

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Informatyki

Praca dyplomowa inżynierska

na kierunku Informatyka
w specjalności Inżynieria systemów informatycznych

Wykorzystanie kamery przez aplikacje w systemie Android

Adamski Maciej
Numer albumu 300184

promotor
prof. Przemysław Rokita

WARSZAWA 2021

Spis treści

1. Dataset	5
2. Detekcja twarzy	6
2.1. Cascading Classifier	6
2.1.1. Haar Cascade	6
2.1.2. LBP Cascade	6
2.2. Deep Neural Network	6
2.3. Filtrowanie wyników	6
3. Porównanie algorytmów detekcji twarzy	7
3.1. Testowanie na statycznych zdjęciach	7
3.1.1. Oczekiwany wynik	8
3.1.2. Sposób testowania	9
3.1.3. Zbierane dane	9
3.1.4. Wyniki	10
3.2. Dodatkowe testy <i>DNN Caffe</i>	11
3.2.1. Wpływ wielkości zdjęcia na czas przetwarzania	11
3.2.2. Porównanie precyzji detekcji zależnie od sposobu filtrowania	12
3.3. Testowanie na obrazie z kamery na żywo	12
4. Detekcja oczu	13
4.1. Obcięcie obszaru detekcji	13
4.2. Filtrowanie wyników	14
5. Detekcja żrenic	15
5.1. Algorytm CDF	15
5.1.1. Kroki algorytmu	16
5.1.2. Wynik kolejnych etapów algorytmu	17
5.2. Algorytm PF	17
5.3. Algorytm EA	17
6. Landmarks	18
6.1. OpenCV-contrib	18
6.1.1. FacemarkLBF	18
7. EAR - Eye Aspect Ratio	19
7.1. Wzór obliczania EAR	19
7.2. Zasada działania EAR w kontekście mrugania	19
7.3. Testowanie z użyciem landmarków LBF opencv-contrib	20
7.3.1. Test z użyciem kamery na żywo	20
7.3.2. Niedokładność nakładania landmarków	23
Bibliografia	25
Spis rysunków	26

0. Spis treści

Spis tabel	26
Spis załączników	27

1. Dataset

Na potrzeby badań i porównania algorytmów przygotowałem dataset składający się z 80 zdjęć zawierających twarze.

Źródła zdjęć:

- 50 zdjęć wybranych z datasetu *Young and Old Images Dataset* [1]
- 30 zdjęć mojego autorstwa

Dataset został przygotowany w taki sposób, żeby zawierał zróżnicowane zdjęcia pod wieloma względami, takimi jak: jakość obrazu, oświetlenie, powierzchnia zajmowana przez twarz, kolorystyka, płeć, kolor skóry, częściowe zakrycie twarzy, okulary czy odwrócona w bok głowa. Wszystkie 80 zdjęć zostało opisanych przeze mnie na potrzeby badań, w szczególności: obszar twarzy, oczu czy środek źrenic.

Dodatkowo do badania detekcji źrenic zostanie użyty *MRL Eye Dataset* [2] zawierający zdjęcia oczu wraz z pozycją środka źrenicy.

2. Detekcja twarzy

_____ TODO _____

2.1. Cascading Classifier

_____ TODO _____

2.1.1. Haar Cascade

_____ TODO _____

2.1.2. LBP Cascade

_____ TODO _____

2.2. Deep Neural Network

Głębokie sieci neuronowe bazujące na różnych modelach dostają na wejście obraz. Na ich podstawie tworzą plamki o ustalonej wielkości, a następnie przepuszczają je przez kolejne warstwy celem wykrycia pożądanych obiektów.

W projekcie zostanie wykorzystany do tego pakiet pakiet *OpenCV-contrib* [3] zawierający implementację DNN.

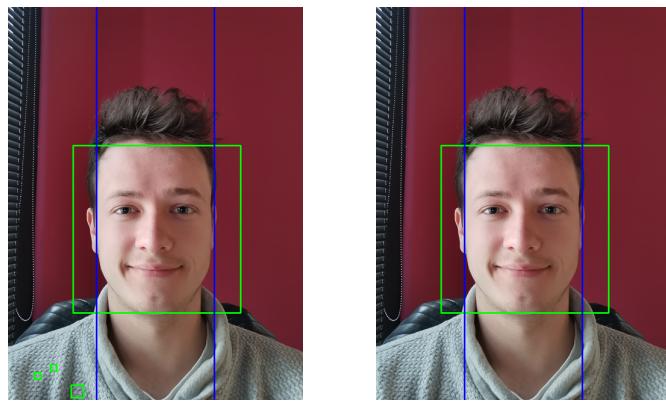
Jednym z modeli dostępnych do detekcji twarzy przy pomocy głębokich sieci neuronowych są modele Caffe (*Convolutional Architecture for Fast Feature Embedding*) [4]. Aktualnie używany w projekcie wzorzec caffe to *res10_300x300_ssd_iter_140000_fp16* [5].

2.3. Filtrowanie wyników

Użyte algorytmy mogą dawać w wyniku błędnie określone obszary twarzy. Z tego względu zwróconą tablicę obszarów poddaje filtrowaniu.

Etapy filtracji obrazów:

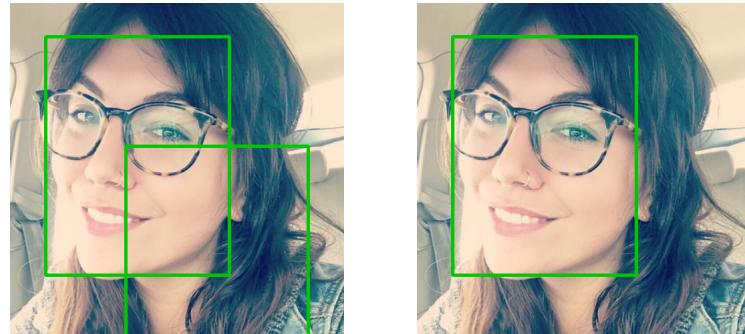
- Na początku odrzucam obszary, których środek znajduje się poza ustalonym pionowym obszarem (przyjąłem przedział [0.25, 0.75] szerokości). Wynika to z założeń, że osoba używająca telefonu, korzysta z niego patrząc na wprost, a nie z boku. Natomiast odchył od pionu to indywidualne preferencje - dlatego nie określам poziomego obszaru. (patrz *Rysunek 2.1*)
- Kolejnym etapem jest odrzucenie tych detekcji, które wychodzą zbyt daleko poza zdjęcie. Jeśli którykolwiek z boków prostokąta wystaje pionowo/poziomo o odległość większą niż 10% odpowiednio wysokości/szerokości to zostaje odrzucony. (patrz *Rysunek 2.2*)
- Z pozostałych obszarów wybieram ten, który zajmuje największą powierzchnię. Taki wybór motywuję własnymi obserwacjami zachowania algorytmów detekcji twarzy oraz tym, że głowa użytkownika telefonu na obrazie z kamery przedniej zajmuje



(a) Przed filtrowaniem zależnym od położenia

(b) Po filtrowaniu

Rysunek 2.1. Działanie filtrowania detekcji twarzy w oparciu o położenie twarzy w centralnej części zdjęcia.



(a) Przed filtrowaniem zależnym od wystawiania poza obraz

(b) Po filtrowaniu

Rysunek 2.2. Działanie filtrowania detekcji twarzy w oparciu o odległość wykrytego obszaru poza zdjęcie.

większą część płaszczyzny, ponieważ korzystając z urządzenia nie trzymamy go bardzo daleko od siebie. (patrz *Rysunek 2.3*)

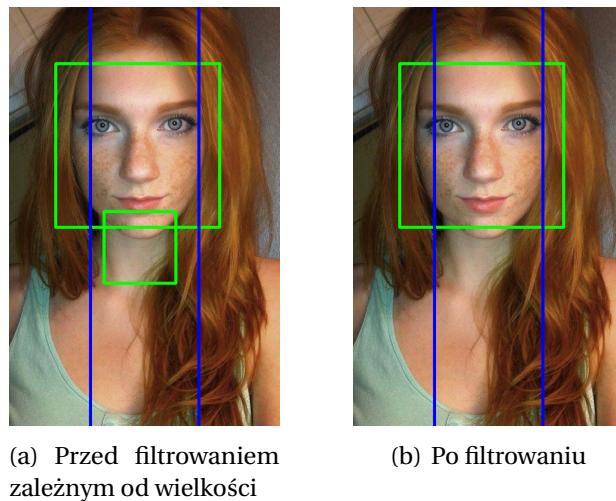
3. Porównanie algorytmów detekcji twarzy

_____ TODO _____

3.1. Testowanie na statycznych zdjęciach

Pierwszy etap testowania algorytmów detekcji twarzy będzie bazował na statycznych zdjęciach z głównego datasetu (patrz rozdz. 1.*Dataset*). Pozwoli to przetestować na

3. Porównanie algorytmów detekcji twarzy



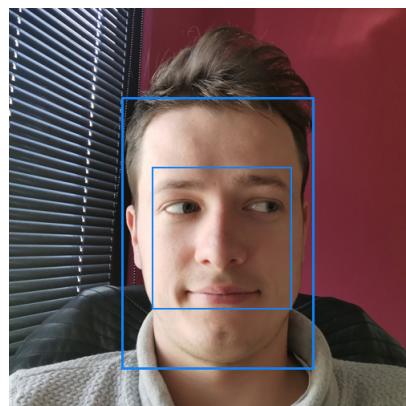
Rysunek 2.3. Działanie filtrowania detekcji twarzy w oparciu o wielkość wykrytego obszaru. Źródło zdj.: [6]

jednakowych danych wszystkie metody pod względem ich skuteczności wykrywania docelowych obszarów w różnych warunkach oraz uzyskać miarodajne wyniki.

3.1.1. Oczekiwany wynik

Każde zdjęcie z datasetu do tego etapu zostało opisane przez dwa prostokąty między którymi powinna się znaleźć wykryta przez algorytm twarz. Obszar ten został dobrany w następujący sposób:

- Wewnętrzna część obejmuje minimalny obszar, na którym znajdują się brwi, oczy, nos i usta.
- W zewnętrznym prostokącie powinna znaleźć się cała twarz. Powiększony jest on o pewną tolerancję.



Rysunek 3.1. Oczekiwany obszar detekcji twarzy.

3.1.2. Sposób testowania

Dla każdego algorytmu zostanie przeprowadzone testy na zestawach obrazów o następujących rozdzielczościach i przestrzeniach barw:

- 300x300 RGB
- 500x500 RGB
- 300x300 skala szarości
- 500x500 skala szarości

W przypadku *DNN Caffe* nie jest możliwe przeprowadzenie badań dla zdjęć w skali szarości, ponieważ wymaga on obrazu z trzema kanałami barw.

3.1.3. Zbierane dane

Dla każdego rodzaju testu i algorytmu zostaną zebrane następujące dane:

- **Prawidłowe detekcje** - suma perfekcyjnych i częściowo dobrych detekcji
- **Perfekcyjne detekcje** - jeśli wykryty obszar w pełni znajduje się pomiędzy oczekiwany prostokątami
- **Częściowo dobre detekcje** - jeśli są krawędzie, które znajdują się poza oczekiwany obszarem, ale w zadowalającej odległości (patrz niżej - *Uwaga 1.*)
- **Z 3 na 4 krawędzie perfekcyjne detekcje** - jeśli tylko jedna krawędź znajduje się poza oczekiwany obszarem w zadowalającej odległości (patrz niżej - *Uwaga 2.*)
- **Złe detekcje** - jeśli twarz nie została wykryta lub wskazany obszar jest niezadowalający
- **Twarze niewykryte** - jeśli całkowicie nie udało się wykryć twarzy (patrz niżej - *Uwaga 3.*)
- **Średnia procentowa odległość błędnych krawędzi** - wyrażony w procentach średni stosunek odległości krawędzi do szerokości/wysokości maksymalnego dopuszczalnego obszaru dla błędnie wykrytych stron
- **Średni czas przetwarzania jednego zdjęcia** - wyrażony w sekundach czas potrzebny na przetworzenie jednego zdjęcia przez daną metodę (patrz niżej - *Uwaga 4.*)

Uwaga 1. Obszar uznany jest za częściowo dobry jeśli żadna krawędź nie jest oddalona o więcej niż $1.2x$ i maksymalnie jedna oddalona jest o długość z przedziału $[1.1x, 1.2x]$. Odległość x to szerokość lub wysokość (zależnie od krawędzi) maksymalnego oczekiwanej obszaru twarzy.

Uwaga 2. Obszar zaliczony jest do grupy 3/4 perfekcyjnych detekcji, jeśli 3 krawędzie znajdują się w oczekiwany obszarze, a czwarta odchylona od normy w przedziale $[1.0x, 1.2x]$.

Uwaga 3. Dodatkowy podział złej detekcji na niewykryte twarze wynika z faktu, że metody oparte o *Cascading Classifier* na wyjściu podają obszar kwadratowy i przy rozciągniętej lub pochylonej twarzy boczne obszary mogą być bardzo oddalone od oczekiwanej wartości, ale dalej wykryć twarz.

3. Porównanie algorytmów detekcji twarzy

Uwaga 4. Celem miarodajnego wyniku czasu przetwarzania każdy test zostanie przeprowadzony 20 razy, a wyniki uśrednione.

3.1.4. Wyniki

Tabela 3.1. Wynik porównania algorytmów detekcji twarzy dla obrazów RGB

	Prawidłowa detekcja	Perfekcyjna detekcja	Częściowa dobra detekcja	3/4 krawędzie perfekcyjne	Zła detekcja	Niewykryte twarze	Średnia odległość zlych	Średni czas przetwarzania pojedynczego zdjęcia
Haar Cascade 500x500	69	4	65	32	11	9	6,07 %	0,072 s
Haar Cascade 300x300	68	5	63	33	12	9	6,29 %	0,028 s
LBP Cascade 500x500	58	4	54	29	22	22	6,22%	0,039 s
LBP Cascade 300x300	61	4	57	34	19	17	6,29 %	0,014 s
DNN Caffe 500x500	80	68	12	11	0	0	5,49 %	0,079 s
DNN Caffe 300x300	80	62	18	18	0	0	4,87 %	0,073 s

Tabela 3.2. Wynik porównania algorytmów detekcji twarzy dla obrazów w skali szarości

	Prawidłowa detekcja	Perfekcyjna detekcja	Częściowa dobra detekcja	3/4 krawędzie perfekcyjne	Zła detekcja	Niewykryte twarze	Średnia odległość zlych	Średni czas przetwarzania pojedynczego zdjęcia
Haar Cascade 500x500	69	4	65	31	11	9	6,40 %	0,073 s
Haar Cascade 300x300	67	4	63	34	13	9	6,43 %	0,028 s
LBP Cascade 500x500	60	5	55	30	20	17	6,17 %	0,038 s
LBP Cascade 300x300	60	4	56	31	20	17	6,65 %	0,013 s
DNN Caffe 500x500	nd.	nd.	nd.	nd.	nd.	nd.	nd.	nd.
DNN Caffe 300x300	nd.	nd.	nd.	nd.	nd.	nd.	nd.	nd.

Metoda *Haar Cascade* daje średnio ~ 69/80 (86%) dobrych detekcji. Jest to dosyć przecienny wynik. Na taki rezultat składa się kilka problemów tej metody. Nie radzi sobie ona dobrze z częściowo zakrytymi twarzami lub gdy głowa jest pochylona w bok. Kolejnymi czynnikiem wpływającym negatywnie na detekcję jest światło - problem z wykrywaniem występuje gdy zdjęcie jest zbyt jasne, twarz oświetlona lub źródło światła świeci prosto w obiektyw. Na plus tej metody można zapisać małą ilość zwróconych przez nią dodatkowych, błędnych obszarów, które musiały zostać odfiltrowane.

Cascading Classifier bazując na modelu *LBP* miał najgorsze wyniki detekcji twarzy, na poziomie ~ 60/80 (75%). Jednak co zwraca uwagę to fakt, że bardzo duży odsetek twarzy nie został w ogóle wykryty. Występują tu te same problemy co w *Haar Cascade*, ale dodatkowo algorytm nie radzi sobie gdy twarz zajmuje prawie całe zdjęcie.

Najlepszy wynik detekcji uzyskał bezdyskusyjnie *DNN Caffe*. Fakt, że w każdym z dwóch testów wykrył on 100% twarzy robi wrażenie. Co więcej perfekcyjne detekcje były na poziomie ~ 65/80 (81,25%). Nie występują tu problemy takie jak w poprzednich algorytmach. Radzi sobie on dobrze w złych warunkach oświetleniowych. Częściowe zakrycie twarzy nie wpływa na detekcję. Wykrywa on dobrze zarówno pochylone jak i odwrócone twarze. Jedyną negatywnym zjawiskiem, które zaobserwowałem w tej metodzie to zwracanie wielu dodatkowych obszarów, które są błędne. Zastosowanie filtrowania pozwoliło jednak odrzucić wszystkie błędne obszary.

Różnica w procencie perfekcyjnych detekcji pomiędzy *DNN Caffe*, a *LBP* i *Haar* wynika z rodzaju obszarów zwracanych przez te algorytmy. Metoda oparta na głębkich sieciach neuronowych zwraca prostokąt o dowolnym stosunku boków, natomiast druga grupa zwraca kwadrat. Dzięki temu *DNN* lepiej dopasowuje się do kształtu twarzy niż *Cascading Classifier*.

Najszybszy okazał się algorytm operujący na modelu *LBP*. *DNN Caffe* dla zdjęć 500×500 był porównywalnie szybki jak *Haar Cascade*, natomiast już w przypadku 300×300 około 2.5 razy wolniejszy.

Co ciekawe i wartere odnotowania to fakt, że algorytm *DNN* przetwarzają prawie tak samo szybko obie rozdzielczości zdjęć. Można wysnuć tezę, że dla tej metody wielkość obrazu nie ma wpływu na szybkość przetwarzania. Ze względu, że taka właściwość może okazać się przydatna w perspektywie dalszych etapów projektu, zamierzam zbadać tę zależność w kolejnym rozdziale.

Zmiana detekcji z trójkanałowej RGB na skalę szarości nie przyniosło żadnej zmiany zarówno w skuteczności algorytmów, jak również nie skróciło czasu detekcji.

3.2. Dodatkowe testy *DNN Caffe*

3.2.1. Wpływ wielkości zdjęcia na czas przetwarzania

Tabela 3.3. Wpływ rozdzielczości zdjęcia na detekcję DNN

	Prawidłowa detekcja	Perfekcyjna detekcja	Częściowa dobra detekcja	3/4 krawędzie perfekcyjne	Zła detekcja	Niewykryte twarze	Średnia odległość złych	Średni czas przetwarzania pojedynczego zdjęcia
300x300	80	62	18	18	0	0	4,87 %	0,064 s
500x500	80	68	12	11	0	0	5,49 %	0,065 s
1000x1000	80	67	13	13	0	0	5,31 %	0,066 s
2000x2000	80	65	15	15	0	0	4,62 %	0,062 s

Test ten potwierdza postawioną przeze mnie wcześniej tezę, że wielkość zdjęcia nie ma wpływu na szybkość przetwarzania algorytmu *DNN Caffe*. Testy w każdej rozdzielczości zostały wykonane mniej więcej w tym samym czasie, a różnica zapewne jest skutkiem obciążenia urządzenia w danej chwili i jest pomijalna.

Prawdopodobnie wynika to z faktu, że metoda ta tworzy na podstawie zdjęcia wejściowego plamki o podanej wielkości niezależnie od rozdzielczości. W zaimplementowanym algorytmie jest to rozmiar 300×300 . Dzięki temu zawsze ma on do przetworzenia taką samą ilość danych, więc czas powinien być w przybliżeniu stały.

Uwaga. Różnica czasów DNN między tym testem, a poprzednim (*patrz rozdz. 3.1.4. Wykresy*) prawdopodobnie wynika z różnych obciążień telefonu w danym momencie.

3. Porównanie algorytmów detekcji twarzy

3.2.2. Porównanie precyzji detekcji zależnie od sposobu filtrowania

Metoda oparta na głębkich sieciach neuronowych na wyjściu zwraca wiele obszarów wraz ze wskaźnikiem pewności detekcji. Im większy współczynnik tym w teorii większa szansa, że jest to obiekt, który chcieliśmy wykryć.

Z tego powodu postanowiłem porównać autorskie filtrowanie opisane wcześniej (patrz rozdz. 2.3.*Filtrowanie wyników*) i wybór detekcji z największym procentem pewności.

Tabela 3.4. Wynik porównania sposobów filtrowania detekcji

	Prawidłowa detekcja	Perfekcyjna detekcja	Częściowa dobra detekcja	3/4 krawędzie perfekcyjne	Zła detekcja	Niewykryte twarze	Średnia odległość złych	Średni czas przetwarzania pojedynczego zdjęcia
Autorskie filtrowanie 300x300	80	62	18	18	0	0	4,87 %	0,069 s
Autorskie filtrowanie 500x500	80	68	12	11	0	0	5,49 %	0,072 s
Najwyższy współczynnik pewności 300x300	76	58	18	18	4	4	4,87 % %	0,064 s
Najwyższy współczynnik pewności 500x500	76	64	12	11	4	4	5,49 %	0,068 s

Jak widać zaproponowana przeze mnie wcześniej sekwencja filtrowania wykrytych obszarów daje lepsze rezultaty niż wybór najwyższego współczynnika pewności.

3.3. Testowanie na obrazie z kamery na żywo

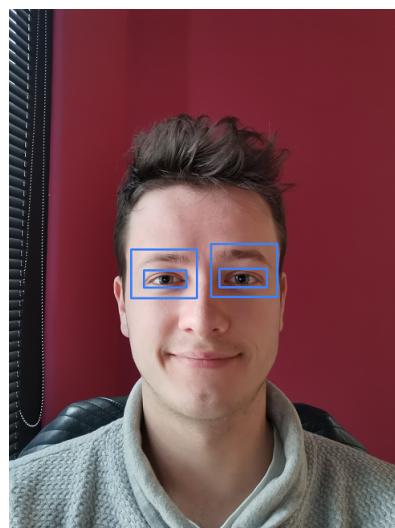
_____ TODO _____

4. Detekcja oczu

Przed przystąpieniem do detekcji oczu należy wyznaczyć obszar, na którym wykryta została twarz (patrz rozdz. 2.*Detekcja twarzy*).

Następnie obcinając klatkę tylko do ustalonego prostokąta wykrywam oczy za pomocą *Cascading Classifier* - podobnie jak twarz. ————— to będzie usunięte jeśli dam więcej algorytmów —————

Wynik który chcę uzyskać - wykryte oczy powinny się znaleźć pomiędzy naniesionymi prostokątami: ————— to będzie przeniesione do badania ewentualnie, jeśli dam więcej algorytmów —————



Rysunek 4.1. Przybliżony obszar oczu, który chcę wykrywać

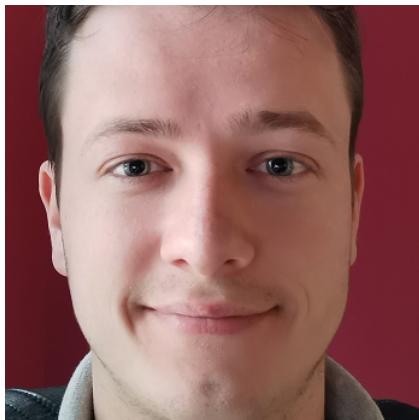
4.1. Obcięcie obszaru detekcji

Dodatkowo - prócz detekcji jedynie na obszarze twarzy - zdecydowałem się zawęzić płaszczyznę przeszukiwań. Wstępnie metodą prób i błędów dobrałem następujące parametry obcięcia obszaru:

- Góra: 0.1
- Dół: 0.45
- Lewo: 0.1
- Prawo: 0.1

Parametr określa jaka część obszaru zostaje pominięta z poszczególnych stron. Wynik takiego obcięcia widoczny jest na *rysunku 4.2*.

Takie zawężenie obszaru detekcji pozwoliło wyeliminować część z błędnie oznaczonych oczu. Na *rysunku 4.2* widoczne są dodatkowe wykryte obszary oraz ich odrzucenie dzięki temu przekształceniu.

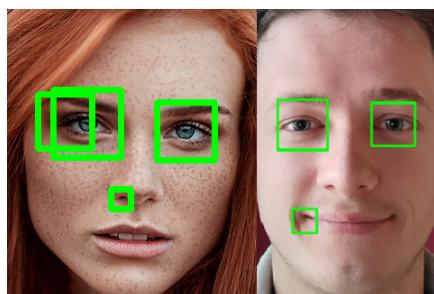


(a) Wykryty obszar twarzy

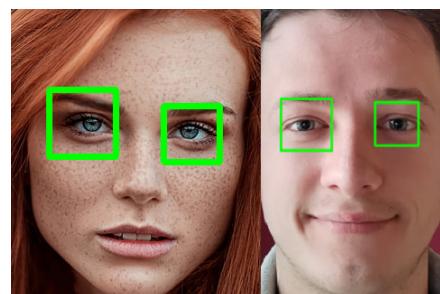


(b) Wycięty obszar oczu

Rysunek 4.2. Obcięcie obszaru detekcji oczu zgodnie z dobranymi wcześniej parametrami



(a) Wykrywanie oczu bez dodatkowego obcięcia obszaru



(b) Wykrywanie oczu z dodatkowym obcięciem obszaru

Rysunek 4.3. Odrzucenie błędnych rezultatów detekcji oczu po dodatkowym obcięciu obszaru.
Źródło pierwszego zdj.:[7]

_____ przyszłości zrobić testy na wielu zdjęciach wraz z wynikami przed/po w formie liczbowej. Wtedy dobrać testowo nowe parametry _____

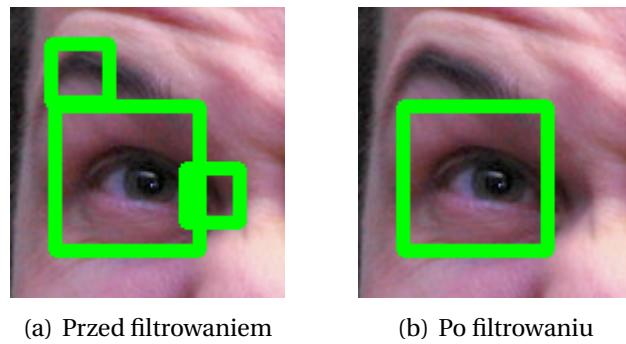
4.2. Filtrowanie wyników

Ze względu na możliwość błędnych wskazań wprowadziłem filtrowanie wyników detekcji oczu.

Algorytm filtrowania składa się z dwóch etapów:

- Podzielenie wykrytych obszarów na dwie grupy - na lewą i prawą stronę twarzy
- W obu grup wybranie największego obszaru

Przykład rezultatu takiego filtrowania pokazany jest na *rysunku 4.4*. Dodatkowo pozwoliło to na łatwe zidentyfikowanie który obszar to które oko i ich posortowanie.



Rysunek 4.4. Efekt filtrowania obszarów detekcji oczu

5. Detekcja źrenic

Do wykrywania źrenic, najpierw musimy wyznaczyć obszar oczu (patrz rozdz. 4.*Detectja oczu*). Następnie korzystając z jednych z poniższych metod ustalamy interesujący nas środek źrenicy.

Wynik, który w przybliżeniu chcę uzyskać: — jeśli będzie więcej metod to przesunąć to do testów —

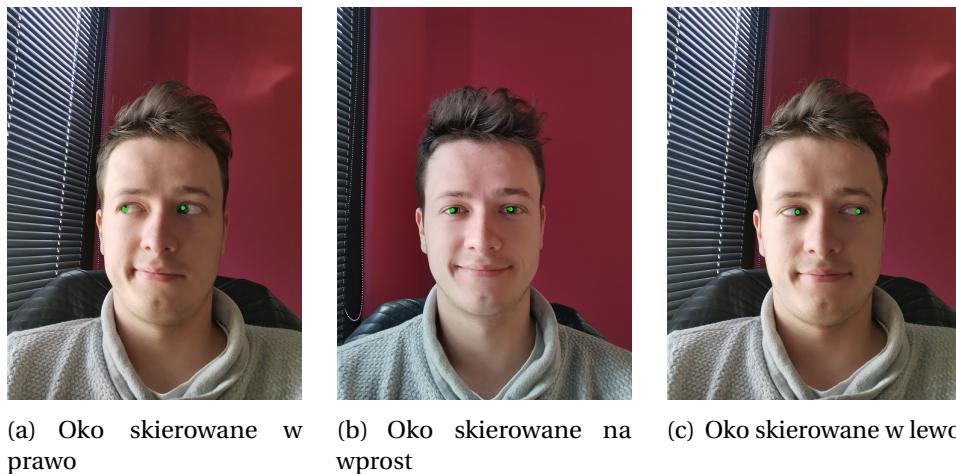


Rysunek 5.1. W przybliżeniu środek źrenic, który chcę uzyskać

5.1. Algorytm CDF

Algorytm zaimplementowany na podstawie dwóch artykułów o detekcji źrenic [8] [9]. Opiera się w głównej mierze na progowaniu za pomocą dystrybuanty. Cały algorytm przetwarza obszar oka w skali szarości.

Metoda ta daje całkiem dobre i prawdopodobnie wystarczające rezultaty.



Rysunek 5.2. Rezultat wykrywania źrenic metodą CDF

5.1.1. Kroki algorytmu

- Za pomocą progowania z użyciem dystrybuanty tworzymy obraz binarny.

$$CDF(r) = \sum_{w=0}^r p(w) \quad (1)$$

Gdzie $p(w)$ to prawdopodobieństwo znalezienia punktu o jasności równej w - określone przy pomocy dystrybuanty dystrybuanty.

$$I'(x, y) = \begin{cases} 255, & CDF(I(x, y)) < a \\ 0, & \text{inne} \end{cases} \quad (2)$$

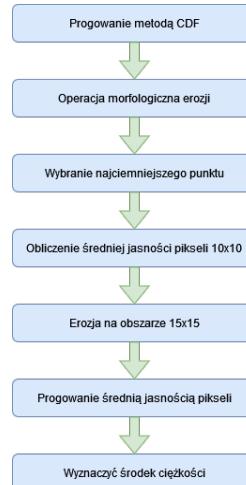
Gdzie I to jasność piksela, natomiast a to ustalony próg

- Na uzyskany obraz binarny nakładamy operację morfologiczną erozji (filtr minimałny), celem usunięcia pojedynczych ciemnych pikseli
- Znajdujemy najciemniejszy piksel na oryginalnym obrazie wśród tych, które mają wartość 255 (są białe) na obrazie binarnym
- Obliczamy średnią jasność pikseli w kwadracie 10x10 wokół wybranego najciemniejszego punktu
- Nakładamy erozję na obszarze 15x15 wokół wybranego punktu
- Na tym obszarze stosujemy progowanie

$$I'(x, y) = \begin{cases} 255, & I(x, y) < AVG_I \\ 0, & \text{wp.p.} \end{cases} \quad (3)$$

Gdzie AVG_I to średnia jasność obszaru obliczona wcześniej

- Środkiem źrenicy będzie środek ciężkości białych punktów na binarnym obszarze, który uzyskaliśmy



Rysunek 5.3. Kroki algorytmu metodą CDF

5.1.2. Wynik kolejnych etapów algorytmu

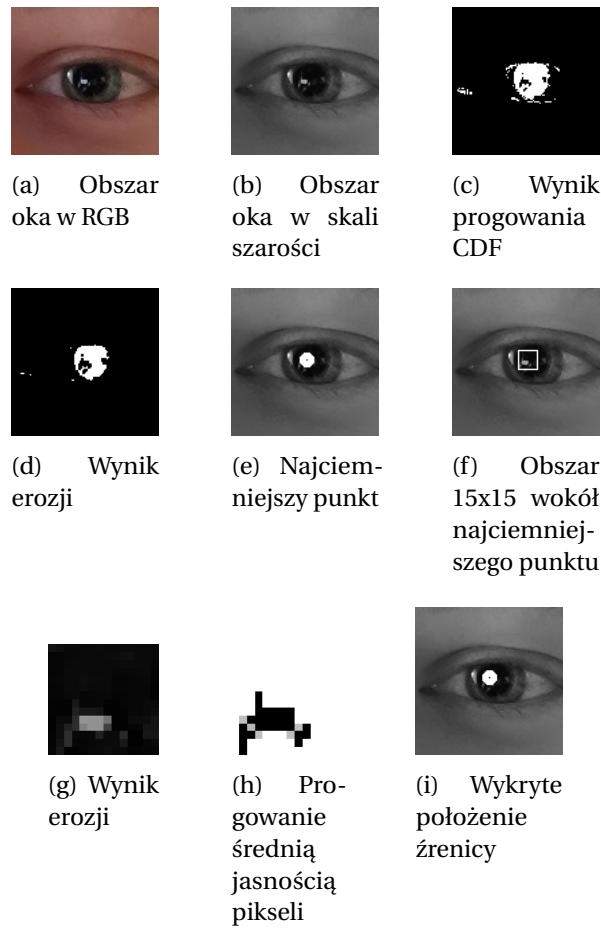
Na *rysunku 5.4* przedstawiony jest na przykładzie wynik kolejnych etapów detekcji źrenic za pomocą metody CDF.

5.2. Algorytm PF

____ W przyszłości do testów zaimplementować algorytm PF[9] ____

5.3. Algorytm EA

____ W przyszłości do testów zaimplementować algorytm EA[9] ____



Rysunek 5.4. Kolejne etapy wykrywania źrenic metodą CDF

6. Landmarks

Są to punkty nakładane na twarz wokół interesujących obszarów - takich jak oczy, nos czy usta. Pozwalają określić położenie, rozmiar czy kształt tych obiektów. Mogą być również użyte do predykcji czy mamy zamknięte/otwarte oczy lub czy się uśmiechamy.

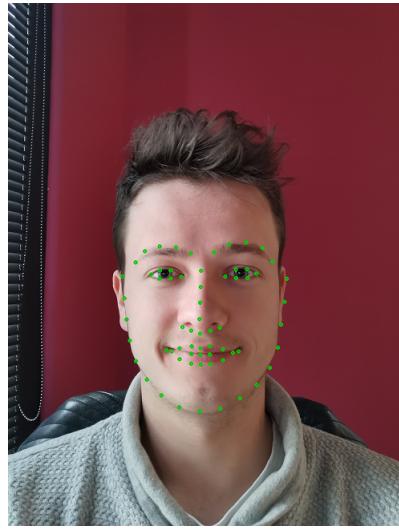
6.1. OpenCV-contrib

Dodatkowe moduł opencv facemark (*OpenCV-contrib*) zawiera trzy algorytmy detekcji landmarków:

- Kazemi
- AAM
- LBF

6.1.1. FacemarkLBF

Używając metody FacemarkLBF oraz modelu *lbfmodel.yaml* określiłem punkty orientacyjne twarzy. [10]



Rysunek 6.1. Twarz z naniesionymi landmarkami

7. EAR - Eye Aspect Ratio

Metoda polegająca na obliczeniu *EAR* [11] [12], czyli stosunek otwarcia oczu - wysokość do szerokości widocznej części gałki ocznej. Wykorzystuje się tu landmarki (patrz rozdz. 6.*Landmarks*) naniesione dookoła oczu.

7.1. Wzór obliczania EAR

Zależnie od ilości punktów wokół oka będzie różny wzór obliczania EAR.

Dla 6 punktów:

$$EAR = \frac{dist(L_0, L_1) + dist(L_2, L_4)}{2 * dist(L_3, L_5)} \quad (4)$$

Natomiast, dla 4 punktów:

$$EAR = \frac{dist(L_0, L_2)}{dist(L_1, L_3)} \quad (5)$$

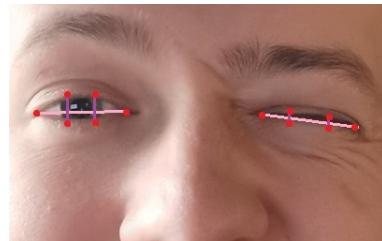
Gdzie L_x to kolejne landmarki dokoła oczu, a *dist* to odległość między dwoma punktami (odległość euklidesowa).

7.2. Zasada działania EAR w kontekście mrugania

W teorii otwarte oczy będą miały większy wymiar liczbowy EAR, niż oczy zamknięte. Na rysunku 7.1 widać, że oko otwarte ma większe odległości między punktami pionowymi niż w przypadku oka zamkniętego. Dzięki takim różnicą możemy wykryć spadek wskaźnika

7. EAR - Eye Aspect Ratio

EAR poniżej pewnego ustalonego poziomu oznaczający zamknięcie oka, natomiast wzrost otwarcie oka. Całkowicie obserwując zmiany np. za pomocą pochodnej jesteśmy w stanie stwierdzić mrugnięcie.



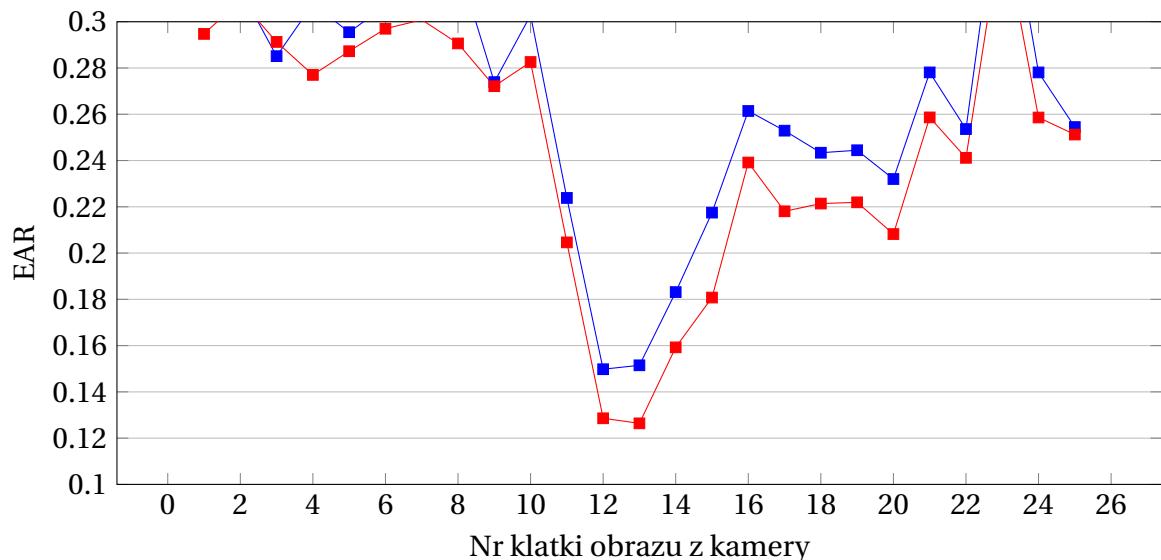
Rysunek 7.1. Teoretyczny rozmieszczenie landmarków wokół oczu wraz z naniesionymi połączonymi do obliczenia EAR

7.3. Testowanie z użyciem landmarków LBF opencv-contrib

7.3.1. Test z użyciem kamery na żywo

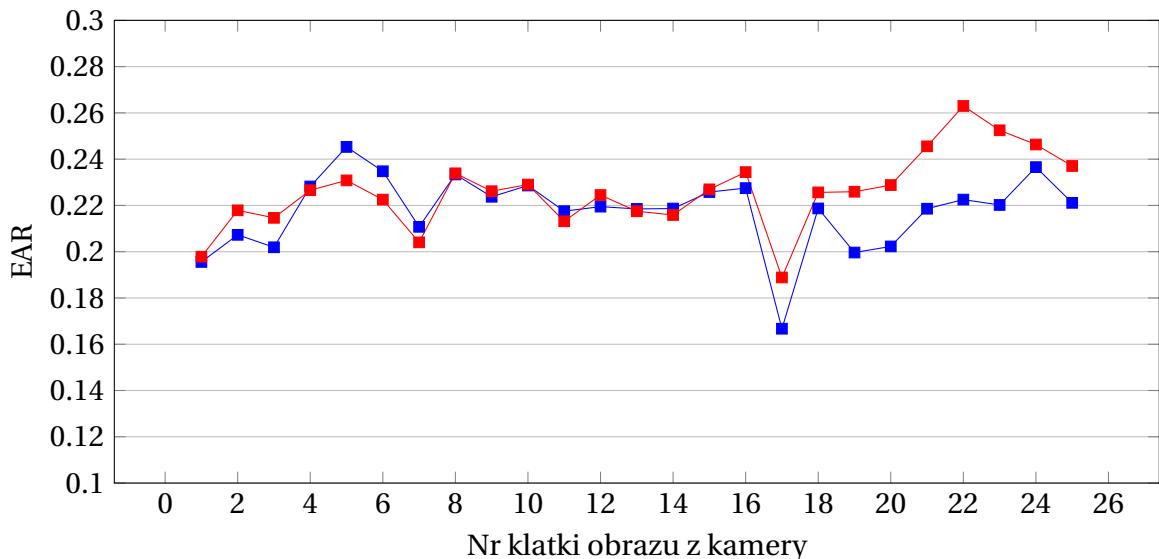
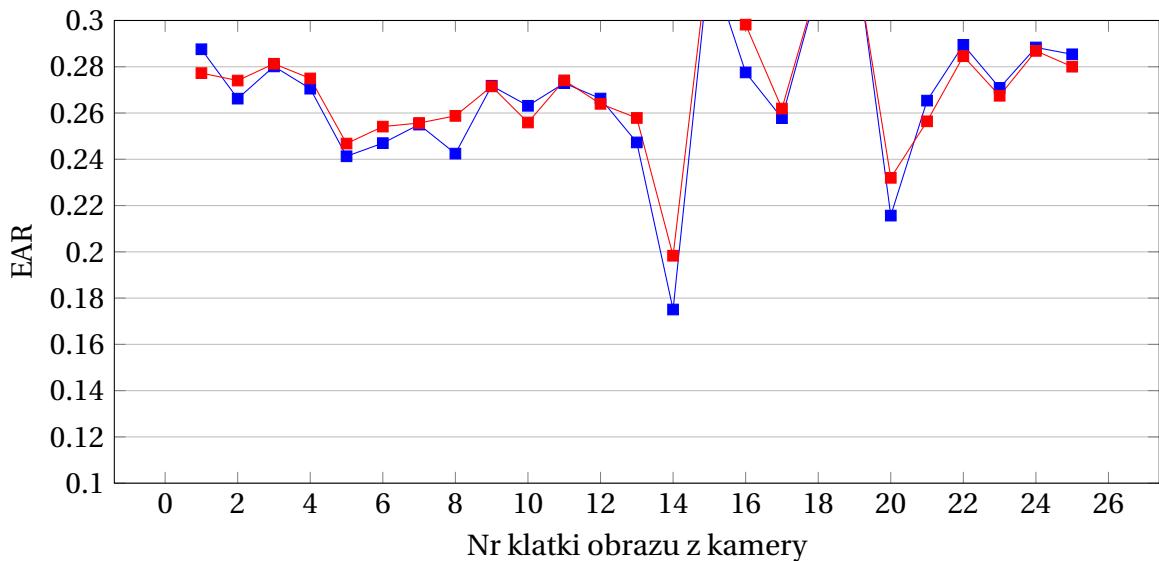
Wykonałem kilka krótkich testów z użyciem obrazu pochodzącego z przedniej kamery telefonu.

Poniżej znajdują się trzy testy, na których mrugnąłem tylko raz - lewym okiem, prawym i oboma na raz.



Rysunek 7.2. Mrugnięcie lewym okiem

Testy z pojedynczym mruganiem w krótkim okresie czasu dają przyzwoite wyniki i można na nich określić moment mrugania. W szczególności przy mrugnięciu jednym okiem.

**Rysunek 7.3.** Mrugnięcie prawym okiem**Rysunek 7.4.** Mrugnięcie oboma oczami na raz

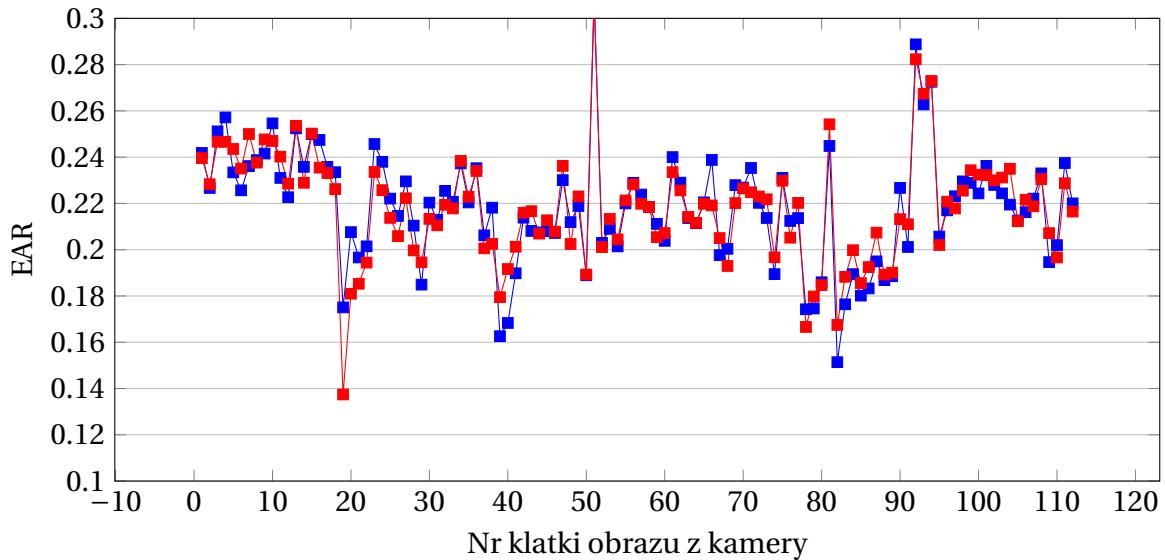
Poniżej rozciągnąłem w czasie test na sekwencje kilku mrugnięć.

Tu również z dużym prawdopodobieństwem można określić, w których klatkach wystąpiło mrugnięcie - widać gwałtowne obniżenie wartości EAR.

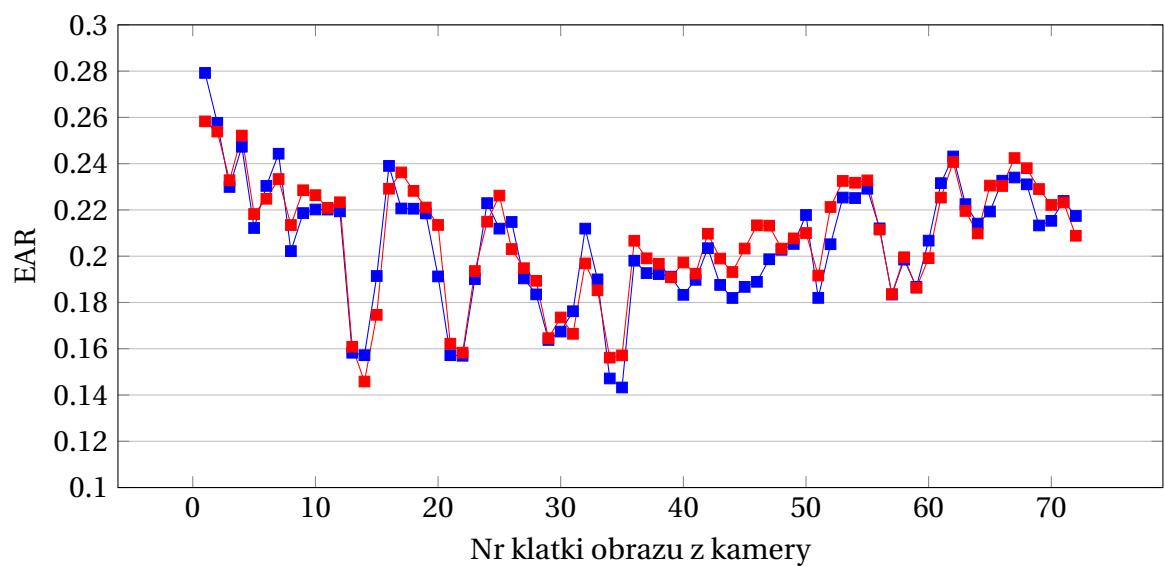
Najlepsze wyniki były w przypadku pierwszego testu, ponieważ widać wtedy znaczną różnicę EAR między otwartym ($\sim 0.26 - 0.30$), a zamkniętym okiem ($\sim 0.12 - 0.18$). Na pozostałych testach różnice nie były już tak znaczące - na ostatnich dwóch wykresach otwarte i zamknięte oko ma niewielką różnicę w EAR. Takie małe różnice wyników mogą uniemożliwić prawidłową detekcję.

W każdym z przypadków testowych EAR dla jednego i drugiego oka prawie się pokry-

7. EAR - Eye Aspect Ratio



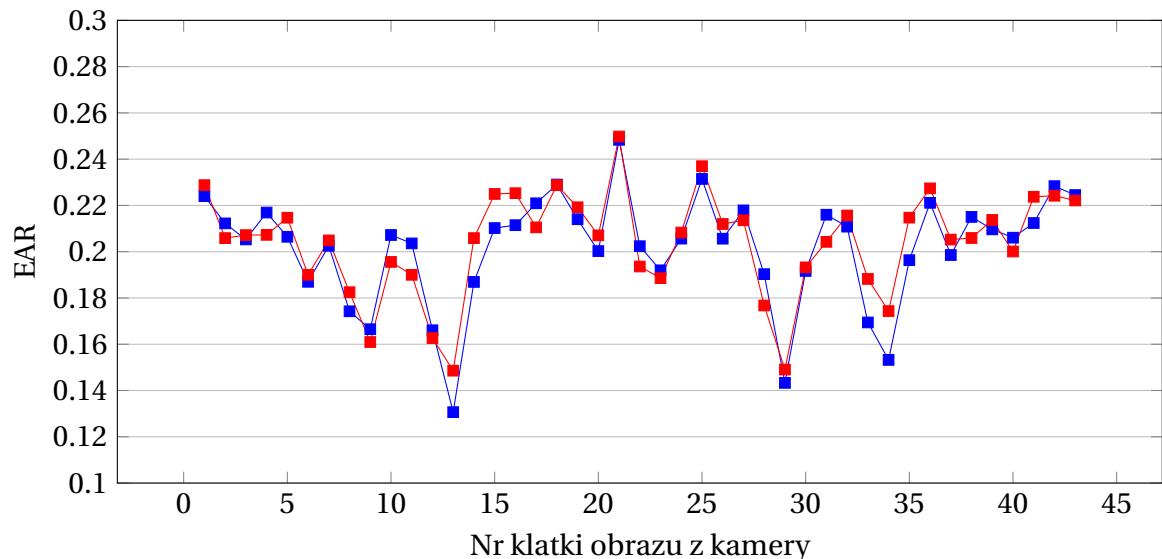
Rysunek 7.5. Kilka mrugnięć



Rysunek 7.6. Kilka mrugnięć

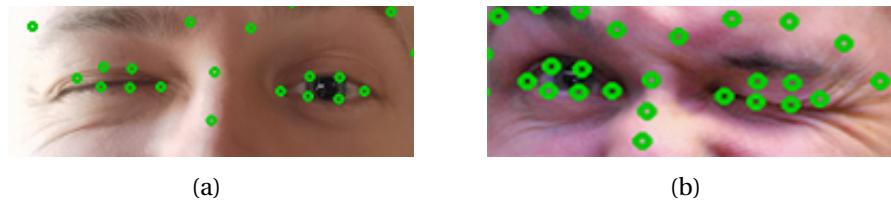
wają. Nawet przy mruganiu tylko jednym. Nie pozwoli to więc określić, którym okiem użytkownik mrugał.

Niewątpliwą trudnością w przypadku tej metody byłoby określenie progu wartości EAR, które zakwalifikowałbym jako mrugnięcie. Patrząc na wykresy powyżej można by przyjąć, że jest to wartość koło 0.18. Jednak w przypadku ostatecznego wyboru tej metody wymagałoby to dodatkowych badań celem określenia tej wartości.

**Rysunek 7.7.** Kilka mrugnięć

7.3.2. Niedokładność nakładania landmarków

Patrząc jednak na położenie tych landmarków wokół oczu mam wątpliwości co do skuteczności tej metody:

**Rysunek 7.8.** Landmarki na oczach otwartych/zamkniętych

Jak widać algorytm całkiem dobrze radzi sobie z rozmieszczeniem landmarków w przypadku otwartych oczu. Natomiast gdy oczy są zamknięte widać dużą niedokładność, która na pewno w dużym stopniu utrudnia prawidłową detekcję mrugnięcia.

Bibliografia

- [1] A. Yanamandra. "Young and Old Images Dataset". (2019), adr.: <https://www.kaggle.com/abhishekryana/young2old-dataset> (term. wiz. 18.10.2021).
- [2] M. R. L. (MRL). "MRL Eye Dataset". (), adr.: <http://mrl.cs.vsb.cz/eyedataset> (term. wiz. 18.10.2021).
- [3] OpenCV. "OpenCV-contrib". (), adr.: https://github.com/opencv/opencv_contrib (term. wiz. 24.10.2021).
- [4] Y. Jia, E. Shelhamer, J. Donahue i in., "Caffe: Convolutional Architecture for Fast Feature Embedding", *arXiv preprint arXiv:1408.5093*, 2014.
- [5] S. Mallick. "Face Detection comparison - models". (), adr.: <https://github.com/spmallick/learnopencv/blob/master/FaceDetectionComparison/models/> (term. wiz. 28.10.2021).
- [6] L. P. LLC. "Always Bet on Red". (), adr.: <https://pl.pinterest.com/pin/169025792249522554/> (term. wiz. 20.09.2021).
- [7] Nikolaj. "2018". (2013), adr.: https://www.zastavki.com/eng/Girls/Beautiful_Girls/wallpaper-126539.htm (term. wiz. 20.09.2021).
- [8] M. Asadifard i J. Shanbezadeh, "Automatic Adaptive Center of Pupil Detection Using Face Detection and CDF Analysis", w *Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 Vol I*, IMCES, Hong Kong, 2010, March 17–19.
- [9] M. Cieśla i P. Koziół, *Eye Pupil Location Using Webcam*, Wydział Fizyki, Astronomii i Informatyki Stosowanej, Uniwersytet Jagielloński, ul. Reymonta 4, 30-059, Kraków, Polska, 2012.
- [10] S. Mallick. "Facemark: Facial Landmark Detection using OpenCV". (2018), adr.: <https://learnopencv.com/facemark-facial-landmark-detection-using-opencv/> (term. wiz. 10.09.2021).
- [11] N. Kamarudin, N. A. Jumadi, N. L. Mun i in., "Implementation of Haar Cascade Classifier and Eye Aspect Ratio for Driver Drowsiness Detection Using Raspberry Pi", *Universal Journal of Electrical and Electronic Engineering*, t. 6, nr. 5B, s. 67–75, 2019.
- [12] A. Rosebrock. "Eye blink detection with OpenCV, Python, and dlib". (2017), adr.: <https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/> (term. wiz. 13.09.2021).

Spis rysunków

2.1 Działanie filtrowania detekcji twarzy w oparciu o położenie twarzy w centralnej części zdjęcia.	7
2.2 Działanie filtrowania detekcji twarzy w oparciu o odległość wykrytego obszaru poza zdjęcie.	7
2.3 Działanie filtrowania detekcji twarzy w oparciu o wielkość wykrytego obszaru. Źródło zdj.: [6]	8
3.1 Oczekiwany obszar detekcji twarzy.	8
4.1 Przybliżony obszar oczu, który chcę wykrywać	13
4.2 Obcięcie obszaru detekcji oczu zgodnie z dobranymi wcześniej parametrami .	14
4.3 Odrzucenie błędnych rezultatów detekcji oczu po dodatkowym obcięciu obszaru. Źródło pierwszego zdj.:[7]	14
4.4 Efekt filtrowania obszarów detekcji oczu	15
5.1 W przybliżeniu środek źrenic, który chcę uzyskać	15
5.2 Rezultat wykrywania źrenic metodą CDF	16
5.3 Kroki algorytmu metodą CDF	17
5.4 Kolejne etapy wykrywania źrenic metodą CDF	18
6.1 Twarz z naniesionymi landmarkami	19
7.1 Teoretyczny rozmieszczenie landmarków wokół oczu wraz z naniesionymi połączeniami do obliczenia EAR	20
7.2 Mrugnięcie lewym okiem	20
7.3 Mrugnięcie prawym okiem	21
7.4 Mrugnięcie oboma oczami na raz	21
7.5 Kilka mrugnięć	22
7.6 Kilka mrugnięć	22
7.7 Kilka mrugnięć	23
7.8 Landmarki na oczach otwartych/zamkniętych	23

Spis tabel

3.1 Wynik porównania algorytmów detekcji twarzy dla obrazów RGB	10
3.2 Wynik porównania algorytmów detekcji twarzy dla obrazów w skali szarości .	10
3.3 Wpływ rozdzielczości zdjęcia na detekcję DNN	11
3.4 Wynik porównania sposobów filtrowania detekcji	12

Spis załączników