Inteligentny System Analizy Zagrożeń na Drodze w Czasie Rzeczywistym

Koncepcja Pracy Magisterskiej

Autor: Jakub Jurzak

27 listopada 2025

Rozdział 1

1.1 System bedzie detektować zagrożenia na drodze w czasie rzeczywistym poprzez:

1. **4 specjalistyczne modele YOLOv8n** (samochody, światła, osoby, przeszkody-/rowery),
2. **Równoległe wykonanie na osobnych watkach** ($\approx$ 15ms per frame jeśli byśmy to robili szeregowo to $\approx$ 60ms bo sumujemy),
3. **Transformer-based fusion** do agregacji detekcji,
4. **Ego-motion integration** z `egomotion.parquet` (predkość, przyspieszenie, krzywizny),
5. **Dynamiczne ocenanie zagrożeń** (confidence $\times$ zone $\times$ class $\times$ speed $\times$ maneuver),
6. **5 adaptacyjnych stref zagrożeń** (CRITICAL/WARNING/PERIPHERAL/TOP/SIDE),
7. **Generowanie alertów** (SAFE/WARNING/CRITICAL) z wizualizacja w czasie rzeczywistym.

Parametry Systemu

Parametr	**Wartość**
Rozdzielczość wejściowa	1920 $\times$ 1080 (FHD)
Rozdzielczość przetwarzania	1280 $\times$ 720 (HD)
Spodziewane opóźnienie	$\approx$ 28–32 ms
FPS output	$\approx$ 31–35 FPS
VRAM wymagany	$\approx$ 7–8 GB

Rozdział 2

2.1 Dlaczego 4 Specjalistyczne Modele (zamiast 1 Multi-Class)?

Aspekt	**Multi-Class**	**4 Specjalistyczne**
Opóźnienie	$\approx$ 25ms	$\approx$ 28–32 ms (parallel)
Accuracy	0.68 mAP	0.73–0.75 mAP

2.2 Dlaczego równoległe wykorzystanie watków (zamiast sekwencyjnego)?

Sekwencyjnie:

$$\text{Model1} \rightarrow \text{Model2} \rightarrow \text{Model3} \rightarrow \text{Model4} = 60 \text{ ms} \quad (\textbf{ za wolno}) \tag{2.1}$$

Równolegle:

$$4 \text{ modele jednocześnie} = 15 \text{ ms} \quad (\checkmark) \tag{2.2}$$

Wielowatkowość zapewnia, że 4 modele pracuja **jednocześnie**, osiagajac opóźnienie prawie identyczna jak single model multiclass.

2.3 Dlaczego i po co Ego-Motion Integration?

Bez ego-motion:

$$\text{threat\_score} = \text{confidence} \times \text{zone} \times \text{class} \tag{2.3}$$

Z ego-motion:

$$\text{threat\_score} = \text{confidence} \times \text{zone} \times \text{class} \times \text{speed\_mult} \times \text{maneuver\_mult} \tag{2.4}$$

Przykład: Pieszy w strefie krytycznej

- Pojazd 10 km/h: WARNING (spokojnie)
- Pojazd 80 km/h + hamowanie: CRITICAL (ALARM!)

2.4 Wyniku Treningu: 1280×720 (HD)

Aspekt	**Wartość**	**Uwagi**
640×480	2–3 h treningu, ≈ 15 ms inference	0.62 mAP
1280×720	**4–5 h treningu, 20–25 ms inference**	**0.71 mAP**
1920×1080	8–10 h treningu, 45–60 ms inference	0.76 mAP

Rozdział 3

Architektura Systemu

3.1 Równania Threat Scoring

3.1.1 Threat Score - Równanie Podstawowe

$$\text{threat\_score} = \text{conf} \times m_{\text{zone}} \times m_{\text{class}} \times m_{\text{speed}} \times m_{\text{maneuver}} \tag{3.1}$$

gdzie:

$$\begin{aligned}
\text{conf} &\in [0, 1] \quad \text{(detection confidence)} \\
m_{\text{zone}} &\in \{0.8, 1.1, 1.2, 1.3, 1.5\} \quad \text{(zone multiplier)} \\
m_{\text{class}} &\in [0.5, 1.2] \quad \text{(class multiplier)} \\
m_{\text{speed}} &= 1.0 + \min\left(\frac{v_{\text{kmh}}}{30}, 1.0\right) \times 0.5 \quad \text{(speed multiplier)} \\
m_{\text{maneuver}} &\in [1.0, 1.2] \quad \text{(maneuver multiplier)}
\end{aligned}$$

3.1.2 Speed Multiplier (z Ego-Motion)

$$m_{\text{speed}} = 1.0 + \min\left(\frac{\sqrt{v_x^2 + v_y^2 + v_z^2} \times 3.6}{30}, 1.0\right) \times 0.5 \tag{3.2}$$

Interpretacja: przy wyższej predkości, system jest bardziej czuły (max +50% przy ≥ 30 km/h).

3.1.3 Maneuver Multiplier (z Ego-Motion)

$$m_{\text{maneuver}} = \begin{cases} 1.2 & \text{if } a_x < -0.5 \text{ m/s}^2 \text{ (hamowanie)} \\ 1.15 & \text{if } |\text{curvature}| > 0.01 \text{ (ostry skret)} \\ 1.0 & \text{inaczej} \end{cases} \tag{3.3}$$

3.1.4 Alert Logic

$$\text{ALERT} = \begin{cases} \text{CRITICAL (RED)} & \text{if } \max(\text{threat\_scores}) \geq 0.8 \\ \text{WARNING (YELLOW)} & \text{if } 0.6 \leq \max(\text{threat\_scores}) < 0.8 \\ \text{SAFE (GREEN)} & \text{if } \max(\text{threat\_scores}) < 0.6 \end{cases} \tag{3.4}$$

Rozdział 4

Analiza synchronizacji

4.1 linia czasu per klatka (30 fps wejście)

Czas	**Etap**	**Czas trwania w ms.**
$t = 0$ ms	Przechwycenie klatki (raw 1920×1080)	–
$t = 0$–2 ms	Preprocessing (resize do 1280×720)	2 ms
$t = 2$ ms	START 5 watków	–
$t = 2$–17 ms	Równoległa egzekucja (4 modele + ego)	15 ms
$t = 17$ ms	łaczenie watków	–
$t = 17$–18 ms	zbieranie rezultatów z zapytań	1 ms
$t = 18$–23 ms	Fuzja transformera	5 ms
$t = 23$–27 ms	Mapowanie stref + ocena zagrozeń	4 ms
$t = 27$ ms	Generowanie altertów	< 1 ms
$t = 27$–30 ms	Wizualizacja	3 ms
	OPÓŹNEINEI DLA KLATKI	**30 ms**

$$\text{FPS} = \frac{1000 \text{ ms}}{30 \text{ ms}} = 33 \text{ FPS} \checkmark \tag{4.1}$$

Rozdział 5

Wartości parametry

Parametr	**Wartość**	**Uwagi**
Wejściowa rozdzielczość	1920×1080 (FHD)	Wejście wideo
Rozdzielczość preprocessing	1280×720 (HD)	Rozdzielczość treningowa
Opóźnenie per klatka	28–32 ms	Równoległe wykonanie
FPS Wyjściowy	31–35 FPS	Czas rzeczywisty, 'płynny'
Pamieć GPU (VRAM)	7–8 GB	4 models + buffers
Wolna pamieć GPU	2–3 GB	RTX 4070 Ti = 12 GB
Użycie CPU	30–40%	Małe obciażenie wielowatkowościa
Użycie RAM	2–3 GB	Out of 32 GB
Użycie GPU (%)	70–80%	Bezpieczne obciażenie
Wielkość modeli	200 MB (INT8)	4×50 MB każdy około
Czas trenowania (per model)	4–5 h	Dla RTX 4070 Ti
Stabilność interfejsu	Dobra/Wzorowa	Brak wycieków danych

Rozdział 6

Dataset Strategy

6.1 Dane Źródłowe

- **Dataset**: NVIDIA PhysicalAI-Autonomous-Vehicles (Hugging Face)
- **Liczba klipów**: 20–40 selected
- **Duration**: 7–14 minut nagrań ogólnie
- **Klatki po zcrawlowaniu klipów około**: 8,000–16,000 klatek
- **Train/Val/Test Split**: 70% / 15% / 15%

6.2 Selekcja Klipów

1. Różnorodne`hour_of_day` (dzień/noc)
2. Różnorodne `miesiace` (sezony)
3. Różnorodne kraje (różne infrastruktury)
4. `camera_front_wide_120fov == True` (tylko frontalna kamera)

6.3 Preprocessing Danych – Pozyskiwanie oznakowanych danych z uyżyciem Autodistill

1. Pobieramy klatki co 2–3 klatki (ograniczamy nadmiarowość)
2. Resize do 1280x720
3. **Auto-label via Autodistill + GroundingDINO/SAM** (Podstawowy model)
   (a) Setup Autodistill: `uv pip install autodistill autodistill-grounded-sam autodistill-yolov8`
   (b) Definiuj CaptionOntology z promptami dla każdej klasy:
   - `"car"` → `car`
   - `"red traffic light"` → `traffic_light_red`
   - `"green traffic light"` → `traffic_light_green`
   - `"person"` → `person`
   - `"bicycle"` → `bicycle`
   - `"obstacle"` → `obstacle`

   (c) Autodistill: `base_model.label(input_folder, output_folder)`
   (d) Output: YOLO-compatible dataset format
4. Słaba manualna rewizja (5–10% próbki, tylko niepewne oznakowania)
5. Mapa YOLO classes do 4 specjalistycznych kategorii dla treningu

Rozdział 7

Wartość Naukowa

7.0.1 Zintegrowana ocena zagrożeń z wykorzystaniem technologii Ego-Motion

- Dynamiczne dostrajanie zagrożeń na podstawie predkości/przyspieszenia
- Wykrywanie z uwzglednieniem manewrów
- Alerty uwzgledniajace kontekst

7.0.2 Multi-Model z równoległym przetwarzaniem wielowatkowym

- Porównanie: 4 specialist models vs. 1 multi-class
- Analiza kompromisów: dokładność a opóźnienie a złożoność
- Praktyczne informacje dotyczace wdrażania (wielowatkowość, zarzadzanie procesorem graficznym)

7.0.3 Wnioskowanie w czasie rzeczywistym na obrzeżach sieci

- System gotowy do produkcji z RTX 4070 Ti
- $<$ 35 ms opóźnienia realene do osiagniecia
- Kwestie zwiazane z wdrożeniem

7.1 Analiza porównawcza

Metrika	**Multi-Class**	**4 Specialist**	**Delta**	
Latency	25 ms	30 ms	+5 ms (20% wolniej)	
Accuracy (mAP50)	0.68	0.73	+0.05 (+7%)	
Small Objects	0.55	0.68	+0.13 (+24%)	
VRAM	4.5 GB	7/8 GB	+2 GB	
Złożność kodu	Mała	Średnia	Wysoka + utrzymanie	
Training Time	4–5 h	8–10 h	+4–5 h	

Rozdział 8

Praktyczny Roadmap

8.1 Faza 1: Dataset Preparation (3–4 dni)

- ☐ Pobieranie 20–40 klipów z PhysicalAI
- ☐ Crawlowanie klatek (1280×720)
- ☐ Auto-label via YOLOv8
- ☐ Sprawdzenie reczne (10–15%)
- ☐ Podział na train/val/test

ETA: 3–4 dni

8.2 Faza 2: Trenowanie modelu (5–6 dni)

- ☐ Trenowanie modelu samochodów (4–5 h)
- ☐ Trenowanie modelu świateł (4–5 h)
- ☐ Trenowanie modelu dla osób (4–5 h)
- ☐ Trenowanie modelu przeszkód (4–5 h)
- ☐ Kwantyzacja wszystkiego do INT8

ETA: 5–6 dni

8.3 Faza 3: Implementacja systemu (3–4 dni)

- ☐ Infrastruktura watków (5 watków)
- ☐ Zarzadzanie kolejka
- ☐ Warstwa fuzji transformatora
- ☐ Silnik mapowania stref

- ☐ Logika oceny zagrożeń (z ego-motion)
- ☐ Generowanie alertów
- ☐ Moduł wizualizacji

ETA: 3–4 dni

8.4 Faza 4: Integracja & Testowanie (2–3 dni)

- ☐ Testy kompleksowe
- ☐ Profilowanie opóźnień
- ☐ Profilowanie pamieci
- ☐ Testy obciażeniowe (ponad 100 klatek)
- ☐ Przypadki skrajne (noc, deszcz, zasłony)
- ☐ Benchmark a wartość bazowa

ETA: 2–3 dni

8.5 Faza 5: Dokumentacja i wyniki (2–3 dni)

- ☐ Napisz rozdziały pracy dyplomowej
- ☐ Wygeneruj wykresy porównawcze
- ☐ Stwórz analize porównawcza
- ☐ Film demonstracyjny
- ☐ Dokumentacja kodu

ETA: 2–3 dni