Sprawozdanie z Projektu w ramach kursu "Inżynieria Wiedzy i Symboliczne Uczenie Maszynowe"

Autorzy: Mateusz Knap, Tomasz Policht **Data:** 22 czerwca 2025

---

1. Wstęp

Celem niniejszego sprawozdania jest przedstawienie przebiegu prac nad projektem realizowanym w ramach kursu "Inżynieria Wiedzy i Symboliczne Uczenie Maszynowe". Pierwotnym założeniem projektu było stworzenie zaawansowanego systemu do dopasowywania parametrycznego modelu 3D głowy (FLAME) do danych pochodzących z nagrań wideo. Projekt miał na celu zbadanie metod uśredniania i agregacji danych z wielu klatek w celu uzyskania bardziej precyzyjnego i metrycznie poprawnego modelu 3D twarzy.

W trakcie realizacji napotkano jednak na fundamentalne problemy techniczne, związane z konfiguracją środowiska i kompatybilnością kluczowych bibliotek, które uniemożliwiły kontynuację prac. W związku z tym, po wyczerpaniu dostępnych metod rozwiązania problemów, podjęto decyzję o zmianie tematu projektu.

Nowy projekt skupił się na praktycznym zastosowaniu metod optymalizacyjnych i zrealizowano go w formie aplikacji do przeprowadzania testów akustycznych z wykorzystaniem optymalizacji bayesowskiej. Poniższe sprawozdanie szczegółowo opisuje obie fazy projektu: pierwotną koncepcję oraz jej finalną, zrealizowaną wersję.

2. Pierwotna Koncepcja Projektu: Dopasowanie Modelu 3D Głowy

2.1. Cel i Motywacja

Głównym celem projektu było opracowanie algorytmu, który na podstawie krótkiego nagrania wideo głowy użytkownika byłby w stanie automatycznie wygenerować spersonalizowany, metryczny model 3D. Przetwarzanie wideo, w odróżnieniu od pojedynczego zdjęcia, miało dostarczyć danych z wielu perspektyw, co teoretycznie pozwoliłoby na wyeliminowanie niepewności związanych z estymacją niewidocznych części głowy (np. profilu czy tyłu głowy). Taki system mógłby znaleźć zastosowanie w branży AR/VR, medycynie (np. planowanie operacji) czy przy personalizacji produktów (np. wirtualne przymierzalnie okularów).

2.2. Planowane Metody i Technologie

1. **Model Parametryczny:** Jako podstawę geometrii twarzy wybrano model **FLAME**, który jest standardem w badaniach nad rekonstrukcją 3D twarzy i oferuje bogaty zbiór parametrów do kontroli kształtu, ekspresji i pozy.
2. **Sieć Neuronowa do Estymacji:** Do transformacji obrazu na parametry modelu FLAME wybrano najnowocześniejszą wówczas (z dostępnym kodem) sieć **MICA**. Model ten był trenowany w sposób nadzorowany, co pozwalało na uzyskanie metrycznie poprawnych rekonstrukcji, w przeciwieństwie do wielu metod samonadzorowanych.
3. **Zbiór Danych do douczania:** Planowano douczyć model MICA na zbiorze **The Headspace dataset**. Zbiór ten zawiera precyzyjne skany 3D głów, co miało kluczowe znaczenie dla poprawy rekonstrukcji geometrii uszu, które w oryginalnym modelu MICA miały niską wagę i były niedokładnie odwzorowywane.

4. **Przetwarzanie Wideo:** Finalnym elementem miał być system, który:
    - Przetwarzałby nagranie wideo, wybierając z niego kluczowe klatki.
    - Dla każdej klatki uruchamiałby zmodyfikowany model MICA w celu uzyskania zestawu parametrów FLAME.
    - Agregowałby (np. poprzez uśrednianie ważone) wyniki z wielu klatek, aby uzyskać jeden, stabilny i wiarygodny model 3D.

3. Problemy Techniczne i Przyczyny Zmiany Tematu

Mimo jasno określonego planu, realizacja projektu natrafiła na serię problemów technicznych, które okazały się niemożliwe do przezwyciężenia w dostępnym czasie. Problemy te dotyczyły głównie konfiguracji środowiska programistycznego i zależności między bibliotekami.

3.1. Problemy z Uruchomieniem Modelu MICA

Główną przeszkodą okazała się instalacja i kompilacja zależności MICA:

- **Problemy systemowe:** Brak wielu wymaganych pakietów w systemie Windows.
- **Konflikty w środowisku Conda:** Występowały liczne konflikty między wymaganymi wersjami pakietów, a niektóre ze starszych zależności nie były już dostępne w repozytoriach Conda.
- **Problemy z kompilacją:** Wystąpiły błędy linkowania narzędzi `g++` i `gxx` w środowisku Conda oraz niepowodzenia podczas kompilacji biblioteki **PyTorch3D** w podsystemie WSL (Windows Subsystem for Linux).
- **Problemy ze sterownikami:** Pojawiły się trudności z poprawnym działaniem technologii CUDA w środowisku WSL, co jest kluczowe dla wydajności obliczeń na GPU.

Podjęto liczne próby rozwiązania tych problemów, w tym wielokrotne reinstalacje środowisk z różnymi wersjami pakietów, reinstalację WSL, a także próbę konteneryzacji projektu przy użyciu Dockera. Niestety, żadna z tych prób nie zakończyła się sukcesem, a niektóre doprowadziły do niestabilności, a nawet uszkodzenia środowisk WSL i Docker.

3.2. Próby Użycia Alternatywnych Modeli

Wobec niepowodzeń z MICA, podjęto próby wykorzystania innych, podobnych modeli:

- **DECA:** Starszy, ale wciąż ceniony projekt. Niestety, napotkano na podobne problemy z zależnościami, głównie z biblioteką PyTorch3D.
- **EMOCA:** Nowsze rozwinięcie DECA, z aktywnym wsparciem społeczności. Problemy z instalacją okazały się analogiczne.
- **Pixel3DMM:** Najnowszy z testowanych modeli, o obiecującej architekturze. Ponownie, bariera w postaci problemów środowiskowych uniemożliwiła jego uruchomienie.

Wspólnym mianownikiem dla wszystkich niepowodzeń były problemy z kompilacją i konfiguracją biblioteki `pytorch3d` oraz zależnościami sprzętowymi (CUDA) w dostępnych środowiskach deweloperskich.

4. Zrealizowany Projekt: Kalibrator Słuchu z Optymalizacją Bayesowską

Ze względu na opisane wyżej, niemożliwe do pokonania trudności techniczne, podjęto decyzję o zmianie tematu projektu. Nowy projekt skupił się na innym, interesującym zagadnieniu z obszaru uczenia

maszynowego – **optymalizacji bayesowskiej** – i jej praktycznym zastosowaniu.

4.1. Opis i Cel Aplikacji

Zrealizowany projekt to aplikacja webowa o nazwie **"Kalibrator Słuchu z Optymalizacją Bayesowską"**. Jej celem jest wyznaczenie indywidualnej **krzywej równej głośności (izofony)** użytkownika w sposób interaktywny i wydajny. Aplikacja prosi użytkownika o porównywanie głośności tonów o różnych częstotliwościach z tonem referencyjnym o stałej głośności i częstotliwości.

4.2. Kluczowe Technologie i Metodologia

Sercem aplikacji jest **optymalizacja bayesowska**, zaimplementowana z użyciem **Procesów Gaussowskich (Gaussian Processes)**. Metoda ta działa w następujący sposób:

1. **Modelowanie Funkcji:** Problem znalezienia krzywej słuchu jest traktowany jako aproksymacja nieznanej funkcji `Głośność = f(Częstotliwość)`.
2. **Zbieranie Danych Początkowych:** Test rozpoczyna się od kilku pomiarów w losowo wybranych punktach (tzw. *seeding*), aby zbudować wstępny model.
3. **Inteligentny Wybór Kolejnych Punktów:** Po każdej odpowiedzi użytkownika model (Proces Gaussowski) jest aktualizowany. Następnie, za pomocą **funkcji akwizycji** (w tym przypadku opartej na zasadzie *Upper Confidence Bound*), algorytm identyfikuje częstotliwość, dla której niepewność modelu co do poprawnej głośności jest **największa**.
4. **Minimalizacja Liczby Testów:** Użytkownik jest proszony o ocenę właśnie w tym "najbardziej informacyjnym" punkcie. Takie podejście drastycznie redukuje liczbę potrzebnych pomiarów w porównaniu do liniowego przeszukiwania całego pasma częstotliwości. Pozwala to uzyskać precyzyjną krzywą już po 10-15 testach.

4.3. Architektura Aplikacji

- **Backend:** Zbudowany w języku Python z użyciem frameworka **FastAPI**. Odpowiada za logikę optymalizacji bayesowskiej (biblioteka `scikit-learn`), generowanie i odtwarzanie dźwięku (`sounddevice`, `numpy`) oraz komunikację z frontendem przez WebSocket.
- **Frontend:** Minimalistyczny interfejs użytkownika napisany w czystym HTML, CSS (Pico.css) i JavaScript. Działa w przeglądarce bez potrzeby kompilacji. Wizualizacja krzywej słuchu w czasie rzeczywistym jest realizowana za pomocą biblioteki **Plotly.js**.

Aplikacja jest wieloplatformowa i prosta w uruchomieniu, co stanowiło celowy kontrast w stosunku do problemów napotkanych przy pierwotnym projekcie.

5. Podsumowanie i Wnioski

Prace nad projektem stanowiły cenne doświadczenie, choć ich finał odbiega od pierwotnych założeń. Próba implementacji najnowocześniejszych rozwiązań z dziedziny rekonstrukcji 3D twarzy unaoczniła, jak dużym wyzwaniem w pracy badawczej jest zarządzanie zależnościami, konfiguracja środowiska i replikacja wyników. Problemy z bibliotekami takimi jak PyTorch3D pokazują, że nawet publicznie dostępny kod może być niezwykle trudny do uruchomienia poza precyzyjnie zdefiniowanym środowiskiem autorów.

Mimo niepowodzenia w realizacji pierwszego celu, zmiana tematu pozwoliła na zgłębienie innego zaawansowanego zagadnienia – **optymalizacji bayesowskiej**. Zrealizowana aplikacja do kalibracji słuchu jest

w pełni funkcjonalnym i praktycznym przykładem zastosowania tej techniki. Projekt ten dowiódł umiejętności adaptacji do nieoczekiwanych problemów oraz zdolności do pomyślnego wdrożenia złożonych koncepcji algorytmicznych w działającym produkcie.