

[INZ] Notatki

Adamski Maciej

Październik 2021

1 Dataset

Główny dataset przygotowany do wykorzystania podczas badań i porównania składa 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

2.1 Cascading Classifier

2.1.1 Haar Cascade

2.1.2 LBP Cascade

2.2 Deep Neural Network

Jeden z modułów *OpenCV-contrib* [3]. Służy do ładowania modeli głębokich sieci neuronowych i przepuszczaniu przez nie obrazów (lub ich części) celem wykrycia różnych obiektów.

2.2.1 Caffemodel

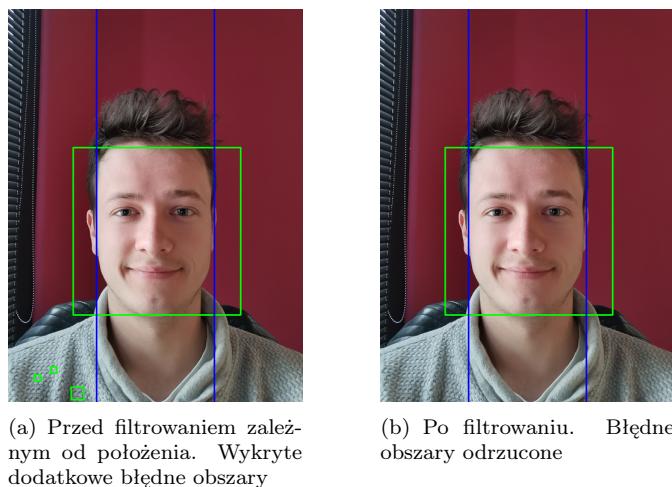
Jednym z modeli dostępnych do detekcji twarzy przy pomocy głębokich sieci neuronowych są modele Caffe (*Convolutional Architecture for Fast Feature Embedding*) [4].

Aktualnie używany w projekcie wzorzec caffe to *res10_300x300_ssd_iter_140000_fp16*.

2.3 Filtrowanie wyników

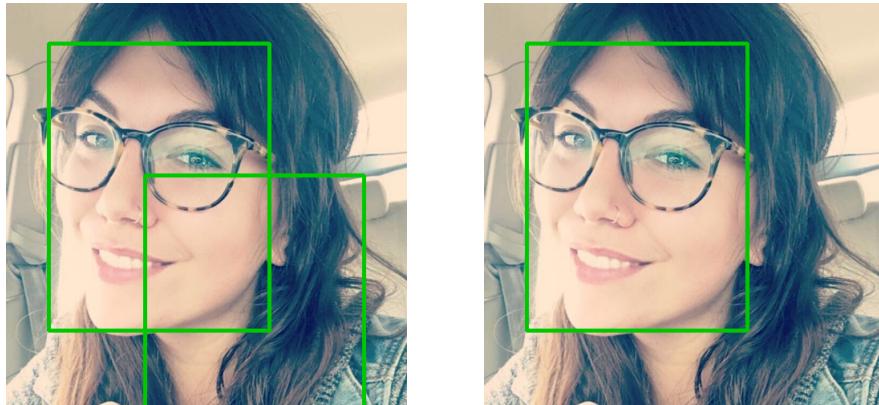
Użyte algorytmy mogą dawać w wyniku błędnie określone obszary twarzy. Z tego względu zwróconą tablicę obszarów poddaje filtrowaniu.

- Pierwszym etapem jest odrzucenie obszarów, 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.



Rysunek 1: Zielone obszary - obszary, w których według klasyfikatora może znajdować się twarz. Niebieskie pasy - obszar, w którym musi znajdować się środek twarzy.

- Kolejnym etapem jest odrzucenie wykrytych obszarów, które wychodzą zbyt daleko za obszar zdjęcia. Jeśli którykolwiek z boków prostokąta wydziela się pionowo/poziomo o odległość większą niż 10% odpowiednio wysokości/szerokości to zostaje odrzucony.

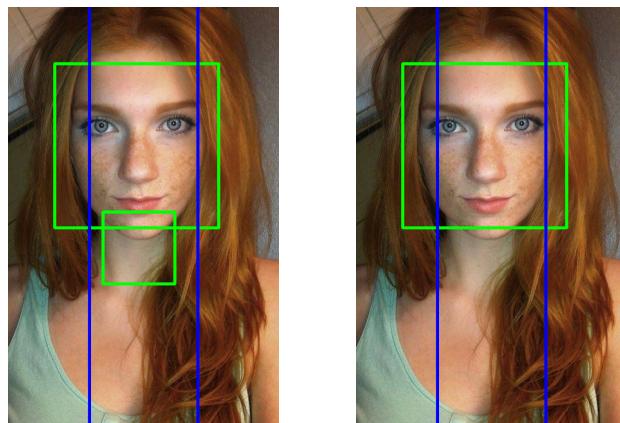


(a) Przed filtrowaniem zależnym od wystawiania poza obraz. Wykryty dodatkowy błędny obszar

(b) Po filtrowaniu. Błędny obszar odrzucony

Rysunek 2: Zielone obszary - obszary, w których według klasyfikatora może znajdować się twarz. Niebieskie pasy - obszar, w którym musi znajdować się środek twarzy.

- Z pozostałych obszarów wybieram ten, który zajmuje największą powierzchnię. Taki wybór motywuję tym, że twarz użytkownika telefonu na obrazie z kamery przedniej zajmuje większą część płaszczyzny, ponieważ korzystając z urządzenia nie trzymamy go bardzo daleko od siebie oraz własnymi obserwacjami zachowania algorytmów wykrywania twarzy.



(a) Przed filtrowaniem zależnym od wielkości. Wykryty dodatkowy błędny obszar

(b) Po filtrowaniu. Błędny obszary odrzucony

Rysunek 3: Kilka wykrytych obszarów w środkowej części. Wybieram największy. [5]

3 Porównanie algorytmów detekcji twarzy

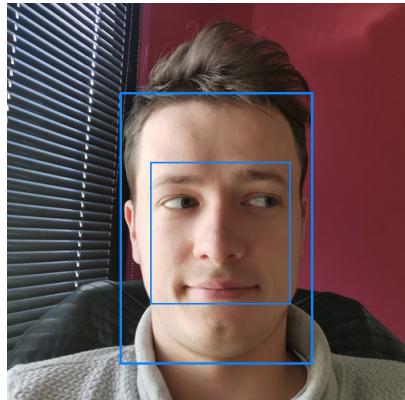
3.1 Testowanie na statycznych zdjęciach

Pierwszy etap testowania algorytmów detekcji twarzy będzie bazował na statycznych zdjęciach z datasetu (Patrz rozdz. [1. Dataset](#)). Pozwoli to przetestować na 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 powinien znaleźć się 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 o pewną tolerancję.



Rysunek 4: Oczekiwany obszar detekcji twarzy.

3.1.2 Sposób testowania

Dla każdego algorytmu zostanie przeprowadzone testy na zestawach obrazów o następujących właściwościach:

- 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 uwaga niżej)
- **Z 3 na 4 krawędzie perfekcyjne detekcje** - jeśli tylko jedna krawędź znajduje się poza oczekiwany obszarem w zadowalającej odległości (Patrz uwaga niżej)
- **Złe detekcje** - jeśli twarz nie została wykryta lub wskazany obszar jest niezadowalający
- **Twarze niewykryte** - jeśli całkowicie nie udało się wykryć twarzy
- **Średnia procentowa odległość błędnych krawędzi** - wyrażony w procentach średni stosunek odległości krawędzi do szerokości/wysokości maskymalnego dopuszczalnego obszaru dla błędnie wykrytych stron
- **Średni czas przetwarzania jednego zdjęcia**

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 oczekiwany 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 pochyłonej twarzy boczne obszary mogą być bardzo oddalone od oczekiwanej wartości, ale dalej wykryć twarz.

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

3.1.4 Wyniki

	Prawidłowa detekcja	Perfekcyjna detekcja	Częściowa dobra detekcja	3/4 krawędzie perfekcyjne	Zła detekcja	Niewykryte twarze	Średnia odległość zły	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

Tablica 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 zły	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.

Tablica 2: Wynik porównania algorytmów detekcji twarzy dla obrazów w skali szarości

Metoda *Haar Cascade* daje średnio 69/80 (86%) dobrych detekcji. Jest to dosyć przeciętny wynik. Na taki rezultat składa się kilka problemów tej metody. Nie radzi sobie ona dobrze z częściowo zakrytymi twarzami lub gdy głowa jest pochycona w bok. Nie wykrywa w ogóle twarzy jeśli 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 pochyonne 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 obszaru zwracanym przez te algorytmy. Metoda oparta na głębokich 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 500x500 był porównywalnie szybki jak *Haar Cascade*, natomiast już w przypadku 300x300 około 2.5 razy wolniejszy.

Co ciekawe i warte odnotowania to fakt, że algorytm *DNN* przetwarzał prawie tak samo szybko obie rozdzielnosci 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 następnym rozdziale.

Zmiana detekcji z trójkanałowej RGB na skalę szarości nie przyniosło żad-

nej 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

	Prawidłowa detekcja	Perfekcyjna detekcja	Częściowa dobra detekcja	3/4 krawędzie perfekcyjne	Zła detekcja	Niewykryte twarze	Średnia odległość zły	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

Tablica 3: Wpływ rozdzielczości zdjęcia na detekcję DNN

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 wynika z urządzenia w różnych chwilach i jest pomijalna.

Uwaga. Różnica czasów DNN między tym testem, a *poprzednim* prawdopodobnie wynika z jakichś obciążień telefonu w danym momencie.

3.2.2 Porównanie precyzji detekcji zależnie od sposobu filtrowania

Metoda oparta na głębokich sieciach neuronowych na wyjściu zwraca wiele obszarów wraz z 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.

	Prawidłowa detekcja	Perfekcyjna detekcja	Częściowa dobra detekcja	3/4 krawędzie perfekcyjne	Zła detekcja	Niewykryte twarze	Średnia odległość zły	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

Tablica 4: Wynik porównania sposobów filtrowania detekcji

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

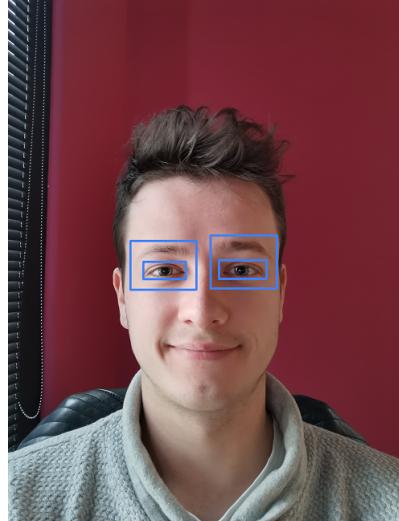
----- TODO -----

3.3 Testowanie na obrazie z kamery na żywo

4 Detekcja oczu

Przed przystąpieniem do detekcji oczu należy wyznaczyć obszar, na którym wykryta została twarz. Następnie obcinając klatkę tylko do ustalonego prostokąta wykrywam oczy za pomocą Cascading Classifier - podobnie jak twarz.

Wynik który chcę uzyskać - wykryte oczy powinny się znaleźć pomiędzy naniesionymi prostokątami:



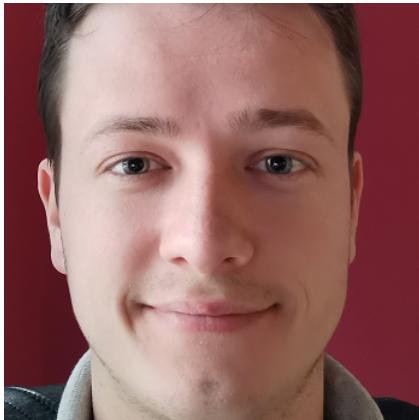
Rysunek 5: Przybliżony obszar oczu, który chcę wykrywać

4.1 Obcięcie obszaru detekcji

Dodatkowo - prócz detekcji jedynie na obszarze twarzy - zdecydowałem się za- węzić 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.



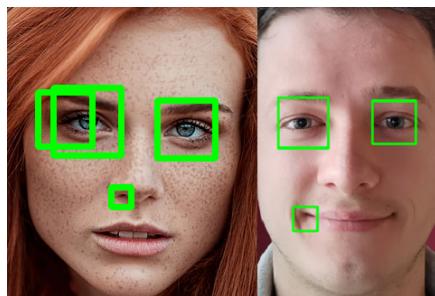
(a) Wykryty obszar twarzy



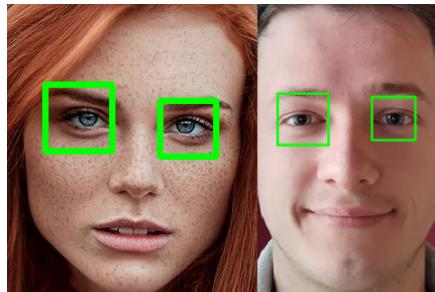
(b) Wycięty obszar oczu

Rysunek 6: Obcięcie obszaru detekcji oczu

Takie zawężenie obszaru detekcji pozwoliło wyeliminować część z błędnie oznaczonych oczu - poniżej środka twarzy czy w bok od rzeczywistego położenia:



(a) Wykrywanie oczu bez dodatkowego obcięcia obszaru



(b) Wykrywanie oczu z dodatkowym obcięciem obszaru

Rysunek 7: Poprawienie rezultatu detekcji oczu po dodatkowym obcięciu obszaru. [6]

przyszłości zrobić testy na wielu zdjęciach wraz z wynikami przed/po w formie liczbowej.

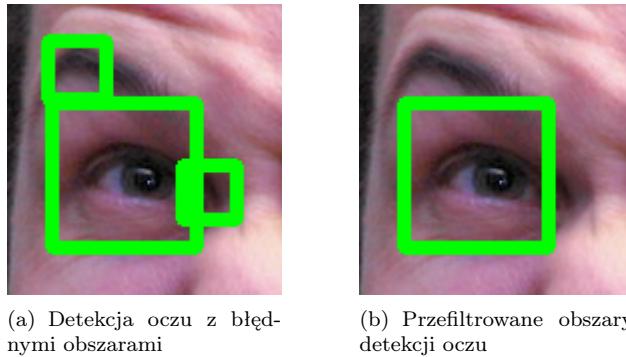
4.2 Filtrowanie wyników

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

Dodatkowo pozwoliło to na łatwe zidentyfikowanie który obszar to które oko i ich posortowanie.



Rysunek 8: Efekt filtrowania obszarów detekcji oczu

5 Detekcja źrenic

Do wykrywania źrenic, najpierw musimy wyznaczyć obszar oczu. Następnie korzystając z jednych z poniższych metod ustalam interesujący nas środek.

Wynik, który w przybliżeniu chcę uzyskać:



Rysunek 9: W przybliżeniu środek źrenic, który chcę uzyskać

5.1 Algorytm CDF

Algorytm zaimplementowany na podstawie dwóch artykułów o detekcji źrenic [7] [8]. 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.



(a) Oko skierowane w prawo

(b) Oko skierowane na wprost



(c) Oko skierowane w lewo

Rysunek 10: Rezultat wykrywania źrenic metodą CDF

5.1.1 Kroki algorytmu

- Za pomocą progowania z użyciem dystrybuanty CDF 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

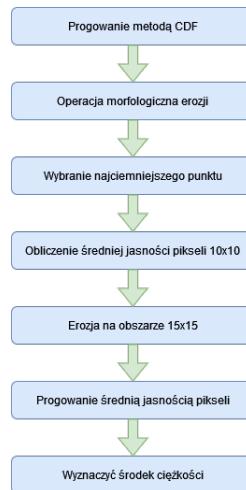
- Na uzyskany obraz binarny nakładamy operację morfologiczną erozji (filtr minimalny), celem usunięcia pojedynczych ciemnych pikseli

- Znajdujemy najciemniejszy piksel na oryginalnym obrazie wśród tych, które mają wartość 255 (są białe) na obrazie binarnym
- Obliczamy średnią jasność pikseli w kawdracie 10x10 wokół wybranego najciemniejszego punktu
- Nakładamy erozję na obszarze 15x15 wokół wybranego punktu
- Na tym obszarze stosujemy progowanie

$$I^*(x, y) = \begin{cases} 255, & I(x, y) < AVG_I \\ 0, & \text{wpp} \end{cases} \quad (3)$$

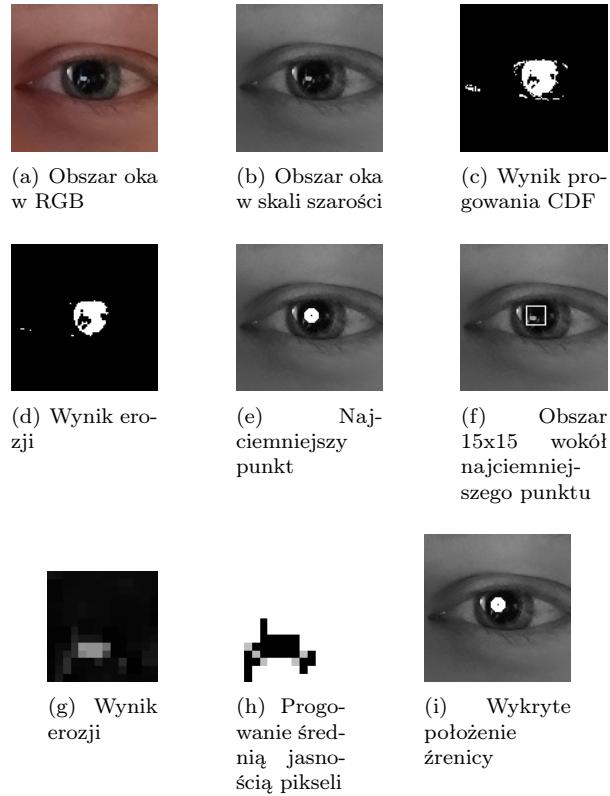
Gdzie AVG_I to średnia jasność obszaru obliczona wcześniej

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



Rysunek 11: Kroki algorytmu metodą CDF

5.1.2 Wynik kolejnych etapów algorytmu



Rysunek 12: Kolejne etapy wykrywania żrenic metodą CDF

5.2 Algorytm PF

W przyszłości do testów zaimplementować algorytm PF[8]

5.3 Algorytm EA

W przyszłości do testów zaimplementować algorytm EA[8]

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.

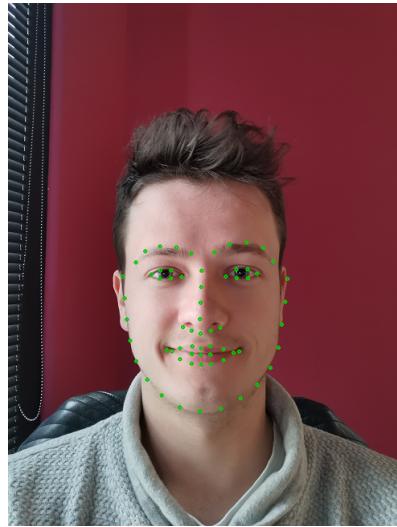
6.1 OpenCV-contrib

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. [9]



Rysunek 13: Twarz z naniesionymi landmarkami

7 EAR - Eye Aspect Ratio

Metoda wykorzystująca landmarki na twarzy. Posiadając oznaczone za pomocą tej metody oczy możemy obliczyć tzw. *EAR*, czyli stosunek otwarcia oczu - wysokość do szerokości widocznej części gałki ocznej. [10] [11]

Zależnie od ilości punktów wokół oka będzie różny wzór obliczania EAR.
Dla 6 punktów:

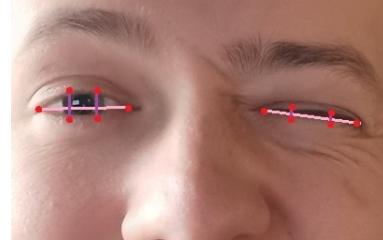
$$EAR = \frac{dist(L_0, L_1) + dist(L_2, L_4)}{2 * dist(L_3, L_5)} \quad (4)$$

Natomiast, dla 4 punktów:

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

Gdzie L_x to kolejne landmarkkiokoło oczu, a $dist$ to odległość między dwoma punktami (odległość euklidesowa).

W teorii otwarte oczy będą miały większy wymiar liczbowy EAR, niż oczy zamknięte. Na zdjęciu poniżej widać, że oko otwarte ma większe odległości między punktami pionowymi niż w przypadku oka zamkniętego.



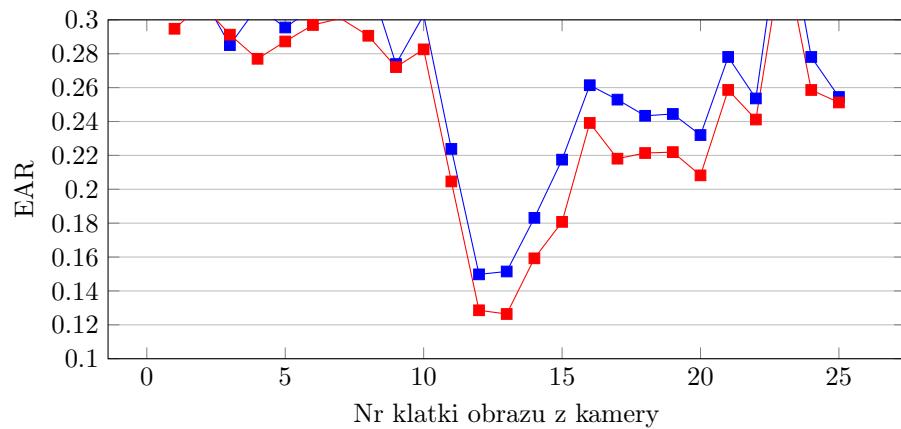
Rysunek 14: Teoretyczny rozmieszczenie landmarków wokół oczu wraz z nansionymi połączeniami do obliczenia EAR

7.1 Testowanie z użyciem landmarków LBF opencv-contrib

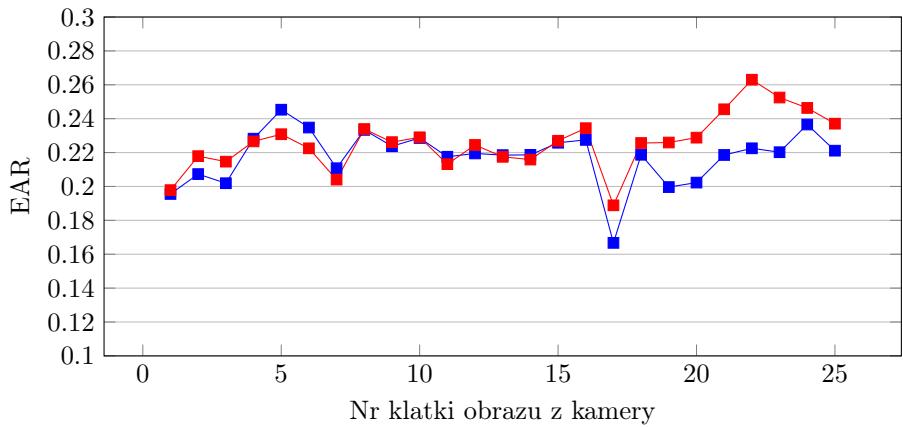
7.1.1 Test z użyciem kamery na żywo

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

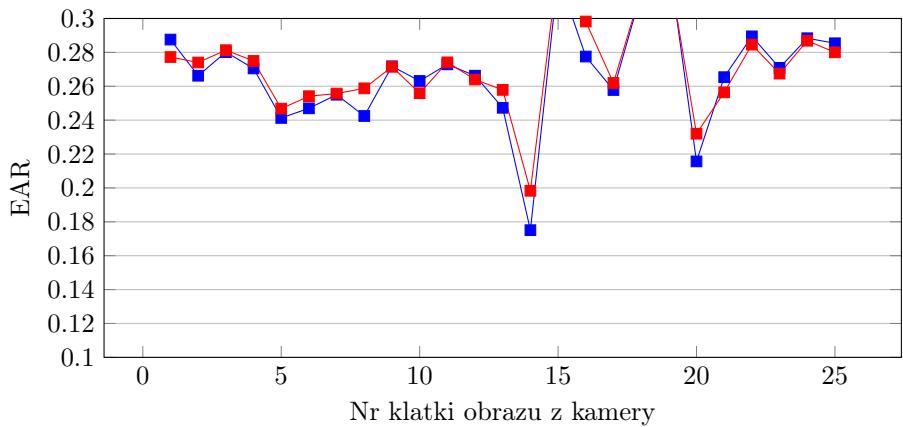
Poniżej znajdują się trzy testy, na których mrugnąłem tylko raz - lewym okiem, prawym i oboma na raz.



Rysunek 15: Mrugnięcie lewym okiem



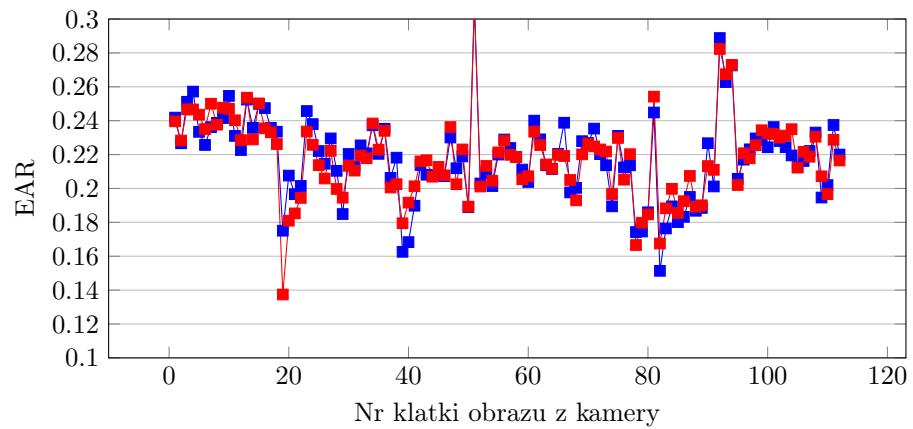
Rysunek 16: Mrugnięcie prawym okiem



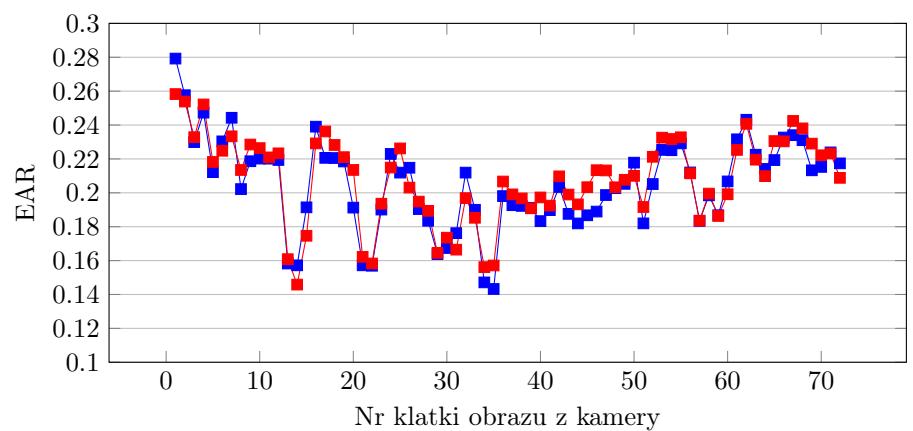
Rysunek 17: Mrugnięcie oboma oczami na raz

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

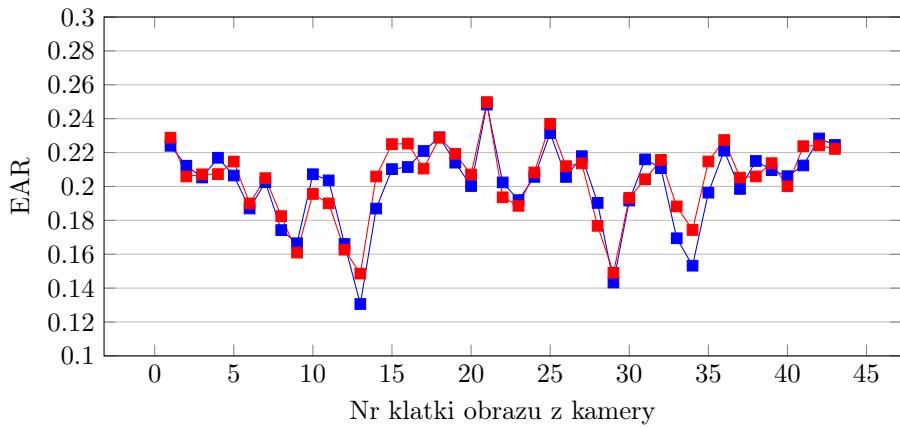
Poniżej rozciągnąłem w czasie test na sekwencje kilku mrugnięć.



Rysunek 18: Kilka mrugnięć



Rysunek 19: Kilka mrugnięć



Rysunek 20: Kilka mrugnięć

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

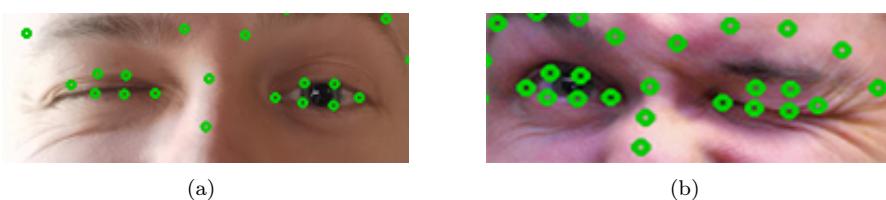
Najlepsze wyniki były w przypadku pierwszego testu, ponieważ widać wtedy znaczną różnicę EAR między otwartym ($\sim 0.26 - 0.30$), a zamkniętym okiem ($\sim 0.12 - 0.18$). Na pozostałych testach różnice nie były już tak znaczące - na ostatnich dwóch wykresach otwarte i zamknięte oko ma niewielką różnicę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

W każdym z przypadków testowych EAR dla jednego i drugiego oka prawie się pokrywają. 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.

7.1.2 Niedokładność nakładania landmarków

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



Rysunek 21: 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] Abhishek Yanamandra. *Young and Old Images Dataset*. 2019. URL: <https://www.kaggle.com/abhishekya/young2old-dataset> (term. wiz. 18. 10. 2021).
- [2] Media Research Lab (MRL). *MRL Eye Dataset*. URL: <http://mrl.cs.vsb.cz/eyedataset> (term. wiz. 18. 10. 2021).
- [3] OpenCV. *OpenCV-contrib*. URL: https://github.com/opencv/opencv_contrib (term. wiz. 24. 10. 2021).
- [4] Yangqing Jia i in. „Caffe: Convolutional Architecture for Fast Feature Embedding”. W: *arXiv preprint arXiv:1408.5093* (2014).
- [5] Legacy Photography LLC. *Always Bet on Red*. URL: <https://pl.pinterest.com/pin/169025792249522554/> (term. wiz. 20. 09. 2021).
- [6] Nikolaj. 2018. 2013. URL: https://www.zastavki.com/eng/Girls/Beautiful_Girls/wallpaper-126539.htm (term. wiz. 20. 09. 2021).
- [7] Mansour Asadifard i Jamshid Shanbehzadeh. „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.
- [8] Michał Cieśla i Przemysław Kozioł. *Eye Pupil Location Using Webcam*. Wydział Fizyki, Astronomii i Informatyki Stosowanej, Uniwersytet Jagielloński. ul. Reymonta 4, 30-059, Kraków, Polska, 2012.
- [9] Satya Mallick. *Facemark: Facial Landmark Detection using OpenCV*. 2018. URL: <https://learnopencv.com/facemark-facial-landmark-detection-using-opencv/> (term. wiz. 10. 09. 2021).
- [10] Nora Kamarudin i in. „Implementation of Haar Cascade Classifier and Eye Aspect Ratio for Driver Drowsiness Detection Using Raspberry Pi”. W: *Universal Journal of Electrical and Electronic Engineering* 6.5B (2019), s. 67–75.
- [11] Adrian Rosebrock. *Eye blink detection with OpenCV, Python, and dlib*. 2017. URL: <https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/> (term. wiz. 13. 09. 2021).