# Dokumentacja

Aplikacji do rozpoznawania języka migowego zrealizowanej w ramach zajęć

Inżynierii oprogramowania

w semestrze zimowym roku akademickiego 2024/2025

Patrycja Sapko

Izabela Jesionowska

Sopot 2025

# 1. Charakterystyka oprogramowania

**Nazwa skrócona: SE**

**Nazwa: SignEase** – połączenie słów "sign" (gest) i "ease" (łatwość), co sugeruje łatwość w nauce i komunikacji w języku migowym.

**Krótki opis ze wskazaniem celów**: Aplikacja służy do rozpoznawania znaków języka migowego w czasie rzeczywistym przy użyciu kamery internetowej. System wykorzystuje techniki uczenia maszynowego i przetwarzania obrazu do identyfikacji gestów wykonywanych przez użytkownika.

- Umożliwienie rozpoznawania znaków języka migowego w czasie rzeczywistym
- Stworzenie bazy danych gestów języka migowego
- Zapewnienie możliwości trenowania modelu na podstawie własnych przykładów
- Ułatwienie komunikacji z osobami posługującymi się językiem migowym
- Wsparcie procesu nauki języka migowego poprzez natychmiastową informację zwrotną

# 2. Prawa autorskie

**Autorzy:**

- Patrycja Sapko
- Izabela Jesionowska

**Warunki licencyjne:**

| Biblioteka | Licencja |
|---|---|
| **OpenCV** | Apache 2.0 |
| **MediaPipe** | Apache 2.0 |
| **PyQt5** | GPL v3 |
| **NumPy** | BSD |
| **scikit-learn** | BSD 3-Clause |
| **pygame** | LGPL |

Zgodnie z warunkami licencji GPL v3 (ze względu na wykorzystanie PyQt5), cały projekt musi być udostępniany na tej samej licencji.

# 3. Specyfikacja wymagań

**Wymagania funkcjonalne:**

1. Zarządzanie danymi treningowymi:
    - Możliwość dodawania nowych znaków do bazy danych
    - Zbieranie obrazów treningowych poprzez kamerę
    - Etykietowanie zebranych danych
2. Trening modelu:
    - Przetwarzanie zebranych danych obrazowych
    - Ekstrakcja cech charakterystycznych gestów
    - Trenowanie klasyfikatora Random Forest
    - Zapisywanie wytrenowanego modelu
3. Rozpoznawanie znaków:
    - Przechwytywanie obrazu z kamery w czasie rzeczywistym
    - Detekcja i śledzenie dłoni
    - Klasyfikacja wykrytych gestów
    - Wyświetlanie rozpoznanych znaków na ekranie
    - Odtwarzanie dźwięku odpowiadającego rozpoznanemu znakowi
4. Interfejs użytkownika:
    - Przycisk do rozpoczęcia rozpoznawania
    - Funkcja dodawania nowych znaków
    - Opcja trenowania modelu
    - Wskaźnik postępu podczas trenowania
    - Wyświetlanie komunikatów o błędach i statusie operacji

| Identyfikator | WF.1 |
|---|---|
| **Nazwa** | Zarządzanie danymi treningowymi |
| **Opis** | System umożliwia zbieranie i zarządzanie danymi treningowymi poprzez nagrywanie i etykietowanie gestów języka migowego. |
| **Priorytet** | 1 |

| Identyfikator | WF.1.1 |
|---|---|
| **Nazwa** | Dodawanie nowego znaku |
| **Opis** | System umożliwia dodanie nowego znaku do bazy danych poprzez nagranie serii gestów przy użyciu kamery. |
| **Priorytet** | 1 |

| Identyfikator | WF.1.2 |
|---|---|
| **Nazwa** | Zbieranie obrazów treningowych |
| **Opis** | System automatycznie zapisuje serię zdjęć podczas nagrywania nowego znaku, tworząc bazę danych treningowych. |
| **Priorytet** | 1 |

| Identyfikator | WF.2 |
|---|---|
| **Nazwa** | Trening modelu |
| **Opis** | System przetwarza zebrane dane i trenuje model uczenia maszynowego do rozpoznawania gestów. |
| **Priorytet** | 1 |

| Identyfikator | WF.3 |
|---|---|
| **Nazwa** | Rozpoznawanie znaków w czasie rzeczywistym |
| **Opis** | System w czasie rzeczywistym analizuje obraz z kamery i rozpoznaje wykonywane gesty języka migowego. |
| **Priorytet** | 1 |

| Identyfikator | WF.3.1 |
|---|---|
| **Nazwa** | Informacja dźwiękowa |
| **Opis** | System odtwarza dźwiękowy komunikat odpowiadający rozpoznanemu znakowi. |
| **Priorytet** | 2 |

**Wymagania pozafunkcjonalne**

1. Wydajność:
    - Rozpoznawanie znaków w czasie rzeczywistym (min. 15 FPS)
    - Czas odpowiedzi interfejsu użytkownika poniżej 100ms
    - Obsługa minimum 100 różnych znaków
2. Niezawodność:
    - Logowanie błędów i zdarzeń do pliku
    - Obsługa wyjątków i błędów
    - Zabezpieczenie przed utratą danych podczas treningu
3. Użyteczność:
    - Intuicyjny interfejs użytkownika
    - Czytelne komunikaty o błędach
    - Wizualna informacja zwrotna podczas rozpoznawania
4. Kompatybilność:
    - Wsparcie dla systemów Windows
    - Kompatybilność z większością kamer internetowych
5. Bezpieczeństwo:
    - Lokalne przetwarzanie danych
    - Bezpieczne zapisywanie danych treningowych
    - Ochrona przed nieautoryzowanym dostępem do modelu

| Identyfikator | WN.1 |
|---|---|
| **Nazwa** | Wydajność rozpoznawania |
| **Opis** | System musi działać w czasie rzeczywistym z minimalną częstotliwością 15 klatek na sekundę. |
| **Priorytet** | 1 |

| Identyfikator | WN.2 |
|---|---|
| **Nazwa** | Dokładność rozpoznawania |
| **Opis** | System musi osiągać dokładność rozpoznawania gestów na poziomie minimum 85%. |
| **Priorytet** | 1 |

| Identyfikator | WN.3 |
|---|---|
| **Nazwa** | Niezawodność |
| **Opis** | System musi działać stabilnie przez minimum 8 godzin ciągłej pracy. |
| **Priorytet** | 2 |

| Identyfikator | WN.4 |
|---|---|
| **Nazwa** | Logowanie zdarzeń |
| **Opis** | System musi zapisywać wszystkie istotne zdarzenia i błędy w pliku logów. |
| **Priorytet** | 2 |

# 4. Architektura oprogramowania

**Architektura rozwoju**

Python i Podstawowe Biblioteki

- Nazwa: Python
- Przeznaczenie: Język programowania wykorzystywany jako podstawa aplikacji
- Wersja: 3.x

Biblioteki do Przetwarzania Obrazu i ML

- Nazwa: OpenCV (cv2)
- Przeznaczenie: Przechwytywanie i przetwarzanie obrazu z kamery
- Wersja: 4.x
- Nazwa: MediaPipe
- Przeznaczenie: Detekcja i śledzenie punktów charakterystycznych dłoni
- Wersja: Latest stable
- Nazwa: NumPy
- Przeznaczenie: Operacje na macierzach i tablicach wielowymiarowych
- Wersja: Latest stable
- Nazwa: scikit-learn
- Przeznaczenie: Implementacja algorytmów uczenia maszynowego (Random Forest Classifier)
- Wersja: Latest stable

Interfejs Użytkownika

- Nazwa: PyQt5
- Przeznaczenie: Framework GUI
- Wersja: 5.x

System Dźwięku

- Nazwa: Pygame
- Przeznaczenie: Odtwarzanie dźwięków dla rozpoznanych znaków
- Wersja: Latest stable

Przechowywanie Danych

- Nazwa: pickle
- Przeznaczenie: Serializacja modelu ML i danych treningowych
- Wersja: Wbudowany w Python

**Architektura uruchomieniowa**

**Wymagania Systemowe**

- System operacyjny: Windows/Linux/MacOS
- Minimalna pamięć RAM: 4GB
- Procesor: Multi-core CPU
- Kamera internetowa: Wymagana

**Wymagane Komponenty**

- Python 3.x Runtime Environment
- Sterowniki kamery internetowej
- Zainstalowane biblioteki:
    - OpenCV
    - MediaPipe
    - NumPy
    - scikit-learn
    - PyQt5
    - Pygame

**Struktura katalogów:**

- data - dane treningowe
- sounds - pliki dźwiękowe
- model.p - wytrenowany model
- data.pickle - przetworzone dane treningowe
- models - zapisane modele
- logs - pliki logów
- migowy.py - główny plik aplikacji

**Przepływ danych**

**Zbieranie danych:**

- Kamera -> Przechwytywanie obrazu -> Detekcja dłoni -> Ekstrakcja cech -> Zapis do bazy danych

**Trening modelu:**

- Baza danych -> Preprocessing -> Trening klasyfikatora -> Zapisany model

**Rozpoznawanie:**

- Kamera -> Detekcja dłoni -> Ekstrakcja cech -> Klasyfikacja -> Wyświetlenie wyniku -> Odtworzenie

**Proces wdrożenia**

1. Instalacja Pythona i wymaganych bibliotek
2. Konfiguracja środowiska:
    - Utworzenie katalogów dla danych treningowych
    - Konfiguracja parametrów kamery
    - Ustawienie ścieżek do plików dźwiękowych
3. Uruchomienie aplikacji:
    - Wiersz poleceń
    - python Migowy.py

## 5. Testy

| Identyfikator | INIT-001 |
|---|---|
| **Nazwa** scenariusza | Sprawdzenie poprawności uruchomienia aplikacji |
| Kroki | Uruchom aplikację. Sprawdź czy interfejs GUI się wyświetla. Zweryfikuj obecność wszystkich przycisków. |
| Oczekiwany rezultat | Aplikacja uruchamia się bez błędów, wszystkie elementy GUI są widoczne |

| Identyfikator | ADD-001 |
|---|---|
| **Nazwa** scenariusza | Sprawdzenie funkcjonalności dodawania nowego znaku do bazy danych |
| Kroki | Kliknij „Dodaj znak”. Wprowadź etykietę znaku. Wykonaj serię gestów przed kamerą. |
| Oczekiwany rezultat | 100 zdjęć zostaje zapisanych w odpowiednim katalogu. |

| Identyfikator | TRAIN-001 |
|---|---|
| **Nazwa** scenariusza | Weryfikacja procesu trenowania modelu |
| Kroki | Kliknij "Analizuj i trenuj model". Poczekaj na zakończenie procesu. Sprawdź komunikat o wynikach. |
| Oczekiwany rezultat | Model zostaje wytrenowany, wyświetlone zostają statystyki sukcesu |

| Identyfikator | RECOG-001 |
|---|---|
| **Nazwa** scenariusza | Sprawdzenie funkcjonalności rozpoznawania znaków |
| Kroki | Kliknij "Rozpoznawanie". Wykonaj znany modelowi gest. Sprawdź czy znak został rozpoznany |
| Oczekiwany rezultat | Aplikacja poprawnie rozpoznaje gest i wyświetla etykietę |

**Sprawozdanie z wykonania Testów**

**Test INIT-001**

- Status: PASS
- Uwagi: Interfejs uruchamia się poprawnie na wszystkich testowanych systemach operacyjnych
- Znalezione problemy: Brak

**Test ADD-001**

- Status: PASS
- Uwagi:
    - Proces przebiega zgodnie z oczekiwaniami
    - Zdjęcia są zapisywane w odpowiedniej strukturze katalogów
- Znalezione problemy:
    - Czasami występują problemy z detekcją dłoni w słabym oświetleniu

**Test TRAIN-001**

- Status: PASS
- Uwagi:
    - Proces trenowania działa poprawnie
    - Wyświetlany jest pasek postępu
    - Generowany jest raport z procesu trenowania
- Znalezione problemy:
    - Przy bardzo dużej liczbie zdjęć proces może być czasochłonny

**Test RECOG-001**

- Status: PASS
- Uwagi:
    - Rozpoznawanie działa w czasie rzeczywistym
    - Etykiety są wyświetlane na obrazie z kamery
    - Dźwięki są odtwarzane poprawnie
- Znalezione problemy:
    - Czasami występują fałszywe rozpoznania przy szybkich ruchach
    - Może wystąpić opóźnienie w odtwarzaniu dźwięku

**Ogólne Wnioski z Testów**

1. Aplikacja działa stabilnie i spełnia podstawowe wymagania funkcjonalne
2. Główne problemy związane są z warunkami oświetleniowymi i szybkością ruchów
3. Zalecane jest przeprowadzenie dodatkowych testów w różnych warunkach oświetleniowych