Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej
Instytut Fizyki

Kamil Pleskot
nr albumu: 267567

Praca Inżynierska na kierunku Informatyka Stosowana

Eyetracker: aparatura i oprogramowanie

Opiekun pracy dyplomowej
dr hab. Jacek Matulewski
Zakład Mechaniki Kwantowej

Toruń 2017

Pracę przyjmuję i akceptuję

.....

data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

.....

data i podpis pracownika dziekanatu

Dziękuję mojemu promotorowi dr hab. Jackowi Matulewskiemu za cenne porady, a także za życzliwą i miłą atmosferę sprzyjającą pisaniu pracy dyplomowej.

Spis treści

1. Wstęp

Moją inspiracją przy wyborze tematu pracy inżynierskiej był film „Motyl i skafander" w reżyserii Juliana Schnabla. Główny bohater utworu w wieku 43 lat zostaje sparaliżowany na skutek udaru mózgu. Cierpi na zespół zamknięcia (ang. *locked-in syndrom*). Jego jedyną formą komunikacji z otoczeniem pozostaje jego oko. Terapeutka przedstawia prymitywną metodę konwersacji - podczas gdy osoba rozmawiająca z pacjentem odczytuje litery ułożone według częstotliwości ich występowania, pacjent mruga powieką w odpowiednim momencie, a składane w ten sposób litery tworzą wyrazy i zdania.

Obecnie możemy pomóc takim osobom wykorzystując okulografy, czyli urządzenia śledzące ruch gałki ocznej. Narzędzia te są szeroko stosowane w takich dziedzinach jak psychologia, medycyna, ergonomia, interakcja człowiek-komputer, czy marketing. Niestety są to drogie urządzenia Cena nie zawsze wynika z tego, że producenci stosują najlepsze dostępne na rynku kamery, lecz przede wszystkim z powodu udoskonalanego przez lata oprogramowania.

Celem projektu jest stworzenie aparatury pozwalającej śledzić ruch jednej gałki ocznej. Urządzenie powinno być zbudowane niewielkim kosztem oraz niezbyt skomplikowane– tak, aby można było je w łatwy i szybki sposób odtworzyć. Jeżeli kamera pomimo słabej jakości będzie w stanie dokładnie rejestrować obszar oka, to potwierdzi moje przypuszczenie, że jednym z głównych czynników wpływających na cenę profesjonalnych sprzętów jest oprogramowanie. Wiedzę niezbędną do stworzenia takiego urządzenia uzyskałem czytając liczne artykuły naukowe [1-5].

Zasadniczą częścią projektu będzie również stworzone przeze mnie oprogramowanie, które pozwoli pobrać obraz z aparatury, a następnie umożliwi jego obróbkę i wyznaczenie punktu spojrzenia.

W przypadku konieczności zastosowania zewnętrznych bibliotek programistycznych, priorytet zostanie skierowany na darmowe otwarte oprogramowanie (ang. *open source*), by na każdym kroku redukować koszt finalny.

Planowana funkcjonalność programu:

- ustalenie lokalizacji źrenicy na obrazie z kamery i jej śledzenie,
- środowisko do przeprowadzania testów,

- mapowanie położenia źrenicy (układ współrzędnych kamery) na położenie punktu spojrzenia (układ współrzędnych ekranu),
- wielofunkcyjność (opcje dostosowania),
- łatwe w obsłudze.

Sprzęt i oprogramowanie stworzą uzupełniającą się całość, platformę, która jest w stanie realnie ułatwić życie osobom sparaliżowanym, a przy tym niezamożnym. Moja praca ma również na celu pomóc osobom zainteresowanym tematyką okulografii, lecz nieposiadającym odpowiedniej wiedzy technicznej.

Publikacje opisujące tematykę śledzenia ruchu oka [6-8], często dostępne są tylko w językach obcych, głównie w języku angielskim, często w postaci płatnych artykułów naukowych. Niewiele z nich opisuje szczegóły umożliwiające konstrukcję własnego eyetrackera. W związku z tym postanowiłem napisać pracę w języku polskim, aby po pierwsze promować samą dziedzinę okulografii, a po wtóre aby ułatwić zrozumienie szczegółów technicznych, które są z nią związane.

2. Konstrukcja Aparatury

Celem tego rozdziału jest przedstawienie krok po kroku konstrukcji aparatury służącej do śledzenia ruchów gałki ocznej. Do stworzenia urządzenia postanowiłem wybrać części o najniższych możliwych cenach, by umożliwić każdemu jego odtworzenie. Tworząc, a następnie opisując własny eyetracker, w znacznej mierze opierałem się na artykule Michała Kowalika [5] oraz artykułach [1-4]. Sprzęt wraz z oprogramowaniem opisanym w rozdziale trzecim umożliwi śledzenie ruchu źrenicy oraz ustalanie punktu spojrzenia na ekranie, na który patrzy użytkownik.

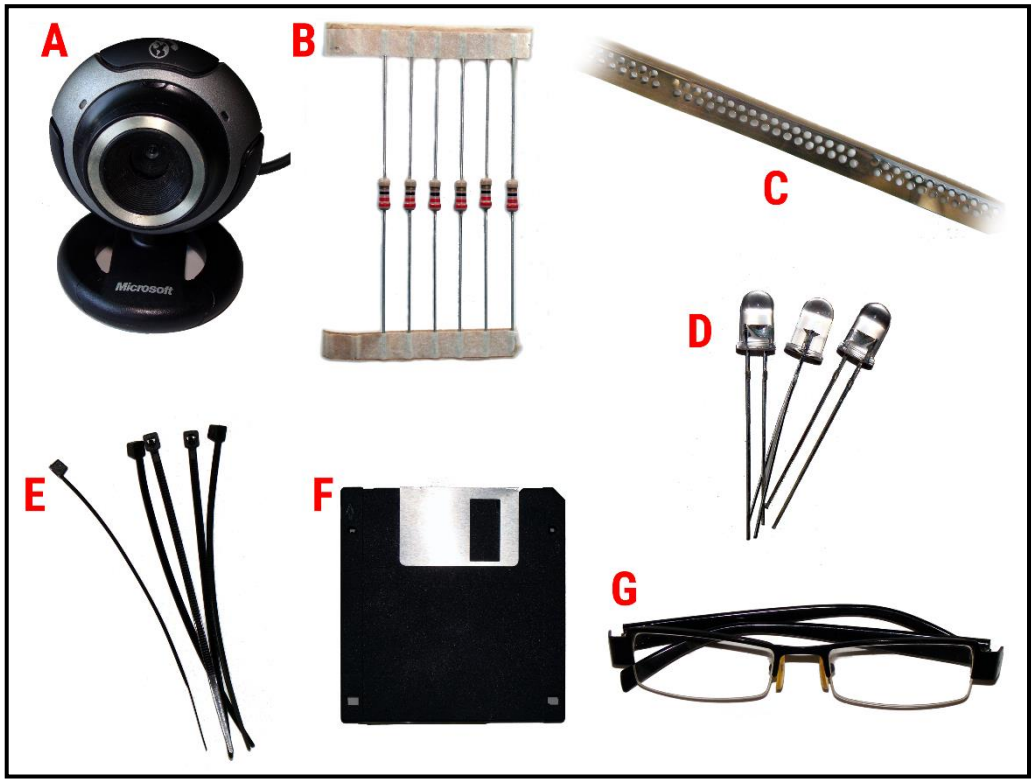
2.1. Potrzebne materiały

Głównym elementem mojego układu jest kamera internetowa firmy Microsoft VX-3000 [9] wyposażona w matrycę CMOS VGA. Kamera pozwala na nagrywanie filmów w rozdzielczości 640×480 pikseli. Układ zostanie dodatkowo wzbogacony o serię diod (rys. 1.B.) emitujących światło podczerwone w celu doświetlenia scenerii oraz uwypuklenia cech morfologicznych obszaru oka badanej osoby. Oprawki okularów zostaną połączone z układem w całość za pośrednictwem kątownika aluminiowego (rys. 1.C.). Kątownik posiada otwory ułatwiające montaż. Ze względu na minimalizowanie ceny urządzenia i konieczność ograniczenia jego wagi, korzystam tylko z jednej kamery zamontowanej na kątowniku, a więc śledzony będzie ruch tylko jednej gałki ocznej.

Tabela (Tab.1.) przedstawia dokładną listę potrzebnych elementów w celu konstrukcji naszego układu. Dodatkowo potrzebne będą narzędzia widoczne na rys. 2.

Tabela 1. **Lista elementów potrzebnych do skonstruowania aparatury**

Nazwa elementu	ilość	Przykładowa cena zakupu w przeliczeniu na sztukę
Kamera internetowa Microsoft VX-3000	1	10 zł
Rezystor 22R 0.25W 5%	1	1 gr
Kątownik aluminiowy	1	2zł
Diody LED 5mm IR 940nm	2~	30 gr
Zaciski plastikowe	10~	-
Dyskietka	1	3zł
Oprawka okularów	1	10zł



Rysunek 1. Elementy składowe Aparatury: A) kamera internetowa Microsoft VX-3000, B) rezystor, C) kątownik aluminiowy, D) dioda IR, E) zaciski plastikowe, F) dyskietka, G) oprawka okularów

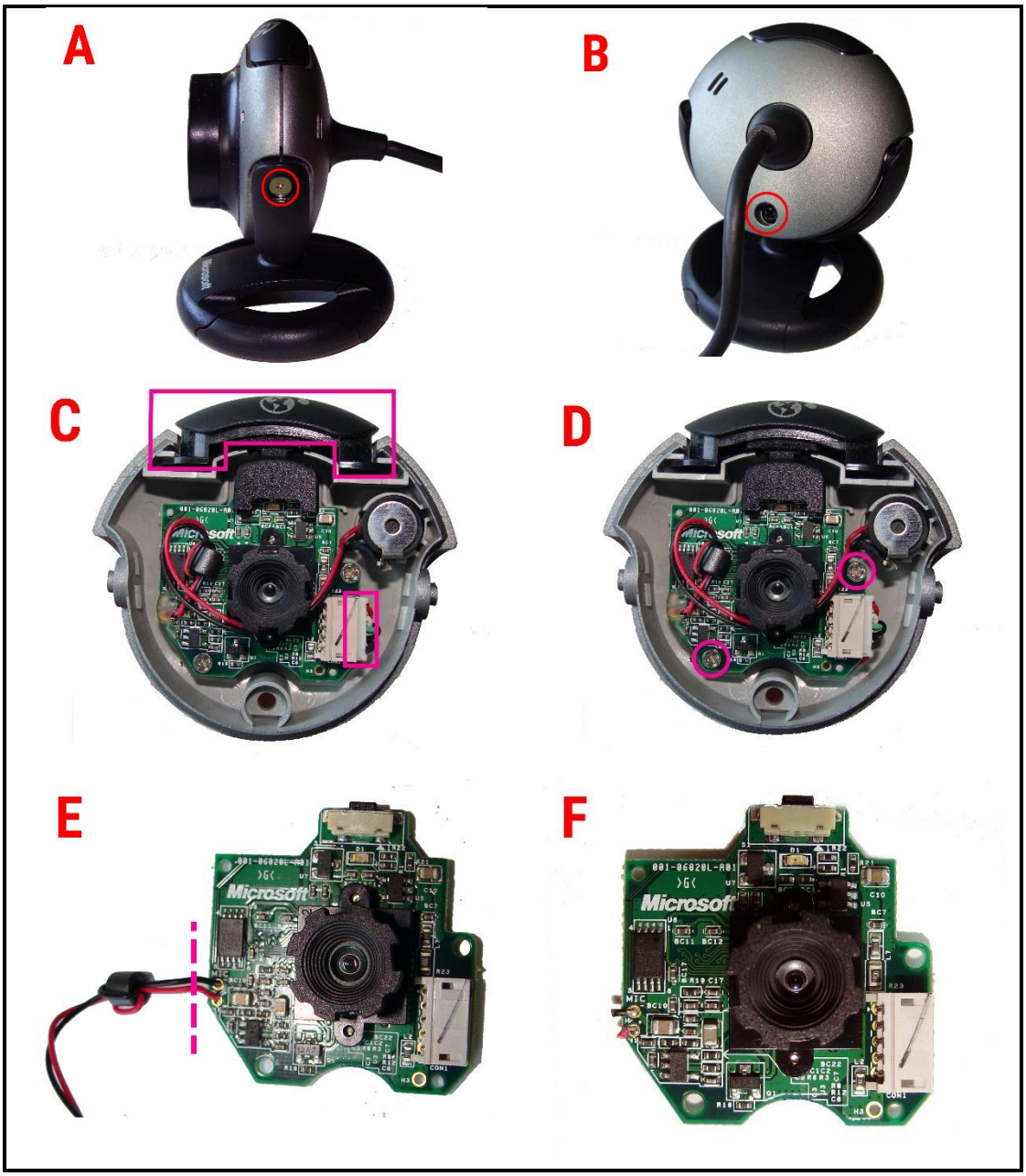
Rysunek 2. Niezbędne narzędzia: A) lutownica, B) cyna, C) śrubokręt, D) ostry nóż

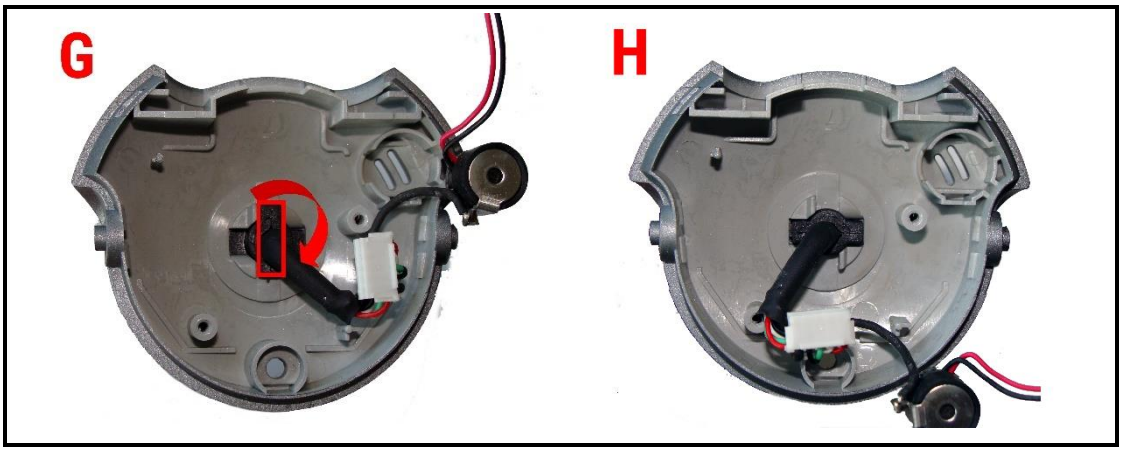
2.2. Demontaż kamery

Prace należy rozpocząć od rozłożenia modelu kamery, co pokażę na przykładzie kamery Microsoft VX-3000 (rys. 1.A.). Podobnie należy postępować w przypadku jej bliźniaczej konstrukcji tj. kamery Microsoft VX-1000 [10].

Pierwszym krokiem jest usunięcie dwóch śrub montażowych umieszczonych po przeciwnych stronach obudowy (rys. 3.A.). Łącząca one górną część obudowy z podstawką. Po wykonaniu tej czynności odczepiamy dolną część obudowy kamery. Odkręcamy główną śrubkę (rys. 3.B.) dzięki czemu jesteśmy w stanie dostać się do wnętrza obudowy. Wyjmujemy przycisk oraz odpinamy przewód od płyty drukowanej (rys. 3.C.). Wykręcamy kolejne dwie śrubki (rys. 3.D.), które łączą układ scalony z obudową. Mój model kamery wyposażony jest w mikrofon do rozmów przez Internet. Jest on niepotrzebny w przypadku okulografu, dlatego też możemy go usunąć poprzez odlutowanie odpowiednich przewodów lub ich odcięcie (rys. 3.E.). Następnie wyjmujemy płytę drukowaną i odkładamy ją w bezpieczne miejsce. Ostatnim krokiem jest wyswobodzenie przewodu USB ze środka obudowy. W tym celu zabezpieczenie ustawione w pozycji zablokowanej (rys. 3.G) przesuwamy o 90° w prawo do pozycji odblokowanej (rys. 3.H)

Do wyodrębnionego układu (rys. 3.F.) wpinamy przewód USB aby sprawdzić czy kamera jest nadal sprawna. Podłączamy kamerę do komputera. **Uwaga!,** kamery po podłączeniu nie możemy dotykać, gdyż może to spowodować spięcie i zepsuć układ. Jeżeli kamera nie działa, upewnij się czy niezbędne sterowniki do jej prawidłowej pracy zostały zainstalowane.



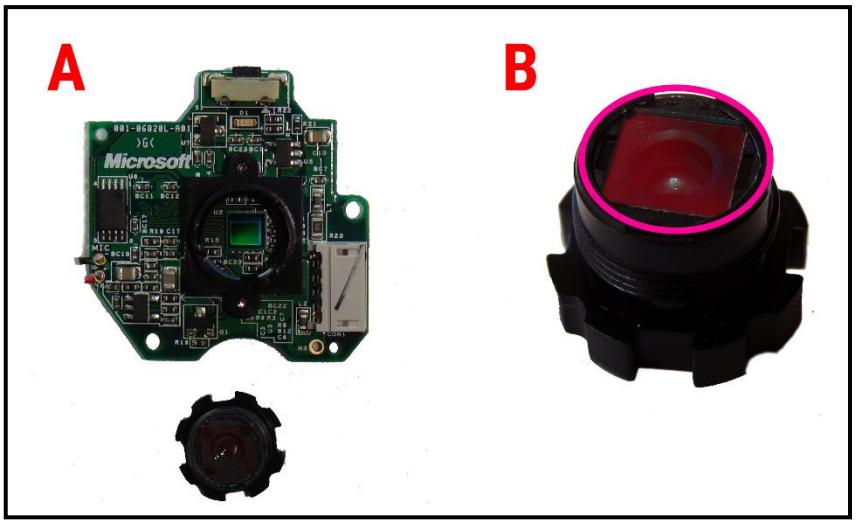


Rysunek 3. Etapy demontażu kamery. A) jedna z dwóch śrubek mocujących podstawkę kamery, B) główna śruba z tyłu obudowy, C) przycisk oraz port przewodu USB, D) śruby mocujące płytę drukowaną do obudowy, E) przykładowe miejsce przecięcia przewodu, F) wymontowany układ scalony, G) mechanizm zabezpieczający przewód w pozycji zamkniętej, H) mechanizm zabezpieczający przewód w pozycji otwartej

2.3. Modyfikacja fabrycznej kamery

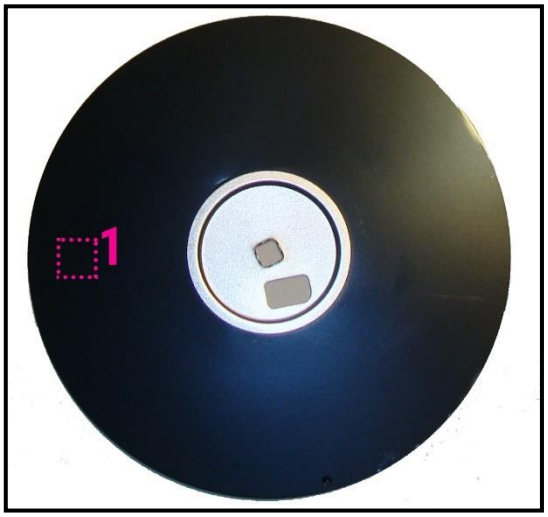
Wykorzystywana przeze mnie kamera, nie została stworzona w celu badania ruchu gałki ocznej, dlatego wymaga serii modyfikacji, mających na celu poprawę dokładności rejestrowania obszaru oka.

Modyfikację rozpoczynamy od demontażu soczewki (rys. 4.A.) oraz filtru światła widzialnego (rys. 4.B.) uważając by nie porysować znajdującej się pod spodem soczewki. Filtr jest zbędny gdyż nie potrzebujemy dodatkowych informacji o kolorach, które mogą chociażby wydłużyć czas przetwarzania każdorazowo odebranej klatki obrazu. W związku z tym nie trzeba przejmować się jeżeli filtr podczas usuwania ulegnie ukruszeniu (może się nawet całkowicie połamać).



Rysunek 4. A) Zdemontowana soczewka i B) filtr światła znajdujący się na soczewce (w okręgu)

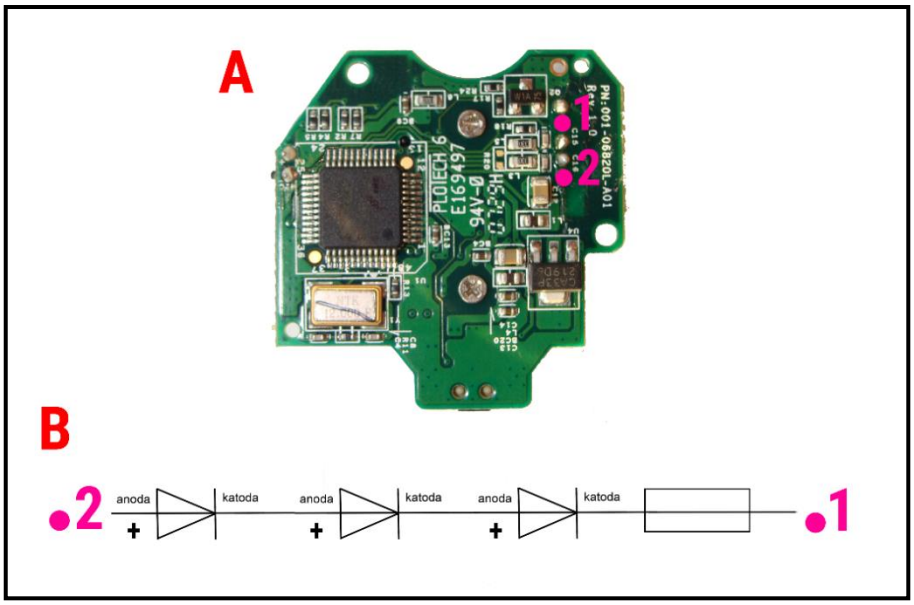
Na jego miejsce montujemy filtr blokujący promienie podczerwieni, który tanim kosztem jesteśmy w stanie uzyskać odpowiednio docinając nośnik dyskietki (rys. 5.1.). Alternatywą dla dyskietki jest klisza fotograficzna. Nowy element możemy przymocować za pomocą np. kleju.



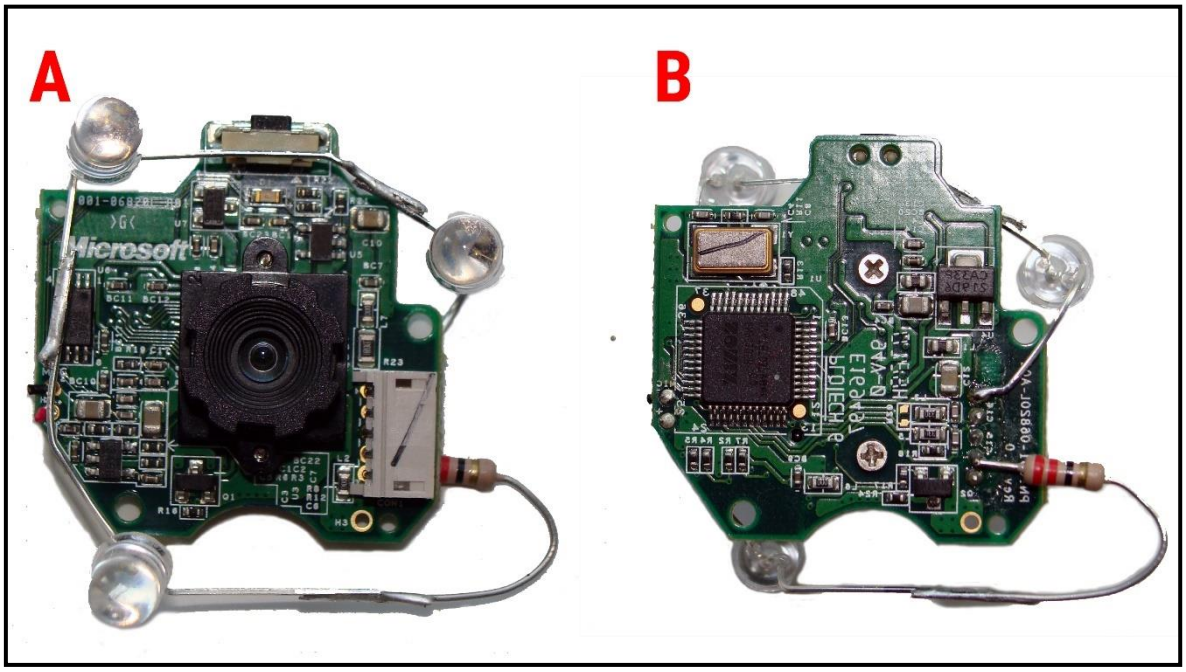
Rysunek 5. Nośnik dyskietki. 1) przykładowy obszar wycięcia filtru podczerwieni

Kolejną modyfikacją jest dodanie do układu scalonego serii diod podczerwieni, doświetlających obszar oka. Światło podczerwone jest niewidoczne dla ludzkiego wzroku, a zatem nie będzie go razić. Jednak kamera, w przeciwieństwie do człowieka, rejestruje te promienie świetlne, co może powodować nadmiernie prześwietlony obraz. Właśnie dlatego stosujemy filtr podczerwieni, który zapobiegnie takiej sytuacji.

Do montażu diod wykorzystuję dwa piny układu scalonego: 5V oraz uziemienie (rys. 6.A.1 i 6.A.2). Elementy te lutuję bezpośrednio do płyty drukowanej. Montaż należy wykonać według schematu z rys. 6.B. W razie obawy uszkodzenia płyty, można przylutować diody do przewodu USB (żyła czerwona - 5V, żyła czarna - uziemienie.



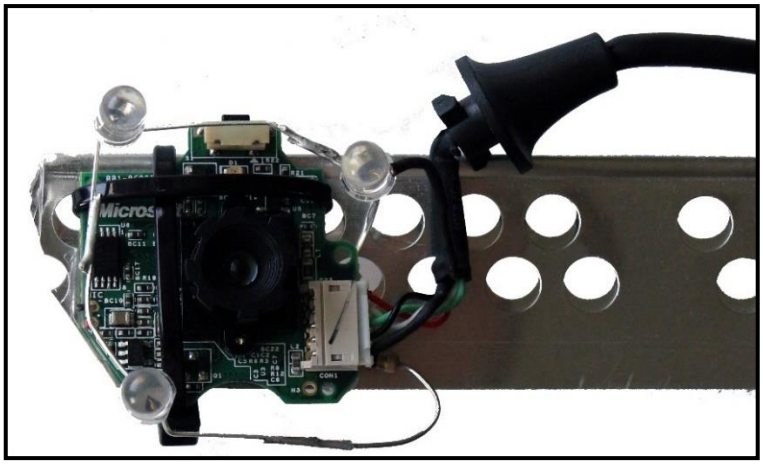
Rysunek 6. A) Układ scalony z zaznaczeniem 1) pin uziemienie, 2) pin zasilający 5V. B) Schemat elektryczny lutowanego układu.



Rysunek 7. Układ scalony po modyfikacji: A) front, B) tył

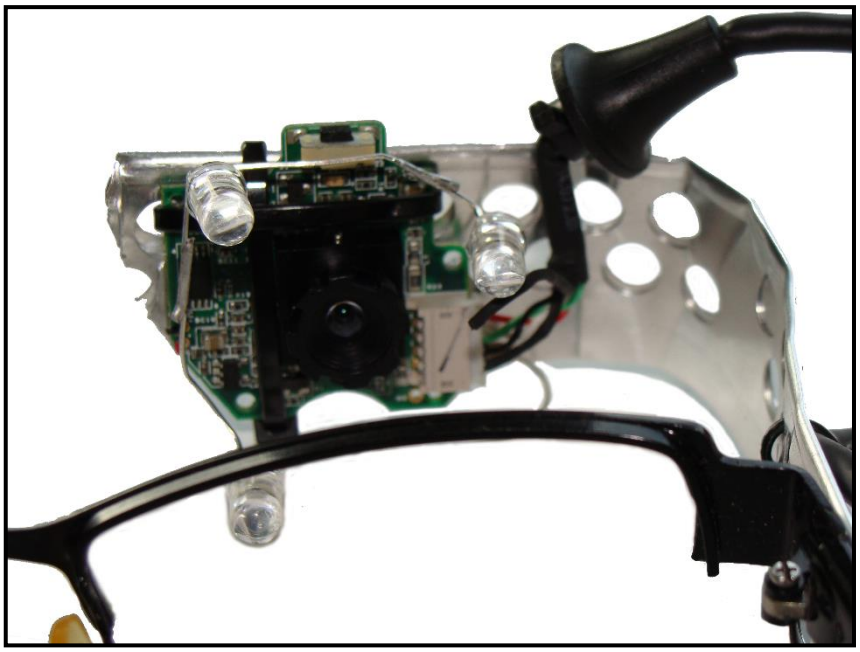
2.4. Montaż części składowych

Po przygotowaniu układu scalonego przystępujemy do kolejnego etapu, jakim jest montaż wszystkich potrzebnych części w całość. Rozpoczynamy od umocowania płytki drukowanej do jednego z końców aluminiowego kątownika za pomocą śrubek bądź też plastikowych zacisków (rys. 8.). **Uwaga!,** końcówkę należy wpierw odizolować np. owijając ją plastikową torebką.



Rysunek 8. Proponowana metoda montażu układu scalonego do aluminiowego kątownika za pomocą plastikowych opasek

Drugi koniec kątownika przycinamy w odpowiednim miejscu i łączymy z zausznikiem oprawki okularów również za pomocą plastikowych opasek. Część zakończoną płytką drukowaną doginamy tak, aby po założeniu okularów sensor kamery znajdował się w takiej pozycji, aby osie oka (przy patrzeniu przed siebie) i kamery były równoległe (rys. 9.).



Rysunek 9. Przykład dogięcia aluminiowego kątownika z kamerą

Rysunek 10. Finalna postać jednego z prototypów

3. Oprogramowanie

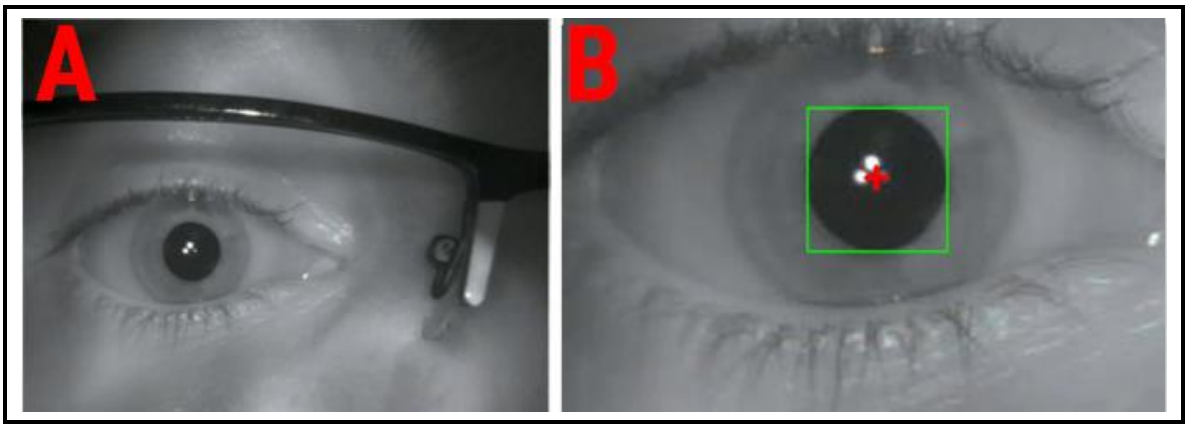
Oprogramowanie analizujące obraz oka z kamery jest równie ważnym elementem eyetrackera, jak opisana wyżej aparatura. Również w jego przypadku moim celem jest osiągnięcie celu w jak najprostszy i najmniej kosztowny sposób. Dlatego do analizy obrazu użyję gotowej biblioteki OpenCV, a dokładnie jej nakładki dla platformy .NET – EmguCV. W rozdziale 7 zaprezentuje wyniki testów, które pokazują, że stworzone w ten sposób oprogramowanie z powodzeniem pozwala na uzyskanie zadowalających wyników.

3.1. Technologia

Program został napisany w języku programowania C# z wykorzystaniem trzech zewnętrznych bibliotek:

- *EmguCV* [11] – wieloplatformowa biblioteka, umożliwiająca analizę obrazu pozyskanego z kamery. EmguCV zostało wyposażone w zbiór klas pozwalających na uruchomienie z poziomu platformy .NET wszystkich metod z niezarządzanej biblioteki OpenCV, która została napisana w języku C.
- *DirectShowLib* [12] – projekt wykorzystuje tylko jedną metodę z tej biblioteki, która pozwala ustalić listę zainstalowanych urządzeń video wraz z odpowiadającymi im nazwami kamer w danym systemie. Bez wykorzystania biblioteki DirectShow udało mi się jedynie pobrać liczbę zainstalowanych urządzeń video (bez ich nazw).
- *MathNet.Numerics* [13] – biblioteka dostarczająca metod i algorytmów służących do obliczeń numerycznych, m.in. do operacji na macierzach, których użyłem w projekcie.

Powyższe element można w prosty sposób uwzględnić w tworzonym projekcie wykorzystując w tym celu wbudowaną funkcję menadżera dodatków NuGet [14], wbudowanego w środowisko programistyczne Visual Studio.

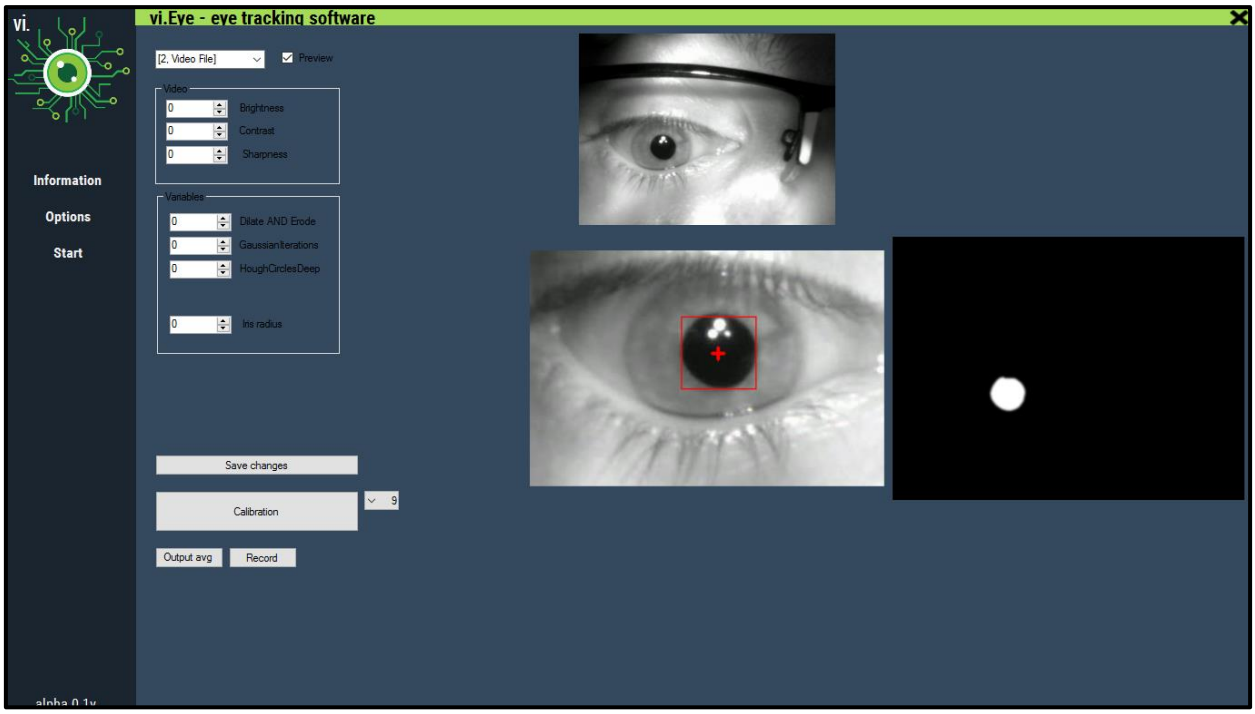


Rysunek 11. A) rejestrowany przez kamerę obszar. Na obrazie widać, że jedna z diod świeci słabiej, nie ma to wpływu na działanie aparatury - jest to głównie spowodowane nadmiernie odgiętą diodą, B) wynik przeprowadzenia obrazu przez algorytm lokalizacji źrenicy

Program został wyposażony w czytelny interfejs graficzny zbudowany z pomocą kontrolek *Windows Forms*. Rozwiązanie to umożliwia użytkownikowi proste nawigowanie w obrębie dostępnych opcji.

Funkcje aplikacji to:

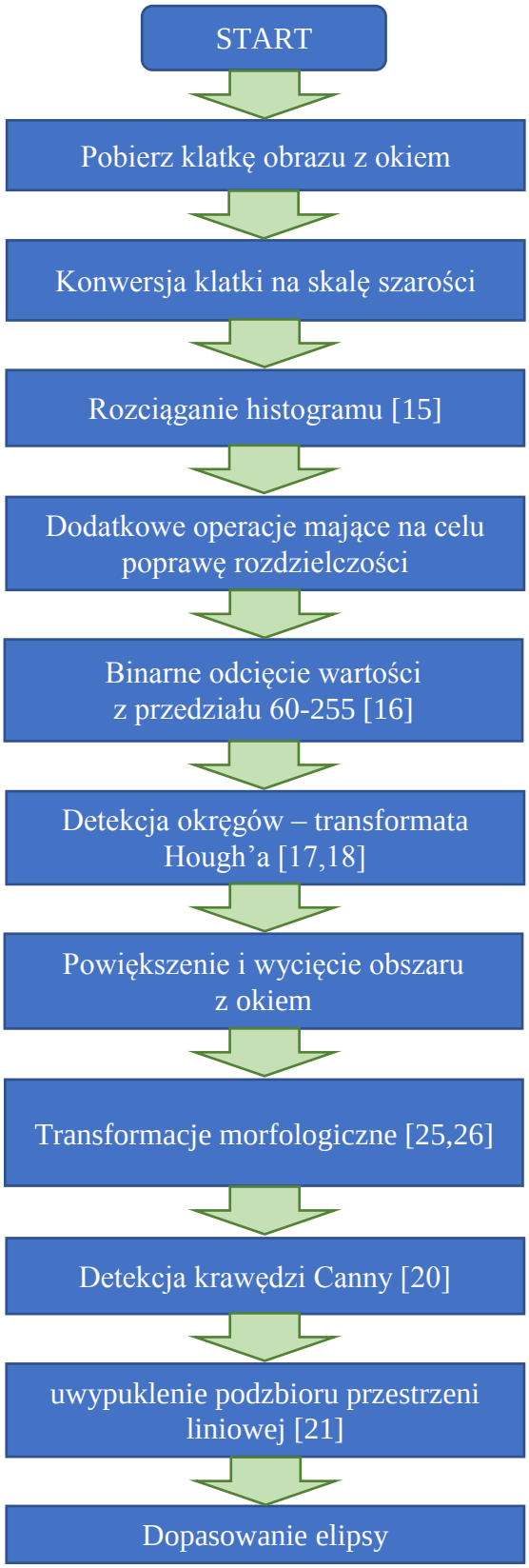
- podgląd na żywo obrazu transmitowanego z uprzednio wybranej kamery,
- możliwość nagrywania filmów oraz ich odtwarzania,
- konfigurację ustawień odbieranego sygnału wideo:
    - jasność,
    - kontrast,
    - ostrość,
- rozbudowany algorytm pozwalający śledzić źrenicę (zob. opis w rozdziale 4.),
- kalibrację urządzenia (zob. opis w rozdziale 5.),
- przedstawienie w postaci graficznej współrzędnych punktu spojrzenia (we współrzędnych ekranu).



Rysunek 12. Zrzut prezentujący zakładkę *Opcji* aplikacji testowej.

4. Analiza obrazu z kamery

4.1. Algorytm wyszukiwania fragmentu obrazu z okiem



4.2. Definicje

- **Rozciąganie histogramu** – „Rozciągnięcie histogramu powoduje zmianę jasności pikseli wg. jakiejś funkcji np. liniowej, wykładniczej lub logarytmicznej (...) Spowoduje to wykorzystanie w obrazie wszystkich możliwych wartości jasności, a zatem poprawi rozróżnialność elementów obrazu.” [22]
- **Binarne odcięcie wartości** – metoda wykonywana dla każdego piksela obrazu, która sprawdza czy znajduje się on w przedziale parametrów do odcięcia (np. jasność z przedziału od 60 do 255). Jeżeli warunek jest spełniony, piksel jest zmieniany na przezroczysty lub o ustalonym kolorze tła.
- **Transformata Hough’a służąca do wykrywania okręgów** [23]. – okrąg o promieniu R i środku (a,b) może zostać opisany poniższym równaniem parametrycznym:

$$x = a + Rcos(\theta)$$
$$y = b + Rsin(\theta) \quad \textbf{(1)}$$

Wyznaczając wszystkie pary wartości (x,y) dla wszystkich wartości kąta $\theta \in (0,360)$ wyznaczamy współrzędne punktów należących do okręgu. W przypadku analizy obrazu w celu znalezienia promienia i środka okręgu, zadanie jest odwrotne: znając położenie wielu punktów (x,y) będących współrzędnymi pikseli na okręgu staramy się znaleźć współrzędne środka i promień tj. parametry (a, b, R) opisujące okręgi na tym obrazie, tworzą trójwymiarową tablicę (potocznie nazywaną akumulatorem). Sprawdzanie dla każdego piksela wszystkich możliwych promieni oraz kątów θ niesie ze sobą ogromny koszt obliczeniowy. W związku z tym OpenCV/EmguCV wykorzystuje sprytne rozwiązanie – metodę gradientu Hougha (ang. *Hough gradient method*). Metoda polega na tym, że zdjęcie najpierw poddawane jest obróbce znajdowania krawędzi oraz naniesienia maski zawierającej kolor biały dla pikseli należących do krawędzi, a kolor czarny dla pozostałych. Efektem jest czarny obraz z niezerowymi pikselami opisującymi krawędzie. Następnie dla każdego zapisanego piksela, wyznaczamy lokalny gradient. Zastosowanie tej metody powoduje zmniejszenie akumulatora o jeden wymiar, tym samym zmniejsza się liczba operacji.

- **Transformacje morfologiczne**. Zastosowałem dwie podstawowe operacje:
  - Erozja – „pocienianie elementów zawartych na obrazie, powoduje zmniejszanie obiektów, a nawet znikanie małych elementów.” [24] Erozję można zdefiniować następującym wzorem:

$$A \ominus B = \bigcap_{b \in B} (A + b) \quad \textbf{(2)}$$

    Erozja jest przecięciem (częścią wspólną) wszystkich translacji obiektu *A* o elementy zbioru *B* [26,31].
  - Dylacja – pogrubienie „elementów obrazu, powoduje zwiększenie obiektów, znikanie dziur w obiektach i zanikanie detali.” [24] Dylację można zdefiniować następującym wzorem:

$$A \oplus B = \bigcup_{b \in B} (A + b) \quad \textbf{(3)}$$

    Dylacja jest połączeniem wszystkich przesunięć obiektu *A* o elementy zbioru *B* [25,31].

    Gdzie [24]:
    - *A* jest obrazem,
    - *B* elementem strukturalnym - wykorzystywany jako przestawny obszar. Element strukturalny jest przemieszczany po całym obrazie tak, by analizowany piksel był punktem centralnym elementu strukturalnego. Element strukturalny może przybrać dowolny kształt i wielkość, zawierać dowolną kombinację wartości 0 i 1.
    - *b* to piksel elementu strukturalnego.

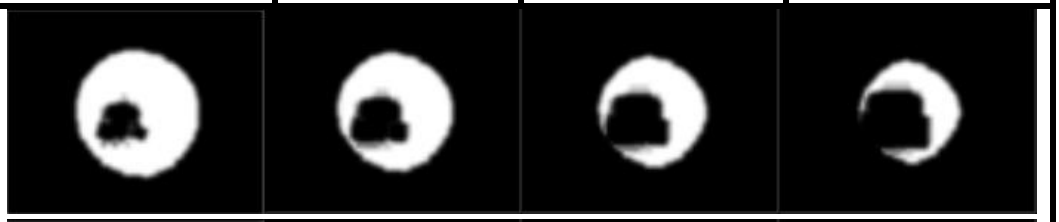
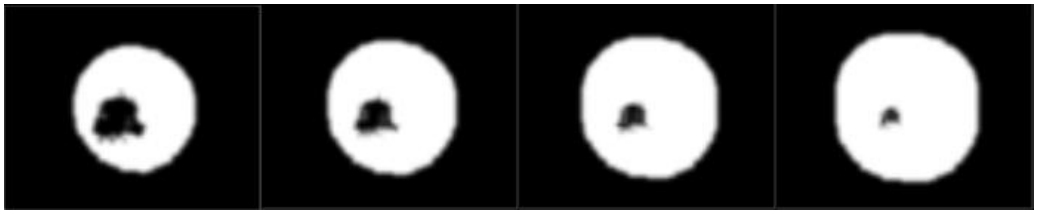
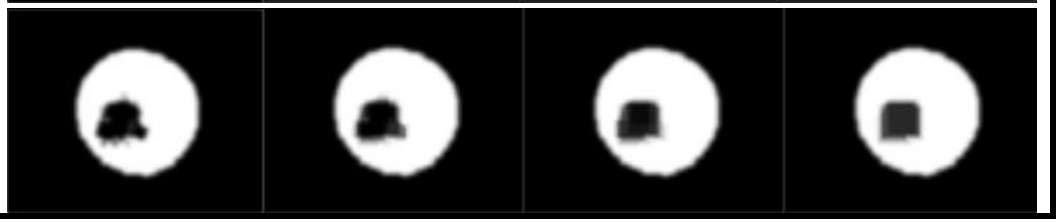
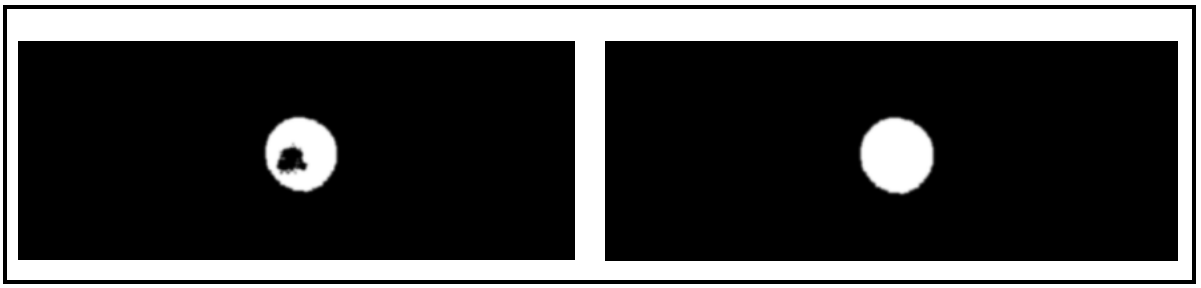
	Oryginał	1 iteracja	2 iteracja	3 iteracja
Erozja				
Dylacja				
Dylacja oraz Erozja				

Tabela 2. Zastosowanie erozji oraz dylacji.

Dla uzyskania lepszego efektu, operacje morfologiczne wykonywane są wielokrotnie. Głównym powodem, dla którego ja je stosuję, jest wypełnienie luki spowodowanej odbiciem promieni podczerwieni w zarejestrowanej źrenicy na potrzeby wykrywania jej położenia za pomocą algorytmu opartego na transformacie Hough'a.



Rysunek 13. Z lewej) źrenica przed transformacjami morfologicznymi, Z prawej) źrenica po wielokrotnych transformacjach morfologicznych.

- **Detekcja krawędzi Canny –** metoda wykorzystuje algorytm *Canny*, składający się z czterech zasadniczych kroków:
  - redukcja szumów, za pomocą filtru Gaussa (rozmycie obrazu),
  - Szukanie natężenia gradientu obrazu - detekcja poziomych, pionowych oraz przekątnych krawędzi na wygładzonym obrazie,

- usuwanie niemaksymalnych pikseli - erozja krawędzi w sposób zapewniający ich ciągłość. Efektem jest ciągła linia złożona z pojedynczych pikseli.
- Progowanie z histerezą – progowanie ma na celu usunięcie nieistotnych krawędzi, które mają nachylenie (stromość) poniżej ustawionego progu. Progowanie z histerezą powoduje, że do już wykrytych krawędzi są dołączane następne piksele mimo spadku nachylenia, aż do osiągnięcia dolnego progu wykrywania. Takie postępowanie zapobiega dzieleniu krawędzi w miejscach słabszego kontrastu.

- **Znajdowanie otoczki wypukłej podzbioru przestrzeni liniowej**. „Niech Q będzie skończonym zbiorem punktów na płaszczyźnie. Najmniejszy zbiór wypukły taki, że każdy punkt zbioru Q znajduje się albo w jego wnętrzu albo na brzegu, nazywamy otoczką wypukłą zbioru Q (ang. *convex hull*).” [27]. W celu zlokalizowania otoczki posłużę się gotową metodą `ConvexHull` z wykorzystywanej biblioteki EmguCV, która wykorzystuje algorytm *Sklansky'ego*.
- **Test Grubbsa –** służy do wykrycia wyniku, który został obarczony błędem grubym, tj. znalezienia i wykluczenia wartości znacznie odstających (ang. *outlier*) od pozostałych wyników w zbiorze. Błędy grube mogą powstawać na skutek ruchu głowy pacjenta podczas badania, nie poprawnego zdefiniowania obszaru oka przez program.

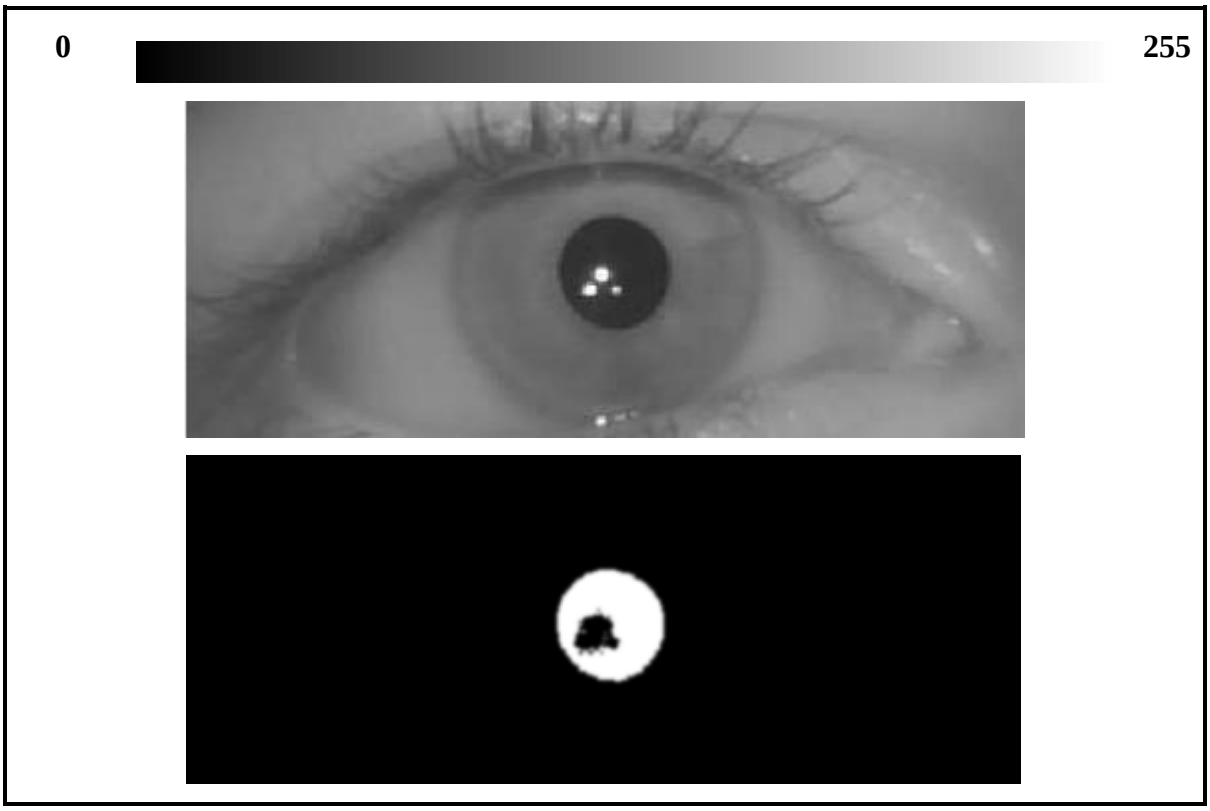
4.3. Omówienie algorytmu

Wykorzystywane przeze mnie metody wymagają by obraz wejściowy zawierał jedynie informacje o jasności pikseli (w skali szarości), dlatego już na samym początku dokonuje jego konwersji (metoda `convert`). W konsekwencji w dalszym opisie przez piksele będę rozumiał tylko jedną wartość z zakresu od 0 do 255.

W celu uwypuklenia różnicy w barwach pomiędzy z jednej strony jasną twardówką i skórą, a z drugiej strony ciemną źrenicą, wykorzystuję metodę rozciągania histogramu (metoda `_EqualizeHist` [15]). Po poprawieniu rozdzielczości obrazu dokonuję binarnego odcięcia wartości z zakresu 60-255 (metoda `ThresholdBinaryInv` [16]). Metoda sprawdza czy wartość danego piksela mieści się w ustalonym przez nas zakresie –

jeżeli tak to jego wartość ustawiana jest na 0, w przeciwnym wypadku ustawiana jest wartość 255.

Przedział w zależności od zastosowanego urządzenia może się różnić, dlatego też użytkownik programu ma możliwość dostosowania wszystkich ustawień w panelu opcji. W moim przypadku zakres 60-255 został dostosowany metodą prób i błędów. Efektem naniesienia takiej maski na obraz będzie wyróżnienie obszaru źrenicy kolorem białym (255).



Rysunek 14. Obraz przed i po zastosowaniu metody `ThresholdBinaryInv`..

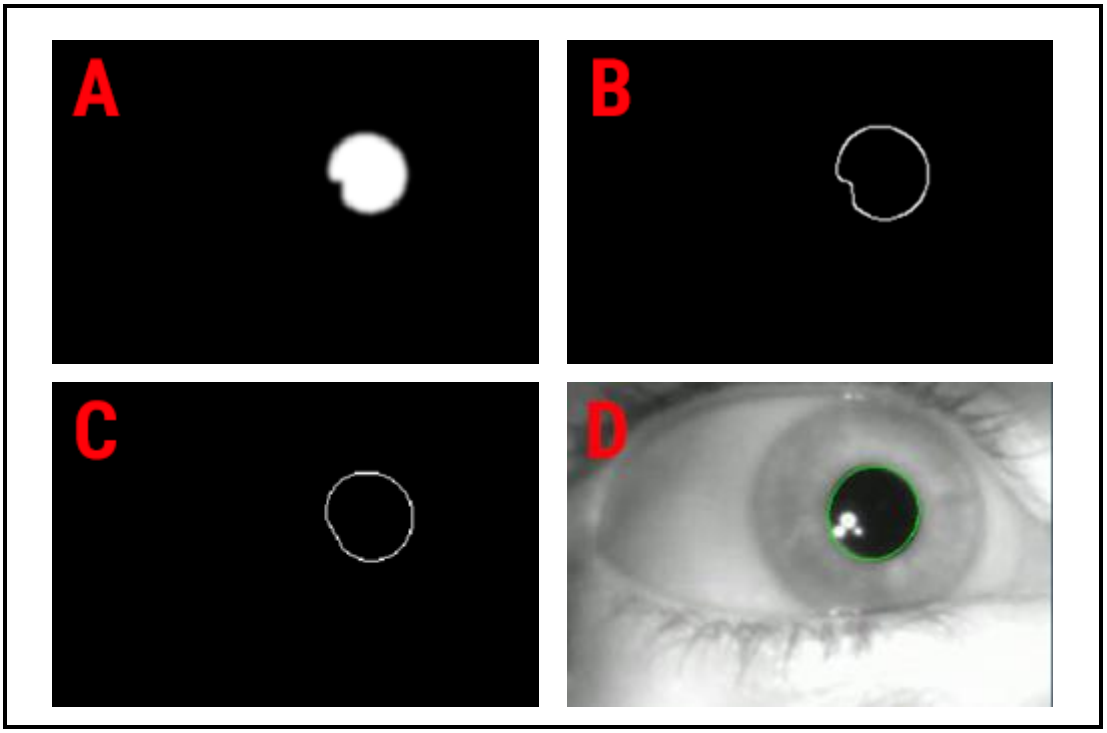
Jak widać na (rys. 15.), odcięcie nie wyeliminowało obszaru źrenicy rozjaśnionego przez diody podczerwieni, które utrudni wykrywanie okręgu źrenicy. Zrobią to dopiero dwie operacje morfologiczne: dylacja (metoda `Dilate`) i erozja (metoda `Erode`) (tab. 2.)

Ponieważ obraz oczu pochodzi z kamery zamocowanej na oprawce do okularów (eyetracker nagłowny), to pozycja oczu na obrazie zasadniczo się nie zmienia. Wobec tego nie jest konieczne obliczanie położenia źrenicy względem odbić Purkiniego diod podczerwonych (odbicia od rogówki oka), a wystarczy wykrywanie ich położenia względem krawędzi nagranego obrazu, o ile interesuje nas punkt spojrzenia względem układu odniesienia związanego z eyetrackerem (na wirtualnym ekranie przed badanym). To oczywiście jest przybliżenie, zakładające że oprawki okularów są całkowicie nieruchome względem głowy, nawet gdy badany nią porusza. Ponadto jeżeli interesuje nas

położenie spojrzenia na rzeczywistym monitorze komputera, należy albo unieruchomić głowę, albo mierzyć położenie i orientację głowy względem ekranu. Takie rozwiązanie wymagałoby zastosowania dodatkowych czujników, co zwiększyłoby całkowity koszt aparatury.

Na tak przygotowanym obrazie poszukujemy okręgów (metoda `HoughCircles` [17,18]). Po znalezieniu koła teoretycznie można zakończyć proces poszukiwania źrenicy. Warto jednak zauważyć, że podczas „skrajnych" spojrzeń w narożniki ekranu, wykryta źrenica przybiera kształt elipsy. Wówczas wyszukiwanie okręgów może wskazać niedokładny wynik.

Aby zminimalizować ten problem, najpierw szukamy krawędzi źrenicy (metoda `Canny` [20]) (rys. 16.B.), a następnie wyznaczamy jej otoczkę wypukłą (metoda `ConvexHull` [21]) (rys. 16.C.). Po wykonaniu powyższych czynności dopasujemy do niej elipsę (metoda `EllipseLeastSquareFitting`) (rys. 16.D).



Rysunek 15. A) wynik wielokrotnej transformacji morfologicznej, B) kontur źrenicy, C) uwypuklenie otrzymanego podzbioru przestrzeni liniowej, D) efekt dopasowania elipsy.

5. Kalibracja

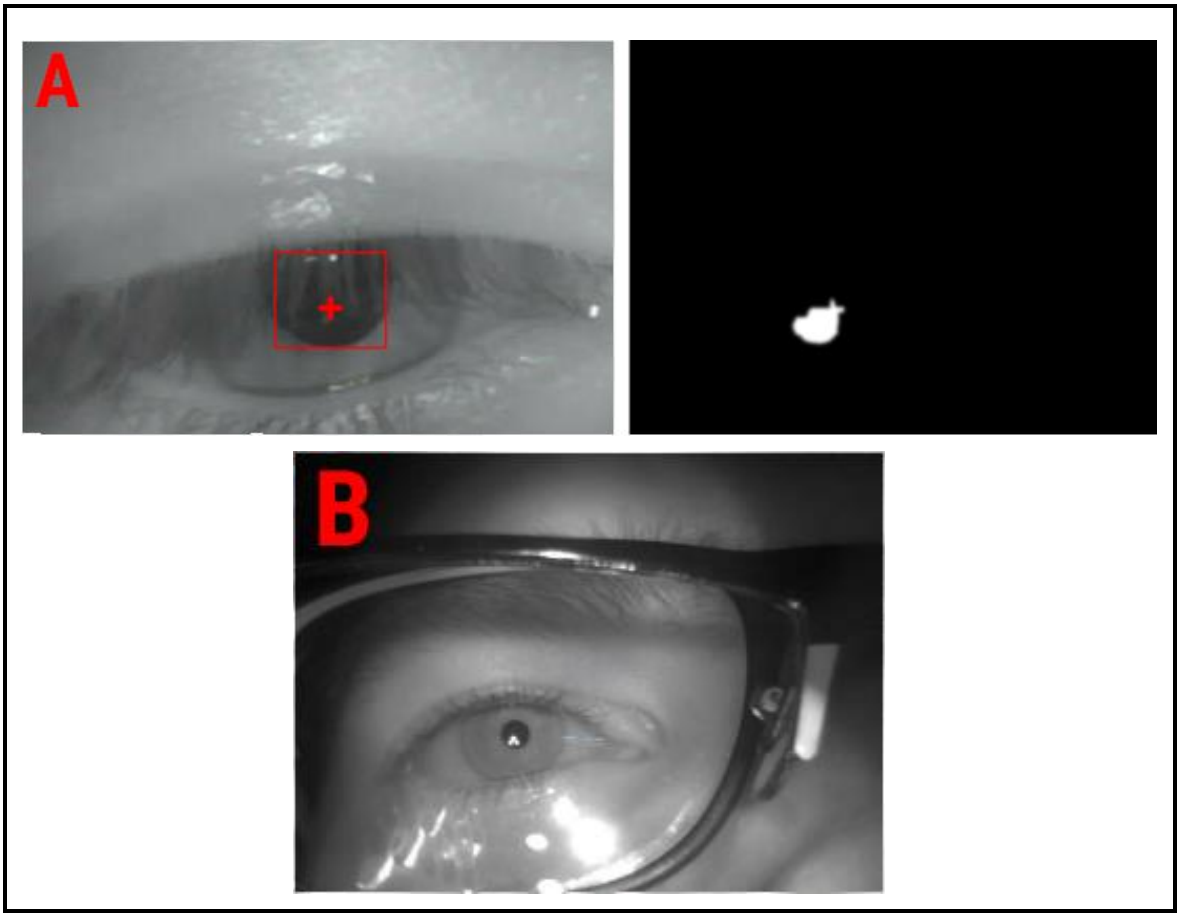
Aparatura badawcza wymaga wstępnej kalibracji dla każdego nowego użytkownika. Bez tego nie jest w stanie poprawnie wyznaczyć punkt spojrzenia na ekranie monitora. W przypadku mojego projektu dokonywana jest ona za pośrednictwem serii punktów kalibracyjnych, wyświetlanych w regularnych odstępach czasu. Ilość punktów jest ustalana w panelu konfiguracyjnym z predefiniowanej listy zawierającej dwie możliwości: 4 lub 9 punktów. Celem kalibracji jest zebranie informacji o skrajnych położeniach środka źrenicy podczas ruchu, które tworzą obszar, po jakim porusza się oko. Wyznaczony rejon może wówczas zostać odwzorowany na odpowiadający mu region ekranu monitora.

5.1. Wstępne przygotowania

Przed przystąpieniem do kalibracji warto sprawdzić jak algorytm wyszukiwania źrenicy radzi sobie z badaną osobą. Możemy natrafić na wiele różnorakich problemów. Do najczęstszych z nich należą [7]:

- Opadnięte powieki –mogą być spowodowany wiekiem lub chorobą. Opadnięte powieki są jednym z poważniejszych problemów w dziedzinie śledzenia ruchu oka, gdyż powodują zakrywanie obszaru źrenicy, deformując tym samym jej rejestrowany kształt (rys. 17.A.). Efekt ten jest dodatkowo powiększany przez rzęsy. Problem ten najłatwiej dostrzec, gdy osoba spogląda w dół. Rozwiązaniem tego przypadku jest poproszenie badanej osoby o rozluźnienie się, świadome szersze otwarcie oka, przestawienie kamery by rejestrowała pod większym kątem (od dołu) lub w ostateczności przytrzymanie ręką opadającej powieki.
- Okulary – mogą powodować wiele negatywnych efektów, wśród nich. np. tworzenie niepotrzebnego cienia, który niweluje kontrast pomiędzy źrenicą a resztą obserwowanego obszaru. Niektóre programy do śledzenia ruchu oka, oprócz pozycji środka źrenicy wykorzystują dodatkowo informację o punkcie, w którym następuje odbicie promieni światła z diod podczerwieni – wówczas odbicie światła od szkieł okularów może wprowadzać zakłócenie. Znaczenie może mieć również spadek jakości obrazu oka wynikający z porysowanych szkieł. Rozwiązaniem najkorzystniejszym jest zdjęcie okularów na czas badania.

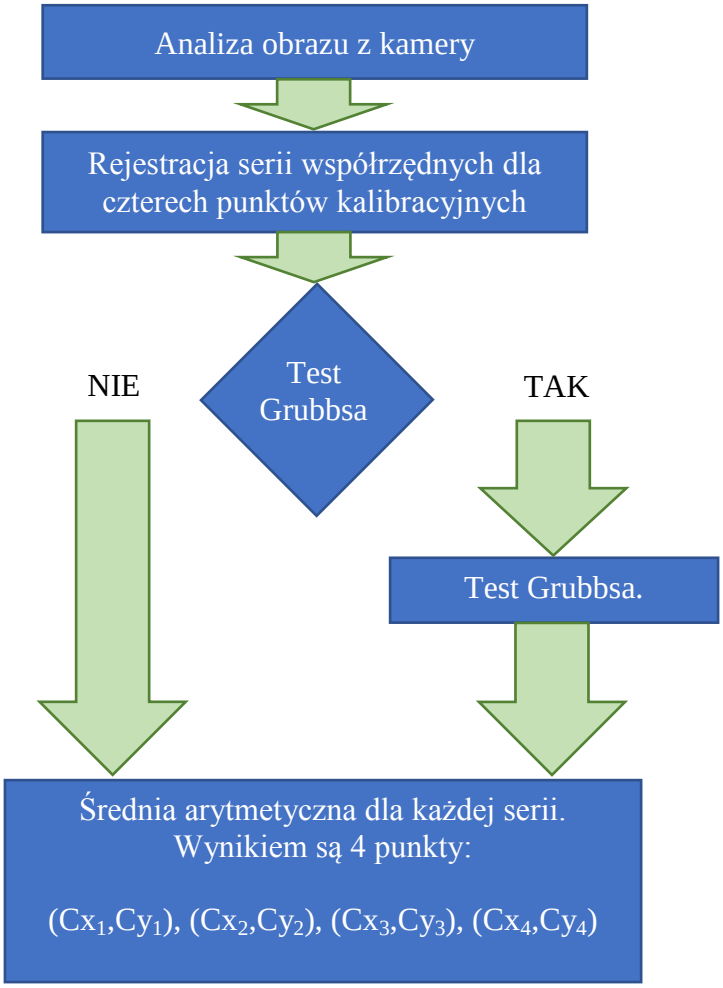
- Światło – Najlepszy efekt uzyskamy jeżeli jedynym źródłem światła będzie to generowane przez diody podczerwieni zamocowane na eyetrackerze. Wszystkie lampy, światło słoneczne mogą doprowadzić do nieprawidłowego wykrywania położenia źrenicy.



Rysunek 16. A) Nieprecyzyjne zlokalizowanie źrenicy wynikające z opadłem powieki, B) Refleksy promieni podczerwieni na szkłach okularów

- Miękkie Soczewki kontaktowe: mogą prowadzić do gromadzenia bąbelków powietrza, które powodują rozszczepienie promieni światła podczerwonego. Bąbelki podczas ruchu oka mogą się przemieszczać, więc usunięcie ich programowo może być trudne.
- Powszechne problemy takie jak, kurz na soczewce kamery, jej porysowanie lub brak ostrości.

5.2. Algorytm



5.3. Implementacja

Użytkownik podczas procesu kalibracji proszony jest o patrzenie na kolejno wyświetlane na ekranie punkty kalibracyjne (rys. 18.A-D.). Dla każdego punktu zapisywana jest lista współrzędnych środka źrenicy; trzy sekundowy czas nagrywania pozwala otrzymać ok. 25 współrzędnych w tablicy. Zwiększenie czasu wyświetlania jednego punktu kalibracji może pozwolić uzyskać więcej danych, a zatem dokładniejsze przybliżenie. Należy jednak pamiętać żeby nie ustawić zbyt długiego interwału, ponieważ możemy uzyskać efekt przeciwny do zamierzonego – zbyt długa fiksacja może zmęczyć badanego, w efekcie czego nastąpi mimowolna zmiana punktu spojrzenia lub mrugnięcie.

Dyskomfort związany z kalibracją można redukować implementując dodatkowe mechanizmy. Ji Woo Lee, Hwan Heo oraz Kang Ryoung Park [28] przedstawiają metodę polegającą na ograniczeniu procesu kalibracji do czterech fizycznych punktów uzupełnionych pięcioma punktami wirtualnymi generowanymi przez algorytm MLP (ang. *multilayer perceptron*).

Po zebraniu danych dla danego punktu kalibracji, możemy obliczyć odpowiadającą mu średnią wartość punktu spojrzenia (C_x,C_y). Może się zdarzyć że na skutek nieprzewidzianego błędu program zapisze wartości, które odstają od pozostałych. W celu ich wyeliminowania odrzucane są dane, których współrzędne *x* lub *y* (osobno) są oddalone od wartości średnich o więcej niż odchylenie standardowe skorygowane poprawką t-Studenta (test Grubbsa) [29]:

- Wyznaczamy średnią arytmetyczną danych w zbiorze

$$\overline{C_{x1}} = \frac{C_{x1_1} + C_{x1_2} + \cdots C_{x1_n}}{n} \qquad \overline{C_{y1}} = \frac{C_{y1_1} + C_{y1_2} + \cdots C_{y1_n}}{n} \tag{4}$$

- Obliczamy odchylenie standardowe

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(C_{x1_i} - \overline{C_{x1}}\right)^2} \qquad s_y = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(C_{y1_i} - \overline{C_{y1}}\right)^2} \tag{5}$$

- Wartość τ

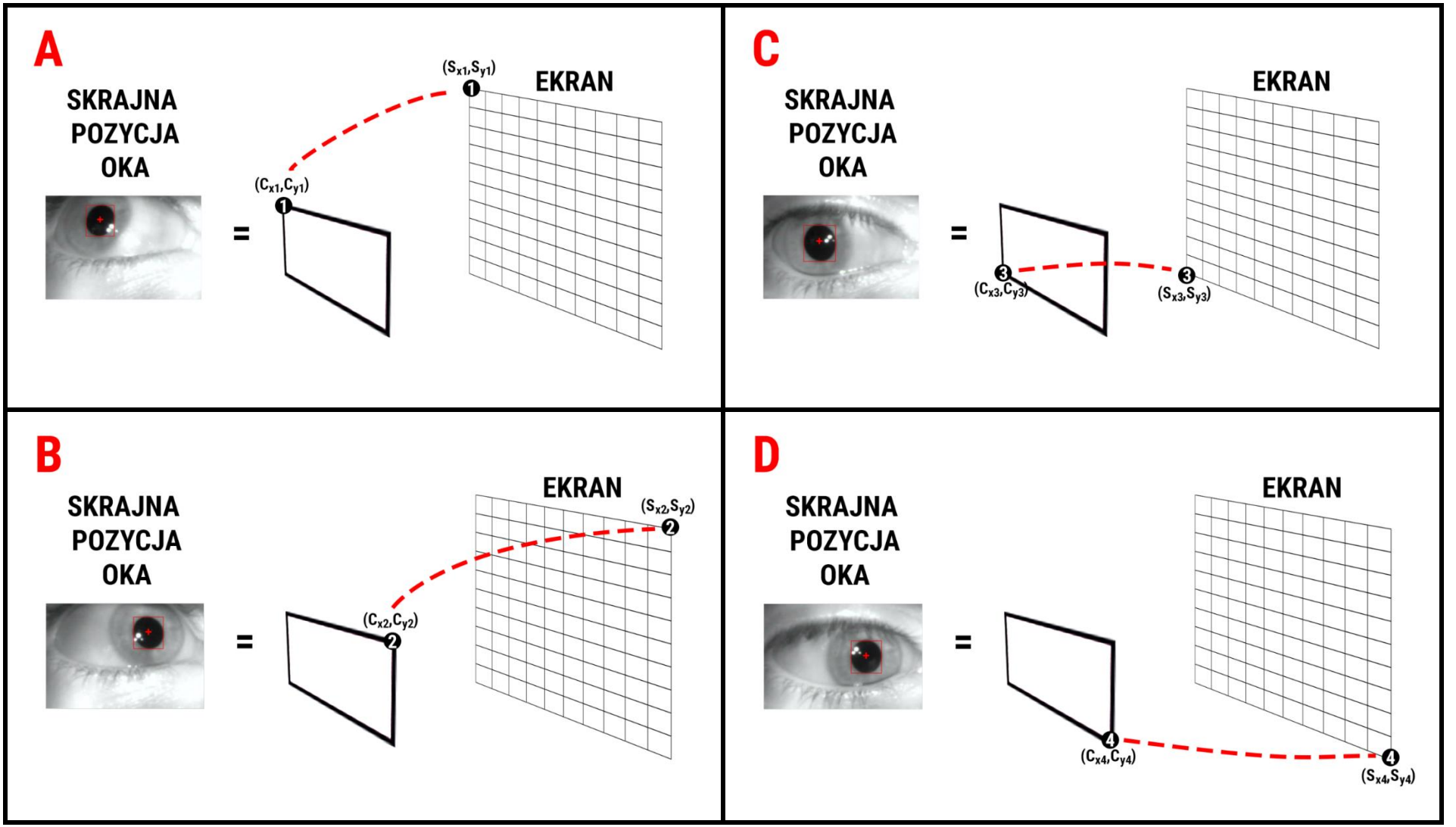
$$\tau = \frac{t_{\alpha/2} \cdot (n-1)}{\sqrt{n}\sqrt{n-2+t_{\alpha/2}^2}} \tag{6}$$

  Gdzie $t_{\alpha/2}$ jest wartością pobraną z tablicy rozkładu t-Studenta dla następujących argumentów $\alpha = 0.05$ oraz stopni swobody (df) $= n - 2$.
  Liczba stopni swobody *df* (ang. *degrees of freedom*) – liczba niezależnych wyników obserwacji pomniejszona o liczbę związków, które łączą te wyniki ze sobą.

- Jeżeli dla któregokolwiek C_{x1_i} lub C_{y1_i} prawdą jest że:

$$\left|C_{x1_i} - \overline{C_{x1}}\right| > \tau s_x \qquad \left|C_{y1_i} - \overline{C_{y1}}\right| > \tau s_y, \tag{7}$$

  to taka wartość taka jest wartością odstającą i należy ją pominąć.



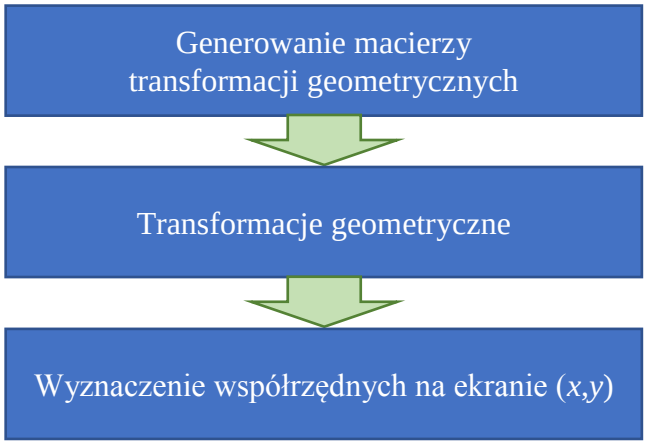
Rysunek 17. A) Lewy górny skrajny punkt kalibracji, B) Prawy górny skrajny punkt kalibracji C) Lewy dolny skrajny punkt kalibracji, D) Prawy dolny skrajny punkt kalibracji

W zebranych przeze mnie danych nie pojawiały się wartości odstające, gdyż podczas testów osoba badana była proszona o nie ruszanie głową, więc postanowiłem zrezygnować z implementacji tego filtra.

Po przeprowadzeniu kalibracji otrzymujemy cztery położenia źrenicy na obrazie z kamery eyetrackera [(Cx_1,Cy_1), (Cx_2,Cy_2), (Cx_3,Cy_3), (Cx_4,Cy_4)], odpowiadające czterem punktom kalibracji znajdującym się w rogach ekranu [(Sx_1,Sy_1), (Sx_2,Sy_2), (Sx_3,Sy_3), (Sx_4,Sy_4)]. Odwzorowanie to ilustrują rysunki 18.A-D.

6. Mapowanie współrzędnych oka na płaszczyznę ekranu

6.1. Algorytm



6.2. Implementacja

Jeżeli urządzenie zostało poprawnie skalibrowane, jesteśmy w stanie przejść do wyznaczania współrzędnych ekranu, na które użytkownik w danym momencie patrzy. W tym celu wykorzystujemy funkcję mapującą. Może to być wielomian pierwszego bądź drugiego stopnia.

W przypadku wielomianu pierwszego stopnia relacja pomiędzy środkiem źrenicy (C_x,C_y), a współrzędną ekranu (S_x,S_y) opisana jest w następujący sposób:

$$\begin{aligned} S_x &= a \cdot C_x + b \cdot C_y + c \cdot C_x \cdot C_y + d \\ S_y &= e \cdot C_x + f \cdot C_y + g \cdot C_x \cdot C_y + h \end{aligned} \quad \textbf{(8)}$$

Wielomian pierwszego stopnia (8) zawiera osiem parametrów określających transformację geometryczną. Z kolei wielomian drugiego stopnia można zapisać jako:

$$\begin{aligned} S_x &= a \cdot C_x^2 + b \cdot C_y^2 + c \cdot C_x + d \cdot C_y + e \cdot C_x \cdot C_y + f \\ S_x &= g \cdot C_x^2 + h \cdot C_y^2 + i \cdot C_x + j \cdot C_y + k \cdot C_x \cdot C_y + l \end{aligned} \quad \textbf{(9)}$$

Równania (8) oraz (9) mogą zostać przedstawione odpowiednio jako:

$$\begin{bmatrix} S_x \\ S_y \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_x \\ C_y \\ C_x C_y \\ 1 \end{bmatrix} \quad \textbf{(10)}$$

$$\begin{bmatrix} S_x \\ S_y \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a & b & c & d & e & f \\ g & h & i & j & k & l \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_x^2 \\ C_y^2 \\ C_x \\ C_y \\ C_x C_y \\ 1 \end{bmatrix} \quad \textbf{(11)}$$

W moim programie wykorzystuję wielomian pierwszego stopnia. Autorzy Ji Woo Lee, Hwan Heo oraz Kang Ryoung Park [28] sugerują w przypadku większej liczby punktów kalibracyjnych, na przykład dziewięciu, które dzielą ekran na cztery regiony, skorzystać z wielomianu pierwszego stopnia dla każdego z rejonu osobno w celu uzyskania lepszej dokładności.

Przekształcając równanie (10) jesteśmy w stanie wyznaczyć osiem niewiadomych argumentów wykorzystując w tym celu dane zapisane podczas procesu kalibracji

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} S_{x1} & S_{x2} & S_{x3} & S_{x4} \\ S_{y1} & S_{y2} & S_{y3} & S_{y4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} C_{x1} & C_{x2} & C_{x3} & C_{x4} \\ C_{y1} & C_{y2} & C_{y3} & C_{y4} \\ C_{x1}C_{y1} & C_{x2}C_{y2} & C_{x3}C_{y3} & C_{x4}C_{y4} \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \quad \textbf{(12)}$$

Po wyznaczeniu niewiadomych jesteśmy w stanie na bieżąco przeliczać współrzędne oka na odpowiadające współrzędne na ekranie monitora, wykorzystując w tym celu równanie (10).

Obliczenia na macierzach dokonuję z wykorzystaniem dodatkowej biblioteki *MathNet.Numerics,* która pozwala w prosty sposób:

- tworzyć macierze,
- mnożyć macierze,
- wyznaczać wyznacznik macierzy,
- odwracać macierz.

Oczywiście zamiast wykorzystywać gotową bibliotekę możemy rozwiązać równanie (10) analitycznie i zaimplementować gotowy wzór. Głównym problemem jest znalezienie macierzy odwrotnej do iloczynu macierzy z prawej strony wyrażenia (12). Można ją znaleźć analitycznie, lecz jej jawna postać jest zbyt rozbudowana, żeby ją tu przytaczać. Co więcej ze względu na mały rozmiar macierzy, korzystanie z jawnego wzoru nie jest wcale znacznie bardziej korzystne ze względu na koszt obliczeń.

7. Testy

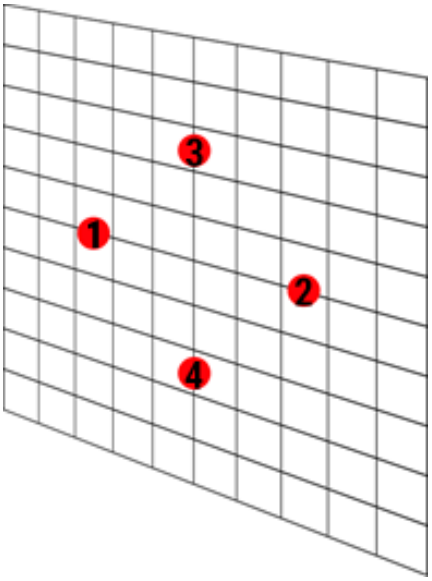
Celem tego rozdziału jest sprawdzenie dokładności stworzonej przeze mnie aparatury. Procedura przeprowadzonych przeze mnie testy składała się z:

- optymalizacji parametrów udostępnionych w zestawie opcji,
- kalibracji w oparciu o skrajne cztery punkty na ekranie monitora laptopa,
- obserwacji czterech punktów testowych (rys. 19.) – dla każdej kropki przez dwie sekundy zapisujemy wszystkie zarejestrowane współrzędne,
- sporządzenie wykresu przedstawiającego wszystkie zarejestrowane współrzędne względem punktów testowych,
- wyznaczeniu współczynnika dokładności eyetrackera.

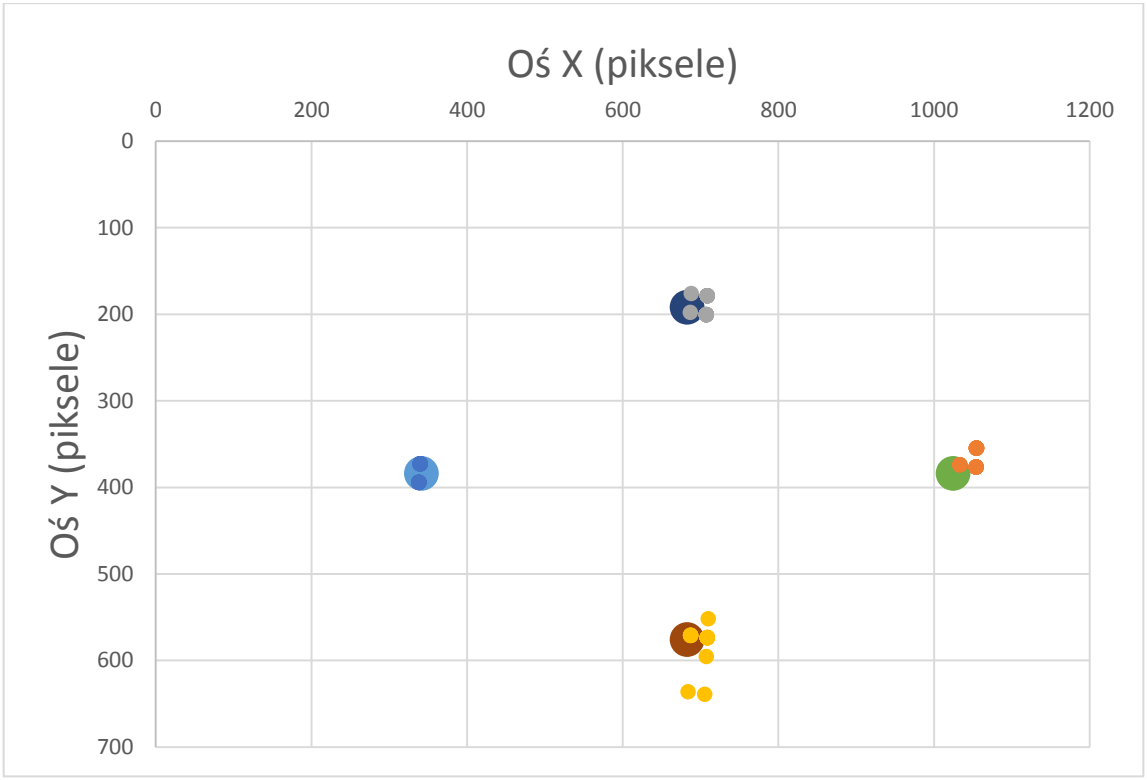
Twarz osoby badanej jest oddalona od ekranu monitora o ok. 80 cm. Badany nie nosi okularów oraz nie jest w żaden sposób usztywniony. Został poinstruowany, że nadmierny ruch może powodować zmniejszenie dokładność urządzenia.

Cały test został powtórzony pięciokrotnie na jednym badanym. Pomiędzy cyklami testu badany odpoczywał przez pięć minut, co miało zagwarantować wypoczęte oka w trakcie kalibracji.

W badaniu użyte zostały cztery punkty o współrzędnych (341,5; 384), (1024,5; 384), (683;192) i (683; 576) (oznaczone na rys. 19 numerami od 1 do 4), na ekranie o rozdzielczości 1366 x 768.

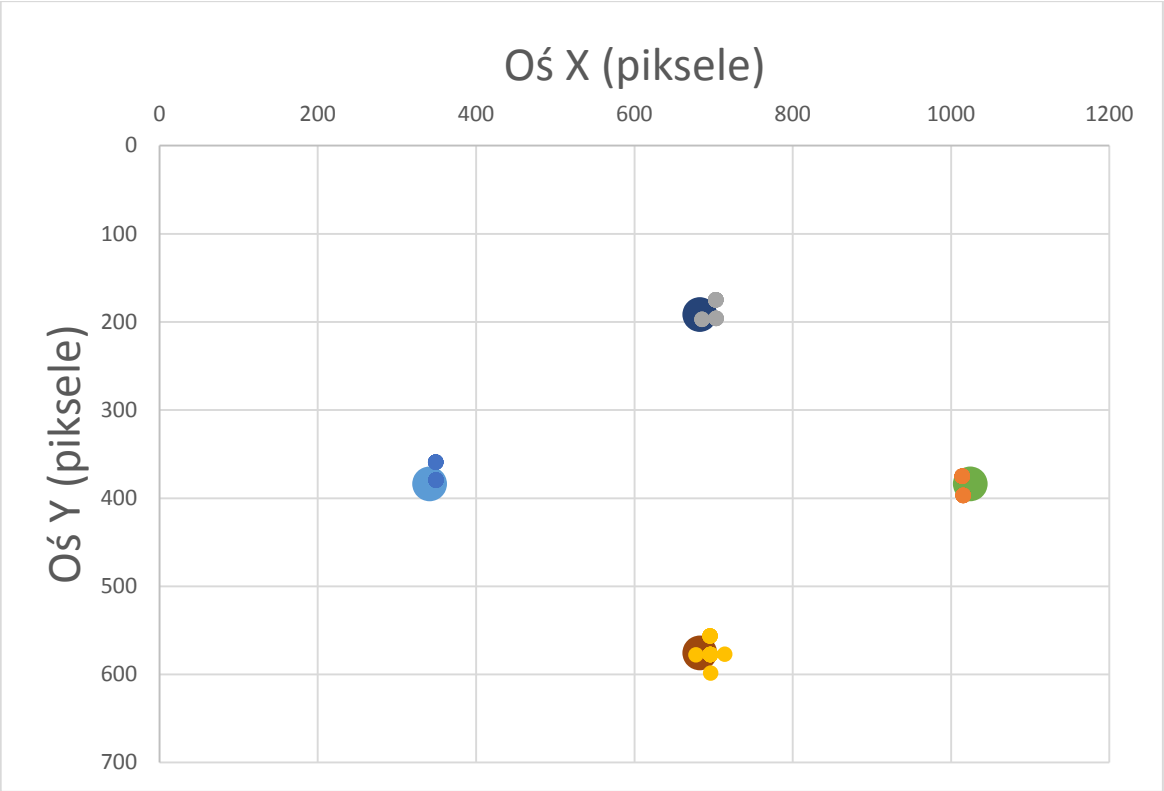


Rysunek 18. Ilustracja czterech punktów testowych na ekranie.



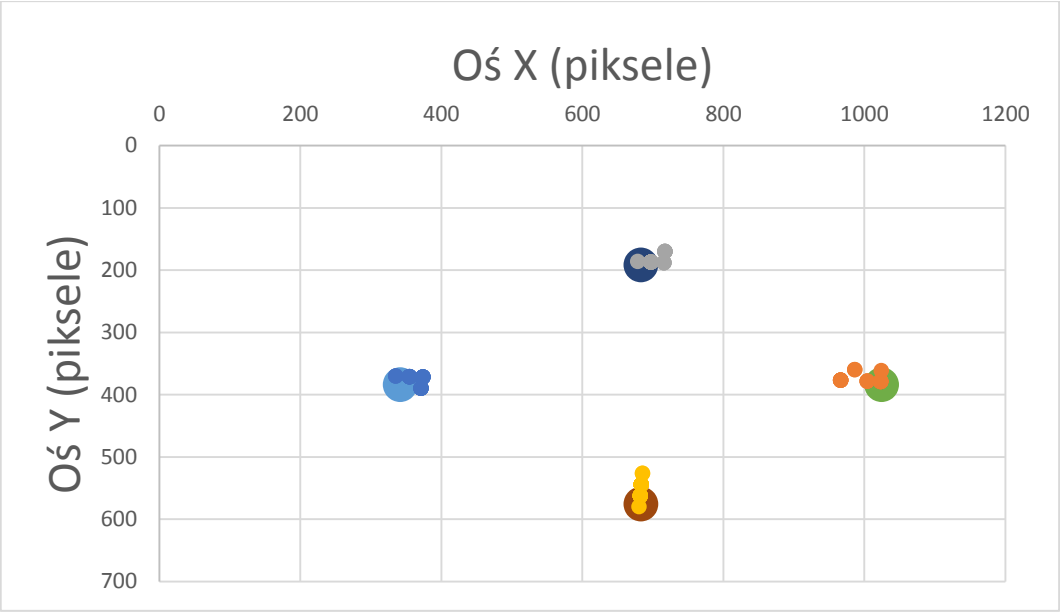
Rysunek 19. Test 1.

Rysunek 20. Test 2.



Rysunek 21. Test 3.

Rysunek 22. Test 4.



Rysunek 23. Test 5.

Tabela 3 przedstawia różnicę pomiędzy średnią arytmetyczną zarejestrowanych współrzędnych (dla każdego punktu) a położeniem punktu testowego. Wyniki podane są w pikselach.

	Punkt 1 (*x*, *y*)	**Punkt 2 (*x*, *y*)**	**Punkt 3 (*x*, *y*)**	**Punkt 4 (*x*, *y*)**
Test 1	2,60; 0,33	26,70; 17;41	23,14; 1,41	19,03; 3,54
Test 2	6,87; 25,80	16,06; 12,83	16,45; 20,38	3,82; 21,68
Test 3	7,95; 12,43	9,47; 0,16	17,57; 8,34	13,14; 2,49
Test 4	4,72; 11,14	27,50; 24,12	14,06; 28,13	1,87; 10,32
Test 5	21,13; 8,04	24,81; 8,49	21,05; 8,98	0,07; 24,29
Średnia	8,65; 11,54	20,90; 12,62	18,45; 13,42	7,58; 12,46

Tabela 3. Podsumowanie wyników wszystkich pięciu cykli badawczych.

Średnie odchylenie od prawidłowej wartości na osi X wynosi 13,9 pikseli ; Y wynosi 12,5 pikseli. Z wyników wynika, że błąd wzdłuż osi Y jest niezależny od punktu i wynosi około 11-14 pikseli. Znacznie bardziej zmienia się błąd na osi X. Dla punktów 1 i 4 jest on mniejszy od 10 pikseli, podczas, gdy dla punktów 2 i 3 rośnie do około 20. Może to wynikać np. z ruchu głowy podczas badań (naturalna chęć skręcenia głowy przy patrzeniu na skrajnie położone punkty) albo po prostu ze zbyt małej liczby powtórzeń testu.

8. Podsumowanie

Celem projektu było stworzenie taniego eyetrackera, który może być łatwo zbudowany samodzielnie, a który będzie w stanie stosunkowo wiarygodnie wyznaczać położenie punktu spojrzenia na podstawie śledzonego ruchu gałki ocznej. Wielokrotnie przeze mnie podkreślanym kryterium była cena: zależało mi, aby koszt ukończenia projektu był jak najniższy. To zostało osiągnięte. Szacuję, że całkowity koszt projektu nie przekracza 30 zł.

Materialnym efektem mojej pracy są dwa prototypy eyetrackerów. Przygotowałem także oprogramowanie analizujące obraz z kamery, wyznaczające położenie punktu spojrzenia, a tym samym demonstrujące funkcjonalność prototypów. Program w obecnej postaci nie obsługuje wszystkich możliwych przypadków. Jeżeli osoba badana nosi okulary, to aby korzystać z urządzenia zmuszona jest je zdjąć. Znajduję dwa powody takiej sytuacji:

- mój eyetracker został wykonany z użyciem oprawki od okularów, a ciężko jest nosić dwie pary okularów w tym samym czasie. Problem ten chciałbym w przyszłości rozwiązać stosując aparaturę podwieszaną pod monitorem lub montując kamerę do specjalnego kasku.
- szkła okularów odbijają zbyt wiele refleksów promieni światła podczerwonego (refleksy widoczne na powierzchni szkieł). Problem ten jest dodatkowo potęgowany przez nadmiernie prześwietlony obraz. Refleksy bardzo utrudniają poprawne śledzenie ruchu gałki ocznej. Jeden z dostępnych w sieci artykułów [30] pokazuje, jak w sposób programowy można wyeliminować ten mankament, wykorzystując różny czas naświetlania obrazu.

Należy jednak zaznaczyć, że problemy z badaniem osób noszących okulary pojawia się także w przypadku profesjonalnych eyetrackerów, nawet tych z najwyższej półki (np. SMI Red za 250 tys. zł).

Sprzęt w obecnej postaci prezentuje się raczej surowo (rys. 10.). Szczególnie razi kątownik użyty do zamocowania kamery do oprawki okularów. Jest to jednak spowodowane przez narzucone na projekt kryterium ceny. Uważam, że aparatura powinna zostać wzbogacona o wydrukowany model 3D obudowy. Można również ulepszyć projekt przenosząc cały układ scalony kamery na bok oprawki pozostawiając na „wysięgniku”

jedynie sam sensor z diodami. Taki manewr wymaga oczywiście większych umiejętności technicznych, lecz zmniejszyłby ciężar wysuniętej części okulografu, co powinno ustabilizować obraz przy poruszaniu głową oraz pozwoliłby zwiększyć pole widzenia pacjenta..

W trakcie prac nad prototypami pojawił się pomysł, aby skonstruować również jedno urządzenie wyposażone w kamerę o wyższej rozdzielczości w celu porównania wyników. Niestety zastosowana kamera nie przetrwała procesu demontażu – jej matryca uległa uszkodzeniu.

Przygotowane przeze mnie oprogramowanie również podatne jest na usprawnienia. Odwzorowanie położenia źrenicy na punkt spojrzenia opisane jest obecnie wielomianem pierwszego stopnia. Metodę można rozbudować o wielomian drugiego stopnia albo o wspomnianą w rozdziale 6. metodę wieloobszarowego wielomianu pierwszego stopnia.

Istnieje również możliwość dodania funkcjonalności śledzenia odbić promieni podczerwieni na powierzchni oka – przy dodatkowej implementacji podczas kalibracji można uzyskać lepszą dokładność i uniezależnić się od ruchów eyetrackera względem głowy.

W trakcie testów warto byłoby również użyć uchwytu stabilizującego głowę. Stworzona przeze mnie podpórka nie w pełni spełniła swoje zadanie. Nie posiada regulowanej wysokości, więc nie jest uniwersalna oraz pozwala na nadmierny ruch głowy w jednej osi. Dlatego podczas przeprowadzonych testów nie została nawet wykorzystana.

Uzyskane wyniki badań potwierdzają, że stworzony przeze mnie eyetracker jest w stanie wiarygodnie wskazywać położenie punktu spojrzenia. Przy tym stosunek jakości do ceny jest bardzo korzystny. Wydaje mi się, że zastosowanie opisanych powyżej modyfikacji, pozwoliłoby dodatkowo poprawić dokładności pomiaru nie zwiększając znacznie ceny urządzenia.

9. Literatura

[1] Jason S.Babcock and Jeff B. Pelz. Building a lightweight eyetracking headgear. Eye Tracking Research & Application, Texas, 2004.

[2] J. Babcock, J. Pelz and J. Peak. The Wearable Eyetracker: A Tool for the Study of Highlevel Visual Tasks. February 2003

[3] Li, D., Babcock, J., Parkhurst, D. J. openEyes: A low-cost head-mounted eye-tracking solution. Proceedings of the ACM Eye Tracking Research and Applications Symposium 2006. 04.

[4] Javier San Agustin, Henrik Skovsgaard, John Paulin Hansen, Dan Witzner Hansen. LowCost Gaze Interaction: Ready to Deliver the Promises. CHI 2009, Boston, Massachusetts, USA.

[5] *http://rmantiuk.strony.wi.ps.pl/projects/diy/data/How_to_build_DIY.pdf* -dokument PDF. Michał Kowalik, How to build low cost eye tracking glasses for head mounted system. (Link z dnia 28.01.2017r)

[6] Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies, 2012. ISBN 978-1-61350-098-9).

[7] Eye Tracking A Comprehensive Guide to Methods and Measure, 2011. ISBN 978-0-19969708-3

[8] Eye Tracking Methodology Theory and Practice Second Edition, 2007. ISBN 978-1-84628-608-7

[9] *https://www.microsoft.com/accessories/pl-pl/d/lifecam-vx-3000* - strona producenta zawierająca specyfikację techniczną wykorzystanej kamery internetowej. (Link z dnia 28.01.2017r)

[10] *https://www.microsoft.com/accessories/pl-pl/d/lifecam-vx-1000* - strona producenta zawierająca specyfikację techniczną bliźniaczej kamery internetowej. (Link z dnia 28.01.2017r)

[11] *http://www.emgu.com/wiki/index.php/Main_Page* - oficjalna strona internetowa, biblioteki EmguCV (Link z dnia 28.01.2017r)

[12] *https://msdn.microsoft.com/en-us/library/dd390351(v=vs.85).aspx* - oficjalna strona internetowa, zawierająca wprowadzenie do wykorzystanej biblioteki DirectShowLib (Link z dnia 28.01.2017r)

[13] *https://numerics.mathdotnet.com* - oficjalna strona internetowa, biblioteki MathNet.Numerics (Link z dnia 28.01.2017r)

[14] *https://www.nuget.org* - oficjalna strona internetowa menadżera dodatków, zaimplementowanego w środowisku programistycznym Visual Studio 2015r. (Link z dnia 28.01.2017r)

[15] *http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html* - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca metodę rozciągania histogramu (Link z dnia 28.01.2017r) –

[16] *http://www.docs.opencv.org/2.4.10/doc/tutorials/imgproc/threshold/threshold.html* - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca metodę binarnego odcięcia wartości (Link z dnia 28.01.2017r)

[17] *http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles#houghcircles* - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca dokumentację metody detekcja okręgów -Transformata Hough'a (Link z dnia 28.01.2017r)

[18] *http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html* - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca wykorzystanie metody detekcja okręgów -Transformata Hough'a (Link z dnia 28.01.2017r)

[19] *http://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html* - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca wykorzystanie operacji morfologicznych (Link z dnia 28.01.2017r)

[20] *http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=canny* - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca wykorzystanie operacji detekcji krawędzi - Canny (Link z dnia 28.01.2017r)

[21] *http://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/hull/hull.html* - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca w jaki sposób uwypuklić podzbiór przestrzeni liniowej (Link z dnia 28.01.2017r)

[22] *http://home.agh.edu.pl/~awrobel/resources/Zarys%20fotogrametrii.pdf* - Andrzej wróbel Fotogrametria, Histogram obrazu cyfrowego strona 5

[23] http://pforczmanski.zut.edu.pl/homepage/wp-content/uploads/w07-cechy_ksztaltu.pdf - dr inż. Paweł Forczmański, Prezentacja Detekcja kształtów i wybrane cechy obrazów konturowych. Zachodniopomorski Uniwersytet Technologiczny w Szczecinie. (Link z dnia 11.02.2017r)

[24] http://www.fizyka.umk.pl/~gniewko/python/OpenCV_przetwarzanie/#transformacje_morfologiczne - Strona internetowa, dr hab. inż. Gniewomira Sarbickigo, wykładowcy na uniwersytecie Mikołaja Kopernika. Dokumentacja opisująca bibliotekę OpenCV. (Link z dnia 28.01.2017r)

[25] http://docs.opencv.org/trunk/d4/d86/group__imgproc__filter.html#ga4ff0f3318642c4f469d0e11f242f3b6c - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca dokumentację metody morfologicznej Dylacji (Link z dnia 28.01.2017r)

[26] http://docs.opencv.org/trunk/d4/d86/group__imgproc__filter.html#gaeb1e0c1033e3f6b891a25d0511362aeb - Oficjalna strona internetowa biblioteki OpenCV, przedstawiająca dokumentację metody morfologicznej Erozji (Link z dnia 28.01.2017r)

[27] http://edu.pjwstk.edu.pl/wyklady/asd/scb/asd13/main13_p3.html - Strona internetowa Polsko-Japońskiej akademii technik komputerowych, Wykład nr 13.3 Problem otoczki wypukłej. (Link z dnia 28.01.2017r)

[28] http://www.mdpi.com/1424-8220/13/8/10802 - Ji Woo Lee, Hwan Heo and Kang Ryoung Park, Artykuł „ A Novel Gaze Tracking Method Based on the Generation of Virtual Calibration Points” Sensors 2013, 13, 10802-10822; doi:10.3390/s130810802. ISSN 1424-8220

[29] http://mne.psu.edu/cimbala/me345/Lectures/Outliers.pdf - John M. Cimbala, Penn State University Latest revision: 12 September 2011 „ Outliers” (Link z dnia 28.01.2017r) –

[30] http://www.mdpi.com/1424-8220/14/2/2110 - Su Yeong Gwon, Chul Woo Cho, Hyeon Chang Lee, Won Oh Lee and Kang Ryoung Park, Artykuł „ Gaze Tracking System for User Wearing Glasses” Sensors 2014, 14, 2110-2134; doi:10.3390/s14020211. ISSN 1424-8220

[31] http://docplayer.pl/2524635-Operacje-morfologiczne-w-przetwarzaniu-obrazu.html - Strona internetowa, z prezentacją Pani Izabeli Karczewskiej. Operacje morfologiczne w przetwarzaniu obrazu (Link z dnia 28.01.2017r)