|  |  |  |
| --- | --- | --- |
|  | | |
| **Závěrečná studijní práce**  **dokumentace** | | |
| **Silniční radar** | | |
| Jáchym Václav Hanke | | |
|  | | |
|  | |  |
| **Obor:** | 18-20-M/01 INFORMAČNÍ TECHNOLOGIE  se zaměřením na počítačové sítě a programování | |
| **Třída:**  **Školní rok:** | IT4  2024/2025 | |

**Poděkování**

*Rád bych poděkoval panu doc. Ing. Petru Čermákovi, Ph.D. za jeho rady a podporu během zpracování této práce. Dále chci poděkovat pánům učitelům Mgr. Marku Lučnému a Ing. Petru Grussmannovi za jejich rady a připomínky spojené s touto prací.*

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. 2024

*podpis autora práce*

**ABSTRAKT**

Tento projekt slouží jako nástroj pro detekci a měření rychlosti vozidel s využitím technologie počítačového vidění (OpenCV). Systém identifikuje projíždějící automobily, vizuálně je označuje a sleduje jejich pohyb. Pro výpočet rychlosti využívá dvě referenční čáry umístěné na záznamu, mezi nimiž měří čas průjezdu vozidla. Na základě této doby a známé vzdálenosti mezi čárami je spočítána okamžitá rychlost. Výsledná data jsou následně ukládána do souboru pro další zpracování. Tento program může sloužit k monitorování dopravy nebo analýze průjezdu vozidel.

**ABSTRACT**

This project serves as a tool for detecting and measuring the speed of vehicles using computer vision technology (OpenCV). The system identifies passing cars, visually tags them and tracks their movements. To calculate the speed, it uses two reference lines placed on the record, between which it measures the time of the vehicle's passage. Based on this time and the known distance between the lines, the instantaneous speed is calculated. The resulting data is then stored in a file for further processing. This program can be used for traffic monitoring or vehicle passage analysis.

OBSAH

[Úvod 5](#_Toc187612483)

[1 Teoretická a metodická východiska 6](#_Toc187612484)

[1.1 Principy počítačového vidění (OpenCV) 6](#_Toc187612485)

[1.2 Měření času a vzdálenosti 6](#_Toc187612486)

[1.3 Výpočet rychlosti 6](#_Toc187612487)

[1.4 Ukládání a správa dat 7](#_Toc187612488)

[1.5 Výběr a nastavení technického vybavení 7](#_Toc187612489)

[1.6 Metodologie testování a validace 7](#_Toc187612490)

[1.7 Možnosti rozšíření a budoucí směřování 8](#_Toc187612491)

[2 Využité technologie 9](#_Toc187612492)

[2.1 OpenCV (Open Source Computer Vision Library) 9](#_Toc187612493)

[2.2 Programovací jazyk Python 9](#_Toc187612494)

[2.3 Hardware 10](#_Toc187612495)

[2.3.1 Kamera 10](#_Toc187612496)

[2.3.2 Výpočetní zařízení 10](#_Toc187612497)

[3 Způsoby řešení a použité postupy 11](#_Toc187612498)

[3.1 Příprava videa k měření 11](#_Toc187612499)

[3.2 Inicializace prostředí 11](#_Toc187612500)

[3.3 Architektura programu 12](#_Toc187612501)

[3.3.1 Import knihoven 12](#_Toc187612502)

[3.3.2 Inicializace proměnných 12](#_Toc187612503)

[3.3.3 Detekce měřitelného auta 12](#_Toc187612504)

[3.3.4 Měření času průjezdu 13](#_Toc187612505)

[3.3.5 Výpočet rychlosti 14](#_Toc187612506)

[3.3.6 Zápis do souboru 14](#_Toc187612507)

[3.4 Testování systému 14](#_Toc187612508)

[4 Výsledky řešení, výstupy, uživatelský manuál 15](#_Toc187612509)

[4.1 Splněné cíle 15](#_Toc187612510)

[4.2 Nesplněné cíle 15](#_Toc187612511)

[4.3 Uživatelský manuál 15](#_Toc187612512)

[4.4 Zhodnocení práce 16](#_Toc187612513)

[Závěr 17](#_Toc187612514)

[Seznam použitýCH INFORMAČNÍCH ZDROJů 18](#_Toc187612515)

Úvod

Dopravní systémy a jejich efektivní monitorování hrají klíčovou roli v zajištění bezpečnosti na silnicích. Jedním z klíčových aspektů je sledování rychlosti projíždějících vozidel, což umožňuje identifikovat případy překročení povolených limitů a přispívá k prevenci dopravních nehod.

Tento projekt se zaměřuje na vývoj jednoduchého a efektivního silničního radaru využívajícího moderní technologie počítačového vidění (OpenCV). Program dokáže detekovat projíždějící automobily, sledovat jejich pohyb a analyzovat rychlost pomocí dvou referenčních čar. Na základě času průjezdu mezi těmito čárami a jejich vzájemné vzdálenosti se vypočítá aktuální rychlost vozidla. Naměřená data jsou poté ukládána do souboru pro další zpracování či analýzu. Tento systém nabízí snadno nasaditelné a ekonomicky dostupné řešení pro měření rychlosti vozidel, které může najít uplatnění v oblastech, jako je monitorování dopravy, analýza dopravního toku nebo zajištění dodržování dopravních předpisů.

Práce je dělena do čtyř kapitol. Nejdříve je teorie celého projektu, následují využité technologie, dále postupy a architektura projektu a nakonec splněné cíle, nesplněné cíle, manuál a zhodnocení práce.

# Teoretická a metodická východiska

## Principy počítačového vidění (OpenCV)

Počítačové vidění je obor, který se zaměřuje na získávání, zpracování a interpretaci obrazových dat z digitálních zdrojů, jako jsou kamery. V projektu je klíčové:

* **Detekce objektů:** Identifikace automobilů v obraze pomocí algoritmů, jako je detekce hran, rozdíl mezi snímky (background subtraction) nebo pokročilejší metody (např. Haar cascades, HOG, nebo neuronové sítě jako YOLO).
* **Sledování objektů:** Po detekci je důležité sledovat pohyb objektů v následujících snímcích. To může být realizováno metodami jako Optical Flow nebo jednoduchým propojením pozic detekovaných objektů v časové sekvenci.
* **OpenCV:** OpenCV poskytuje širokou škálu nástrojů pro zpracování obrazu, jako je detekce hran (Canny), označování oblastí zájmu nebo aplikace geometrických transformací.

## Měření času a vzdálenosti

Pro měření rychlosti vozidla je nezbytné přesně určit:

* **Čas průjezdu:** Využití časových razítek z videozáznamu nebo snímků pro určení doby, kterou vozidlo potřebovalo k překonání vzdálenosti mezi dvěma referenčními čárami.
* **Vzdálenost:** Kalibrace obrazu, aby reálná vzdálenost mezi čárami odpovídala vzdálenosti v pixelech. To zahrnuje použití známého měřítka (např. měřením objektu v obraze) nebo použití standardních poměrů.
* **Přesnost:** Zajištění minimálních chyb při měření vzdálenosti a času, například kontrolou snímkové frekvence a správného umístění čar.

## Výpočet rychlosti

Výpočet rychlosti vozidla vychází z jednoduchého fyzikálního vzorce: **v = s / t**

* **Proměnné:** s (vzdálenost) je kalibrovaná vzdálenost mezi čárami a t (čas) je změřená doba, kterou vozidlo potřebovalo na její překonání.
* **Přesnost výpočtu:** Přesnost měření času je klíčová, protože drobné odchylky mohou významně ovlivnit výslednou rychlost. Důležitá je i dostatečně vysoká snímková frekvence kamery.
* **Jednotky:** Výsledky se obvykle převádějí do standardních jednotek, jako jsou km/h nebo m/s.

## Ukládání a správa dat

Naměřené hodnoty, jako je čas průjezdu, rychlost vozidla a další údaje, se ukládají pro další zpracování.

* **Struktura souboru:** Obvykle se používají jednoduché formáty, jako je CSV nebo txt, které umožňují snadnou čitelnost a kompatibilitu s jinými programy.
* **Obsah:** Každý záznam obsahuje čas průjezdu, identifikaci vozidla (pokud je k dispozici), rychlost a případně další údaje.
* **Dlouhodobé ukládání:** Data mohou být ukládána lokálně nebo na vzdáleném serveru pro další analýzu.

## Výběr a nastavení technického vybavení

Kvalita technického vybavení výrazně ovlivňuje výkon systému:

* **Kamera:** Musí mít dostatečné rozlišení a snímkovou frekvenci pro přesnou detekci a měření.
* **Umístění:** Kamera by měla být umístěna tak, aby měla dobrý výhled na silnici, ideálně z výšky nebo pod správným úhlem.
* **Podmínky prostředí:** Systém by měl být schopen fungovat za různých světelných a povětrnostních podmínek.

## Metodologie testování a validace

Před nasazením je nutné systém důkladně otestovat:

* **Testovací scénáře:** Ověření funkčnosti systému při různých rychlostech, vzdálenostech a podmínkách.
* **Kalibrace:** Měření přesné vzdálenosti mezi čárami a kontrola správnosti výpočtů.
* **Porovnání s reálnými daty:** Validace naměřených hodnot s údaji ze spolehlivých zdrojů, například pomocí radarového měřiče rychlosti.

## Možnosti rozšíření a budoucí směřování

Do budoucna lze systém dále vylepšovat:

* **Rozpoznávání SPZ:** Integrace technologie OCR (optické rozpoznávání znaků) pro identifikaci vozidel.
* **Klasifikace vozidel:** Identifikace typů vozidel (osobní auta, nákladní auta, motocykly) pomocí strojového učení.
* **Zpracování v reálném čase:** Optimalizace výkonu pro okamžité měření a ukládání dat.
* **Analýza dopravy:** Rozšíření funkcionality o měření hustoty dopravy, průměrné rychlosti nebo dalších parametrů dopravního toku.

# Využité technologie

## OpenCV (Open Source Computer Vision Library)

OpenCV je jedna z nejpopulárnějších a nejvíce používaných knihoven pro počítačové vidění. V projektu slouží jako klíčový nástroj pro zpracování obrazových dat a implementaci funkcí detekce a sledování objektů. Hlavní vlastnosti a funkce využité v projektu:

* **Detekce objektů:** OpenCV umožňuje detekci vozidel v obraze pomocí algoritmů jako background subtraction (rozdíl mezi snímky) nebo pokročilejších metod, jako je HOG (Histogram of Oriented Gradients).
* **Sledování objektů:** Knihovna poskytuje nástroje pro sledování pohybu detekovaných objektů v čase. V projektu může být použito Optical Flow nebo jednoduché propojení detekcí v jednotlivých snímcích.
* **Zpracování obrazu:** Základní operace, jako je převod obrazu do odstínů šedi, rozmazání obrazu (GaussianBlur) pro redukci šumu, nebo detekce hran (Canny edge detection), pomáhají zvýšit přesnost detekce vozidel.
* **Snadná integrace:** OpenCV je dobře dokumentováno, podporuje více programovacích jazyků a díky Python API umožňuje rychlý vývoj i testování.

## Programovací jazyk Python

Python byl vybrán jako hlavní programovací jazyk projektu díky své jednoduchosti, široké podpoře knihoven a silné komunitě:

* **Jednoduchost a čitelnost:** Python je díky své srozumitelné syntaxi ideální pro rychlý vývoj a snadné ladění aplikací.
* **Široká škála knihoven:** Python poskytuje mnoho knihoven, které podporují počítačové vidění (OpenCV), práci s daty (NumPy, Pandas) a vizualizaci (Matplotlib, Seaborn).
* **Podpora práce s daty:** NumPy umožňuje efektivní manipulaci s maticemi a vektory, což je klíčové při zpracování obrazu.
* **Rozšiřitelnost:** Python umožňuje snadné přidání dalších funkcí, jako je rozpoznávání SPZ nebo integrace s databázemi a cloudovými službami.

## Hardware

Hardware hraje klíčovou roli při zajištění správné funkce a přesnosti systému. V projektu byly využity následující komponenty:

### Kamera

* **Rozlišení:** Kvalitní rozlišení (např. Full HD) je nezbytné pro přesnou detekci objektů a kalibraci vzdálenosti.
* **Snímková frekvence:** Vyšší snímková frekvence (např. 30 fps) umožňuje přesnější měření času průjezdu vozidel.
* **Umístění:** Kamera by měla být umístěna tak, aby měla dobrý přehled o sledované oblasti, ideálně s výhledem na celou šířku silnice.
* **Použití:** Použil jsem vlastní fotoaparát Canon EOS 7D

### Výpočetní zařízení

* **PC nebo notebook:** Pro zpracování obrazu a výpočet rychlosti v reálném čase.
* **Raspberry Pi nebo podobná zařízení:** Pro použití v kompaktnějších a přenosnějších aplikacích. Raspberry Pi je oblíbené pro svou nízkou cenu, energetickou nenáročnost a dostatečný výkon pro základní zpracování obrazu.
* **Osvětlení:** V případě horších světelných podmínek může být nutné použít přídavné osvětlení (např. infračervené světlo), aby byl obraz dobře čitelný za všech okolností.

# Způsoby řešení a použité postupy

K finalizaci tohoto projektu vedlo několik kroků:

## Příprava videa k měření

Použil jsem svoji digitální zrdcadlovku Canon EOS 7D v kombinaci s teleobjektivem a umístil jsem ji z výšky směrem na silnici. Dále jsem použil svoje auto a natočil průjezdy s rychlostí 50 km/h a 60 km/h. Po drobné úpravě videa jsem měl připravené testovací video a mohl jsem přejít k dalšímu kroku.

Ukázka průjezdu:

Obsah obrázku vozidlo, Pozemní vozidlo, venku, pneumatika

Popis byl vytvořen automatickyObsah obrázku vozidlo, Pozemní vozidlo, venku, silnice

Popis byl vytvořen automatickyObsah obrázku vozidlo, Pozemní vozidlo, venku, kolo

Popis byl vytvořen automaticky

## Inicializace prostředí

Použil jsem vývojové prostředí Visual Studio Code. Následovalo pár kroků:

* Vytvoření projektové složky
* **Inicializace enviromentu:** Program jsem spouštěl ve virtuálním enviromentu od Pythonu
* **Instalace potřebných knihoven:** Byly použity knihovny: opencv-python, numpy, time
* Vytvoření samotného programu

## Architektura programu

### Import knihoven

*import cv2*

*import numpy as np*

*import time*

### Inicializace proměnných

Inicializace videa (popř. kamery), oddělovače pozadí, proměnných na rychlost a detekci, souřadnice a vzdálenost referenčních čar a otevření souboru k zápisu:

*# Načtení videa*

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

*# Detekce pohybu pomocí modelu pozadí MOG2.*

*fgbg = cv2.createBackgroundSubtractorMOG2()*

*# Inicializace proměnných*

*speed = 0*

*max\_speed = 50*

*center\_x = 0*

*center\_y = 0*

*start\_time = None*

*elapsed\_time = None*

*license\_plate = None*

*# Nastavení pozic čar v pixelech a vzdálenosti mezi nimi v metrech*

*line1\_y = 460*

*line2\_y = 860*

*lines\_distance = 25*

*# Inicializace souboru (vymaže obsah, pokud soubor již existuje)*

*with open("rychlosti.txt", "w") as file:*

*file.write("Rychlosti:\n")*

### Detekce měřitelného auta

Detekci jsem udělal pomocí oddělovače pozadí a následného filtrování kontur (obrysů objektů). Podle vyfiltrovaných kontur označí program auto obdélníkem a do středu umístí červenou tečku určenou k následné detekci pomocí referenčních čar.

*while cap.isOpened():*

*ret, frame = cap.read()*

*if not ret:*

*break*

*fgmask = fgbg.apply(frame)*

*\_, thresh = cv2.threshold(fgmask, 200, 255, cv2.THRESH\_BINARY)*

*contours, \_ = cv2.findContours(thresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)*

*if contours:*

*# Funkce na najdení největší kontury*

*largest\_contour = max(contours, key=cv2.contourArea)*

*if cv2.contourArea(largest\_contour) > 1000: # Odfiltrování šumu*

*x, y, w, h = cv2.boundingRect(largest\_contour)*

*cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)*

*center\_x = x + w // 2*

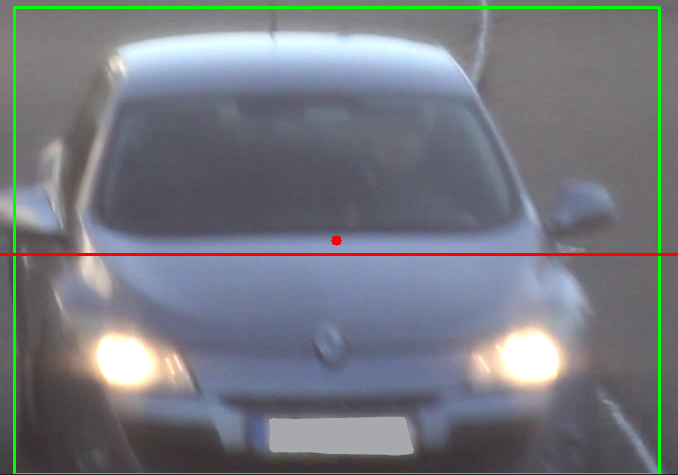
*center\_y = y + h // 2*

*current\_position = (center\_x, center\_y)*

*current\_time = time.time()*

*# Zaznačení středu detekovaného auta*

*cv2.circle(frame, (center\_x, center\_y), 5, (0, 0, 255), -1)*



### Měření času průjezdu

Měření času jsem udělal pomocí dvou referenčních čar umístěných na dvou kanálech na silnici a pomocí map jsem změřil vzdálenost. Samotné měření probíhalo tak, že když se středový bod auta dotkl první čáry, spustil se časovač. Při dotyku s druhou čárou se časovač zastavil a čas zůstal v proměnné.

*# Zjištění jestli se auto dotýká první čáry*

*if center\_y >= line1\_y and start\_time is None:*

*start\_time = current\_time*

*# Zjištění jestli se auto dotýká druhé čáry*

*if center\_y >= line2\_y and start\_time is not None:*

*elapsed\_time = current\_time - start\_time*

*start\_time = None # Resetování času pro další měření*

### Výpočet rychlosti

Rychlost jsem počítal pomocí změřené vzdálenosti mezi čárami a času průjezdu a následně zapsal rychlost do souboru včetně možného překročení rychlosti.

*if elapsed\_time is not None and elapsed\_time > 0:*

*# Výpočet rychlosti*

*speed = (lines\_distance / elapsed\_time) \* 3.6*

*license\_plate = detect\_license(frame)*

*if speed < max\_speed:*

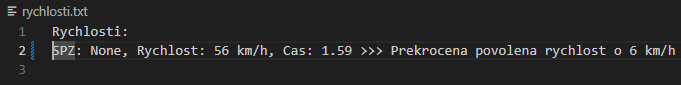
*write\_line(f'SPZ: {license\_plate}, Rychlost: {int(speed)} km/h, Cas: {elapsed\_time:.2f}')*

*else:*

*write\_line(f'SPZ: {license\_plate}, Rychlost: {int(speed)} km/h, Cas: {elapsed\_time:.2f} >>> Prekrocena povolena rychlost o {int(speed)-max\_speed} km/h')*

### Zápis do souboru

Data jsou zapisována do textového souboru. Každý zápis obsahuje SPZ (zatím nefunguje), rychlost, čas průjezdu, popř. jestli překročil povolenou rychlost a o kolik ji překročil.



## Testování systému

Systém jsem nastavil, aby co nejvíce odpovídal reálné vzdálenosti na silnici a otestoval jsem ho s různými rychlostmi vozidel.

# Výsledky řešení, výstupy, uživatelský manuál

## Splněné cíle

* **Detekce auta z videa**
  + Program je schopen detekovat, označit a následovat auto na videu.
* **Kalibrace**
* Podařilo se mi správně kalibrovat program na co nejpřesnější měření času průjezdu.
* **Měření rychlosti**
  + Program je schopen změřit rychlost průjezdu na přibližně +/-5 km/h, záleží na okolnostech měření.

## Nesplněné cíle

* **Rozpoznání SPZ**
  + Přesto, že jsem vytvořil funkci na rozpoznání SPZ, tak se mi ji nepodařilo dostat do funkčního stavu.
* **Integrace knihovny SIFT**
  + Tato knihovna by byla schopná přesnější detekce a měření rychlosti, ale nebyl jsem schopen ji správně zapojit do projektu.
* **Integrace GUI**
  + Byl jsem rád, že se mi podařilo zprovoznit projekt a na GUI jsem neměl moc času

## Uživatelský manuál

1. **Instalace:**

* Otevřete složku s programem v příkazovém řádku nebo ve vámi vybraném programu, jako například VSCode.
* Nainstalujte Python na vaše zařízení.
* V projektové složce vytvořte pomocí Pythonu virtuální enviroment pomocí příkazu: *python -m venv .venv*
* Aktivujte enviroment pomocí příkazu:
  + Linux: *. .venv/bin/activate*
  + Windows: *.venv\Scripts\activate*
* Instalace knihoven:
  + *pip install opencv-python easyocr*

1. **Kalibrace:**
   * V souboru radar.py upravte následující údaje podle potřeby:

* line1\_y, line2\_y – zde napište počty pixelů shora obrazovky, kde budou následně vytvořeny referenční čáry na měření rychlosti (na zobrazení pozic čar budete muset spustit program).
* lines\_distance – zde napište vzdálenost v metrech mezi referenčními čárami
* cap = cv2.VideoCapture(\*) – místo \* napište buď 0 pro záznam z kamery nebo název vašeho videa do jednoduchých uvozovek.

1. **Spuštění**
   * Program spustíte pomocí příkazu: *python radar.py*

## Zhodnocení práce

Projekt dokázal úspěšně detekovat a změřit rychlost projíždějícího auta a zapsat data do souboru. Projekt lehce závisí na výkonu zařízení uživatele. Nižší výkon může znamenat horší a pomalejší výsledky měření.

# Závěr

Projekt silničního radaru přinesl komplexní řešení pro detekci a měření rychlosti vozidel pomocí moderních technologií počítačového vidění. Díky využití knihovny OpenCV a programovacího jazyka Python bylo možné vytvořit efektivní systém, který nejen rozpozná vozidla, ale také přesně vypočítá jejich rychlost na základě měření času mezi dvěma referenčními body.

Projekt se zaměřil na dosažení co nejvyšší přesnosti prostřednictvím kalibrace vzdálenosti mezi čarami a optimalizace algoritmů detekce a sledování. Výstupy systému jsou ukládány ve strukturovaném formátu, což usnadňuje další analýzu nebo vizualizaci naměřených dat. Celý proces byl průběžně testován a validován v různých podmínkách, což zajistilo spolehlivost výsledků.

Co se týče budoucího vývoje by se dala zprovoznit funkce na detekci SPZ, přesnější měření pomocí funkcí SIFT nebo přidání GUI.

Závěrem lze říci, že tento projekt představuje jednoduchý, ale účinný nástroj pro měření rychlosti vozidel, který může najít uplatnění v různých dopravních aplikacích, od analýzy dopravy až po její řízení. Další rozšíření systému by mohlo zahrnovat například rozpoznávání registračních značek nebo pokročilejší metody pro klasifikaci vozidel.

Tento projekt tak demonstruje praktické využití moderních technologií pro zlepšení bezpečnosti a efektivity dopravy.

Seznam použitýCH INFORMAČNÍCH ZDROJů

[1] OpenAI. ChatGPT [online]. Poslední revize neuvedena [cit. 2025-01-12].

<https://openai.com/chatgpt>

[2] Stack Overflow [online]. Poslední revize neuvedena [cit. 2025-01-12].

<https://stackoverflow.com>

[3] GitHub [online]. Poslední revize neuvedena [cit. 2025-01-12].

<https://github.com>

[4] YouTube [online]. Poslední revize neuvedena [cit. 2025-01-12].

<https://www.youtube.com>