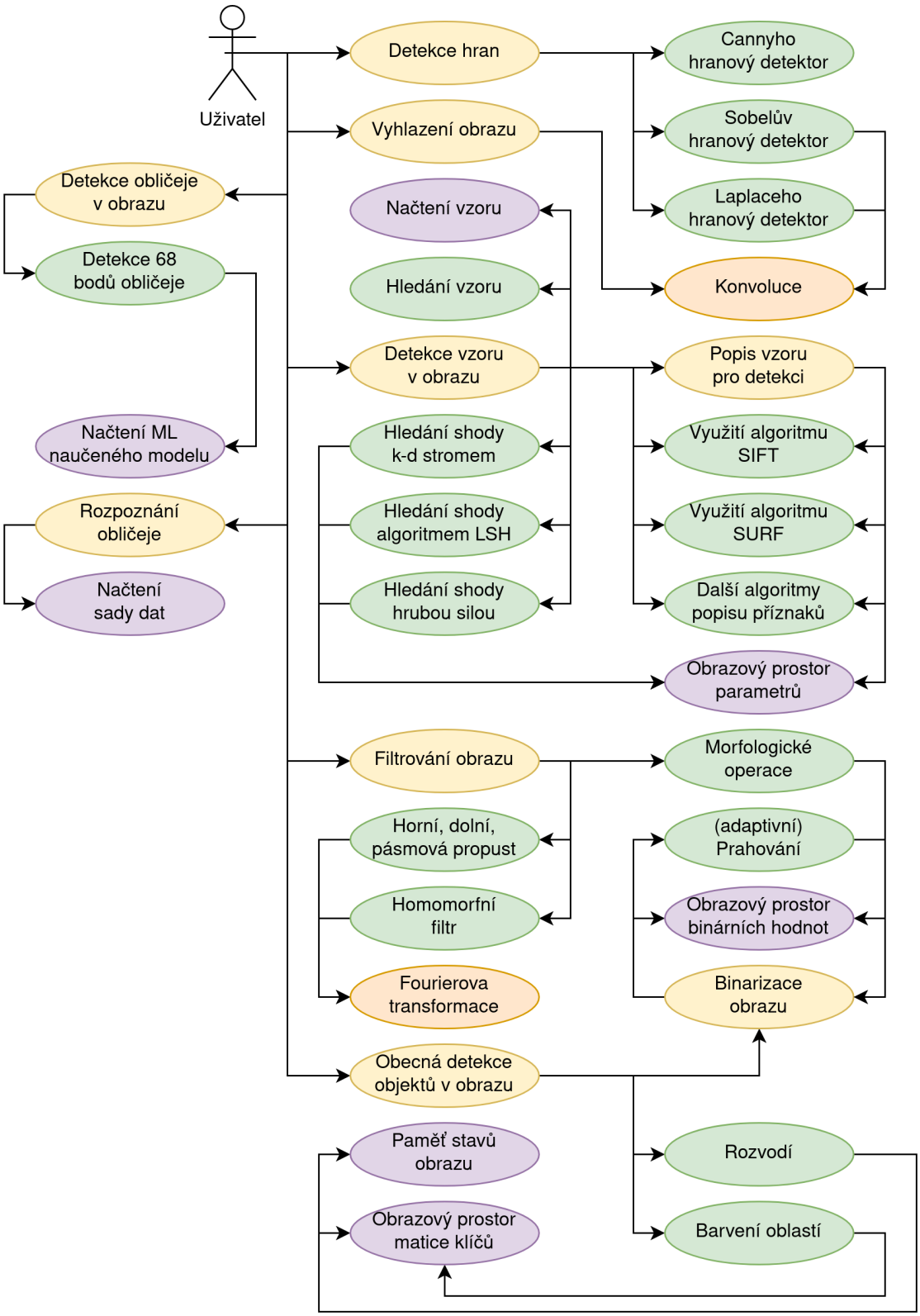
Některá řešení prototypu byla předělána pro jiné účely, např. uložení souboru; Případy použití poskytnuté jednoduchým použitím prototypu v bash skriptu (řetězení, dávkové zpracování) ve výsledném programu obsluhují cykly nad existujícím kódem.

Ve výsledném programu je změna parevného prostoru nahrazena funkcí vytažení komponenty, která vnitřně zahrnuje převod RGB obrazu do barevného prostoru, ze kterého je požadovaná komponenta vytažena. Předávání barevného obrazu v různých barevných prostorech neneslo žádnou výhodu oproti předávání barevného obrazu v podobě RGB a převedení v rámci funkce, která může být nastavena pro práci s jiným barevným prostorem.

2.3 Diagram použití systému

Následující usecase diagram, 2.1, popisuje částečný výčet typů úloh, které uživatel může s pomocí systému řešit. Jsou vynechány funkce programu společné pro prakticky všechny řešené úlohy, například načtení obrazu, uložení posloupnosti, vytažení komponenty nebo pomocné funkce vizualizace výsledku.

Žlutě jsou v diagramu vyznačeny typy úloh řešitelné systémem, zeleně konkrétní algoritmy zprostředkované systémem, fialově pomocné funkce programu a oranžově vnitřní funkce sdílené několika algoritmy.



Obrázek 2.1: Diagram použití systému.

3 Realizace systému

3.1 Interakce uživatele se systémem

3.1.1 Panel posloupnosti operací

V panelu posloupnosti operací může uživatel přidávat a odebírat funkce postupně použité na obraz a měnit jejich pořadí. Rozhraní poskytuje několik možností, jak těchto operací docílit:

- Tlačítky vedle seznamu operací.
- Pravým tlačítkem myši.
- Skrze menu ”Process” v liště s menu.
- Klávesovými zkratkami, zobrazenými v menu ”Process.”

Kromě algoritmů zpracování a rozpoznání obrazu do seznamu uživatel může přidat pomocné funkce, například komentář, uložení stavu obrazu v konkrétním bodě průběhu sekvence pro pozdější použití, nebo překrytí barevného obrazu binárním nebo objektovým výstupem pro snadnější vizualizaci.

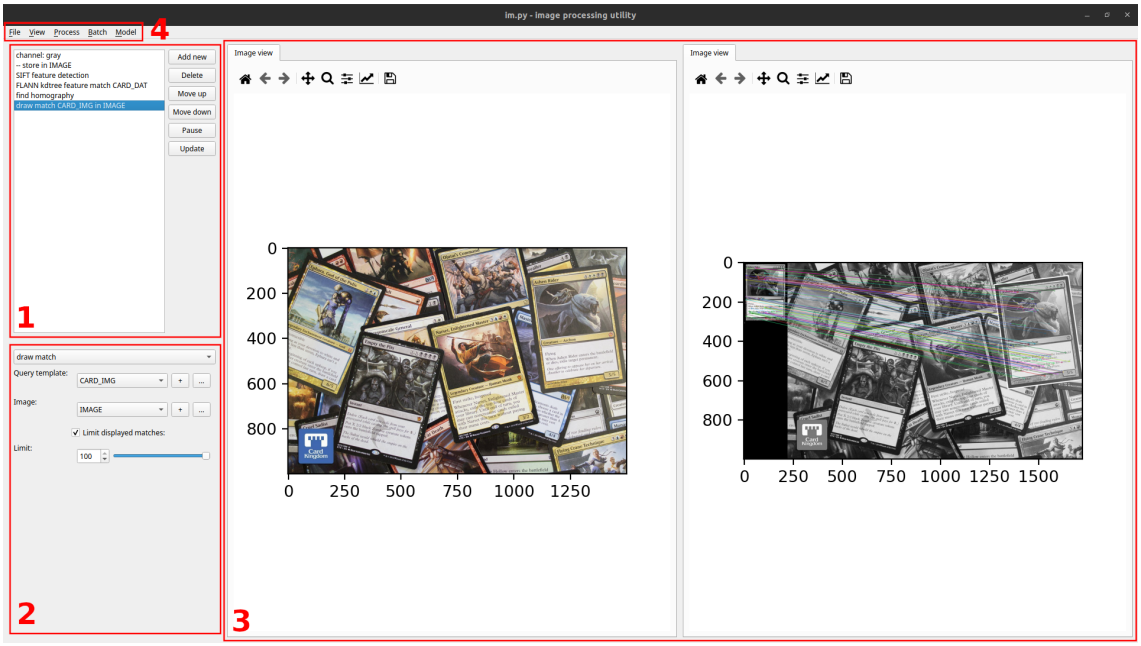
V případě, že program zachytí chybu nebo odhalí neplatné použití algoritmu (špatná kombinace parametrů, špatný prostor obrazu vcházejícího do algoritmu), chyba je zobrazena ikonou vedle názvu algoritmu na místě, kde nastala; Text chyby je dostupný v nápovědné bublině.

Některé algoritmy a pomocné funkce mají parametry, které je vhodné vidět na první pohled, a tedy v seznamu zobrazují parametrizované názvy.

3.1.2 Panel s parametry

V panelu jsou zobrazeny ovládací prvky parametrů aktuálně zvýrazněné operace v panelu posloupnosti operací, uživatel může přistoupit k parametrům kterékoli instance operace klepnutím myši na opdovídající řádek v seznamu.

Aplikace reague v reálném čase na změny hodnot v tomto panelu, a to i během plynulé manipulace s ovládacím prvkem posuvníku. Pro výpočetně náročné operace může být vhodné pozastavit automatické aktualizace výstupu tlačítkem „Pause“ nacházejícím se v tlačítkové části panelu posloupnosti. Výstup je možné ručně obnovit po změnách tlačítkem „Update“ ve stejné části.



Obrázek 3.1: Rozhraní programu. (1) panel posloupnosti operací, (2) panel s parametry, (3) oblast vizualizace, (4) lišta s nástroji.

3.1.3 Oblast vizualizace

Program zobrazuje efekt sekvence funkcí podobou srovnání vstupního a výstupního obrazu ve dvou podoknech.

Každé podokno obsahuje lištu nástrojů z knihovny `matplotlib` pro přiblížení a pohyb v přiblíženém obraze, zpět/vpřed v historii přiblížení, a návrat na původní přiblížení.

Nad lištou nástrojů se nachází lišta karet, kterými lze přepínat mezi pohledy na obraz, pokud tak prostor aktuálně zobrazovaného obrazu umožňuje. Konkrétně v tuto chvíli existují alternativní pohledy pro následující prostory:

- Šedotónový obraz: Histogram, spektrogram
- Matice klíčů: Histogram, pohled rozvodí (zatmavený klíč odpovídající pozadí)

3.1.4 Lišta s nástroji

Lišta kromě alternativního přístup k uživatelské interakci se sekvencí algoritmů obsahuje sadu pomocných funkcí:

- Načtení a uložení obrazu a sekvence operací s parametry
- Načtení a správa vedlejších souborů a proměnných paměti stavů obrazu
- Dávkové zpracování obrazů aktuální sekvencí operací

Dále je zde možné přizpůsobit rozložení prvků uživatelského rozhraní v okně.

Typ	Načtení	Uložení	Vedlejší soubor
Barevný obraz	ano	ano	ano
Šedotónový obraz	ano	ano	ne*
Soubor objektů	ne	ano	ano

Tabulka 3.1: Platná práce s obrazovými soubory. *Načten jako barevný.

3.2 Formáty souborů

3.2.1 Obrazu

Program umožňuje práci s barevnými a šedotónovými obrazovými soubory formátů .jpg a .png, a se soubory .json obsahující obraz v objektovém prostoru, například prostoru příznaků typu SIFT nebo SURF. Barevné obrazy je možné načíst jako šedotónové okamžitým vytažením komponenty při načtení, nebo převést na šedotónové funkcí vytažení komponenty.

3.2.2 Sevence operací

Operace jsou ukládány do souborů s koncovkou `.v1.csv` s úmyslem zachování zpětné kompatibility v případě budoucího nového formátu. Každý řádek obsahuje identifikátor operace následovaný jednotlivými parametry v pořadí jejich definice, obojí ve formátu hodnot oddělených čárkami.

3.3 Pomocné funkce programu

3.3.1 Načtení vedlejších souborů

Některé algoritmy zpracování obrazu vyžadují podstatně složitější parametry než skaláry, booleovské hodnoty či výběr z listu. Tyto případy obstarává možnost načtení parametrů v podobě souborů nebo dokonce skupin souborů.

Uživatel může načíst obraz v typickém prostoru nebo v objektovém prostoru pro použití jako šablona. Vyhledávání objektu v obraze, ať vyhledáváním šablony nebo hledáním shod v prostoru příznaků, vyžaduje obraz reprezentující hledaný objekt a obraz, ve kterém je vyhledáván, a který byl volen jako vstup.

Algoritmus detekce obličeje a 68 význačných bodů obličeje pracuje s neuronovou sítí, kterou je nutno stáhnout externě a načíst do programu. Výhodou tohoto přístupu je možnost vlastního přetrénování sítě jinými prostředky.

Rozpoznání obličeje vyžaduje sadu příkladů pro každou rozpoznávanou osobu.

3.3.2 Paměť stavů obrazu

Část algoritmů a pomocných funkcí poskytovaných systémem pracuje s několika různě zpracovanými kopiemi vstupního obrazu, paměť stavů tedy umožňuje uložit

obraz zpracovaný jedním způsobem, načíst starší stav, a zpracovat jej další sekvencí operací.

Uložení a načtení obrazu v konkrétní části sekvence algoritmů zprostředovávají funkce vložené do této sekvence. Pod jakým názvem proměnné se obraz uloží specifikuje parametr s ovládacím prvkem rozbalovacího seznamu. Obrazové proměnné lze vytvořit a smazat skrze tlačítka vedle rozbalovacího seznamu, nebo pomocí menu „Model“ v liště s menu.

Algoritmy vyžadující více podob orbazu využívají totožný ovládací prvek.

3.3.3 Funkce pro prezentaci výsledků

Pro některé úlohy není prezentace výsledků srovnáním vstupu a výstupu postačující, pro tyto případy program poskytuje pomocné funkce překrytí a zobrazení nalezené shody.

3.3.4 Dávkové zpracování obrazů

Program umožňuje použít aktuální posloupnost operací pro zpracování celé složky obrazových souborů najednou, pro účely zpracování datasetu pro použití v rámci programu i mimo program. Uživatel zvolí vstupní složku, výstupní složku, a suffix pro názvy výstupních souborů vložený před koncovku. Neobrazové soubory ve vstupní složce jsou automaticky ignorovány.

3.3.5 Parametry z příkazové řády

Spuštění programu příkazovou řádkou nebo přes zástupce podporujícího parametry z příkazové řádky umožňuje ovlivnit počáteční stav programu několika způsoby:

- `-i, --image PATH`: Načtený obraz, barevně pokud není určeno jinak.
- `-c, --channel CHANNEL`: Komponenta vytažená z obrazu načteného `-i`.
- `-p, --procedure PATH`: Cesta k souboru posloupnosti operací pro načtení.
- `-t, --template PATH`: Cesta k obrazu pro načtení jako vedlejší soubor. Umožňuje načíst i obraz v prostoru příznaků ze souboru formátu .json. Parametr může být specifikován vícenásobně.
- `-L, --layout INDEX`: Volba rozvržení okna 1-5 dle pořadí v menu „View“.
- `-M, --maximize`: Okno bude spuštěno zvětšené na celou obrazovku

Parametry jsou přebírány POSIX-kompatibilním způsobem. Nápovědu k parametrům lze zobrazit patametrem `-h`.

4 Dokumentace tříd systému

Následující kapitola je věnována dokumentaci použitých tříd a jejich metod. Vynechány zde jsou třídy a funkce obalující konkrétní implementace algoritmů zpracování obrazu; Rozboru konkréních algoritmů je věnována vlastní kapitloa, zde jsou reprezentovány jejich abstraktní třídou. Konkrétní obrazové prostory a ovládací prvky jsou rozebrány v rámci sekcí odpovídajících jejich abstraktním třídám.

4.1 ImageProcess

Třída `ImageProcess` poskytuje abstrakci nad všemi algoritmy zpracování obrazu, které uživatel může v rámci programu použít. Každá instance třídy v modelové části programu odpovídá jednomu řádku v panelu posloupnosti operací v grafické části programu. Instance třídy obsahuje jméno algoritmu, funkci odpovídající algoritmu, seznam přijímaných parametrů typu `InputSource`, kontejner jejich aktuálních hodnot a odkaz na funkci aktualizace výstupu dále delegovanou ovládacím prvkům parametrů.

Metoda `apply(self, subject)` aplikuje algoritmus s parametry na příchozí obraz a vrací výsledný obraz; Konkrétní algoritmus, který tato metoda volá, je určen podtřídou.

Metody `load(self, vals)` a `save(self)` umožňují načíst a uložit stav instance ve formátu seznamu hodnot parametrů.

Metoda `_add_arg(self, input_source)` přidá objekt definující parametr funkce a spojí jej odkazem s kontejnerem hodnot parametrů a funkcí aktualizace výstupu. Seznam parametrů je definován podtřídou, která implementuje abstraktní funkci `_add_args(self)`.

Metoda `assert_type(self, input_type)` obstarává kontrolu obrazového prostoru. Vstupní a výstupní prostor definuje podtřída ve funkci `type_constraints(self)`. Pokud je vstupní typ kompatibilní se vstupním prostorem definovaným podtřídou, metoda vrátí výstupní prostor, opačný případ je obsloužen výjimkou.

Metoda `display_settings_at(self, layout)` zobrazí ovládací prvky všech parametrů v panelu `layout`, metoda je využita pro realizaci panelu parametrů.

Metoda `long_name(self)` vrací parametrizovaný název algoritmu.

4.2 InputType

Třída `InputSource` slouží jako abstrakce nad různými typy parametrů a jejich grafickými ovládacími prvky. Obsahuje logiku sledování změn ovládacího prvku, okamžité reakce na změny získáním hodnoty, jejím zapsáním do kontejneru parametrů sdíleného se třídou `ImageProcess`, a zavoláním funkce aktualizace výstupu.

Metoda `connect(self, args, on_update)` je volána z kontextu `_add_arg` třídy `ImageProcess` a umožňuje sdílení kontejneru parametrů a delegaci aktualizace v reálném čase.

Metody `load_value(self, value)` a `save_value(self)` jsou realizovány podtřídami a obsluhují načtení a uložení hodnoty a nastavení odpovídajícího ovládacího prvku na správnou hodnotu.

Metoda `_store(self)` synchronizuje hodnotu v kontejneru parametrů s hodnotou ovládacího prvku. Čtení z ovládacího prvku realizuje podtřída abstraktní metodou `_get_new_value(self)`.

Metoda `_create_widget(self)` implementovaná podtřídou slouží k vytvoření ovládacího prvku.

4.2.1 IntSlider

Podtřída `IntSlider` je kombinovaný ovládací prvek vstupního pole s tlačítky přidat/odebrat a posuvníku, na kterém je možné volit rozsah celočíselných hodnot s určeným minimem, maximem a krokem.

4.2.2 StrSelect

Podtřída `StrSelect` umožňuje volbu ze seznamu platných textových hodnot prezentovaných uživateli podobou rozbalovacího seznamu.

4.2.3 ChannelPicker

Podtřída `ChannelPicker` poskytuje ovládací prvek o deseti přepínačích, které odpovídají třem kanálům pro každý ze tří barevných prostorů a jasu.

4.2.4 TemplateSelect

Podtřída `TemplateSelect` umožňuje volbu ze seznamu načtených vedlejších obrazových souborů prezentovaných uživateli podobou rozbalovacího seznamu. Ovládací prvek obsahuje dvě tlačítka pro načtení nového vedlejšího obrazového souboru a správu načtených vedlejších souborů.

4.2.5 ImageSelect

Podtřída `ImageSelect` umožňuje volbu ze seznamu uložených stavů zpracovávaného obrazu prezentovaných uživateli podobou rozbalovacího seznamu. Ovládací prvek

obsahuje dvě tlačítka pro vytvoření nové proměnné stavu obrazu a správu proměnných stavu obrazu.

4.2.6 IntVectorBox

Podtřída `IntVectorBox` odpovídá ovládacímu prvku vstupního pole celočíselných hodnot oddělených čárkami pro vektorový vstup, který nelze nahradit vhodnějšími prvky kvůli variabilní délce vektoru.

4.2.7 StrVectorBox

Podtřída `StrVectorBox` odpovídá ovládacímu prvku vstupního pole textových hodnot oddělených čárkami, konkrétně navrženému pro zadání libovolného počtu identifikátorů souborů dat.

4.2.8 BoolCheckBox

Podtřída `BoolCheckBox` reprezentuje dvoustavové zaškrtávací tlačítko a odpovídá binárnímu parametru.

4.3 ImageType

Třída `ImageType` poskytuje abstrakci nad prostory obrazu, ve kterých je předáván mezi jednotlivými algoritmy posloupnosti, umožňuje zobecnit kontrolu typové kompatibility dvou následujících algoritmů, načítání a ukládání obrazových souborů a vykreslení platných pohledů na obraz.

Metoda `can_accept(self, img_type)` po přepsání podtřídou umožňuje rozšířit jednoduchou kontrolu správnosti barevného prostoru rovností o složitější logiku pro implicitní převod obrazu, kde možné, například binárního obrazu na šedotónový.

Metodu `can_read(self, path)` může podtřída rozšířit a vrátit `True`, pokud je obrazový soubor podle koncovky vhodný ke zpracování touto třídou.

Metodami `read(self, path)` a `write(self, path, subject)` podtřída realizuje zapsání obrazu do souboru a čtení obrazu ze souboru.

Metody `create_input_tab_widget(self, tab)` a `create_output_tab_widget(self, tab)` umožňují zobrazení obrazu tohoto typu v oblastech přepínaných kartami ve vizualizační části okna, konkrétní pohledy a tedy karty definují podtřídy v metodách `_populate_input_tab_widget(self, tab)` a `_populate_output_tab_widget(self, tab)`, kde každý pohled tvoří pomocnou metodou `_create_pyplot_page(self, name, tab)`.

Metody `draw_input(self, subject, tab_index)` a `draw_output(self, subject, tab_index)` slouží k vykreslení konkrétního obrazu do vstupní nebo výstupní části vizualizační oblasti. `tab_index` slouží k optimalizaci, jen aktuálně zobrazený pohled je aktualizován. Barevný prostor obrazu na vstupu a výstupu určují, u které podtřídy je tato metoda volána.

4.3.1 RgbType

Podtřídou `RgbType` je reprezentován barevný prostor. Podtřída umí číst a psát soubory typu .jpg a .png a vykreslit obsah obrazu ve vizualizační části. Nezavádí žádné jiné pohledy na obraz. Po načtení RGB obrazu má uživatel možnost vytáhnout z obrazu komponentu a načíst obraz jako šedotónový.

4.3.2 GrayType

Podtřída `GrayType` zprostředkovává prostor stupňů šedi a pohled na reprezentaci obrazu barevnou mapou, histogram obrazu a spektrogram obrazu. Načtení šedotónového obrazu je obstaráno předzpracováním obrazu načteného třídou `RgbType`, třída obstarává uložení obrazu.

4.3.3 BinType

Podtřída `BinType` odpovídá prostoru matice jedniček a nul, binární obraz odpovídající rozdělení například na popředí a pozadí nebo hranové a nehranové body. Binární obraz nemůže být na vstupu, a implementuje tedy jenom metody souvislé s kreslením výstupu.

4.3.4 KeyType

Podtřída `KeyType` odpovídá prostoru matice klíčů, které odpovídají jednotlivým objektům v obraze. Klíčový obraz nemůže být na vstupu, a implementuje tedy jenom metody souvislé s kreslením výstupu. Třída poskytuje pohled reprezentace klíčů odstíny (hue), pohled zatmavení klíče pozadí z algoritmu rozvodí, a histogram reprezentace klíčů v obraze.

4.3.5 DatType

Podtřída `DatType` implementuje objektový prostor, ve kterém je obraz reprezentován příznaky a případně příznakovými vektory. Obrazy v objektovém prostoru lze načíst jen jako vedlejší soubor a uložit, třída tedy poskytuje jen metody vykreslení na výstupu, kde je obsah jednoduše vypsán - pro užitečnější pohled existují pomocné funkce v panelu posloupnosti operací. Soubory odpovídající objektovému prostoru jsou ukládány a načítány ve formátu JSON, pro kódování a dekódování nestandardních typů je použita třída `JsonAdapter`.

4.4 JsonAdapter

Třída `JsonAdapter` poskytuje zobecnění zápisu nestandardních typů do souborů JSON a jejich čtení ze souborů. Konkrétně je v programu využita pro zápis a čtení objektů `NumPy.ndarray` a `cv2.KeyPoint`.

Metoda `with_type(self, type, encoder, decoder)` v instanci třídy zaregistruje přijatý typ a náležící kodér a dekodér, dvě funkce. Funkce `encoder` převede objekt uvedeného typu na slovník a `decoder` slovník na objekt uvedeného typu.

Metoda `encode(self, obj)` přijímá objekt zaregistrovaného typu a aplikuje na něj náležící funkci `encoder`. Do výsledného slovníku je přidán klíč `$type` s hodnotou stringové reprezentace typu, než je metodou vrácen. Pokud typ nebyl zaregistrovaný, je vyhozena výjimka.

Metoda `encode(self, obj)` přijímá slovník a aplikuje na něj funkci `decoder` dle hodnoty pod klíčem `$type`. Pokud typ nebyl zaregistrovaný nebo slovník klíč identifikující typ nebosahuje, je vrácen původní slovník.

4.5 VariableContainer

Třída `VariableContainer` tvoří jmenný prostor pro načtené vedlejší obrazové soubory a proměnné stavů zpracovávaného obrazu. Skládá se ze dvou instancí třídy `VariableDict`, která rozšiřuje slovník a přepisuje metody `__getitem__(self, key)` a `__setitem__(self, key, value)`, ve kterých kromě původní implementace volá metody přebírané při inicializaci.

Metody `hook_image(self, func)` a `hook_template(self, func)` umožňují přidat funkce, které budou zavolány, když se změní stav obsažených slovníků. V programu jsou použity pro automatickou synchronizaci ovládacích prvků `TemplateSelect` a `ImageSelect` se jmenným prostorem načtených vedlejších souborů a proměnných stavů obrazu.

5 Použité algoritmy práce s obrazem

5.1 Operace nad barevným prostorem

5.1.1 Vytažení komponenty

V barevném obrazu každý bod obsahuje trojrozměrnou informaci, vytažení komponenty umožňuje obrazové body reprezentovat mapou skalárů intenzity, které lze např. porovnávat, prahovat nebo interpretovat jako výškovou mapu.

Program umožňuje vytáhnout komponentu z RGB, HSV a YCbCr reprezentace obrazu automatickým převedením, nebo obraz převést na jas.

5.1.2 Ekvalizace histogramu

Ekvalizace histogramu je funkce globálně aplikovaná na každý bod šedotónového obrazu a přibližuje histogram ideálnímu, rovnoměrnému histogramu. Z celého obrazu I o $M \times N$ bodech je vytvořena mapa z prostoru stupňů šedi do prostoru stupňů šedi:

$$T(g) = \left\lfloor \frac{H_c(g) - H_{min}}{MN - H_{min}} * 255 \right\rceil \tag{5.1}$$

Kde H_c je kumulativní histogram:

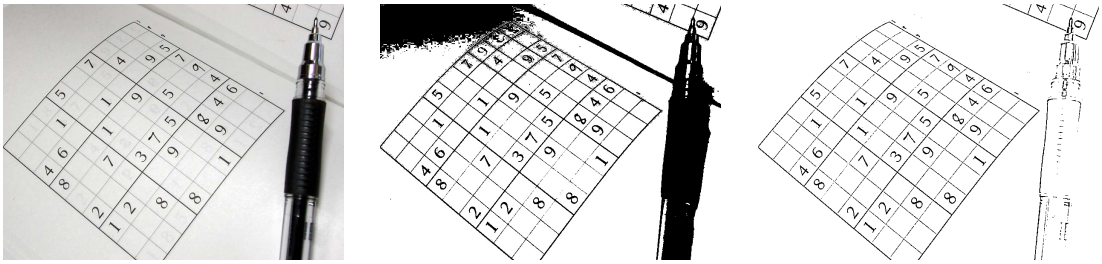
$$H_c(g) = \sum_{x,y} \begin{cases} 1 & \text{pokud } I_{x,y} \leq g \\ 0 & \text{pokud } I_{x,y} > g \end{cases} \tag{5.2}$$

g je stupeň šedi a H_{min} je první nenulová hodnota H_c. Intenzita každého výstupního bodu je funkcí jeho vstupní intenzity a této globální mapy. Body o stejné hodnotě na vstupu tedy nutně mají stejnou hodnotu i na výstupu. Ekvalizace zároveň zachovává nerovnosti.[1]

5.2 Binarizace obrazu

5.2.1 Globální prahování

Globální prahování porovná každý bod obrazu I s předem zvolenou prahovou hodnotou t, a na výstupu mu přiřadí binární hodnotu:



(a) Originál (b) Globální práh (c) Adaptivní práh

Obrázek 5.1: Porovnání globálního a lokálního prahování. Příklady vytvořeny v programu zrealizovaném v rámci práce. Ručně zadaný globální práh je příliš nízký pro hrany ve světlé části obrazu, ale příliš vysoký pro tmavou část.

$$g_{i,j} = \begin{cases} 1 & \text{pokud } I_{i,j} \geq t \\ 0 & \text{pokud } I_{i,j} < t \end{cases} \tag{5.3}$$

Jedná se o triviální způsob, jak šedotónový obraz převést do binárního prostoru, ale je nutné předem určit práh (ekvalizace může prahu dát intuitivnější roli) a neumožňuje zohlednit různé světelné podmínky v různých oblastech obrazu.[1]

5.2.2 Adaptivní prahování

Algoritmus adaptivního prahování umožňuje pro každý bod předpočítat vhodný práh z okolních obrazových bodů. Práh je nastaven na průměr $N \times N$ okolních bodů, případně jejich průměr vážený $N \times N$ maticí Gaussova vyhlazení pro zdůraznění v závislosti na vzdálenosti od bodu. K předpočteným prahům je přičten bias, který zamezí artefaktům v přibližně hladkých oblastech obrazu. Následně je každý bod zpracován stejně jako v případě globálního prahování, ovšem se svým náležícím prahem.[1]

5.3 Segmentace obrazu

5.3.1 Barvení oblastí

Barvení oblastí pracuje sekvenčně s body obrazu po řádcích a v rámci řádku zleva doprava. Na každý bod je aplikováno totožné jádro M, matice o velikosti 3x3, konstruované v závislosti na požadované spojitosti C:

$$M_{i,j} = \begin{cases} 1 & \text{pokud } |i-1| + |j-1| \leq C \\ 0 & \text{pokud } |i-1| + |j-1| > C \end{cases} \tag{5.4}$$

Jednička v jádře označuje sousednost. Vstupem algoritmu je binární obraz, kde jednička odpovídá popředí a nula pozadí. Když je jádro aplikováno na obrazový bod s hodnotou 1, je označkován v závislosti na označkovaných sousedech:

- Nová značka, pokud žádný soused není označkovaný
- Odpovídající značka, pokud alespoň jeden soused je označkovaný

Pokud se mezi označkovanými sousedy vyskytne více unikátních značek, skutečnost je zaznamenána, a v druhém kroku jsou tyto značky sloučeny.

Definice jádra je symetrická dle jejo středu, jedná se o centrosymetrickou matici. To odpovídá skutečnosti, že sousednost dvou bodů nezávisí na jejich pořadí. Pro každý symetrický pár jedniček platí, že v pořadí chodu algoritmu jedna vždy následuje za středem jádra, tedy nikdy neodpovídá již označkovanému obrazovému bodu. Jako optimalizace může tedy být nahrazena nulou.

5.3.2 Algoritmus rozvodí

Algoritmus rozvodí je založen na skutečných rozvodích, kde voda vždy teče dolů k lokálnímu minimu, kterému v reálném světě vypovídá ústí řeky či jezero. Algoritmus rozvodí pracuje s šedotónovým obrazem jako s výškovou mapou, označí každé lokální minimum jako objekt a následuje výpočtem rozvodí tohoto minima, tedy tvaru odpovídajícího objektu.

Obrazové body jsou zpracovány v pořadí dle intenzity (výšky) vzestupně, pro každou úroveň intenzity k:

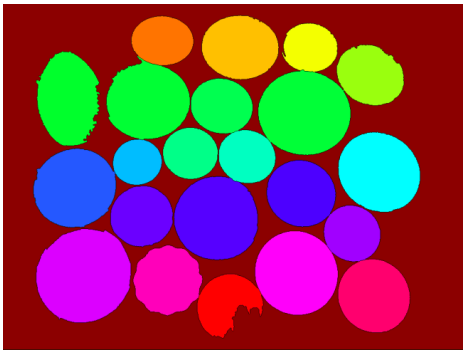
1. Za předpokladu, že každému bodu p o intenzitě $I_p < k$ je přiřazeno rozvodí, je pro každé rozvodí vypočtena geodesická oblast vlivu po úroveň k, tvořená spojitými body q, $I_q <= k$, které nejsou blíže jinému rozvodí platnému dle stejné podmínky.
2. Body odpovídající intenzity jsou postupně přiřazeny do rozvodí:
   (a) Pokud se bod nachází na rozhraní oblastí vlivu dvou nebo více rozvodí, je označen jako hranice,
   (b) Pokud bod náleží oblasti vlivu existujícího rozvodí, je mu přiřazen,
   (c) Pokud bod žádné oblasti vlivu nenáleží, jedná se o lokální minimum, kterému je vytvořeno nové rozvodí.

Protože v typickém obraze neodpovídá každé lokální minimum objektu, krok 2.c bývá nahrazen předurčeným souborem rozvodí nalezeným jinou formou segmentace.[1]

5.4 Morfologické operace

Binární morfologické operace jsou definovány nad obrazem v prostoru binárních hodnot, tj. obraz je reprezentován maticí jedniček a nul. Jednička odpovídá bodu popředí, nula bodu pozadí, a z této interpretace jsou odvozeny názvy konkrétních morfologických operací.

(a) Originál



(b) Objekty nalezené rozvodím

Obrázek 5.2: Ukázka nalezení tvaru rozvodím. Příklady vytvořeny v programu zrealizovaném v rámci práce. Barvení oblastí je součástí hledání objektů rozvodím, ale oddělení objektů vyžaduje značnou deformaci tvarů.

Morfologická transformace je určena vztahem obrazu a strukturního prvku. Některými běžnými strukturními prvky jsou například následující matice:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & \mathbf{1} & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 1 & \mathbf{0} & 1 \\ 0 & 0 & 0 \end{bmatrix}, \tag{5.5}$$

Kde tučně je zvýrazněn bod považován za střed, $O = (0, 0)$. Poslední příklad znázorňuje případ, kdy střed není součástí strukturního prvku, a tedy ve výsledku morfologické transformace každý bod závisí jen na okolí, ne na své hodnotě.

Obraz I lze vyjádřit jako množinu bodů X, kde $I_X = 1$, a strukturní prvek B jako množinu bodů X relativně vůči středu O o stejné podmínce, tedy $B_{X+O} = 1$. Množinové vyjádření druhého příkladového prvku by vypadalo následovně:

$$B = \{(-1, 0), (1, 0), (0, -1), (0, 1), (0, 0)\} \tag{5.6}$$

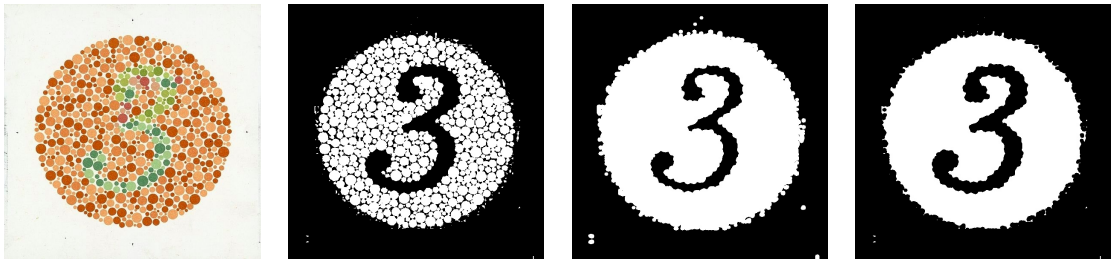
Pro morfologické operace bude uvažováno množinové vyjádření. Notace I_b označuje množinu vektorů $i \in I$ posunutých vektorem b:

$$X_b = \bigcup_{x \in X} x + b \tag{5.7}$$

5.4.1 Diletace a eroze

Výsledek diletace pro bod X je 1, pokud a pouze pokud po přiložení středu strukturního prvku alespoň jedna jednička prvku přiléhá jedničce v obraze. Matematicky je přiložení reprezentováno posunem množinového vyjádření obrazu o vektor množinového vyjádření strukturního prvku, a diletace tedy vypadá následovně:

$$I \oplus B = \bigcup_{b \in B} I_b \tag{5.8}$$



(a) Originál (b) Práh složky H (c) Jen diletace (d) Uzavření

Obrázek 5.3: Ukázka morfologických operací. Příklady vytvořeny v programu zrealizovaném v rámci práce. Diletace postačuje k zaplnění mezer, uzavření je nutné k zachování velikosti.

Eroze je duální operace k diletaci a výsledek je 1, pokud a pouze pokud po přiložení všechny jedničky strukturního prvku přiléhají jedničkám v obraze. Množinově:

$$I \ominus B = \bigcap_{b \in B} I_{-b} \tag{5.9}$$

Operace nejsou inverzní, eroze i diletace jsou ztrátové operace, erozí jsou odstraněny detaily tvořené jedničkami menší než strukturní prvek a diletací detaily tvořené nulami. Právě proto je jedním možným použitím odstranění šumu. Operace ale mění velikost objektů, pokud jsou velikosti důležité, je potřeba k odstranění šumu zvolit otevření a/nebo uzavření.

5.4.2 Otevření a uzavření

Otevření a uzavření jsou operace složené z diletace a eroze. Otevření je operace tvořená erozí následovanou diletací. Eroze odstraní detaily tvořené jedničkami a následná diletace navrátí obraz do zjednodušené podoby o původní velikosti.

$$I \circ B = (I \ominus B) \oplus B \tag{5.10}$$

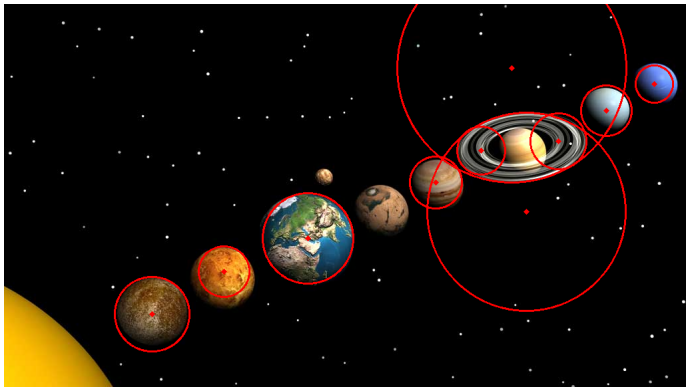
Uzavření je duální operací otevření a zaplňuje otvory tvořené nulami. Operaci tvoří diletace následovaná erozí.

$$I \bullet B = (I \oplus B) \ominus B \tag{5.11}$$

5.5 Houghovy transformace

5.5.1 Transformace na přímky

Obecná Houghova transformace umožňuje detekovat přímky a parametrické křivky (kružnice, elipsy, paraboly...) převodem bodů z prostoru obrazového do parametrického. Obraz je nejdříve binarizován např. prahováním na popředí a pozadí. Každý



Obrázek 5.4: Transformace grafiky solárního systému na kružnice. Příklady vytvořeny v programu zrealizovaném v rámci práce. Úloha je značně zesložitěna stíny.

bod popředí je postupně dosazován do rovnice hledané parametrické křivky a je přidán jeden hlas pro každou kombinaci parametrů, pro kterou křivka tímto bodem prochází.

Prostor parametrů je diskretizovaný a ohraničený dle velikosti obrazu. Z rovnice křivky je vyjádřen jeden parametr, přiřazení hlasů pro každý bod je provedeno přes všechny možné hodnoty všech ostatních parametrů.

Houghova transformace na přímky využívá polární formu rovnice přímky:

$$\rho = x \cos\theta + y \sin\theta \tag{5.12}$$

Rovnice má dva volné parametry, prostor parametrů je tedy dvourozměrný, a po dosazení bodu zůstane jen jeden prostor volný.[1]

5.5.2 Transformace na kružnice

Rovnice kružnice má tři volné parametry:

$$r^2 = (x-a)^2 + (y-b)^2 \tag{5.13}$$

Trojrozměrný parametrový prostor se dvěma volnými parametry pro každý bod je výpočetně náročný, běžně je tedy obecný postup rozšířen o metodu gradientu, která umožňuje v první části zanedbat poloměr a hledat v dvourozměrném prostoru středů kružnic:[1]

1. Šedotónový obraz je binarizován na hrany Cannyho hranovým detektorem.
2. V bodech hran je Sobelovým operátorem zjištěn směr gradientu.
3. Protože tečna kružnice je kolmá na poloměr, a gradient je kolmý na hranu, bodům v a proti směru gradientu je přiřazen hlas.

5.6 Detekce hran

5.6.1 Sobelův operátor

Sobelův operátor se skládá ze dvou matic o rozměrech 3x3[2]:

$$x - component = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{5.14}$$

$$y - component = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \tag{5.15}$$

V obou případech jde o separabilní konvoluční jádro tvořené jednorozměrným Gaussovským vyhlazujícím filtrem a jednorozměrným filtrem první derivace. Tuto myšlenku lze zobecnit na větší filtry a vyšší řády derivace: Knihovna OpenCV implementuje jádra o velikosti až 7x7 a řády derivace až do šestého[3]. Pro názornost, 5x5 jádro x-komponenty čtvrté derivace vypadá následovně:

$$\begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ 4 & -16 & 24 & -16 & 4 \\ 6 & -24 & 36 & -24 & 6 \\ 4 & -16 & 24 & -16 & 4 \\ 1 & -4 & 6 & -4 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \end{bmatrix} \tag{5.16}$$

Konvoluce obrazu se Sobelovým operátorem umožňuje detekovat hrany ve směru derivace, pokud se definuje nějaký konkrétní práh minimální derivace, kterou považujeme za hranu. Hodnoty derivace přitom nabývají kladných i záporných hodnot, a tak pro prahování je nutné nejdříve vzít absoutní hodnotu výsledku.

Častějším použitím Sobelova operátoru je výpočet gradientu a tedy směru hrany.[1]

5.6.2 Laplaceův operátor

Laplaceův operátor aproximuje druhou derivaci konvolučním jádrem o rozměru minimálně 3x3, které může mít několik podob[1]:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}, \tag{5.17}$$

Poslední příklad reprezentuje separabilní jádro tvořené dvěma jednorozměrnými filtry derivace druhého řádu:

$$\begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} = - \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \tag{5.18}$$

5.6.3 Cannyho hranový detektor

Cannyho algoritmus pro detekci hran je založen na kvazi-konvoluci, jejíž konvoluční jádro je závislé na uvažovaném obrazovém bodě (a jeho okolí), a tedy nejde o lineární operaci. Algoritmus vyžaduje konvoluci s derivací filtru gaussovského vyhlazení G ve směru gradientu v bodě, na který je jádro právě aplikováno:

$$\frac{\partial G}{\partial n} * f, \text{ kde } n = \frac{\nabla(G * f)}{|\nabla(G * f)|} \tag{5.19}$$

Tuto nelinearitu je možné odstranit díky asociativitě konvoluce a tuto část algoritmu rozložit na několik konvolucí. Průběh algoritmu potom vypadá takto:[1]

1. Konvoluce obrazu s jádrem gaussovského vyhlazení
2. Konvoluce výsledku se Sobelovým operátorem[4] pro odhad směru gradientu obrazu
3. Porovnání bodů s body v a proti směru gradientu, bod je zvažován pokud je lokálním maximem. [4] Tím je aproximováno zvažování nulových bodů druhé derivace ve směru gradientu.
4. Výpočet skóre hrany v každém bodě jako velikosti gradientu
5. Porovnání bodů s vyšším a nižším prahem. Hrany jsou body, které přesahují nižší práh, a jsou součástí skupiny bodů spojených sousedností, která obsahuje alespoň jeden bod přesahující vyšší práh.

5.7 Detekce příznaků

5.7.1 SIFT: Scale Invariant Feature Transform

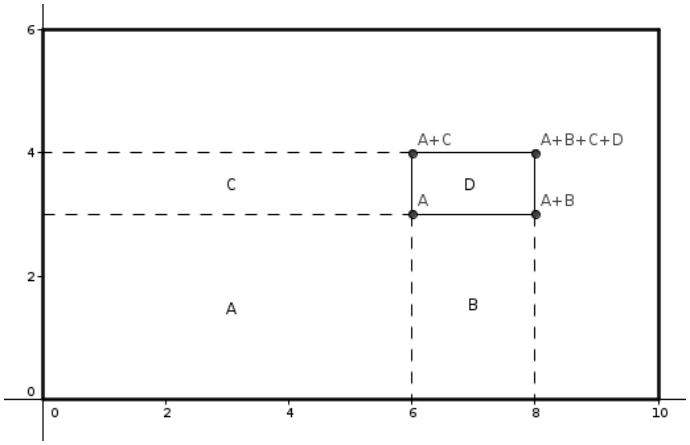
Příznakový detektor SIFT aproximuje LoG operátor (Laplacian of Gaussian) rozdílem dvou Gaussových filtrů o různých rozptylech, tzv. DoG (Difference of Gaussian). Filtr Gaussova vyhlazení je pro optimalizaci možné přibližně separovat na dva jednorozměrné filtry.[1]

Pro obraz je sestaven trojrozměrný prostor, kdy třetí prostor je tvořen skládáním na sebe výstupy konvoluce vstupního obrazu s postupně škálovanými filtry DoG, umožňující hledání postupně větších příznaků. Příznaky jsou lokální maxima v tomto prostoru, ve kterém každý bod je srovnávan s 26 body na $3 \times 3 \times 3$ sousedství.

Pro každý příznak je hledán směr ve vyhlazeném obraze L odpovídajícím škále příznaku. Na okolí bodu, kterému příznak náleží, jsou sesbírány velikosti a úhly gradientů získaných z rozdílů intenzit bodů:

$$d_x = L_{x+1,y} - L_{x-1,y} \tag{5.20}$$

$$d_y = L_{x,y+1} - L_{x,y-1} \tag{5.21}$$



Obrázek 5.5: Integrální obraz. Obsah oblasti D lze spočítat jako (A+B+C+D) - (A+C) - (A+B) + A.

$$\theta(x, y) = \arctan(d_y/d_x) \tag{5.22}$$

$$m(x, y) = \sqrt{d_x^2 + d_y^2} \tag{5.23}$$

Z velikostí vážených Gaussovým vyhlazením kolem bodu příznaku je sestaven histogram dle úhlu o 36 desetistupňových intervalech. Inteval s nejvyšším součtem velikosti je volen jako směr příznaku; Pro další lokální maxima histogramu o velikosti alespoň 80% globálního maxima jsou vytvořeny další příznaky ve stejném bodě.

Příznakový vektor je vytvořen přiložením 32×32 mřížky, dělené na 4×4 oblasti o 8×8 vzorcích, orientované ve směru příznaku. V rámci každé oblasti jsou gradienty vzorků rozděleny do histogramu dle úhlu o 8 intervalech; Toto přiřazení je trilineárně interpolovaserviceservicené mezi blízké intervaly pro zamezení náhlých změn. Těchto 8 hodnot pro každou ze 16 oblastí, po oříznutí a normalizaci, tvoří 128-rozměrný příznakový vektor.[5]

5.7.2 SURF: Speeded-Up Robust Features

Algoritmus SURF jako konvoluční jádra využívá filtry složené ze spojitých obdélníkových oblastí stejné hodnoty (konkrétně 1 a -1). Součet hodnot v obdélníkové oblasti o velikosti MxN lze urychlit využitím integrálního obrazu, ze složitosti O(MN) na složitost O(1). SURF této nezávislosti na veliosti regionu využívá k postupnému zvětšování filtru a tedy hledání příznaků o různých velikostech.

Filtr je zvětšován lineárně v rámci oktávy. Proces zvětšení jádru zvýší počet řádků i sloupců o 6, a v n-té oktávě je vždy přeskočeno $2^{n-1} - 1$ kroků zvětšení. Výstupy konvoluce obrazu s filtry o rostoucí velikosti jsou naskládány na sebe, čímž vznikne trojrozměrný prostor; Příznaky jsou lokální maxima v tomto trojrozměrném prostoru, a pozice těchto maxim nese informaci o pozici příznaku v obrazu a o jeho velikosti.

Následuje výpočet příznakového vektoru pro každý příznak (škálovaný velikostí filtru, které náleží):[6]

1. Vypočtou se odezvy na filtr Haarovy vlnky ve směru x a y na kruhovém sousedství, vážené gaussovským filtrem.

2. Zjistí se největší součet odezev přes pozice otáčivého okénka; Směr vektoru udává směr příznaku. Výpočet deskriptoru je otočen o absolutní směr příznaku v obraze, čímž je zajištěna neměnnost deskriptoru vůči otočení.

3. Filtr Haarovy vlnky ve směru x a y se aplikuje na 20x20 navzorkovaných bodů obrazu tvořících mřížku kolem příznaku, rozdělených do 4x4 regionů o 5x5 bodech. Odezvy jsou váženy gaussovským filtrem.

4. Pro každý region je spočten součet odezev ve směru x d_x, součet odezev ve směru y d_y, $\sum |d_x|$ a $\sum |d_y|$.

5.7.3 FAST: Features from Accelerated Segment Test

Příznakový detektor FAST zařadí každý z 16 obrazových bodů $p_{1,2..16}$ na kružnici kolem zkoumaného bodu p do jedné ze tří kategorií, v závislosti na hodnotě I_{x_n} relativně k hodnotě I_p a předem zvoleného prahu t:

$$s_{p,n} = \begin{cases} d & \text{pokud } I_{p_n} \leq I_p - t \text{ (darker; tmavší)} \\ b & \text{pokud } I_{p_n} \geq I_p + t \text{ (brighter; světlejší)} \\ s & \text{jinak (similar; podobný)} \end{cases} \tag{5.24}$$

Roh je takový bod p, pro který 12 spojitých bodů na kružnici náleží do kategorie světlejší, nebo do kategorie tmavší. Volba 12 spojitých bodů umožňuje zrychlení algoritmu zajištěním, že pro čtyři ekvidistantní body oblouk spojitých bodů musí procházet třemi; Dva takové body a oblouk mezi nimi tvoří segment o velikosti pěti bodů, a na oblouk odpovídající detekci rohu by jich pak zbylo jen jedenáct.

Tento algoritmus je poté využit k detekci rohů pro strojové učení varianty tohoto algoritmu založené na rozhodovacím stromu nad daty ze vzorkovaných šestnáctni bodů. Pro každou relativní pozici n, tedy $n \in 1, 2, ...16$, je přes $q \in Q$ (množina všech bodů) sezbírána hodnota $s_{q,n}$ a binární proměnná K, která obsahuje, zda bod q má být klasifikován jako roh.

Soubor hodnot přes každé n odpovídá rozřazení množiny Q všech obrazových bodů q do tří množin Q_b, Q_s, Q_d, závislému na třech možných hodnotách $s_{q,n}$. Klasifikace proběhne volbou jednoho z těchto šestnácti rozřazení, konkrétně toho, které nese nejvyšší informaci o proměnné K. Informační zisk volby dělení n je vypočten z entropie K pro množinu všech bodů a entropie K pro jednotlivé výstupní množiny tohoto dělení:

$$H(Q) = (c + \bar{c}) \log_2(c + \bar{c}) - c \log_2 c - \bar{c} \log_2 \bar{c}; \text{ c = zlomek rohů v množině Q} \tag{5.25}$$

$$H(Q) - H(Q_b) - H(Q_s) - H(Q_d) \tag{5.26}$$

Postup je rekurzivně opakován nad každou výstupní množinu Q_b, Q_s, Q_d zvoleného dělení, pokud množina obsahuje smíšené hodnoty K. Každé dělení na tři podmnožiny odpovídá uzlu rozhodovacího stromu, ze kterého vycházejí tři větve; Pokud množina obsahuje jen totožné hodnoty K, odpovídá listu rozhodovacího stromu a klasifikaci jako K.[7]

5.7.4 BRIEF: Binary Robust Independent Elementary Features

Algoritmus BRIEF umožňuje vytvoření podstatně kratších příznakových vektorů příznaků detekovaných jiným algoritmem a navrhuje rychlejší funkci srovnání dvou takových vektorů pro urychlení hledání shod mezi dvěma obrazy.

V rámci algoritmů SIFT a SURF, příznakový vektor bodu je tvořen vektorem desetinných čísel, reprezentovaných hodnotou s plovoucí desetinnou čárkou. Příznakový vektor BRIEF ukládá pouze jednobitové výstupy testu prováděného nad různými páry bodů v blízkosti popisovaného bodu:

$$\tau(X,Y) = \begin{cases} 1 & \text{pokud } I_X < I_Y \\ 0 & \text{pokud } I_X \geq I_Y \end{cases} \tag{5.27}$$

Obraz je před provedením testu vyhlazen, aby každý jednotlivý bod nesl informaci o blízkém okolí. Volbu párů na okolí popisovaného bodu lze provést různými způsoby:

- Rovnoměrné rozložení bodů na okolí
- Gaussovo rozložení bodů se středem v popisovaném bodě
- Obdobné rozložení prvních bodů testu, druhé body dle Gaussova rozložení se středem v prvním bodě testu
- Diskrétní rozložení bodů na hrubé polární mřížce
- První bod testu vždy popisovaný bod, druhý bod nabývá všech pozicí na obdobné mřížce

Podobnost dvou příznaků je měřena Hammingovou vzdáleností jejich příznakových vektorů, která odpovídá počtu binárních testů, ve kterých se liší.

Algoritmus BRIEF není invariantní vůči otočení, protože při otočení obrazu každý bit odpovídá testu nad jiným párem bodů originálního obrazu. Do cca 15 stupňů ale algoritmus funguje dostatečně.[8]

5.7.5 STAR, CenSurE

Podobně jako SIFT a SURF, CenSurE (Center Surround Extrema) umožňuje hledání příznaků v trojrozměrném prostoru space-scale, kde odezvy obrazu na postupně zvětšovaný filtr jsou naskládány na sebe.

Principem je aproximace dvojúrovňového prstencového vlnkového filtru, který aproximuje filtr Laplacian-of-Gaussian. Nejjednodušší aproximací je rozdíl čtverců, filtr který zároveň nabízí triviální optimalizaci integrálními obrazy, jako tomu je v případě algoritmu SURF.

V zájmu redukce citlivosti na otočení obrazu je možné filtr aproximovat rozdílem dvou pravidelných n-úhelníků, kde $n > 4$, například šestiúhelníků nebo osmiúhelníků. Obsah osmiúhelníku není možné spočítat integrálním obrazem, CenSurE tedy využívá zkosených integrálních obrazů. Osmiúhelník je rozložen na lichoběžníky, jejichž obsah je spočten z integrálních obrazů zkosených o úhly odpovídajících jejich lichoběžným hranám.[9]

STAR je varianta CenSurE, která místo aproximace n-úhelníkem využívá dvou filtrů rozdílu čtverců, jeden otočený o 45 stupňů.

5.7.6 ORB: Oriented FAST and rotated BRIEF

ORB rozšiřuje algoritmy FAST a BRIEF o orientovanou složku oFAST a rBRIEF.

K detekci otočení algoritmus oFAST využívá diskretizovanou podobu výpočtu prvních momentů okolí příznaku ve směru x a y integrováním:

$$m_{p,q} = \sum_{x,y} x^p y^q I_{x,y} \tag{5.28}$$

Příznak je pro výpočet považován za bod $(0, 0)$. Z prvních momentů je odvozen centroid okolí příznaku, $(m_{1,0}/m_{0,0}, m_{0,1}/m_{0,0})$. Směr příznaku je poté fáze vektoru ze středu příznaku do centroidu, $atan2(m_{0,1}, m_{1,0})$. Interpretace směru jako "dovnitř" nebo "ven z" rohu závisí na relativní intenzitě rohu a okolí, nicméně pro totožný roh napříč různými pohledy je konzistentní.

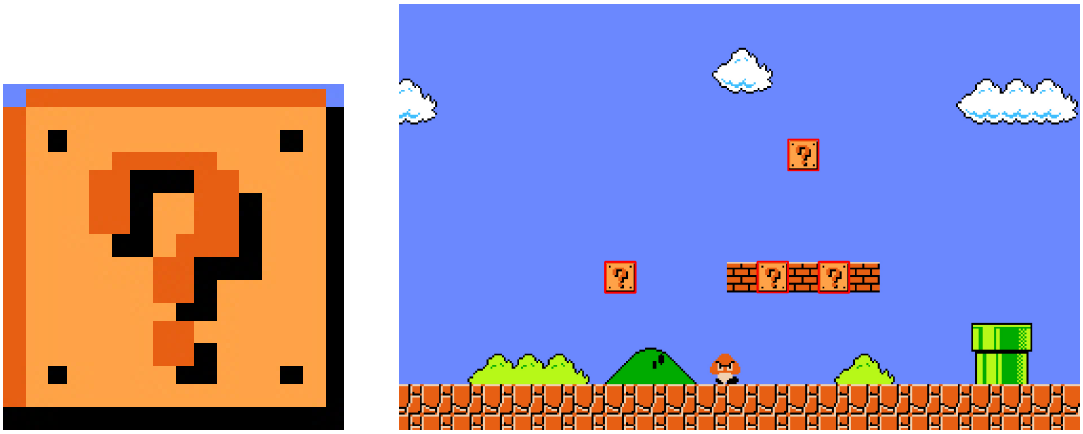
rBRIEF z algoritmu BRIEF vznikne otočením bodů binárních testů kolem vyšetřovaného bodu dle úhlu detekovaného algoritmem oFAST. Formálně je otočená verze testů (X_1, Y_1) až (X_n, Y_n) definována násobením matice otočení a $2 \times 2N$ matice složené z bodů testů.

Prakticky je úhel diskretizován na 12-stupňové kroky a pro každý jsou otočené testy předpočítány. Sada testů pro algoritmus rBRIEF se liší od sady testů pro algoritmus BRIEF; Metoda jejich volby je navržená tak, aby optimalizovala žádané charakteristiky (nesouvztažnost testů, vysoký rozptyl) s ohledem na skutečnost, že jejich význam je relativní vůči směru rohu.[10]

5.8 Hledání objektu v obraze

5.8.1 Vyhledávání vzoru

Vyhledávání vzoru je nejjednodušší případ hledání objektu v obraze a funguje za předpokladu, že se v obraze I jednodušše vyskytuje přibližná kopie šablony T s odpovídajícími relativními pozicemi odpovídajících bodů; Algoritmus je tedy citlivý vůči otočení, zvětšení, perspektivě a posunu rozsahu intenzit.



(a) Vzor (b) Obraz s vyznačeným nalezeným vzorem

Obrázek 5.6: Vyhledání vzoru v obraze. Příklady vytvořeny v programu zrealizovaném v rámci práce.

Pro každý obrazový bod je vypočítán součet kvadratických chyb odpovídajících si bodů za předpokladu, že k němu je "přiložen" levý horní roh hledaného vzoru bez jakékoli transformace:

$$E(x,y) = \sum_{i}^{w} \sum_{j}^{h} (T_{i,j} - I_{i+x,j+y})^2 \tag{5.29}$$

Nejlepší shodu šablony s obrazem určuje pozice globálního minima, hledání více než jedné pozice může být docíleno např. prahováním.

5.8.2 Hledání hrubou silou

Jednoduchou metodou hledání shody mezi příznakovými vektory ve dvou obrazech v prostoru příznaků, například předlohou a obrazem pro vyhledání objektu v obraze, je spočítat vzdálenost každého příznakového vektoru v prvním obraze od každého příznakového vektoru v obraze druhém. Pro každý příznakový vektor v prvním obraze je zvolen nejbližší vektor ve druhém obraze jako shoda.

Algoritmus funguje pro jakoukoli metriku vzdálenosti příznakových vektorů, jak Euklidovskou vzdálenost pro porovnání příznaků SIFT nebo SURF, tak Hammingovu vzdálenost pro příznakové vektory BRIEF a ORB. Zároveň je algoritmus triviálně rozšířen o nalezení druhé nejbližší shody, a porovnáním vzdáleností poměrovým testem je určena kvalita příznaku; Pokud nejbližší shoda není o moc lepší než druhá nejbližší, pravděpodobně není příliš unikátní.

5.8.3 Hledání metodou k-rozměrného stromu

Pro příznaky tvořené vektory reálných čísel a porovnávané Euklidovskou vzdáleností, hledání nejbližšího souseda lze urychlit sestavením k-rozměrného stromu nad vektory v prostoru příznaků nalezených ve druhém obraze iterativním algoritmem:

1. Nechť Q je množina N-rozměrných příznakových vektorů $q = (q_1, q_2, ..., q_N)$. Je zvoleno i tak, aby volba maximalizoval rozptyl q_i:

$$i^* = \arg\max_i \sum_{q \in Q} \|q_i - \mu_i\|^2 \tag{5.30}$$

2. Množina Q je rozdělena rovinou procházející mediánem q_i a kolmou na rozměr i.

3. Nerovnice odpovídající dělení je přiřazena vrcholu stromu, v počátku algoritmu kořenu. Pro dvě podmnožiny Q vzniklé rozdělením, Q_1 a Q_2, je algoritmus rekurzivně opakován, čímž vzniknou dva potomci tohoto vrcholu.

Blízkost ve výsledné struktuře vypovídá o blízkosti v příznakovém prostoru. Binární úplný strom je sestaven do hloubky d a má tedy 2^d listů, každý list odpovídá podprostoru obsahujícímu přibližně stejný počet prvků původní množiny Q (díky použití mediánu pro dělení).

Každý příznak prvního obrazu je stromem zařazen do podmnožiny, nad kterou je provedeno úplné hledání nejbližšího souseda. Následuje hledání ve zbytku stromu; Pokud je celý podprostor některého listu vzdálenější než zatím nalezený nejbližší soused, všechny vektory obsažené v listu je možné přeskočit.[1]

Knihovna FLANN nabízí modifikovanou metodu hledání přibližného nejbližšího souseda (ANN, Approximate Nearest Neighbour) k-rozměrným stromem. Do volby rozměru i pro dělení prostoru je zanesena náhoda, a algoritmus je opakován několikrát, čímž vznikne les stromů odpovídající několika různým dělením. Krok hledání ve zbytku stromu je pak částečně zanedbán

5.8.4 Hledání metodou LSH, lokálně citlivého hašování

V případě příznaků tvořených binárními řetězci a porovnávané Hammingovou vzdáleností, hledání přibližného nejbližšího souseda lze urychlit metodou lokálně citlivého hašování (LSH, locality-sensitive hashing).

Principem algoritmů LSH je rozřazení příznakových vektorů do skupin dle kolizí hašovací funkce, která je navržena tak, aby kolize byla pravděpodobnější pro bližší vektory; Konkrétně funkce h nad metrickým prostorem S je (r_1, r_2, p_1, p_2)-citlivá, pokud pro každé $p, q \in S$:[11]

$$\|p - q\| \leq r_1 \implies Pr(h(p) = h(q)) \geq p_1 \tag{5.31}$$

$$\|p - q\| \geq r_2 \implies Pr(h(p) = h(q)) \leq p_2 \tag{5.32}$$

Pro míru vzdálenosti je funkce užitečná, pokud $p_1 > p_2 \wedge r_1 < r_2$. Rodina takových funkcí H pro Hammingovu vzdálenost nad prostorem $S = \{1, 0\}^d$ jsou funkce $h_i(p) : \{1, 0\}^d \to \{1, 0\} = p_i; i \in \{1, 2, ..., d\}$, tedy funkce vzorkování jednoho bitu. Hammingova vzdálenost dvou vektorů nad takovým prostorem je korelovaná s pravděbodobností, že se budou v kterémkoli bitu lišit.

Obrázek 5.7: Nalezení objektu v obraze. Příklady vytvořeny v programu zrealizovaném v rámci práce. Pro tento příklad byl použit algoritmus SIFT, k-rozměrný strom, a metoda RANSAC.

Z rodiny H je vytvořena rodina G, kde každá funkce g je tvořena r různými funkcemi h_i. Výstupem funkce $g_{i_1,i_2,\ldots,i_r}$ je r-rozměrný bitový řetězec odpovídající vzorkování r bitů. Protože h_i jsou nezávislé funkce, rodina G je $(r_1, r_2, (p_1)^r, (p_2)^r)$-citlivá.[12]

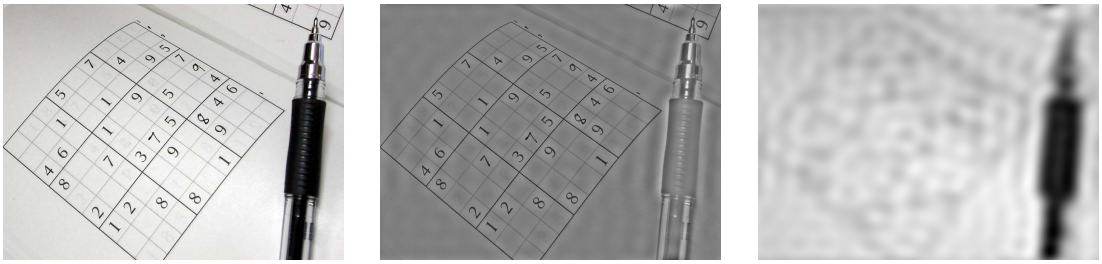
Ve výsledku je tedy hledání přibližného nejbližšího souseda Hammingovou vzdáleností urychleno předvýpočtem Hammingovy vzdálenosti na náhodném menším vzorku bitů.

5.8.5 Homografie metodou RANSAC nebo LMedS

Homografie je transformace o čtyřech volných parametrech určená transformací čtverce na libovolný čtyřúhelník. Umožňuje zachytit posun, otočení, zkosení a perspektivu. Ze známé shody příznakových bodů mezi dvěma obrazy by transformace byla možná spočítat regresí, ale existují metody, které lépe zvládají odlehlé hodnoty vzniklé nesprávnou shodou příznaků.

Metody RANSAC (Random Sample Consensus) a LMedS (Least Median of Squares) opakovaně náhodně volí čtyři shody, kterými ukotví homografii. Očekává se, že když jsou zvoleny správně spárované body, projekce velkého množtví bodů takto ukotvenou transformací dopadne v blízkosti skutečně nalezeného odpovídajícího příznaku. Měří se tedy kritérium reprojection error, tj. vzdálenost mezi projekcí a naměřenou pozicí.

Metoda RANSAC zvolí tu projekci, pro kterou se kritérium reprojection error nachází pod předem určeným prahem pro nejvyšší počet příznaků. Metoda LMedS zvolí projekci s nejnižším mediánem kritéria reprojection error.



(a) Originál (b) Horní propust (c) Dolní propust

Obrázek 5.8: Porovnání informace zachované horní a dolní propustí. Příklady vytvořeny v programu zrealizovaném v rámci práce. Dolní propust zachovává oblasti, horní propust hrany.

5.9 Operace nad frekvenční doménou

5.9.1 Obrazové filtry

Horní propust, dolní propust a pásmová propust jsou filtry umožňující potlačit informaci o oblastech nebo o hranách. Obraz I o rozměrech $M \times N$ je převeden do frekvenční domény Diskrétní Fourierovou transformací:

$$F_{u,v} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I_{m,n} e^{-2\pi i \left(\frac{mu}{M} + \frac{nv}{N}\right)} \tag{5.33}$$

V praxi je DFT spočtena některým algoritmem FFT (Fast Fourier Transform), například Cooley-Tukey algoritmem, o složitosti $O(n \log n)$. Následuje prohození kvadrantů, tedy přesunutí informace o nízkých frekvencích doprostřed frekvenční domény; Výšku frekvence je teď možné definovat jako Euklidovskou vzdálenost od středu frekvenční domény (jiné metriky připadají v úvahu). Filtr propusti poté funguje následovně:[1]

$$d(u,v) = \sqrt{\left(u - \frac{M}{2}\right)^2 + \left(v - \frac{N}{2}\right)^2} \tag{5.34}$$

$$G_{u,v} = \begin{cases} F_{u,v} & \text{pokud } d(u,v) \geq t_{min} \wedge d(u,v) < t_{max} \\ 0 & \text{pokud } d(u,v) < t_{min} \vee d(u,v) \geq t_{max} \end{cases} \tag{5.35}$$

Pro pásmovou propust jsou oba prahy t_{min} a t_{max} nastaveny na kladná celá čísla, a $t_{min} < t_{max}$. Pro horní propust platí $t_{min} = 0$ a pro horní propust platí $t_{max} = \infty$

5.9.2 Homomorfní filtr

Homomorfní filtr umožňuje zvýraznit vyšší frekvence a potlačit nižší, ale přitom všechny zachovat, a s plynulým přechodem mezi potlačovanými a zvýrazňovanými frekvencemi. Operace umožňuje zvýraznit detail a reflektivitu objektů oproti globálním světelným podmínkám.

Volen byl filtr tvořen funkcí hladkého kroku, jejíž definiční obor a obor hodnot byly lineárně interpolované podle vzdálenosti mezi zvolenými prahy a podle konstant potlačení a zdůraznění:

$$s(x) = 3x^2 - 2x^3 \tag{5.36}$$

$$G_{u,v} = \begin{cases} c_1 F_{u,v} & \text{pokud } d(u,v) < t_1 \\ c_2 F_{u,v} & \text{pokud } d(u,v) \geq t_2 \\ \left(c_1 + (c_2 - c_1) \cdot s\left(\frac{d(u,v)-t_1}{(t_2-t_1)}\right)\right) F_{u,v} & \text{jinak} \end{cases} \tag{5.37}$$

5.10 Detekce a rozpoznání obličeje

5.10.1 Detekce 68 význačných bodů obličeje

Model nalezení 68 význačných bodů obličeje se slkádá z řetězu regresorů, do kterého na počátku vchází obraz obličeje a výchozí odhad bodů. Každý regresor potom postupně odhady přibližuje správným význačným bodům.

Trénovací data jsou tvořena ze souboru dat v podobě (I_1, S_1) až (I_n, S_n), kde I_j je obraz obličeje a S_j vektor o délce dvakrát hledanému počtu význačných bodů, který obsahuje jejich x a y souřadnice. Použitá trénovací data mají podobu $(I_{\pi_i}, S_i^{(0)}, \Delta S_i^{(0)})$, kde:

$$I_{\pi_i} \in \{1, 2...n\} \tag{5.38}$$

$$S_i^{(0)} \in \{S_1, S_2...S_n\} \backslash S_{\pi_i} \tag{5.39}$$

$$\Delta S_i^{(0)} = S_{\pi_i} - S_i^{(0)} \tag{5.40}$$

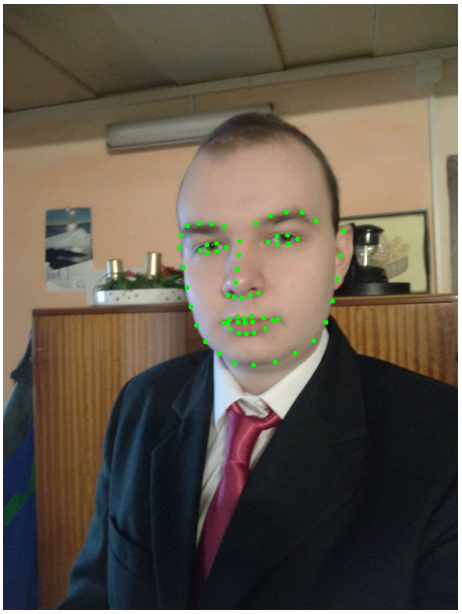
Každá trojice tedy obsahuje obraz obličeje, nesprávný počáteční odhad tvaru obličeje, a vektor rozdílu oproti správnému tvaru obličeje, který se regresor snaží naučit a tím posunout počáteční odhad směrem ke správnému tvaru. Na těchto datech je trénovaný regresor r_0, a každý další regresor je trénován na datech zpracovaných předchozím:

$$S_i^{(t+1)} = S_i^{(t)} + r_t(S_i^{(t)}) \tag{5.41}$$

$$\Delta S_i^{(t+1)} = S_{\pi_i} - S_i^{(t+1)} \tag{5.42}$$

Regresory jsou trénovány posilováním gradientu postupným větvením regresního stromu, ve kterém listy opdovídají konstantám a ostatní vrcholy nerovnicím. Nerovnice jsou voleny volbou parametru θ tak, aby minimalizovaly chybu definovanou následovně:

$$E(Q, \theta) = \sum_{s \in \{l,r\}} \sum_{i \in Q_{\theta,s}} \|r_i - \mu_{\theta,s}\|^2 \tag{5.43}$$



Obrázek 5.9: Detekce 68 význačných bodů obličeje. Příklady vytvořeny v programu zrealizovaném v rámci práce.

kde Q je množina indexů příkladů na vstupu uvažovaného vrcholu, $Q_{\theta,s}$ podmnožina pro kterou volba parametru θ určí pokračování stromem ve směru s, r je vektor odchylek aktuálního regresního modelu od skutečných hodnot, a $\mu_{\theta,s}$ je průměr těchto odchylek pro podmnožinu $Q_{\theta,s}$. Parametrem θ se zde rozumí práh a dva relativní obrazové body, nad kterými je nerovnice spočtena.[13]

5.10.2 Rozpoznání obličeje vnořením a k-NN

Z obrazu obličeje I a vektoru tvaru S zjištěného předchozím algoritmem je spočten 128-rozměrný příznakový vektor, a stejným způsobem jsou předpočteny příznakové vektory pro každý vzorek každé třídy (osoby), čímž je umožněno porovnávat vzdálenosti mezi obličeji v takto vytvořeném 128-rozměrném prostoru.

Vzorky jsou seřazeny dle vzdálenosti jejich příznakových vektorů od vektoru zkoumaného obličeje a zvoleno je k nejbližších vzorků (kde k je konstantní, nebo určeno maximální vzdáleností od vektoru zkoumaného obličeje). Přes zvolené vzorky je sesbírána třída, kterou reprezentují; Obličej je rozpoznán jako nejzastoupenější třída.

6 Popis funkčnosti

Tato kapitola je věnována názornému přiblížení funkcí programu a jejich použití koncovým uživatelem na podrobně rozebraném příkladu. Místy jsou zmíněny funkce, které nejsou použity, ale ke kterým má uživatel v daný okamžik přístup.

6.1 Příklad použití

Zvoleným příkladem je nalezení transformovaného objektu v obraze příznakovým detektorem SIFT a hledáním shody metodou k-rozměrného stromu.

6.1.1 Načtení hledaného objektu

Ihned po otevření aplikace je možné načíst obraz ke zpracování. Alternativně lze obraz a mnohé další předat z příkazové řádky při spuštění obrazu. Obraz lze načíst z menu File, viz 6.1. V menu jsou zároveň vidět klávesové zkratky, například ctrl+O právě pro otevření obrazu.

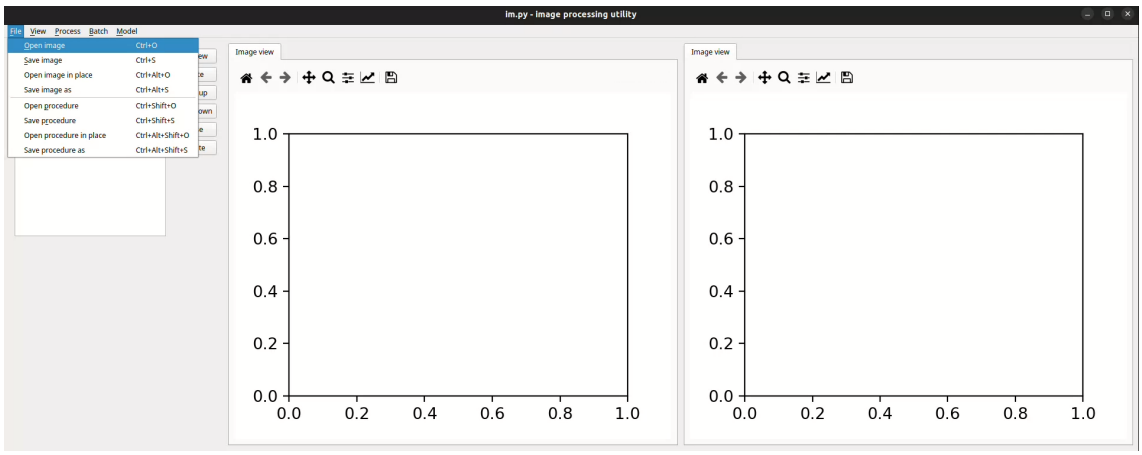
Položka menu nebo klávesová zkratka zobrazí typický dialog volby souboru. Volbu následuje dialogové okno možností načtení obrazu, viz 6.2. V okně je možné zvolit komponentu, která bude z obrazu vytažena. Zaškrtávací políčko Load colorful copy obraz načte jako barevný a vygeneruje operaci vytažení zvolené komponenty. Dále je zde možné vytvořit ukázkovou posloupnost. Pro tento příklad je objekt jednoduše načten jako komponenta jasu.

Pro obraz v šedotónovém prostoru lze pohlédnout na histogram nebo spektrogram přepnutím záložky v horní části okna.

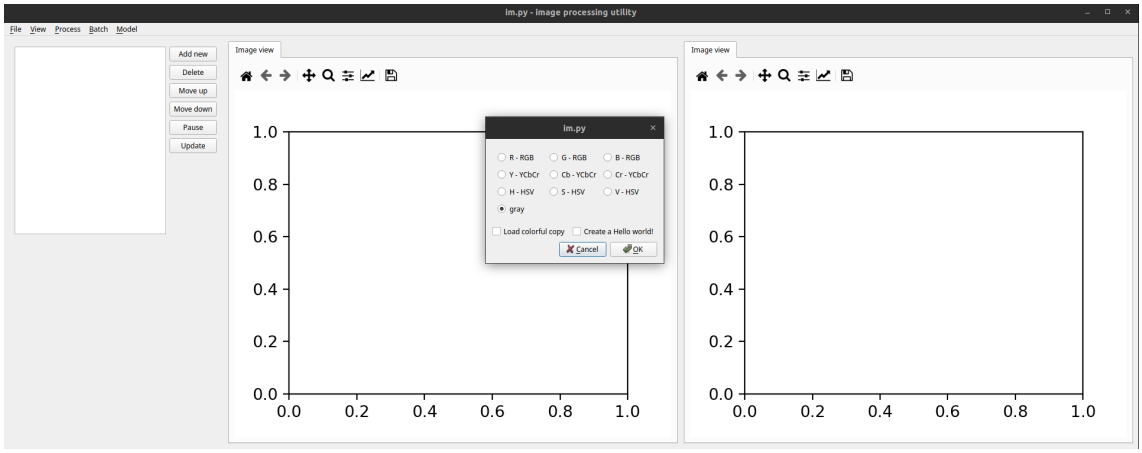
6.1.2 Zpracování hledaného objektu

Novou operaci nad obrazem lze do posloupnosti zavést různými způsoby: Tlačítkem Add new, pravým myšítkem, z menu process, nebo zkratkou ctrl+N. Kterákoli cesta zobrazí totožné dialogové okno, viz 6.3. Textové vyhledávací pole filtruje zobrazené algoritmy v reálném čase. Pro tento příklad je načten příznakový detektor SIFT.

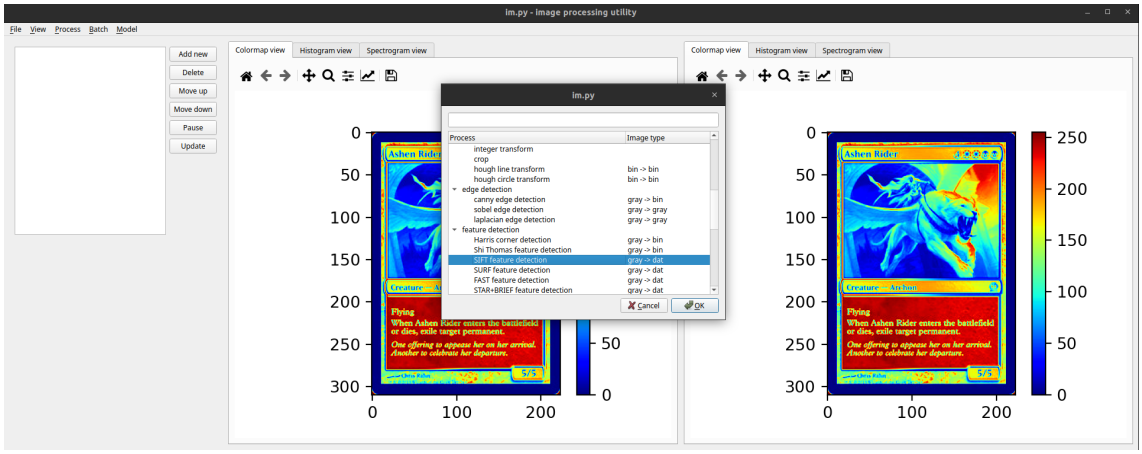
Výstupem algoritmu SIFT jsou příznakové vektory, které nesou informaci užitečnou pro zpracování dalšími algoritmy. Protože tento výstup bude použit jako parametrizovaný hledaný objekt, je prozatím uložen; Uložení výstupního obrazu rozpozná, že se jedná o data, a nabídne uložení jako soubor JSON.



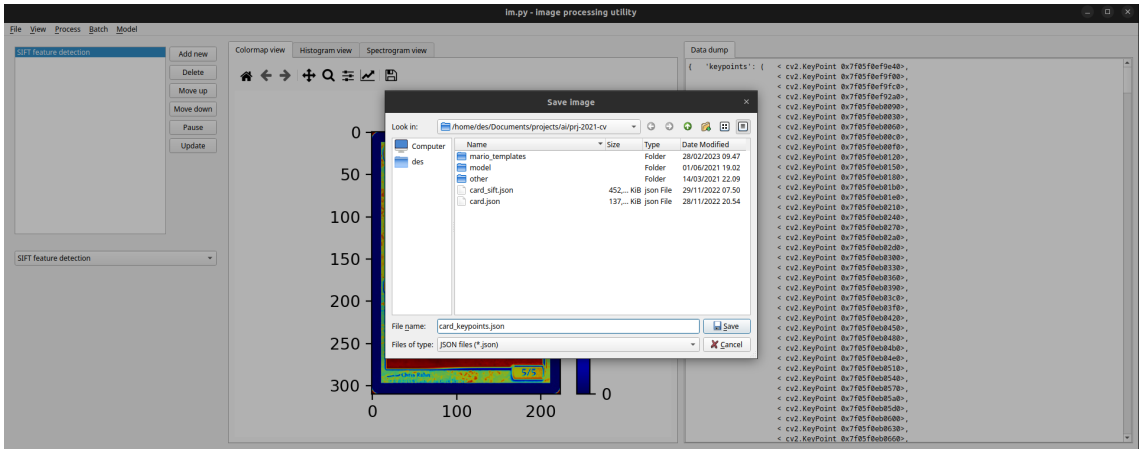
Obrázek 6.1: Po otevření bez parametrů příkazové řádky je pohled prázdný. Obraz lze otevřít například z menu File.



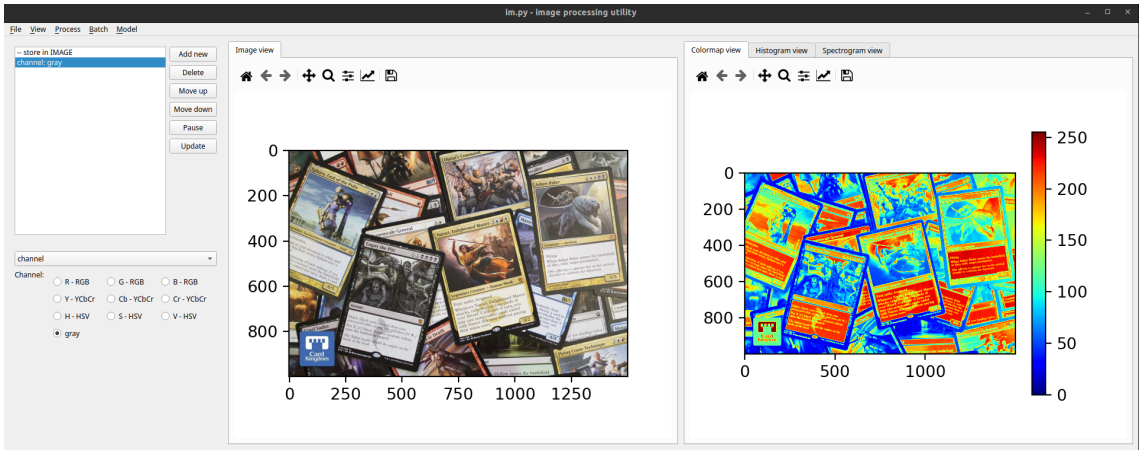
Obrázek 6.2: Po otevření obrazu je možné načíst jako šedotónový obraz.



Obrázek 6.3: Načtený obraz na vstupu a výstupu, a okno přidání operace do sekvence.



Obrázek 6.4: Na výstupu algoritmu SIFT jsou data k dalšímu zpracování. Dialog uložení obrazu proto nabízí uložení ve formátu JSON.



Obrázek 6.5: Načtení obrazu jako barevný s funkcí vytažení komponenty.

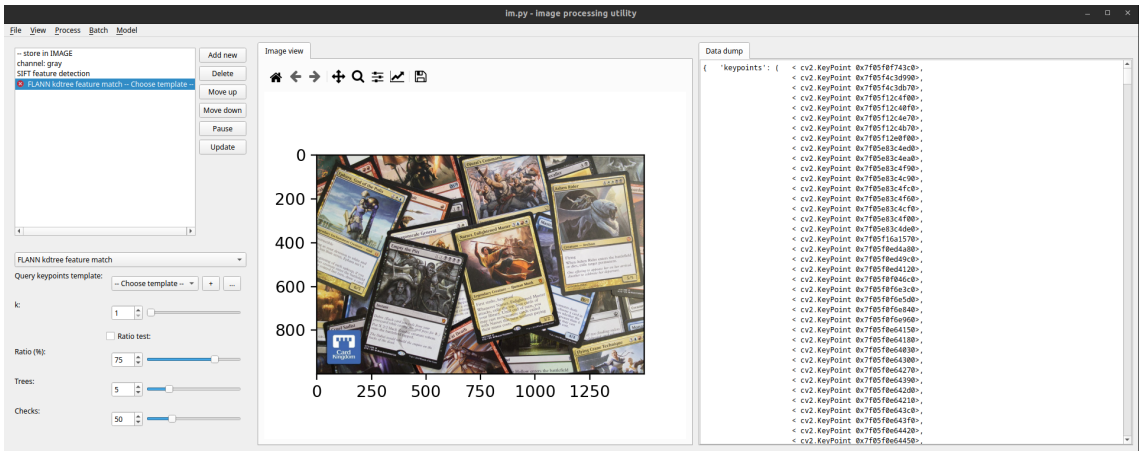
6.1.3 Vyhledání objektu

Tentokrát je volbou v dialogu po načtení obrazu zvolena možnost načíst obraz jako barevný. Tím je vytvořen pomocný proces vytažení komponenty, ale i pomocná funkce uložení stavu obrazu do automaticky vytvořené obrazové proměnné IMAGE. Vytvořené obrazové proměnné a načtené vzory lze spravovat z menu Model.

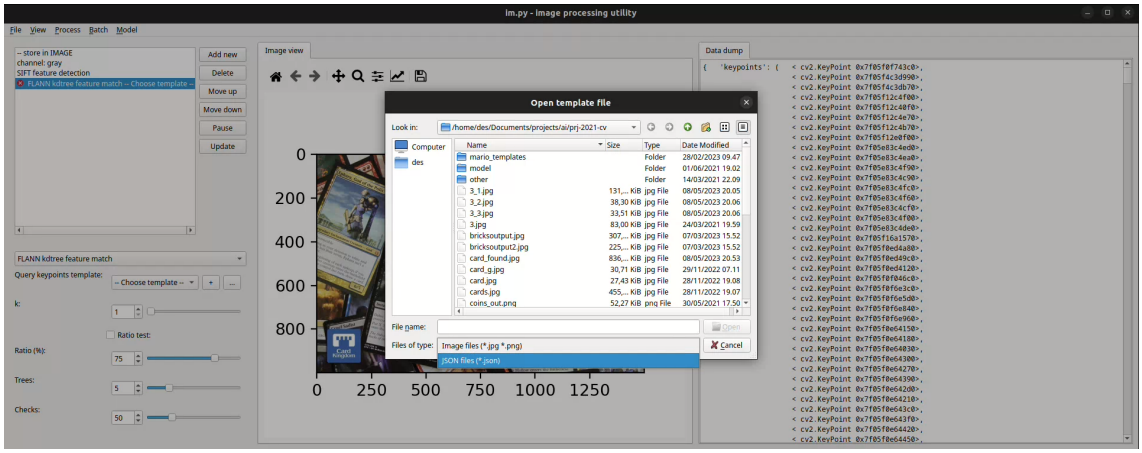
Totožným postupem jako v případě zpracování jsou do posloupnosti operací přidány algoritmy SIFT a hledání shody metodou k-d tree, pro kterou je uživateli poskytnuto množství parametrů. Na změnu parametrů program reaguje v reálném čase, ale hledání shody je pomalý proces. Algoritmus k-rozměrného stromu hlásí chybu, viz 6.6, protože v tuto chvíli není načten a vybrán hledaný vzor. Vzor lze načíst tlačítkem + vedle ovládacího prvku volby vzoru, nebo z menu Model.

Dialogové okno volby vzorového souboru umožňuje přepnout na soubory typu JSON, viz 6.7. Po zvolení vzoru následuje okno, viz 6.8, kde je vzor pojmenován. Pod tímto názvem je poté v ovládacím prvku zvolen.

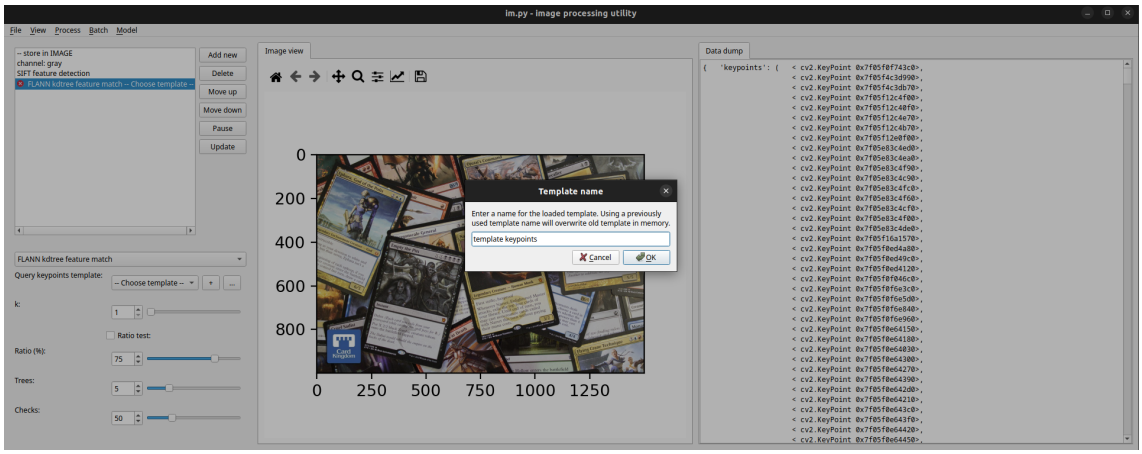
Protože shoda je stále v datovém formátu, je nutné ji vykreslit pomocnou funkcí



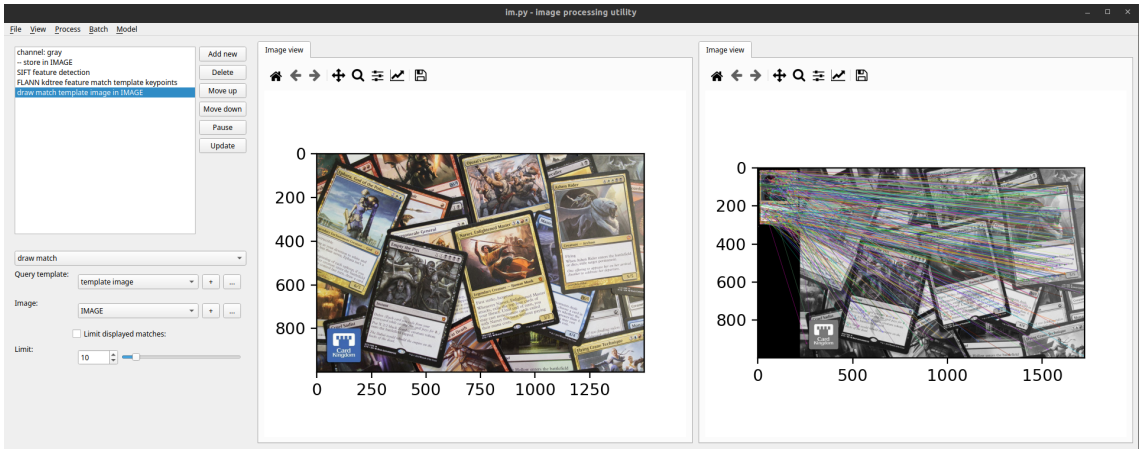
Obrázek 6.6: Do posloupnosti byly zavedeny algoritmy SIFT a k-d tree match. U funkce k-d tree match je vidět chyba; Zatím není s čím hledat shodu.



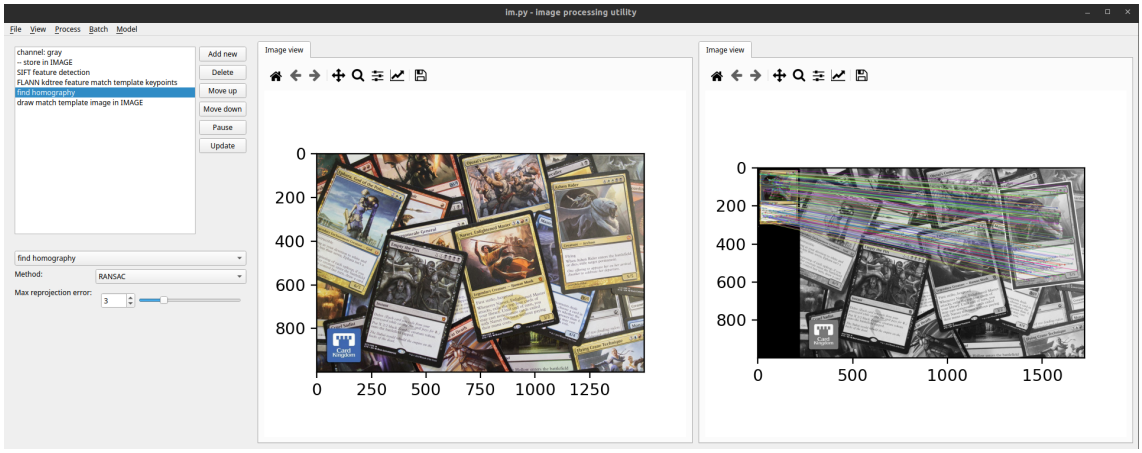
Obrázek 6.7: Načtení vzoru v podobě příznaků.



Obrázek 6.8: Dialog pojmenování načteného vzoru.



Obrázek 6.9: Zavedení a vyplnění pomocné funkce vykreslení shody.



Obrázek 6.10: Nalezení homografie algoritmem RANSAC zachová pouze shody odpovídající nalezené transformaci

vykreslení shody. Funkce požaduje odkaz na originální obraz v prostoru stupňů šedi, který je obdržen posunutím dříve automaticky vytvořené pomocné funkce uložení stavu obrazu pod vytažení komponenty. Obrazová reprezentace vzoru je načtena jako další vzor.

Výsledek, viz 6.9, osahuje množství nesprávně přiřazených příznaků. Do programu je tedy vložen krok nalezení homografie, ve kterém je nalezena transformace odpovídající zobrazení objektu v obrazu, a následně jsou odstraněny příznaky, které transformaci nevyhovují, viz 6.10.

7 Závěr

V rámci práce byl zhotoven program s grafickým rozhraním, který umožňuje použít a vizualizovat algoritmus nebo posloupnost algoritmů zpracování obrazu. Algoritmy byly voleny z knihoven OpenCV, FLANN a dlib. Program byl dále rozšířen o množství pomocných funkcí, které obstarávají požadavky složitějších algoritmů a případů použití programu.

V textu této práce byly představeny a demonstrovány jednotlivé algoritmy, které výsledný program zprostředkovává, přiblížena práce s programem a jeho funkce, vyčteny podrobnosti vnitřní struktury programu podstatné pro možné budoucí rozšíření, a část byla věnovaná postupu a metodice použité ve fázi návrhu.

Program v aktuálním stavu nabízí možná uplatnění: Budování citu pro vliv parametrů v rámci zájmu nebo výuky, rychlé prototypování předzpracování a následná dávková aplikace na soubor dat nebo úprava obrazu např. vyhlazením šumu nebo jasovou korekcí.

Systém pokrývá množství užitečných algoritmů především z knihovny OpenCV a struktura systému byla uskutečněna otevřeně vůči budoucímu zavedení dalších z více než 2500 algoritmů poskytovaných touto knihovnou. Pozornost byla věnována i návrhovému vzoru oddělení grafického rozhraní od logiky programu, čímž připadají v úvahu i úplně nové funkce.

Seznam obrázků

Použitá literatura

[1] SONKA, Milan, Vaclav HLAVAC a Roger BOYLE. *Image processing, analysis and and machine vision (4. ed.)*. 2014. ISBN 978-1-133-59360-7.

[2] SOBEL, Irwin. An Isotropic 3x3 Image Gradient Operator. *Presentation at Stanford A.I. Project 1968*. 2014.

[3] *Image filtering*. [B.r.]. Dostupné také z: https://docs.opencv.org/3.4/d4/d86/group___imgproc___filter.html#gacea54f142e81b6758cb6f375ce782c8d.

[4] *Canny edge detection*. [B.r.]. Dostupné také z: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.

[5] LOWE, David. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*. 2004, roč. 60, s. 91–. Dostupné z DOI: 10.1023/B:VISI.0000029664.99615.94.

[6] BAY, Herbert, Tinne TUYTELAARS a Luc VAN GOOL. SURF: Speeded up robust features. In: 2006, sv. 3951, s. 404–417. ISBN 978-3-540-33832-1. Dostupné z DOI: 10.1007/11744023_32.

[7] ROSTEN, Edward a Tom DRUMMOND. Machine Learning for High-Speed Corner Detection. In: 2006, sv. 3951. ISBN 978-3-540-33832-1. Dostupné z DOI: 10.1007/11744023_34.

[8] CALONDER, Michael et al. BRIEF: Binary Robust Independent Elementary Features. In: 2010, sv. 6314, s. 778–792. ISBN 978-3-642-15560-4. Dostupné z DOI: 10.1007/978-3-642-15561-1_56.

[9] AGRAWAL, Motilal, Kurt KONOLIGE a Morten BLAS. CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. In: 2008, sv. 5305, s. 102–115. ISBN 978-3-540-88692-1. Dostupné z DOI: 10.1007/978-3-540-88693-8_8.

[10] RUBLEE, Ethan et al. ORB: an efficient alternative to SIFT or SURF. In: 2011, s. 2564–2571. Dostupné z DOI: 10.1109/ICCV.2011.6126544.

[11] INDYK, Piotr a Rajeev MOTWANI. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*. 2000, roč. 604-613. Dostupné z DOI: 10.1145/276698.276876.

[12] ANAND, Rajaraman a Ullman JEFFREY DAVID. *Mining of massive datasets*. Cambridge University Press, 2011.

[13] KAZEMI, Vahid a Josephine SULLIVAN. One Millisecond Face Alignment with an Ensemble of Regression Trees. In: 2014. Dostupné z DOI: 10.13140/2.1.1212.2243.