Politechnika
Śląska

PROJEKT INŻYNIERSKI

System wizyjny do śledzenia ruchu gałek ocznych

Bartosz WUWER
Nr albumu: 296949

Kierunek: Automatyka i Robotyka
Specjalność: Technologie informacyjne w automatyce i robotyce

PROWADZĄCY PRACĘ
Dr inż. Krzysztof Jaskot
KATEDRA Automatyki i Robotyki
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2025

Tytuł pracy

System wizyjny do śledzenia ruchu gałek ocznych

Streszczenie

(Streszczenie pracy – odpowiednie pole w systemie APD powinno zawierać kopię tego streszczenia.)

Słowa kluczowe

(2-5 slow (fraz) kluczowych, oddzielonych przecinkami)

Thesis title

Vision system for tracking the movement of the eyeballs

Abstract

(Thesis abstract – to be copied into an appropriate field during an electronic submission – in English.)

Key words

(2-5 keywords, separated by commas)

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie i osadzenie w dziedzinie

Śledzenie ruchu gałek ocznych (ang. eye-tracking), zwane również jako okulografia, jest techniką, która badana jest od ponad stu lat. Istotność tej techniki wynika z faktu, że ruchy gałek ocznych są ściśle związane z procesami poznawczymi, takimi jak uwaga, percepcja, pamięć, czy procesy decyzyjne. Skupiając wzrok na danym punkcie, umieszczamy go w centralnym obszarze naszego pola widzenia które charakteryzuje się największą rozdzielczością, co pozwala na dokładne analizowanie szczegółów. Ten fakt wpływa także na proces skupienia – gdy koncentrujemy się na danym obiekcie lub obszarze, skupiamy na nim wzrok (często wystarczy jedynie krótki moment).

Możliwość rejestrowania ruchów oczu pozwala na zrozumienie w jaki sposób obserwator eksploruje otaczający go świat. Posiadając tą wiedzę możliwe jest wyciągnięcie wniosków na temat tego co jest interesujące lub istotne dla obserwatora, jakie emocje się z tym wiążą, czy nawet jakie procesy poznawcze zachodzą w jego umyśle, czy rozumie on to co widzi. Nie trudnym jest zauważyć jak cenne mogą być te informacje w szerokim spektrum dziedzin.

Okulografia odgrywa kluczową rolę w psychologi poznawczej, psychologi społecznej, neurobiologii, marketingu, czy medycynie. W psychologii poznawczej ruch oczu jest ściśle związany z pamięcią, podejmowaniem decyzji, obciążeniem poznawczym i uczeniem się asocjacyjnym. W psychologii społecznej eye-tracking pozwala na wgląd w zachowania społeczne i ich analizę, co pozwala badać empatię, prospołeczność, czy fobie społeczne [32]. W neurobiologii bada się powiązania ruchu oczu ze szlakami neuronowymi odpowiedzialnymi za podejmowane akcje i procesy myślowe, dając możliwość w diagnozach i wsparciu osób dotkniętych chorobą Parkinsona [12], Alzheimera [11], a także autyzmem czy łagodnym upośledzeniem funkcji poznawczych [31]. W medycynie okulografia pozwala na diagnozę między innymi oczopląsu, który może być objawem np. stwardnienia rozsianego, uszkodzenia śródmózgowia lub urazu ucha wewnętrznego [4]. Niezaprzeczalnie technika ta jest

niezwykle cenna i efektywna na wielu płaszczyznach naukowych i praktycznych.

1.2 Cel oraz zakres pracy

Celem niniejszej pracy jest wykonanie narzędzia które pozwoli na uniwersalne śledzenie ruchu gałek ocznych, bez zdefiniowanej docelowej grupy użytkowników. System ten powinien być prosty w obsłudze i niewymagający zaawansowanej technologii, ale nie ograniczający w razie potrzeby bardziej zaawansowanych użytkowników, prezentując użyteczność zarówno dla naukowców, studentów, jak i hobbystów. Tego typu narzędzie pozwoli na eksplorację danych eye-trackingowych, otwierając możliwości na dalszą analizę i interpretację wyników, a także zapewni solidną podstawę w razie potrzeby modyfikacji lub rozbudowy systemu w bardziej ukierunkowany sposób.

Praca ta skupia się przede wszystkim na samym procesie wykrywania źrenic i śledzenia ruchu gałek ocznych na ich podstawie oraz implementacji systemu wizyjnego, który pozwoli na zapis, wizualizację i potencjalną dalszą analizę danych zebranych przez kamerę, w tym także kamerę internetową. Całość zrealizowana jest w języku Python, z wykorzystaniem bibliotek takich jak OpenCV, NumPy, pandas oraz Matplotlib, a wszelkie prezentowane dane zostały uchwycone przy użyciu wbudowanej kamery laptopa. Praca nie będzie obejmować rozległej analizy zebranych danych, ani eksperymentów przeprowadzonych z użyciem owego systemu. Nie została także stworzona grupa testowa, więc większość testów przeprowadzone zostały na autorze pracy.

- wprowadzenie w problem/zagadnienie
- osadzenie problemu w dziedzinie
- cel pracy
- zakres pracy
- zwięzła charakterystyka rozdziałów
- jednoznaczne określenie wkładu autora, w przypadku prac wieloosobowych – tabela z autorstwem poszczególnych elementów pracy

Rozdział 2

Śledzenie ruchu gałek ocznych – analiza tematu

2.1 Sformułowanie problemu

Widzeniem plamkowym (fovealnym) nazywamy wspomniane już wcześniej zjawisko, że widzimy w największej rozdzielczości jedynie centrum pola widzenia. Wynika to z budowy siatkówki, która posiada plamkę żółtą (fovea) – jest to niewielki obszar, który charakteryzuje się największym zagęszczeniem czopków. Taka budowa siatkówki sprawia, że nieustannie poruszamy oczami. Skupiając wzrok na danym punkcie, musimy być gotowi na to, że cała reszta pola widzenia straci dla nas na ostrości. Jest to proces, który przypomina filtracje wielu otaczających nas informacji, do paru, które mają w tym momencie znaczenie.

2.1.1 Uwaga wzrokowa

Uwaga wzrokowa może zostać opisana przez idiomy „gdzie" i „co", które pozwalają na zobrazowanie jej selektywnej natury. „Gdzie" to proces wizualnego wyszukiwania i wyboru lokalizacji, która zwróciła naszą uwagę, w celu dokładniejszego zbadania. Istotnym aspektem tego procesu selekcji jest widzenie peryferyjne, czyli takie obejmujące obszar oddalony od naszego punktu skupienia, ale wciąż pozwalający na wyodrębnienie kształtów i ruchu, co w pewien sposób prowadzi nasze centralne spojrzenie. Przykładem może być spojrzenie przez okno – w pierwszym momencie nasz wzrok kierowany jest na wyraźne kształty, światła czy nagłe ruchy jak np. lądujący ptak.

„Co" można nazwać odwrotnością „gdzie" – jest to proces szczegółowego badania danego obszaru, charakteryzują go takie zjawiska jak fiksacja, czyli stabilizacja wzroku na danym punkcie oraz ruchy sakadowe, czyli szybkie mimowolne ruchy oczu pomiędzy kolejnymi punktami, które pozwalają na obserwacje. Całość tworzy kanał percepcyjny o ograniczonym zasięgu przestrzennym, obejmujący interesujący obszar, budując w ten

sposób naszą świadomość i zrozumienie celu naszego spojrzenia. Wracając do poprzedniego przykładu, po zauważeniu ptaka, nasze spojrzenie skupi się na nim (fiksacja), a ruchy sakadowe oczu pozwolą na dokładne zbadanie jego kształtu, koloru i detali. To właśnie połączenie „gdzie" i „co" pozwala na głębsze zrozumienie otaczającego nas świata, a owe ścieżki skanowania dają nam wgląd w proces poznawczy badanej osoby, jako że pomiar widzenia fovealnego w czasie odzwierciedla chwilowe i jawne skupienie uwagi wzrokowej obserwatora.

2.1.2 Poszukiwanie wzrokowe

Tutaj warto wspomnieć o poszukiwaniu wzrokowym, jest to proces aktywnego przeszukiwania pola widzenia w celu znalezienia konkretnego celu, a częścią tego procesu jest etap przeduwagowy. Etap ten równolegle analizuje duży obszar widzenia jednocześnie, dzieje się to automatycznie, nieświadomie i nie wymaga widzenia fovealnego. Co charakteryzuje ten proces to umiejętność rozpoznania czterech podstawowych cech: kolor, rozmiar, orientacja oraz obecność i/lub kierunek ruchu. Etap przeduwagowy jest pierwszym krokiem poszukiwania wzrokowego i obejmuje dużą część poszukiwania typu równoległego (np. próba zauważenia nagłego ruchu spadającej gwiazdy) i relatywnie niewielką część poszukiwania typu seryjnego (np. przeszukiwanie obiektów na stole w poszukiwaniu kluczy), które wymaga uwagi i przenoszenia wzroku od obiektu do obiektu.

2.1.3 Analiza ruchów oczu

Powyższe rozważania pozwalają przejść do głównego nurtu tematu śledzenia ruchu gałek ocznych, czyli analiza ruchów oczu. Rozróżniamy pięć podstawowych typów ruchów oczu: wspomniane już wcześniej sakadyczne, wergencyjne, przedsionkowe, płynne podążanie (smooth pursuit) oraz oczopląs fizjologiczny, który jest naturalny dla zdrowej osoby i często występuje podczas fiksacji. Ruchy te można podzielić na dobrowolne, mimowolne i odruchowe, a sygnały je kontrolujące pochodzą z obszarów korowych mózgu. Do opisania tych procesów można posłużyć się modelowaniem matematycznym. Do zrozumienia jawnej uwagi wzrokowej wystarczy modelowanie trzech typów ruchów: fiksacji, która pokazuje chęć utrzymania wzroku na stacjonarnym obiekcie, sakad, które mogą wskazywać na chęć zmiany punktu uwagi oraz płynnych pościgów, które podobnie do fiksacji, pozwalają na śledzenie obiektu, ale ruchomego.

Ruchy sakadowe

Sygnał sakad można opisać jako funkcja impuls/skok, gdzie impuls na wejściu reprezentuje prędkość, a skok pozycję, impuls jest przepuszczany przez filtr, który przekształca go w skok. Prostą reprezentacją ruchu sakadowego jest filtr liniowy różniczkujący, który

dokonuje potrzebnej konwersji informacji prędkość na przemieszczenie. Wzór owego filtra w dziedzinie czasu można zapisać równaniem 2.1.

$$x_t = g_0 \cdot s_t + g_1 \cdot s_{t-1} + g_2 \cdot s_{t-2} + \dots \tag{2.1}$$

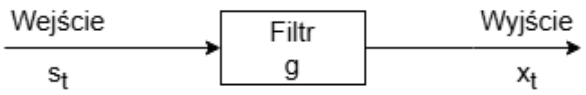
Wyjście x_t zależy od bieżącego wejścia s_t oraz jego poprzednich wartości $(s_{t-1}, s_{t-2}, \dots)$ ważonych odpowiednimi współczynnikami filtra $(g_0, g_1, g_2, \dots)$. Wzór 2.1 można przedstawić sumarycznie równaniem 2.2.

$$x_t = \sum_{k=0}^{\infty} g_k \cdot s_{t-k} \tag{2.2}$$

Filtr Haara jest jednym z przykładów filtrów który przybliża różniczkowanie, jest to filtr o długości 2, czyli operujący jedynie na dwóch kolejnych próbkach sygnału wejściowego (s_t, s_{t-1}), a przybliża on pierwszą pochodną, przyjmując, że prędkość zmian sygnału jest stała w czasie trwania dwóch próbek. Współczynniki filtra są równe $g_0 = 1$ oraz $g_1 = -1$, co oznacza że różniczkowanie jest zrealizowane poprzez różnicę między dwiema kolejnymi próbkami sygnału. W związku z tym transmitancję filtru Haara można zapisać jako równanie 2.3.

$$\begin{aligned} x_t &= g_0 \cdot s_t + g_1 \cdot s_{t-1} \\ x_t &= 1 \cdot s_t - 1 \cdot s_{t-1} \\ x_t &= s_t - s_{t-1} \\ \mathcal{Z}\{x_t\} &= \mathcal{Z}\{s_t - s_{t-1}\} \\ X(z) &= (1 - z) \cdot S(z) \\ \frac{X(z)}{S(z)} &= 1 - z \end{aligned} \tag{2.3}$$

Diagram omawianego modelu przedstawiony jest na rysunku 2.1.



Rysunek 2.1: Diagram modelu ruchów sakadowych z filtrem liniowym różniczkującym.

Płynne podążanie

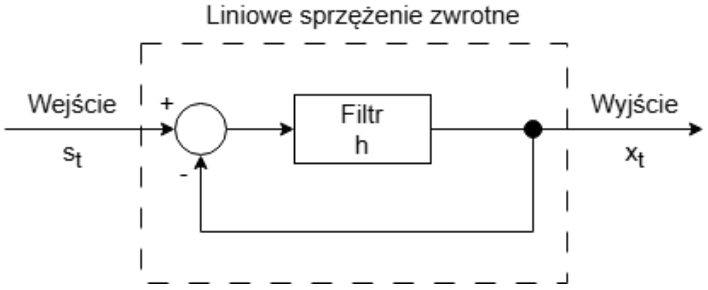
Płynne podążanie występuje gdy obserwator śledzi obiekt w ruchu, ruch ten oczywiście nie może być zbyt gwałtowny. Tego typu śledzenie jest przykładem systemu sterowania z ujemnym sprzężeniem zwrotnym. Do modelowania owych ruchów używana jest prosta pętla, którą można opisać nastepujacym równaniem 2.4 w dziedzinie czasu.

$$h \cdot (s_t - x_t) = x_{t+1} \tag{2.4}$$

Wejście s_t to pozycja celu, wyjście x_t to pożądana pozycja oka. Można zauważyć, że w tym przypadku nie jest wymagana transformacja informacji wejściowej na wyjściową, a jedynie zmodyfikowanie jej wartości, w związku z tym h jest liniowym niezmiennym w czasie filtrem, czyli po prostu wzmocnieniem systemu. Przeprowadźmy teraz transformację Z na powyższym równaniu 2.4 by uzyskać transmitancję systemu przedstawioną wzorem 2.5.

$$\begin{aligned} \mathcal{Z}\{h \cdot (s_t - x_t)\} &= \mathcal{Z}\{x_{t+1}\} \\ H(z) \cdot (S(z) - X(z)) &= X(z) \\ H(z) \cdot S(z) - H(z) \cdot X(z) &= X(z) \\ H(z) \cdot S(z) &= (H(z) + 1) \cdot X(z) \\ \frac{X(z)}{S(z)} &= \frac{H(z)}{H(z) + 1} \end{aligned} \tag{2.5}$$

W ten sposób sygnał z receptorów oka służy za błąd, który następnie jest kompensowany w celu utrzymania obrazu w przestrzeni widzenia plamkowego. Diagram modelu przedstawiony jest na rysunku 2.2.



Rysunek 2.2: Diagram modelu płynnego podążania z liniowym sprzężeniem zwrotnym.

Fiksacja

Można zauważyć, że fiksacja jest procesem podobnym do płynnego podążania, z tą różnicą, że w tym przypadku obiekt jest nieruchomy. Jednakże proces ten nie da się bezpośrednio z sobą porównać i najprawdopodobniej nie ma wspólnego obwodu neuronalnego. Nasze komórki systemu wzrokowego są fizjologicznie wrażliwe na ruch, gdyby dany obiekt został unieruchomiony względem siatkówki, po krótkim czasie widzenie zaniknie. Powoduje to konieczność mikrosakad i innych mimowolnych, drobnych ruchów oczu. Można więc uznać, że model fiksacji jest podobny do modelu płynnego podążania, który próbuje utrzymać pozycję oka na danym punkcie, a mikrosakady i inne ruchy można uznać za szum w systemie kontrolnym, który można wyrazić jako $e_t = x_t - s_t$. Jest to losowa fluktuacja wokół punktu fiksacji, a jej wartość średnia pozostaje stała.

Ruch gałek ocznych i związane z nim widzenie jest bardzo szerokim tematem interdyscyplinarnym, na potrzeby tego projektu powyższe sformułowanie problemu, chociaż uproszczone i nie wyczerpujące, pozwala na zrozumienie w pełni istoty prezentowanego systemu.

2.2 Osadzenie tematu w kontekście aktualnego stanu wiedzy

Pomiar ruchu oczu był realizowany już w latach 70. XX wieku, nic dziwnego więc, że od tego momentu powstały różne techniki pozwalające na zebranie owych danych. Należy podkreślić, że metody te można podzielić na mierzące ruch oczu względem głowy, oraz takie które mierzą orientacje oczu w przestrzeni, czyli „punkt spojrzenia".

2.2.1 Metody śledzenia ruchu oczu: przegląd historyczny i alternatywy

Najstarszą z metod śledzenia ruchu oczu jest elektro-okulografia (EOG ang. *electrooculography*), wciąż wykorzystywana np. w badaniach klinicznych. Metoda ta polega na umieszczeniu elektrod na skórze twarzy wokół oczu i pomiarze różnicy potencjałów. Metoda ta z założenia mierzy ruch oczu względem głowy, ale można rozszerzyć ją o pomiar ruchu głowy dając możliwość wyliczenia punktu spojrzenia.

Jedna z najdokładniejszych metod pomiaru ruchu oczu polega na umieszczeniu soczewki kontaktowej bezpośrednio na gałce ocznej. Soczewka ta powinna być odpowiednio duża, by objąć zarówno rogówkę jak i twardówkę, a na jej powierzchni umieszczany jest albo obiekt optyczny, którego zdaniem jest dokładne odbijanie światła lub dostarczenie wyraźnych kształtów potrzebnych do śledzenia, albo cewka wykonaną z drutu, która poruszając się w polu elektromagnetycznym pozwala na wykonanie pomiaru. Metoda ta jest

bardzo dokładna, ale niezwykle inwazyjna i nieprzyjemna dla badanego, a także, podobnie do elektro-okulografii, mierzy ruch oczu względem głowy.

Wideo-okulografia i foto-okulografia tworzą razem szeroką grupę metod, które opierają się na analizie wyróżniających się cech oka podczas jego obrotu, takich jak: kształt źrenicy, pozycja granicy tęczówki i twardówki, czy odbicie światła (często podczerwonego) od rogówki. Metody te są nieinwazyjne, ale same w sobie nie pozwalają na określenie punktu spojrzenia, więc w celu wyznaczenia owego punktu często stosuje się unieruchomienie głowy osoby badanej, wyznaczenie punktu odniesienia np. przez odbicia światła od powierzchni oka, lub odpowiednią kalibrację np. prosząc osobę badaną o utrzymanie wzroku na danym punkcie ekranu.

2.2.2 Analiza wideo z wykorzystaniem źrenicy i odbicia rogówkowego

Opisane metody wymagają unieruchomienia głowy (przez różnego rodzaju podpórki pod brodę lub głowę, albo nawet belki nagryzowe) lub zastosowanie dodatkowego sprzętu mierzącego ruch głowy, by wyznaczyć punkt spojrzenia. Jest to główna wada tych metod, jako że miejsce skupienia uwagi daje istotne informacje, które są najczęściej pożądane przez użytkowników okulografów. Dlatego metoda należąca do grupy wideo-okulografii, opierająca się na analizie obrazu wideo z kombinacją wykrycia źrenicy i odbicia rogówkowego, która pozwala na określenie punktu spojrzenia z dużą dokładnością, jest jedną z najpopularniejszych wyborów zarówno w świecie naukowym jak i komercyjnym. System prezentowany w tej pracy można zaliczyć do wideo-okulografii, z możliwością włączenia trybu z pomiarem odbicia rogówkowego, więc metoda ta zostanie omówiona szerzej od pozostałych.

Wykrywanie punktu spojrzenia metodą detekcji źrenicy i odbicia rogówkowego wymaga kilku kluczowych elementów. Potrzebna jest kamera wideo, która zarejestruje aktualny stan gałek ocznych oraz sprzęt, który pozwoli na przetworzenie obrazu i ostatecznie wykrycie interesujących cech, najlepiej w czasie rzeczywistym. Aktualnie oba wymienione elementy stają się coraz tańsze i bardziej wydajne, tak naprawdę większość osób posiada je w swoim smartfonie, co sprawia, że technika ta staje się bardziej dostępna dla szerokiego grona użytkowników. Aparatura ta jest także coraz bardziej miniaturyzowana, dzięki czemu aktualnie dostępne są systemy montowane na głowie, jak i montowane na biurku, które działają na tej samej zasadzie, różniąc się zasadniczo jedynie wielkością. Warto jednak zaznaczyć, że systemy montowane na głowie zazwyczaj posiadają także dodatkową kamerę wyznaczającą tak zwany POV (ang. point of view, czyli punkt widzenia), a spojrzenie jest monitorowane względem tego obrazu, w odróżnieniu do montowanych na biurku, które zazwyczaj mierzą spojrzenie względem pewnej powierzchni np. monitora. Należy omówić także źródło światła, którego odbicie rejestrowane jest podczas detekcji.

Możliwe jest wykorzystanie punktowego źródła światła białego o odpowiednim natężeniu, jednakże może to prowadzić do dyskomfortu przy użytkowaniu lub nawet chwilowego oślepienia osoby badanej, wytrącając ją z stanu skupienia. W związku z tym najczęściej wykorzystywanym są źródła światła podczerwonego, które znajdują się w większości poza widzialnym spektrum, nie irytując oka, nie zakłócając procesu badania i pozwalając na użycie wyższego natężenia, operowanie na takim zakresie częstotliwości światła wymaga użycia specjalnych kamer, które są w stanie rejestrować światło IR (ang. infrared).

Omawiane odbicie rogówkowe nazywane jest obrazem Purkinjego, budowa oka sprawia, że pojawiają się cztery odbicia, pierwsze to odbicie od zewnętrznej powierzchni rogówki (najbardziej widoczne), drugie od wewnętrznej powierzchni rogówki, trzecie od zewnętrznej powierzchni soczewki, a czwarte od wewnętrznej powierzchni soczewki. By poprawnie wyznaczyć punkt spojrzenia, wymagane jest wyznaczenie dwóch punktów odniesienia, pierwszy określający obrót oczu w oczodole, a drugi określający stałe położenie względem oczu. Pierwszym punktem zazwyczaj jest środek źrenicy, a drugim najczęściej jest pierwszy obraz Purkinjego, powstały przez źródło światła ustabilizowane względem głowy, lub powierzchni badanej. Przy nieruchomej głowie, ruch oczu odczytywany będzie jako zmieniająca się różnica pomiędzy tymi dwoma punktami, w odwrotnym przypadku, gdy głowa poruszy się, a oczy pozostaną w fiksacji na danym punkcie, dla systemu montowanego na biurku różnica ta pozostanie stała, a dla systemu montowanego na głowie, różnica będzie zmieniać się proporcjonalnie do zmiany punktu widzenia. Istnieją także okulografy piątej generacji z technologią DPI (ang. Dual Purkinje Image), mierzą one dodatkowo czwarty obraz Purkinjego, dzięki temu, mogą rozróżniać ruchy translacyjne oka od ruchów rotacyjnych, jako że przy translacji obrazy Purkinjego poruszają się o tą samą odległość, a przy rotacji zmieniają swoje rozdzielenie, co zwiększa precyzję pomiaru, ale wymaga bardziej skomplikowanego oprogramowania, a także unieruchomienie głowy może okazać się konieczne.

2.3 Studia literaturowe

Powyższe sformułowanie problemu i jego osadzenie w kontekście aktualnego stanu wiedzy zostało przedstawione w oparciu o książkę „Eye Tracking Methodology: Theory and Practice" autorstwa Andrew T. Duchowskiego [5]. Książka ta dogłębnie omawia temat śledzenia ruchu oczu, pozwalając na zgłębienie zarówno podstaw procesów wzrokowych w kontekstach różnych dziedzin naukowych, a także szeroko omawia praktykę pomiaru ruchu oczu. Powyższa analiza tematu skupia się jedynie na najistotniejszych aspektach, które pozwolą na zrozumienie istoty prezentowanego systemu, a wiedzę na ich temat można poszerzyć lekturą wspomnianej książki.

Eye-tracking staje się coraz bardziej dostępny, co sprawia że technologia ta znajduje zastosowanie w wielu obszarach, zarówno w badaniach naukowych jak i do użytku konsu-

menckiego czy komercyjnego. Część producentów owych systemów udostępnia publikacje naukowe, w których zostały one wykorzystane, a ich analiza przedstawia różne rozwiązania omawianego tematu.

2.3.1 Metoda źrenicy i odbicia rogówkowego w badaniach

Jednym z przedstawicieli firm skupiających się na analizie zachowań ludzkich jest iMotions, wyróżniając się modularnym systemem z centralnym hubem. Firma ta oferuje biosensory w formie modułów, pozwalające na np. śledzenie oczu, analizę wyrazu twarzy, rejestrowanie aktywności elektrycznej mózgu i rejestrowanie aktywności serca. Dane zebrane przez różne czujniki są integrowane w jednym oprogramowaniu co pozwala badaczom na prostą synchronizację i analizę danych w jednym systemie [22]. Jednak mimo tak obszernego asortymentu biosensorów, to właśnie moduł śledzenia wzroku na ekranie (montowany na biurku) był wykorzystywany najczęściej w badaniach naukowych, z czego eye-tracking był najczęściej wykorzystywany w systemach jednomodułowych [20]. Pokazuje to jak uniwersalnym i podstawowym narzędziem jest eye-tracker w badaniach behawioralnych i nie tylko.

Jednym z przeprowadzonych badań z użyciem technologii iMotions było „Using Psychophysiological Data to Facilitate Reflective Conversations with Children about their Player Experiences". Podobne badania były przeprowadzane wielokrotnie w stosunku do osób dorosłych, jednak w przypadku grupy badanej składającej się z dzieci, badacze mogą napotkać znaczny problem z odpowiednim zrozumieniem wywiadu, w którym uczestnicy dzielą się swoimi wrażeniami z przeprowadzonej rozgrywki. Omawiane badanie miało głównie na celu sprawdzenie czy zebrane dane z biosensorów są wstanie wspomóc refleksje dzieci na temat ich doświadczenia w grze i czy dzieci napotkają problemy z interpretacją prezentowanych danych, jako że odnoszenie się do stanu psychofizjologicznego może zminimalizować problem z komunikacją pomiędzy dzieckiem, które posiada ograniczone umiejętności werbalne, a badaczem.

Do śledzenia ruchu gałek ocznych na ekranie zastosowano okulograf Smart Eye AI-X o częstotliwości próbkowania 60 Hz, posługujący się techniką detekcji źrenic i odbicia rogówkowego, pozwalając na sporą swobodę ruchu głowy w przestrzeni 35 cm na 30 cm, jednocześnie posiadając dokładność (różnica pomiędzy rzeczywistą pozycją spojrzenia a pozycją spojrzenia zarejestrowaną przez okulograf) 0,5° i precyzję (średnia kwadratowa punktów mierzonych w jednej pozycji oka) 0,1°, zwracając dane wyjściowe binokularowe z wskaźnikiem jakości zawierające: punkt spojrzenia, średnicę źrenicy oraz znacznik czasowy [21]. Oprócz tego zastosowano moduł rejestrujący aktywność sercową, analizujący wyrazy twarzy oraz mierzący reakcję skórno-galwaniczną.

Badanie składało się z dwóch etapów, wpierw zebrano dane w laboratorium podczas gdy badane dzieci grały w dwie gry oraz podczas wywiadów bezpośrednio po zakończonej

rozgrywce, a następnie prezentowano badanym momenty z pierwszego etapu z uwzględnieniem danych z biosensorów i zadawano pytania o ich doświadczenia. Badania wskazały, że dzieci są wstanie zrozumieć i odpowiednio odnieść się do prezentowanych danych psychofizjologicznych, a oparcie się o nie może pomóc w zwerbalizowaniu swojego doświadczenia i informacji zwrotnej przez badane dzieci. Dzieci nie miały problemu z zrozumieniem większości pomiarów za wyjątkiem reakcji skórno-galwanicznej, jednakże dane eye-trackingowe były swego rodzaju wyjątkiem, jako że nie prezentowały bezpośrednio doświadczenia i uczuć związanych z danym wydarzeniem w grze, a raczej dawały wgląd w strategię objęta przez dziecko zmagające się z aktualnym wyzwaniem, mimo tego prezentowane śledzenie punktu spojrzenia było naturalne w interpretacji przez badanych [10]. Publikacja ta pokazuje jak okulografia może zostać skutecznie użyta w badaniach naukowych, ale przede wszystkim przedstawia istotność tej techniki w zastosowaniach komercyjnych np. testując doświadczenia graczy przy produkcji gier wideo dla różnych grup wiekowych.

2.3.2 Porównanie z video-okulografią z użyciem kamerki internetowej

Warto omówić także przykłady systemów opierających się jedynie na analizie wideo z kamerki internetowej, jako że system prezentowany w tej pracy działa przede wszystkim na tej zasadzie. Ten typ okulografii opartej na analizie wideo w czasie rzeczywistym cieszy się dużą popularnością wśród konsumentów, jako że sprzętowo wymaga od użytkownika posiadanie jedynie komputera z wbudowaną lub zewnętrzną kamerą, a całość przetwarzania obrazu, wyświetlania i zbierania danych oraz wyliczania punktu spojrzenia realizowane jest przez software producenta eye-trackera. Istotność takiego rozwiązania pojawia się także gdy przeprowadza się badania na dużej grupie, jako że osoba badana sama dostarcza sprzęt w formie laptopa czy nawet smartfona, a zbierane dane przesyłane są do badaczy przez internet, eliminując potrzebe stwarzania przestrzeni laboratoryjnej i planowania dogodnych terminów eksperymentów. Dlatego firmy specjalizujące się w okulografii, często oferują także oprogramowanie do śledzenia ruchu gałek ocznych przez kamerkę internetową, jak wspomniana już wcześniej firma iMotions. Często także dostępne są rozwiązania od niezależnych dostawców oprogramowania, które za subskrypcją, opłatę, lub całkowicie bezpłatnie oferują śledzenie punktu spojrzenia przez kamerę komputera, jednym z takich programów, który zostanie omówiony jest darmowy GazeRecorder.

W publikacji „WebET 3.0 -Validation Study Report” przeprowadzono badania sprawdzające efektywność śledzenia ruchu gałek ocznych przez oprogramowanie WebET 3.0 firmy iMotions z wykorzystaniem kamerek internetowych uczestników, bez względu na ich oświetlenie, rozdzielczość kamery czy prędkość internetu. Grupa badana została utworzona tak by reprezentować różne pochodzenie, wiek, płeć i kolor oczu, uczestnicy mogli także posiadać owłosienie twarzy oraz nosić okulary. By stworzyć punkty odniesienia po-

kazano badanym gify z kotami oraz emotikony w tych samych miejscach, a następnie obliczono dokładność mierząc odległości pomiędzy zarejestrowanymi punktami spojrzenia, a ustalonymi punktami odniesienia.

Wyniki dokładności wskazały, że system iMotions jest niezależny na zmienne wynikające z pochodzenia, koloru oczu, wieku, płci oraz owłosienia twarzy, jednak posiadanie okularów mogło wpływać na wynik śledzenia, rodzaj i natężenie oświetlenia także nie wpływał znacząco na działanie WebET 3.0. Ponad 90% uczestników posiadało dokładność mniejszą od 5° dokładności, 70% badanych posiadało dokładność mniejszą od 3° dokładności, a mediana dokładności wynosiła 2,08° [23]. Dokładność alternatywnego darmowego oprogramowania GazeRecorder została zaprezentowana w przeglądzie systematycznym i wynosiła 1,43° z możliwością poprawy do 1,3° przez unieruchomienie głowy, pozwalając na noszenie okularów za wyjątkiem okularów antyrefleksyjnych [8].

2.3.3 Podsumowanie metod w literaturze

Metoda śledzenia ruchu gałek ocznych przy użyciu źrenicy i odbicia rogówkowego w podczerwieni znacznie przewyższa – pod względem dokładności – metodę polegającą na kamerce internetowej, w związku z tym przy badaniach skupiających się na analizie gęsto ułożonych informacji wyspecjalizowany okulograf może być bardziej pożądany. Jednakże aktualna dostępność sprzętu i oprogramowania potrzebnego do śledzenia z użyciem kamerki internetowej daje możliwość na bardziej obszerne zastosowania, a ich dokładność pozwala na efektywne śledzenie punktu spojrzenia w przypadkach gdy informacje są ułożone z odpowiednimi dystansami.

- sformułowanie problemu
- osadzenie tematu w kontekście aktualnego stanu wiedzy (*state of the art*) o poruszanym problemie
- studia literaturowe [16, 17, 15, 14] - opis znanych rozwiązań (także opisanych naukowo, jeżeli problem jest poruszany w publikacjach naukowych), algorytmów,

Rozdział 3

Wymagania i narzędzia

3.1 Wymagania funkcjonalne

Postawienie konkretnych granic funkcjonalnych projektu jest kluczowym elementem tworzenia rozwiązania. Działają one na zasadzie zabezpieczenia przed usprawnianiem i rozszerzaniem projektu bez końca, a także pozwalają na skupienie uwagi inżyniera na istotnych aspektach. Można zauważyć korelację pomiędzy rolą wymagań funkcjonalnych w tworzeniu projektu, a plamką żółtą w oku omawianej w podrozdziale 2.1, oba elementy dbają o to by wiele informacji zostało przefiltrowane do tych, które są najważniejsze.

3.1.1 Limitacje sprzętowe i środowiskowe

Do śledzenia ruchu gałek ocznych używany jest bardzo szeroki wachlarz sprzętów i związanych z nimi rozwiązań technicznych omówionych w podrozdziałach 2.2 i 2.3, jednakże w założeniu tego projektu było opierać się na systemie wizyjnym, co pozwala na ograniczenie znanych metod do video-okulografii. Kolejną ustaloną limitacją było opieranie się jedynie na aktualnie dostępnym sprzęcie, którym był laptop ASUS Vivobook 15 z wbudowaną kamerką internetową USB2.0 HD UVC o częstotliwości próbkowania 30 Hz, podczas korzystania z kamery automatycznie włączane było także niewielkie źródło światła przylegające z jej prawej prawej strony. Tego typu laptop z wbudowaną kamerą reprezentuje jakość sprzętu często używaną do zastosowań biurowych i użytku własnego, dlatego też założono, że źródła światła powinny zostać ograniczone do podstawowego oświetlenia wewnętrznego, tworząc środowisko typowe do pokoju mieszkalnego, w którym korzysta się z komputera.

3.1.2 Wykorzystanie platform i narzędzi programistycznych

By uprościć użycie komputera (wspomnianego w punkcie 3.1.1), zdecydowano się na pozostaniu przy już zainstalowanym systemie operacyjnym, którym jest Windows 11.

Podobnie postawiono wymaganie względem edytora kodu źródłowego i systemu kontroli wersji, jako że użycie znajomego środowiska, które już jest zainstalowane i było wcześniej używane na wykorzystanym laptopie, pozwala na szybkie zamknięcie przygotowań do pracy. W związku z tym skupiono się na rozwiązaniach, które są kompatybilne z edytorem tekstu Visual Studio Code i klientem Git z graficznym interfejsem użytkownika GitHub Desktop, jako że narzędzia te ściśle z współpracują z sobą i serwisem hostingowym GitHub tworząc intuicyjną i znajomą w użyciu całość. Stawianym wymaganiem do języka programowania, oprócz wspomnianej kompatybilności, jest jego popularność, uniwersalność, klarowność i szeroki wybór bibliotek. Język o takich cechach znacznie zwiększy produktywność, uprości stosowanie bardziej skomplikowanych narzędzi i pozwoli na efektywne uczenie się i rozwiązywanie problemów opierając się na społeczności użytkowników. Zalety te wspomogą uzyskanie natychmiastowych rezultatów pracy oraz skoncentrowanie uwagi na poprawie i rozwój funkcjonalności systemu. Łatwo dostępnym i popularnym językiem o wysokim poziomie abstrakcji jest Python, który jest oficjalnie wspierany przez Visual Studio Code. Język ten posiada możliwość korzystania z biblioteki OpenCV, która daje dostęp do wielu rozwiązań przetwarzających obraz. Ostatnim, bardziej personalnym atutem proponowanego języka, jest niewielkie ówczesne doświadczenie programowania w nim, a co za tym idzie szansa na zyskanie cennych umiejętności w trakcie pracy.

3.1.3 Zakres działania systemu

Nakreślone limitacje w punkcie 3.1.1 i 3.1.2 odpowiednio ograniczyły możliwe rozwiązania, pozwalając na ustalenie realnych funkcjonalności programu. Aspektem, który przede wszystkim determinuje stopień skomplikowania stawianych wymagań, jest ograniczony czas na wykonanie omawianego systemu. W związku z tym, rozwiązaniem, które może zapewnić użyteczność planowanego systemu, nawet w przypadku jego niepełnego ukończenia, jest skoncentrowanie się na podstawowych funkcjonalnościach. Dzięki temu powstanie solidny szkielet projektu, oferujący funkcje dostępne na danym etapie prac. W konsekwencji zakres działania systemu śledzącego ruch oczu przedstawia się następująco:

1. Przetwarzanie obrazu z kamery powinno odbywać się w czasie rzeczywistym. Dzięki takiemu rozwiązaniu nie będzie potrzebne nagrywanie, a także pozwoli to na weryfikację pracy systemu w trakcie użytkowania. Biblioteka OpenCV posiada wiele rozwiązań, które pozwalają na operowaniu na obrazie w czasie rzeczywistym.

2. Proces śledzenia ruchu gałek ocznych powinien przebiegać poprzez wpierw wykrycie twarzy, następnie w rejonie twarzy przeprowadzanie detekcja oczu, a dopiero w rejonie oczu dojdzie do wykrycia źrenicy. Dzięki takiemu założeniu obraz będzie stopniowo ograniczany do rejonu zainteresowania, gwarantując ograniczenie błędnych detekcji.

3. System powinien działać w sposób binokularny, pozwalając na wykrycie i zidentyfikowanie obu oczu (a następnie źrenic) z rozróżnieniem lewego i prawego oczodołu. Dzięki takiemu rozwiązaniu dane wyliczane przez system będą skategoryzowane, zabezpieczając przed błędami i niespójnościami w analizie.

4. Wykrywanie źrenic powinno być uniwersalne względem oświetlenia otoczenia. Oczywistym jest, że skrajnie złe oświetlenie uniemożliwi poprawne działanie, ale program powinien cechować się pewnym stopniem elastyczności, nie wymuszając użytkowania w jednym konkretnym środowisku.

5. By obliczyć przemieszczenie źrenicy musi zostać wyznaczony umowny punkt odniesienia względem powierzchni, na której skupiona jest uwaga, punkt ten powinien być odporny na niewielkie ruchy głowy. Dodatkowym atutem będzie możliwość prostej kalibracji owego punktu. Jest to kluczowy element pozwalający na detekcję ruchu oczu.

6. Wyznaczanie ruchu oczu poprzez jedynie umowny punkt odniesienia, o ograniczonej możliwości weryfikacji poprawności pomiaru, nie jest rozwiązaniem idealnym i może być niewystarczające dla bardziej zaawansowanych użytkowników. W związku z tym system powinien oferować opcję wyznaczenia ruchu gałek ocznych względem realnego punktu odniesienia np. poprzez zastosowanie dodatkowego sprzętu.

7. Wyliczone przemieszczenie źrenicy w czasie rzeczywistym powinno zostać zapisane do pliku, który pozwoli na analizę zebranych danych w tym pozycję źrenicy, indeks wskazujący przynależność do odpowiedniego oka oraz moment czasowy zebranego pomiaru.

3.2 Wymagania niefunkcjonalne

Jak ustalono w punkcie 3.1.3 projekt ten ma skupić się na utworzeniu solidnego szkieletu programu, takie rozwiązanie kładzie nacisk przede wszystkim na funkcjonalność i wymusza jednocześnie zastosowanie jedynie najważniejszych elementów interfejsu, pozwalając użytkownikowi na wykorzystanie jego pełni możliwości. Wymagania niefunkcjonalne mogą w takim razie być skromniejsze względem funkcjonalnych, ale wciąż należy je dokładnie określić by uniknąć utraty kluczowych elementów przez zaniedbanie tej części. Wymagania te prezentują się następująco:

1. Program powinien wyświetlać obraz z kamery, jest to najbardziej podstawowe wymaganie, ale także najbardziej istotne. Obserwacja obrazu daje informację zwrotną użytkownikowi, pozwalając na szybką weryfikację działania kamery, a także analizę przystępności otoczenia, w którym dokonuje śledzenia.

2. Jako że możliwe jest uszkodzenia pliku programu, lub braku odpowiedniego sprzętu wymaganego do przeprowadzenia detekcji, istotnym jest wdrożenie podstawowej kontroli błędów zwracającej odpowiednią informacje i zakańczając działanie programu w przypadku wykrycia nieprawidłowości. Informacja ta powinna nakierować użytkownika na źródło problemu, pozwalając na łatwiejsze dojście do jego rozwiązania.

3. W punkcie 3.1.3 wymieniono różne funkcjonalności programu, część z nich (np. rodzaj wyznaczania punktu odniesienia) z reguły nie będzie działać równolegle, wymaga to więc prostego sposobu na kontrolowanie, która z użyteczności będzie w tym momencie wykorzystywana.

4. Podczas działania systemu śledzenia ruchu gałek ocznych dokonywana będzie detekcja kilku elementów, takich jak twarz, oczy oraz źrenice, istotnym aspektem wizualnym jest podkreślenie tych obszarów, dając możliwość na naturalną kontrolę działania rozwiązania na każdym jego etapie. Dodatkowo warto zaznaczyć punkt odniesienia i punkt środka źrenicy, obrazując dzielący ich dystans. Jest to duża ilość informacji, która może zaciemnić obraz z kamery, program powinien w związku z tym zadbać o maksymalną przejrzystość.

5. Poza bardziej zrozumiałym empirycznie obrazowaniem działania systemu śledzącego, takim jak zaznaczanie obszarów detekcji, przydatnym może okazać się wizualizowanie technicznego aspektu wykrywania, prezentując to jak komputer odbiera i przekształca obraz z kamery. Tak przedstawione wyniki, chociaż mniej czytelne dla osób niezaznajomionych, pozwolą użytkownikowi na poprawę działania programu pod jego własne preferencje.

6. Przeglądanie zebranych danych może być dużym wyzwaniem, jako że komercyjnie dostępne kamery internetowe mogą rejestrować obraz 30 lub nawet 60 razy na sekundę, co przy dłuższym śledzeniu generuje obszerną ilość pomiarów. Z tego powodu wymagana jest wizualizacja zebranych danych w prosty sposób np. przez wykresy czasowe. Takie rozwiązanie pomoże w dalszej analizie i potencjalnej obróbce pomiarów.

3.3 Przypadki użycia

Wymagania funkcjonalne i niefunkcjonalne omawiane w podrozdziale 3.1 i 3.2 pozwalają dokładnie opisać konkretne elementy systemu, ale mogą utrudniać zrozumienie całości. Diagram UML (Unified Modeling Language) przypadku użycia pozwala na łatwe zobrazowanie każdego elementu w kontekście całości programu, dopełniając analizę sta-

wianych wymagań. Diagram przypadku użycia prezentowanego systemu widoczny jest na rysunku 3.1.

Rysunek 3.1: Diagram UML przypadku użycia systemu do śledzenia ruchu gałek ocznych.

Diagram na rysunku 3.1 prezentuje po lewej użytkownika, który pełni rolę aktora wchodzącego w interakcję z systemem wizyjnym do śledzenia ruchu gałek ocznych. Użytkownik może oddziaływać na system na cztery sposoby, rozpoczynając śledzenie, kalibrując punkt odniesienia, personalizując ustawienia oraz wizualizując zebrane pomiary, akcje są ustawione od góy w kolejności sugerującej korzystanie z programu. Śledzenie ruchu gałek ocznych rozpoczyna się inicjalizacją systemu, jeśli program nie napotka błędu, rozpoczyna wykrywanie obszarów twarzy. Zawężanie obszarów zainteresowań następuje sekwencyjnie, dopiero gdy pojawi się twarz można będzie szukać oczu i następnie źre-

nic. Gdy oczy zostaną wykryte, automatycznie utworzony zostanie punkt odniesienia, a gdy detekcja przebiegnie w całości pomyślnie, zapisany zostanie pomiar pozycji źrenicy względem punktu odniesienia oraz zostanie wyświetlony przetworzony obraz pomagający w dostosowaniu czułości. Użytkownik może następnie przeprowadzić kalibrację punktu odniesienia ręcznie lub kalibracją zaawansowaną, może też w każdej chwili wrócić do kalibracji automatycznej. Personalizacja, pozwalająca dostosować czułość detekcji oraz opcje obrysowania wykrytych rejonów twarzy, może okazać się konieczna w przypadku różnych warunków oświetleniowych, lub zmieniających się preferencji użytkownika. Po udanej sesji śledzenia ruchu gałek ocznych, użytkownik może zdecydować się na wizualizację zebranych pomiarów, generując wykresy czasowe. W ten sposób omawiany diagram UML przedstawia symbolicznie pełen zakres interakcji użytkownika z systemem oraz kolejność operacji, obrazując działanie aplikacji w typowym scenariuszu.

3.4 Narzędzia i metody wykorzystane w pracy

3.4.1 ASUS Vivobook 15

Projekt prezentowany w tej pracy skupia się przede wszystkim na oprogramowaniu, dlatego sprzęt wykorzystany do jego zrealizowania zamyka się na laptopie z wbudowaną kamerką internetową, który jest także źródłem stawianych wymagań w punkcie 3.1.1. Jest to ASUS Vivobook 15 D1502YA-BQ309 posiadający ośmiordzeniowy procesor AMD Ryzen 7 7730U z zintegrowanym układem graficznym ATI AMD Radeon Graphics. Notebook ten oryginalnie stosowany był do nauki i programowania, co pozwoliło na szybkie przystosowanie go do pracy nad prezentowanym rozwiązaniem, wpasowuje się on również w założenie limitacji sprzętowych użytkowników. Wbudowana kamera to USB2.0 HD UVC WebCam producenta Realtek, urządzenie te rejestruje obraz w 30 klatkach na sekundę w rozdzielczości 1280×720, posiada także diodę światła białego, która ułatwia śledzenie w niesprzyjających warunkach oświetleniowych i tworzy widoczne na obrazie odbicie rogówkowe. Mimo że nie jest to kamera specjalistyczna, jej parametry są wystarczające do realizacji założeń projektu, a podobne specyfikacje są powszechne w kamerach wbudowanych w laptopy, co sprawia, że opracowane rozwiązanie może być łatwo wdrożone przez użytkowników bez konieczności zakupu dodatkowego sprzętu.

3.4.2 Narzędzia programistyczne

Github Desktop i Github

Projekt o takiej skali wymaga użycia systemu kontroli wersji, by zabezpieczyć przed utratą postępów i pozwalając na bezproblemowe testowanie nowych funkcji. Do przechowywania kodu w chmurze i zabezpieczenia go na zdalnych serwerach wykorzystano

hosting GitHub. W celu ułatwienia pracy z systemem kontroli wersji Git użyto aplikacji GitHub Desktop, zapewniającej graficzny interfejs użytkownika, co pozwoliło na wygodniejsze korzystanie z poleceń Git, zatwierdzanie zmian z odpowiednimi wiadomościami, operowanie na gałęziach repozytorium, analizowanie historii zmian oraz synchronizację lokalnego repozytorium z jego zdalnym odpowiednikiem na GitHubie.

Visual Studio Code (VS Code)

Visual Studio Code to edytor kodu źródłowego, który świetnie integruje się z Github desktop i przez to także z Github. VS Code mimo niskich wymagań systemowych, oferuje wiele funkcjonalności, co pozwoliło na wykorzystanie go jako główne narzędzie programistyczne. Dzięki kolorowaniu składni, automatycznym podpowiedzą i uzupełnianie kodu, edytor ten przyśpiesza pracę i ogranicza ilość błędów. Obszerna biblioteka wtyczek, pozwala na dostosowanie VS Code do własnych potrzeb, np. dodając obsługę języków programowania takich jak Python, czy LaTeX. Wbudowana integracja z systemem kontroli wersji podkreśla zmodyfikowane pliki, ułatwiając zarządzanie repozytorium. Wbudowana obsługa terminala oraz zaawansowane debugowanie z kontrolą punktu wstrzymania, inspekcją zmiennych i śledzeniem wykonywania kodu w czasie rzeczywistym pozwala na szybką kompilację i naprawę programu, zwłaszcza napisanego w języku Python [9]. Podsumowując, Visual Studio Code okazał się niezastąpionym narzędziem w procesie tworzenia zarówno projektu, jak i niniejszej pracy, pozwalając nie tylko na ułatwioną organizację, ale również zwiększoną efektywność.

Python 3.13.0

Python to wysokopoziomowy język programowania, który charakteryzuje się czytelną i intuicyjną składnią. Cechuje go dynamiczne typowanie, przypisując typy zmiennych w trakcie działania programu oraz automatyczne zarządzanie pamięcią. Największym atutem Pythona jest jego popularność i łatwość nauki, co sprawia, że wokół tego języka powstała spora społeczność programistów. Dzięki temu w sieci można znaleźć liczne fora dyskusyjne, obszerne dokumentacje, a także poradniki i kursy wideo, które okazały się niezastąpionym wsparciem w realizacji niniejszej pracy. Python posiada również obszerną bibliotekę standardową oraz szeroki wybór wysokiej jakości bibliotek zewnętrznych, które oferują gotowe rozwiązania dla niemal każdego zagadnienia programistycznego. Dzięki nim możliwe jest realizowanie złożonych zadań, takich jak analizy danych i przetwarzania obrazów, często przy użyciu zaledwie kilku linijek kodu [7]. W efekcie, Python zaopatrzony w odpowiednie biblioteki, w przypadku realizacji omawianego rozwiązania, okazał się jedynym językiem programowania potrzebnym do implementacji wszystkich funkcjonalności systemu.

OpenCV

OpenCV jest otwartą biblioteką, która posiada ponad 2500 zoptymalizowanych algorytmów, w tym rozpoznawania obrazu i uczenia maszynowego. Jej dodatkowym atutem jest dokładna dokumentacja i szeroka baza użytkowników aktywnie dzielących się swoimi obserwacjami i wyjaśnieniami na forach dyskusyjnych, pozwalając na sprawne wykorzystanie ogromnego potencjału tej biblioteki [29]. To właśnie dzięki niej możliwe było większość operacji związanych z obrazem w czasie rzeczywistym, od przechwycenia go przez kamerę, przez wykrycie oraz obrysowanie twarzy i oczu, po binaryzację i operacje morfologiczne.

NumPy

NumPy to otwarta biblioteka zapewniająca obsługę dużych wielowymiarowych i jednolitych macierzy oraz tablic. OpenCV wykorzystuje i przekazuje tablice NumPy w swoich funkcjach, pozwalając na integrację obu bibliotek i łatwe operowanie na obrazach w formie macierzy [3]. Popularność tej biblioteki, zwłaszcza w użyciu wraz z OpenCV, sprawia, że jest szeroko omawiana przez społeczność programistyczną, co przekłada się na dużą liczbę poradników i zasobów edukacyjnych [13]. Co za tym idzie jej zastosowanie w projekcie gwarantowało dostępność wsparcia w rozwiązywaniu potencjalnych problemów.

Matplotlib

Matplotlib jest biblioteką do tworzenia wykresów w języka Python, bezproblemowo współpracująca z NumPy. Użycie jej razem z modułem Pyplot pozwala na wizualizacje wyników z interfejsem podobnym do programu MATLAB [30]. Zastosowanie jej w pracy jest podparte dopełnieniem wcześniej wymienionych bibliotek, a także zbieżnością z znanym już środowiskiem MATLAB.

pandas

Pandas to biblioteka do analizy i manipulacji danymi, posiadająca narzędzia do czytania i zapisywania danych w plikach CSV oraz plikach tekstowych [18]. Podczas śledzenia ruchu gałek ocznych generowana i zapisywana jest duża ilość danych, a biblioteka ta pozwala na łatwe wydobycie ich z pliku i wykorzystanie w dalszej części programu.

Time

Time jest częścią podstawowej biblioteki Python i nie wymaga dodatkowej instalacji przed użyciem, służy do wykonywanie operacji związanych z czasem. Zbierany pomiary muszą być zapisywane razem z momentem czasowym, by pozwolić na pełną analizę, jest to nieskomplikowane zadanie, które można rozwiązać z użyciem tej biblioteki opierając się na oficjalnej dokumentacji Python [6].

PyInstaller 6.11.1

Następnym krokiem, który warto wykonać po zakończeniu prac nad rozwiązaniem, jest ułatwienie jego dystrybucji i użytkowania. W tym celu wykorzystano PyInstaller, który umożliwia spakowanie skryptu Pythona i wszystkich jego zależności do samodzielnego pliku wykonywalnego (exe). Dzięki temu użytkownicy mogą uruchomić program bez konieczności instalowania interpretera Pythona ani dodatkowych modułów. Warto jednak zaznaczyć, że użyto PyInstaller w wersji na system operacyjny Windows, a oprogramowanie te nie pozwala na kompilację krzyżową (czyli wymaga tego samego systemu od użytkownika) [1]. Taka forma dystrybucji znacząco ułatwia udostępnianie aplikacji, a także eliminuje problemy związane z niedopatrzeniem w konfigurowaniu środowiska.

3.4.3 Metodyka pracy nad projektem

Praca nad projektem przebiegała w sposób iteracyjny, co przypomina metodykę zwinnego wytwarzania oprogramowania, a w szczególności podejście stosowane w Scrum, które opiera się na empiryzmie. Proces ten polegał na wyznaczaniu pojedynczych zadań do realizacji, implementowaniu ich rozwiązań, a następnie analizowaniu wyników i dostosowywaniu kolejnych kroków [19]. Oznacza to, że stawiane wymagania i używane narzędzia, omawiane w rozdziale 3, także były poszerzane w sposób iteracyjny w czasie realizacji projektu. Dzięki takiemu podejściu możliwe było rozpoczęcie pracy z ograniczoną wiedzą i umiejętnościami, a wraz z postępem, rozbijanie głównego założenia projektu na coraz bardziej szczegółowe elementy, z jednoczesnym nabieraniem doświadczenia w używaniu narzędzi i poszerzając wiedzę na temat możliwych rozwiązań. Wraz z pracą funkcjonalności systemu były ulepszane, a problemy zarówno w programie jak i założeniach, szybko wykrywane i eliminowane.

- wymagania funkcjonalne i niefunkcjonalne
- przypadki użycia (diagramy UML) – dla prac, w których mają zastosowanie
- opis narzędzi, metod eksperymentalnych, metod modelowania itp.
- metodyka pracy nad projektowaniem i implementacją – dla prac, w których ma to zastosowanie

Rozdział 4

Specyfikacja zewnętrzna

4.1 Wymagania sprzętowe i programowe

By móc korzystać z programu śledzącego ruch gałek ocznych wymagane jest użycie kamery i komputera wyposażonego w odpowiednie podzespoły i oprogramowanie. Jakość i rodzaj wymaganej kamery internetowej jest z grubsza bez znaczenia, jako że podczas korzystania z tego systemu, można dostosować jego ustawienia rekompensując niedociągnięcia sprzętowe. Równie prostym do wyznaczenia wymogiem jest wolne miejsce na dysku, które odpowiada sumie rozmiarów plików wykonywalnych z teoretycznym zapasem na powiększenie się pliku tekstowego z pomiarami (10% sumarycznego rozmiaru plików programowych), wynoszące 100 MB$\cdot$1.1 = 110 MB. Dzięki plikom wykonywalnym, do uruchomienia programu nie jest wymagane także żadne inne oprogramowanie. Aczkolwiek kwestia wymagań wydajności procesora, rozmiaru pamięci RAM oraz wspieranego systemu operacyjnego wymaga głębszej analizy.

4.1.1 Procesor i pamięć RAM

By ustalić wymagania sprzętowe prezentowanego systemu przeprowadzono obserwację procentowego użycia procesora oraz pamięci RAM w trakcie detekcji. System okazał się używać średnio 36% procesora, z nagłymi wzrostami nawet do $44,1\%$, w konsekwencji przyjęto, że procentowe użycie tego zasobu nie powinna wzrosnąć ponad 45%. Użycie pamięci RAM utrzymywało się na stałym poziomie 170 MB, jest to na tyle niewielka ilość, że nie stanowi istotnego obciążenia dla współczesnych standardów. W związku z tym można przyjąć, że minimalne wymagania pamięciowe pokrywają się z zaleceniami systemu operacyjnego. Znając podzespoły laptopa omówione w punkcie 3.4.1 można oszacować wymagania sprzętowe minimalne i zalecane.

Do wyznaczenia procesora spełniającego minimalne wymagania do stabilnego korzystania z niniejszego rozwiązania, posłużono się największą na świecie stroną internetową z testowaniem wzorcowym procesorów, utrzymywaną przez firmę PassMark Software [24].

Opierając się na punktacji PassMark [26], można oszacować wysokość oceny procesora, który pozwoli na bezproblemowe korzystanie z programu śledzącego. Wiedząc, że procesor laptopa referencyjnego posiada ocenę wysokości 18500 (P_{ref}) [25], a program może wykorzystać do 45% jego mocy obliczeniowej ($CPU_{\%ref}$), wyznaczono ilość punktów procesora, który będzie w stanie utrzymać działanie programu przy pełnym obciążeniu jego zasobów (P_{min}). Przedstawiono to na równaniu 4.1.

$$P_{min} = P_{ref} \cdot CPU_{\%ref} = 18500 \cdot 45\% = 8325 \tag{4.1}$$

Procesorem, który posiada niewiele wyższą ocenę równą 8407 od obliczonego P_{min} jest Intel Xeon E5-4650 @ 2.70GHz [28]. Jednakże dla wygody użytkowania oraz zapewnienia płynnej pracy programu w różnych warunkach, warto zastosować procesor o wyższej wydajności. Dlatego, przyjmując dodatkowy zapas wydajności na poziomie 20%, zalecany wynik PassMark (P_{rec}) obliczony został za pomocą równania 4.2.

$$P_{rec} = 1.2 \cdot P_{min} = 1.2 \cdot 8325 = 9990 \tag{4.2}$$

Przystępnym cenowo procesorem posiadającym wynik 10440, wyższy od obliczonego P_{rec}, jest Intel Xeon E5-2660 v2 @ 2.20GHz [27].

4.1.2 System operacyjny

Kwestia systemu operacyjnego jest przede wszystkim zależna od programu PyInstaller. Jak już wspomniano w punkcie 3.4.2 PyInstaller nie jest narzędziem pozwalajacym na utworzenie plików wykonywalnych z kompilacją krzyżową, w związku z tym konieczne jest by użytkownik korzystał z systemu Windows firmy Microsoft. System wizyjny do śledzenia ruchu gałek ocznych został przetestowany na systemie Windows 11 (również na innej maszynie), gwarantując stabilną pracę w tej konkretnej wersji. Starsze wersje owego systemu operacyjnego, jako że nie zostały przetestowane, nie gwarantują poprawnego funkcjonowania. Warto jednak zaznaczyć, że użyte biblioteki oraz wersja Pythona i PyInstallera wspierają wersje systemu Microsoft od Windows 8 wzwyż, a tak stworzone pliki wykonywalne działają na zasadzie dołączonego interpretatora języka Python [2], w konsekwencji prezentowany w tej pracy program, nie powinien napotkać problemu działając na systemie Windows 8 lub nowszym.

4.1.3 Podsumowanie wymagań

System: Windows 11 (stabilna praca), Windows 8 i Windows 10 (potencjalna kompatybilność – nie przetestowane)

Procesor: Xeon E5-4650 @ 2.70GHz (minimalnie), Intel Xeon E5-2660 v2 @ 2.20GHz lub lepszy (zalecane)

Pamięć RAM: 2 GB (wymaganie systemowe Windows 8 wersja 64-bitowa)

Miejsce na dysku: 110 MB

Dodatkowy sprzęt: Kamera internetowa, klawiatura

4.2 Sposób instalacji

Rozwiązanie dzieli się na dwa pliki wykonywalne `eye_detect.exe` i `eye_tracking_plot.exe`, które nie wymagają przeprowadzenia procesu instalacji. By uruchomić owe programy, wystarczy umieścić je na komputerze przy użyciu przenośnej pamięci USB, lub pobierając je z repozytorium Github, albo z udostępnionego folderu na dysku google pod linkiem `https://drive.google.com/drive/folders/1RGOLKfGqTqCwREWvMt3sQEOPwD7aKOdE?usp=sharing`. System do śledzenia ruchu gałek ocznych uruchomi się przez otworzenie pliku `eye_detect.exe`, a zebrane dane można wyświetlić w formie wykresów czasowych otwierając plik `eye_tracking_plot.exe`. Plik `eye_tracking_data.txt` nie jest wymagany do uruchomienia `eye_detect.exe`, który w przypadku jego braku utworzy go na koniec śledzenia, jednak jest on wymagany do uruchomienia `eye_tracking_plot.exe`. Plik tekstowy `eye_tracking_data.txt` oraz `Readme.txt` zawierający instrukcję instalacji i użytkowania dostępny jest razem z resztą plików wykonywalnych na dysku Google i Github.

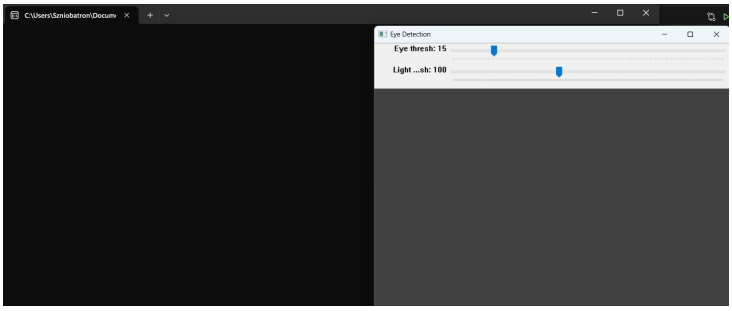
4.3 Kategorie użytkowników

Prezentowany system wizyjny do śledzenia ruchu gałek ocznych nie jest kierowany do konkretnej grupy odbiorców, jest on z założenia uniwersalny w zastosowaniu i niewymagający sprzętowo, pozwalając każdej osobie wyposażonej w komputer z kamerą i posiadającej zainteresowanie w dziedzinie okulografii cieszyć się jego możliwościami. Chociaż, tak jak wspomniano w punkcie 3.1.3, rozwiązanie te oferuje przede wszystkim podstawowe funkcje śledzenia, może one okazać się użyteczne zarówno dla hobbystów, jak i studentów, czy badaczy. Bardziej zaawansowani użytkownicy mogą skorzystać z funkcji pomiaru względem odbicia rogówkowego, na przykład wyposażając system w kamerę NIR i diodę podczerwoną. Z drugiej strony osoby mniej zaznajomione z tematem, lub posiadające

gorszej jakości sprzęt, mogą wykorzystać kalibrację ręczną jako prosty sposób na badanie skupienia uwagi na jednym punkcie ekranu.

4.4 Sposób obsługi

W pierwszej kolejności użytkownik musi zdecydować się czy chce rozpocząć śledzenie, czy wizualizacje pomiarów zebranych w pliku tekstowym, jako że funkcje te uruchamiane są przez osobne pliki wykonywalne. By rozpocząć śledzenie ruchu gałek ocznych, należy otworzyć plik `eye_detect.exe` np. poprzez dwukrotne kliknięcie. Otwarty zostanie terminal na lokalizacji, w której znajduje się plik wykonywalny oraz okno „eye detection", w którym, po uzyskaniu dostępu do kamery, wyświetlony zostanie obraz. Proces uruchamiania pokazano na rysunku 4.1.



Rysunek 4.1: Główne okna programu `eye_detect.exe` w momencie uruchamiania, przed połączeniem z kamerą.

Jeśli program uzyska dostęp do kamery automatycznie zacznie pobierać z niej obraz i wyszukiwać na nim twarz, oczy oraz źrenice, które oznaczy i wyświetli w oknie „eye detection". Podczas działania programu mogą otworzyć się jeszcze dwa dodatkowe okna: „Bin eyes for testing", pojawiający się w momencie wykrycia oczu oraz „Bin reflection for testing", które otworzy się w momencie włączenia trybu kalibracji zaawansowanej. „Bin eyes for testing" obrazuje binaryzację źrenic przed (po lewej) i po (po prawej) operacjach morfologicznych, z kolei „Bin reflection for testing" pokazuje binaryzację dla odbicia rogówkowego. Okna te są wizualizacją techniczną ułatwiającą dostrojenie programu i są przedstawione na rysunku 4.2.

Użytkownik może wchodzić w interakcję z systemem na dwa sposoby, poprzez interfejs graficzny w formie suwaków widoczny na rysunku 4.1, lub przez naciśnięcie odpowiedniego klawisza na klawiaturze. Nad obrazem w oknie „eye detection" widnieją dwa suwaki, których pozycje można zmieniać poprzez kliknięcie lub przeciąganie myszą. Suwak „Eye tresh" (pierwszy od góry), odpowiada progowi używanego w binaryzacji źrenic, a jego wartość ustawiona jest na 15, pozwalając na stabilne śledzenie przy dobrym oświetleniu frontalnym. Jednakże, w zależności od warunków oświetleniowych, a także preferencji związanych z stabilnością i dokładnością pomiaru, użytkownik powinien ręcznie dosto-

Rysunek 4.2: Wizualizacja obrazów binarnych wykorzystywanych w procesie wykrywania źrenic i odbicia rogówkowego.

sować wartość progu, aby zapewnić optymalną detekcję źrenic. Następny suwak „Light tresh" (drugi od góry), używany jest jedynie w przypadku kalibracji zaawansowanej, on także odpowiada progowi binaryzacji, ale w tym przypadku odbicia rogówkowego. Użytkownik wyposażony w odpowiednio intensywne źródło światła i decydujący się na tego typu kalibrację punktu odniesienia, powinien ręcznie wybrać wartość na suwaku tak, aby na obrazie binaryzacji pozostał jedynie niewielki jasny punkt.

Reszta funkcjonalności dostępna jest po naciśnięciu danego klawiszu na klawiaturze, ich wykaz prezentuje się następująco:

z (kalibracja automatyczna) – tryb kalibracji punktu odniesienia, który ustawiany jest w momencie uruchomienia programu. Pozwala na pomiar ruchu źrenic względem środka obszaru wykrytego oka.

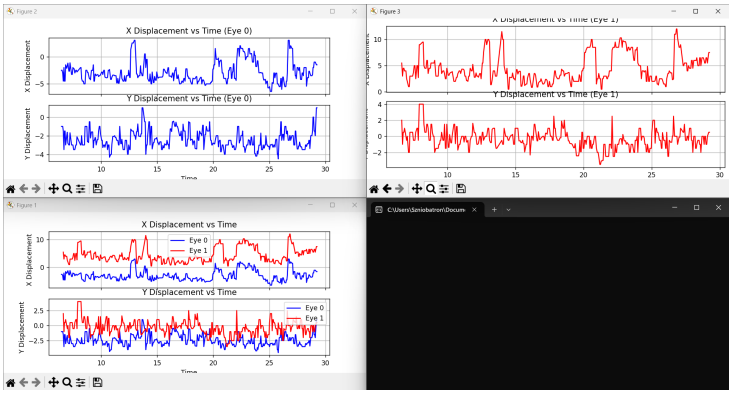
x (kalibracja zaawansowana) – tryb kalibracji punktu odniesienia na podstawie obrazu binarnego oka. Pozwala na pomiar ruchu źrenic względem wykrytego odbicia rogówkowego oka.

c (kalibracja ręczna) – tryb kalibracji punktu odniesienia na aktualną pozycję źrenic. Pozwala na pomiar ruchu źrenic względem ustalonego miejsca w obszarze wykrytego oka.

r (przełączanie trybów rysowania) – są dostępne trzy tryby rysowania wykrytych obszarów na obrazie z kamery. Pierwszy tryb jest ustawiony wraz z uruchomieniem programu obrysowując: twarz, ograniczony obszar twarzy, oczy, źrenice, środki źrenic oraz punkt odniesienia. Drugi tryb pozwala na większą przejrzystość rysując najistotniejsze elementy detekcji, czyli środek źrenic i punkt odniesienia. Trzeci tryb wyłącza rysowanie, zwracając niezmodyfikowany obraz z kamery.

q (wyłączenie programu) – jedyny poprawny sposób na zatrzymanie systemu, po naciśnięciu bezpiecznie zamyka plik tekstowy z pomiarami, zwalnia kamerę i zamyka wszystkie okna programu.

Po udanej sesji śledzenia ruchu gałek ocznych, można zwizualizować zebrane pomiary zapisane w pliku tekstowym, otwierając plik `eye_tracking_plot.exe` umieszczony w tym samym folderze co plik tekstowy. Wygenerowane zostaną trzy zestawy wykresów w osobnych oknach, przedstawiając przemieszczenie źrenicy w osi względem punktu odniesienia (dla obu oczu wspólnie i osobno). Zostały one utworzone przy pomocy Pyplot, pozwalając użytkownikowi na swobodne przybliżanie i poruszanie się po wykresach. Możliwe jest także zapisanie utworzonego wykresu w wielu popularnych rozszerzeniach. Okna z wykresami wraz z terminalem przedstawione są na rysunku 4.3.



Rysunek 4.3: Okna programu `eye_tracking_plot.exe`, przedstawiające wykresy czasowe przemieszczenia źrenicy i terminal.

W momencie zamknięcia programu, wszystkie okna, a w tym okno terminalu, zostają zamknięte. Z racji tego użytkownicy zainteresowani informacjami zwracanymi w terminalu – takimi jak informacja o błędzie, które mogą zostać zwrócone w przypadku uszkodzonego pliku, lub braku dostępu do kamery – mogą uruchomić dany plik przy użyciu terminala. Wystarczy ustawić katalog roboczy terminala na ten, w którym znajduje się plik wykonywalny, a następnie wpisać jego nazwę wraz z rozszerzeniem i nacisnąć enter. W ten sposób po zakończeniu działania programu informacje w terminalu zostaną zachowane.

4.5 Kwestie bezpieczeństwa

System do śledzenia ruchu gałek ocznych w żaden sposób nie przechowuje oraz nie nagrywa obrazów z kamery komputera, a wszelkie wykorzystywane zasoby zostają zwolnione w momencie zakończenia działania. Jedyną zapisywaną informacją są pomiary w pliku tekstowym, dostępnym w tym samym folderze co plik `eye_detect.exe`, które można z łatwością usunąć przenosząc do kosza, a następnie opróżniając go. Prezentowany pro-

gram nie korzysta także z połączenia sieciowego, wykonując wszystkie operacje lokalnie w katalogu, w którym się znajduje.

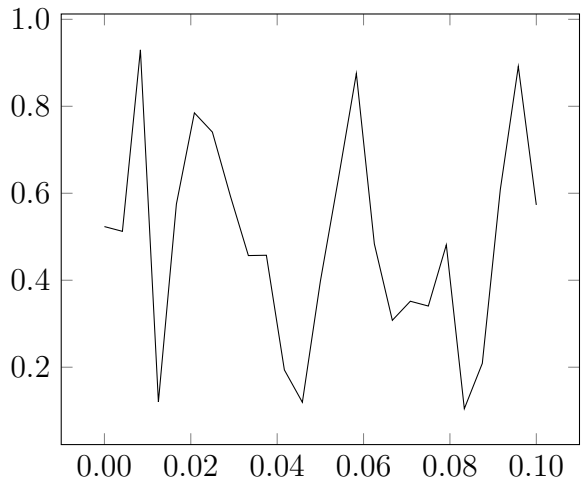
PyInstaller ostrzega, że istnieją niebezpieczeństwa związane z używaniem plików wykonywalnych w formie jednego pliku. PyInstaller rozpakowuje część swoich zasobów do tymczasowego folderu (nazwy zaczynające się od _MEI) podczas uruchamiania aplikacji. Jeśli program zostanie nieoczekiwanie przerwany np. przez awarię, lub zakończenie przez Menedżera Zadań, folder ten może nie zostać usunięty. W skrajnych przypadkach – przy częstych awariach – może dojść do gromadzenia się niepotrzebnych plików, co w długoterminowej perspektywie może zużywać miejsce na dysku. Możliwe jest ręczne usunięcie nadmiarowych plików, które w przypadku systemu Windows zazwyczaj znajdują się w folderze „Temp", można wyszukać jego lokalizację wpisując komendę `ECHO %Temp%` w wierszu poleceń. Nie należy także otwierać plików wykonywalnych utworzonych przez PyInstaller z uprawnieniami administratora, ponieważ daje to teoretyczną możliwość, że atakujący mógłby modyfikować pliki w tymczasowym folderze. Taka modyfikacja mogłaby potencjalnie prowadzić do eskalacji uprawnień [2].

4.6 Przykład działania

4.7 Scenariusze korzystania z systemu

Jeśli „Specyfikacja zewnętrzna":

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa
- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)



Rysunek 4.4: Podpis rysunku po rysunkiem.

Rozdział 5

[Właściwy dla kierunku – np. Specyfikacja wewnętrzna]

Jeśli „Specyfikacja wewnętrzna”:

- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe
- diagramy UML

Krótka wstawka kodu w linii tekstu jest możliwa, np. `int a;` (biblioteka `listings`). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys 5.1, a naprawdę długie fragmenty – w załączniku.

```
1  class test : public basic
2  {
3      public:
4        test (int a);
5        friend std::ostream operator<<(std::ostream & s,
6                                       const test & t);
7      protected:
8        int _a;
9
10 };
```

Rysunek 5.1: Pseudokod w `listings`.

Rozdział 6

Weryfikacja i walidacja

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)
- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

	metoda						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Rozdział 7

Podsumowanie i wnioski

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danych prac (rozbudowa funkcjonalna . . .)
- problemy napotkane w trakcie pracy

Bibliografia

[1] David Cortesi, Giovanni Bajo, William Caban i Gordon McMillan. *PyInstaller Manual*. 2024. URL: https://pyinstaller.org/en/stable/index.html#pyinstaller-manual (term. wiz. 31. 01. 2025).

[2] David Cortesi, Giovanni Bajo, William Caban i Gordon McMillan. *What PyInstaller Does and How It Does It*. 2024. URL: https://pyinstaller.org/en/stable/operating-mode.html#what-pyinstaller-does-and-how-it-does-it (term. wiz. 02. 02. 2025).

[3] NumPy Developers. *NumPy quickstart*. 2024. URL: https://numpy.org/doc/stable/user/quickstart.html (term. wiz. 31. 01. 2025).

[4] Redakcja Diagnostyki. *Oczopląs - co może oznaczać? Jakie badania warto wykonać?* 2023. URL: https://diag.pl/pacjent/artykuly/oczoplas-co-moze-oznaczac-jakie-badania-warto-wykonac/#2-O-czym-swiadczy-oczoplas (term. wiz. 10. 01. 2025).

[5] Andrew T. Duchowski. *Eye Tracking Methodology: Theory and Practice*. Londyn: Springer, 2007. ISBN: 978-1-84628-609-4.

[6] Python Software Foundation. *time — Time access and conversions*. 2025. URL: https://docs.python.org/3/library/time.html (term. wiz. 31. 01. 2025).

[7] Swaroop C. H. *A Byte of Python*. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2015. ISBN: 1514828146.

[8] Melanie Heck, Christian Becker i Viola Deutscher. „Webcam Eye Tracking for Desktop and Mobile Devices: A Systematic Review". W: *Proceedings of the Annual Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences, 2023. DOI: 10.24251/HICSS.2023.825.

[9] Yohan Lasorsa i Christopher Maneu. *Visual Studio Code - The Essentials*. URL: microsoft.github.io/vscode-essentials/en/ (term. wiz. 30.01.2025). Londyn: Opublikowane samodzielnie, 2024. ISBN: 9798321674161.

[10] Janelle E. MacKenzie, Madison Klarkowski, Ella M. Horton, Maryanne Theobald, Susan Danby, Lisa Kervin, Lance Barrie, Philippa K. Amery, Manesha Andradi, Simon S. Smith, Regan L. Mandryk i Daniel Johnson. „Using Psychophysiological Data to Facilitate Reflective Conversations with Children about their Player Experiences". W: *Proc. ACM Hum.-Comput. Interact.* 8.CHI PLAY (2024). DOI: 10.1145/3677112. URL: https://doi.org/10.1145/3677112.

[11] Ieva Miseviciute. *4 methods to assess Alzheimer's with eye tracking*. 2025. URL: https://www.tobii.com/resource-center/learn-articles/4-methods-to-assess-alzheimers-with-eye-tracking (term. wiz. 09. 01. 2025).

[12] Ieva Miseviciute. *How does Parkinson's disease alter visual search?* 2025. URL: https://www.tobii.com/resource-center/scientific-publications/how-does-parkinsons-disease-alter-visual-search (term. wiz. 09. 01. 2025).

[13] Alexander Mordvintsev i Abid Rahman K. *Introduction to OpenCV-Python Tutorials*. 2025. URL: https://docs.opencv.org/4.x/d0/de3/tutorial_py_intro.html (term. wiz. 31. 01. 2025).

[14] Imię Nazwisko i Imię Nazwisko. *Tytuł strony internetowej*. 2021. URL: http://gdzies/w/internecie/internet.html (term. wiz. 30. 09. 2021).

[15] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu konferencyjnego". W: *Nazwa konferecji*. 2006, s. 5346–5349.

[16] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. „Tytuł artykułu w czasopiśmie". W: *Tytuł czasopisma* 157.8 (2016), s. 1092–1113.

[17] Imię Nazwisko, Imię Nazwisko i Imię Nazwisko. *Tytuł książki*. Warszawa: Wydawnictwo, 2017. ISBN: 83-204-3229-9-434.

[18] pandas. *About pandas*. 2025. URL: https://pandas.pydata.org/about/index.html (term. wiz. 31. 01. 2025).

[19] Ken Schwaber i Jeff Sutherland. *The 2020 Scrum Guide™*. 2020. URL: https://scrumguides.org/scrum-guide.html (term. wiz. 31. 01. 2025).

[20] iMotions Science Team. *iMotions Publications: 2023 Report*. 2024. URL: https://imotions.com/wp-content/uploads/brochures/Research%20report%202024.pdf (term. wiz. 20. 01. 2025).

[21] iMotions Science Team. *Smart Eye AI-X*. 2025. URL: https://imotions.com/products/hardware/smart-eye-ai-x/#product-specifications (term. wiz. 20. 01. 2025).

[22] iMotions Science Team. *Who we are: about us*. 2025. URL: https://imotions.com/about-us/ (term. wiz. 26. 01. 2025).

[23] Divya Seernani, Morten Mosbaek Pedersen i Kerstin Wolf. *WebET 3.0 -Validation Study Report*. Spraw. tech. 2023. DOI: 10.13140/RG.2.2.32959.07849.

[24] PassMark® Software. *About PassMark Software*. 2025. URL: https://www.passmark.com/about/index.php (term. wiz. 01. 02. 2025).

[25] PassMark® Software. *CPU Benchmarks: AMD Ryzen 7 7730U*. 2025. URL: https://www.cpubenchmark.net/cpu.php?cpu=AMD+Ryzen+7+7730U&id=5215 (term. wiz. 01. 02. 2025).

[26] PassMark® Software. *CPU Benchmarks: CPU Test Information*. 2025. URL: https://www.cpubenchmark.net/cpu_test_info.html (term. wiz. 01. 02. 2025).

[27] PassMark® Software. *CPU Benchmarks: Intel Xeon E5-2660 v2 @ 2.20GHz*. 2025. URL: https://www.cpubenchmark.net/cpu.php?id=2184&cpu=Intel%20Xeon%20E5-2660%20v2%20@%202.20GHz (term. wiz. 01. 02. 2025).

[28] PassMark® Software. *CPU Benchmarks: Intel Xeon E5-4650 @ 2.70GHz*. 2025. URL: https://www.cpubenchmark.net/cpu.php?id=1225&cpu=Intel%20Xeon%20E5-4650%20@%202.70GHz (term. wiz. 01. 02. 2025).

[29] OpenCV team. *About*. 2025. URL: https://opencv.org/about/ (term. wiz. 30. 01. 2025).

[30] The Matplotlib development team. *Pyplot tutorial*. 2025. URL: https://matplotlib.org/stable/tutorials/pyplot.html (term. wiz. 31. 01. 2025).

[31] Tobii. *Eye tracking in neurology and psychiatry research*. 2025. URL: https://www.tobii.com/resource-center/reports-and-papers/eye-tracking-neurology-and-psychiatry-research (term. wiz. 09. 01. 2025).

[32] Tobii. *Psychology and neuroscience use cases*. 2025. URL: https://www.tobii.com/products/eye-trackers/screen-based/psychology-neuroscience-use-cases (term. wiz. 09. 01. 2025).

Dodatki

Spis skrótów i symboli

DNA kwas deoksyrybonukleinowy (ang. *deoxyribonucleic acid*)

MVC model – widok – kontroler (ang. *model–view–controller*)

N liczebność zbioru danych

μ stopnień przyleżności do zbioru

$\mathbb{E}$ zbiór krawędzi grafu

$\mathcal{L}$ transformata Laplace'a

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown number of clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
          iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
          epsilon set — you should set either number of iterations
          or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

Spis tabel