|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Softwarová čtečka SPZ** | | |
| Adam Hujar | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2021/2022 | |

Prohlašuji, že jsem závěrečnou práci vypracoval samostatně a uvedl veškeré použité   
informační zdroje.

Souhlasím, aby tato studijní práce byla použita k výukovým účelům na Střední průmyslové   
a umělecké škole v Opavě, Praskova 399/8.

V Opavě 31. 12. 2021

*podpis autora práce*

**ANOTACE**

Projekt se zabývá čtečkou státních poznávacích značek (dále už jen SPZ). Kód je napsaný v programovacím jazyce Python a  pomocí funkce EasyOCR, která dokáže rozeznat text na fotce, přečte SPZ, označí ji na obrázku a její text zapíše do souboru. Kód bych chtěl dále upravovat a zkusit ho využít například u brány k příjezdové cestě.

OBSAH

[Úvod 4](#_Toc92263722)

[1 Teoretická a metodická východiska 5](#_Toc92263723)

[1.1 Počáteční stav 5](#_Toc92263724)

[1.2 Využití 5](#_Toc92263725)

[1.3 Běžně používaný způsob 5](#_Toc92263726)

[2 Využité technologie 6](#_Toc92263727)

[2.1 OpenCV 6](#_Toc92263728)

[2.2 imutils 7](#_Toc92263729)

[2.3 EasyOCR 7](#_Toc92263730)

[3 Způsoby řešení a použité postupy 8](#_Toc92263731)

[4 Výsledky řešení, výstupy, uživatelský manuál 9](#_Toc92263732)

[Závěr 10](#_Toc92263733)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 11](#_Toc92263734)

Úvod

Cílem práce bylo vytvořit program, na kterém bych porozuměl základům práce s videem a fotografiemi pomocí programovacích jazyků. S touto částí programování jsem neměl žádnou předešlou zkušenost.

Hlavní motivací, byla možnost dotknout se nového tématu v programování, ke kterému bych se jinak nejspíše nedostal.

# Teoretická a metodická východiska

## Počáteční stav

S Pythonem jako takovým jsem až na malou ukázku ve třetím ročníku, kde jsme pouze zkoušeli lehčí projekty, neměl zkušenosti.

## Využití

Tento kód má, díky dnešní době, ve které je jednoduché získat vozidla, velký potenciál. Většina cest vozidlem by měla mít cíl. Nejčastěji cílem bývá nějaké parkoviště ať pracovní, u supermarketu nebo před domem. A právě zde má můj kód většinu jeho využití. Například databáze pro příjezdy a odjezdy vozů, bezkontaktní otevírání vrat nebo časová kontrola příjezdu a odjezdu z parkovišť u obchodních domů. Samozřejmě toto nejsou jediné možnosti, tento projekt by se také dal použít k odstranění dálničních známek. Vozidlu, které by projelo po dálnici, by byla zkontrolovaná SPZ, zda je nebo není v seznamu vozidel, které mají platnou dálniční známku.

## Běžně používaný způsob

Na počátku při mém prvním průzkumu jsem zjistil, že je určitý druh kamer, který je přímo určený na přečtení SPZ. Zní to skvěle, ale jsou na rozdíl od normálních kamer velmi drahé. Také musíme vzít v potaz, že se kamery mohou zničit, mohou být odcizeny, a také nemusí fungovat podle představ. Takže při použití mého projektu, by firma ušetřila na kamerách a zároveň by mohli velmi jednoduše kód upravit podle jejich potřeb.

# Využité technologie

## Python

Program je napsán v jazyce Python, kvůli jeho jednoduchosti a přehlednosti. Právě díky těmto vlastnostem, jsem měl možnost porozumět každému příkazu.

## OpenCV

OpenCV je otevřená multiplatformní knihovna pro manipulaci s obrazem. Knihovnu je možné využit v C, C++ a Pythonu. V mém kódu je OpenCV například použito pro otevření souboru pro kód.

img=cv2.imread(**'frame%d.jpg'** %number)

Také otevření videa:

video = cv2.VideoCapture(**'video1.mp4'**)

a spousty dalších funkcí na úpravy fotografií například BGR filtr, bilateralální filtr a canny:

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
bfilter = cv2.bilateralFilter(gray, 11, 17, 17)  
edged = cv2.Canny(bfilter, 30, 200)

Tato knihovna je také v mém kódu využita na úpravu fotografie:



BGR a Bilateralní filtr



Canny(detekce rohů)

a také konečné zapsání textu do fotografie.

## Imutils

Knihovna pro manipulaci s obrazem (např. změna velikosti, rotace…). V mém kódu pomáhá na hledání obdélníku, který dále určuje velikost obrázku, ze kterého čte knihovna EasyOCR.

contours = imutils.grab\_contours(keypoints)

## EasyOCR

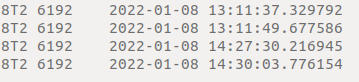
Je knihovna pro čtení textu z obrazu (OCR je zkratka pro Optical Character Recognition). Knihovna se mi zalíbila díky své jednoduchosti. Stačí pouze nainstalovat vyvolat v programu a pak už jen použít její funkce na vyčtení textu z obrazu. Čtení z upraveného obrázku jsem se bál nejvíce, naštěstí mi tato knihovna velmi usnadnila práci.

reader = easyocr.Reader([**'en'**])result = reader.readtext(cropped\_image)

# Způsoby řešení a použité postupy

Jelikož jsem neměl žádné zkušenosti, začal jsem dotazy na pana učitele Grussmanna a hledání na internetu. Můj první pokus začal s programem OpenALPR. Tento kód byl ale bohužel nepřesný, napsán z velké části v C++ a při jeho použití bych se nenaučil vše, co bych chtěl. Takže jsem hledal dále a narazil jsem na spoustu dalších projektů, ale jediný, o kterém bych se chtěl zmínit, je UltimateALPR. Na konec jsem ho nepoužil kvůli zbytečně velkému obsahu funkcí, který celý kód dělají extrémně složitý a tedy znovu nesmyslný pro mé použití. Takže můj finální projekt je založen na kódu od Nicholas Renotte ANPR s použitím pythonu. Kód je velmi jednoduchý, skvěle popsaný a dobrý základ na jakoukoliv čtečku SPZ.

Dalším krokem tedy bylo stažení tohoto kódu, jeho porozumění a následující přepisování na výpis do souborů pro jednodušší kontrolu funkčnosti.

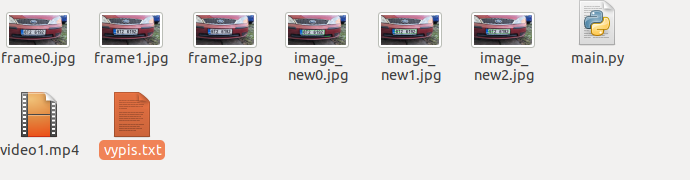


Kód jsem poté upravil, tak aby zapisoval do textového souboru, v jaký čas a co přečetl.

Následovalo dlouhé listování internet za snahou pochopit, jak převést daný kód, tak, aby použil jen určité snímky z videa. A to se mi na poslední chvíli podařilo. Kód si otevře video, ze kterého si udělá snímky a s nimi poté pracuje dále stejně jako s fotografiemi.

# Výsledky řešení, výstupy, uživatelský manuál

Výsledkem projektu je tedy kód, který použije určité snímky z videa a upraví je do podoby, ze které dokáže EasyOCR přečíst text. Upravený obrázek a textový soubor si uloží do kódem nastavené složky.



Soubory frame.jpg jsou snímky z videa a image\_new.jpg jsou už upravené fotografie.

Program bohužel v tuto chvíli není schopen použít živé vysílání.

# Závěr

Projekt tedy umí číst SPZ z videa, dokáže ji zapsat do souborů a díky tomu je připraven na lehké připojení dalších funkcí nebo databáze.

Kód sice není naprosto dle mých představ, ale hlavní cíl splnil, spoustu jsem se z něj naučil a další projekt, už bude mnohem lepší.

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] Python ANPR with OpenCV and EasyOCR in 25 Minutes | Automatic Number Plate Recognition Tutorial [online].

Dostupné z: <<https://www.youtube.com/watch?v=NApYP_5wlKY&t=610s>>

 [2] W3Schools: Online Web Tutorials [online].

Dostupné z: <<https://www.w3schools.com>>

 [3] [python read a video frame by frame Code Example – Code Grepper](https://www.codegrepper.com/code-examples/python/python+read+a+video+frame+by+frame) [online]

Dostupné z: <<https://www.codegrepper.com/code-examples/python/python+read+a+video+frame+by+frame>>

 [4] [GitHub: Where the world builds software](https://github.com/) [online]

Dostupné z: <<https://github.com/>>

 [5] [Stack Overflow - Where Developers Learn, Share, & Build](https://stackoverflow.com/) [online]

Dostupné z: <<https://stackoverflow.com/>>

[6] Python – Extracting and Saving Video Frames – [Stack Overflow - Where Developers Learn, Share, & Build](https://stackoverflow.com/) [online]

Dostupné z: <<https://stackoverflow.com/questions/33311153/python-extracting-and-saving-video-frames>>

[7] Cutting Video on frames, save only every 3rd? - [Stack Overflow – Where Developers Learn, Share, & Build](https://stackoverflow.com/) [online]

Dostupné z: <https://stackoverflow.com/questions/43197892/cutting-video-on-frames-save-only-every-3rd>