Rozpoznawanie obrazów: Wykrywanie orientacji zdjęć przez lokalizację twarzy

Marek Ruciński Bartosz Szopka

24 czerwca 2006

1 Cel i motywacje

Celem projektu było stworzenie programu umożliwiającego automatyczne wykrycie orientacji zdjęcia na podstawie zlokalizowanych na fotografii twarzy. Motywacją do podjęcia badań był fakt sporej popularności cyfrowej fotografii i, co za tym idzie, ogromna ilość zdjęć wykonywanych aparatami cyfrowymi. Jednakże większość obecnie wykorzystywanych urządzeń (szczególnie tych z „niższej półki") nie ma wbudowanej funkcji automatycznego zapisu danych o orientacji aparatu w trakcie wykonywania zdjęcia. W rezultacie użytkownik zmuszony jest do uciążliwego ręcznego obracania zdjęć.

Rok wcześniej w ramach laboratorium Rozpoznawania Obrazów powstał projekt wykrywający orientację zdjęcia osiągający dobre rezultaty dla fotografii robionych na otwartych przestrzeniach. Bardzo trudno było znaleźć cechy obrazu pozwalające na ustalenie orientacji zdjęć robionych w zamkniętych pomieszczeniach.

Prace nad opisywanym w tym sprawozdaniu projektem rozpoczęły się od spostrzeżenia, że zdecydowana większość zdjęć nie zawierających pejzaży zawiera ludzi. A spośród tych, w przeważającej części orientacja twarzy jest zgodna z orientacją zdjęcia. Dla potwierdzenia tej hipotezy przeanalizowano przykładowy zbiór zdjęć (sesja zdjęciowa z III szkoły naukowej SKNO, 25-29/10/2004, Karpacz).

- 178 – łączna liczba zdjęć,
- 23 – pejzaże,
- 30 – zdjęcia bez ludzi nie będące pejzażami,
- 125 – zdjęcia zawierające ludzi (wśród nich niemal w 100% orientacja twarzy była zgodna z orientacją zdjęcia).

Powyższa analiza pokazała, że podejście bazujące na lokalizacji i badaniu orientacji twarzy na zdjęciach może dać dobre wyniki praktyczne.

W ramach tego sprawozdania opisano proponowane rozwiązanie wraz z założeniami (patrz: 2), przedstawiono wykorzystany klasyfikator *Haar-like* (patrz: 3) i próbę stworzenia własnego klasyfikatora (patrz: 4) oraz omówiono wyniki przeprowadzonego eksperymentu (patrz: 5).

2 Proponowane rozwiązanie

2.1 Założenia

Specyfika obranego podejścia do wykrywania orientacji zdjęcia przez lokalizację twarzy stawia jedno ważne założenie: analizowane zdjęcię musi zawierać co najmniej jedną twarz i (w celu zapewnienia możliwości otrzymania poprawnego wyniku) orientacja twarzy musi być zgodna z docelową orientacją zdjęcia.

2.2 Algorytm

W ramach stworzonej aplikacji wykorzystano klasyfikator typu *Haar-like features* pozwalający na lokalizację w obrazie „pionowo" utawionych twarzy. Proponowane rozwiązanie opiera się zatem na obracaniu badanego zdjęcia o kąty 0°, 90°, 180° i 270° oraz porównywaniu liczby wykrytych na tych obrazach twarzy. Łatwo można zauważyć, że w przypadku doskonałego klasyfikatora (takiego, który zlokalizuje na obrazie wszystkie twarze i tylko twarze) kąt obrotu ustawiający zdjęcie w prawidłowej orientacji powinien w wyniku lokalizacji zwrócić liczbę faktycznie obecnych na zdjęciu twarzy, natomiast przy pozostałych kątach klasyfikator nie powinien wykryć żadnej twarzy. Zatem badanie orientacji zdjęcia polegałoby na sprawdzeniu, dla którego kąta obrotu znajdowane są na zdjęciu jakieś twarze. W praktyce jednak nie został jak dotąd stworzony klasyfikator o tak perfekcyjnych paramertach. Dlatego nierzadko pojawiają się błędy niewykrycia istniejących twarzy (*false negative*) i wykrycia obiektów nie będących twarzami (*false positive*). W przypadku korzystania z takiego rzeczywistego klasyfikatora należy wziąć pod uwagę, że przy każdym kącie obrotu zdjęcia mogą zostać wykryte jakieś „twarze" – prawdziwe bądź fałszywe. Jednak przy założeniu, że klasyfikator popełnia niewiele błędów można spróbować znaleźć kąt obrotu, dla którego wykrywana jest największa liczba twarzy i ten kąt zaproponować jako wynikową orientację. Agorytm może wstrzymać się od głosu jeśli dla więcej niż jednej orientacji wystąpi maksimum.

Zatem algorytm proponowanego w ramach tego projektu rozwiązania wygląda następująco:

1. Przyjmij możliwe kąty obrotu jako [0°, 90°, 180°, 270°].
2. Dla każdego kąta obróć zdjęcie i policz zlokalizowane twarze.
3. Znajdź kąt, dla którego liczba zlokalizowanych twarzy jest największa.
4. Podaj znaleziony kąt jako wynikową orientację zdjęcia.

2.3 Implementacja

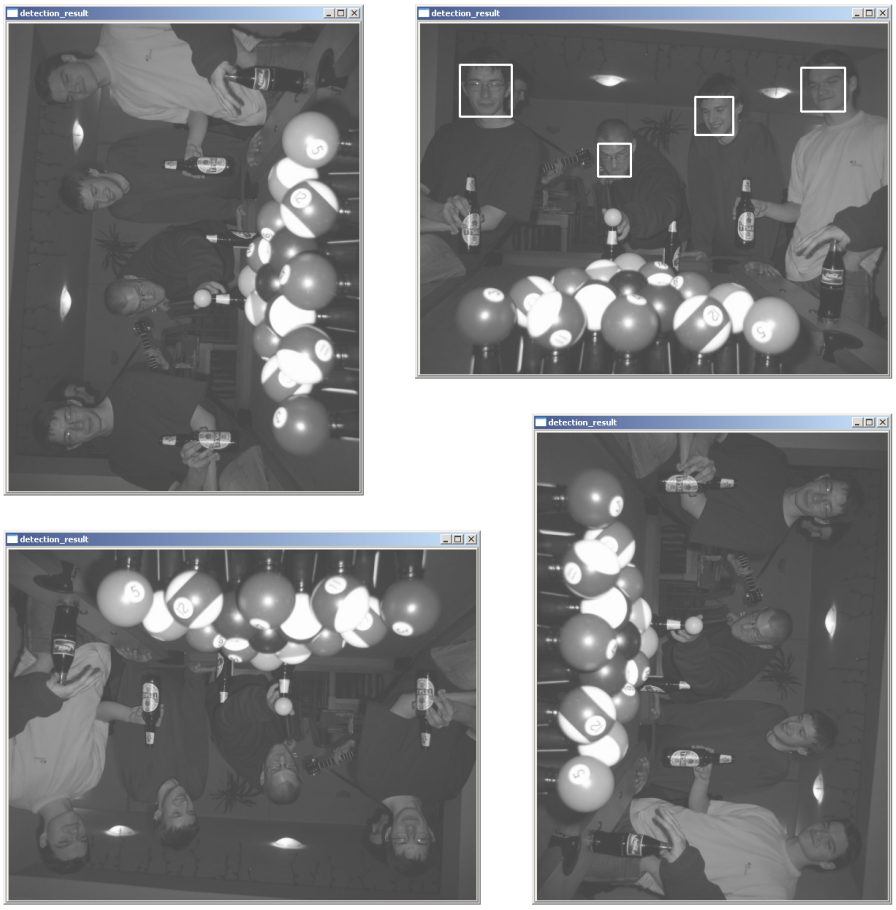
W celu realizacji powyższego algorytmu stworzono aplikację w środowisku Visual Studio z wykorzystaniem biblioteki OpenCV. Ponieważ z założenia program miał posłużyć do testów analizowanego podejścia nie został on wyposażony w rozbudowany interfejs graficzny. Aplikacja działa z poziomu lini poleceń i przyjmuje jako parametr nazwę pliku ze zdjęciem do przeanalizowania lub nazwę katalogu z badanymi obrazami. W wyniku swojego działania program wypisuje na standardowe wyjście liczbę zlokalizowanych dla każdego kąta obrotu twarzy.

Uruchomienie programu z przełącznikiwm *-c=<nazwa pliku xml z klasyfikatorem>* umożliwia wybór własnego klasyfikatora Haara, a przełącznik *-s* pozwala na podgląd analizowanego zdjęcia i twarzy zlokalizowanych dla każdego kąta obrotu. Przykładowe wyniki działania aplikacji z podglądem zdjęć ukazano na rysunkach 2.1 (prawidłowa lokalizacja) i 2.2 (błędy *false positive* i *false negative*).

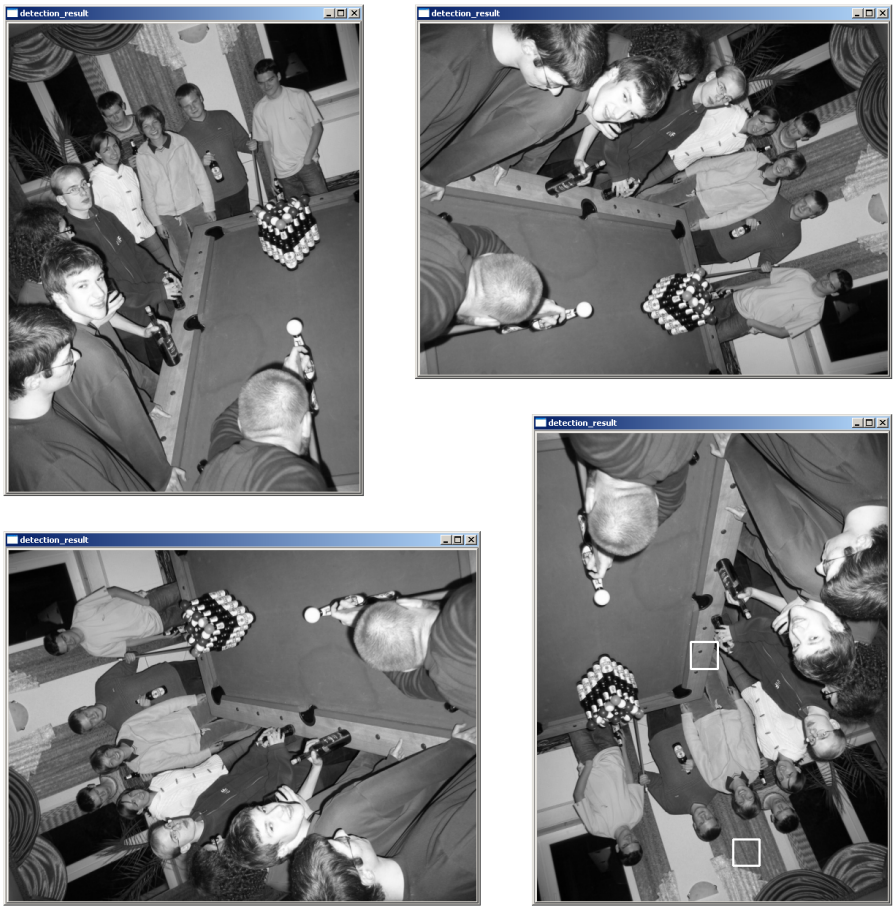
3 Klasyfikator *Haar-like features*

Lokalizacja obiektów (w tym przypadku twarzy) oparta o metodę *Haar-like features* wykorzystuje klasyfikator „kaskadowy", w którym weryfikacja hipotezy podzielona jest na etapy. Można powiedzieć, że ostateczna decyzja podejmowana jest na podstawie decyzji podjętych na „niższym szczeblu". Na każdym etapie weryfikacji wykorzystywany jest złożony klasyfikator wykorzystujący *boosting* (Discrete Adaboost, Real Adaboost, Gentle Adaboost, Logitboost), a podstawowym klasyfikatorem jest drzewo decyzyjne (z co najmniej 2 liśćmi).

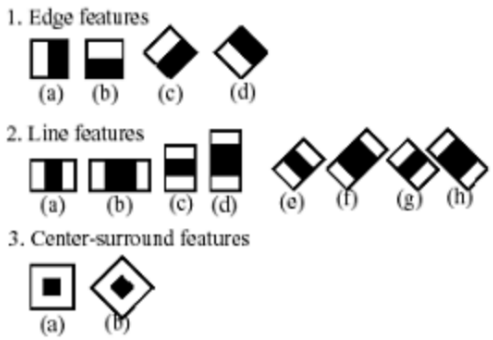
Decyzje „najniższego szczebla" podejmowane są w oparciu o tzw. „cechy Haara" (*Haar-like features*). Są to proste „maski" pozwalające na mierzenie podstawowych cech obrazu. Wykorzystywane w klasyfikatorach cechy przedstawiono na rysunku 3.1.



Rysunek 2.1: Wynik działania – poprawna lokalizacja twarzy



Rysunek 2.2: Wynik działania – błędna lokalizacja twarzy



Rysunek 3.1: *Haar-like features* – cechy wykorzystywane w klasyfikacji

W celu lokalizacji obiektów w większym obrazie stosowana jest technikę przesuwnego okna. Obraz jest skanowany klasyfikatorem w celu znalezienia pozycji obiektu. Konstrukcja klasyfikatora *Haar-like* umożliwia jego skalowanie (zamiast skalowania obrazu)

Niewątpliwą zaletą tego klasyfikatora jest prostota jego użycia. Wykorzystana w implementacji biblioteka OpenCV zawiera wszystkie narzędzia niezbędne do tworzenia klasyfikatorów Haara i ich użycia we własnym programie. Dodatkowym plusem jest możliwość wyuczenia lokalizowania dowolnych obiektów.

4 Konstrukcja własnego klasyfikatora *Haar-like features*

Wraz z biblioteką OpenCV dostarczone są narzędzia umożliwiające samodzielną konstrukcję klasyfikatorów korzystających z cech *Haar-like features*. Programy te znajdują się w podkatalogu *bin* katalogu głównego biblioteki. Dostępne są również ich kody źródłowe, co umożliwia m.in. przekompilowanie ich tak, by wykorzystywały możliwości najnowszych wielordzeniowych procesorów.

Przed przystąpieniem do uczenia klasyfikatora, należy zgromadzić dwa zbiory przykładów uczących: zbiór przykładów negatywnych oraz zbiór przykładów pozytywnych. Zbiór przykładów negatywnych powinien zawierać jak najwięcej obrazów (5000 do 10000). Niestety z dokumentacji nie wynika jasno co mają te obrazy przedstawiać – czy sceny nie zawierające wykrywanych obiektów, czy przykłady innych (nie tych klasyfikowanych) obiektów. Nazwy zgromadzonych plików wraz z ewentualnymi ścieżkami dostępu należy umieścić w pliku tekstowym (każda w osobnej linii).

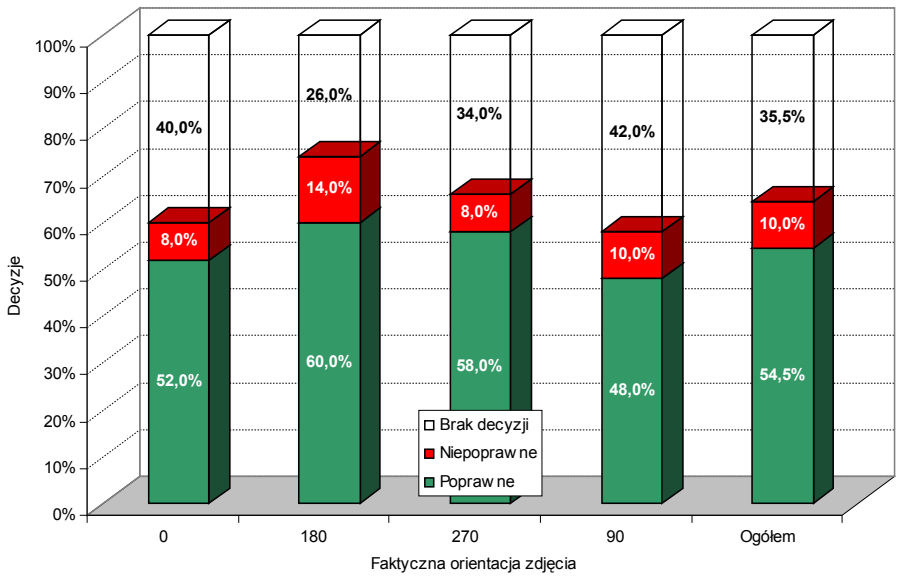
Jako zbiór przykładów pozytywnych zaleca się stosować jak najwięcej naturalnych zdjęć klasyfikowanych obiektów. Dostarczono co prawda narzędzie umożliwiające symulowanie różnych warunków oświetlenia i orientacji, jednakże oczywistym jest że lepsze wyniki uzyskuje się stosując ujęcia naturalne. Zbiór przykładów pozytywnych powinien mieć postać obrazów zawierających klasyfikowane obiekty. Znowu należy utworzyć plik tekstowy z listą wszystkich plików, dodatkowo jednak dla każdego z nich należy podać liczbę klasyfikowanych obiektów znajdujących się w danym obrazie oraz odpowiadające tym obiektom współrzędne prostokątów ograniczających.

W następnej kolejności należy ustalić, do jakich rozmiarów mają być skalowane klasyfikowane obiekty podczas nauczania klasyfikatora (domyślnie 24x24 piksele) oraz czy posiadają one pionową oś symetrii. Dysponując tymi informacjami, należy przetworzyć zbiór przykładów pozytywnych do postaci wektora uczącego (plik *.vec) korzystając z narzędzia `createsamples.exe`. Szczegółowy opis sposobu użycia tego narzędzia wraz z wszyskimi opcjami można znaleźć w podkatalogu *apps/HaarTraining/doc/* katalogu biblioteki OpenCV. Otrzymany plik *.vec wykorzystuje się do uczenia klasyfikatora.

Uczenie klasyfikatora odbywa się przy użyciu narzędzia `haartraining.exe`. Jako dane wejściowe przyjmuje on utworzony w poprzednim etapie plik *.vec, plik zawierający zbiór przykładów negatywnych oraz katalog docelowy, w którym zostanie zapisany utworzony klasyfikator. Można również podać szereg parametrów sterującym uczeniem oraz budową klasyfikatora. Ich szczegółowy opis można znaleźć we wspomnianym wcześniej katalogu z dokumentacją. Samo uczenie odbywa się w etapach. Pełny proces uczenia powinien się składać z 20-30 etapów. W każdej chwili proces uczenia można przerwać, otrzymując klasyfikator z ostatniego ukończonego etapu. Naukę można następnie wznowić, a rozpocznie się ona od momentu, w którym została przerwana.

Do testowania otrzymanego klasyfikatora służy narzędzie `performance.exe`. Należy oczywiście przygotować osobne zestwy plików z przykładami pozytywnymi i negatywnymi. Niewskazane jest, aby zbiory te zawierwały obrazy wykorzystywane do uczenia klasyfikatora. Szczegóły na temat sposobu użycia narzędzia można znaleźć we wspomnianej dokumentacji.

W ramach niniejszego projektu dokonano próby konstrukcji własnego klasyfikatora. Przygotowano zbiór uczący na podstawie sesji zdjęciowych z III Szkoły Naukowej SKNO (Karpacz, 2004)



Rysunek 5.1: Wyniki działania dla klasyfikatora `haarcascade-frontalface-alt.xml`.

oraz warsztatów Workshop on Scheduling and Bioinformatics (Lessach, 2006). Początkowy zbiór uczący zawierał 178 wyciętych twarzy oraz 176 przykładów obiektów nie będących twarzami. Konstrukcja wektora uczącego nie odbyła się bezproblemowo, a na minus wspomnianych narzędzi należy zapisać mało zrozumiałe komunikaty o błędach (użyteczne bardziej dla programisty niż dla użytkownika). Ostatecznie uruchomiono jednak proces uczenia. Ten jednak uległ znaczącemu spowolnieniu na 3-im etapie. Po kilku godzinach bezowocnych oczekiwań na jakiekolwiek oznaki postępu ze strony programu, musiano zaprzestać dalszych prób.

5 Przeprowadzony eksperyment

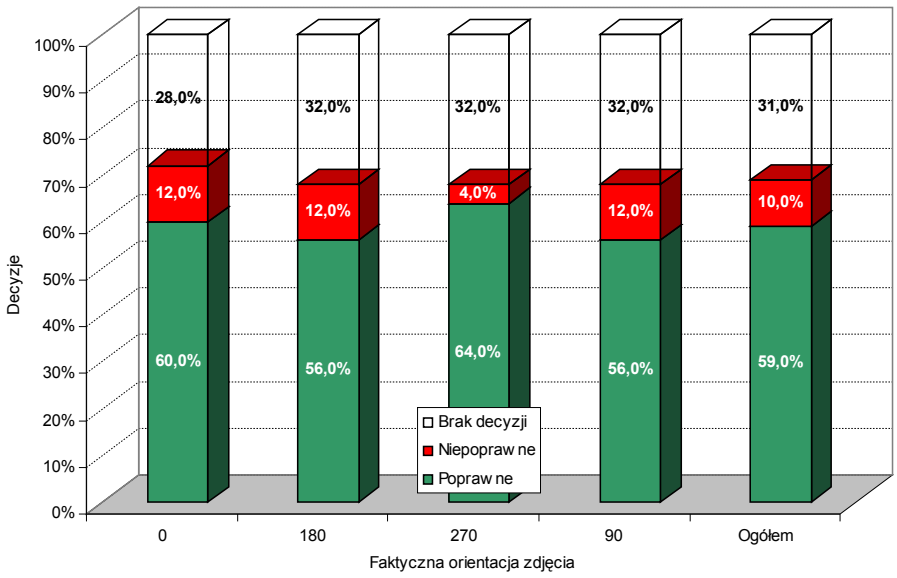
W obliczu niepowodzenia przy konstrukcji własnego klasyfikatora, dokonano testów wykorzystujących klasyfikatory dostarczane wraz z biblioteką OpenCV. Utworzono 3 warianty programu:

- wykorzystujący klasyfikator `haarcascade_frontalface_alt.xml`;
- wykorzystujący klasyfikator `haarcascade_frontalface_al_tree.xml`;
- wykorzystujący klasyfikatory `haarcascade_frontalface_alt.xml` oraz `haarcascade_profileface.xm` podejmujący decyzję o orientacji zdjęcia na podstawie agregacji wyników działania obu klasyfikatorów.

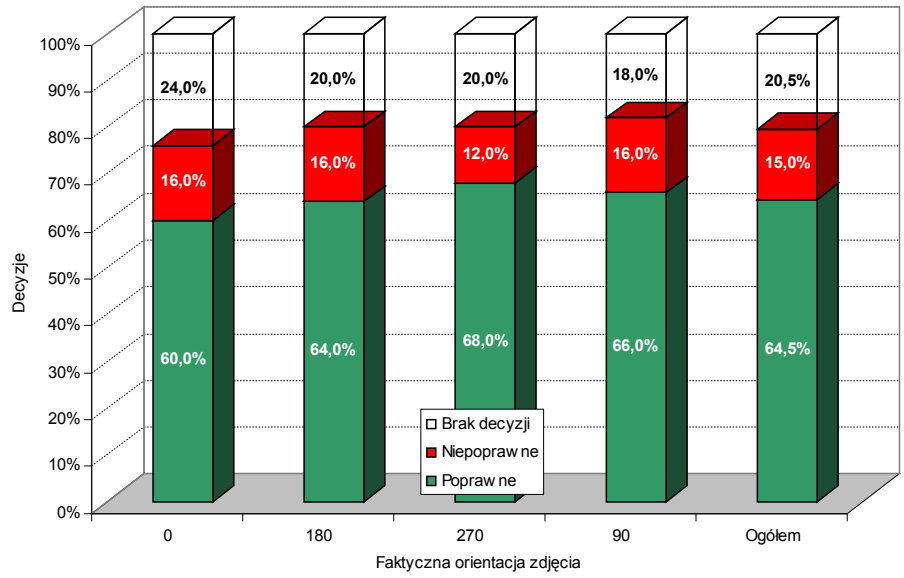
Na potrzeby testów przygotowano 200 zdjęć zawierających twarze, po 50 dla każdej z czterech możliwych orientacji. Następnie uruchomiono każdy z trzech wariantów programu na tym zbiorze oraz zebrano wyniki. Dopuszczono przy tym sytuację w której program powstrzymuje się od decyzji. Wyniki działania poszczególnych wariantów programu zebrano na wykresach 5.1, 5.2 oraz 5.3.

Jak widać, dla wszystkich użytych klasyfikatorów dość duży odsetek stanowią sytuacje, w których program wstrzymuje się od podjęcia decyzji. Pod tym względem najlepiej wypadł klasyfikator złożony (wykrywający twarze ustawione frontalnie oraz profilem), osiągając wyniki o ok. 10% lepsze od dwóch pozostałych. Trafność decyzji w sytuacji gdy zostaje ona podjęta zilustrowano na wykresie 5.4.

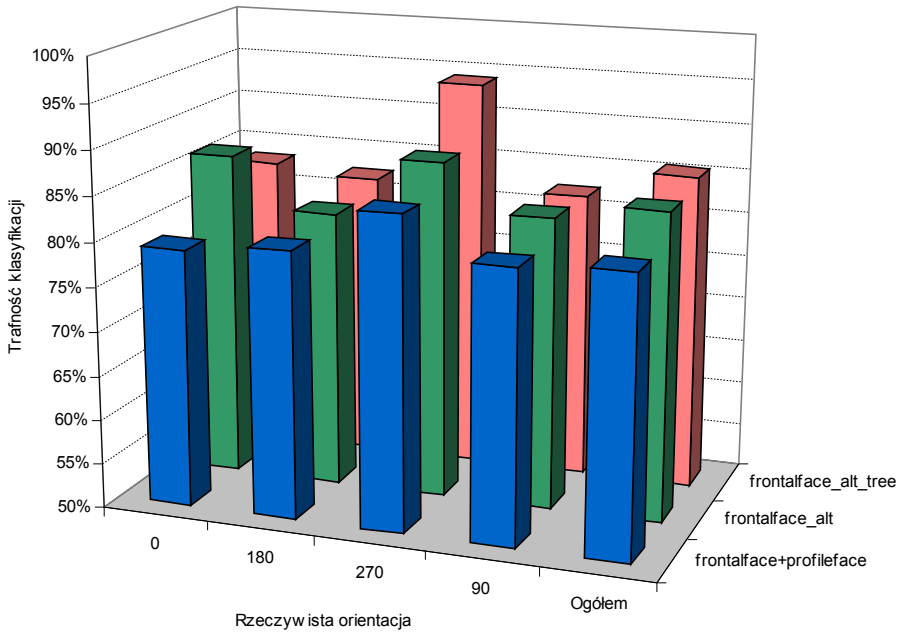
Wynika z niego, że mimo tego iż program wykorzystujący klasyfikator złożony udziela odpowiedzi częściej, to ogólnie cechuje się najsłabszą trafnością klasyfikacji. Najlepiej pod tym



Rysunek 5.2: Wyniki działania dla klasyfikatora `haarcascade-frontalface-alt-tree.xml`.



Rysunek 5.3: Wyniki działania dla klasyfikatora złożonego (front i profil).



Rysunek 5.4: Trafność podejmowanych przez program decyzji.

względem spisuje się klasyfikator `haarcascade_frontalface_alt_tree.xml`, wykorzystujący klaskadę klasyfikatorów o strukturze drzewiastej zamiast liniowej. Różnice w trafności klasyfikacji dla różnych początkowych orientacji zdjęć wynikają ze zróżnicowania zbioru testującego.

Należy jeszcze wspomnieć o czasie działania programu. Z uwagi na swoją specyfikę, klasyfikatory typu *haar-like features* okazały się zdecydowanie zbyt powolne, by można było mówić o praktycznej przydatności stworzonego programu. Czas przetwarzania pojedynczego zdjęcia wynosił ponad 10 sekund, co sprawiało, że łączny czas potrzebny na przetworzenie 200-zdjęciowego zbioru testowego wynosił ponad pół godziny. Jest to zdecydowanie za dużo, jesli wziąć pod uwagę fakt, że wygoda użycia aparatów cyfrowych polega właśnie na możliwości generowania dużej ilości zdjęć. I przecież właśnie duża ich liczba była jedną z głównych motywacji niniejszego projektu.

6 Wnioski i plany na przyszłość

Wykorzystane podejście miało następujące zasadnicze wady:

- binarną informację wyjściową - wykrycie obiektu w danym wycinku obrazu bądź nie – brak informacji o mierze pewności wykrycia obiektu;
- silnie zdyskretyzowaną informację o orientacji wykrytych obiektów – jedna z czterech wartości, niektóre obiekty o pośredniej orientacji w ogóle nie były wykrywane;
- bardzo powolne działanie.

Usunięcie pierwszych dwóch wad mogłoby polegać na zwracaniu przez klasyfikator wartości rzeczywistej z przedziału [0;1] będącej miarą „obecności” poszukiwanego obiektu w danym fragmencie obrazu. Dodatkowo, klasyfikator ten powinien zwracać wartość rzeczywistą kąta określającego orientację znalezionego obiektu. Pozwoliłoby to na bardziej wiarygodne postawienie hipotezy o faktycznej orientacji zdjęcia, przez odsianie fałszywych trafień oraz dokładniejszą informację o orientacji twarzy. Jeśli chodzi o punkt trzeci, spełnienie go jest warunkiem praktycznej przydatności rozważanego narzędzia.

7 Bibliografia

W trakcie projektu korzystano z następujących źródeł informacji:

- *OpenCV – Open Source Computer Vision Library Community* (http://groups.yahoo.com/group/OpenCV/)
- *OpenCV Wiki* (http://opencvlibrary.sourceforge.net/)
- Tony Lindeberg, *Scale-space: A framework for handling image structures at multiple scales* (http://www.nada.kth.se/~tony/cern-review/cern-html/cern-html.html)
- *Image Processing Learning Resources* (http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm)
- Tony S. Jebara , *3D Pose Estimation and Normalization for Face Recognition* (http://www1.cs.columbia.edu/ jebara/htmlpapers/UTHESIS/)
- Sebastien Marcel, *Face Detection Demo* (http://www.idiap.ch/ marcel/en/facedemos.php)