UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2024 Aleš Moc

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Rozpoznávání cen pohonných hmot mobilní aplikací

Bakalářská práce

2024 Aleš Moc

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Aleš Moc**
Osobní číslo:	**I20130**
Studijní program:	**B0688A140009 Informační technologie**
Téma práce:	**Rozpoznávání cen pohonných hmot mobilní aplikací**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je navrhnout a implementovat metody, které používají techniky z oblasti počítačového vidění k rozpoznání cen pohonných hmot na tabulích čerpacích stanic. Pro tento účel bude vytvořena mobilní aplikace, která umožní vyfocené ceny pohonných hmot odeslat na vybraný informační portál. Rozhraní aplikace bude navrženo tak, aby bylo možné ceny před odesláním ručně modifikovat ve formuláři (pro případ neúspěchu správného rozpoznání). Vzhledem k tomu, že různé čerpací stanice budou vyžadovat specifické metody úprav obrazu, bude aplikace navržena tak, aby bylo možné nové metody přidávat. Cílem práce je vytvořit alespoň jednu metodu pro vybranou značku čerpacích stanic, nejlépe pro tu, která nebývá na portálech s cenami zastoupena. Teoretická část práce bude kromě samotného řešení obsahovat popis běžně používaných metod preprocesingu (prahování, morfologie, konvoluční filtry, hranové detektory, Houghova transformace apod.) a zhodnocení úspěšnosti rozpoznávání navržené metody za různých světelných podmínek.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

DOBEŠ, Michal. *Zpracování obrazu a algoritmy v C#*. 1. vyd. Praha: BEN — technická literatura, 2008. 144 s. ISBN 978-80-7300-233-6

KRÁL, Stanislav. *Mobilní aplikace pro rozpoznávání registračních značek vozidel*. Plzeň, 2020. Bakalářská práce (Bc.). Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, 7-5-2020.

Vedoucí bakalářské práce: **Ing. Radek Matoušek**
Katedra informačních technologií

Datum zadání bakalářské práce: **16. prosince 2022**
Termín odevzdání bakalářské práce: **12. května 2023**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2023

Prohlašuji:

Práci s názvem Rozpoznávání cen pohonných hmot mobilní aplikací jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že 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, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 10. 5. 2024 Aleš Moc

PODĚKOVÁNÍ

Tímto bych chtěl poděkovat Ing. Radku Matouškovi za odborné vedení práce a cenné rady k jejímu vypracování.

ANOTACE

Bakalářská práce se zaměřuje na návrh a implementaci postupu pro zpracování digitální fotografie stojanu čerpací stanice použitím technik počítačového vidění. Cílem metody je rozpoznání cen pohonných hmot. Metodu je možné využívat skrze mobilní aplikaci pro operační systém Android. Ta pomocí API komunikuje se serverovou částí, která obstarává komunikaci s databází čerpacích stanic, zpracování zaslané fotografie a následné odeslání informací o cenách pohonných hmot na informační portál.

KLÍČOVÁ SLOVA

Zpracování obrazu, počítačové vidění, Android, Kotlin, C#, API

TITLE

Recognition of gas prices using a mobile application

ANNOTATION

Bachelors thesis focuses on designing and implementing algorithm for processing digital photograph of gas price sign by using techniques of computer vision. The objective of the method is recognising the prices of gas. Method can be used through mobile application for Android operating system. The app uses API for communicating with server side, which handles communication with database of gas stations, processing of incoming photograph and sending information about gas prices to a informational portal.

KEYWORDS

Image processing, computer vision, Android, Kotlin, C#, API

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK

Zkratka	Význam
AI	Artificial Intelligence
API	Application Programming Interface
CV	Computer Vision
GUI	Graphical User Interface
HDR	High Dynamic Range
IDE	Integrated Development Enviroment
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
MVC	Model-View-Controller
OCR	Optical Character Recognition
OS	Operating System
RDBMS	Relational Database Management System
SQL	Structured Query Language

ÚVOD

Tato práce se zabývá problematikou implementace metody pro rozpoznání cen pohonných hmot z fotografie stojanu čerpací stanice, použitím mobilní aplikace.

Cílem této práce je návrh a implementace takové mobilní aplikace pro operační systém Android. Aplikace je řešena ve dvou úrovních. Serverová obslužná aplikace vytvořena v jazyce C# obstarává komunikaci s MySQL databází čerpacích stanic, metodu zpracování obrazu pro detekci cen pohonných hmot z fotografie a také odesílání získaných hodnot na informační portál.

Mobilní aplikace vytvořená v jazyce Kotlin obstarává využívání funkcí serverové aplikace skrze komunikaci pomocí API.

K implementaci metody rozpoznání cen pohonných hmot je využíváno technik počítačového vidění, čímž se zabývá teoretická část práce. Po krátkém úvodu o historii tabulí čerpacích stanic je čtenář seznámen s termínem „počítačové vidění“ a jeho využití v praxi. Další část pojednává o digitální fotografii a způsobech jejího ukládání a poté pokračuje barevnými modely. Dále následují techniky jasové transformace, filtrace obrazu, morfologické operace a segmentace obrazu. Závěr pojednává o optickém rozpoznání znaků.

1 TEORETICKÁ ČÁST

1.1 Historie tabulí cen čerpacích stanic

Na začátku dvacátého století zažil automobilismus až nečekaný boom. Tehdy neexistovaly stanice, jak je známe dnes. V roce 1904 Český klub motocyklistů v království českém odhlasoval návrh na vznik stanic klubu. Byly tak vytipovány místa pro skladování větších objemů benzínu, obvykle v okresních městech. Prodejny ale nebyly specializované, demižony a později sudy prodávaly hotely, hokynářství, nebo i lékárny. V té době se benzín ještě neprodával po litrech, ale po kilogramech.

V průběhu dvacátých let se začala objevovat čerpadla, která usnadňovala doplnění pohonných hmot. Nejprve mobilní, na které se dal připojit zakoupený sud a později i stacionární, s vestavěnou větší nádrží. Ty byly většinou provozovány přímo distribučními společnostmi rafinerií. Jak postupem času začal bujet konkurenční boj, přišla logicky i reklama. Kromě inzercí a plakátů se objevují i reklamní figury u silnice. Dají se považovat za předchůdce nynějších totemů, tehdy ale ještě nezobrazovaly cenu pohonných hmot. Ta se tehdy zobrazovala jen přímo na čerpacím stojanu [1].

Později se začaly objevovat totemy se segmentovými displeji zobrazujícími ceny. Byly ale mechanické, tedy s bílými číslicemi na černém pozadí, které nebyly nijak podsvícené a nepříliš tak tedy vystupovaly z pozadí. S rostoucí popularitou světelné reklamy, která potenciálního zákazníka snadno zaujme, byly mechanické displeje nahrazeny těmi využívající světelné diody. A tyto čísla zářící jasnými barvami tak nejsou snadněji rozpoznatelné jen lidským okem, ale také počítačovým programem.

1.2 Počítačové vidění

Jelikož je pro člověka zrak hlavním nástrojem pro poznávání okolního světa, je pro něj rozpoznání objektu triviální záležitostí, nad kterou se ani nemusí pozastavit. Pro počítače byl ale tento úkol dlouhou dobu takřka nemožný. Díky nárůstu výkonu počítačů a pokrokům v oblasti strojového učení je už dnes pro mnoho lidí samozřejmostí, že stiskem několika tlačítek na svém mobilním telefonu mohou pořídit fotografii a během několika okamžiků poměrně spolehlivě získat žádané informace [2].

Počítačové vidění není jen zajímavým doplňkem mobilního telefonu, ale má velmi široké praktické využití a například v některých z následujících odvětví hraje velkou roli:

- **Výrobní průmysl** – kamera umístěná na výrobní lince může sledovat kvalitu produktů a také je klíčová pro automatizaci s použitím robotů, jejichž pohyby musí být precizní
- **Zdravotnictví** – analýza lékařských snímků může zachytit určité vzory naznačující velmi ranné stádium nemoci, asistence při operacích vyžadující maximální přesnost
- **Zemědělství** – výškové snímky pořízené drony mohou být zpracovány pro monitorování škůdců, zdraví plodin a predikci sklizně
- **Bezpečnost** – technologie rozpoznání obličejů může být využita pro monitorování míst s vysokou mírou ohrožení, jako třeba významná veřejná prostranství, letiště, nebo třeba banky

- **Automobilový průmysl** – sadou kamer umístěných zvenčí celého vozidla lze využívat asistenty pro držení v jízdním pruhu a automatické regulaci rychlosti, představa kompletně autonomního automobilu se posledních pár let blíží skutečnosti
- **Digitalizace dokumentů** – fyzické listiny není třeba ručně přepisovat do digitální podoby, dnes k tomu stačí pouhý snímek dokumentu
- **Biometrie** – autentizace pomocí otisku prstu nebo skenu obličeje je tak rozšířená, že je dnes základní výbavou mobilních zařízení

1.3 Digitální fotografie

Obraz, jak ho vnímá člověk je světelný signál, který zachytí sítnice v oku a optickým nervem odešle mozku, který signály zpracuje. Výsledek ale existuje pouze v našich hlavách, a ne každý signál interpretuje stejně. Nabízí se tedy otázka, jak tento signál digitalizovat, tak, aby ho dokázal interpretovat jak člověk, tak stroj.

Digitální kamera disponuje senzorem složeným z pixelů, který stejně jako lidské oko převede světelný signál na hodnotu, která reprezentuje barvu. Složením všech pixelů do matice tak vznikne obraz – digitální fotografie. A matice hodnot už je formát, se kterým si počítač dokáže poradit.

1.3.1 Formáty ukládání

Matici hodnot je poté potřeba nějakým způsobem uložit. Formátů je mnoho, většinou byly vytvořeny s vybranou hlavní předností, se kterou ale přichází s ní spojené nevýhody. Dělí se na rastrové a vektorové. Rastrové jsou maticí pixelů, což je ideální pro zpracování počítačem. Vektorové jsou definicemi geometrických tvarů jako body, přímky, křivky a polygony. Využívají se převážně v situacích vyžadující vysokou geometrickou přesnost, například v softwarech pro inženýrské a architektonické nákresy [3].

JPEG

Zkratka pro Joint Photographic Experts Group, tedy organizaci stojící za vznikem tohoto formátu v roce 1992. Hlavním cílem formátu je udržení rovnováhy mezi kvalitou obrazu a velikostí souboru. Toho je docíleno ztrátovou kompresí. Obraz je rozdělen na bloky o velikosti 8×8 pixelů, na které je jednotlivě aplikována komprese. Dochází tak ke ztrátě dat, což se typicky projevuje artefakty v oblastech ostrého kontrastu. Zmenšení velikosti souboru je ale takovou výhodou, že se jedná o dodnes nejpoužívanější formát ve světě internetu. Typicky je tak i výchozím formátem pro neprofesionální fotoaparáty a kamery mobilních telefonů.

PNG

Zkratka pro Portable Network Graphic je formát standardizován roku 2004, původně vytvořen jako vylepšená náhrada formátu GIF, jehož hlavní nevýhodou byla limitovaná barevná hloubka. Dnes je de facto protipólem formátu JPEG, jelikož používá bezztrátovou kompresi, čímž docílí zachování všech původních detailů. Je tak oblíbeným formátem pro grafiku obsahující text, rovné hrany a ostrý kontrast, což jsou oblasti kde typicky u formátu JPEG kompresí vznikají artefakty.

TIFF

Zkratka pro Tagged Image File Format je formát vyvinut v roce 1996 společností Aldus, se záměrem vytvořit standardní formát pro skenery. Je to velmi komplexní formát, jehož hlavní výhodou ve vysoká flexibilita. Nabízí rozsáhlou škálu typů ztrátových i bezztrátových kompresí a barevných modelů. Kromě typicky používaného barevného modelu RGB, vhodného pro zobrazení na obrazovkách, umí například model CMYK používaný pro tisk. Dále dokáže v jediném souboru ukládat vícero stran. Toto množství funkcí je ale logicky vykoupeno násobně větší velikostí souboru v porovnání s předchozími formáty. Využívá se tak obvykle u profesionálních úprav fotografií, nebo třeba v grafickém designu.

RAW

Nejedná se o zkratku, nýbrž anglické slovo *raw* (surový). Jak název napovídá, soubor obsahuje minimálně zpracovaná data získaná ze senzoru digitálního fotoaparátu. Soubor ale není určen k přímému zobrazení obrazu, obsahuje pouze nezpracovaná data, která slouží k jeho vytvoření. Formát není unifikovaný, každý výrobce fotoaparátu má vlastní implementaci a většinou i tak vlastní software pro konverzi. To je společně s enormní velikostí souboru hlavní nevýhodou, přináší ale velikou flexibilitu, kterou využívají hlavně profesionální fotografové.

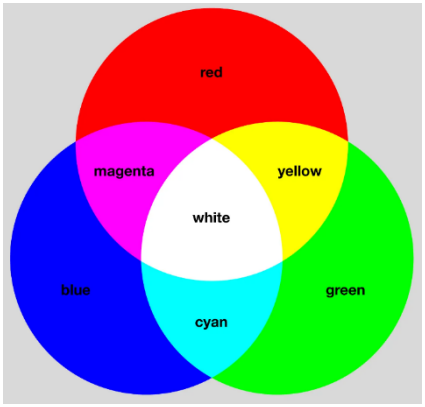
1.3.2 Barevné modely

Sítnice lidského oka obsahuje tyčinky, které reagují na intenzitu světla, a tři druhy čípků, reagující na červené, zelené a modré světlo. Kombinací jejich signálů dokáže člověk vnímat široké spektrum barev. Aby bylo možné tento specifický rozsah barev plně adaptovat do digitálního prostoru, vznikly matematické modely popisující barvy jako kombinaci jednotlivých složek. Tyto složky jsou většinou základní barvy barevného světla, existují ale i modely pracující s dalšími parametry.

RGB

Zkratka RGB znamená red (červená), green (zelená) a blue (modrá). Pracuje tak se stejnými základními barvami jako čípky v lidském oku. Na rozdíl od oka ale nepracuje s parametrem jasu. Je totiž aditivní model fungující na skládání červeného, zeleného a modrého světla s různou intenzitou. Maximální intenzita všech tří kanálů tak vytvoří bílou barvu, minimální intenzita pak barvu černou. V digitální podobě se obvykle hodnoty ukládají v osmibitovém formátu, tedy v rozsahu 0 až 255. (255,0,0) je tak čistě červená, (0,255,0) čistě zelená a (0,0,255) čistě modrá.

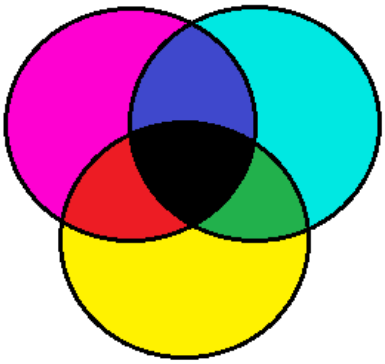
Využívá se hlavně v digitálních zařízeních. Pixely obrazovek jsou složeny z podsvícených bodů červené, zelené a modré barvy, které různým poměrem rozsvícení zobrazují požadovanou barvu. Senzory digitálních fotoaparátů a skenerů každý bod ukládají jako intenzitu těchto tří barev. Pro svou jednoduchost je také standardně používán v softwaru pro tvorbu počítačové grafiky a pro tvorbě webových stránek [4].



Obrázek 1: Barevný model RGB (zdroj [4])

CMYK

Zkratka znamená *cyan* (tyrkysová), *magenta* (purpurová), *yellow* (žlutá) a *key* (klíč neboli černá). Na rozdíl od modelu RGB pracuje s odečítáním světla, je tedy modelem subtraktivním. Výsledná barva je kombinací tří kanálů (CMY) na bílém pozadí. Tyrkysová absorbuje červené světlo, zelené a modré odráží. Purpurová absorbuje zelené světlo a zbylé odráží, žlutá absorbuje světlo modré a zbylé téže odráží. Teoreticky se tak kombinací všech absorbuje veškeré světlo a vznikne barva černá. Jelikož ale v praxi nějaké světlo bude i tak odraženo, nevznikne čistá černá barva, k čemuž slouží čtvrtý kanál key – černá barva. Samostatná černá barva je používána i z ekonomických důvodů, tento barevný model se totiž převážně používá u tisku, kde tiskárna používá čtyři tonery obsahující inkoust nebo prášek základních barev, jejich nanášením v různém poměru na bílý papír skládá požadované barvy [5].

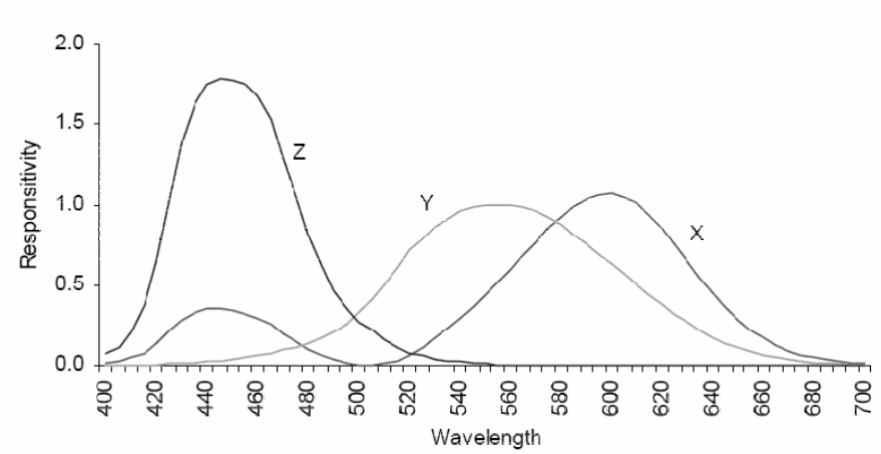


Obrázek 2: Barevný model CMYK (zdroj [5])

CIE XYZ

CIE (z francouzského Commission internationale de l'éclairage) je zkratka Mezinárodní komise pro osvětlování. V roce 1931 vytvořila jeden z prvních matematických modelů pro definici barev, nazvaný CIE XYZ. Vznikl prováděním experimentů, jejichž účastníci upravovali červené, zelené a modré světlo tak, aby vytvořili zadanou barvu. Výsledkem měření je funkce shody barev a jejich zprůměrováním vznikla funkce, která je základem modelu XYZ. Parametr Y reprezentuje světelnost a odvozené parametry X a Z určují barvu.

Cílem formátu je reprezentace světelných hodnot skutečně vnímaných lidským okem. Je tak nezávislý na zobrazovacím zařízení, na rozdíl od formátů RGB a CMYK [6].

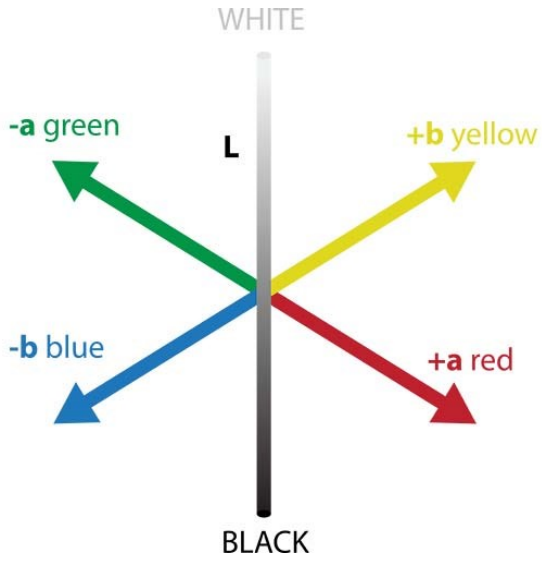


Obrázek 3: Funkce shody barev modelu CIE XYZ (zdroj [6])

CIELAB (CIE L*a*b)

Hlavní nevýhodou modelu CIE XYZ je neintuitivní znázornění barvy v geometrickém prostoru. Například barvy, které jsou vizuálně velmi odlišné, se mohou nacházet v geometrickém prostoru poměrně blízko. Používání takového modelu je tak neintuitivní, což se snaží napravit model CIELAB, vytvořen roku 1976 také organizací CIE. V tomto modelu tak rozdíl mezi vzdáleností a změnou barvy více konzistentní a intuitivní. Na rozdíl od standardně používaných formátů RGB a CMYK zahrnuje kompletní barevný gamut vnímaný lidským okem.

Osa L (*lightness)* udává světelnost. Osa udává poměr zelené a červené barvy, kde je zelená na negativním konci (-a) a červená na pozitivním (+a). Osa b je poměr modré a žluté barvy, kde je modrá na negativním konci (-b) a žlutá na pozitivním (+b). Při vyvážení os **a** i **b**, tedy nastavení hodnot na 0, vznikne šedivá barva, jejíchž intenzita je kontrolována parametrem L, který je udáván procenty. 0 % je barva černá a 100 % barva bílá [7].



Obrázek 4: Barevný model CIELAB (zdroj [7])

1.5 Jasové transformace

Jasové transformace jsou matematické operace, které podle daného pravidla mění jasové hodnoty jednotlivých pixelů v obraze. Provádí se například pro úpravu kontrastu, korekci nerovnoměrného osvětlení, nebo zvýraznění detailů. Lze je rozdělit do tří kategorií – globální, lokální a bodové [8].

- **Globální** – nová hodnota pixelu je vypočítána z hodnot celého obrazu
- **Lokální –** nová hodnota pixelu je vypočítána z hodnot okolních pixelů
- **Bodové** – nová hodnota pixelu je vypočítána z hodnoty toho samého pixelu

Histogram

Histogram obrazu je grafické znázornění rozložení jasových hodnot v obraze. Lze zobrazit sloupcovým grafem, kde osa x zobrazuje možné jasové hodnoty a osa y jejich četnost.
Z histogramu lze rychle zjistit, jak světlý obraz je (podle hromadění pixelů), nebo úroveň kontrastu (špičaté vrcholy značí vysoký kontrast, plochý histogram značí nízký kontrast).

1.6 Filtrace obrazu

Filtrace obrazu je proces jeho úpravy za účelem zvýšení kontrastu, zvýraznění detailů, nebo třeba redukci šumu [9].

Šum v obraze

Šum v obraze jsou náhodné variace jasových hodnot pixelů, které jsou většinou nežádoucí. Může pocházet z různých zdrojů, jako je například šum senzoru fotoaparátu, což se projevuje tečkami nebo pruhy v obraze. Dále může být způsoben nedostatečným osvětlením, či kompresí s vysokou ztrátovostí.

1.6.1 Konvoluce

Konvoluce je matematická operace, která se používá pro kombinaci dvou funkcí. Spojitá konvoluce je definována tímto vztahem:

$$g(t) = f(t) * h(t) = \int_{-\infty}^{\infty} f(t)\, h(t-\tau)\, \mathrm{d}\tau = \int_{-\infty}^{\infty} f(t-\tau)\, h(t)\, \mathrm{d}\tau$$

- **Funkce f(t)** – při zpracování obrazu představuje obraz
- **Funkce h(t)** – nazývá se konvoluční jádro, při zpracování obrazu představuje filtr aplikovaný na obraz
- **Funkce g(t)** – výsledná funkce vzniklá kombinací **f(t)** a **h(t)**

Konvoluci si jde představit jako posouvání funkce **h(t)** po ose x a násobení jejich hodnot odpovídajícími hodnotami funkce **f(t)**. Výsledkem je funkce **g(t)**.

1.6.2 Vyhlazování obrazu

Vyhlazování obrazu je technika zpracování obrazu, jejíž cílem je redukce šumu a odstranění drobných detailů. Princip spočívá v aplikaci konvolučního jádra, které definuje váhy pro sousední pixely, na každý pixel obrazu. Existuje více typů vyhlazování, která mají na výsledný obraz různý dopad.

Vyhlazování průměrováním

Při vyhlazování průměrováním jádro přiřazuje všem pixelům v okně stejnou váhu. Je jednoduché na výpočet, ale může způsobit ztrátu detailů a zploštění obrazu.

Příklad jádra 3×3:

[1/9, 1/9, 1/9]

[1/9, 1/9, 1/9]

[1/9, 1/9, 1/9]

Gaussovo vyhlazování

Při Gaussově vyhlazování jádro přiřazuje pixelům váhu podle Gaussovy funkce, která klesá s rostoucí vzdáleností od středu okna. Je výpočetně náročnější než vyhlazování průměrováním, ale zachovává více detailů v obrazu.

Příklad jádra 3×3:

[0.0625, 0.125, 0.0625]

[0.125, 0.25, 0.125]

[0.0625, 0.125, 0.0625]

Mediánová filtrace

Při této filtraci jádro nepoužívá váhy, ale nahrazuje hodnotu pixelu mediánem hodnot pixelů v okně. Zachovává detaily v obrazu, ale může způsobit jeho rozostření. Nejvíce se hodí pro potlačení šumu typu „sůl a pepř“ (izolované světlé a tmavé pixely).

Příklad jádra 3×3:

[100, 98, 102]

[99, 105, 101]

[95, 100, 255]

V tomto případě je aritmetický průměr jádra je 117,2, medián je ale 100. Pixel s hodnotou 255 tak neovlivní hodnotu ostatních pixelů, ale právě on je potlačen na hodnotu více odpovídající jeho okolí.

1.6.3 Detekce hran

Detekce hran je technika zpracování obrazu, jejíž cílem je nalezení oblastí v obraze, kde se prudce mění jas. Právě tyto oblasti jsou pro lidské oko klíčové při hledání objektů v okolí [10].

Sobelův operátor

Sobelův operátor používá dvě předem definované matice, jednu pro horizontální a druhou pro vertikální změny. Tyto matice mají obvykle velikost 3×3 a obsahují váhy, které umožňují detekci hran daném směru. Konvoluce těchto matic s obrazem poskytuje dvě aproximace gradientu obrazu v horizontálním a vertikálním směru. Výsledkem jsou míry změny jasu v daném směru.

Cannyho operátor

Cannyho operátor na rozdíl od Sobelova operátoru, který se zaměřuje pouze na velikost gradientu, zohledňuje i další faktory pro přesnější detekci hran. Postup má několik kroků:

1. **Výpočet směrů gradientu** – lze použít například Sobelův operátor.
2. **Potlačení nemaximálních hodnot** – pixely, které nejsou lokálními maximy ve svém směru jsou potlačeny a zůstávají tak jen pixely reprezentující hřbety gradientu. Tato technika se nazývá ztenčení hran.
3. **Hysteréze** – zvolí se dva prahy, maximální a minimální. Pixely s hodnotou gradientu nad maximální prahem jsou považovány za součást hrany. Pixely s hodnotou gradientu pod minimálním prahem nejsou považovány za součást hrany. Pro zbývající pixely nacházející se mezi prahy se provede trasování směrem od pixelů nad horním prahem a pokud je jeho soused součástí hrany, je do ní započítán taky. Tímto jsou eliminovány nevýznamné hrany.

1.7 Morfologie

Morfologie obrazu je oblast zabývající se analýzou a manipulací s obrazovými daty na základě jejich struktury, tvaru a vzájemných vztahů mezi objekty v obraze. Matematické operace se aplikují na binární, nebo šedotónové obrazy [11].

Morfologická transformace je dána operací nad množinou pixelů reprezentující obraz s použitím menší množiny, která se nazývá strukturálním elementem. Aplikace transformace odpovídá systematickému posunu strukturálního prvku po obraze.

Morfologie pracuje s dvěma základními operacemi, diletací a erozí.

1.7.1 Diletace

Diletace je operace, která rozšiřuje objekty v obraze o daný strukturální prvek. Při diletaci se po každém pixelu v původním obraze postupně posouvá strukturální prvek. Pokud alespoň jeden pixel ve strukturálním prvku odpovídá objektovému pixelu v původním obraze, pak je výstupem transformace nový objektový pixel v odpovídající pozici.

Diletace se používá k zaplnění malých děr a úzkých zálivů v objektech. Společně s tím se ale zvětší celý objekt. Pro navrácení do původní velikosti lze použít erozi.

vlevo – originál, vpravo – dilatace.

Obrázek 5: Objekt před a po diletaci (zdroj [11])

1.7.2 Eroze

Eroze je operace, která ztenčuje objekty v obraze o daný strukturální prvek. Při diletaci se po každém pixelu v původním obraze postupně posouvá strukturální prvek. Pokud alespoň jeden pixel ve strukturálním prvku odpovídá pozadí původního obrazu, pak do odpovídajícím pozice není umístěn nový objektový pixel.

Eroze se používá pro odstranění malých izolovaných objektů (šum), tenkých výběžků, vyhlazení hran a oddělení těsně spojených objektů.

Obrázek 6: Objekt před a po erozi (zdroj [11])

1.8 Segmentace obrazu

Segmentace obrazu spočívá v oddělení obrazu na jednotlivé oblasti se společnými vlastnostmi s ohledem na jejich význam v daném kontextu [12].

Cíle segmentace

- **Identifikace objektů** – segmentace umožňuje identifikovat a oddělit jednotlivé části odpovídající objektům reálného světa
- **Analýza vlastností objektů** – na základě segmentovaných oblastí lze analyzovat vlastnosti objektů, například jejich tvar, velikost nebo vzájemné uspořádání
- **Klasifikace objektu** – na základě analyzovaných vlastností je možné objekty klasifikovat do určitých kategorií

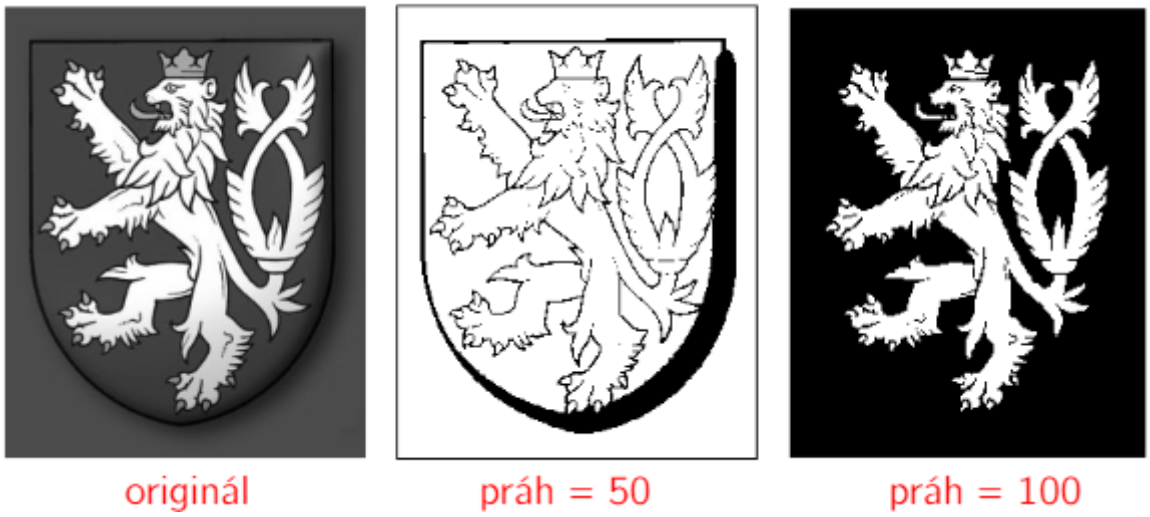
1.8.1 Segmentace prahováním

Objekty a oblasti jsou charakterizovány tím, že jejich povrch je do jisté míry konzistentní, což je dáno jasem a barvou. Tím vystupují z pozadí, čehož využívá prahování.

Prahování je bodová jasová transformace vstupního obrazu na výstupní binární obraz. Pro tuto operaci je klíčový práh. Pixely, které mají jas vyšší, než určený práh nabydou hodnoty jedna. Pixely s jasem nižším, než určený práh nabydou hodnoty nula. Hledané objekty tak ve výsledku mají hodnotu jedna a jejich pozadí hodnotu nula.

Prosté prahování

Při použití prostého prahování je určen jediný práh. Pixely jsou rozděleny dvě části, pod a nad prahem.



Obrázek 7: Obraz před a po prostém prahování (zdroj [12])

Prahování s více prahy

Lze ale určit více prahů než jeden. Výsledkem pak nemusí být rozdělení na černé a bílé pixely. Každý práh může určovat úroveň na vlastní šedotónové škále.

Adaptivní prahování

U adaptivního prahování je práh funkcí polohy v obraze, není tak určován pro celý obraz, ale jen jeho část. Obraz je rozdělen na menší části, na které je jednotlivě aplikováno prahování s vlastním prahem, například získaným střední hodnotou jasu v dané části.

Tento způsob prahování je velmi výpočetně náročný, nicméně je vhodný pro snímky s nerovnoměrným osvětlením.

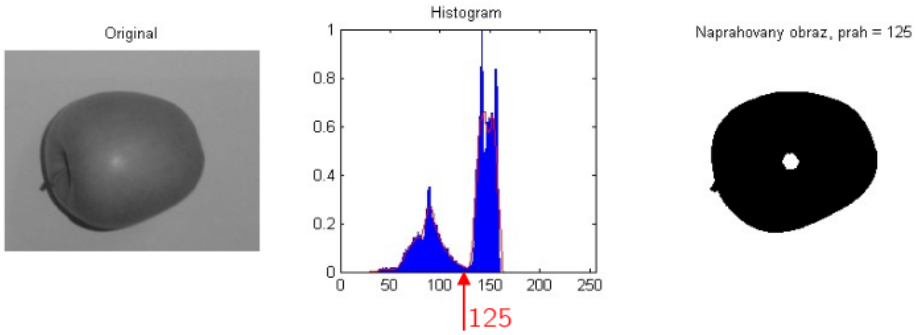
Obrázek 8: Porovnání globálního a adaptivního prahování (zdroj [12])

Určení prahu

Úspěšnost prahování je dána určením ideálního prahu. Ten lze nalézt několika způsoby:

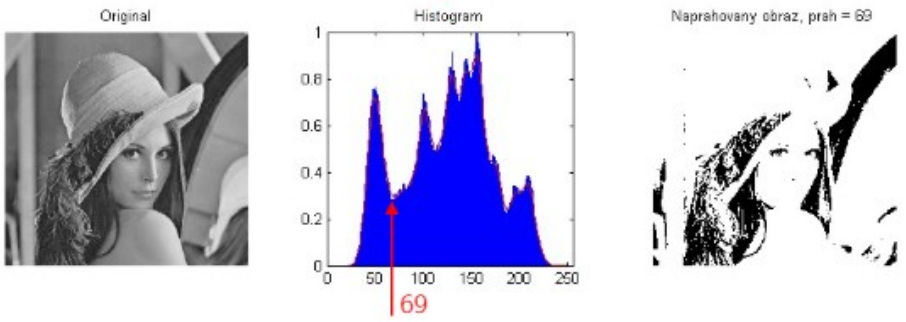
- **Experimentování** – metodou pokus omyl lze manuálně postupně zvyšovat práh, dokud není nalezen vyhovující

- **Histogram** – pokud histogram obsahuje dvě maxima, lze usoudit, že jedno přísluší pozadí a druhé objektům. Nalezením lokálního minima mezi maximy, nebo jednoduše rozpůlením vzdálenosti mezi nimi lze nalézt požadovaný práh. Tato metoda je vhodná pro obrazy jednolitým pozadím. Komplexní obrazy mají maxim histogramu více, nelézt to požadované je pak obtížné.



Obrázek 9: Nalezení prahu histogramem (zdroj [12])

- **Otsuova metoda** – z histogramu získá všechny body, ve kterých by se mohl nacházet práh. Následně pro všechny vypočte rozptyl. Bod, pro který byl vypočten maximální rozptyl je určen jako práh.



Obrázek 10: Nalezení prahu Otsuovou metodou (zdroj [12])

- **Procentní metoda** – vychází z odhadu plochy, kterou zabírá hledaný objekt vzhledem k celému obrazu. Tato metoda je vhodná například pro stránky potištěné textem.

1.8.2 Segmentace z obrazu hran

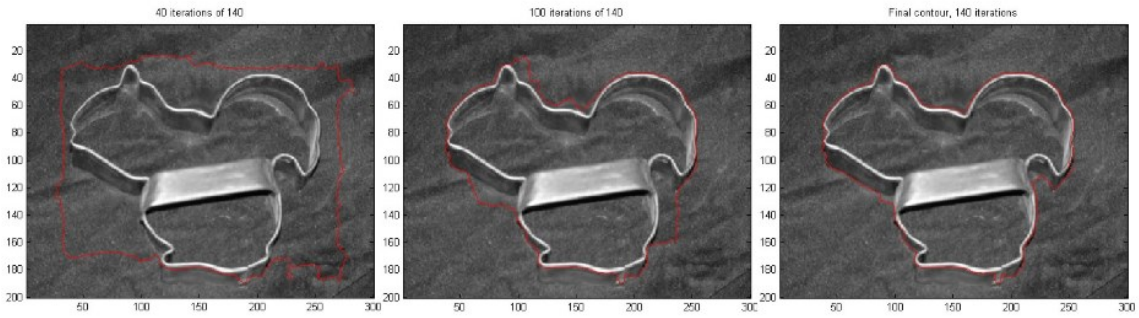
Segmentace z obrazu hran je metoda zaměřující se na využití hranových detektorů k nalezení hran, které jsou využity k rozdělení na segmenty odpovídající daným objektům. Tato metoda segmentace je náchylná na následující body:

- **Minimální počet chyb** – segmentace bude chybná, pokud je opomenuta některá z významných hran, nebo byla detekována místa, která nejsou hranami
- **Přesnost** – rozdíl mezi nalezenou a skutečnou hranou musí být minimání
- **Jednoznačnost** – na každou hranu musí být reagováno jen jednou, což se nestane při detekci duplikátní hrany

Aktivní kontury (snakes)

Aktivní kontura, tedy uzavřená křivka postupně deformuje tak, aby se přizpůsobila hranicím objektu v obraze. Křivka deformuje třemi silami:

- **Vnitřní síly** – kontrolují hladkost průběhu
- **Obrazové síly** – směřují tvarování křivky směrem k hraně objektu
- **Vnější síly** – jsou výsledkem počátečního umístění křivky



Obrázek 11: Segmentace aktivní konturou (zdroj [12])

Houghova transformace

Houghova transformace je metoda vhodná v případě, že hledaný objekt má známý analytický popis. Například jednoduché tvary jako přímka, elipsa, trojúhelník nebo čtverec. Nejlépe funguje na binárních snímcích (po prahování).

Je založena na principu mapování obrazového prostoru do prostoru parametrů, tedy body se mapují na křivku a naopak. Následně je provedeno hlasování, tedy sčítání, kolik bodů patří k danému tvaru, čímž je vytvořen takzvaný akumulační prostor. Nakonec proběhne hledání maxima.

Výhodou metody je nízká citlivost na šum a potušení hranic. Nicméně je časově náročná a nemusí být příliš přesná, může docházet k duplikaci přímek.

1.8 Optické rozpoznávání znaků (OCR)

Optické rozpoznávání znaků (anglicky Optical Character Recognition) je proces, který dokáže převést obraz textu do strojově čitelného textového formátu. Příkladem je fotografie stránky knihy, která je převedena do textového dokumentu. Technologie využívá techniky počítačového vidění a strojového učení [13].

Postup procesu rozpoznání

- **Preprocessing** – technikami počítačového vidění jsou provedeny úpravy, které očistní obraz od chyb, které by mohly znemožňovat následné rozpoznání. Mezi takové úpravy patří například vyhlazení pro redukci šumu (prach na fyzickém dokumentu), morfologické operace pro získání ideální tloušťky znaků, nebo otočení obrazu tak, aby řádky textu byly vodorovné.
- **Rozpoznání znaků** – pomocí modelů natrénovaných strojovým učením proběhne rozpoznání jednotlivých znaků. Obvykle je použit jeden ze dvou hlavních algoritmů.
    - *Hledání shody se vzorem* – obraz izolovaného jednoho znaku je porovnáván s uloženými obrazy znaků, u kterých je známa jejich hodnota. Nevýhodou tohoto algoritmu je vázanost na font. Uložené vzory jsou obvykle v běžných fontech a pro málo používané fonty nemusí existovat vzor.

    - *Extrakce vlastností* – obraz izolovaného jednoho znaku je rozložen do vlastností jako jsou přímky a jejich směřování a protínání nebo uzavřené smyčky. Na základě těchto vlastností se pokusí najít znak s co nejvíce shodnými vlastnostmi.
- **Postprocessing** – po provedené analýze jsou získaná data převedena do požadovaného typu souboru

Využití

- **Konverze fyzických archivů do snadno přístupných digitálních archivů** – digitalizace tištěných knih a rukopisů
- **Bankovnictví** – automatické zpracování papírových dokumentů o finančních transakcí, čímž je zajištěna vyšší bezpečnost
- **Zdravotnictví** – automatické zpracování lékařských záznamů pacienta nebo dokumentaci o pojištění
- **Logistika** – zrychlení trasování balíků, automatické zpracování faktur a dalších finančních dokumentů

2 EXPERIMENTÁLNÍ ČÁST

2.1 Použité technologie

2.1.1 Operační systémy

Android

Android je open source operační mobilní systém představen veřejnosti roku 2008 společností Google, která je zároveň jeho hlavním vývojářem. Otevřenost systému přispěla k jeho rozšířenosti, výrobci mobilních zařízení místo vytvoření vlastního operačního systému často používají vlastní nadstavbu Androidu. Je používán 70,69 % trhu s mobilními zařízeními, jediným velkým konkurentem je operační systém iOS s 28,58 %, ostatní systémy mají tak jen zlomky procent. Pokud je tedy žádoucí vytvořit mobilní aplikaci pro co největší množství uživatelů, operační systém Android by měl být první volbou.

Od začátku vývoje aplikací pro Android byla Java preferovaným programovacím jazykem, což bylo doporučováno i samotným Googlem. Toto místo ale roku 2019 zaujal jazyk Kotlin [14].

2.1.2 Programovací jazyky

Programovací jazyk C#

C# je vysokoúrovňový objektově orientovaný jazyk zveřejněn roku 2000 společně s platformou .NET Framework. Oblíbený je zejména pro jeho víceúčelovost. Je používán pro desktopové aplikace pomocí Windows Forms, nebo modernějšího WPF (Windows Presentation Foundation). Dále je používán v kombinaci s frameworkem ASP.NET pro vývoj webových aplikací. Populární je i pro vývoj počítačových her, například s enginem Unity. Lze použít i pro vývoj multiplatformních mobilních aplikací frameworkem Xamarin [15].

Programovací jazyk Kotlin

Kotlin je multiplatformní staticky typovaný programovací jazyk vyvíjený společností JetBrains od roku 2011. Je plně kompatibilní s jazykem Java, jeho knihovnami a runtime prostředím JVM (Java Virtual Machine). Roku 2019 společnost Google ustanovila Kotlin jako preferovaný jazyk pro vývoj Android aplikací namísto jazyku Java. Výhodou Kotlinu je totiž čistší a jednodušší syntax, což zlepšuje čitelnost a rychlost psaní kódu [16].

2.1.3 Knihovny

MySqlConnector

MySqlConnector je open source knihovna napsaná v jazyce C#, určená pro komunikaci s SQL databází skrze jazyk C#. Podporuje nejen MySQL, jak napovídá název, ale například i MariaDB, Amazon Aurora a další [17].

Selenium WebDriver

Selenium WebDriver je sadou nástrojů určených pro automatizaci interakcí s webovým prohlížečem. Skrze kód je tak možné například otevřít webovou stránku, vyplnit a odeslat formulář. WebDriver není vázán na konkrétní programovací jazyk, existují tak knihovny pro různé programovací jazyky, skrze které obsluhující WebDriver API [18].

Emgu CV

Emgu CV je knihovnou určenou pro všechny jazyky kompatibilní s platformou .NET. Knihovna samotná nic neimplementuje, je pouze obalem pro knihovnu OpenCV. OpenCV je open source knihovna implementující algoritmy z oblasti počítačového vidění, napsaná v jazyce C++ [19].

Tessaract

Tessaract je open source OCR (Optical Character Recognition) engine napsaný v jazyce C++, původně vyvinut společností Hewlett-Packard a od roku 2006 ve vývoji pokračuje společnost Google. Umožňuje extrakci textu z digitálního obrazu. Je velmi flexibilní, podporuje širokou škálu vstupních formátů i výstupních formátů a poskytuje připravené datové sady pro více než 100 jazyků a umožňuje i trénování pro možnost rozpoznání dalších jazyků. Dále je multiplatformní, podporuje operační systémy Windows, MacOS i Linux a obalové knihovny pro použití nalezneme ve většině populárních programovacích jazyků [20].

Volley

Volley je open source Android knihovna pro obsluhu HTTP požadavků, vyvíjená společností Google od roku 2013. Její výhodou je rychlost a jednoduchost použití, není ale vhodná pro práci s velkým objemem dat [21].

2.1.4 Databázové servery

MySQL

MySQL je open source systém řízení báze dat používající relační databázový model (RDBMS). Byl vyvinut párem nezávislých vývojářů v roce 1995 a později odkoupen společností Oracle. Výhodou je možnost nasazení na široké škále operačních systému. Populární je hlavně u webových aplikací založených na databázi [22].

2.1.5 Integrovaná vývojová prostředí

Integrované vývojové prostředí (zkratka IDE z anglického Integrated Development Enviroment) je počítačový program, který slouží k psaní zdrojového kódu. Kromě editace kódu ale nabízí další nástroje, které pomáhají s vývojem softwaru, jako je například kontrola syntaxe v reálném čase, našeptávač, nástroje pro kompilaci nebo interpretaci kódu, debugger či sledování využití prostředků.

Existují IDE zaměřené na jediný programovací jazyk, často jich ale podporují hned několik. Mezi nejznámější poskytovatele patří Microsoft (Visual Studio, Visual Studio Code), JetBrains (IntelliJ IDEA, PyCharm, Android Studio) nebo Eclipse Foundation (Eclipse) [23].

Visual Studio

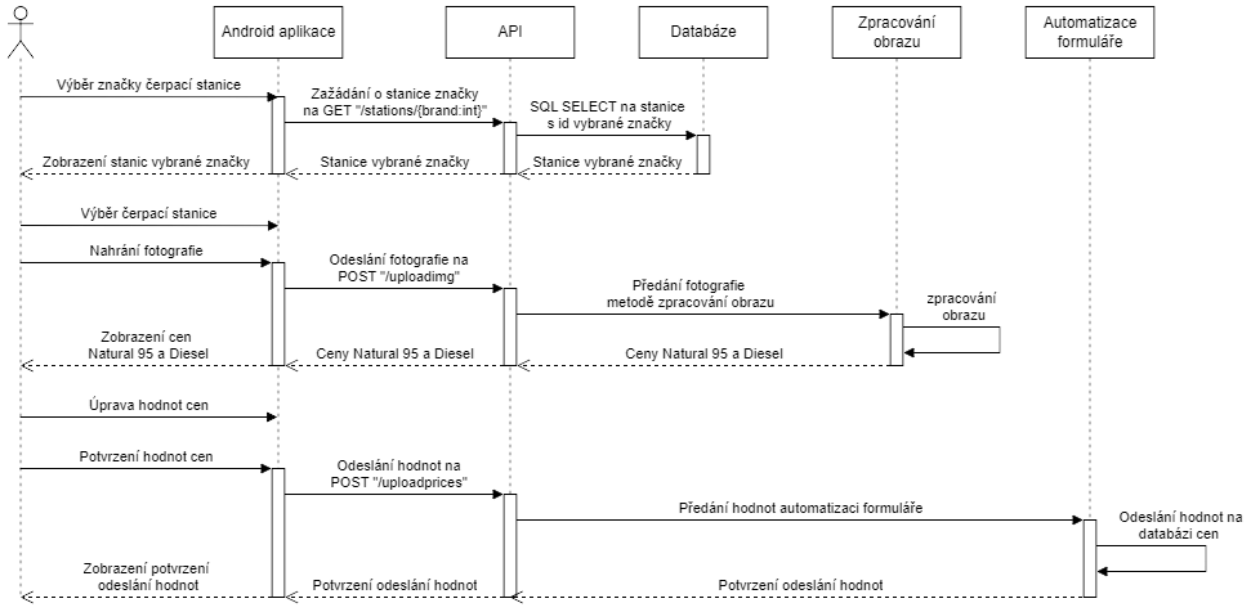
Visual Studio je IDE vyvíjené společností Microsoft od roku 1997. Na rozdíl od sesterského IDE Visual Studio Code zaměřujícího se na minimální výchozí funkce, které jsou přidávány nezávislými rozšířeními, je Visual Studio komplexním nástrojem. Má vestavěnou podporu několika programovacích jazyků, jak vlastních Microsoftu (Visual Basic, C#, F#), tak i dalších jako C, C++, HTML, CSS, JavaScript a XML. Prostřednictvím rozšíření lze používat i další jazyky jako například Python nebo Ruby [24].

Mezi vestavěné nástroje patří našeptávač IntelliSense, nástroje pro sestavení a ladění, správu balíčků NuGet (knihovny) a dokonce i přímé propojení IDE s databází, serverem, nebo úložištěm Git.

Android Studio

Android Studio je IDE společně vyvíjené společností Google a JetBrains. Je postavené na IDE IntelliJ IDEA, specificky upravené pro vývoj mobilních aplikací pro operační systém Android. Kromě oficiálně podporovaného programovacího jazyku Kotlin umožňuje i vývoj v jazyce Java a C++. Mimo to obsahuje emulátor zařízení s operační systémem Android a grafické prostřední pro design aplikací [25].

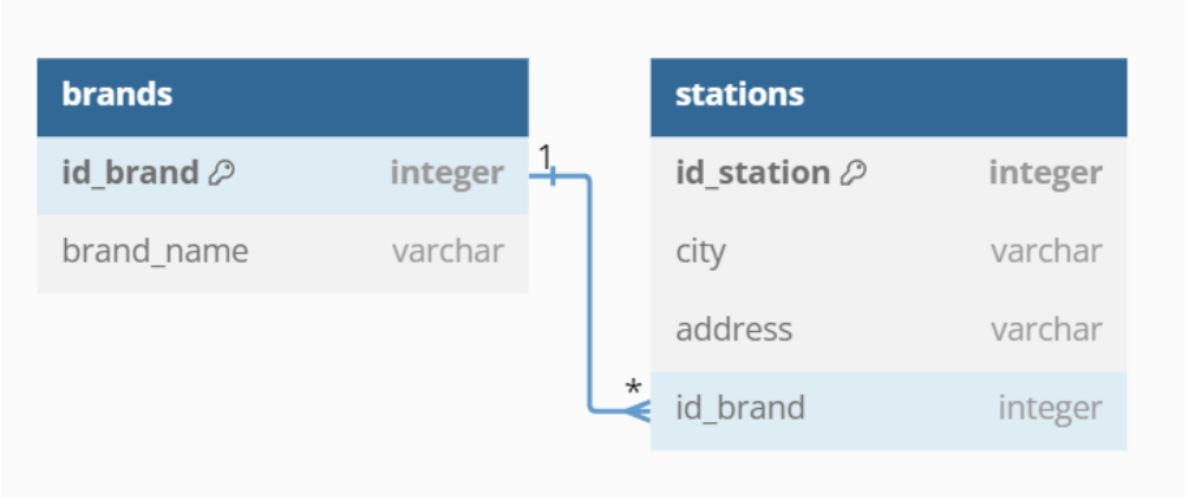
2.2 Implementace mobilní aplikace pro rozpoznání cen pohonných hmot



Obrázek 12: Flowchart aplikace

Hlavním cílem projektu je vytvoření metody využívající technik počítačového vidění pro získání cen pohonných hmot z digitální fotografie stojanu čerpací stanice. Metoda je zakomponována do serverové aplikace. V této aplikaci také probíhá načítání dat z databáze čerpacích stanic a odesílání získaných informací na webový portál ceskybenzin.cz. Tyto funkce aplikace je možné obsluhovat pomocí webové API. Pro obsluhu této API je vytvořena mobilní aplikace pro operační systém Android.

2.2.1 Databáze čerpacích stanic



Obrázek 13: Relační model databáze čerpacích stanic

MySQL databáze slouží k ukládání informací o čerpacích stanicích. Je tím tak zajištěna snadná škálovatelnost, přidání dalších značek a stanic. Jediná potřebná změna kódu aplikace je tak přidání metody pro zpracování obrazu pro novou stanici.

Tabulka značek čerpacích stanic

Tabulka značek čerpacích stanic, pojmenovaná *brands*, obsahuje dva sloupce

- **id_brand** – primární klíč, datový typ integer, automatická inkrementace
- **brand_name** – datový typ varchar, udává název značky

Tabulka čerpacích stanic

Tabulka čerpacích stanic, pojmenovaná *stations*, obsahuje čtyři sloupce

- **id_station** – primární klíč, datový typ integer, hodnota je dána unikátním id stanice pocházejícím z portálu ceskybenzin.cz (např. https://m.ceskybenzin.cz/aktualizace.php?id=3094)
- **city** – datový typ varchar, udává město, kde se stanice nachází
- **address** – datový typ varchar, udává adresu, kde se stanice nachází
- **id_brand** – cizí klíč, odkazuje na primární klíč tabulky *brands*

2.2.2 Serverová obslužná aplikace

Serverová obslužná aplikace je naprogramována v jazyce C# s využitím IDE Visual Studio. Jedná se o webovou API využívající technologii ASP.NET. Pro přehlednost kódu je aplikace rozdělena do několika tříd zorganizovaných do složek.

Třída Program

Třída Program je hlavní třídou, obsahuje totiž metodu `Main` a kód této třídy tak proběhne při spuštění aplikace. Nejprve proběhne připojení k databázi MySQL za pomocí knihovny

`MySqlConnector`. Poté je instanciován objekt třídy `WebApplication`, na který jsou postupně namapovány čtyři koncové body API.

- **/uploadimage** – příchozí fotografie v datovém typu `IFormFile` je převedena na datový typ `Bitmap`. Poté je vytvořena instance třídy `ImageProcessor`, která vyvolá metodu `GetPrices(Bitmap bitmap, string stationBrand)`, která do parametrů pojme fotografii a název přijatého souboru, který by měl obsahovat název značky čerpací stanice. Pokud metoda dokázala rozpoznat ceny, navrátí koncový bod status 200 OK společně s nalezenými cenami. V opačném případě navrátí status 404 Not Found.
- **/uploadprices** – příchozí data o cenách pohonných hmot v datovém typu `GasStationPrices` jsou předány do metody `SendInfo(GasStationPrices prices)`, vyvolané instancí třídy `FormFiller`. Poté navrátí status 200 OK.
- **/brands** – koncový bod získá z databáze všechny značky stanic SQL příkazem (`SELECT id_brand, brand_name FROM brands ORDER BY brand_name;`). Získaná data navrátí datovým typem `List<Brand>`.
- **/stations/{brand:int}** – koncový bod získá z databáze všechny stanice té značky, které přísluší id přijatého čísla, SQL příkazem (`SELECT id_brand, brand_name FROM brands ORDER BY brand_name;`). Získaná data navrátí datovým typem `List<GasStation>`.
- Nakonec objekt třídy `WebApplication` zavolá metodu `Run`, čímž je aplikace webové API spuštěna.

Tabulka 1: Koncové body API

URI	HTTP metoda	Účel
/uploadimage	POST	Zaslání fotografie stojanu čerpací stanice a navrácení získaných hodnot
/uploadprices	POST	Zaslání cen pohonných hmot na informační server
/brands	GET	Získání seznamu všech značek čerpacích stanic z databáze
/stations/{brand:int}	GET	Získání všech stanic dané značky z databáze

Třídy ve složce Model

Třídy v této složce definují vlastní datové typy použité v aplikaci. Obsahují tak pouze několik veřejných vlastností (`public <datový typ> <název> { get; set; }`).

- **Brand** – `int Id, string Name`
- **GasStation** – `int Id, string City, string Address, Brand Brand`
- **GasStationPrices** – `int StationId, double Natural95, double Diesel`

2.2.2.1 Metoda zpracování obrazu

Třídy ve složce ImageProcessing

- **ImageProcessor** – obsahuje jedinou veřejnou metodu `GetPrices(Bitmap bitmap, string stationBrand)`, která přepínačem podle `stationBrand` vyvolá metodu rozpoznání obrazu příslušné značky čerpací stanice. Vrací datový typ `GasStationPrices?`, tedy pokud metoda pro danou stanici neexistuje, vrátí `null`.
- **ProcessingMethod** – tato třída obsahuje statické veřejné metody, které implementují zpracování obrazu pro dané značky čerpacích stanic. Metody přijímají parametr datového typu `Bitmap` a vrací získané hodnoty datovým typem `GasStationPrices`.

Metoda BenzinaProcess

V aplikaci je implementována metoda pro zpracování fotografie stojanu sítě čerpacích stanic Benzina. Historie sítě sahá až do roku 1920, dnešní podoba značky vznikla po privatizaci do holdingové skupiny Unipetrol v roce 1994. V průběhu psaní této práce začala probíhat restrukturalizace Unipetrolu a čerpací stanice po celém území České republiky postupně mění název na Orlen i s tím přichází i nové logo. Stojan s cenami pohonných hmot ale zůstává v podstatě stejný, jen získává nové logo ta vrcholu stojanu.

Obrázek 14: Fotografie stojanů sítě Benzina před a po změně názvu na Orlen

Jak je z fotografie patrné, ceny pohonných hmot jsou zobrazeny segmentovým displejem s jasně červenými diodami, které vystupují z okolí. Cílem metody je tohoto využít, tedy detekovat tyto červené pixely fotografie, jejichž výřezy budou obsahovat jednotlivá čísla. Tyto výřezy poté dostane ke zpracování Tessaract OCR.

Prvním krokem je převedení přijatého objektu datového typu Bitmap na typ Mat, který používá knihovna Emgu CV.

```
Mat originalImage = bitmap.ToMat();
```

Vyhlazení obrazu

Druhým krokem je vyhlazení obrazu. Po přiblížení fotografie lze zpozorovat, že mezi jednotlivými diodami displeje jsou viditelné mezery. Uvnitř obrazu celé číslice tak nejsou pixely jednotné barvy, lze nalézt pixely žádoucí červené barvy, pixely splývající s černým pozadím a přelivy mezi nimi. Vyhlazením obrazu tak většina těchto pixelů nabude stejné barvy, čímž je později zajištěna detekce celého čísla.

Konkrétní použitou metodou je Gaussovo vyhlazování. Velikost jádra je nastavena na jednu setinu šířky obrazu. Velikost jádra musí být lichá, pokud je tedy setina šířky sudá, je připočtena hodnota jedna.

```
int kernelSize = (originalImage.Size.Width / 100) % 2 == 0
      ? (originalImage.Size.Width / 100) + 1
      : originalImage.Size.Width / 100;

CvInvoke.GaussianBlur(originalImage, blurredImg, new Size(kernelSize,
kernelSize), 0);
```

Obrázek 15: Segmentový displej před a po vyhlazení obrazu

Převod do barevného prostoru LAB a oddělení kanálu A

Pro lidské oko jsou sice na první pohled červená čísla nejvýraznějšími objekty ve fotografii, ale pokud bude provedeno prahování přímo na vyhlazený barevný obraz, výsledek nebude nijak užitečný. Pro tyto účely bude potřeba využít oddělený kanál barevného modelu.

Jelikož je obraz pochází z fotografie formátu .jpeg používá barevný prostor RGB. Nabízí se tak možnost oddělení kanálu červené barvy R. Červená čísla jsou sice jedněmi z nejvýraznějších objektů v tomto kanálu, to ale platí i pro bílé objekty jako jsou mraky. Důvod je ten, že barevný model RGB nemá vlastní kanál pro svítivost. V tomto aditivním modelu bílá barva odpovídá maximálním hodnotám ve všech kanálech (255,255,255). Objekty velmi světlých barev, jako jsou v tomto případě například mraky na obloze a střecha odrážející sluneční svit, mají v kanále R taktéž vysoké hodnoty. Pro účely separace červené barvy je tak barevný prostor RGB zcela nevhodný.

Obrázek 16: Prahování barevného obrazu, kanál R z barevného prostoru RGB a jeho prahování

Pro tyto účely je třeba použít barevný model, který separuje informace o svítivosti do vlastního kanálu. Nabízí se tak využití modelu LAB, který svítivost ukládá do kanálu L. Zbývají dva kanály. Kanál A udávající poměr mezi zelenou a červenou a kanál B udávající poměr mezi modrou a žlutou. Čistě červená se nachází na pozitivní straně osy kanálu A, tudíž barvy nejblíže červené budou ve výsledném obraze nejsvětlejší. Narozdíl od modelu RGB zde objekty bílé barvy nedělají potíž, jelikož šedivé barvy nacházející se na škále mezi černou a bílou jsou v modelu LAB na prostředku os A a B.

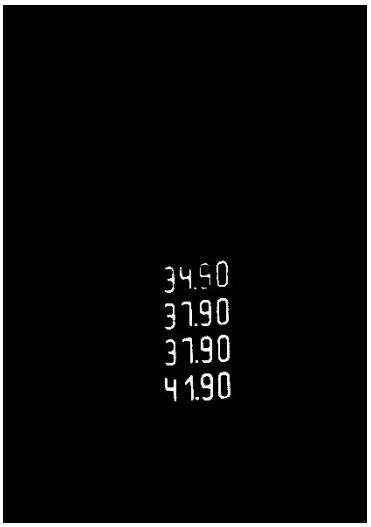
Obrázek 17: Kanály obrazu převedeného do prostoru LAB (v pořadí L, A, B)

Prahování

Jako metoda prahování je vybráno globální binární. Jelikož je získán kanál A, kde jsou pixely čísel nejbělejší v celém obrazu, není třeba automaticky hledat práh, například Otsuovou metodou. Lokální prahování také není zapotřebí, jelikož je barva pixelů čísel sjednocena, čehož bylo docíleno předchozím vyhlazováním.

Na základě testování několika snímků v různých světelných podmínkách je ustanoven práh vyhovující všem podmínkám. Konstantní hodnota prahu je tak stanovena na 69 %, což v rozmezí 0 až 255 odpovídá hodnotě 175.

```
CvInvoke.Threshold(AChannel, threshImg, 175, 255, ThresholdType.Binary);
```



Obrázek 18: Kanál A po prahování

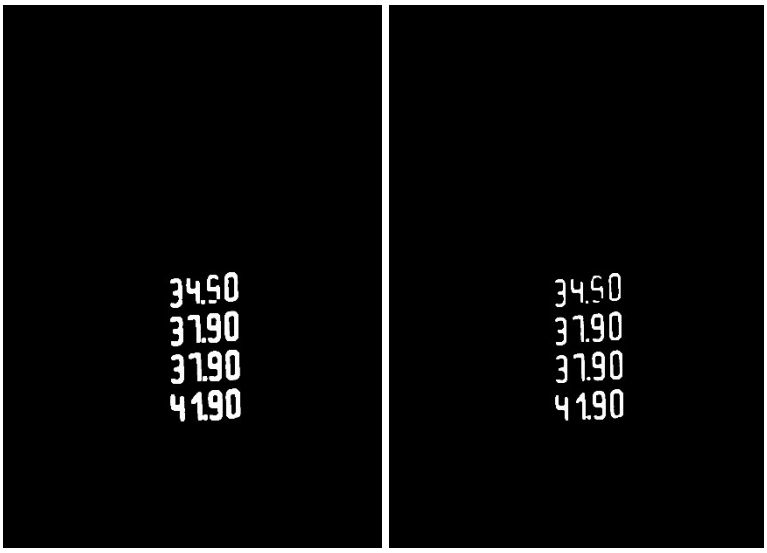
Diletace a eroze

Přesto že bylo aplikováno vyhlazování, nemusí být vždy po prahování číslo jednolité. Následkem špatných světelných podmínek, nebo špatné kvality fotografie se mohou uvnitř čísla nacházet malé oblasti černých pixelů, nebo může být číslo fragmentováno malými mezerami.

Pro zaplnění těchto oblastí je využita diletace na pixely bílé barvy. Velikost strukturního elementu (`anchor`) je stanovena na jednu pětinu jádra použitého při vyhlazení obrazu. Jelikož je opět vyžadována lichá hodnota, je přičtena hodnota jedna, pokud je vypočtená pětina sudá. Jelikož je žádoucí maximální zachování tvaru čísel, je diletace opakována třikrát, namísto trojnásobné velikosti strukturního elementu.

Po diletaci jsou linie čísel příliš široké, pro lepší čitelnost je aplikována eroze, čímž je velikost linií čísel navrácena do té původní, nyní ale se zaplněnými dírami. Eroze je opakována dvakrát, se stejným strukturálním elementem, jako diletace.

```
int anchorSize = (kernelSize / 5) % 2 == 0
      ? (kernelSize / 5) + 1
      : kernelSize / 5;
Mat anchor = new(kernelSize, kernelSize, DepthType.Cv8U,
threshImg.NumberOfChannels);

CvInvoke.Dilate(threshImg, dilatedImg, anchor, new Point(anchorSize / 2,
anchorSize / 2), 3, BorderType.Reflect, new MCvScalar(255));
CvInvoke.Erode(dilatedImg, erodedImg, anchor, new Point(anchorSize / 2,
anchorSize / 2), 2, BorderType.Reflect, new MCvScalar(255));
```



Obrázek 19: Prahovaný obraz po diletaci a následné erozi

Detekce hran

Nyní jsou čísla dostatečně čitelné a jednolité, nicméně zabírají poměrně malý prostor černém pozadí v obraze. Aby bylo možné najít co nejmenší plochu ohraničující bílé pixely, je nejprve potřeba detekovat hrany bílých objektů. Jelikož je potřeba najít jen hrany nejblíže každému rohu, postačí detekce externích hran (návratový mód `External`). Nalezení vnořených hran (v tomto případě díry v číslech 6, 8, 9 a 0) není nutné.

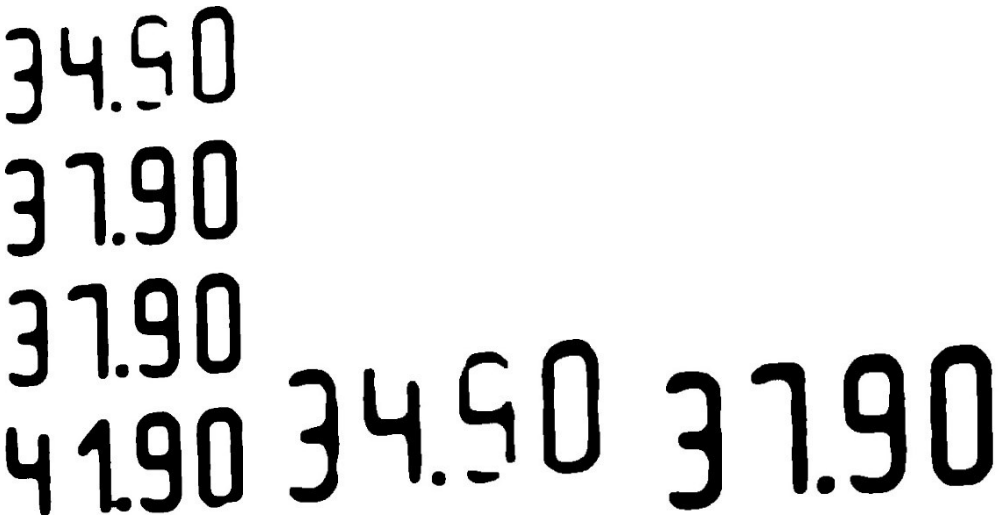
```
var contours = new VectorOfVectorOfPoint();
var hierarchy = new Mat();
CvInvoke.FindContours(erodedImg, contours, hierarchy, RetrType.External,
ChainApproxMethod.ChainApproxSimple);
```

Obrázek 20: Nalezené hrany bílých objektů, znázorněny zelenou barvou

Výřez oblastí s jednotlivými čísly

Po nalezení hran je možné nalézt hraniční body. Ty nalezneme iterací všech nalezených hran a jejich bodů, čímž nalezneme dva body s nejmenším a největším *x* a *y*. Pomocí těchto bodů lze sestavit nejmenší obdélník obléhající všechny bílé pixely. Následně je tento obdélník horizontálně rozdělen na čtyři stejně velké části. Tím jsou získány čtyři obrazy a každý obsahuje právě jedno číslo, které určuje cenu příslušné pohonné hmoty. Mimo to proběhne inverze barev pro spolehlivější zpracování dalším krokem.

```
Rectangle boundingRect = new(minX, minY, width, height);
Mat croppedImage = new(erodedImg, boundingRect);
Mat invertedCropImg = new();
CvInvoke.BitwiseNot(croppedImage, invertedCropImg);

Mat firstNumber = new(invertedCropImg, new Rectangle(0, 0,
invertedCropImg.Width, invertedCropImg.Height / 4));
Mat secondNumber = new(invertedCropImg, new Rectangle(0,
invertedCropImg.Height / 4, invertedCropImg.Width, invertedCropImg.Height /
4));
```



Obrázek 21: Obraz získaný ořízntutím podle nejzazších nalezených hran a jeho rozdělení

OCR

Nakonec jsou jednotlivé obrazy obsahující jedno číslo zpracovány OCR enginem Tessaract. V předchozím kroku byla provedena inverze barev, jelikož podle dokumentace Tessaract spolehlivěji detekuje černý text na bílém pozadí. Jelikož jsou v tomto projektu detekována pouze čísla a žádná písmena, pro datovou sadu používanou enginem je vybrána předtrénovaná datová sada pro anglický jazyk, která je v projektu uložena ve složce `tessdata`. Tato sada je volně dostupná v GitHub repozitáři projektu Tessaract.

Dále je potřeba vybrat segmentační metodu. Výchozí metoda očekává obraz stránky textu, což v případě obrazu jednoho čísla neposkytuje dobé výsledky. Nejlepší výsledky ze všech dostupných metod produkuje *Raw line*, jelikož očekává, že obraz je právě jeden řádek textu bez dalších specifikací.

Nalezené textové řetězce jsou nakonec konvertovány na datový typ `double` a metoda `BenzinaProcess` výsledné hodnoty navrátí v datovém typu `GasStationPrices`.

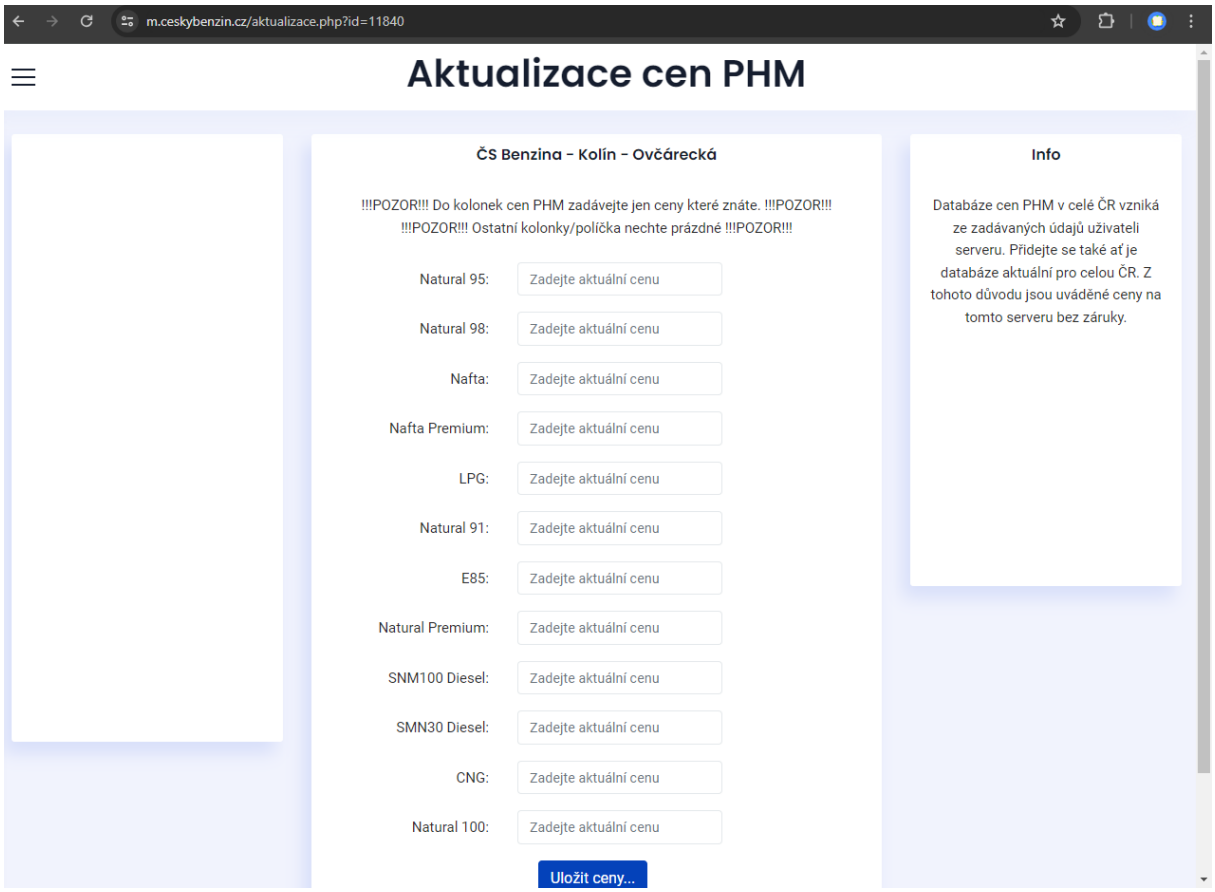
```
var engine = new TesseractEngine("./tessdata", "eng");
var tFirstNumber = Pix.LoadFromFile("./first.jpg");
```

```
var tSecondNumber = Pix.LoadFromFile("./second.jpg");

var page1 = engine.Process(tFirstNumber, PageSegMode.RawLine);
string result1 = page1.GetText();
page1.Dispose();
var page2 = engine.Process(tSecondNumber, PageSegMode.RawLine);
string result2 = page2.GetText();
engine.Dispose();

return new GasStationPrices { StationId = 0, Natural95 =
ConvertToDouble(result2), Diesel = ConvertToDouble(result1) };
```

2.2.2.2 Odeslání na informační portál

Pro odesílání získaných dat je používán portál ceskybenzin.cz. Je databází pohonných hmot v celé České republice, udržovaný od roku 2007 jen na dobrovolných aktualizacích uživatelů. Nalezneme zde jen mapu a výpis čerpacích stanic, jejichž údaje lze aktualizovat formulářem. Jedná se tak a jednoúčelovou webovou aplikaci s jednoduchým uživatelským prostředím, na rozdíl od konkurenčního portálu mbenzin.cz. Ten mimo databázi stanic a jejich cen eviduje také pokuty udělené čerpacím stanicím, srovnání cen STK, ceny paliva v Evropě a také publikuje diskusní články o novinkách.

K odeslání cen pohonných hmot poslouží webová stránka s formulářem na adrese https://m.ceskybenzin.cz/aktualizace.php?id=xxx, kde je xxx nahrazeno unikátním id stanice, která jsou uložena ve vlastní MySql databázi.



Obrázek 22: Stránka s formulářem pro aktualizaci cen na portálu ceskybenzin.cz

Inspekcí webové stránky pomocí vývojářských nástrojů prohlížeče lze najít css skupinu tlačítka pro souhlas se zpracováním osobních údajů (*.fc-button.fc-cta-consent.fc-primary-button*) a id textových polí Natural 95 (*p1*) a Nafta (*p3*). Tyto údaje jsou potřeba pro pozdější automatizaci.

Třída FormFiller ve složce WebAutomation

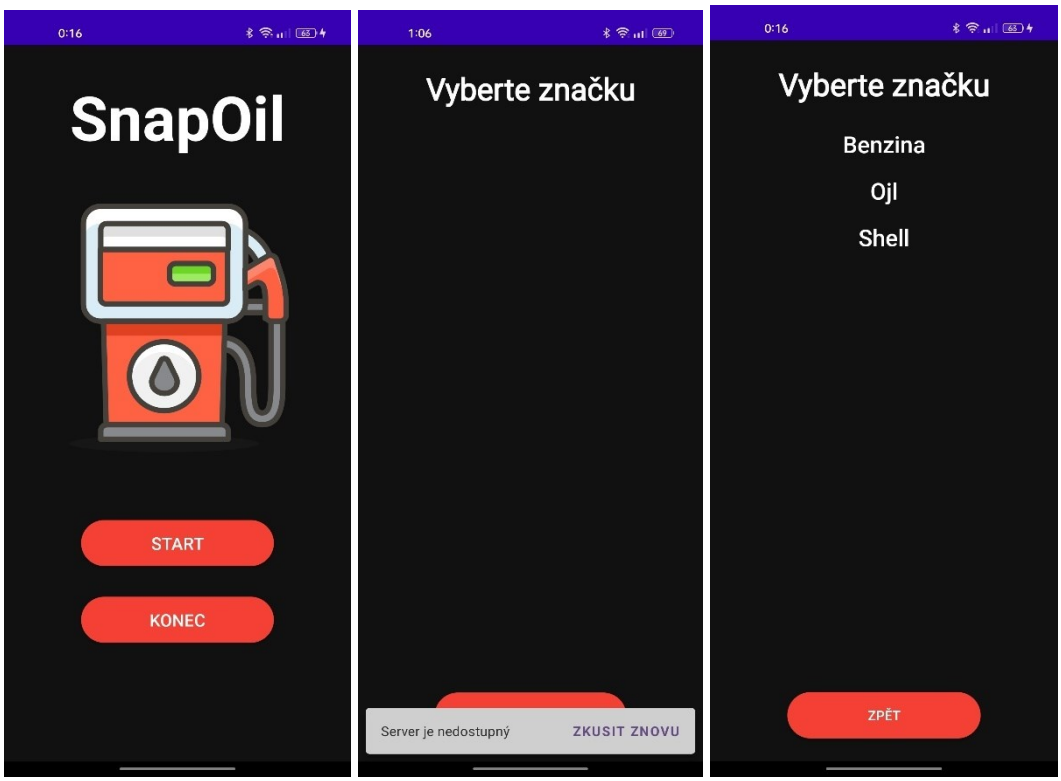
Obsahuje jedinou metodu `SendInfo(GasStationPrices prices)`. Ta vytvoří instanci třídy `ChromeDriver` z knihovny `Selenium`. Driver poté naviguje na stránky s formulářem aktualizace cen podle id stanice z `prices`. Dále odklikne tlačítko *Souhlas* pro povolení používání osobních údajů. Poté nalezne textové pole pro Natural 95, do kterého vyplní data z `prices`. Stejnou operaci provede u textového pole Diesel. Nakonec formulář odešle a zavře driver.

2.2.3 Mobilní aplikace pro Android

Mobilní aplikace je navržena pro obsluhu všech koncových bodů API serverové obslužné aplikace. Aplikace je naprogramována v jazyce Kotlin pro verzi Android 13 (SDK 33) s využitím IDE Android Studio. Používá knihovnu Volley pro HTTP komunikaci, což umožňuje navázání spojení se serverovou obslužnou aplikací. Design aplikace je založen na designovém jazyku *Material Design,* který Google využívá pro své aplikace. Vnitřní struktura aplikace je uzpůsobena návrhovému vzoru MVC (Model-view-controller).

Úvodní obrazovka

Na úvodní obrazovce je zobrazen název aplikace a její logo. Tlačítkem *Start* se pokračuje dále na obrazovku vybrání značky, tlačítkem *Konec* je aplikace ukončena a proběhne přesun na hlavní plochu.



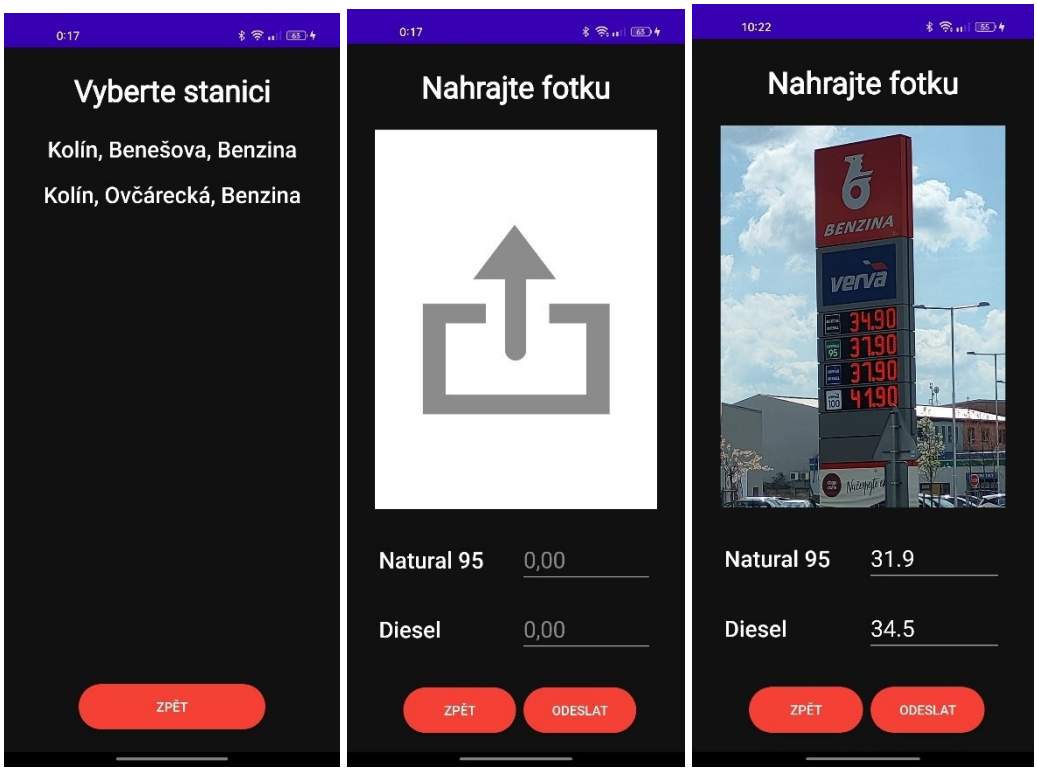
Obrázek 23: Úvodní obrazovka; Výběr značky při nedostupném serveru; Výběr značky

Obrazovka výběru značky

Na této obrazovce je hlavní komponentou `RecyclerView`. Při vytvoření aktivity proběhne vyvolání HTTP požadavku na koncový bod API **/brands** pomocí knihovny Volley. Přijatá data jsou ve formátu JSON a jsou následně převedeny do datové struktury `List<Brand>`. Datový typ `Brand` přesně kopíruje stejnojmenný datový typ v C# aplikaci. Získaný List je poté nastaven jako adaptér `RecyclerView`, čímž se veškeré názvy značek stanic z databáze zobrazí na obrazovce. Kliknutím na název v seznamu proběhne přesměrování na další obrazovku. Pokud se nepodaří připojení k serveru, je na to uživatel upozorněn vyskakující komponentou `Snackbar` v dolní části obrazovky. Kliknutím na *Zkusit znovu* proběhne opakovaný pokus o připojení. Ve spodní části obrazovky se nachází také tlačítko *Zpět,* které přesměruje uživatele na předchozí obrazovku.

Obrazovka výběru stanice

Tato obrazovka funguje podobně jako ta předchozí. Zde je taktéž hlavní komponentou `RecyclerView`. Při vytvoření aktivity opět proběhne vyvolání HTTP požadavku, tentokrát na koncový bod API **/stations/{brand:int}**. Číslo stanice je určeno výběrem značky v předchozí aktivitě. Přijatá data jsou ve formátu JSON a jsou následně převedeny do datové struktury `List<GasStation>`. Datový typ `GasStation` přesně kopíruje stejnojmenný datový typ v C# aplikaci. Získaný List je poté nastaven jako adaptér `RecyclerView`, čímž se adresy stanic vybrané značky z databáze zobrazí na obrazovce. Kliknutím na adresu v seznamu proběhne přesměrování na další obrazovku. Stejně jako v předchozí aktivitě neúspěšné připojení k serveru vyvolá stejnou komponentou `Snackbar` v dolní části obrazovky. I zde se nachází tlačítko *Zpět,* které přesměruje uživatele na předchozí obrazovku, pokud by chtěl změnit výběr značky.

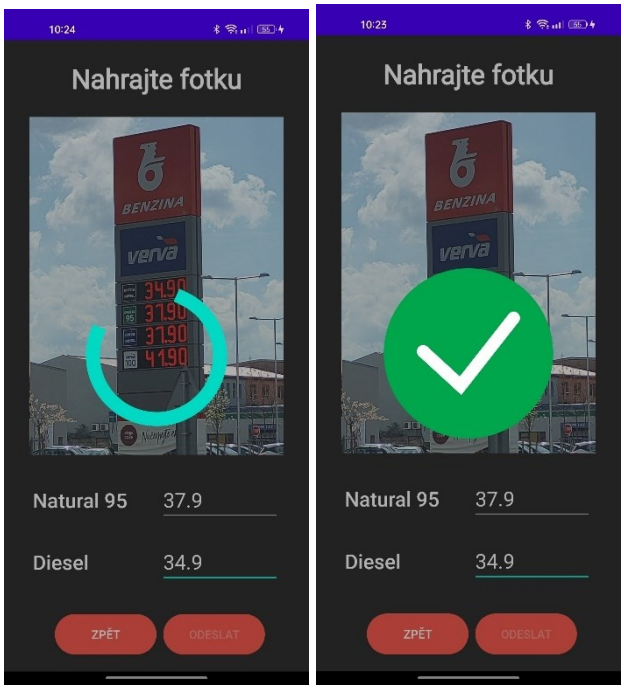


Obrázek 24: Výběr stanice; Nahrání fotky; Obrazovka nahrání fotky po vybrání fotky

Obrazovka nahrání fotografie

Tato obrazovka slouží k nahrání fotografie stojanu čerpací stanice a odeslání informací na informační portál. Kliknutím na logo nahrání uprostřed obrazovky je vyvolána akce otevření galerie mobilního telefonu, výběrem fotografie je uživatel přesměrován zpět na tuto obrazovku. Vybraná fotografie se zobrazí namísto loga nahrání a automaticky proběhne její odeslání ke zpracování vyvoláním HTTP požadavku na koncový bod API **/uploadimage**. Přijatá data ve formátu JSON jsou následně převedena do datového typu `GasStationPrices`. Datový typ `GasStation`Prices přesně kopíruje stejnojmenný datový typ v C# aplikaci. Tato data jsou poté vložena do příslušných textových polí *Natural 95* a *Diesel* pod fotografií. Pokud je server nedostupný, nebo nebyla nalezena žádná data ve fotografii, je na to uživatel upozorněn komponentou `Snackbar` v dolní části obrazovky. Opětovným kliknutím na fotografii lze znovu otevřít galerii a vybrat jinou fotografii ke zpracování.

Následně je možné ručně přepsat hodnoty v textových polích, pokud nebyly detekovány správné ceny. Pod textovými poli se nachází tlačítko *Zpět,* které přesměruje uživatele na předchozí obrazovku, pokud by chtěl změnit výběr stanice. Vedle něho je tlačítko *Odeslat,* jehož stisknutím jsou ceny vyplněné v textových polích společně s id stanice vybrané na předchozí obrazovce převedeny do datového typu `GasStationPrices` a tato data jsou odeslána vyvoláním HTTP požadavku na koncový bod API **/uploadprices**. Pokud je některé z textových polí prázdné, požadavek není odeslán a je na to uživatel upozorněn komponentou `Snackbar` v dolní části obrazovky, která se objeví i pokud je v momentě odeslání server nedostupný. Pokud ale vyvolání http požadavku proběhne, obrazovka lehce zešedne, tlačítko odeslání je vypnuto a uprostřed obrazovky se zobrazí komponenta `ProgressBar`, která provádí animaci po dobu zpracování požadavku. Jakmile je požadavek úspěšně dokončen, `ProgressBar` se změní na zelené logo zaškrtnutí, což značí úspěšné odeslání informací a krátce na to proběhne přesměrování zpět na úvodní obrazovku aplikace.



Obrázek 25: Obrazovka nahrání fotky po stisknutí tlačítka odeslat

2.3 Testování implementované metody pro rozpoznání cen pohonných hmot

Veškeré fotografie pořízené k testování byly pořízeny v průběhu roku 2023 na pobočce sítě čerpacích stanic Benzina nacházející se na adrese Ovčárecká 1459, Kolín 2. K pořízení fotografií byl použit mobilní telefon realme 7 Pro a jeho výchozí aplikace *Fotoaparát*. Použit byl výchozí mód *Fotografie*, bez blesku, s automatickým použitím HDR, ve formátu 4:3. Další nabízené vylepšení pomocí AI nebo filtry nebyly použity, ani žádné další vlastní úpravy po pořízení fotografie.

2.3.1 Galerie testovacích snímků

Snímky byly pořízeny za různých světelných podmínek a podle toho rozděleny do následujících složek:

- **Jasno** – tyto snímky jsou pořízeny v průběhu dne, vždy minimálně dvě hodiny po východu slunce a minimálně dvě hodiny před západem slunce. V těchto snímcích je vždy jasno, nestíněné sluneční paprsky tak dopadají přímo na fotografovaný stojan. Fotografie jsou pořízeny se sluncem v zádech i v čele.
- **Tma** – všechny tyto snímky jsou pořízeny minimálně půl hodiny po západu slunce. V pozadí jsou tak většinou rozsvícena světla pouličního osvětlení a logo Benzina na stojanu je taktéž rozsvíceno
- **Zataženo** – tyto snímky jsou pořízeny v průběhu dne, vždy minimálně dvě hodiny po východu slunce a minimálně dvě hodiny před západem slunce. V těchto snímcích je polojasno, až úplně zataženo, nicméně žádné nestíněné sluneční paprsky nedopadají přímo na fotografovaný stojan.

2.3.2 Postup testování

Pro účely testování byla vytvořena jednoduchá C# aplikace, která obsahuje jedinou třídu obsahující upravenou metodu **BenzinaProcess** ze serverové aplikace. Úprava spočívá v tom, že metoda zpracování obrazu je postupně aplikována na všechny fotografie ve vybrané složce a nalezené hodnoty následně vypisuje do konzole.

2.3.3 Výsledky testování

Výsledky jsou zaneseny do následující tabulky se těmito sloupci:

- **Galerie** – název galerie, ve které proběhlo testování
- **P** – celkový počet testů
- **U** – počet úspěšných testů
- **N** – počet neúspěšných testů
- **Přesnost [%]** – podíl celkového počtu testů a počtu úspěšných testů vyjádřený v procentech

Tabulka 2: Výsledky testování

Galerie	**P**	**U**	**N**	**Přesnost [%]**
Jasno	8	4	4	50 %
Tma	4	0	4	0 %
Zataženo	8	6	2	75 %

2.3.4 Zhodnocení testování

Jasno

Hlavním problémem vyskytujícím se u fotografií pořízených na přímém slunci je neschopnost fotoaparátu pořídit kvalitní fotografii, pokud sluneční paprsky dopadají přímo na segmentový displej ceny (viz obrázek níže). Následkem těchto světelných podmínek tak výchozí rychlost závěrky fotoaparátu nedokáže zachytit celý cyklus obnovy LED displeje. Z výsledné fotografie tak ani lidským okem nelze rozeznat cenu.

Nicméně pokud je za těchto světelných podmínek pořízena fotografie tak, aby bylo slunce v čele, je metoda velmi spolehlivá.

Obrázek 26: Nekvalitní fotografie způsobená přímím slunečním zářením na displej

Tma

Za úplné tmy dochází k rozsvícení dalších světelných prvků stojanu, jako je logo značky a jednotlivých pohonných hmot. Tyto rušivé elementy mají za následek to, že prahováním v této metodě nikdy nezůstanou jen ceny pohonných hmot, ale i některé části loga. Testovaná spolehlivost je nulová, nicméně nebyl pořízen dostatečný počet rozdílných snímků pro potvrzení této skutečnosti.

Zataženo

Tyto světelné podmínky jsou pro metodu jednoznačně nejlepší. Snížené sluneční záření napomáhá výraznosti LED displeje. V tuto denní dobu na rozdíl od doby, kdy je tma, nejsou rozsvíceny další světelné prvky stojanu, které by ovlivňují kvalitu prahování. Jediné neúspěšné snímky byly ty, které byly pořízeny z příliš velké vzdálenosti a rozlišení výřezu s cenami bylo příliš nízké pro spolehlivé rozpoznání.

Závěr

Metoda je nejvíce spolehlivá během dne, kdy nepanují extrémní světelné podmínky jako úplná tma nebo zcela jasná obloha. Dále je nutné vyfotit stojan z co nejbližší možné vzdálenosti, pokud zabírá stojan méně než přibližně čtvrtinu plochy, je jeho samotné rozlišení příliš nízké pro spolehlivé rozpoznání.

Uživatel, který s těmito problémy není obeznámen pravděpodobně nebude schopen pořídit takové snímky, které budou za každé situace dostatečně kvalitní. Nicméně pokud jsou brány v potaz předešlá omezení, je možné pořídit za téměř všech světelných podmínek takový snímek, který má vysokou šanci na úspěch.

ZÁVĚR

V teoretické části byla uvedena krátká historie stojanů čerpacích stanic, po které následoval popis vybraných formátů pro ukládání digitálních fotografií a vybraných barevných prostorů. Zbytek teoretické části se zabývá základními technikami počítačového vidění.

V úvodu experimentální části byly uvedeny použité technologie pro vývoj mobilní aplikace. Ta je rozdělena do dvou částí. Serverová aplikace vytvořena v jazyce C# obstarává obsluhu MySQL databáze. Také obsahuje metodu pro zpracování obrazu stojanu čerpací stanice Benzina za pomoci knihoven Emgu CV a Tessaract. Zajišťuje i odesílání dat na informační portál použitím knihovny Selenium.

Mobilní aplikace pro Android naprogramovaná v jazyce Kotlin poskytuje grafické prostředí pro obsluhu serverové aplikace pomocí API, pro což používá knihovnu Volley.

Aplikace je snadno škálovatelná, jelikož používá SQL databázi pro ukládání čerpacích stanic a přidání další značky vyžaduje jen implementaci příslušné metody rozpoznání obrazu. Rozdělením na dvě části vznikla samostatná serverová aplikace poskytující API. Tím pádem lze snadno vytvářet další aplikace využívající tuto API, například pro mobilní operační systém iOS, nebo webovou aplikaci. Nevýhodou tohoto řešení ale je to, že mobilní aplikace je plně závislá na připojení k serveru a pokud spojení není možné, je tak zcela nepoužitelná.

Metoda rozpoznání obrazu pro síť Benzina je spolehlivá jen za určitých podmínek, nicméně neproběhlo dostatečné testování. To bylo dáno malým množstvím nasbíraných testovacích snímků a také nepříliš proměnlivými cenami pohonných hmot, kvůli čemuž nebylo možné otestovat rozpoznání všech číslic od 0 do 9.

POUŽITÁ LITERATURA

[1] Benzina v proměnách času. In: *Orlen* [online]. 2008 [cit. 2024-05-10]. Dostupné z: https://www.orlen.cz/getmedia/1cd372ac-3d03-4062-8993-77d1ba7dafd3/Benzina-v-promenach-casu_compressed-(1).pdf

[2] What is computer vision? In: *Spiceworks* [online]. 2022 [cit. 2024-05-10]. Dostupné z: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-computer-vision/¨

[3] Image file formats. In: *University of Michigan Library* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://guides.lib.umich.edu/c.php?g=282942&p=1885348

[4] RGB colour model. In: *Encyclopedia Britannica* [online]. 2023 [cit. 2024-05-10]. Dostupné z: https://www.britannica.com/science/RGB-colour-model

[5] Substractive color mixing. In: *L. R. Ingersoll Physics Museum* [online]. 2020 [cit. 2024-05-10]. Dostupné z: https://www.physics.wisc.edu/ingersollmuseum/exhibits/opticscolor/subcolormix/

[6] The CIE XYZ and xyY Color Spaces. In: *Stanford Computer Graphics* [online]. 2010 [cit. 2024-05-10]. Dostupné z: https://graphics.stanford.edu/courses/cs148-10-summer/docs/2010--kerr--cie_xyz.pdf

[7] What is CIELAB? In: *Datacolor* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.datacolor.com/business-solutions/blog/what-is-cielab/

[8] HORÁK, Karel. Jasové transformace. In: *Computer Vision Group VUT* [online]. 2023 [cit. 2024-05-10]. Dostupné z: http://vision.uamt.feec.vutbr.cz/ZVS/lectures/05_Jasove_transformace.pdf

[9] HLAVÁČ, Václav. Předzpracování obrazů v prostoru obrazů, operace v lokálním sousedství. In: *Centrum strojového vnímání ČVUT* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/21ImagPreprocCz.pdf

[10] HLAVÁČ, Václav. Hrany, hranové body a ostření obrazu. In: *Centrum strojového vnímání ČVUT* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://people.ciirc.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/22EdgesInImagesCz.pdf

[11] HLAVÁČ, Václav. Matematická morfologie. In: *Centrum strojového vnímání ČVUT* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://cw.fel.cvut.cz/old/_media/courses/a6m33zmo/71-3matmorpholbincz.pdf

[12] JANÁKOVÁ, Ilona. Segmentace. In: *Machine Vision Group VUT* [online]. 2024 [cit. 2024-05-10]. Dostupné z: http://vision.uamt.feec.vutbr.cz/POV/lectures/05_Segmentace.pdf

[13] What is OCR? In: *Amazon Web Services* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://aws.amazon.com/what-is/ocr/

[14] What is Android? In: *Android* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.android.com/what-is-android/

[15] A tour of the C# language. In: *Microsoft Learn* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[16] Kotlin FAQ. In: *Kotlin* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://kotlinlang.org/docs/faq.html

[17] MySqlConnector. In: *MySqlConnector* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://mysqlconnector.net/

[18] Selenium WebDriver. In: *Selenium WebDriver* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.selenium.dev/documentation/webdriver/

[19] Emgu CV Main Page. In: *Emgu CV* [online]. 2023 [cit. 2024-05-10]. Dostupné z: https://www.emgu.com/wiki/index.php/Main_Page

[20] Tessaract User Manual. In: *Tessaract* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://tesseract-ocr.github.io/tessdoc/

[21] Volley overview. In: *Volley* [online]. 2021 [cit. 2024-05-10]. Dostupné z: https://google.github.io/volley/

[22] What is MySQL? In: *Oracle* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.oracle.com/mysql/what-is-mysql/

[23] What is an IDE? In: *Codecademy* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://www.codecademy.com/article/what-is-an-ide

[24] What is Visual Studio? In: *Microsoft Learn* [online]. 2023 [cit. 2024-05-10]. Dostupné z: https://learn.microsoft.com/en-us/visualstudio/get-started/visual-studio-ide?view=vs-2022

[25] Meet Android Studio. In: *Android Developers* [online]. 2024 [cit. 2024-05-10]. Dostupné z: https://developer.android.com/studio/intro