

**Uniwersytet Warszawski**  
Wydział Fizyki

**JERZY GRYNCZEWSKI**  
Nr albumu: 307401

# GAZESWITCH

Projekt zaliczeniowy  
z przedmiotu  
Programowanie zaawansowane FM i NI



# Program GazeSwitch

Praca stanowi projekt zaliczeniowy z przedmiotu Programowanie Zaawansowane FM i NI na Wydziale Fizyki Uniwersytetu Warszawskiego. W ramach projektu napisano program **GazeSwitch**. Jest to program wykorzystujący sygnał okulograficzny do detekcji spojrzenia w wybranym kierunku. Program wykorzystuje narzędzia i metody cyfrowego przetwarzania obrazu. Sygnał, czyli pojedyńcze klatki strumienia filmowego przesyłanego przez okulograf jest analizowany w programie GazeSwitch. Podstawowym elementem programu jest **algorytm detekcji spojrzenia**. Efektem działania algorytmu jest klasyfikacja sygnału do jednego z dwóch zbiorów – detekcja spojrzenie w kierunku jednej z diód lub brak spojrzenia w tym kierunku. Klasyfikacja sygnału do zbioru odpowiadającego spojrzeniu na jedną z diód wzywała odpowiedź przesyłaną do graficznego interfejsu użytkownika. Efekt jaki wywołuje przesłana odpowiedź zależy od miejsca wykonywania w jakim znajduje się graficzny interfejs użytkownika

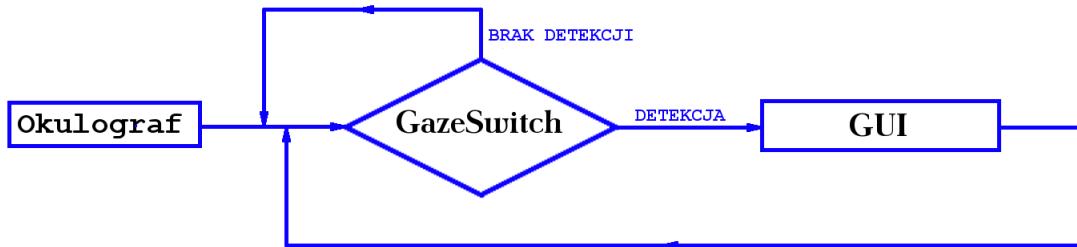
## 0.1 Ogólna struktura programu GazeSwitch

Do akwizycji sygnału okulograficznego wykorzystano urządzenie przedstawione na ilustracji 1



Rysunek 1: Okulograf wykorzystywany przy pisaniu programu.

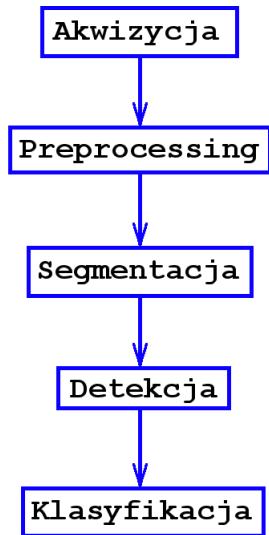
Sygnal zbierany przez prosty okulograf i analizowany przez program GazeSwitch jest przekazywany do graficznego interfejsu użytkownika (GUI). Szkic działania prezentuje ilustracja 2.



Rysunek 2: Schemat blokowy działania systemu.

## 0.2 Ogólna struktura algorytmu detekcji spojrzenia

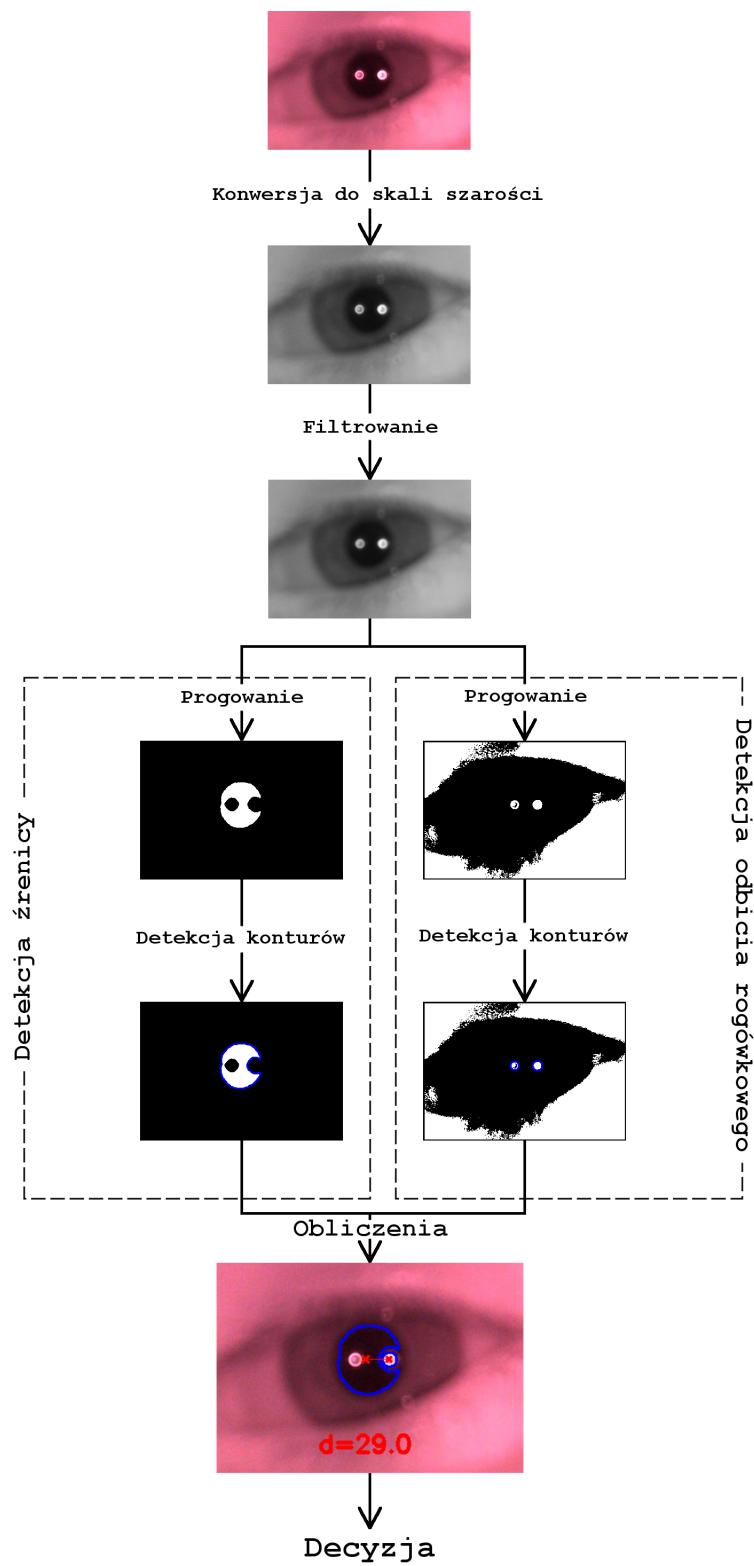
Algorytm detekcji spojrzenia jest oparty na podstawowych metodach cyfrowego przetwarzania obrazu (DIP; ang. *digital image processing*).



Rysunek 3: Ogólna struktura algorytmu detekcji spojrzenia w programie GazeSwitch.

W algorytmie detekcji spojrzenia, wykorzystywanym przez program GazeSwitch można wyróżnić pięć podstawowych etapów przetwarzania obrazu oka. Schematycznie algorytm przedstawia ilustracja 3. Pierwszy etap stanowi akwizycja obrazu, czyli jego przechwytywanie. Drugi etap to przetwarzanie wstępne przygotowujące obraz do obróbki właściwej. W trzecim etapie z obrazu wyodrębnione zostają określone obszary. Czwarty etap stanowi obliczenia wykonywane na wyodrębnionych obszarach, a piąty to przeprowadzona na podstawie obliczeń klasifikacja. Algorytm wykonuje powyższe kroki na każdej klatce strumienia wideo.

Schemat działania algorytmu na konkretnej klatce strumienia przedstawia ilustracja 4.



Rysunek 4: Schemat działania algorytmu detekcji źrenicy.

### 0.3 Szczegóły implementacji

Działanie poszczególnych etapów algorytmu przedstawiane jest na reprezentatywnym obrazie 5.



Rysunek 5: Reprezentatywny obraz oka.

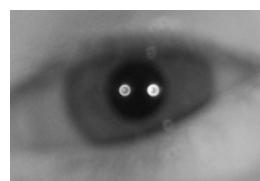
#### 0.3.1 Preprocessing

Preprocessing inaczej nazywany przetwarzaniem wstępny obejmuje nazwą wszystkie operacje wykonywane na obrazie w celu przygotowania go do obróbki właściwej. W algorytmie detekcji spojrzenia do tego etapu można zaliczyć konwersje obrazu do skali szarości oraz filtrowanie.

##### Konwersja obrazu do skali szarości

Konwersje do obrazu monochromatycznego wykonano zgodnie ze wzorem 1. Efekt przedstawia ilustracja 6

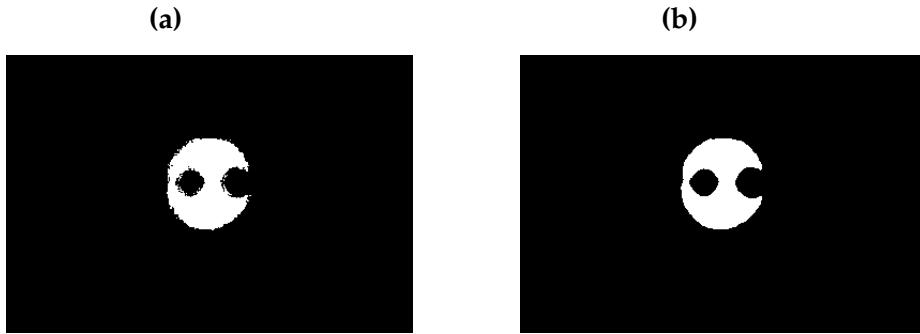
$$GY = 0.299 \cdot R + 0.587 \cdot G + 0.144 \cdot B \quad (1)$$



Rysunek 6: Obraz w skali szarości.

## Filtrowanie

Filtrowanie jest operacją mającą na celu pozbycie się szumu z obrazu. W szczególności filtry wygładzające są stosowane w celu zmniejszenia wariancji wartości pikseli. W algorytmie został zastosowany filtr gaussowski z jądrem  $7 \times 7$ . Efekt działania filtra wyraźny staje się dopiero po przeprowadzeniu segmentacji będącej następnym omawianym w pracy etapem przetwarzania. Rysunek 7 efekt ten ilustruje.



Rysunek 7: a) Obraz poddany segmentacji bez poprzedzającej go filtracji, b) Obraz poddany segmentacji po uprzednim przefiltrowaniu.

### 0.3.2 Segmentacja

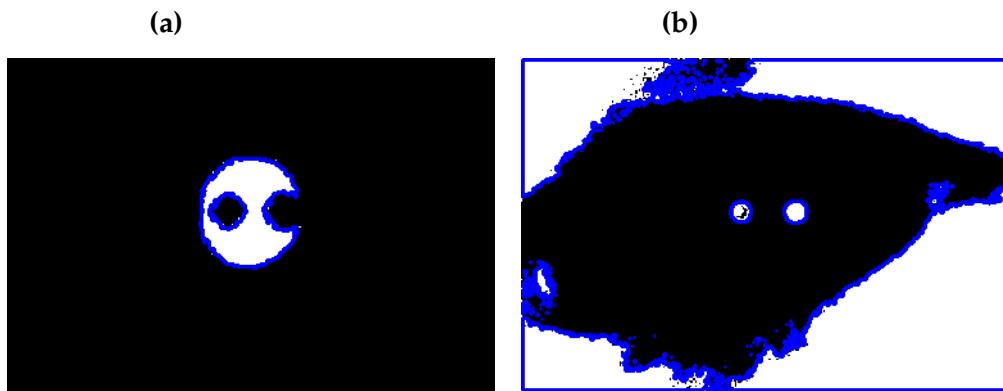
Segmentacja jest najprostszą metodą klasyfikacji. Ze względu jednak na znaczną rolę jaką odgrywa w algorytmie detekcji spojrzenia została ona wyodrębniona jako odzielnny etap. Segmentacja jest podziałem obrazu na spójne fragmenty. Polega na wyodrębnieniu z obrazu określonych obszarów, a następnie zaklasyfikowaniu każdego z pikseli do jednego z tych obszarów. W wyniku segmentacji obraz zostaje podzielony na obszar zainteresowania (ROI, ang. *Region Of Interest*) oraz tło. Do segmentacji zastosowano metodę progowania binarnego, czyli jedną z metod konwersji obrazu z gradacją szarości do obrazu binarnego (biało-czarnego). Efekt progowania ilustruje rysunek 8.



Rysunek 8: Efekt progowania obrazu a) przy detekcji konturu żrenicy, b) przy detekcji konturów odbić od rogówki.

### 0.3.3 Detekcja

Chociaż metody segmentacji pozwalają na wyodrębnienie z obrazu określonych obszarów nie przynoszą one żadnej o nich ilościowej informacji. Następnym krokiem jest sprowadzenie tych obszarów do konturów – obiektów geometrycznych mogących być ilościowo opisywanymi. Kontur jest zbiorem punktów reprezentujących na obrazie krzywą. W algorytmie detekcji spojrzenia do konturyzacji wykorzystano funkcję biblioteki *openCV* o nazwie *cv2.findContours()*. Funkcja jest implementacją algorytmu opisanego w pracy Teh, C., and R.T. Chin. “On the Detection of Dominant Points on Digital Curves.”. Na wejście funkcja przyjmuje obraz binarny, a na wyjście przekazuje w strukturze drzewiastej obliczone kontury. Efekt działania funkcji *cv2.findContours()* przedstawia ilustracja 9.



Rysunek 9: Efekt działania funkcji *cv2.findContours* a) przy detekcji konturu żrenicy, b) przy detekcji konturów odbić od rogówki.

### 0.3.4 Klasyfikacja

Kontury są obiektami geometrycznymi i jak wszystkie obiekty geometryczne posiadają swoje charakterystyki, a w tym i momenty. Momenty są wielkościami opisującymi kształt figury. W przypadku dyskretnym opisane są wzorem ogólnym 2:

$$m_{ij} = \sum_{x,y} \left( \text{obraz}(x,y) \cdot x^i \cdot y^j \right) \quad (2)$$

Rzędem momentu nazywa się wielkość  $(i+j)$ . Moment rzędu 0 wyraża się wzorem 3:

$$m_{00} = \sum_{x,y} (\text{obraz}(x, y)) \quad (3)$$

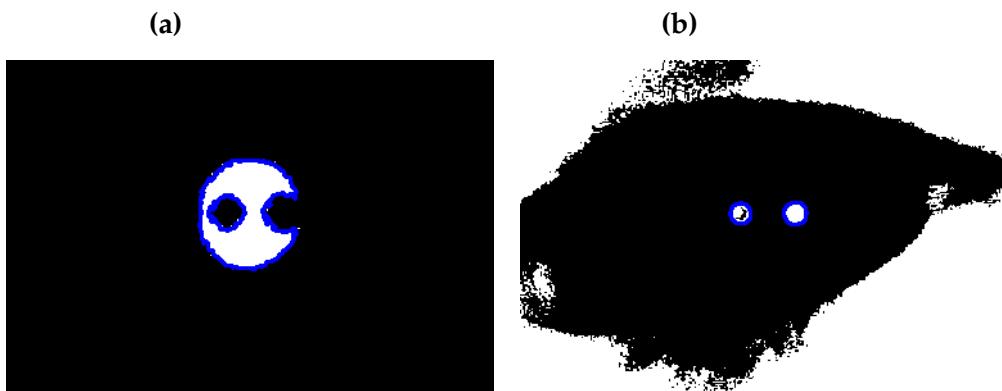
$i$  jest ilością pikseli zawierających się w konturze, będąc tym samym proporcjonalnym do pola jego powierzchni. W celu klasyfikacji na wykryte kontury nakładane są odpowiednie warunki. Kontur źrenicy jest jedynym znacząco dużym obszarem zaciemnienia na obrazie. Dla poprawnego działania algorytmu wystarczy więc zgrubne nałożenie od dołu warunku na wielkość powierzchni konturu:

1.  $m_{00} \geq \text{próg}_Z$ .

Na kontur reprezentujący odbicie rogówkowe nałożone zostały trzy warunki. Pierwszy dotyczy pola powierzchni, dwa pozostałe kolistego kształtu konturu:

1.  $m_{00} \geq \text{próg}_R$ .
2.  $|1 - \frac{m_{00}}{\pi \cdot (\frac{w}{2})^2}| \leq 0.2$ , gdzie:  $w$  - szerokość prostokąta opisanego na konturze.
3.  $|1 - \frac{w}{h}| \leq 0.2$ , gdzie:  $h$  - wysokość prostokąta opisanego na konturze.

Efekt klasyfikacji przedstawia ilustracja 10



**Rysunek 10:** Efekt działania funkcji `cv2.findContours` a) przy detekcji konturu źrenicy, b) przy detekcji konturów odbić od rogówki.

Współrzędne środka konturu reprezentującego źrenicę oraz współrzędne środka konturu reprezentującego odbicie rogówkowe są następnie podawane na wejście funkcji decyzyjnej.

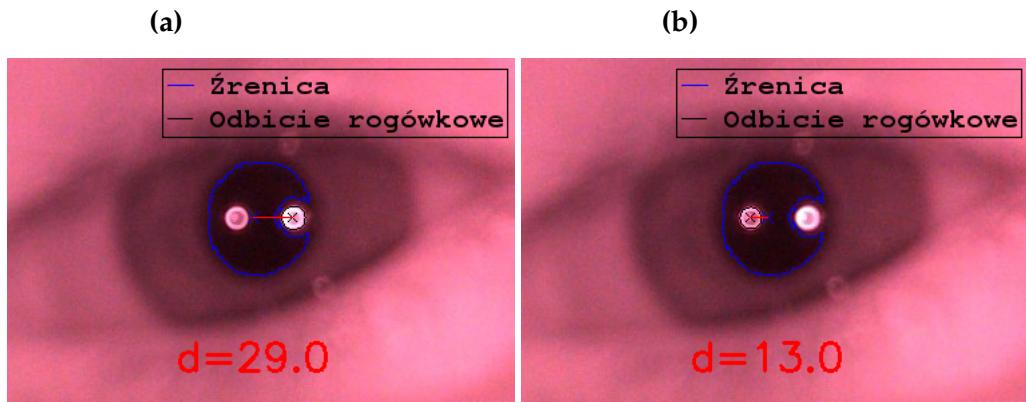
### 0.3.5 Funkcja decyzyjna

Funkcja na podstawie współrzędnych środka źrenicy i odbicia rogówkowego liczy odległość zgodnie ze wzorem:

$$d(\hat{Z}, R) = \sqrt{\left(X_{\hat{Z}} - X_R\right)^2 + \left(Y_{\hat{Z}} - Y_R\right)^2}, \quad (4)$$

gdzie:  $X_{\hat{Z}}, X_R$  – współrzędna horyzontalna środka źrenicy i odbicia rogówkowego,  $Y_{\hat{Z}}, Y_R$  – współrzędna wertykalna środka źrenicy i odbicia rogówkowego.

Efekt obliczeń na obrazie reprezentacyjnym przedstawia ilustracja 11



**Rysunek 11:** Efekt obliczeń a) przy detekcji odbicia rogówkowego od lewej diody, b) przy detekcji odbicia rogówkowego od prawej diody.

Następnie w zależności od zadanego progu funkcja przyjmuje wartość pobudzenie lub brak pobudzenia.

$$\text{Funkcja decyzyjna} = \begin{cases} \text{pobudzenie} & \text{dla } \min(d(\hat{Z}, R)) \leq \text{próg} \\ \text{brak pobudzenia} & \text{w przeciwnym razie} \end{cases}$$

Jeżeli funkcja decyzyjna przyjmie wartość pobudzenie określoną ilość razy pod rząd do GUI zostaje przesłana informacja o pobudzeniu. Efekt pobudzenia zależy od miejsca wykonywania, w którym znajduje się GUI.

### 0.3.6 Struktura katalogu

W katalogu głównym znajdują się cztery pliki źródłowe:

- *gazeSwitch.cpp*
- *show.cpp*
- *steps.cpp*
- *save.cpp*

Plik *gazeSwitch.cpp* to docelowy program. Plik *show.cpp* na strumień wideo nanosi efekt działania algorytmu detekcji źrenicy i tak przetworzone obrazy wyświetla na monitorze. Plik *steps.cpp* to program, który prezentuje kroki algorytmu na przykładowym obrazie podanym jako argument linii komend. Plik *save.cpp* zapisuje w plikach graficznych efekt działania algorytmu detekcji spojrzenia na poszczególnych klatkach strumienia filmowego.

Do komplikacji wykorzystano CMake (Cross-platform Make) - narzędzie do automatycznego zarządzania procesem kompilacji programu. Komunikację z interfejsem użytkownika zaimplementowano w bibliotece ZeroMQ (asynchroniczna biblioteka do realizacji komunikacji w środowisku rozproszonym). Zastosowano protokół nadawca-odbiorca. Program uruchomiono w systemie Linux.

