POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA MAGISTERSKA
na kierunku INFORMATYKA

Michał KOWALCZYK

Nr imm. 197596

Rok. akad. 2008/2009
Warszawa, 15 IV 2008

SYSTEM DETEKCJI I ROZPOZNAWANIA TWARZY

Zakres pracy:

1. *Przedstawienie celu pracy, opis zastosowań i problemów.*
2. *Przegląd zagadnień oraz istniejących algorytmów .*
3. *Opis własnego systemu rozpoznawania twarzy, implementacja, badania.*
4. *Wnioski i podsumowanie.*

Podpis i pieczątka

Kierownika Zakładu Dydaktycznego

Opiekun naukowy:
Dr inż. Witold Czajewski

Termin wykonania: 15 IX 2009

Praca wykonana i zaliczona pozostaje
własnością Instytutu i nie będzie
zwrócona wykonawcy

Spis treści

1. Wstęp

Rozpoznawanie twarzy jest jedną z technik biometrycznych używanych do identyfikacji osób. Początki jej rozwoju sięgają lat 80-tych XX wieku, a pierwsze komercyjne systemy powstały w latach 90-tych XX wieku [32].

Technika ta jest stosowana głównie do zwalczania przestępczości. Systemy identyfikujące ludzi na podstawie obrazu twarzy są obecnie instalowane w więzieniach, na lotniskach, w różnego rodzaju miejscach publicznych lub z ograniczonym dostępem. Służą również do zabezpieczania danych, kontroli/śledzenia pracowników w firmach oraz identyfikacji stałych klientów w sklepach. Z czasem technologia ta zaczęła zyskiwać coraz większą popularność, a wzrost wydajności komputerów przyczynił się do jej masowego wykorzystania. Twórcy komputerów oraz aparatów fotograficznych wyposażają swoje urządzenia w tego rodzaju systemy dając użytkownikom możliwość lepszego zabezpieczenia danych lub, w przypadku aparatów cyfrowych, zdolność śledzenia twarzy, bądź wykrywania uśmiechu w celu automatycznego robienia zdjęć (np. *Nikon D5000*, *Canon EOS 50D*).

Jedną z głównych cech metod rozpoznawania twarzy jest bezinwazyjność pobierania danych wystarczy spojrzeć w kamerę co sprawia że ludzie którzy nigdy by się nie zgodzili na skanowanie siatkówki/tęczówki oka lub badanie *DNA* są gotowi poddać się analizie twarzy. Bezinwazyjność jest również jedną z przyczyn protestów części opinii publicznej. Główne obawy dotyczą możliwości rozmieszczania tego rodzaju systemów w miejscach publicznych w celu zbierania danych bez wiedzy i zgody filmowanych osób. Urządzenia umożliwiające wykrywanie i detekcję twarzy wymagają specyficznych warunków żeby działać poprawnie. Wszelkiego rodzaju zmiany oświetlenia czy położenia twarzy mogą w skrajnych przypadkach całkowicie uniemożliwić identyfikację.

1.1. Cel pracy

Celem niniejszej pracy jest stworzenie systemu, który umożliwi detekcję twarzy w obrazie przechwyconym z kamery, oraz ich identyfikację. Powinien on działać w czasie rzeczywistym i mieć dużą skuteczność rozpoznawania (powyżej 80%). System będzie mógł być wykorzystany np. w firmach do śledzenia pracowników lub rozpoznawania stałych klientów w sklepach. Podobne rozwiązania już istnieją ale są to drogie programy komercyjne o zamkniętych źródłach.

1.2. Zakres pracy

Przedmiotem pracy jest budowa systemu detekcji i identyfikacji twarzy w obrazach przechwyconych z kamery. W tym celu zostały wykorzystane funkcje dostarczane przez bibliotekę *OpenCV*. Detekcja twarzy odbywa się przy użyciu funkcji *Haar-like*, umożliwiającej dość dobrą detekcje obiektów. Dodatkowo metoda jest wspierana przez funkcję wykrywającą skórę w obrazie w celu odrzucenia obiektów nie będących twarzami. Detekcja skóry jest tylko metodą wspierającą i nie musi być użyta. Rozpoznawanie zrealizowane jest za pomocą metody *EigenFace* z wykorzystaniem *PCA* (ang. Principal Component Analysis – analiza składowych głównych) w celu zmniejszenia wymiaru wektorów cech. Zakres pracy obejmuje również testy systemu w różnych warunkach oświetlenia oraz z różnym położeniem twarzy w obrazie. Do identyfikacji nie użyto wektorów opisujących odległości pomiędzy charakterystycznymi punktami twarzy takimi jak nos, kąciki ust czy oczy ponieważ jest ona wrażliwa na obroty. System ma działać na danych dostarczonych z kamery, a więc jest to obraz dynamiczny, w którym twarze mogą się pojawiać pod różnymi kątami.

Podstawowym problemem jest szybkość przetwarzania obrazu, która zależy głównie od zastosowanych algorytmów i mocy obliczeniowej komputera. Należy również zwrócić uwagę na poprawność wykrywania twarzy, gdyż w przypadku uznania za twarz fragmentu tła aplikacja jest narażona na dodatkowe koszta związane z przetwarzaniem niepotrzebnych danych. Do eliminacji obiektów tła wstępnie zakwalifikowanych jako twarz może posłużyć detekcja skóry, z którą również wiąże się kilka problemów zwłaszcza szybkość metody oraz jej skuteczność. Duże znaczenie ma

czas w jakim twarz zostanie zidentyfikowana. Przeszukiwanie bazy danych z danymi twarzy może być bardzo czasochłonne, co w przypadku aplikacji działającej w czasie rzeczywistym jest nie do przyjęcia.

1.3. Układ pracy

Praca składa się z pięciu rozdziałów. We wstępie opisano tematykę pracy, przedstawiono zagadnienia jakie zostały poruszone, wymieniono problemy związane z tą tematyką oraz systemy już istniejące. W rozdziale drugim znajduje się opis teoretyczny zagadnień związanych z tematyką pracy. Opisano biblioteki oraz metody jakie zostały zastosowane w procesie tworzenia systemu. Rozdział trzeci szczegółowo opisuje proces realizacji pracy. Znajduje się tam opis problemu, konfiguracja narzędzi oraz działania aplikacji. Ponadto opisano testy systemu i problemy jakie napotkano podczas jego tworzenia. W podsumowaniu znajdują się informacje o tym, czego udało się dokonać, co sprawiło problemy oraz propozycje dalszego rozwoju aplikacji.

1.4. Zastosowania rozpoznawania twarzy

Systemy rozpoznawania twarzy mają obecnie wiele zastosowań. Przyczyniły się do tego wzrost wydajności komputerów i rozwój algorytmów umożliwiających skuteczną detekcję i rozpoznawanie twarzy. Do najważniejszych należą:

1.4.1 Nadzór

Głównym celem systemów nadzoru jest śledzenie osób na danym terenie. Sprawdzanie czy dana osoba nie wchodzi w obszar do którego nie ma dostępu. Tego typu systemy używane są np. do śledzenia ruchu pieszych w celu zbierania danych statystycznych o kierunku przemieszczania oraz natężeniu ruchu w zależności od pory dnia. Również policyjne kamery mogą być wyposażone w możliwość detekcji i rozpoznawania twarzy w celu odnalezienia osób poszukiwanych.

1.4.2 Kontrola dostępu

Systemy te służą zazwyczaj jako dodatkowe zabezpieczenie przed nieupoważnionym dostępem do danych, budynków czy różnego rodzaju operacji w internecie lub bankach. Do tego rodzaju systemów należą między innymi systemy firmy Bosch oraz wbudowane w niektóre laptopy (*Veriface III* w komputerach *Lenovo*, *SmartLogon* w modelach *Asusa*, *Recognition* w komputerach *Toshiby*) .

1.4.3 Przeszukiwanie bazy zdjęć

Dość często wykorzystywane są w kryminalistyce do przeszukiwania bazy twarzy w celu identyfikacji przestępców. Podobne systemy są również stosowanie na lotniskach, w ambasadach czy przy odprawie celnej.

1.5. Problemy

1.5.1. Wykrywanie

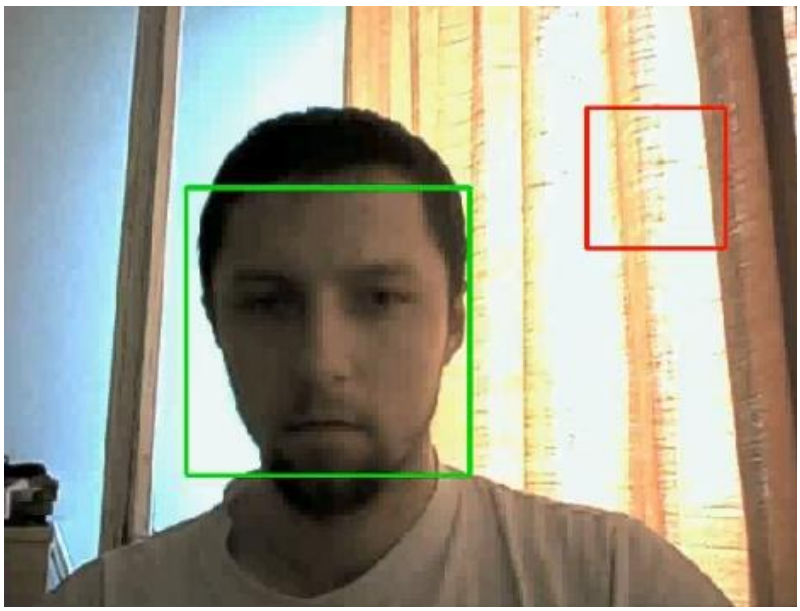
Wykrywanie twarzy nie jest prostym zagadnieniem. Istnieje wiele czynników które są w stanie zakłócić ten proces, są to między innymi:

- Warunki (oświetlenie)

  Jedną z podstawowych wad wykrywania twarzy jest wrażliwość na zmiany oświetlenia. Problem dotyczy zwłaszcza metod bazujących na rozpoznawaniu koloru skóry. Dość często cień zakrywa część twarzy wprowadzając system w błąd. Jest to jedną z przyczyn projektowania systemów wyspecjalizowanych, przystosowanych do konkretnych warunków.

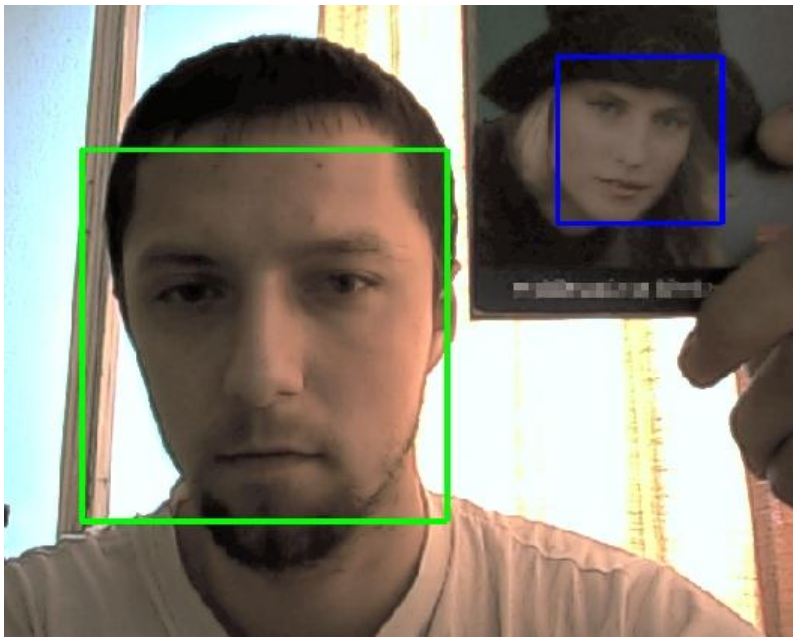
- Tło

Systemy wykorzystujące funkcję *Haar-like* bazują na rozpoznawaniu konkretnych wzorców co powoduje że losowe zakłócenie, którym może być tło lub zła jakość obrazu, prowadzi do wskazania fragmentu obrazu nie zawierającego twarzy.



Rysunek 1: Błędny fragment obrazu (czerwony prostokąt). Tło rozpoznane jako twarz.

Błędy mogą powodować również manekiny na wystawach sklepowych lub zdjęcia na reklamach. Ten problem dotyczy również metod bazujących na kolorze ludzkiej skóry. Dla systemu pracującego w centrum handlowym jest to duże obciążenie. Rozwiązaniem tego problemu jest stosowanie metod umożliwiających śledzenie obiektów poruszających się.



Rysunek 2: Zdjęcie twarzy (niebieski prostokąt).

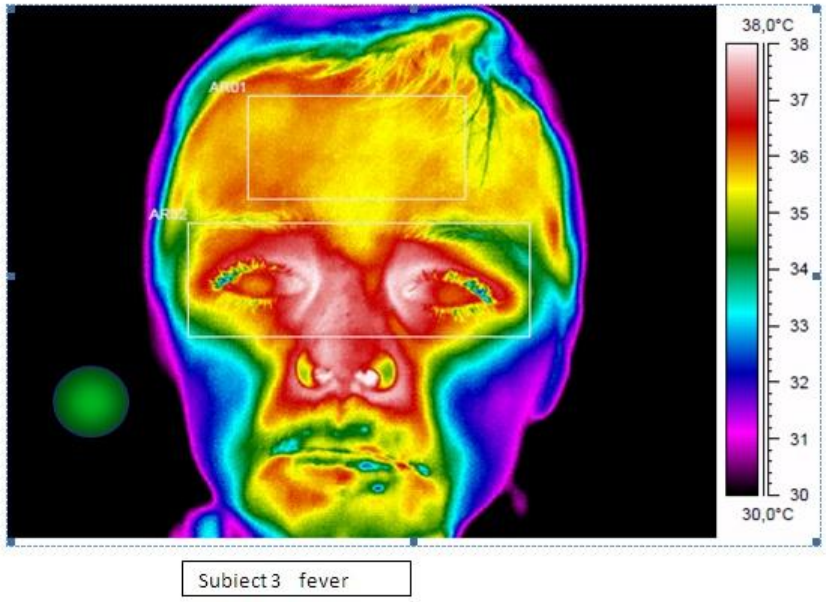
- Twarz

  Sama twarz może być przyczyną dla której nie jest możliwa jej lokalizacja w obrazie. Głównym powodem są obroty/pochylenia, które podobnie jak zarost, okulary, czapki i inne dodatki zasłaniające twarz lub owal głowy, sprawiają problem metodą bazującym na wzorcach. Wykrywanie na podstawie koloru skóry na ogół nie ma problemów z obrotami.

- Kamera

  Duże znaczenie ma jakość obrazu pochodzącego z kamery. Dotyczy to głównie systemów pracujących w czasie rzeczywistym. Niska rozdzielczość sprawia, że twarz może być zbyt zniekształcona aby była możliwa jej poprawna detekcja. Dużo kamer, zwłaszcza internetowych, ma bardzo niską jakość obrazu. Szumy podobnie jak tło dość poważnie zmniejszają szanse na wykrycie twarzy lub prawidłowe jej wskazanie. Do tego dochodzą problemy z auto fokusem oraz automatycznym dobieraniem poziomu bieli. Jeżeli kamera robi to zbyt wolno system nie będzie działał poprawnie. Rozwiązaniem są oczywiście systemy wyspecjalizowane, przystosowane do konkretnych warunków. Jednym z

rozwiązań stosowanych w celu uniknięcia problemów z detekcją twarzy jest użycie kamer termowizyjnych. Metoda ta jest bardzo skuteczna ale i kosztowna. Ceny kamer termowizyjnych są znacznie wyższe niż tradycyjnych, a ich rozdzielczość wciąż jest zbyt mała do większości zastosowań.



Rysunek 3: Obraz twarzy wykonany kamerą termowizyjną [19].

1.5.2. Rozpoznawanie

Do czynników wpływających na rozpoznawanie twarzy można zaliczyć wszystkie zakłócenia, bądź też zniekształcenia obrazu mające wpływ na poprawne wykrycie twarzy. Ponadto dochodzą problemy związane z samą twarzą, jej położeniem oraz fragmentami ubioru mogącymi ją zasłaniać. Należą do nich:

- Mimika

  Wrażliwe na nią są metody bazujące na odległościach pomiędzy charakterystycznymi punktami twarzy np. czubkiem nosa, kącikami ust, położeniem oczu. Część metod wykorzystujących wzorce również może mieć

problemy, o ile nie dysponują odpowiednio dużą bazą zdjęć. Zwykły uśmiech czy zamknięcie oczu może uniemożliwić poprawną identyfikację.

- Położenie twarzy

  Systemy wykorzystujące wzorce i dość dużą bazę danych nie mają większych problemów. Metody, które wykorzystują odległości pomiędzy punktami charakterystycznymi twarzy, nawet jeżeli dysponują dużą bazą danych, mogą mieć poważne problemy ze względu na zmianę odległości pomiędzy punktami podczas obrotów (po rzutowaniu na obraz *2D*). Może się okazać że dwie różne twarze po odpowiednim ustawieniu dadzą dość dużo danych podobnych.

- Zasłonięcie

  Jest to jeden z podstawowych problemów. Wszelkiego rodzaju okulary, maski, zarost, włosy, czapki itp. w znacznym stopniu utrudniają rozpoznanie osoby. Powodem najczęściej jest zasłonięcie punktów charakterystycznych twarzy lub zniekształcenie owalu głowy. Rozwiązaniem może być zastosowanie kamery termowizyjnej, lecz nie zawsze jest to możliwe.

Rysunek 4: Problemy z rozpoznaniem twarzy [20].

1.6. Komercyjne systemy rozpoznawania twarzy

1.6.1. System firmy Bosch

System firmy *Bosch* został stworzony z myślą o wyeliminowaniu konieczności używania biletów w celu uzyskania dostępu do miejsc publicznych, jakimi są między innymi pływalnie, zoo czy różnego rodzaju kluby. Metoda bazuje na wykrywaniu rysów twarzy. Jak podaje producent, wśród ponad 6 miliardów ludzi nie ma 2 identycznych twarzy [15] co sprawia, że ludzka twarz może być użyta jako uniwersalny klucz dostępu.

Obraz twarzy rejestrowany jest przy pomocy elastycznego modelu siatkowego. Analizie poddawane jest 2200 charakterystycznych cech. „Obliczeniom podlegają np. wysokość i szerokość ust oraz nosa, czubek nosa, pozycja i grubość brwi, kształt podbródka oraz szerokość twarzy u nasady nosa.” [15]. Zebrane dane są użyte do stworzenia modelu siatkowego, a następnie porównania go z już istniejącymi w bazie danych.

Rysunek 5: Model siatkowy twarzy [15].

System dostępny jest w dwóch wersjach: statycznej i dynamicznej. Wersja statyczna rozpoznaje twarze za pomocą kamer umieszczonych w konkretnych miejscach np. przy wejściu. Taki system działa obecnie w ogrodzie zoologicznym w Hanowerze gdzie obsługuje ponad 90,000 ludzi posiadających bilety roczne. System dynamiczny ma możliwość wykrywania i identyfikacji ludzi poruszających się w większych grupach.

1.6.2. MSC/FaceFinder

System rozpoznawania twarzy *MSC/FaceFinder* firmy *GEUTEBRÜCK* jest zaawansowanym systemem monitoringu z funkcją wykrywania i identyfikacji twarzy. Urządzenie to zostało wyposażone w opatentowany system przetwarzania obrazów. Dodatkowo wykorzystywana jest baza danych *SQL* pozwalająca na tworzenie bardzo dużych archiwów sekwencji wideo [16].

Główne cechy systemu to:

- skanowanie zdjęć twarzy
- tworzenie modelu maski twarzy
- automatyczne przechwytywanie i rozpoznawanie twarzy na obrazach z kamer
- obserwacja wyników analizy i obrazów z kamer poprzez sieć (*LAN/WAN*)
- wyświetlenie i aktywowanie alarmu w programie w przypadku wykrycia intruza
- do 10.000 twarzy zgromadzonych w bazie danych

Istnieje kilka podstawowych wersji tego systemu:

- System rozpoznawania twarzy MSC/FACE FaceFinder Compact

  Kompaktowy system rozpoznawania twarzy poruszających się osób. Możliwość obsługi do 4 kanałów wideo. Baza danych zawierająca 1,000 twarzy rozszerzalna o 4,000 lub 9,000 twarzy.

- Jednostka systemowa MSC/FACE Engine Station

  Jednostka centralna wyposażona w bazę danych *SQL*. Maksymalnie obsługuje 8 MSC/FACE FaceFinding Station oraz 8 kanałów wideo. Podstawowa baza danych zawiera 1,000 twarzy z możliwością rozszerzenia o kolejne 4,000 lub 9,000 twarzy.

- System przechwytywania twarzy MSC/FACE FaceFinding Station

  Umożliwia przechwytywanie twarzy poruszających się osób oraz porównywanie ich z bazą MSC/FACE Engine Station. Obsługuje do 4 kanałów wideo.

- System rozpoznawania twarzy MSC/FACE View

  System wizualizacji przebiegu detekcji i rozpoznawania twarzy.

Dane techniczne systemu [16]:

- CPU: 2 x Intel XEON co najmniej 3.0 GHz
- RAM: 2 x 1024 MB
- Dysk twardy / System operacyjny: 2 x 80 GB SATA (co najmniej) / Windows XP
- Napędy zewnętrzne: DVD-ROM/CD-RW
- Sieć: 2 x Ethernet, każdy port 1 Gbit/s

Rysunek 6: Aplikacja do wizualizacji [16].

1.6.3. SNAPPY Face Recogniser

Jest to system do ochrony komputera przed niepowołanym dostępem. Wbudowany jest w zwykłą kamerę internetową. Działa w oparciu o analizę geometrii twarzy. System po odejściu użytkownika do komputera uruchamia wygaszacz ekranu. Kiedy użytkownik wraca kamera automatycznie rozpoznaje twarz i wyłącza wygaszacz [17].

System działa dość szybko i jak większość tego typu systemów jest łatwy do oszukania przez zwykłe zdjęcie. Kamera z którą dostarczany jest system rozpoznawania twarzy umożliwia rejestrację wideo w rozdzielczości 640x480 pikseli z prędkością 30 klatek na sekundę oraz wykonywanie zdjęć w rozdzielczości 1280x1024 piksele.

Rysunek 7: Kamera dostarczana wraz z oprogramowaniem [18].

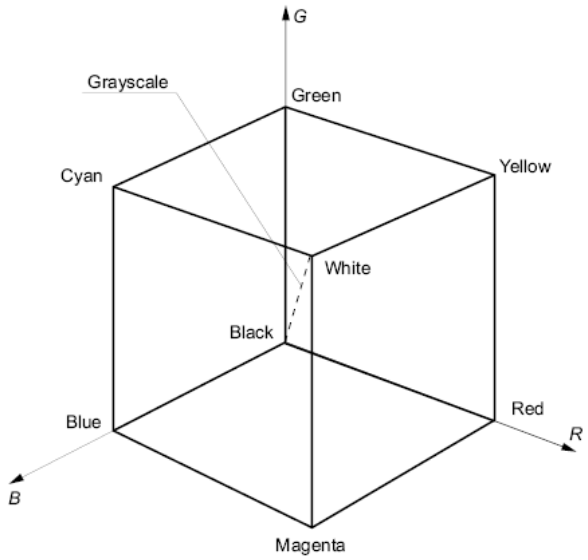
Całość jest bardzo mobilna dzięki niewielkiej wadze i zastosowaniu portu USB można korzystać z tego rozwiązania niemal na dowolnym komputerze.

2. Podstawy teoretyczne

2.1. Modele barw

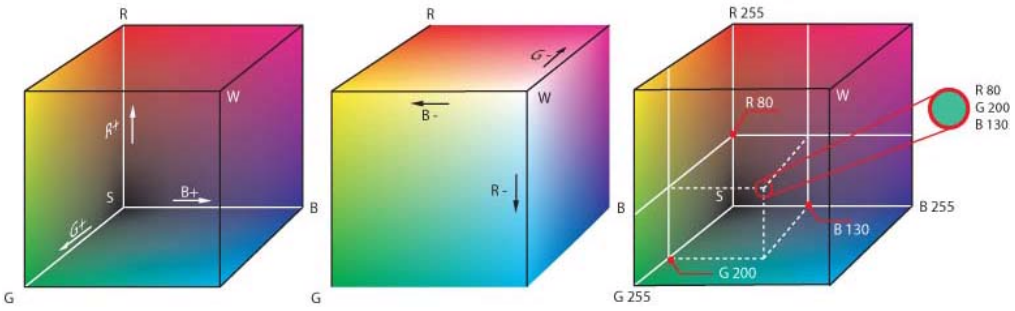
2.1.1 Model RGB

Jest to jeden z podstawowych systemów barw. Opisywany jest za pomocą trzech współrzędnych R (red), G (green) i B (blue). System stosowany jest głównie w grafice komputerowej np. do zapisu palety barw, obecnie monitory wykorzystują właśnie ten model barw do wyświetlania obrazu.
Model ten jest reprezentowany przez sześcian umieszczony na osiach *RGB*. Początek układu współrzędnych ([0,0,0]) odpowiada barwie czarnej, natomiast wektor [1,1,1], białej [1][2][3].



Rysunek 8: Sześcian reprezentujący model RGB [3].

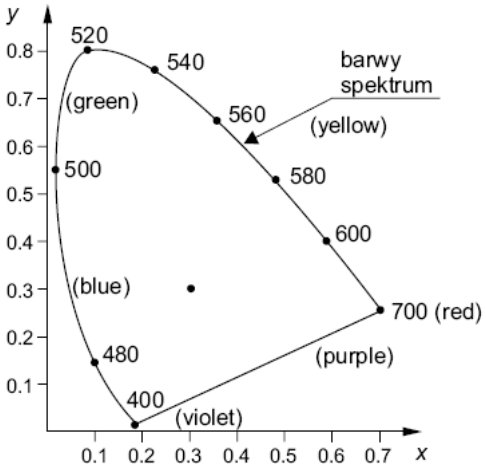
Barwy powstają poprzez mieszanie składowych R, G i B. Każda ze składowych zawiera się w przedziale [0,1], ale używa się również skali [0,255].



Rysunek 9: Rozkład kolorów w sześcianie [4].

2.1.2 Model CIEXYZ

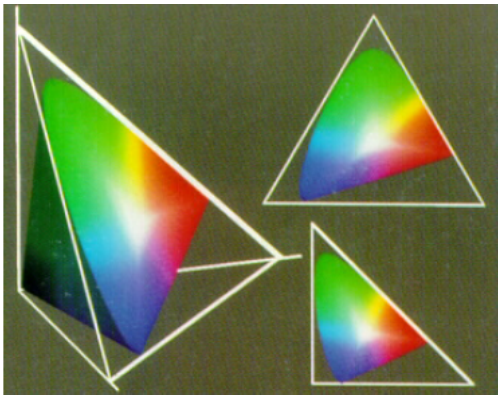
Przestrzeń ta została stworzona w 1931 przez Międzynarodową Komisję Oświetleniową (*Comission Internationale de l'Eclairage*). Jest to specjalna paleta barw zbudowana w oparciu o model widzenia ludzkiego oka. Światło o dowolnej barwie daje się przedstawić za pomocą trzech strumieni X, Y i Z. Składowa Y określa luminację natomiast X i Z określają barwę [1][2][3].
Do jednoznacznego określenia barw stosuje się w współrzędne xy:



Rysunek 10: Diagram chromatyczności CIE [3].

Dla barw widzialnych współczynniki x, y są większe równe zero w odróżnieniu od wartości *RGB,* gdzie nawet dla barw widzialnych występują wartości ujemne, co powoduje pewne zaokrąglenia w grafice cyfrowej.

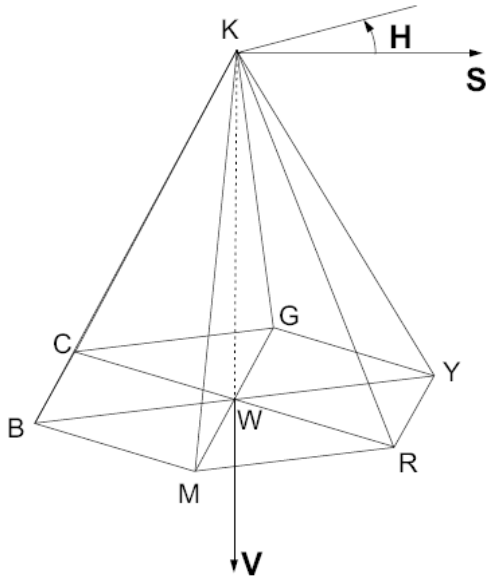
Model *CIEXYZ* jest trójwymiarowy.



Rysunek 11: CIE w układzie współrzędnych XYZ [2].

2.1.3 Model HSV

Model opisu przestrzeni barw zaproponowany w 1978 roku przez Alveya Raya Smitha. Barwy postrzegane są jako światło pochodzące z oświetlenia. Według tego modelu wszelkie barwy wywodzą się ze światła białego, gdzie część widma zostaje wchłonięta a część odbita od oświetlanych przedmiotów. Parametry barwy w modelu *HSV* oznaczają Hue (spektrum), Saturation (nasycenie) oraz Value (wartość). Reprezentacja trójwymiarowa modelu *HSV* możne być przedstawiona za pomocą ostrosłupa foremnego o podstawie sześciokąta. Wierzchołki podstawy symbolizują barwy spektralne. Środek podstawy oznacza barwę białą. Poruszając się zatem po podstawie, od krawędzi do środka sześciokąta otrzymuje się tonalne przejście od czystej barwy spektralnej do bieli. Wysokość ostrosłupa określa ilość czerni dodanej do barwy, tak aby otrzymać ostateczna barwę wynikową [1][2][3].



Rysunek 12: Reprezentacja trójwymiarowa modelu HSV [3].

2.2. Filtry

Filtry obrazu mogą być używane do wyszukiwania interesujących nas obiektów, rozmycia, wyostrzenia obrazu, wykrywania krawędzi czy ogólnej poprawy jakości obrazu. Do tych zastosowań należy również usuwanie szumu. Ze względu na charakter pracy zostały opisane dwa podstawowe filtry jakie zostały w niej zastosowane, a są to filtry medianowy oraz bilateralny nazywany również selektywnym rozmyciem Gaussa.

2.2.1 Filtr medianowy

Filtr ten jest wykorzystywany do usuwania zakłóceń losowych, których poziom intensywności znacznie odbiega od poziomu intensywności punktów sąsiednich. Z pikseli znajdujących się w masce tworzony jest wektor, a następnie jest on sortowany. Jako nowa wartość obrazu zostaje wpisany element środkowy posortowanego wektora. Operacja ta jest powtarzana dla wszystkich punktów obrazu z wyjątkiem n/2 linii będących krawędziami obrazu, gdzie n jest wymiarem maski [23][33].

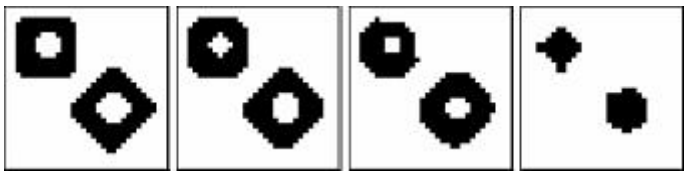
```
Przykład:
Punkty maski:

9 3 1
4 7 2
5 8 6

Wektor maski: 9 3 1 4 7 2 5 8 6

Wektor posortowany: 1 2 3 4 [5] 6 7 8 9

Nowa wartość:

9 3 1
4 5 2
5 8 6
```

Właściwości maski:

- Nie wprowadza do obrazu nowych wartości.
- Dobra do usuwania zakłóceń typu „sól i pieprz”.

Wady:

- Dla dużych masek wzrasta liczba elementów do sortowania.
- „Nadgryzanie” narożników zwłaszcza dla dużych rozmiarów maski.



Rysunek 13: Zniekształcenia rogów obiektów dla masek kolejno 3x3, 5x5, 7x7, 9x9 [33].

W celu przyśpieszenia obliczeń stosuje się medianę obliczaną z 4 sąsiadów analizowanego piksela (zamiast 8).

Można również stosować poniższy wzór, który nie wymaga sortowania elementów:

MED(b,d,e,f,h) = MAX[MIN(b,d,e), MIN(b,d,f), MIN(b,d,h), MIN(b,e,f), MIN(b,e,h), MIN(b,f,h), MIN(d,e,f), MIN(d,e,h), MIN(d,f,h), MIN(e,d,h)]

Rysunek 14: Efekt działania filtru medianowego dla masek kolejno 3x3, 5x5, 7x7, 9x9 [33].

2.2.2 Filtr bilateralny

Jest wiele filtrów do usuwania szumów z obrazu, ale większość z nich powoduje rozmycie krawędzi obrazu bądź też całego obrazu. Jednym z prostych nieiteracyjnych filtrów, które minimalizują ten niepożądany efekt jest filtr bilateralny, który powoduje wygładzenie szumu, przy jednoczesnym zachowaniu wyraźniejszych krawędzi [5].

Opis matematyczny filtru:

Rozważmy przesunięcie niezmienników filtrów dolnoprzepustowych stosowanych do obrazów:

$$h(x) = k_d^{-1} \int_{\infty}^{\infty} \int_{\infty}^{\infty} f(\xi) c(\xi - x) d\xi$$

W celu zachowania składowej stałej, należy:

$$k_d = \int_{\infty}^{\infty} \int_{\infty}^{\infty} c(\xi)\, d\xi$$

Zakres filtrowania jest również zdefiniowany:

$$h(x) = k_r^{-1}(x) \int_{\infty}^{\infty} \int_{\infty}^{\infty} f(\xi)\, s(f(\xi) - f(x))\, d\xi$$

W tym przypadku jądra środków fotometrycznych są podobne dla pikseli, normalizacja stała:

$$k_\Gamma(x) = \int_{\infty}^{\infty} \int_{\infty}^{\infty} s(f(\xi) - f(x))\, d\xi$$

Przestrzenny rozkład intensywności obrazu nie odgrywa żadnej roli w zakresie filtrowania. Rozkład wartości obrazu z dala od x nie powinien mieć wpływu na ostateczną wartość w punkcie x. Następnym krokiem jest połączenie dziedziny i zakresu filtrowania, a tym samym egzekwowanie zarówno geometrycznych jak i fotometrycznych cech. Łączenie można opisać w następujący sposób:

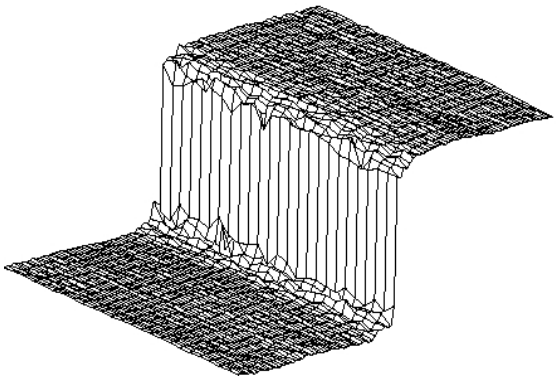
$$h(x) = k^{-1} \int_{\infty}^{\infty} \int_{\infty}^{\infty} f(\xi)\, c(\xi - x)\, s(f(\xi) - f(x))\, d\xi$$

Z normalizacją:

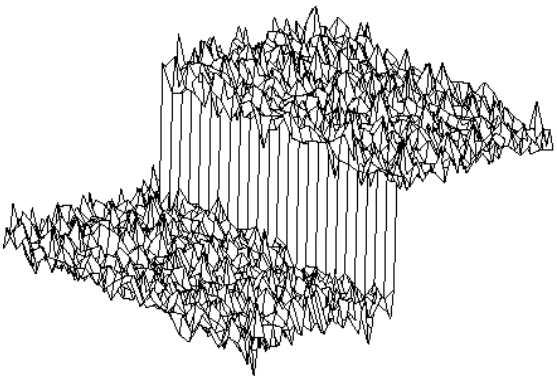
$$k(x) = \int_{\infty}^{\infty} \int_{\infty}^{\infty} c(\xi - x)\, s(f(\xi) - f(x))\, d\xi$$

Właśnie łączenie filtrów jest nazywane filtrem bilateralnym czyli filtrem „dwustronnym”.

Działanie filtru przedstawiają poniższe obrazy:



Rysunek 15: Obraz zaszumiony [5].



Rysunek 16: Obraz po filtracji [5].

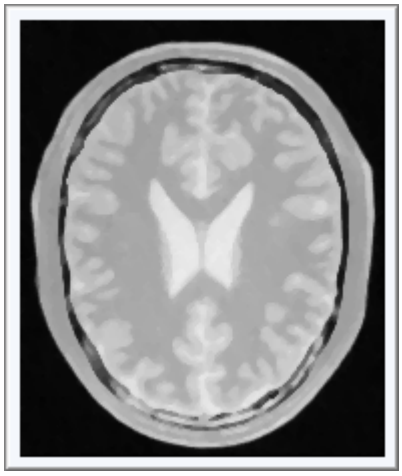
Rysunek 17: Obraz oryginalny [5].

Rysunek 18: Obraz po zastosowaniu filtru bilateralny. Na przykładzie wąsów widać że krawędzie zastały zachowane [5].

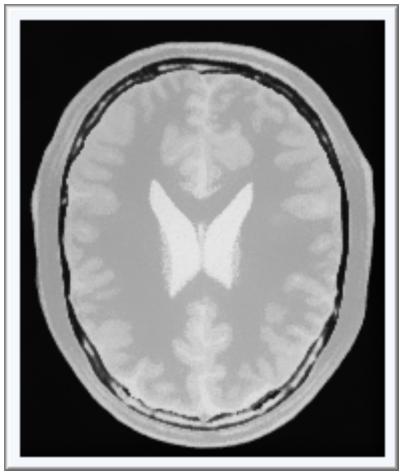
2.2.3 Zestawienie

Poniższe obrazy przedstawiają różnice pomiędzy opisanymi filtrami oraz filtrem uśredniającym:

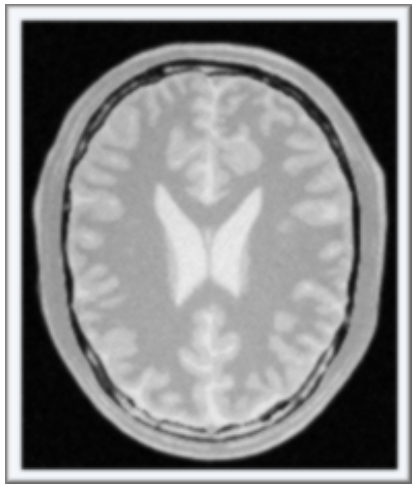
Rysunek 19: Obraz oryginalny.



Rysunek 20: Filtr medianowy.



Rysunek 21: Filtr bilateralny.



Rysunek 22: Filtr uśredniający.

Filtr medianowy dobrze usuwa szum typu „sól i pieprz” oraz podobnie jak filtr bilateralny nie rozmywa krawędzi. Różnią się tym, że filtr bilateralny rozmywa tekstury co powoduje utratę drobnych szczegółów obrazu ale jednocześnie sprawia iż krawędzie stają się bardziej wyraźne. Najgorsze wyniki uzyskuje filtr uśredniający ponieważ rozmywa krawędzie i tekstury.

2.3. Miary odległości

Miary odległości stosuje się do określenia podobieństwa pomiędzy wektorami (odległości pomiędzy nimi). Im bardziej wektory są do siebie podobne tym wartość ta jest mniejsza.

2.3.1 Odległość euklidesowa

Jest jedną z najprostszych metod wyznaczania odległości między wektorami. Opisuje ją wzór:

$$\|x-y\|=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$$

gdzie x i y są wektorami.

Metoda ta jest szczególnym przypadkiem odległości Mahalanobisa.

2.3.2 Odległości Mahalanobisa

Odległość Mahalanobisa stosuje się najczęściej w analizie skupień. Mając dany zbiór punktów tworzących pewną klasę, możemy wyznaczyć dla niego wektor średni $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]$ oraz macierz kowariancji C, które odzwierciedlają pewien charakter tej klasy. Badając przynależność nieznanego wektora losowego $\mathbf{x}$ do danej klasy, mierzy się jego podobieństwo do wektora $\boldsymbol{\mu}$, uwzględniając przy tym informację o wariancjach poszczególnych składowych oraz korelacjach między nimi [6].

2.3.3 Metoda n-najbliższych sąsiadów

Metoda ta polega na obliczeniu odległości wektora szukanego do wszystkich wektorów bazy danych, a następnie wybraniu n najbliższych wektorów, gdzie n jest dowolnie zdefiniowane. W celu dopasowania wektora szukanego do jednej z klas należy sprawdzić, której klasy wektorów jest najwięcej w zbiorze n najbliższych wektorów.

2.4. Biblioteka OpenCV

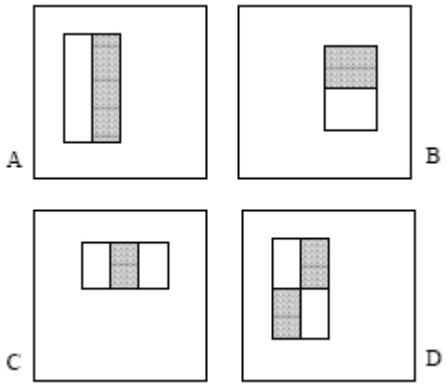
OpenCV jest biblioteką o otwartych źródłach zawierającą wiele funkcji wykorzystywanych podczas przetwarzania obrazów w czasie rzeczywistym, stworzoną przez firmę *Intel*. Wersja alfa została udostępniona publicznie w *IEEE Conference on Computer Vision and Pattern Recognition* w 2000 roku. Wersja 1.0 została wydana dopiero w 2006 roku po serii 5 wersji beta (2001-2005). Biblioteka uzyskała wsparcie ze strony firmy *Willow Garage,* co zaowocowało wydaniem kolejnej wersji 1.1 w połowie 2008 roku [25]. Wieloplatformowość sprawia, że chętnie sięgają po nią użytkownicy systemów *Windows*, *Linux*, ale również *Mac OS*. Wzrost popularności doprowadził do wydania książki „*Learning OpenCV Computer Vision with the OpenCV Library*" autorstwa Gary'ego Bradskiego i Adriana Kaehlera, zawierającej liczne porady oraz przedstawiającej techniki przetwarzania obrazów z wykorzystaniem *OpenCV.* Istnieje również wiele strony *WWW* oraz artykułów poświęconych tej bibliotece [24].

2.5. Haar-like

Haar-like jest metodą wykrywania obiektów w obrazie, zaproponowaną przez Paula Viola i Michaela Jonesa. Celem ich pracy było stworzenie skutecznego systemu wykrywania twarzy, który miał zapewniać dużą szybkość działania.

Pomysł polega na wykorzystaniu trzech prostych funkcji, zamiast operować na pojedynczych pikselach. Zwiększa to wydajność metody. W pierwszej kolejności obliczana jest wartość dwóch funkcji prostokąta, będąca różnicą między sumą pikseli dwóch prostokątnych regionów (o identycznych rozmiarach i kształtach ustawionych

poziomo lub pionowo). Następnie od sumy pikseli centralnego prostokąta odejmowana jest suma pikseli dwóch prostokątów znajdujących się na zewnątrz. Ostatnia funkcja oblicza różnicę między diagonalnymi parami prostokątów (Rysunek 10) [7][8][31].



Rysunek 23: Różnice między jasnymi i ciemnymi prostokątami. A i B przedstawiają pierwszą funkcję natomiast C i D drugą i trzecią [7].

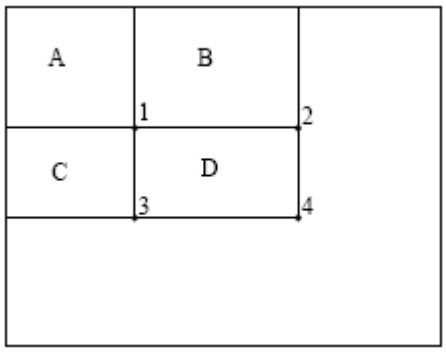
Funkcje dla prostokątów można bardzo szybko obliczyć za pomocą pośrednich reprezentacji obrazu. Obraz w miejscu x, y zawiera sumy pikseli powyżej i po lewej x, y:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

gdzie ii(x,y) jest obrazem pośrednim, a i(x,y) oryginalnym.

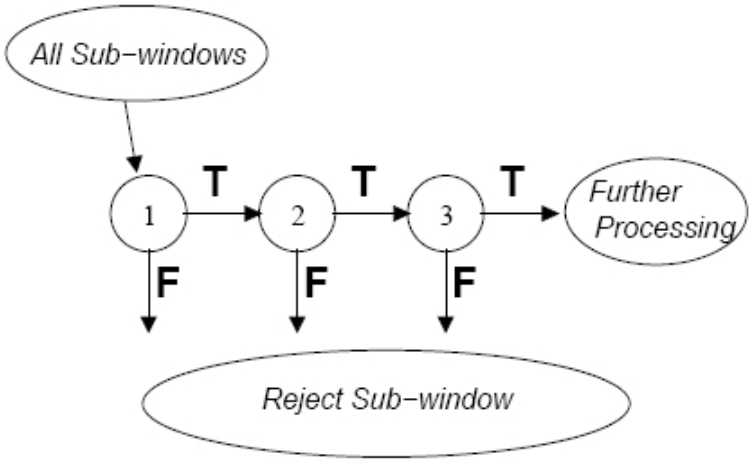
Obliczeń można szybko dokonać wykorzystując wzory:

$$\begin{aligned} s(x,y) &= s(x,y-1) + i(x,y) \\ ii(x,y) &= ii(x-1,y) + s(x,y) \end{aligned}$$



Rysunek 24: Suma pikseli prostokąta D może być obliczona za pomocą czterech tablic odniesień. Wartość obrazu w miejscu 1 jest jest sumą pikseli prostokąta A. Wartość w 2 jest równa A+B w punkcie 3 A+C, a w 4 A+B+C+D. Suma D może być obliczona jako 4+1 – (2+3) [7].

Funkcja do uczenia klasyfikatora kaskadowego wymaga zgromadzenia obrazów pozytywnych i negatywnych. Ponieważ stosowane funkcje dają bardzo dużo kombinacji (180,000), został zastosowany boosting w celu przyśpieszenia znajdowania tych istotnych i szybszego zmniejszania błędu uczenia. Tak zbudowany klasyfikator umożliwia szybkie wykrywanie obiektów, ponieważ decyzje podejmowane są najpierw na najniższych szczeblach, co prowadzi do wstępnego odrzucenia bardzo dużej liczby danych.



Rysunek 25: Proces detekcji z użyciem klasyfikatora kaskadowego. Wstępne odrzucanie dużej ilości negatywnych danych [7].

2.5.1. Tworzenie klasyfikatora kaskadowego

2.5.1.1. Narzędzia

Do zbudowania kaskady umożliwiającej rozpoznawanie obiektów (np. twarzy) można wykorzystać narzędzia dostarczone wraz z *OpenCV* [9][10]:

- Generowanie pliku ze spakowanymi próbkami:

```
createsamples.exe
[-info <description_file_name>]
[-img <image_file_name>]
[-vec <vec_file_name>]
[-bg <background_file_name>]
[-num <number_of_samples = 1000>]
[-bgcolor <background_color = 0>]
[-inv] [-randinv] [-bgthresh
<background_color_threshold = 80>]
[-maxidev <max_intensity_deviation = 40>]
[-maxxangle <max_x_rotation_angle = 1.100000>]
[-maxyangle <max_y_rotation_angle = 1.100000>]
[-maxzangle <max_z_rotation_angle = 0.500000>]
[-show [<scale = 4.000000>]]
[-w <sample_width = 24>]
[-h <sample_height = 24>]
```

- Tworzenie kaskady umożliwiającej wykrywanie obiektów:

```
haartraining.exe
-data <dir_name>
-vec <vec_file_name>
-bg <background_file_name>
[-npos <number_of_positive_samples = 2000>]
[-nneg <number_of_negative_samples = 2000>]
[-nstages <number_of_stages = 14>]
[-nsplits <number_of_splits = 1>]
[-mem <memory_in_MB = 200>]
[-sym (default)] [-nonsym]
[-minhitrate <min_hit_rate = 0.995000>]
[-maxfalsealarm <max_false_alarm_rate = 0.500000>]
[-weighttrimming <weight_trimming = 0.950000>]
[-eqw]
[-mode <BASIC (default) | CORE | ALL>]
[-w <sample_width = 24>]
```

```
[-h <sample_height = 24>]
[-bt <DAB | RAB | LB | GAB (default)>]
[-err <misclass (default) | gini | entropy>]
[-maxtreesplits <max_number_of_splits_in_tree_cascade =
0>]
[-minpos <min_number_of_positive_samples_per_cluster =
500>]
```

- Testowanie kaskady:

```
performance.exe
-data <classifier_directory_name>
-info <collection_file_name>
[-maxSizeDiff <max_size_difference = 1.500000>]
[-maxPosDiff <max_position_difference = 0.300000>]
[-sf <scale_factor = 1.200000>]
[-ni]
[-nos <number_of_stages = -1>]
[-rs <roc_size = 40>]
[-w <sample_width = 24>]
[-h <sample_height = 24>]
```

- Konwersja kaskady do pliku XML:

```
convert_cascade.exe
--size="<sample_width>x<sampe_height>"
<haartraining_ouput_dir>
<ouput_file>
```

- Tworzenie listy próbek:
  - Object Maker

    Program umożliwiający ręczne zaznaczanie twarzy w obrazach oraz automatycznie generujący plik z zaznaczonymi obszarami. [11]

2.5.1.2. Przygotowanie danych

Dane muszą zostać odpowiednio przygotowane przed stworzeniem pliku *.xml zawierającego kaskadę. Dzięki funkcjom dostarczonym przez *OpenCV* każdy może stworzyć własną kaskadę dla dowolnych obiektów [9][10]:

a) Próbki negatywne

Próbki negatywne są obrazami zawierającymi tło utrudniające lub uniemożliwiające poprawne wykrycie obiektu (np. twarzy). Można je wygenerować za pomocą dowolnych programów graficznych (*GIMP*, *Photoshop*), bądź też wykorzystać istniejące bazy danych. Wiele istniejących baz danych zawiera obrazy w formatach skompresowanych np. *.jpg. Zaleca się stosowanie pików o jak najniższym poziomie kompresji (najlepiej *.bmp). Istotne jest, żeby próbek było możliwie dużo (4000-10000).

Jeżeli posiadamy już bazę próbek należny wygenerować ich listę w pliku tekstowym (np. *.txt). Dla systemu *Windows* listę można wygenerować z linii poleceń:

```
dir /B > train-non-face.txt
```

Otrzymany plik train-non-face.txt będzie zawierał listę próbek (plików):

```
Składnia:
[nazwa pliku]
[nazwa pliku]
...
Przykład:
face02134.pgm
face02137.pgm
...
```

b) Próbki pozytywne

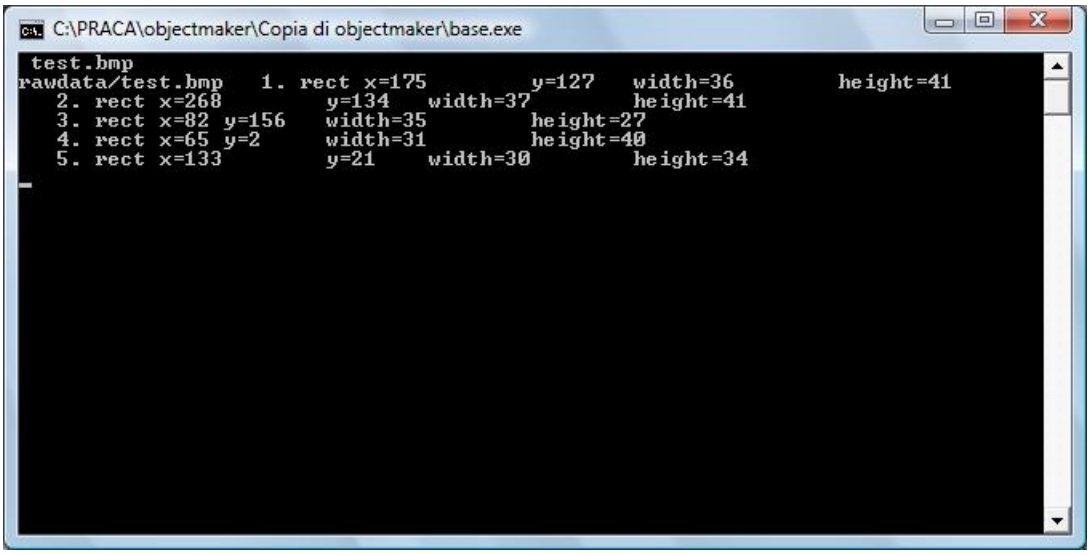
Próbki pozytywne są obrazami zawierającymi obiekty, które chcemy wykrywać (np. twarze). Próbki powinny zawierać obiekty o rożnym oświetleniu, refleksach oraz kątach, pod jakimi są zwrócone do kamery. Podobnie jak w przypadku próbek negatywnych, możemy sami wygenerować zdjęcia lub skorzystać z gotowej bazy danych (zalecane pliki o małym stopniu kompresji). Liczba próbek wpływa na jakość nauki oraz jej czas.

Po zebraniu próbek należny wygenerować ich listę wraz z danymi o położeniu obiektu w próbce (możliwe jest wskazanie kilku obiektów w jednej próbce):

```
Składnia:
[nazwa pliku] [liczba obiektów] [[x y szerokość wysokość] [... 2 obiekt] ...]
Przykład:
face02134.pgm 1 0 0 19 19
face02135.pgm 2 2 17 30 25 50 60 11 12
face02136.pgm 1 15 12 54 65
face02137.pgm 2 0 0 10 10 15 10 11 25
...
```

Jeżeli pojedyncze próbki zawierają kilka obiektów rozmieszczonych nieregularnie, do wygenerowania pliku zawierającego listę obiektów, można wykorzystać program *Object Maker*. Umożliwia on ręczne wskazanie obiektu na zdjęciu oraz sam generuje plik tekstowy zawierający niezbędne dane.

Rysunek 26: Przykład zaznaczania obiektów w programie Object Maker.



Rysunek 27: Podgląd listy obiektów generowany przez Object Maker.

```
Wygenerowany plik:
rawdata/test.bmp 5 175 127 36 41 268 134 37 41 82 156
35 27 65 2 31 40 133 21 30 34
```

c) Przygotowanie próbek do treningu

Wielkość próbek powinna być dość mała, rzędu 20x20 pikseli (przyspiesza obliczenia). Takie rozmiary są wystarczające dla większości obiektów.

Należy przygotować cztery rodzaje próbek:

- pozytywne do treningu
- negatywne do treningu
- pozytywne do testów
- negatywne do testów

Ważne jest, aby próbki testowe nie pokrywały się z próbkami użytymi do trenowania. Następnie, przy pomocy programu *createsamples.exe* generowany jest plik .vec zawierający "spakowane" próbki. Plik ten jest używany przez program *haartraining.exe* w procesie uczenia.

d) Trening

Trenowanie odbywa się przy pomocy programu hartraining.exe. Większość parametrów ustawionych standardowo przez twórców jak np. minhitrate, maxfalsealarm czy weighttrimming nie wymaga zmiany. Należny jedynie pamiętać, żeby liczba etapów uczenia była powyżej 20. Jeżeli liczba etapów będzie zbyt duża, możemy przerwać uczenie w dowolnym momencie, natomiast jeżeli jest zbyt mała, możemy ponownie rozpocząć trening, a nowe etapy zostaną dodane do już istniejącej kaskady. Zaleca się aby ilość błędów była mniejsza równa 5*10(-6).
Maszyna użyta do uczenia powinna być dość silna. Jednym z parametrów jakie możemy zdefiniować jest ilość pamięci *RAM* jaka może być użyta. Ważne jest aby nie użyć całej dostępnej pamięci, gdyż może to spowodować duże spowolnienie systemu lub w przypadku zużycia całej dostępnej pamięci przerwanie obliczeń zakończone błędem przepełnienia pamięci.

e) Generowanie pliku XML

Wynik w postaci kaskady wygenerowany przez program *hartraining.exe* należy skonwertować do formatu *XML* aby możliwe było jego wykorzystanie w programie. Do tego celu służy program *convert_cascade.exe*. Łączy on katalogi powstałe w wyniku tworzenia każdej z kaskad w jeden plik. Możliwe jest ręczne połączenie tych danych, ale jest to dość uciążliwe.

2.6. Eigenface

EigenFace działa w oparciu o przechwytywanie zmian w kolekcji obrazów twarzy oraz wykorzystuje te informacje do kodowania i rozpoznawania twarzy. Ogólnie *EigenFace* jest zbiorem wektorów własnych macierzy kowariancji zbioru obrazów twarzy, w którym obraz złożony z N pikseli jest traktowany jako punkt (wektor) w N-wymiarowej przestrzeni. Pomysł wykorzystania składowych głównych do

reprezentowania ludzkich twarzy został opracowany przez L. Sirovich i M. Kirby [13] i wykorzystywany przez Turka i Pentlanda [14] do wykrywania i rozpoznawania twarzy. Podejście *EigenFace* jest uważane przez wielu jako pierwszy rozdział w technice komputerowego rozpoznawania twarzy i używany jako podstawa większości systemów komercyjnych [12].

Głównymi cechami *EigenFace* sprawiającymi, że jest to metoda popularna są:

- Wyodrębnienie cech twarzy, które nie mogą być zauważone przez człowieka tak jak np. nos, oczy czy usta. Metoda bazuje na uchwyceniu statystycznych zmian pomiędzy obrazami twarzy.
- Skuteczna reprezentacja obrazów twarzy. Aby zmniejszyć złożoność obliczeń i przestrzeni, obraz każdej twarzy można przedstawić za pomocą niewielkiej liczby parametrów.

EigenFaces mogą być traktowane jako zespół cech, które charakteryzują globalne zmiany wśród obrazów twarzy.

Przed obliczeniem *EigenFaces*, zdjęcia twarzy są znormalizowane do linii oczu i ust, a następnie wszystkie obrazy są skalowane do takiego samego rozmiaru. *EigenFaces* zostają wyodrębnione z danych obrazu za pomocą analizy składowych głównych (*PCA*).

Komputerowe liczenie *EigenFace* [12]:

- M obrazów o wymiarach [h x w] zostaje zapisanych jako wektory D (o wymiarach hw) i umieszczone w zbiorze $\{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\}$. Wszystkie obrazy powinny być odpowiednio przeskalowane, a tło jednorodne lub usunięte.
- Każda twarz różni się od średniej o wektor $\Phi_i = \Gamma_i - \Psi$, gdzie średnia twarz jest określona przez $\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$.

- Macierzy kowariancji $C \in \mathbb{R}^{D \times D}$ jest zdefiniowana jako:

$$C = \frac{1}{M} \sum_{i=1}^{M} \Phi_i \Phi_i{}^T = AA^T$$, gdzie $A = \{\Phi_1, \Phi_2, \ldots, \Phi_M\} \in \mathbb{R}^{D \times M}$.

- Ustalenie wektorów własnych C jest trudne dla obrazów o typowych rozmiarach, gdzie $D \gg M$. Aby efektywnie obliczyć wektory własne C można najpierw obliczyć wektory o wiele mniejsze $M \times M$ macierzy $A^T A$. Wektory i wartości własne $A^T A$ są definiowane jako: $V = \{v_1, v_1, \ldots v_\tau\}$ i $\Lambda = diag\{\lambda_1, \lambda_1, \ldots \lambda_\tau\}, \lambda_1 \geqslant \lambda_2 \geqslant \ldots \lambda_\tau > 0$, gdzie τ jest wymiarem A . Należy pamiętać, że wektory odpowiadające zerowym wartością własnym zostały odrzucone.
- Wartości własne i suma macierzy wektorów własnych C są Λ i

$U = AV\Lambda^{-1/2}$, gdzie $U = \{u_i\}$ jest zbiorem *EigenFaces*.

Rysunek 28: Kilka przykładów twarzy ze zbioru PIE CMU (Sim et al. 2003) [12].

Rysunek 29: Średnie twarzy i EigenFaces [12].

W celu redukcji wymiaru wektora stosuje się analizę składowych głównych (*PCA*). *PCA* stara się znaleźć k „osi głównych", które określają ortogonalny układ współrzędnych mogący przechwycić większość wariancji danych. Na przykład, biorąc pod uwagę macierz danych A, macierz kowariancji może być zapisana jako $C = AA^T$ natomiast główne składniki u_i ,

$$AA^T u_i = \lambda_i u_i$$

gdzie u_i są wektorami własnymi AA^T związanymi z wartością λ_i . Kiedy AA^T jest zbyt duży, aby możliwe było jego skuteczne rozłożenie, można ominąć to przez obliczenie wewnętrznej macierzy iloczynu $A^T A$,

$$A^T A v_i = \lambda_i v_i$$

gdzie v_i są wektorami własnymi $A^T A$ związanymi z wartością λ_i . należy zauważyć, że λ_i jest nieujemna, a $u_i = \lambda_i^{-0.5} A v_i$ dla tych niezerowych wartości własnych.

$$A = U \Delta V^T = \sum \delta_i u_i v_i^T$$

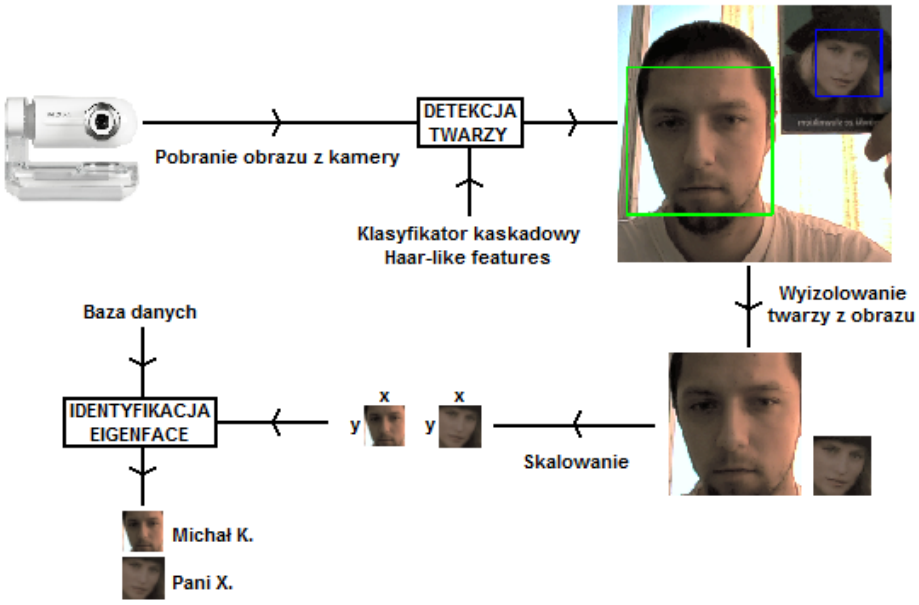
gdzie U i V są prostopadłe, przekątna macierzy Δ zawiera pojedynczą wartość δ_i, która może być pozytywna, zerowa, a nawet ujemna. U i V są wektorami własnymi $AA^T, A^T A, \delta_i^2 = \lambda_i$ dla każdego i.

W celu sprawdzenia czy dana osoba jest tą, za którą się podaje, wystarczy porównać wektor obrazu wejściowego z gotowymi wzorcami umieszczonymi w pamięci. Dzięki *PCA* wektory te nie są duże, co znacznie przyśpiesza proces porównywania. Metoda ta jest zbliżona do tej wykorzystywanej w kryminalistyce, bazującej na portretach pamięciowych. *EigenFace* jest niekiedy nazywane *EigenImage,* ponieważ technika ta jest również stosowana do rozpoznawania tekstu pisanego, mowy oraz symboli języka migowego.

3. Realizacja

3.1. Proponowane rozwiązanie

Do przechwytywania obrazu użyto zwykłej kamery internetowej *Creative Live! Cam Optia,* gdyż posiada ona wystarczającą rozdzielczość aby możliwa była detekcja oraz rozpoznanie twarzy. Oczywiście cena urządzenia nie pozostaje bez znaczenia i waha się w granicach 150-250 zł (dane z 01.04.2009). Kolejnym krokiem jest wykrycie twarzy w obrazie. W tym celu wykorzystano metodę *Haar-like* zapewniającą dużą skuteczność wykrywania obiektów w obrazie oraz szybkość działania. W przeciwieństwie do metod bazujących na kolorze ludzkiej skóry, dobrze radzi sobie ze zmianą oświetlenia. Oczywiście metoda detekcji skóry może wspierać metodę *Haar-like* w eliminowaniu błędnie zakwalifikowanych obiektów będących częściami tła. Wykryte twarze są następnie wycinane z obrazu i skalowane do rozmiaru obrazów zapisanych w bazie danych. Skalowanie jest niezbędne gdyż metoda *EigenFace*, użyta do identyfikacji twarzy, wymaga danych o takich samych rozmiarach jak te, zgromadzone w bazie danych (utworzonej w procesie uczenia przy pomocy *PCA*). Zdjęcia są kolejno analizowane przez algorytm *EigenFace* i porównywane z danymi znajdującymi się w bazie danych. Cały proces przedstawia poniższy schemat:



Rysunek 30: Poglądowy schemat działania programu.

3.2. Problemy

Wiele problemów jakie pojawiają się w trakcie tworzenia systemu rozpoznawania i detekcji twarzy związanych jest z wydajnością przetwarzania grafiki, a więc szybkością komputera. Duży wpływ mają również warunki w jakich aplikacja jest testowana. Ogólnie problemy można podzielić na trzy grupy:

3.2.1 Wydajność

Zarówno wydajność metod jak i maszyny ma duży wpływ na sprawność systemu. Ze względu na działanie systemu w czasie rzeczywistym konieczne było stworzenie metod, które nie obciążą zbytnio komputera. Przetwarzanie obrazu samo w sobie nie należy do najszybszych. Spory wpływ ma wielkość i złożoność przetwarzanego obrazu. W celu przyśpieszenia obliczeń należy dostosować parametry aplikacji do konkretnych warunków tak, aby wyeliminować obszary, których przetwarzanie nie jest potrzebne. Wszystko sprowadza się do ograniczenia ilości danych jakie muszą być przetwarzane.

3.2.2 Baza danych

Duże znaczenie ma rozmiar bazy danych. W calu zapewnienia poprawnego rozpoznawania obrazów należy zapisać 10-20 obrazów dla jednej twarzy. Głównym problemem jest konieczność przeszukiwania bazy w czasie rzeczywistym. W aplikacji rozwiązano to poprzez wstępne załadowanie bazy do pamięci i zastosowanie *PCA,* co powoduje zmniejszenie rozmiaru wektorów obrazów, a w efekcie znacznie przyspiesza wydajność aplikacji.

3.2.3 Warunki oświetleniowe

W celu uniknięcia błędów we wstępnej fazie projektowania systemu konieczne było zapewnienie stałych warunków oświetleniowych. Jest to bardzo istotne ponieważ metody stosowane w tego typu aplikacjach są bardzo wrażliwe na wszelkie zmiany, głównie oświetlenia. Dlatego wszystkie testy jakie zostały przeprowadzone w trakcie pisania metod i dobierania współczynników zostały przeprowadzone w pomieszczeniu z jednym stałym źródłem światła.

3.3. Narzędzia

Do pracy nad aplikacją został wybrany system operacyjny firmy *Microsoft*. Głównym powodem tej decyzji był bezpłatny dostęp do kompilatora *Microsoft Visual Studio 2008*. Program ten w znaczny sposób przyśpiesza tworzenie aplikacji okienkowych z graficznym interfejsem użytkownika (*GUI*). Dodatkowo daje możliwość pisania aplikacji w językach C/C++, co jest ważne ponieważ biblioteka *OpenCV* została napisana w C. Istnieją wersje np. dla C#, ale wydaje się że wersja oryginalna powinna być szybsza oraz zawierać mniej błędów. *OpenCV* jest prawdopodobnie najlepszą biblioteką o otwartych źródłach i główną biblioteką, jaka została użyta. Dostarcza wiele użytecznych funkcji umożliwiających przetwarzanie obrazu jak chociażby tworzenie klasyfikatora funkcji *Haara-like*, wsparcie dla metody *EigenFace*. Do przechwytywania obrazu użyto kamery *Creative Live! Cam Optia* ze względu na dobry stosunek jakości do ceny oraz fakt, że dobrze współpracuje z biblioteką *OpenCV*.

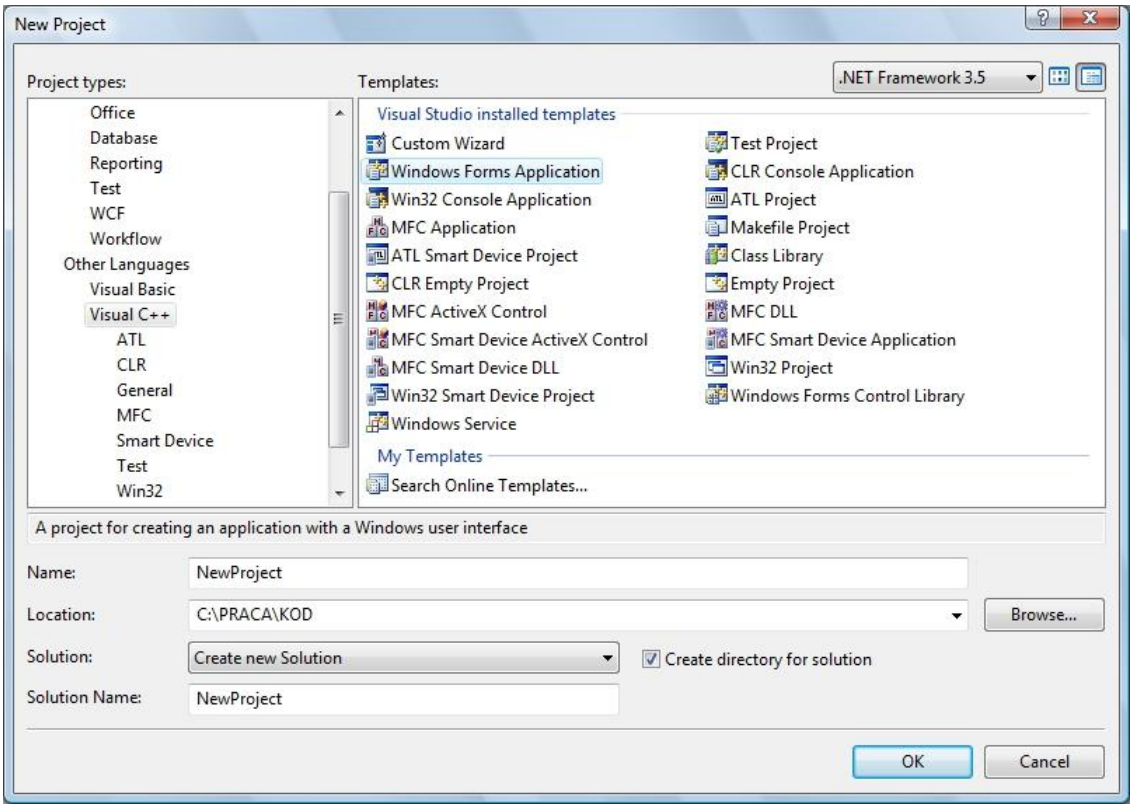
3.3.1. Konfiguracja kompilatora

Przed przystąpieniem do pracy z kompilatorem i biblioteką *OpenCV* należy skonfigurować środowisko tak, aby nie występowały problemy z kompilacją oraz działaniem aplikacji [29][30]:

a) Wybór i stworzenie nowego projektu

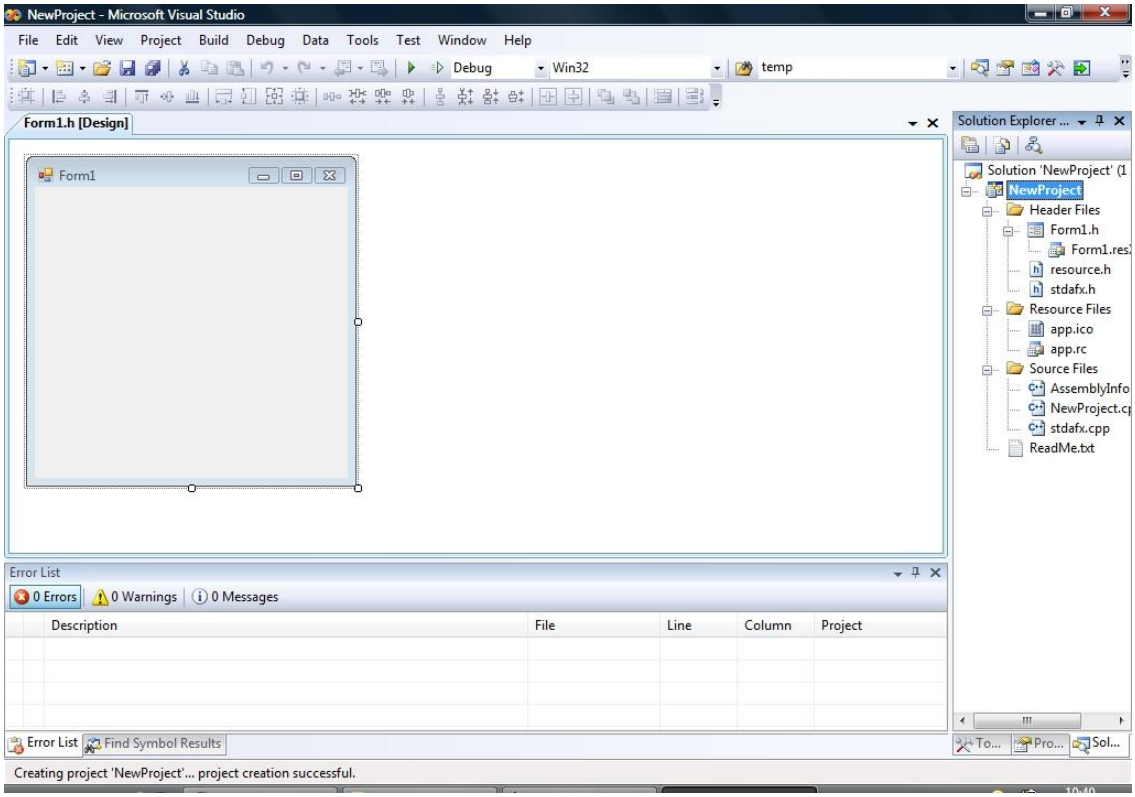
Typ projektu wybrany do stworzenia programu to *Windows Forms Application*. Jest to projekt umożliwiający tworzenie aplikacji okienkowej. Zapewnia szeroki zestaw narzędzi znacznie przyśpieszających i ułatwiających pracę oraz późniejszą obsługę przez użytkownika. Projekt tworzony jest w języku C++.

W celu stworzenia nowego projektu należy wybrać *File* → *New* → *Project...* lub użyć skrótu klawiszowego *Ctrl+Shift+N*.



Rysunek 31: Okno wyboru typu projektu.

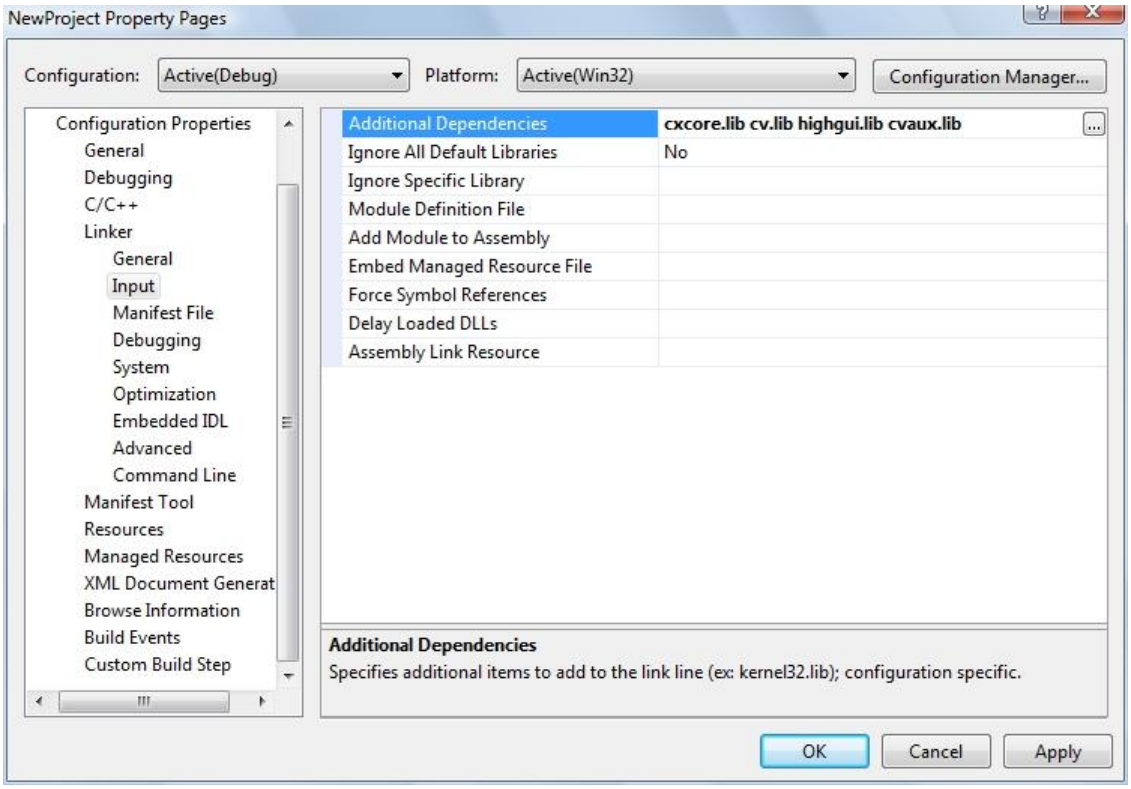
Po wyborze typu projektu oraz nadaniu mu nazwy kompilator stworzy wszystkie niezbędne pliki projektu wraz z domyślną formą.



Rysunek 32: Nowy projekt wygenerowany przez Visual Studio 2008.

b) Linkowanie

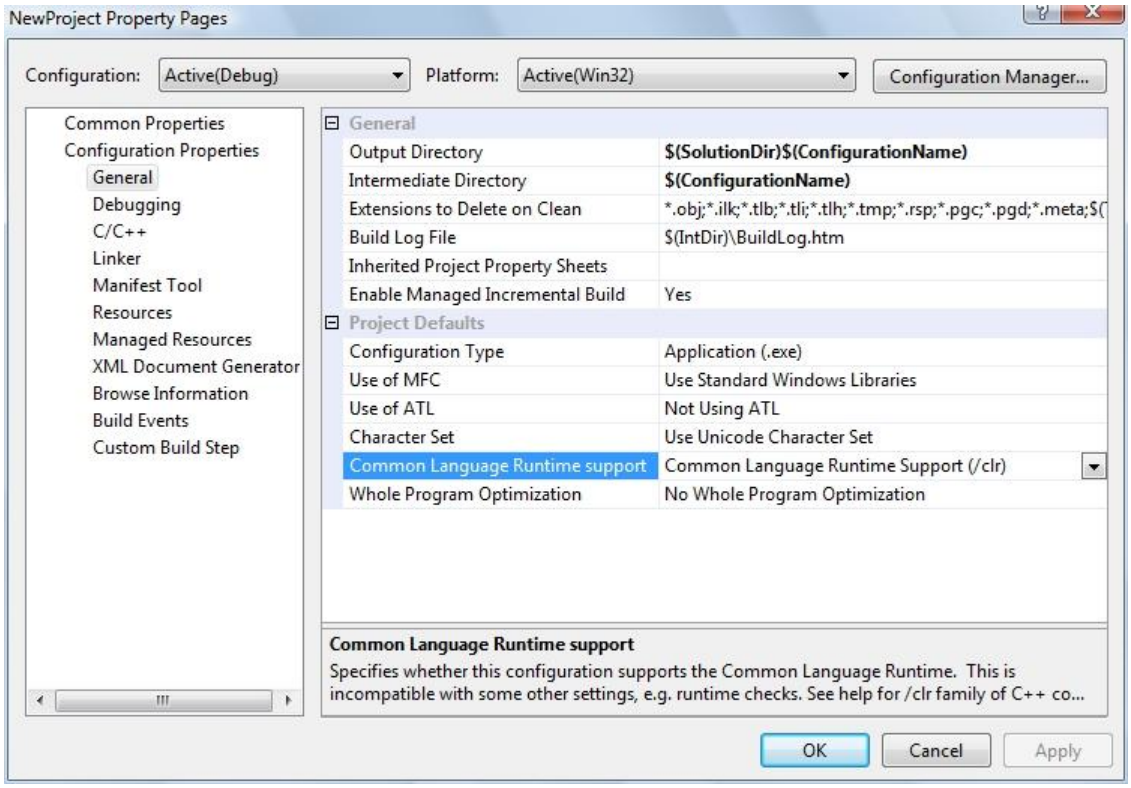
Kolejnym krokiem jest z linkowanie odpowiednich plików. Należy przejść do opcji projektu *Project* → *Properties* i odszukać opcję *Configuration Properties* → *Linker* → *Input*. W linii *Additional Dependencies* wystarczy dodać następujące pliki *cxcore.lib, cv.lib, highgui.lib* oraz *cvaux.lib*.



Rysunek 33: Opcje linkera.

c) Opcje kompilacji

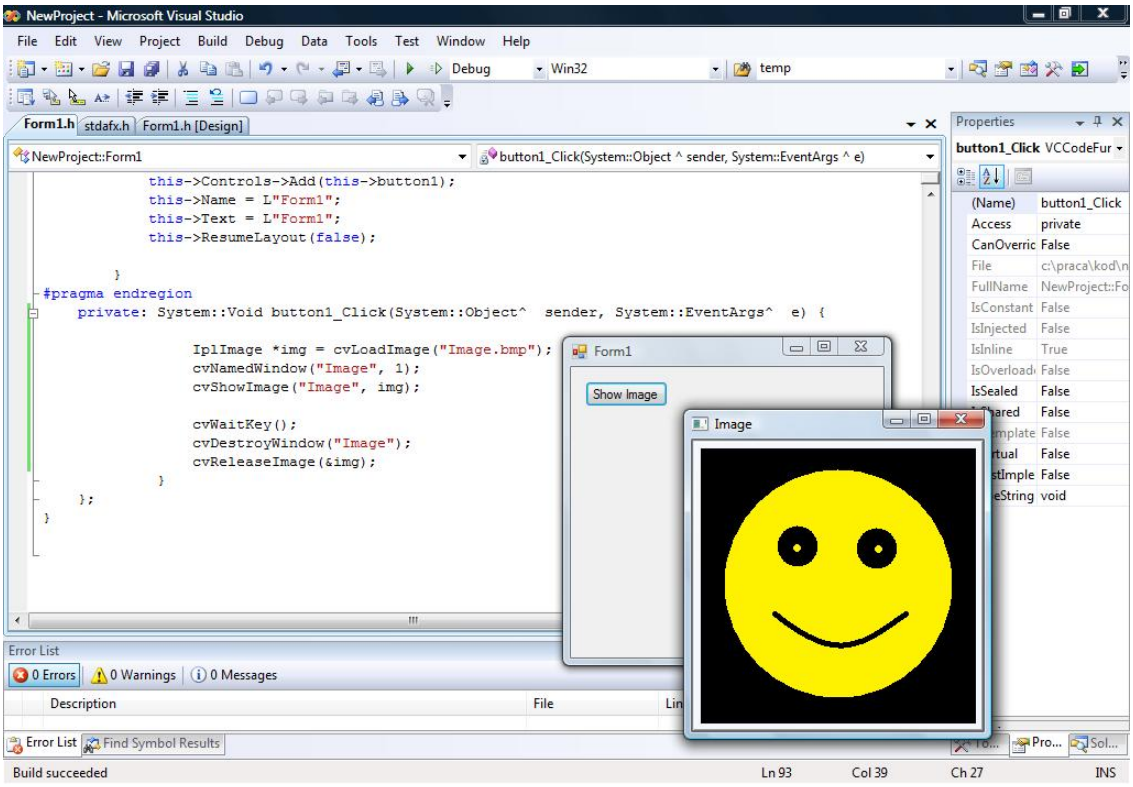
Po dodaniu odpowiednich bibliotek do projektu trzeba jeszcze zmienić opcje kompilacji. W zakładce *Configuration Properties* → *General* należy odszukać opcję *Common Language Runtime support* i zmienić standardowo ustawione „*Pure MSIL Common Language Runtime Support (/clr:pure)*" na „*Common Language Runtime Support (/clr)*". W przeciwnym wypadku kompilator zgłaszać będzie błędy związane z niezgodnością projektu z bibliotekami zewnętrznymi (.dll).



Rysunek 34: Opcje kompilacji.

d) Pierwszy program

W tak skonfigurowanym środowisku możemy napisać pierwszy program z wykorzystaniem biblioteki *OpenCV*.



Rysunek 35: Program wykorzystujący bibliotekę OpenCV po skompilowaniu i uruchomieniu.

3.3.2. Tworzenie klasyfikatora kaskadowego

Proces tworzenia kaskady jest bardzo czasochłonny. Duże znaczenie ma zatem szybkość maszyny na jakiej będzie ten proces przeprowadzony oraz jej bezawaryjność. Czas uczenia może sięgnąć nawet kilku tygodni. Co prawda można przerwać uczenie w dowolnym momencie i rozpocząć później na tej samej bądź innej maszynie. W przypadku przerwania operacji w trakcie trwania danego etapu będzie konieczność ponownego obliczenia tego etapu. Cały proces tworzenia kaskady został podzielony na podetapy w celu częściowego zautomatyzowania.

a) Struktura katalogów

Wszystkie niezbędne dane została umieszczone w odpowiednich katalogach tak, aby możliwe było zautomatyzowanie części pracy i usystematyzowanie danych. Główna zaletą takiego rozwiązania jest możliwość uruchomienia obliczeń, bądź ich kontynuowania na dowolnej maszynie bez konieczności ustalania ścieżek do odpowiednich plików za każdym razem.

Struktura katalogów wygląda następująco:

```
├───code
│   ├───Face List Edit
│   └───Object Maker
├───data
│   ├───cascade
│   ├───cascadeXML
│   ├───faces
│   │   ├───test
│   │   │   ├───face
│   │   │   └───non-face
│   │   └───train
│   │       ├───face
│   │       └───non-face
│   └───samples
├───package
│   └───database
└───programs
    ├───Cascade To XML
    ├───Create Cascade
    ├───Create Samples
    ├───Face List Edit
    ├───Faces List
    └───OpenCV
        └───bin
```

Opis katalogów:

- Code: zawiera kody źródłowe programów.
- Data: przechowuje bazę danych oraz wszystkie wygenerowane pliki używane w procesie uczenia oraz testowania.
- Package: zawiera wszystkie pakiety wykorzystane w pracy.
- Programs: programy w formie wykonywalnej.

b) Baza danych (próbek)

Dokładne dane bazy oraz informacje o jej twórcach [21][22]:

CBCL Face Database #1
MIT Center For Biological and Computation Learning
http://www.ai.mit.edu/projects/cbcl

Zawartość bazy:

Obrazy w skali szarości 19x19 w formacie PGM
Dane do nauki: 2429 twarze / 4548 nie-twarze
Dane testowe: 472 twarze / 23573 nie-twarze

Baza danych zawierająca próbki znajduje się w katalogu PRACA\package\database. Jest to forma spakowana. Próbki przygotowane do użycia znajdują się w PRACA\data\faces posegregowane w odpowiednich podkatalogach.

Ogromna zaletą tej bazy jest duża liczba próbek o jednakowych rozmiarach (19x19). Znacznie przyspiesza to tworzenie danych do nauki.

c) Wstępne przygotowanie próbek

Próbki pozytywne oraz negatywne znajdują się odpowiednio w katalogach PRACA\data\faces\train\face oraz PRACA\data\faces\train\non-face. Z próbek należy wygenerować odpowiednie pliki tekstowe, zawierające wylistowane nazwy próbek oraz, w przypadku próbek pozytywnych, dane o położeniu obiektów. W tym celu został przygotowany specjalny plik *Faces List.bat*:

```
dir /B ..\..\data\faces\test\face >
..\..\data\faces\test\face\test-face.txt
dir /B ..\..\data\faces\test\non-face >
..\..\data\faces\test\non-face\test-non-face.txt
dir /B ..\..\data\faces\train\face >
..\..\data\faces\train\face\train-face.txt
dir /B ..\..\data\faces\train\non-face >
..\..\data\faces\train\non-face\train-non-face.txt

"..\Face List Edit\FaceListEdit.exe"
..\..\data\faces\train\face\train-face.txt " 1 0 0 19
19" ..\..\data\faces\train\face\train-face-info.txt
```

Jak widać korzysta on również z programu FaceListEdit.exe, którego zadaniem jest dodanie danych o położeniu obiektów dla próbek pozytywnych. Wykorzystano tu jedną z zalet bazy próbek - identyczny rozmiar obrazów (19x19 pikseli). Umożliwiło to w prosty sposób zautomatyzowanie tego procesu. Jedyną informacją, jaka jest dodawana do każdej linii pliku z listą próbek jest „*1 0 0 19 19*”. Informuje ona, że na każdej próbce znajduje się jeden obiekt o współrzędnych [0, 0] i rozmiarze 19x19.

Pliki zawierające listy próbek (*train-face.txt* i *train-face-info.txt*) generowane są w katalogach z próbkami.

d) Przygotowanie próbek do nauki

W tym celu został stworzony odpowiedni plik *Create Samples.bat*:

```
..\..\programs\OpenCV\bin\createsamples.exe
-info ..\..\data\faces\train\face\train-face-info.txt
-vec ..\..\data\samples\sample-train-face.vec -num 2429
-w 19 -h 19
```

Jego zadaniem jest wygenerowanie pliku *.vec na podstawie wcześniej przygotowanych list próbek.

e) Tworzenie kaskady

Po wygenerowaniu list oraz pliku *.vec możemy przystąpić do tworzenia kaskady umożliwiającej wykrywanie twarzy. Podobnie jak poprzednie operacje i ta została zapisana w odpowiednim pliku *Create Cascade.bat* dostosowanym do istniejącej bazy danych i struktury katalogów:

```
..\..\programs\OpenCV\bin\haartraining.exe
-data ..\..\data\cascade -vec
..\..\data\samples\sample-train-face.vec -bg
..\..\data\faces\train\non-face\train-non-face.txt
-npos 2429 -nneg 4548 -nstages 30 -mem 1000 -mode ALL
-w 19 -h 19
```

Od doboru parametrów, mocy maszyny i wielkości bazy zależy czas jaki zajmie trening. Przykładowy wydruk z procesu uczenia dziewiątego z trzydziestu etapów wygląda następująco:

```
c:\PRACA\PRACA\programs\Create
Cascade>..\..\programs\OpenCV\bin\haartraining.exe
-data ..\..\data\cascade -vec
..\..\data\samples\sample-train-face.vec -bg ..\
..\data\faces\train\non-face\train-non-face.txt -npos
2429 -nneg 4548 -nstages 3
```

```
0 -mem 1000 -mode ALL -w 19 -h 19
Data dir name: ..\..\data\cascade
Vec file name: ..\..\data\samples\sample-train-face.vec
BG  file name: ..\..\data\faces\train\non-face\train-
non-face.txt
Num pos: 2429
Num neg: 4548
Num stages: 30
Num splits: 1 (stump as weak classifier)
Mem: 1000 MB
Symmetric: TRUE
Min hit rate: 0.995000
Max false alarm rate: 0.500000
Weight trimming: 0.950000
Equal weights: FALSE
Mode: ALL
Width: 19
Height: 19
Applied boosting algorithm: GAB
Error (valid only for Discrete and Real AdaBoost):
misclass
Max number of splits in tree cascade: 0
Min number of positive samples per cluster: 500
Required leaf false alarm rate: 9.31323e-010
Stage 0 loaded
Stage 1 loaded
Stage 2 loaded
Stage 3 loaded
Stage 4 loaded
Stage 5 loaded
Stage 6 loaded
Stage 7 loaded
Stage 8 loaded
Tree Classifier
Stage
+---+---+---+---+---+---+---+---+---+
|  0|  1|  2|  3|  4|  5|  6|  7|  8|
+---+---+---+---+---+---+---+---+---+

   0---1---2---3---4---5---6---7---8

Number of features used : 52126

Parent node: 8

*** 1 cluster ***
POS: 2352 2429 0.968300
NEG: 4403 0.00021988
BACKGROUND PROCESSING TIME: 80854.42
```

```
Number of presorted indexes: 52126
Number of precomputed features: 12744
Precomputing data, that may take a few minutes...
.....................................................
.....................
...................
Precalculation time: 15.02
+----+----+-+---------+---------+---------+---------+
|  N |%SMP|F|  ST.THR |    HR   |    FA   | EXP. ERR|
+----+----+-+---------+---------+---------+---------+
|   1|100%|-| 1.000000| 0.998724| 0.000000| 0.000444|
+----+----+-+---------+---------+---------+---------+
Stage training time: 9.59
Number of used features: 1

Parent node: 8
Chosen number of splits: 0

Total number of splits: 0
```

Parametr *BACKGROUND PROCESSING TIME* podaje czas zakończenia danego etapu w sekundach. Jak widać operacja jest bardzo czasochłonna i w tym przypadku wyniosła ponad 80854 sekund co daje około 22 godziny i 30 minut.

Po zakończeniu danego etapu jest generowany odpowiedni plik *AdaBoostCARTHaarClassifier.txt* w katalogu o nazwie składającej się z numeru danego etapu np. 0 , 1, 2 itd.. Uczenie można przerwać w dowolnym momencie ale zostaną zapisane jedynie etapy już ukończone.

3.4. Opis programu

Program „Detekcja i rozpoznawanie twarzy” umożliwia przechwytywanie obrazu z kamery, wykrywanie oraz identyfikację twarzy. Dostarcza również filtry poprawiające jakość obrazu oraz funkcje wspomagające eliminowanie obrazów, które nie są twarzami. Możliwe jest również ustalenie progu określającego przynależność twarzy do bazy danych oraz samo generowanie bazy twarzy.

Program został podzielony według funkcjonalności na zakładki:

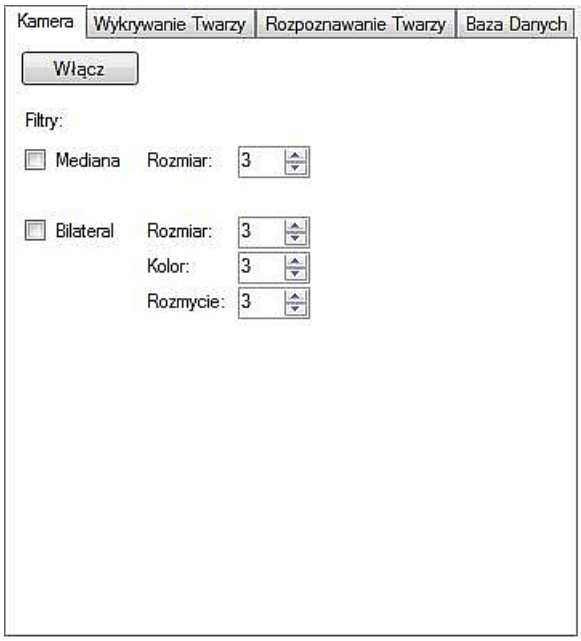
- Kamera
- Wykrywanie Twarzy
- Rozpoznawanie Twarzy
- Baza Danych

3.4.1 Obsługa programu

Obsługa programu jest intuicyjna i prosta dzięki zastosowaniu graficznego interfejsu użytkownika (*GUI*) oraz zautomatyzowaniu części operacji. Całość została podzielona na zakładki tak aby ułatwić szukanie interesujących nas opcji:

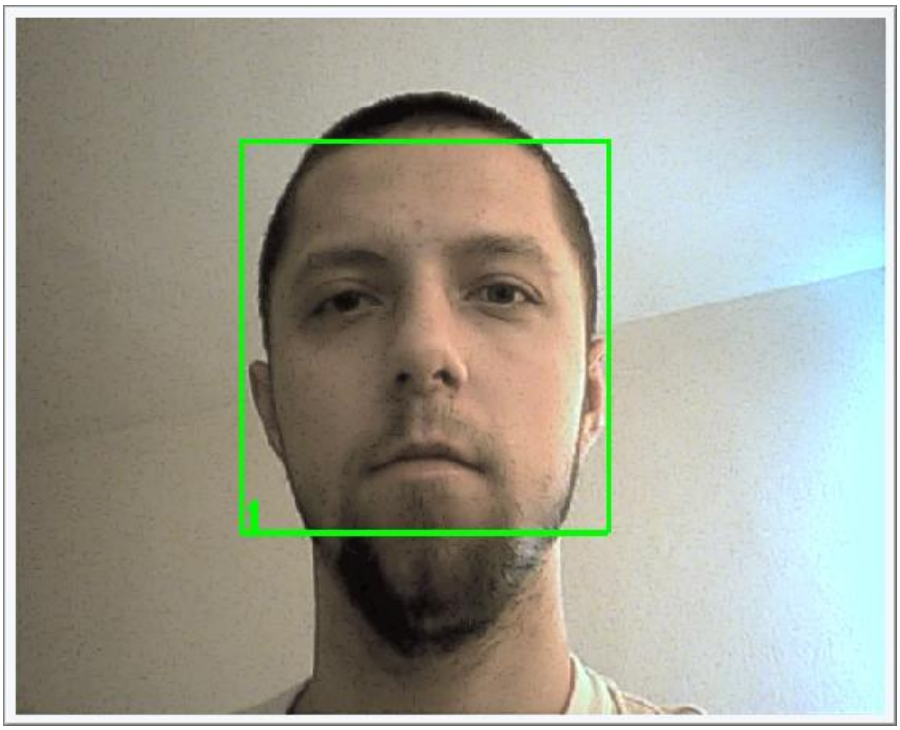
a) Kamera

Zakładka ta umożliwia włączenie oraz wyłączenie kamery, co jest równoznaczne ze startem głównego wątku aplikacji jakim jest detekcja i rozpoznawanie twarzy. Dostarcza również dwa filtry obrazu.



Rysunek 36: Opcje znajdujące się w zakładce Kamera.

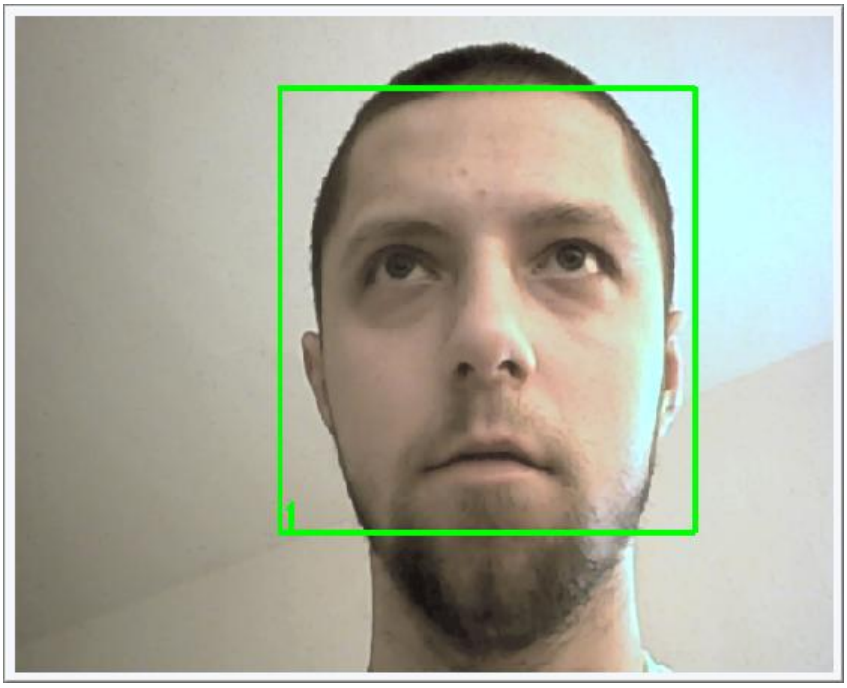
Do włączenia kamery służy przycisk „Włącz”. Po jego naciśnięciu pojawi się okno z wyświetlające obraz z kamery.



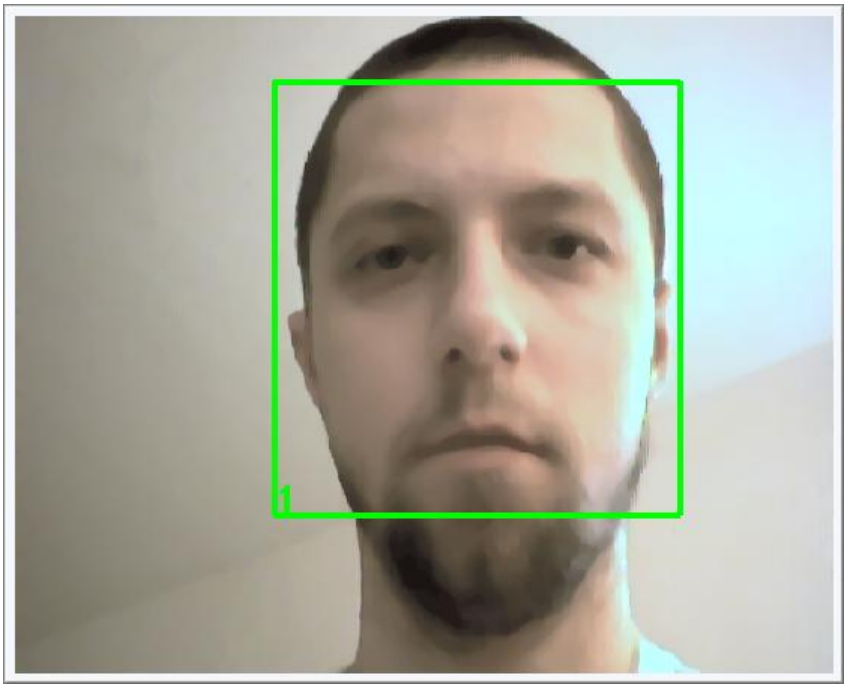
Rysunek 37: Obraz pochodzący z kamery po jej uruchomieniu.

Dostępne są dwa filtry obrazu, które aktywuje się poprzez zaznaczenie odpowiedniego *Check Boxa*:

- Mediana

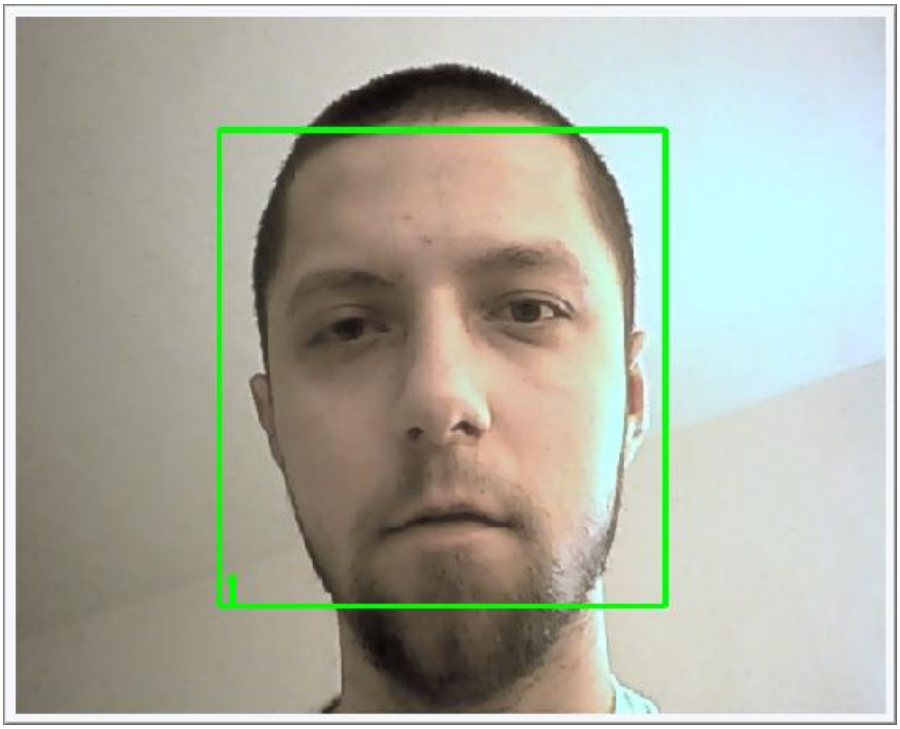


Rysunek 38: Filtr medianowy 3x3.

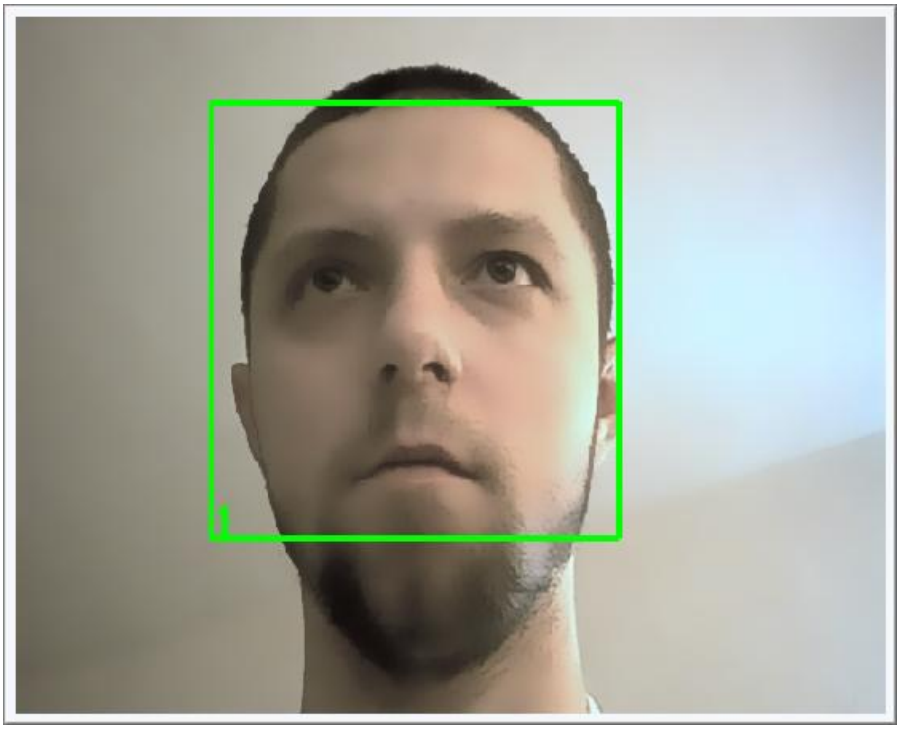


Rysunek 39: Filtr medianowy 9x9.

- Bilateral



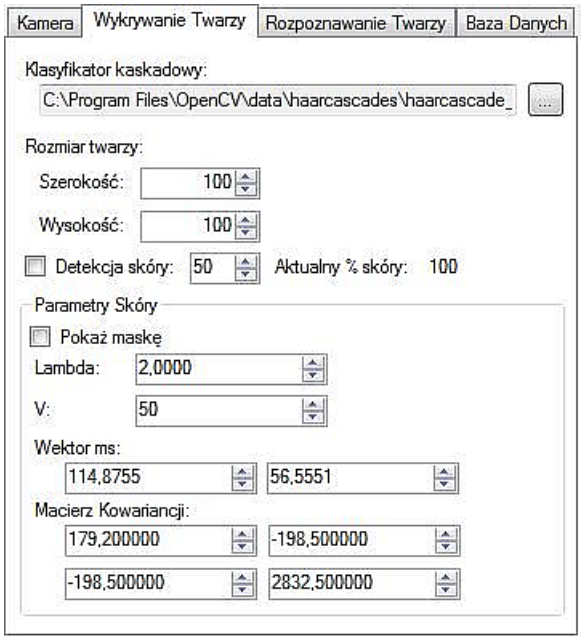
Rysunek 40: Filtr bilateralny 11x11, kolor 100, rozmycie 25.



Rysunek 41: Filtr bilateralny 21x21, kolor 20, rozmycie 50.

b) Wykrywanie Twarzy

Ta część programu udostępnia opcje związane z wykrywaniem twarzy w obrazie.



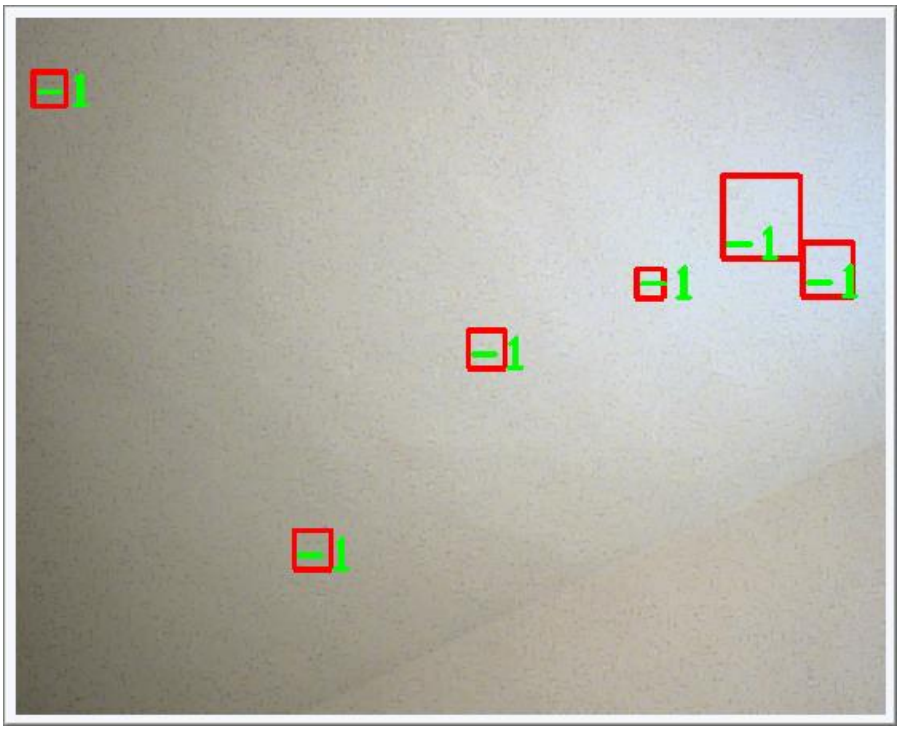
Rysunek 42: Opcje znajdujące się w zakładce Wykrywanie Twarzy

- Klasyfikator kaskadowy

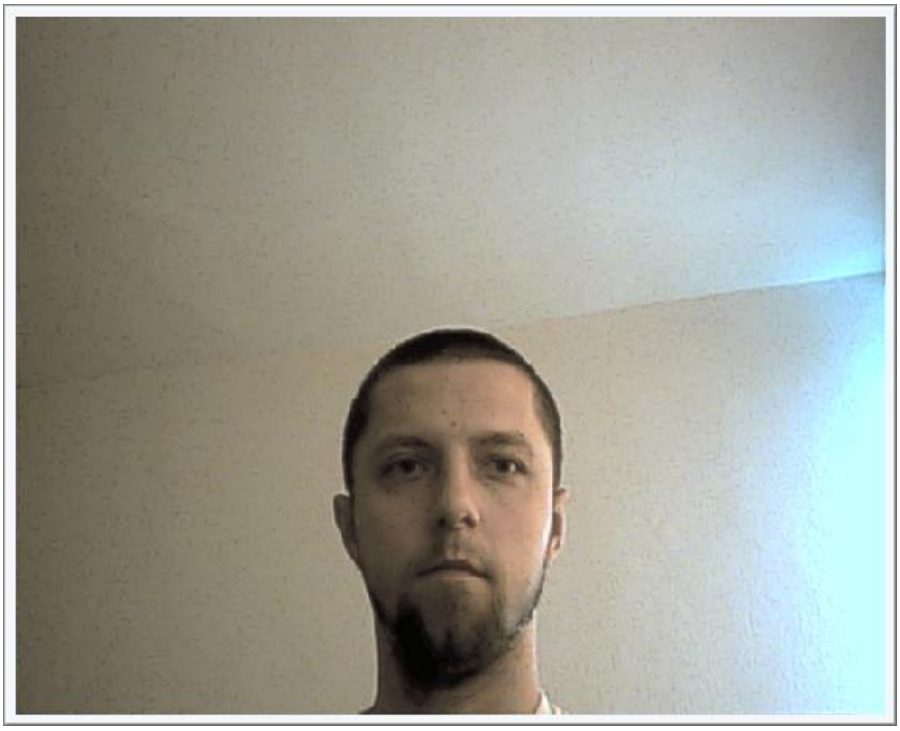
  Opcja umożliwia wczytanie wcześniej przygotowanej kaskady dla funkcji *Haar-like*.

- Rozmiar Twarzy

  Umożliwia określenie minimalnego rozmiaru twarzy jaki będzie wykrywany przez funkcję *Haar-like*. Zbyt mały obszar poszukiwania może spowodować wykrycie jako twarzy elementów tła oraz znacznie zwiększa nakład obliczeń. Duży obszar nie pozwoli natomiast wykryć twarzy znajdujących się w dużej odległości od kamery, ale skutecznie wyeliminuje fałszywe twarze będące elementami tła. Jak większość parametrów musi być ustawiony pod kątem konkretnych warunków w jakich system ma działać.



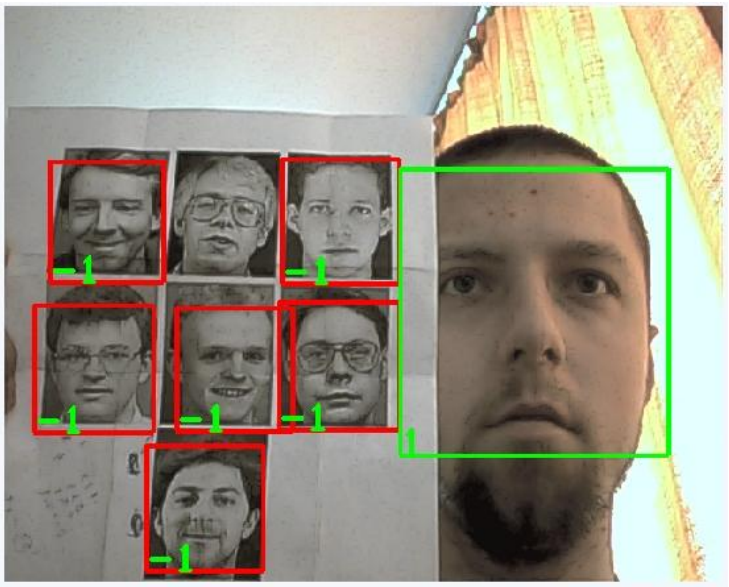
Rysunek 43: Rozmiar twarzy 10x10, dużo elementów tła wykrytych jako twarz.



Rysunek 44: Rozmiar twarzy 250x250, twarz w większej odległości od kamery nie została wykryta.

- Detekcja skóry

Wspiera metodę *Haar-like* w wykrywaniu twarzy poprzez odrzucanie obiektów, których kolor nie odpowiada kolorowi ludzkiej skóry. Została wykorzystana metoda zaproponowana przez Grzegorza Żylińskiego w projekcie „*Rozpoznawanie dłoni*” [26]. Skala jest w % określa, jaki procent ludzkiej skóry znajduje się w obrazie wskazanym przez metodę *Haar-like*. Natomiast „Aktualny % skóry” pokazuje, jaki procent pikseli odpowiada ludzkiej skórze w aktualnie zaznaczonym obiekcie. Informacja służy głównie do poprawnego doboru współczynników.



Rysunek 45: Obraz bez detekcji skóry. Twarze znajdujące się na zdjęciach w odcieniu szarości zostały wykryte.

Rysunek 46: Obraz z detekcja skóry. Twarze znajdujące się na zdjęciach w odcieniu szarości nie zostały wykryte.

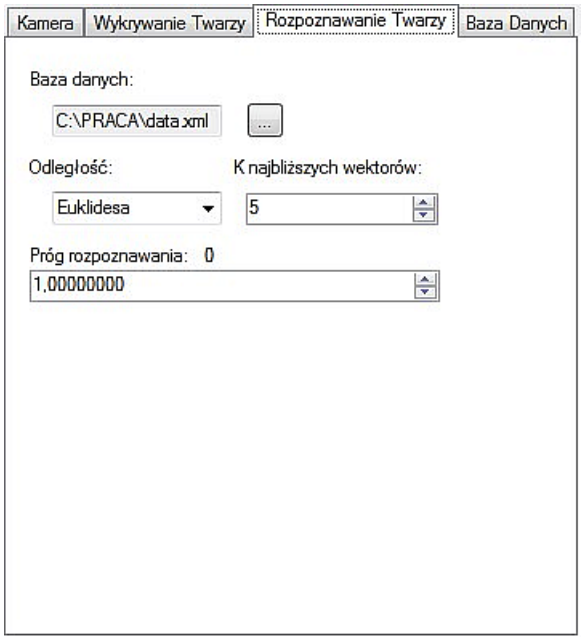
- Parametry skóry

Pole to zawiera opcje umożliwiające dopasowanie współczynników tak, aby w danych warunkach poprawnie identyfikować kolor skóry. Możliwe jest również wyświetlenie maski pokazującej, który obszar jest rozpoznany jako skóra (pola białe). Do tego celu służy opcja „Pokaż maskę”.

Rysunek 47: Maska przedstawiająca obszary wykryte jako ludzka skóra (pola białe).

c) Rozpoznawanie Twarzy

Znajdują się tu opcje odpowiedzialne za rozpoznawanie twarzy wcześniej wykrytych.



Rysunek 48: Opcje znajdujące się w zakładce Rozpoznawanie Twarzy.

- Baza danych

  Umożliwia wskazanie pliku zawierającego bazę twarzy, które maja zostać rozpoznawane przez system.

- Odległość

  Określa sposób porównywania wektorów. Dostępne są trzy podstawowe metody:

  - Euklidesowa
  - Mahalanobisa
  - N-Najbliższych sąsiadów

- K najbliższych wektorów

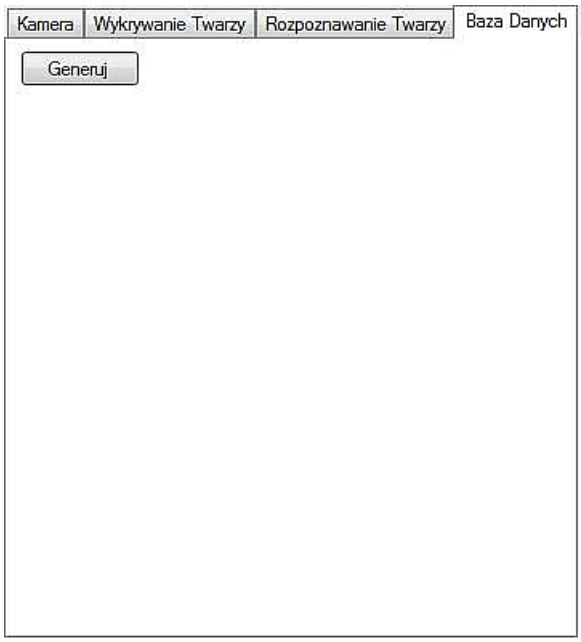
  Umożliwia ustawienie liczby wektorów, na podstawie których metoda n-najbliższych sąsiadów podejmie decyzję o przynależności wektora do jednej z klas.

- Próg rozpoznawania

  Pozwala ustawić próg powyżej jakiego obraz jest uznawany jako tło, jego odległość od obrazów znajdujących się w bazie jest zbyt duża i jest on traktowany jako nowa osoba bądź też element tła w przypadku błędnego wykrycia twarzy w obrazie.

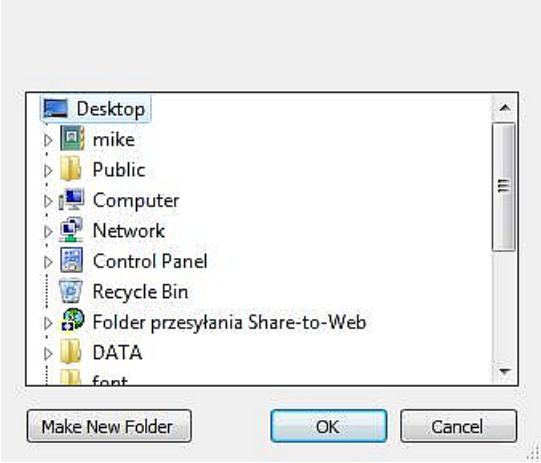
d) Baza Danych

Umożliwia generowanie bazy danych zawierającej twarze.



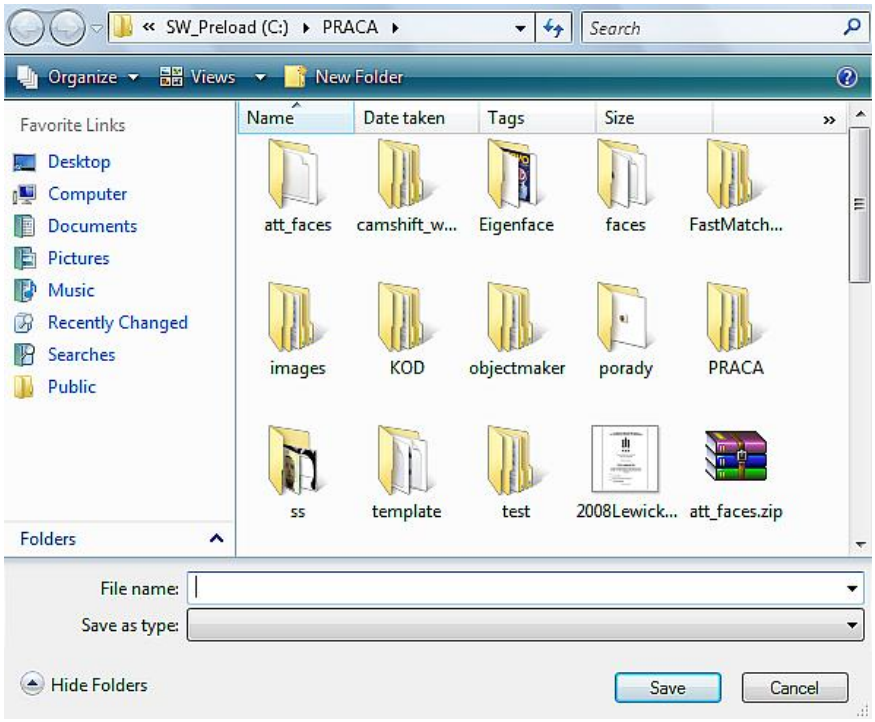
Rysunek 49: Opcje znajdujące się w zakładce Baza Danych.

Po wciśnięciu przycisku „Generuj” zostanie wyświetlone okno umożliwiające wskazanie katalogu zawierającego podkatalogi z twarzami. Każdy podkatalog powinien zawierać twarze jednej osoby, gdyż w procesie tworzenia bazy tak właśnie będą traktowane.



Rysunek 50: Okno wyboru katalogu zawierającego twarze.

Następnie zostanie wyświetlone okno zapisu nowej bazy danych. Należy podać nazwę nowego pliku dla utworzenia nowej bazy lub wskazać już istniejący w celu zastąpienia bazy inną.



Rysunek 51: Wybór pliku z bazą twarzy.

3.4.2 Opis działania programu

a) Kamera

Program umożliwia współpracę z kamerami kompatybilnymi z *OpenCV*. Za obsługę kamery odpowiadają funkcję zawarte w pliku videoCapture.cpp oraz funkcja pomocnicza *captureVideoFrame* sprawdzająca czy obraz nie jest odwrócony. Funkcja *OpenCV* odpowiedzialna za pobieranie danych z kamery to *cvCaptureFromCAM*, zwraca ona strukturę *CvCapture*. Następnym krokiem jest uzyskanie macierzy obrazu IplImage. Pozwala na to kolejna z funkcji *OpenCV*, *cvQueryFrame*. Mając obiekt przechowujący obraz z kamery możliwe jest jego wyświetlenie i dalsza edycja.

b) Wykrywanie twarzy [26]

Funkcje odpowiedzialne za wykrywanie twarzy za pomocą funkcji *Haar-like* znajdują się w pliku *facesDetection.cpp*. Funkcja *detectFaces* wykorzystuje funkcję *OpenCV cvHaarDetectObjects*. Dodatkowo umożliwia zmianę rozmiaru poszukiwania twarzy w czasie rzeczywistym.

Funkcja *Haar-like* jest wspierana przez detekcję skóry w celu wyeliminowania obiektów tła rozpoznanych wstępnie jako twarz. Odpowiada za to funkcja *Detekcja* zaproponowana przez Grzegorza Żylińskiego [26].

c) Rozpoznawanie twarzy [27]

Funkcje odpowiedzialne za rozpoznawanie twarzy znajdują się w klasie *EigenFace*. Za rozpoznawanie twarzy odpowiada funkcja *Rozpoznaj*. Przygotowuje ona wektor wejściowy, obliczając rozkład współczynników *PCA* obrazu wejściowego, a następnie wykonuje funkcję *findCalcNearestNeighbor*. Funkcja ta porównuje wektor wejściowy z wektorami znajdującymi się w bazie danych wykorzystując jedną z trzech podstawowych miar odległości: odległość euklidesową, odległość Mahalanobisa lub metodę n-najbliższych sąsiadów. Wymaga ona jednak wcześniejszego wczytania danych, które zostały zapisane w procesie nauki w pliku *.xml. Do wczytania danych służy funkcja *LoadData* działająca analogicznie jak funkcja odpowiedzialna za zapis danych.

d) Budowanie bazy danych

Obrazy twarzy muszą znajdować się w katalogach, których nazwy określać będą ich numer w bazie danych (mogą to być kolejne liczby). Katalogi te muszą być zgromadzone w jednym katalogu w celu zautomatyzowania całego procesu.

Struktura katalogów powinna wyglądać następująco:

```
└─Twarze
  ├─0
  ├─1
  ├─2
  ...
  ├─100
  ...
```

Numeracja może być dowolna. Należy jednak pamiętać, że obrazy twarzy będą numerowane kolejno od 1. Nazwy katalogów maja jedynie ułatwić przypisanie osoby do danej twarzy.

Każdy katalog powinien zawierać pliki graficzne z twarzami. Wszystkie pliki muszą mieć identyczny rozmiar. Pliki zgromadzone w jednym katalogu są uznawane za obrazy jednej twarzy.

Po wskazaniu przez użytkownika katalogu zawierającego podkatalogi z obrazami twarzy zostaje wygenerowana lista plików, która następnie jest zapisana do pliku pomocniczego listdata. Jest to zwykły plik tekstowy.

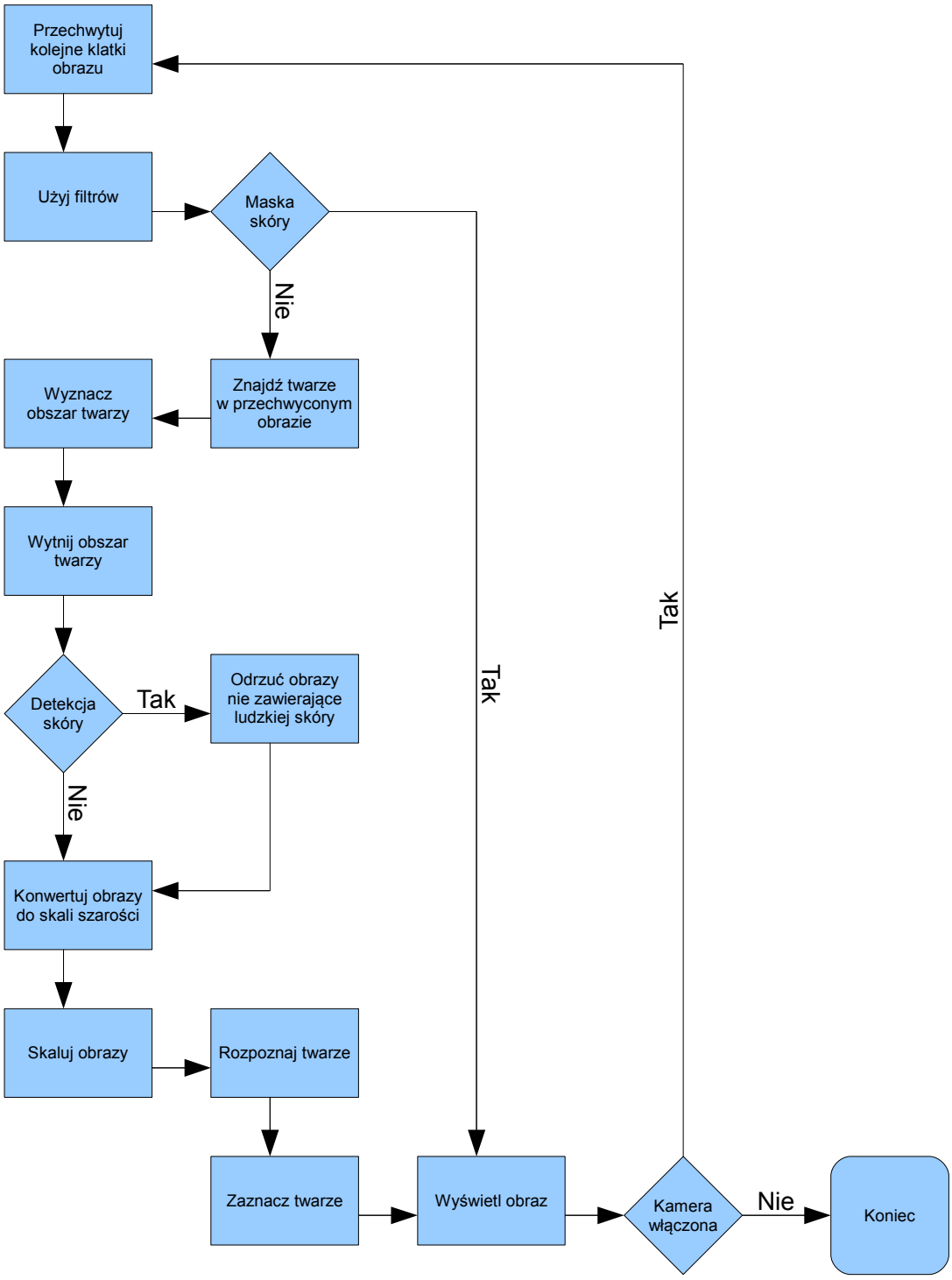
Struktura listdata:

```
Składnia:
[numer katalogu twarzy] [ścieżka do pliku]
Przykład:
1 C:\PRACA\test\s000\Clipboard08.bmp
1 C:\PRACA\test\s000\Clipboard10.bmp
1 C:\PRACA\test\s000\Clipboard12.bmp
2 C:\PRACA\test\s042\Clipboard02.bmp
2 C:\PRACA\test\s042\Clipboard04.bmp
2 C:\PRACA\test\s042\Clipboard06.bmp
```

Lista ta jest użyta w programie do szybkiego wybierania kolejnych plików oraz do wygenerowania wektora umożliwiającego wskazanie numeru twarzy w procesie rozpoznawania.

Dla wszystkich plików szukana jest przestrzeń *PCA*, a następnie dane są rzutowane do tej przestrzeni. Po wygenerowaniu wszystkich danych zostają one zapisane przy pomocy funkcji *SaveData*.

e) Główna pętla programu



Rysunek 52: Schemat blokowy głównej pętli programu.

3.5. Wyniki badań

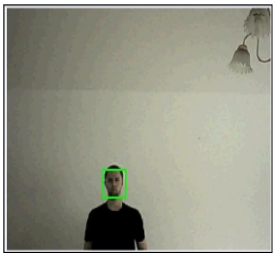
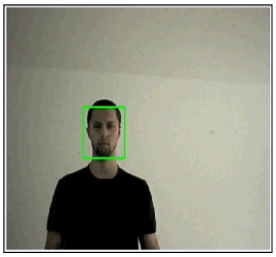
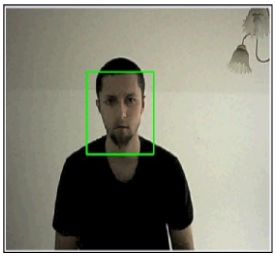
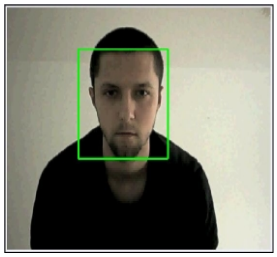
Badania obejmują sprawdzenie poprawności wykrywania twarzy wraz ze zmianą odległości od kamery oraz oświetlenia, poprawność rozpoznania twarzy dla bazy kilku osób oraz zmiany parametrów odpowiedzialnych za poprawną identyfikacje skóry w obrazie.

Wszystkie testy zostały przeprowadzone na komputerze klasy notebook o parametrach:

- Procesor: Intel Core 2 Duo 1.60GHz (niskonapięciowy)
- Pamięć (RAM): 4GB
- System operacyjny: Windows Vista Home Basic

3.5.1 Detekcja twarzy

Testy przedstawione są w filmach *detekcja_50x50.avi*, *detekcja_100x100.avi*, *detekcja_150x150.avi*, *detekcja_200x200.avi*. Pokazują one wpływ wielkości obszaru poszukiwań twarzy na odległość jaka może być pomiędzy twarzą a kamerą, aby możliwa była poprawna detekcja.

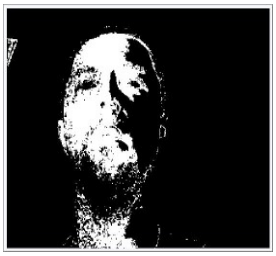
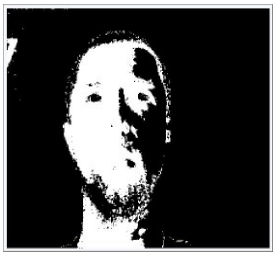
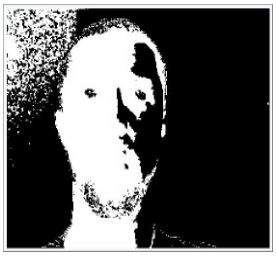
Rozmiar obszaru poszukiwań			
50x50	100x100	150x150	200x200
			

Dla obszaru 50x50 twarz pomimo niskiej jakości obrazu z kamery jest nadal poprawnie wykrywana. Rozmiar obszaru poszukiwań 200x200 i większy może być stosowany w systemie, w którym użytkownik znajduje się blisko kamery. Wyszukiwanie obiektów większych jest znacznie szybsze. Dlatego poprawne ustalenie tego parametru jest takie ważne i ma wpływ na całkowitą wydajność systemu.

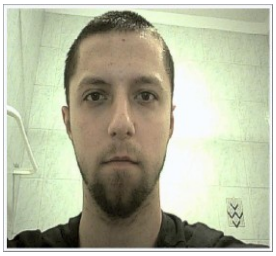
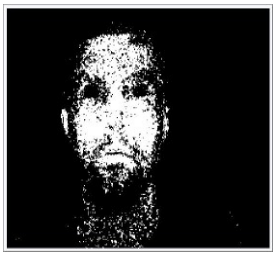
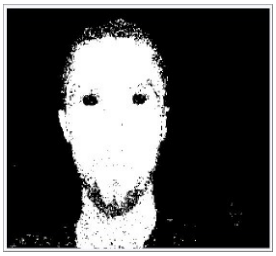
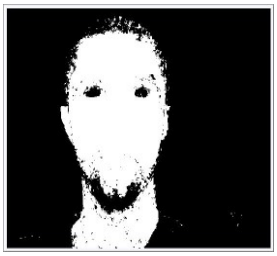
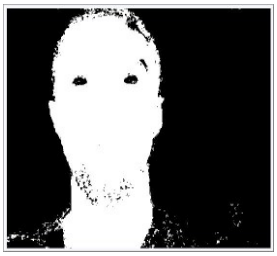
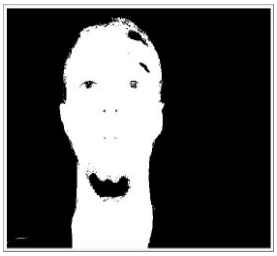
3.5.2 Detekcja skóry

Główne parametry pozwalające określić czy piksel zostanie uznany za skórę to V i Lambda. V jest jedną ze składowych przestrzeni HSV użytej do wykrywania skóry. Umożliwia ustalenie w jakim stopniu pomiar ma być uniezależniony od natężenia oświetlenia. Lambda określa próg od jakiego piksel jest uznawany za skórę. Wartość parametru zależy wyłącznie od użytkownika. Jeżeli wartość dla danego piksela obliczona przez funkcję jest mniejsza równa Lambda to piksel zostaje oznaczony jako skóra, a jego kolor zmieniony na biały. Wraz ze wzrostem wartości parametru Lambda wzrasta liczba pikseli rozpoznanych jako skóra.

a) Oświetlenie dzienne

Oryginalny	Lambda: 1 V: 0	Lambda: 1 V: 50	Lambda: 1 V: 100
			
Lambda: 1 V: 150	Lambda: 1 V: 200	Lambda: 2 V: 0	Lambda: 2 V: 50
			
Lambda: 2 V: 100	Lambda: 2 V: 150	Lambda: 2 V: 200	Lambda: 3 V: 0
			
Lambda: 3 V: 100	Lambda: 5 V: 0	Lambda: 5 V: 100	Lambda: 8 V: 0
			
Lambda: 8 V: 100	Lambda: 12 V: 0	Lambda: 12 V: 100	Z filtrem bilateralnym
			

b) Oświetlenie sztuczne

Oryginalny	Lambda: 1 V: 0	Lambda: 1 V: 50	Lambda: 1 V: 100
			
Lambda: 1 V: 150	Lambda: 1 V: 200	Lambda: 2 V: 0	Lambda: 2 V: 50
			
Lambda: 2 V: 100	Lambda: 2 V: 150	Lambda: 2 V: 200	Lambda: 3 V: 0
			
Lambda: 3 V: 100	Lambda: 5 V: 0	Lambda: 5 V: 100	Lambda: 8 V: 0
			
Lambda: 8 V: 100	Lambda: 12 V: 0	Lambda: 12 V: 100	Z filtrem bilateralnym
			

System powinien być skalibrowany w warunkach w jakich będzie działał. Należy pamiętać, że zmiana oświetlenia wymusza skorygowanie współczynników odpowiedzialnych za kolor skóry. W przypadku błędnego dobrania parametrów metoda wspierająca funkcję *Haar-like* poprzez odrzucanie obszarów nie zawierających skóry może nie działać poprawnie lub uniemożliwić działanie systemu. Film *skin.avi* przedstawia działanie maski wykrywającej skórę. Natomiast na filmie *Test_Skin.avi* przedstawiono jak detekcja skóry eliminuje twarze znajdujące się na czarno-białych zdjęciach.

3.5.3 Rozpoznawanie twarzy

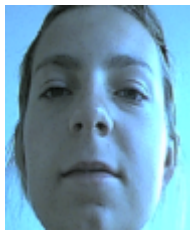
Testy rozpoznawania twarzy zostały przeprowadzone w dwóch rodzajach oświetlenia: dziennym i sztucznym (stałe warunki) z wykorzystaniem trzech metod porównywania wektorów wynikowych: odległość euklidesowa, odległość Mahalanobisa oraz metoda n-najbliższych sąsiadów (dla poniższych testów n = 5). Dodatkowo testy zostały przeprowadzone dla trzech różnych baz danych: baza obrazów pozyskanych przy świetle dziennym, sztucznym oraz mieszana utworzona z połączenia obu baz.

a) Bazy obrazów

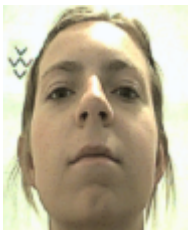
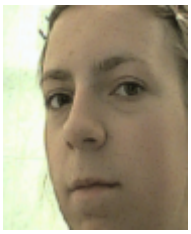
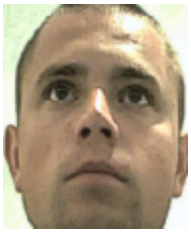
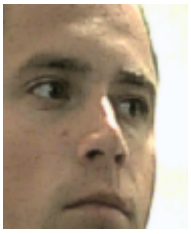
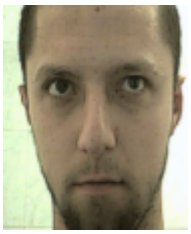
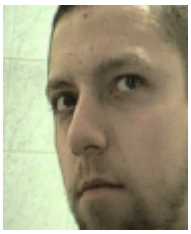
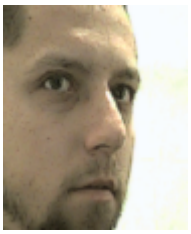
Każda baza zawiera pięć obrazów każdej z osób wykonanych pod różnymi kątami względem kamery. Obrazy maja rozmiary 92x112 pikseli i są w formacie nieskompresowanym *.bmp.

- Oświetlenie dzienne

  Na niebieskie zabarwienie obrazów miał wpływ automatyczny dobór balansu bieli przez zastosowaną kamerę.

Osoba 1				
				
Osoba 2				
				
Osoba 3				
				

- Oświetlenie sztuczne

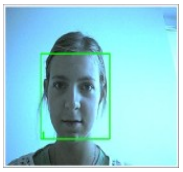
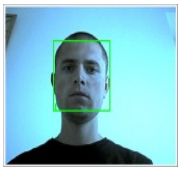
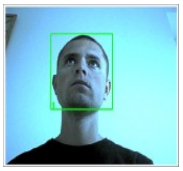
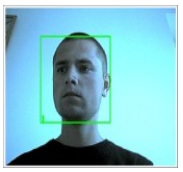
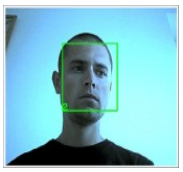
Osoba 1				
				
Osoba 2				
				
Osoba 3				
				

- Oświetlenie mieszane

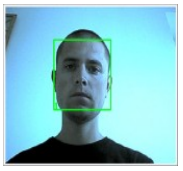
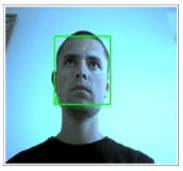
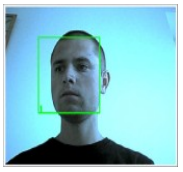
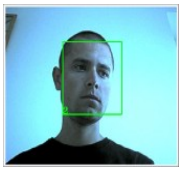
Baza utworzona z połączenia obrazów wykonanych w oświetleniu sztucznym i dziennym.

b) Testy przy oświetleniu dziennym

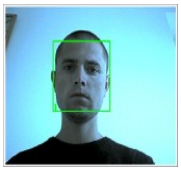
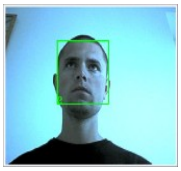
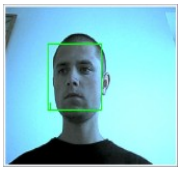
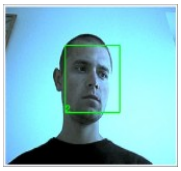
- Baza z oświetleniem dziennym, odległość euklidesowa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Niepoprawne	Poprawne	Niepoprawne
					
Poprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne	Poprawne
					

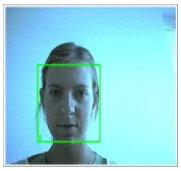
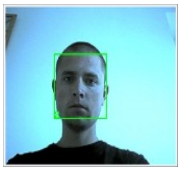
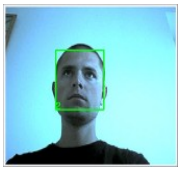
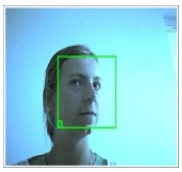
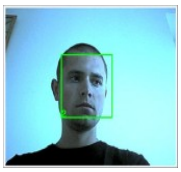
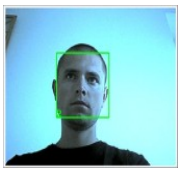
- Baza z oświetleniem dziennym, odległość Mahalanobisa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Poprawne	Poprawne	Poprawne
					
Poprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne	Poprawne
					

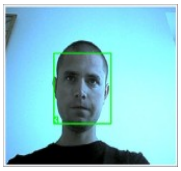
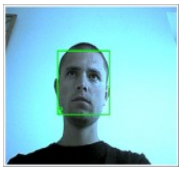
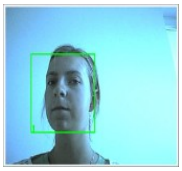
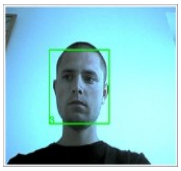
- Baza z oświetleniem dziennym, metoda n-najbliższych sąsiadów

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Poprawne	Poprawne	Niepoprawne
					
Poprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne	Niepoprawne
					

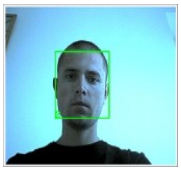
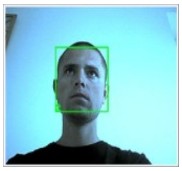
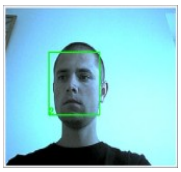
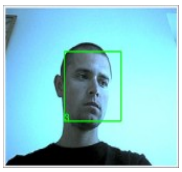
- Baza z oświetleniem sztucznym, odległość euklidesowa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Niepoprawne	Poprawne	Poprawne	Poprawne	Poprawne
					
Niepoprawne	Niepoprawne	Poprawne	Poprawne	Poprawne	Poprawne
					

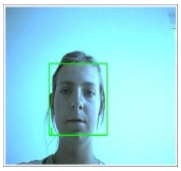
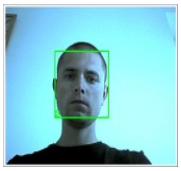
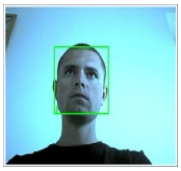
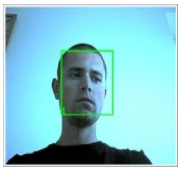
- Baza z oświetleniem sztucznym, odległość Mahalanobisa

Osoba 1		Osoba 2		Osoba 3	
Niepoprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne	Poprawne
					
Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne
					

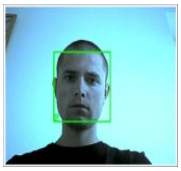
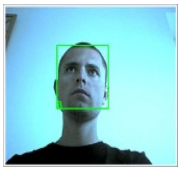
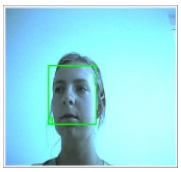
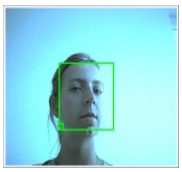
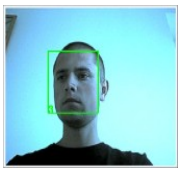
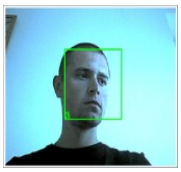
- Baza z oświetleniem sztucznym, metoda n-najbliższych sąsiadów

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Niepoprawne	Poprawne	Poprawne	Poprawne	Poprawne
					
Niepoprawne	Niepoprawne	Poprawne	Niepoprawne	Niepoprawne	Poprawne
					

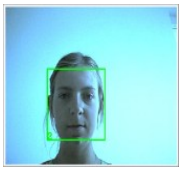
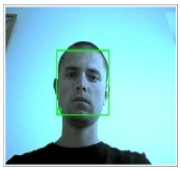
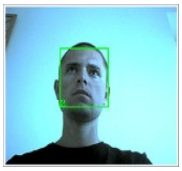
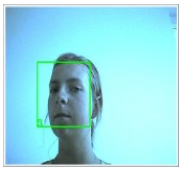
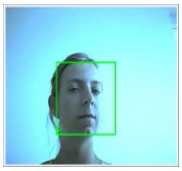
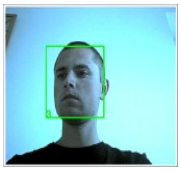
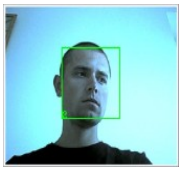
- Baza z oświetleniem mieszanym, odległość euklidesowa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Niepoprawne	Poprawne	Niepoprawne	Poprawne	Niepoprawne
					
Niepoprawne	Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne
					

- Baza z oświetleniem mieszanym, odległość Mahalanobisa

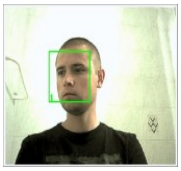
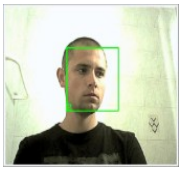
Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Poprawne	Niepoprawne	Poprawne	Niepoprawne
					
Niepoprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne
					

- Baza z oświetleniem mieszanym, metoda n-najbliższych sąsiadów

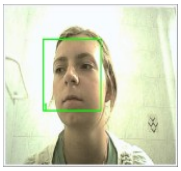
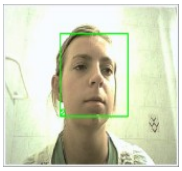
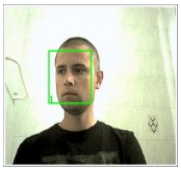
Osoba 1		Osoba 2		Osoba 3	
Niepoprawne	Niepoprawne	Poprawne	Poprawne	Poprawne	Poprawne
					
Niepoprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne	Niepoprawne
					

c) Testy przy oświetleniu sztucznym

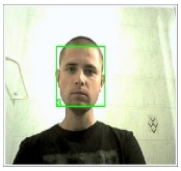
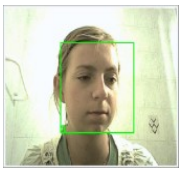
- Baza z oświetleniem dziennym, odległość euklidesowa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Poprawne	Niepoprawne	Niepoprawne
					
Poprawne	Niepoprawne	Niepoprawne	Poprawne	Niepoprawne	Niepoprawne
					

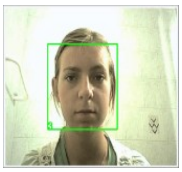
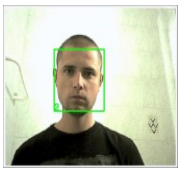
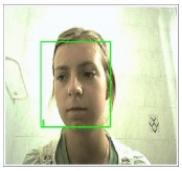
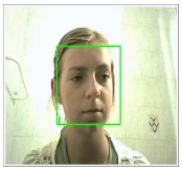
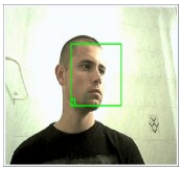
- Baza z oświetleniem dziennym, odległość Mahalanobisa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne
					
Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Niepoprawne	Niepoprawne
					

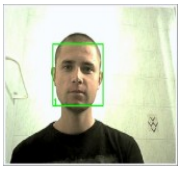
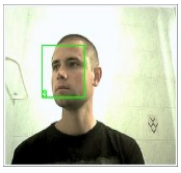
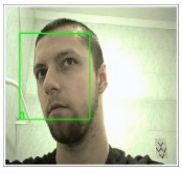
- Baza z oświetleniem dziennym, metoda n-najbliższych sąsiadów

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne
					
Poprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne	Poprawne
					

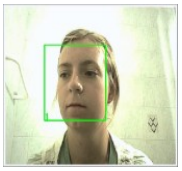
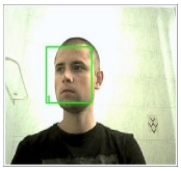
- Baza z oświetleniem sztucznym, odległość euklidesowa

Osoba 1		Osoba 2		Osoba 3	
Niepoprawne	Niepoprawne	Poprawne	Niepoprawne	Poprawne	Niepoprawne
					
Poprawne	Poprawne	Poprawne	Niepoprawne	Poprawne	Poprawne
					

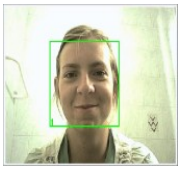
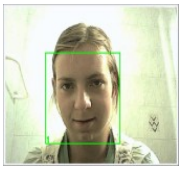
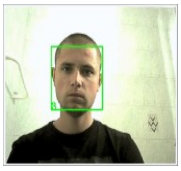
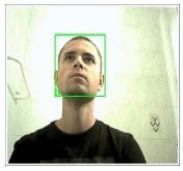
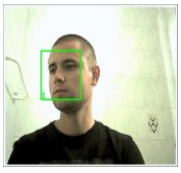
- Baza z oświetleniem sztucznym, odległość Mahalanobisa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne
					
Niepoprawne	Poprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne
					

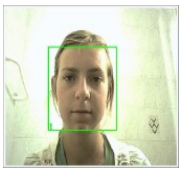
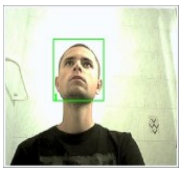
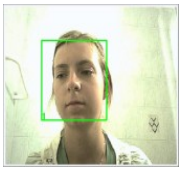
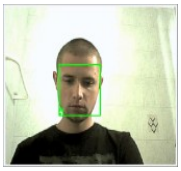
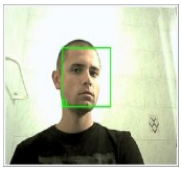
- Baza z oświetleniem sztucznym, metoda n-najbliższych sąsiadów

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne
					
Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne
					

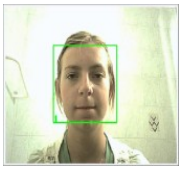
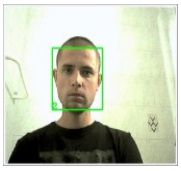
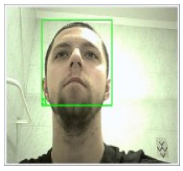
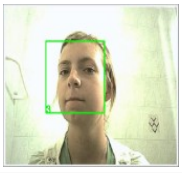
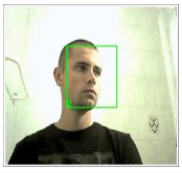
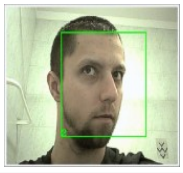
- Baza z oświetleniem mieszanym, odległość euklidesowa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Poprawne	Niepoprawne	Niepoprawne	Poprawne	Niepoprawne
					
Poprawne	Niepoprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne
					

- Baza z oświetleniem mieszanym, odległość Mahalanobisa

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Niepoprawne	Poprawne	Niepoprawne	Poprawne	Niepoprawne
					
Poprawne	Poprawne	Niepoprawne	Niepoprawne	Poprawne	Poprawne
					

- Baza z oświetleniem mieszanym, metoda n-najbliższych sąsiadów

Osoba 1		Osoba 2		Osoba 3	
Poprawne	Niepoprawne	Poprawne	Niepoprawne	Poprawne	Poprawne
					
Niepoprawne	Poprawne	Niepoprawne	Poprawne	Poprawne	Niepoprawne
					

d) Podsumowanie

Poniższe tabele przedstawiają wyniki uzyskane na pojedynczych zdjęciach w różnych warunkach oświetlenia, oraz skuteczność rozpoznania każdej z twarzy.

Skuteczności metod porównywania wektorów w różnych warunkach		
Metoda / Oświetlenie	Dzienne	Sztuczne
Euklidesowa	55,56%	50,00%
Mahalanobisa	58,33%	50,00%
N-Najbliższych sąsiadów	69,44%	55,56%

Tabela przedstawia średnią wyników każdej z metod pozyskanych przy oświetleniu dziennym i sztucznym z zastosowaniem każdej z trzech baz twarzy.

Poprawność rozpoznania każdej z osób w różnych warunkach		
Osoba / Oświetlenie	Dzienne	Sztuczne
1	44,44%	66,67%
2	50,00%	22,22%
3	77,78%	63,89%

Skuteczność detekcji twarzy ustawionych przodem do kamery wyniosła 81,23%. Jest to wynik średni dla wszystkich rodzajów oświetlenia i każdej z trzech baz. Ocenie podlegała jedynie wydajność metod, a nie samego rozpoznawania twarzy. Wyniki zostały uzyskane dla pojedynczych fotografii. Należy podkreślić że baza zawierała jedynie pięć zdjęć dla każdej z osób. Jest to zbyt mała próba aby można było uznać wyniki za wiążące. Dla innego zestawu zdjęć możliwe jest uzyskanie lepszych wyników. Działanie programu przedstawia film *rozpoznawanie.avi*. Wykorzystuje on bazę danych zawierającą 20 zdjęć dla każdej z twarzy. Widać na nim, iż program prawidłowo rozpoznaje twarz i pomimo, że czasami prostokąt zmienia kolor na czerwony, co oznacza większą odległość wektora twarzy niż ustalony próg, to w dalszym ciągu wskazywany jest poprawny numer twarzy.

3.5.4 Detekcja i rozpoznawanie (oświetlenie stałe)

Film *Test.avi* przedstawia wykrywanie oraz detekcję twarzy dla bazy zawierającej 10 zdjęć każdej z 41 osób. Do testu wykorzystano bazę The ORL Database of Faces [34] wzbogaconą o zdjęcia autora pracy (osoba nr 1 w zbiorze). Pokazano również możliwość podejrzenia maski skóry w dowolnym momencie. W trakcie animacji została zliczana liczba klatek, na których wykryto twarz nr 1 oraz całkowita liczba klatek animacji. Po sprawdzeniu wyników okazało się, że skuteczność rozpoznania osoby nr 1 wyniosła 99.8%.

3.5.5 Detekcja i rozpoznawanie (oświetlenie zmienne)

Test przedstawiony jest na filmie *Test_Light.avi*. Pokazano na nim jak program radzi sobie z rozpoznaniem twarzy przy świetle dziennym, w półmroku oraz przy świetle sztucznym. Zastosowana baza danych zawierała po 10 zdjęć zrobionych przy oświetleniu sztucznym i dziennym. Kamera pomimo półmroku była wstanie wyświetlić obraz twarzy. Nie był on jednak dość wyraźny. Spowodowało to niewielkie problemy z detekcją twarzy i samym rozpoznaniem. Przy oświetleniu dziennym oraz sztucznym nie zaobserwowano podobnych problemów. Całkowita skuteczność sprawdzona na podstawie ilości klatek animacji zawierających poprawnie rozpoznaną twarz wyniosła 91%.

3.5.6 Generowanie bazy danych

Testy miały na celu sprawdzenie wydajności tworzenia nowej bazy danych. Wszystkie zdjęcia były w formacie *.bmp o wymiarach 92 x 112 pikseli.

Szybkość generowania bazy twarzy w zależności od liczba zdjęć		
Liczba osób	Liczba zdjęć na osobę	Czas generowania bazy [s]
5	10	3
10	10	8
20	10	14
40	10	32

Jak wykazały testy stworzenie bazy dla 40 osób składającej się łącznie z 400 zdjęć zajęło zaledwie 32s na wolnym komputerze przeznaczonym do pracy biurowej.

4. Podsumowanie

W pracy opisano proces tworzenia systemu umożliwiającego detekcję i rozpoznawanie twarzy. Ponadto omówiono zagadnienia związane z tą tematyką oraz zaproponowano schemat takiego systemu.

Głównym założeniem pracy było stworzenie systemu detekcji i rozpoznawania twarzy. Cel ten udało się zrealizować. System działa poprawnie, zarówno detekcja jak i rozpoznawanie twarzy cechuje się dużą szybkością działania i dobrymi wynikami. Badania wykazały, że aplikacja powinna działać najlepiej w stałych warunkach, co niejako potwierdza wady zastosowanych metod. System dostarcza wiele opcji umożliwiających jego konfigurację. Istnieje również możliwość wyboru bazy zawierającej twarze oraz generowanie własnej na podstawie przygotowanych zdjęć. Klasyfikator kaskadowy wykorzystywany w procesie detekcji również można zmieniać, a w pracy został przedstawiony sposób, jak stworzyć własny klasyfikator oraz stworzono metody umożliwiające automatyzację tego procesu.

Otrzymane wyniki potwierdzają skuteczność metody *Haar-like* w procesie detekcji twarzy. Działa ona dobrze nawet dla twarzy znajdujących się w dużej odległości od kamery oraz na obrazie o niskiej rozdzielczości. Dodatkowo metoda została wzbogacona o możliwość odrzucania obrazów nie zawierających ludzkiej skóry. Z eksperymentów wynika, że nie zmniejsza ona znacząco wydajności systemu. Opcja ta znacznie poprawia działanie systemu w sytuacji, gdy obszar poszukiwania twarzy jest mały i istnieje ryzyko wstępnego zaznaczenia jako twarze wielu elementów tła. Elementy te są odrzucane w dalszym procesie przetwarzania danych, co pozwala znacznie zmniejszyć ilość elementów, jakie muszą być wyszukane i porównane ze wzorcami zawartymi w bazie danych. Również proponowana implementacja metody *EigenFace* działa poprawnie. Dzięki dostosowaniu jej parametrów do konkretnego systemu udało się uzyskać dużą szybkość procesu rozpoznawania twarzy, niemniej jednak wymaga ona stałych warunków oświetleniowych. Jak pokazały istnieje konieczność dostarczenia znacznej liczby zdjęć dla każdej z osób, co w istotny sposób zwiększa rozmiar bazy danych. Testy wykazały, że system może działać w różnych warunkach oświetleniowych jeżeli baza zawiera zdjęcia osób pozyskane w tych

warunkach. Oczywiście rozmiar bazy dodatkowo się powiększa, ale system staje się bardziej elastyczny.

Główne problemy dotyczyły przetwarzania grafiki oraz bazy danych. Dzięki zastosowaniu biblioteki *OpenCV* możliwe było częściowe wyeliminowanie problemów dotyczących przetwarzania obrazów. Dostarczone przez nią funkcje z założenia działają w czasie rzeczywistym. Sama baza danych pomimo dużych rozmiarów umożliwia szybkie wyszukiwanie danych. Osiągnięto to dzięki zastosowaniu *PCA,* co spowodowało znaczne zmniejszenie wymiaru wektorów. Pewną uciążliwością jest konieczność przeskalowania wszystkich obrazów bazy do tego samego rozmiaru. Wynika to z zastosowania *PCA*. Nie ma natomiast znaczenia jaki to będzie rozmiar.

System działa na tyle dobrze, że z powodzeniem może zostać wykorzystany do rozpoznawania stałych klientów w sklepach, śledzenia pracowników w firmach czy obliczania jak długo twarz aktora jest widoczna w filmie. Istnieje możliwość wykorzystania tego systemu w kontroli dostępu, ale niezbędne są do tego stałe warunki. Oczywiście najlepiej aby system był tylko dodatkowym zabezpieczeniem. Istnieje wiele czynników jakie mogą utrudnić poprawne rozpoznanie osoby, bądź oszukać aplikację.

System można wzbogacić o możliwość automatycznego przechwytywania twarzy i umieszczania ich w bazie, co umożliwiłoby zbieranie danych statystycznych o np. kierunku poruszania się pieszych. Dodatkowo można użyć kamery termowizyjnej, co uodporniłoby system na próby oszustwa z wykorzystaniem np. zdjęcia. Kolejnym zabezpieczeniem może być również kazanie osobie np. mrugnąć oczami czy wykonać inną czynność świadczącą o tym, że przed kamerą stoi żywy człowiek. Wszystkie te rozwiązania nie były celem pracy, a jedynie mają na celu przedstawienie możliwości dalszego rozwoju programu. System można również połączyć z systemem bazującym na wyszukiwaniu cech charakterystycznych twarzy, w celu zwiększenia skuteczności rozpoznawania. Jeszcze jednym rozwiązaniem zwiększającym skuteczność systemu w dziedzinie wykrywania twarzy mogłoby być dodanie możliwości śledzenia obiektów. Wyeliminowałoby to wykrywanie np. manekinów czy zdjęć ludzi na reklamach w supermarketach.

5. Bibliografia

Wykaz źródeł

1. Sawicki, D.: „Światło i barwa w grafice komputerowej”, Grafika komputerowa i wizualizacja, [online], http://wazniak.mimuw.edu.pl, 27 listopada 2007, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

   http://wazniak.mimuw.edu.pl/index.php?title=GKIW_Modu%C5%82_2_-_%C5%9Awiat%C5%82o_i_barwa_w_grafice_komputerowej

2. Wojciechowski, A.: „Grafika komputerowa”, [online], Politechnika Łódzka, 6 listopada 2005, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

   http://ics.p.lodz.pl/~adamwoj/WSFI/GK/Wyklad_1_GK.pdf

3. Hebisz, T.: „Model barw”, Multimedia i grafika komputerowa, [online], Instytut Sterowania i Systemów Informatycznych, 13 stycznia 2003, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

   http://astrophysics.fic.uni.lodz.pl/lectures/graf-barwy.pdf

4. „RGB”, [online], http://pl.wikipedia.org, 14 listopada 2004, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

   http://pl.wikipedia.org/wiki/Plik:RGB_farbwuerfel.jpg

5. Tomasi, C.; Manduchi, R.: "Bilateral Filtering for Gray and Color Images", Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India, 1998, Dostępny w Internecie:

   http://www.ece.lsu.edu/gunturk/EE7700/Bilateral.pdf

6. „Odległość Mahalanobisa”, [online], http://pl.wikipedia.org, 8 września 2006, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

   http://pl.wikipedia.org/wiki/Odleg%C5%82o%C5%9B%C4%87_Mahalanobisa

7. Viola, P.; Jones, M.: "Rapid Object Detection using a Boosted Cascade of Simple Features," cvpr, vol. 1, pp.511, 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01) - Volume 1, 2001

8. Viola, P.; Jones, M. J.: „Robust Real-Time Face Detection”, International Journal of Computer Vision, v.57 n.2, p.137-154, Maj 2004

9. Seo, N.: „Tutorial: OpenCV haartraining (Rapid Object Detection With A Cascade of Boosted Classifiers Based on Haar-like Features)”, [online], 2007, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

   http://note.sonots.com/SciSoftware/haartraining.html

10. Florian, A.: „How-to build a cascade of boosted classifiers based on Haar-like features”, OpenCV's Rapid Object Detection, [online], 2 września 2003, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

    http://lab.cntl.kyutech.ac.jp/~kobalab/nishida/opencv/OpenCV_ObjectDetection_HowTo.pdf

11. Obiect Maker, [online], [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.bernardotti.it/portal/attachment.php?s=3d8bd0ab0ff172edf59ebebbafdf93af&attachmentid=137&d=1228128947

12. Zhang, S.; Turk, M.: „Eigenfaces”, [online], http://www.scholarpedia.org, 3(9):4244, 2008, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.scholarpedia.org/article/Eigenfaces

13. Sirovich, I.; Kirby, M.: "Low-dimensional procedure for the characterization of human faces", Journal of Opt. Soc. Am., vol. 4, pp. 519 - 524, 1987, Dostępny w Internecie:

http://www.cs.bgu.ac.il/~icbv071/Readings/1987-Sirovich_and_Kirby-Low_Dimensional_Procedure_for_the_Characterization_of_Human_Faces.pdf

14. Turk, M.; Pentland, A.: „Face Recognition using Eigenfaces”, IEEE Conference on Computer Vision and Pattern Recognition, Maui, HI, Czerwiec 1991, Dostępny w Internecie:

http://www.cs.ucsb.edu/~mturk/Papers/mturk-CVPR91.pdf

15. „Rysy twarzy jako unikalny klucz dostępu”, [online], BoschLive-Magazyn, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.bosch.pl/content/language1/html/715_2886.htm

16. „MSC/FaceFinder”, [online], GEUTEBRÜCK, 2006, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.arpol.pl/plikget.php?pl_id=1866

17. „Informacje o produkcie”, [online], AutoID Polska S.A., 2006, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.snappy.pl/info.html

18. „Parametry techniczne”, [online], AutoID Polska S.A., 2006, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.snappy.pl/tech_specs.html

19. „FLIR Infrared Cameras”, [online], FLIR Systems, 2009, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.flir.com/uploadedImages/Thermography_APAC/Industries/Medical/Face%20fever.jpg

20. „Rozpoznawanie twarzy”, [online], Telsat Electronic Systems, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.555.pl/detekcja_twarzy/kryminalistyka_duze.jpg

21. „Face Base”, [online], The Center for Biological & Computational Learning (CBCL), [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://cbcl.mit.edu/projects/cbcl/software-datasets/faces.tar.gz

22. „Face Data Readme”, [online], MIT Center For Biological and Computation Learning, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://cbcl.mit.edu/projects/cbcl/software-datasets/FaceData1Readme.html

23. Huang, T. S.; Yang, G.J; Tang, G.Y.: "A fast two-dimensional median filtering algorithm", IEEE Transactions on Acoustics, Speech, and Signal Processing, 1979, 27(1):13-18.

24. OpenCV Wiki, [online], [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://opencv.willowgarage.com/wiki/

25. „OpenCV”, [online], http://pl.wikipedia.org, 16 czerwca 2005, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://en.wikipedia.org/wiki/OpenCV

26. Żyliński G.: „Rozpoznawanie układów dłoni”, [online], Politechnika Warszawska, 20 września 2007, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.ee.pw.edu.pl/~czajewsw/forum/download.php?id=196

27. Hewitt, R.: Seeing With OpenCV Part 3, Follow That Face, Servo Magazine, 2007, 36-40.

28. Hewitt R.: Seeing With OpenCV Part 5, Implementing Eigenface, Servo Magazine, 2007, 44-50.

29. „Getting Started with OpenCV in Visual Studio 2005”, I Hate (Love) OpenCV, [online], 2006, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://opencv.blogspot.com/

30. „OpenCV with Visual C++ 6.0, 2005 Express, and 2008 Express”, [online], OpenCV Wiki, 13 czerwca 2008, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://opencv.willowgarage.com/wiki/VisualC%2B%2B

31. Lienhart, R.; Maydt, J.: „An Extended Set of Haar-like Features for Rapid Object Detection”, IEEE ICIP 2002, Vol. 1, 2002, pp. 900-903, Dostępny w Internecie:

http://www.lienhart.de/ICIP2002.pdf

32. „History of Face Recognition”, [online], 2000, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://vismod.media.mit.edu/tech-reports/TR-516/node7.html

33. Wojciechowski A.: „Przetwarzanie obrazów”, [online], 15 stycznia 2000, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://ics.p.lodz.pl/~adamwoj/WSFI/PO/4_wyklad_PO.pdf

34. „The ORL Database of Faces”, [online], AT&T Laboratories Cambridge, 2002, [ostatni dostęp: 15 września 2009], Dostępny w Internecie:

http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Indeks ilustracji