Politechnika Poznańska
Wydział Elektryczny
semestr IV kierunek Informatyka

Dokumentacja projektu
z przedmiotu podstawy teleinformatyki

Temat projektu: Rozpoznawanie twarzy i śledzenie ruchu

Wykonawcy:
Bartłomiej Dura - Nr indeksu:116893
Damian Gołębiowski - Nr indeksu:116892
Wiktoria Leitgeber - Nr indeksu:116933

0. Spis Treści

1. Wstęp

Technologia rozpoznawania człowieka na podstawie obrazu twarzy jest nieinwazyjna, bezkontaktowa i najbardziej naturalna spośród wszystkich metod identyfikacji człowieka. Technologia ta cieszy się coraz większą popularnością, ponieważ wymagania odnośnie bezpieczeństwa stale rosną. Dodatkowo warto dodać, że możliwości wykorzystania jej w praktyce są ogromne.

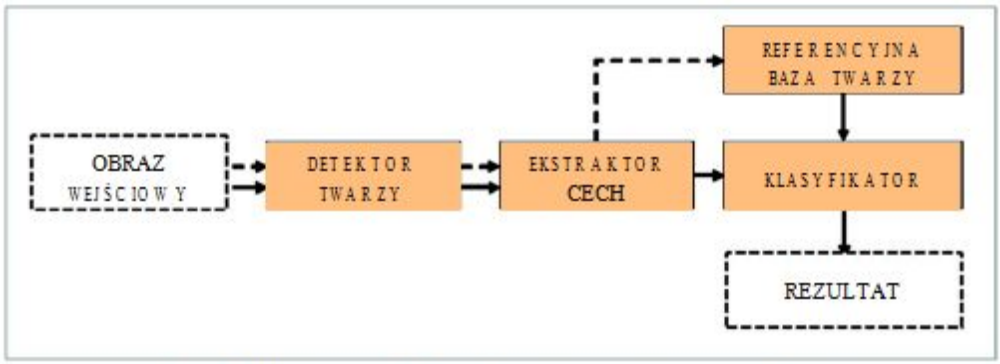
Rozpoznawanie twarzy jest jedną z technologii biometrycznych. Zaliczamy ją do kategorii cech biometrycznych fizycznych. Stosowanie tego rozwiązania niesie za sobą pewne ryzyko, ponieważ stosunkowo łatwo znaleźć osobę mającą podobne rysy twarzy(trzeba również wziąć pod uwagę rodzeństwa będące bliźniętami). Wraz z upływem czasu twarz zmienia się oraz podlega starzeniu się, co wiąże się z dodatkowym ryzykiem uzyskania fałszywie dodatniej identyfikacji.
Do zadań rozpoznawania twarzy realizowanych przez system rozpoznawania twarzy należy weryfikacja i identyfikacja twarzy.

Weryfikacja twarzy zakłada scenariusz kooperatywny. Użytkownik systemu „przedstawia się", a następnie pobierana jest próbka biometryczna (zdjęcie twarzy). Następuje proces porównywania pobranej próbki z wzorcem konkretnej osoby zarejestrowanej w bazie danych systemu. System przyjmuje decyzję o akceptacji, jeżeli wyliczone cechy z pobranej próbki odpowiadają cechom wzorca lub odrzuceniu w przeciwnym przypadku. Mamy do czynienia z porównywaniem typu „1:1" – porównywanie z konkretnym wzorcem.
Biometryczne systemy weryfikacji charakteryzują się wieloma zaletami nad tradycyjnymi rozwiązaniami uwierzytelniania bazującymi na wiedzy (hasła, kody) lub posiadaniu (karty magnetyczne, tokeny, itp.). Tradycyjne systemy mogą być w łatwy sposób oszukane, ponieważ nie gwarantują, że osoba posługująca się dla przykładu kartą kredytową i znająca PIN jest prawowitym właścicielem posiadanej karty. Biometryczne systemy weryfikacji znacznie trudniej jest oszukać. Sprawdzenie użytkownika pod względem tego „co ma?" lub „co wie?" można zastąpić sprawdzeniem „kim naprawdę jest?".

Identyfikacja twarzy (identyfikacja na podstawie obrazu twarzy) polega na podaniu tożsamości lub innych szczegółów powiązanych bezpośrednio z twarzą (np. „zdjęcie przedstawia twarz Jana Kowalskiego"). Podczas identyfikacji używa się porównań „1 do wielu" („1:N") – jedna twarz (jeden obraz twarzy) jest porównywany z innymi (wieloma) twarzami (obrazami twarzy – wzorcami).

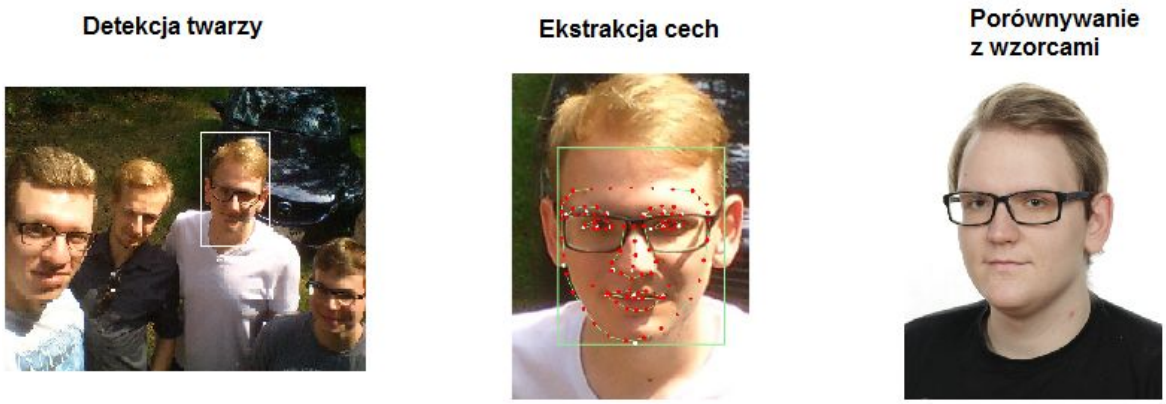
Wyróżniamy dwie podstawowe grupy w metodach rozpoznawania twarzy. Pierwsze wykorzystujące cechy oraz drugie, nazywane holistycznym- oparte na wyglądzie.

Nasza aplikacja będzie korzystała z drugiego wariantu rozpoznawania twarzy, gdzie istotne są globalne właściwości wzorca twarzy. Podejście te operują bezpośrednio na intensywności pikseli(macierzy), bez wykonywania detekcji cech twarz. Polegają one na transformacji dwuwymiarowego obrazu twarzy w jednowymiarową przestrzeń cech- wektor cech. Modele 2D wiążą się z ryzykiem oszustwa systemu, dlatego nowoczesne systemy rozpoznawania twarzy coraz częściej wykorzystują informację przestrzenną. Obejmują one podejścia wykorzystujące modele 3D przedniej części głowy. Informacja z trzeciego wymiaru może wpływać na skuteczność systemu rozpoznawania twarzy. Tę informację można uzyskać z mapy głębokości obrazu twarzy (stosować można specjalne kamery, specjalistyczne oświetlenie rzucające siatkę lub zestaw dwóch kamer). Innym rozwiązaniem jest analityczne tworzenie trójwymiarowego obrazu twarzy na podstawie zestawu dwuwymiarowych obrazów twarzy lub pojedynczego obrazu twarzy i algorytmu rekonstrukcji 2D do 3D.



Rys. 1. Bazowa struktura systemu rozpoznawania twarzy

Na obrazie wejściowym w pierwszym etapie należy dokonać detekcji twarzy – określenia czy i gdzie znajdują się twarze. Detekcja twarzy poprzedza etapy ekstrakcji cech i klasyfikacji – składające się na właściwe rozpoznawanie twarzy. Przy czym, ekstraktor cech odpowiada za wydobycie istotnej informacji z obrazu twarzy, natomiast klasyfikator za porównywanie reprezentacji odpowiadającej obrazowi wejściowemu z reprezentacjami wzorców, zapisanymi referencyjnej bazie twarzy.



Rys. 2 Wizualizacja podstawowych założeń projektowych

2. Harmonogram i podział pracy

Zadania:	Podzadania:	Osoby:			T1	T2	T3	T4	T5	T6	T7	T8
		Dura:	Gołębiowski:	Leitgeber:								
Wstęp do	Podział na grupy	X	X	X								

projektu	projektowe											
	Wybór tematu projektu	X	X	X								
	Zreferowanie projektu	X	X	X								
Stworzenie interfejsu graficznego	Projekt interfejsu graficznego			X								
Stworzenie funkcji do detekcji twarzy	Wczytywanie obrazu z pliku oraz bezpośrednio z kamery		X									
Klasyfikator kaskadowy	Wytrenowanie klasyfikatora kaskadowego	X										
Wektora kwadrató	Stworzenie wektora		X									

w	kwadratów											
Konwersja odcieni szarości	Konwersja obrazów do odcieni szarości			X								
Wykrywanie elementów twarzy	Algorytm wykrywania twarzy i funkcje wykrywające poszczególne elementy twarzy	X	X	X								
Stworzenie bazy danych z twarzami	Importowanie bazy do programu w postaci pliku csv		X									
Stworzenie skryptu do obsługi	Implementacja w Python	X										

bazy danych												
Testy	Wnioski i podsumowanie	X	X	X								

Tab. 1 Spis i podział zadań do wykonania oraz terminy ich realizacji

3. Uzasadnienie wyboru tematu

Będąc na specjalności Bezpieczeństwo Sieci Informatycznych chcieliśmy podjąć się realizacji projektu, który pozwala poznać nowoczesne technologie, z którymi nie mieliśmy wcześniej styczności. System rozpoznawania twarzy może w przyszłości również posłużyć do autoryzacji w innych naszych projektach. Jest to temat ciekawy, a jego realizacja pozwoli nam zagłębić się w temacie biometrii.

4. Wykorzystane technologie

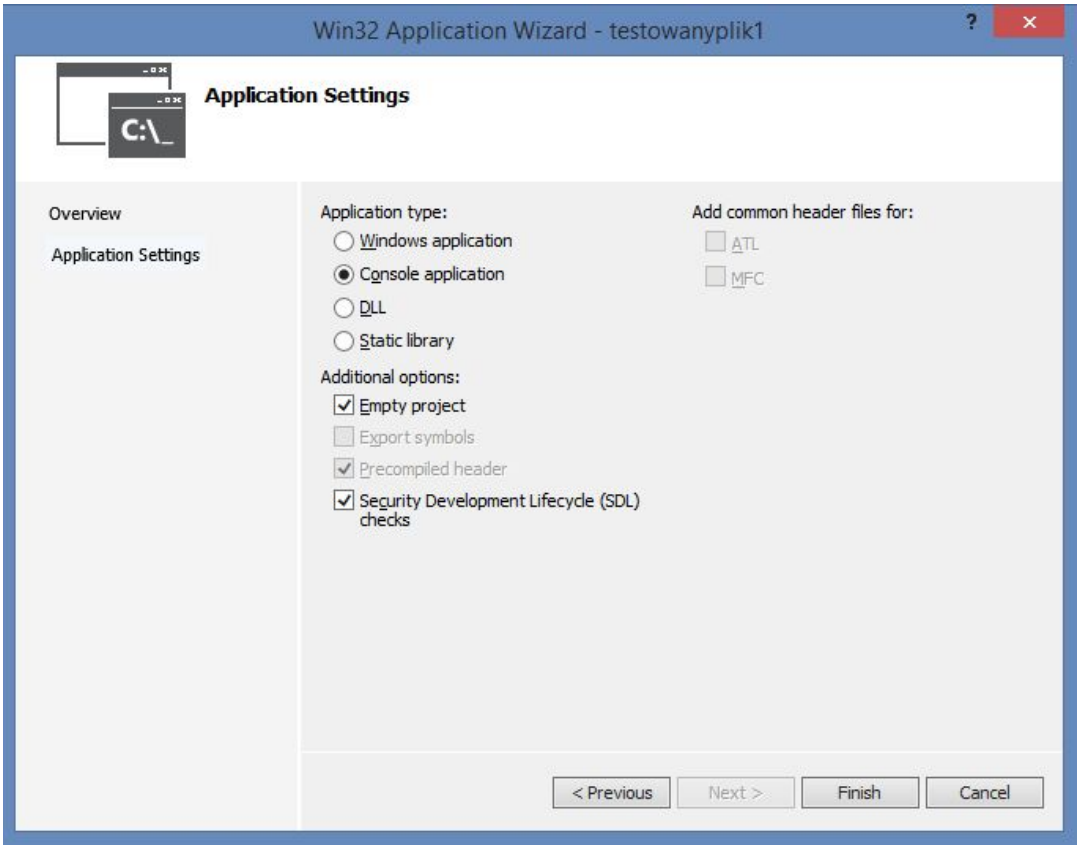
- Microsoft Visual Studio 2013,
- C# .NET,
- Python,
- OpenCV 2.4.10 z rozszerzeniem Emgu CV

.

5. Instalacja i konfiguracja OpenCV 2.4.10

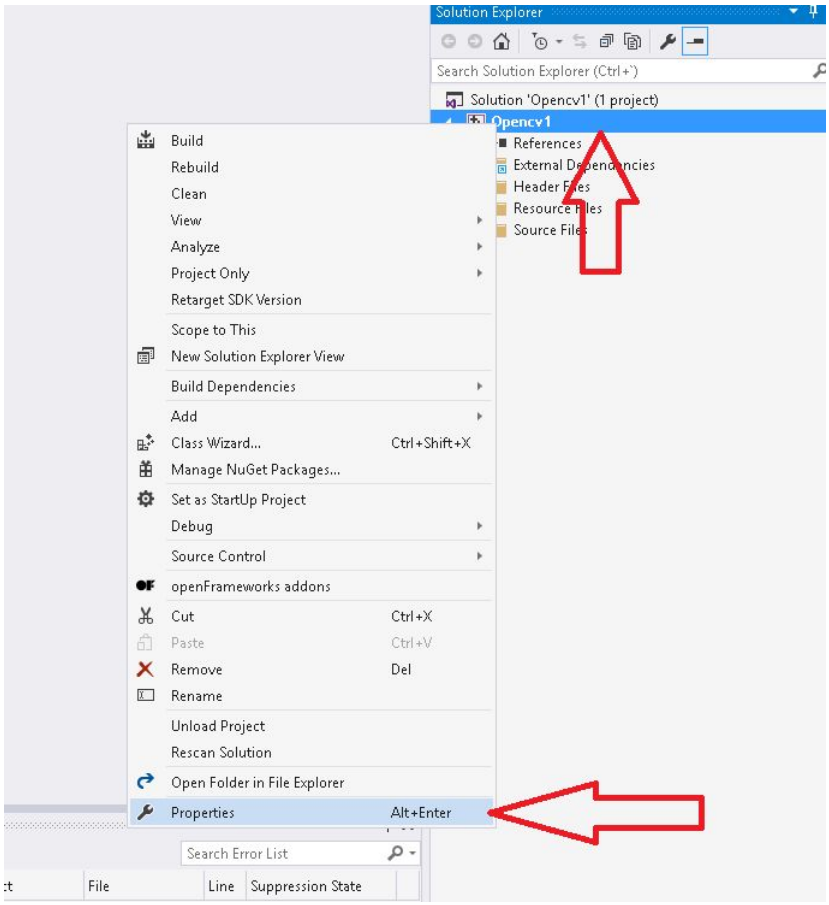
- Pobieramy OpenCV 2.4.10 https://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.10/opencv-2.4.10.exe/download
- Tworzymy nowy folder na dysku o nazwie C:\OpenCV2410, do którego rozpakowujemy pobrany wcześniej plik(jest to wykonywalny, który po otwarciu sam rozpocznie rozpakowywanie, musimy tylko wskazać lokalizację).

- Teraz uruchamiamy Visual Studio 2013 i tworzymy nowy projekt(Win 32 Console Application), wybieramy empty project.



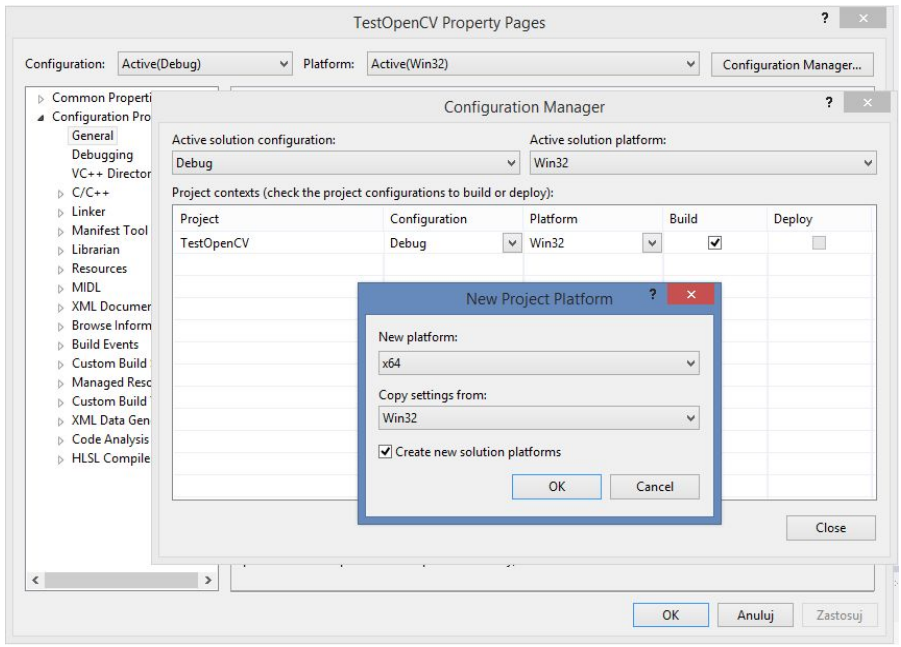
Rys. 3 Tworzenie projektu

- Do nowego projektu dodajemy plik o rozszerzeniu .cpp.(Source Files->Add->New Item)
- Klikamy prawym przyciskiem na nazwę naszego projektu widoczną na pasku Solution Explorer i wybieramy zakładkę Properties



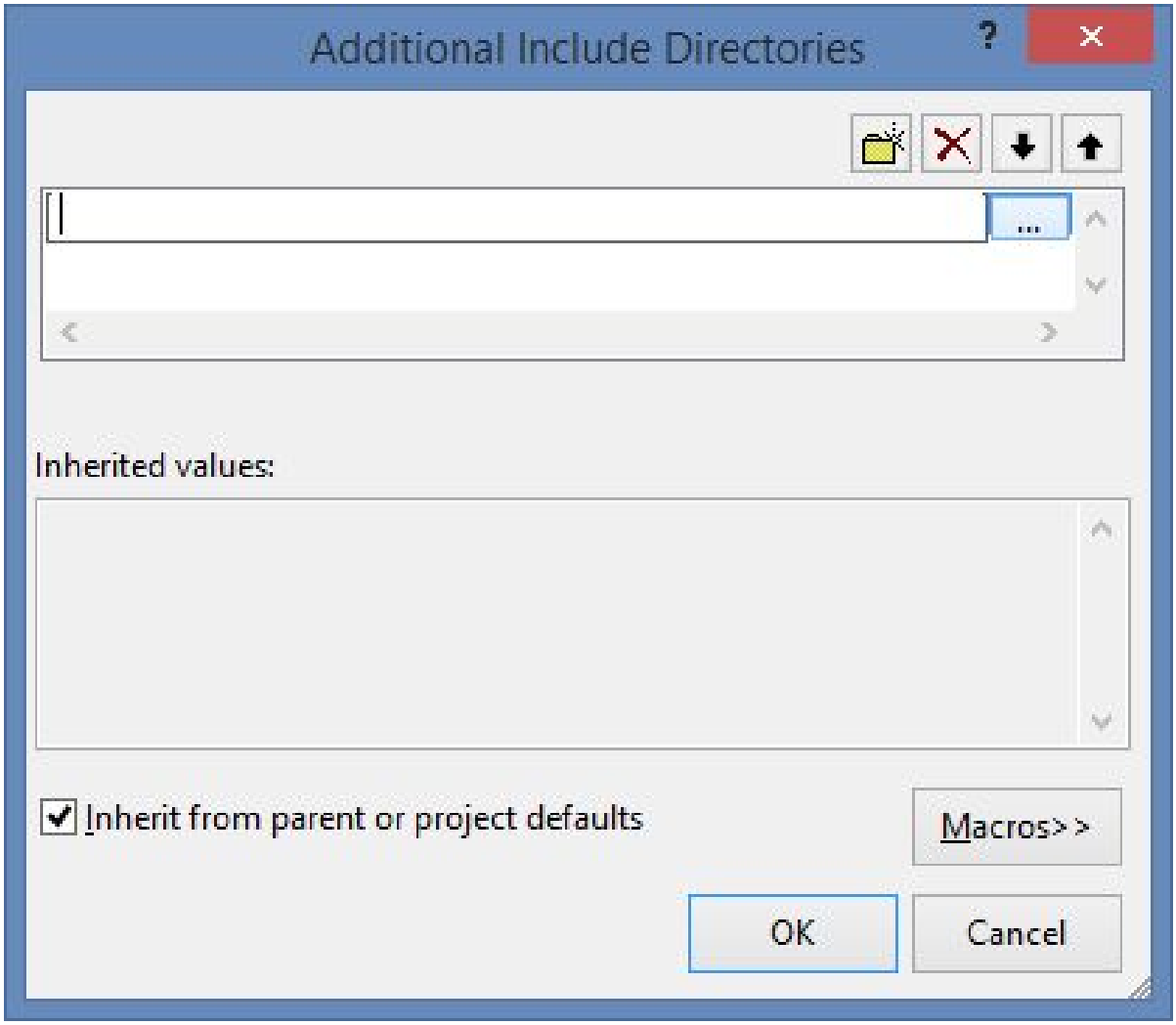
Rys. 4 Otwieranie ustawień projektu

- Zmieniamy platformę z Win32 na x64 używając do tego Configuration Manager i kopiujemy ustawienia z Win32



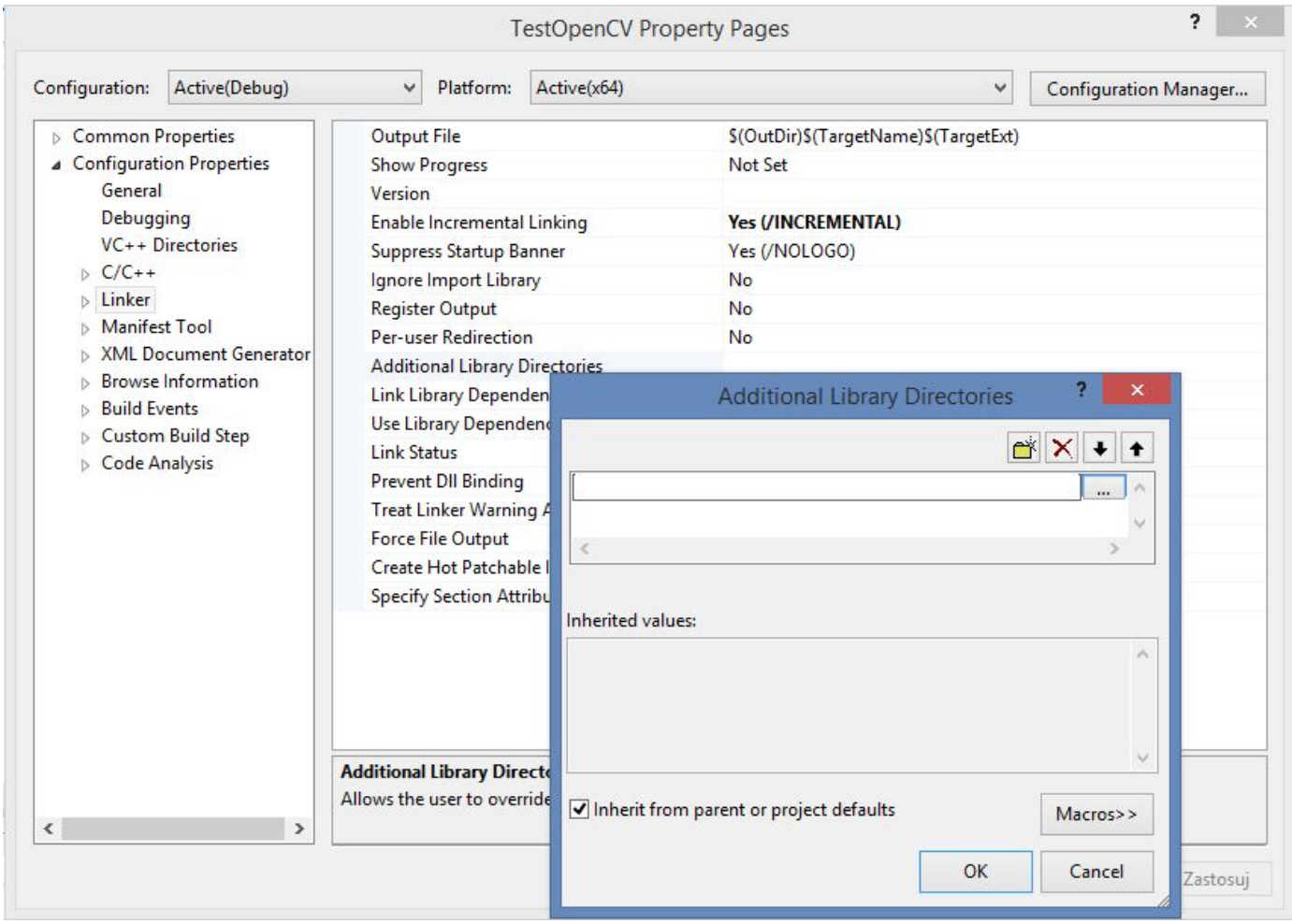
Rys. 5 Konfiguracja projektu

- Przechodzimy do zakładki C/C++ --> General --> Additional Include Directories, wyszukujemy wcześniej podaną lokalizację i wskazujemy ścieżki do folderów:
    - C:\OpenCV2410\opencv\build\include
    - C:\OpenCV2410\opencv\build\include\opencv
    - C:\OpenCV2410\opencv\build\include\opencv2



Rys. 6 Konfiguracja projektu

- Przechodzimy do zakładki Linker --> General --> Additional Library Directories i podajemy ścieżkę: C:\OpenCV2410\opencv\build\x64\vc12\lib



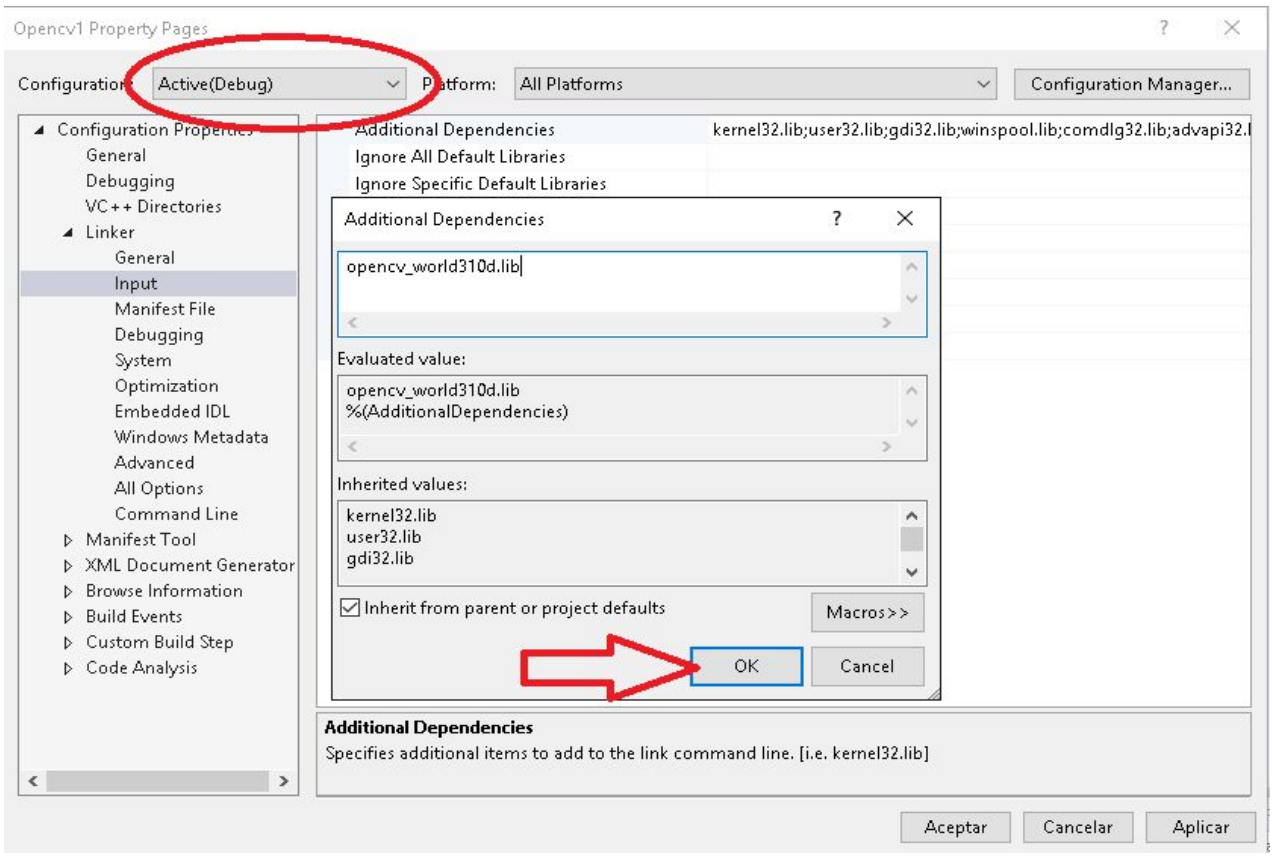
Rys. 7 Konfiguracja projektu

- Przechodzimy do zakładki Linker --> Input --> Additional Dependencies (podajemy wszystkie pliki z lokalizacji C:\OpenCV2410\opencv\build\x64\vc12\lib\ kończące się na d).

Dla ułatwienia podaje te pliki:

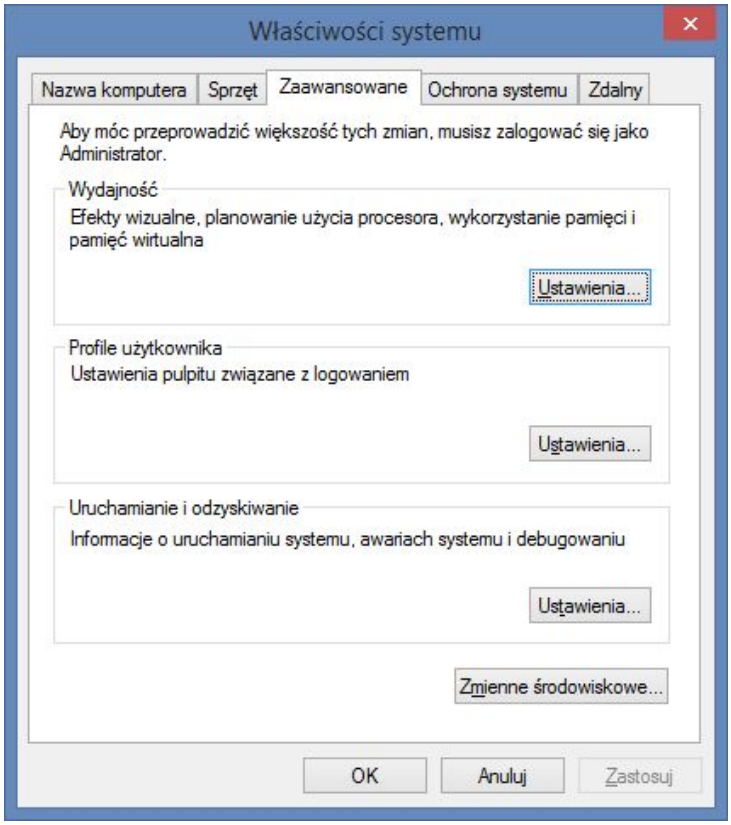
- opencv_calib3d2410d.lib
- opencv_contrib2410d.lib
- opencv_core2410d.lib
- opencv_features2d2410d.lib
- opencv_flann2410d.lib
- opencv_gpu2410d.lib
- opencv_highgui2410d.lib

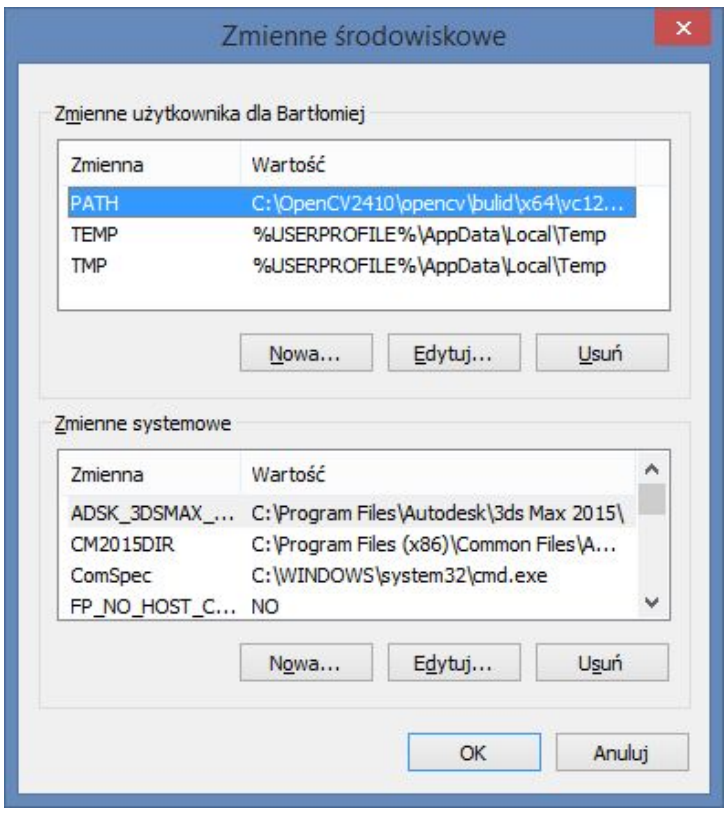
- opencv_imgproc2410d.lib
- opencv_legacy2410d.lib
- opencv_ml2410d.lib
- opencv_nonfree2410d.lib
- opencv_objdetect2410d.lib
- opencv_ocl2410d.lib
- opencv_photo2410d.lib
- opencv_stitching2410d.lib
- opencv_superres2410d.lib
- opencv_ts2410d.lib
- opencv_video2410d.lib
- opencv_videostab2410d.lib



Rys 8. Konfiguracja projektu

- Dodajemy ścieżkę opencv do zmiennych środowiskowych C:\OpenCV2410\opencv\build\x64\vc12\bin



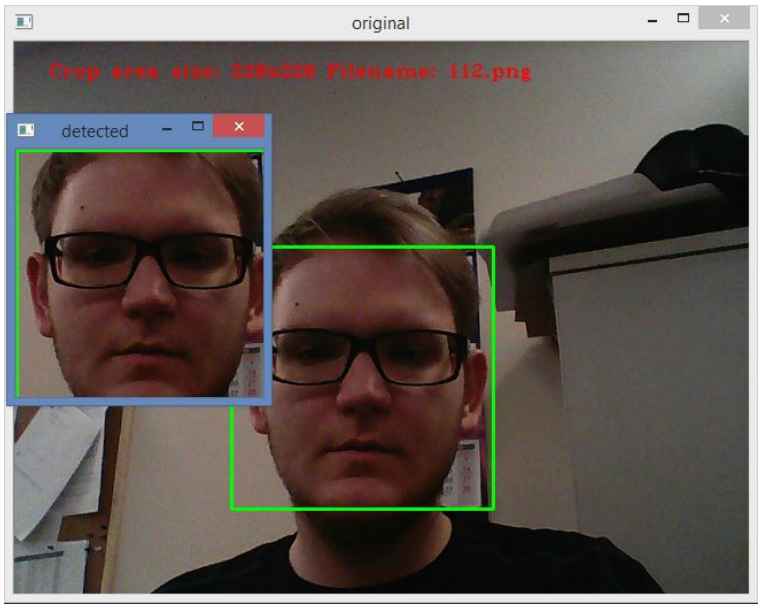


Rys. 9 Ustawianie zmiennych środowiskowych

- Przykładowy film z instalacją: https://www.youtube.com/watch?v=e_TQ9c3n_d8

6. Implementacja

➢ Iteracja pierwsza zawiera wykrywanie twarzy i śledzenie jej ruchów oraz zapisywanie klatek do pliku, które posłużą jako baza danych dla systemu rozpoznawania twarzy.



Rys 10. Detekcja twarzy

- Kilka przykładowych klatek zapisanych w skali szarości przez program

Rys. 11 Zapisane przez program klatki przechwytywane z kamery

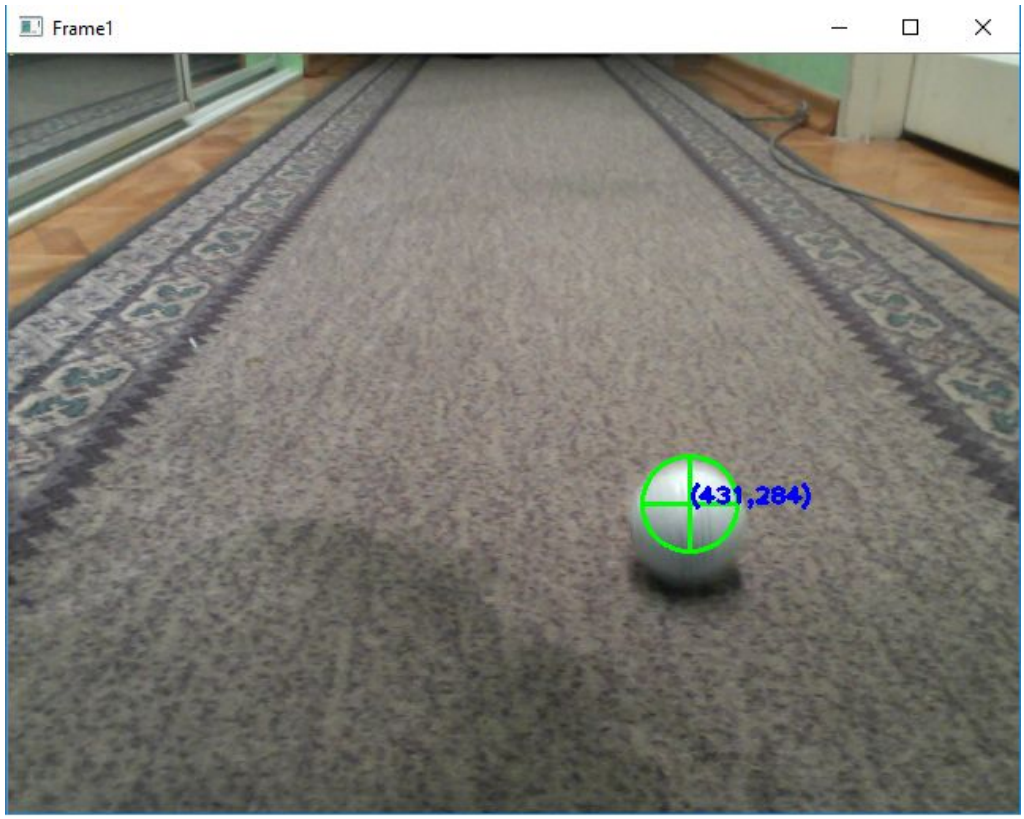
- Link do repozytorium: https://github.com/Golebiowski/podstawy_teleinformatyki
- Link do filmu prezentującego 1. iterację: https://youtu.be/hChaNNZQTBs

➢ Iteracja druga zawiera śledzenie ruchu

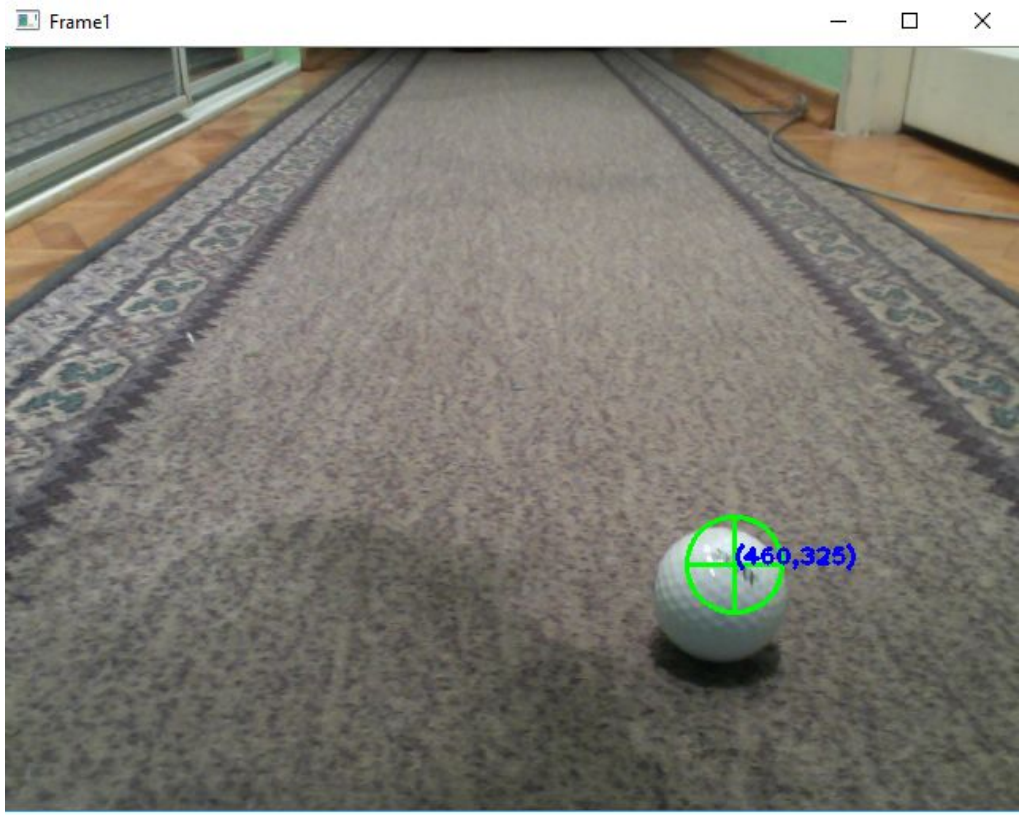
Jedna piłeczka (tracking mode):

Rys. 12 Detekcja ruchu piłki golfowej turlanej po podłodze

Rys. 13 Detekcja ruchu piłki golfowej toczonej po podłodze

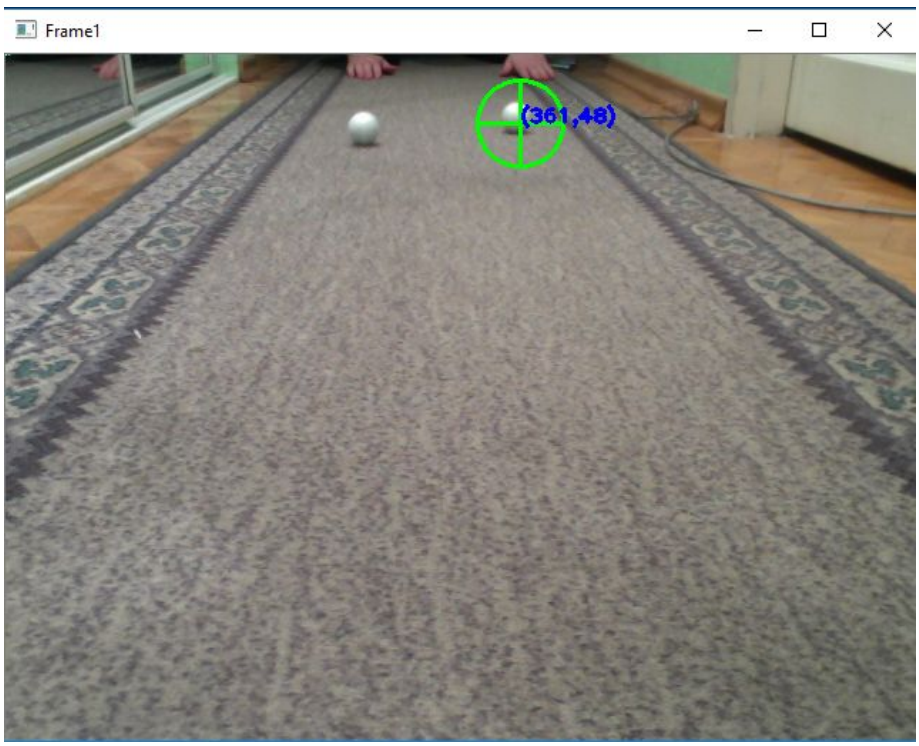


Rys. 14 Detekcja ruchu piłki golfowej turlanej po podłodze

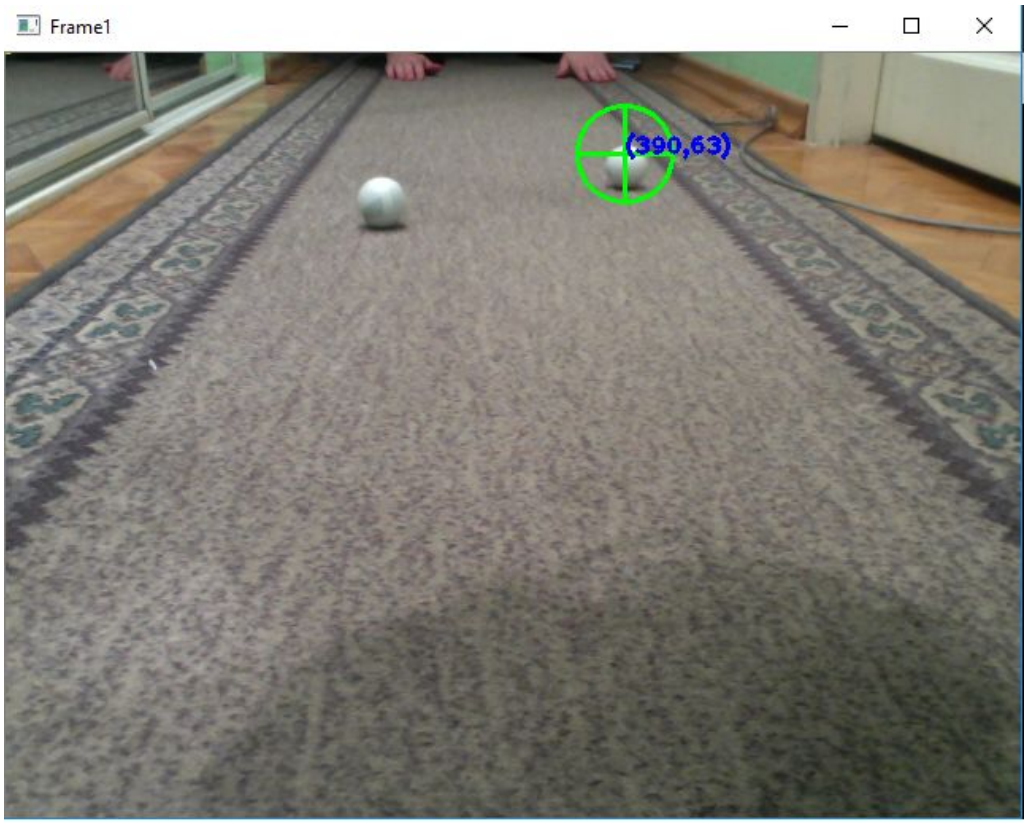


Rys. 15 Detekcja ruchu piłki golfowej turlanej po podłodze

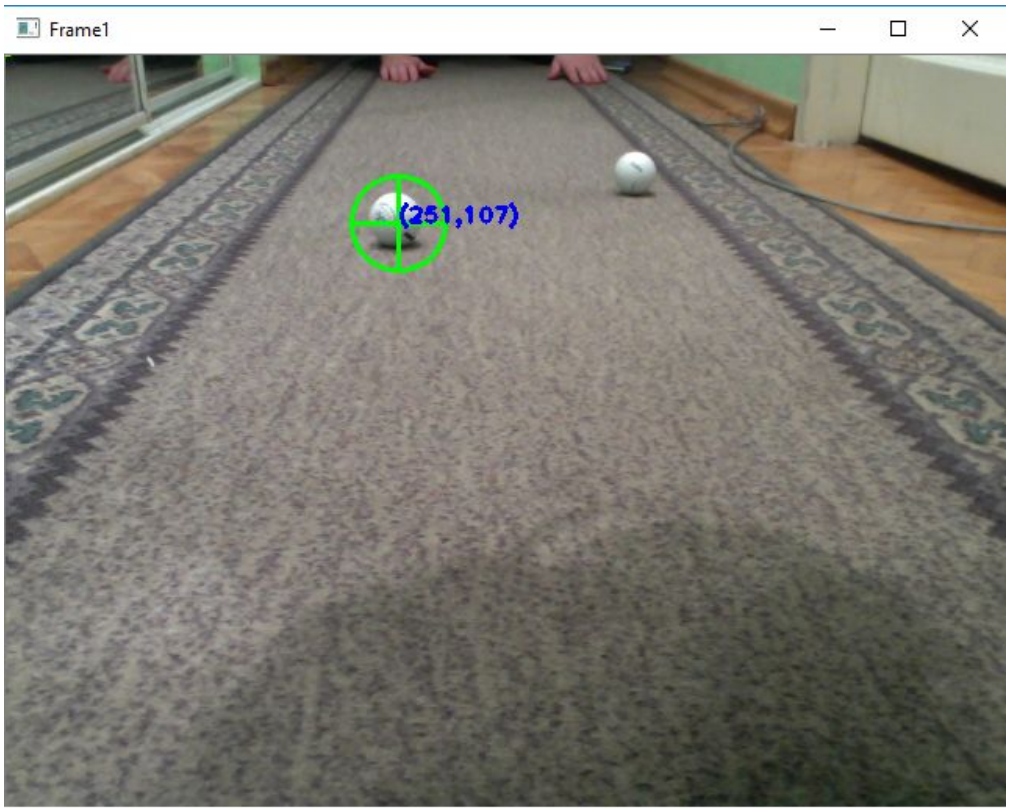
Dwie piłeczki naraz (tracking mode):



Rys. 16 Detekcja ruchu piłki golfowej turlanej po podłodze

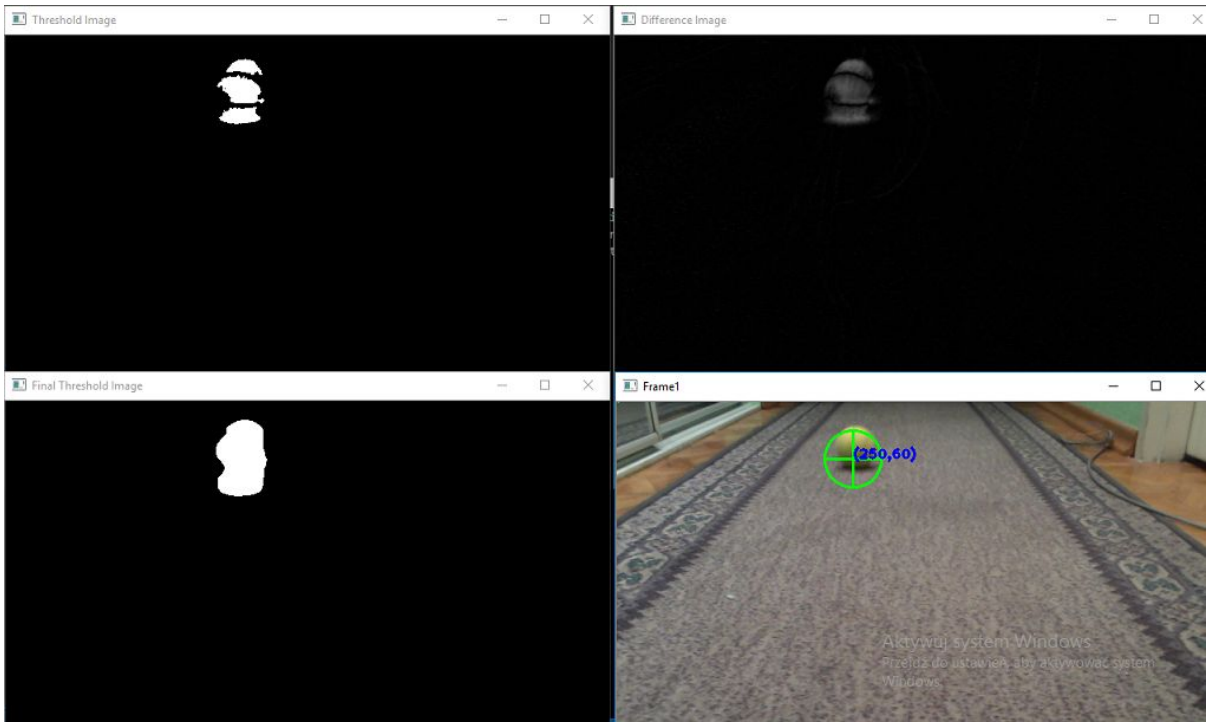


Rys. 17 Detekcja ruchu piłki golfowej turlanej po podłodze

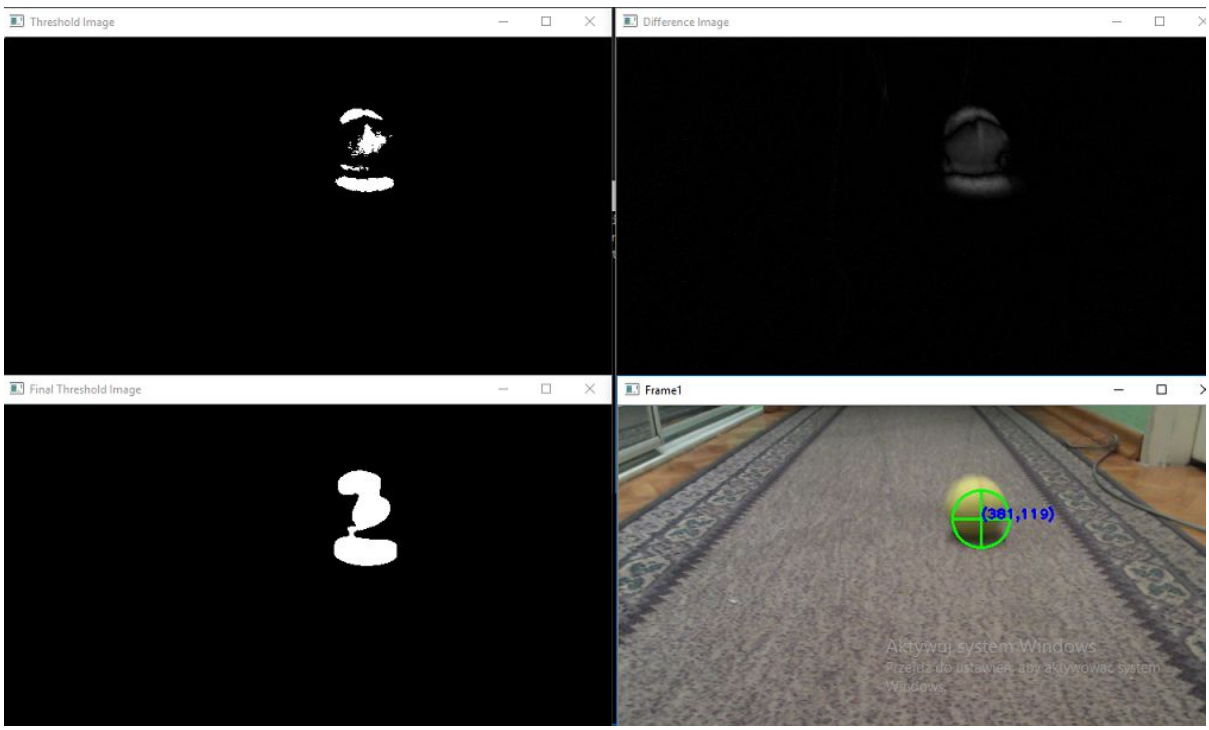


Rys. 18 Detekcja ruchu piłki golfowej turlanej po podłodze

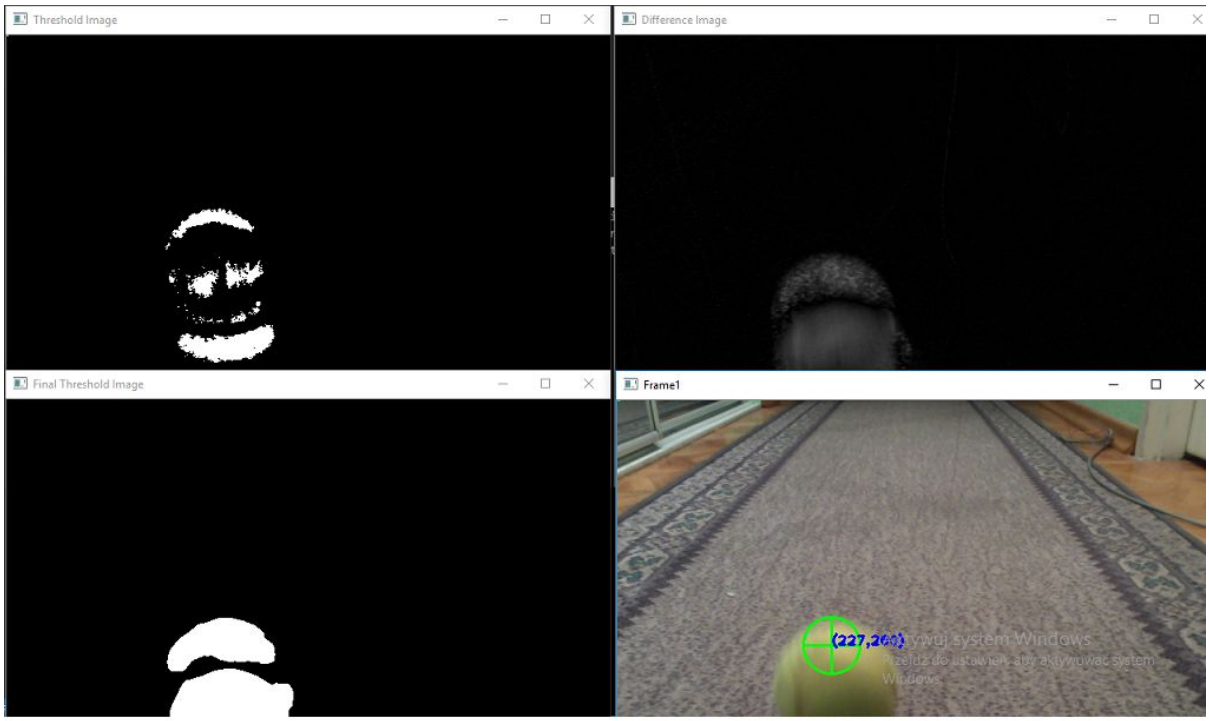
Debug mode:



Rys. 20 Wizualizacja działania metody kontrastu wykorzystywanej do śledzenia ruchu

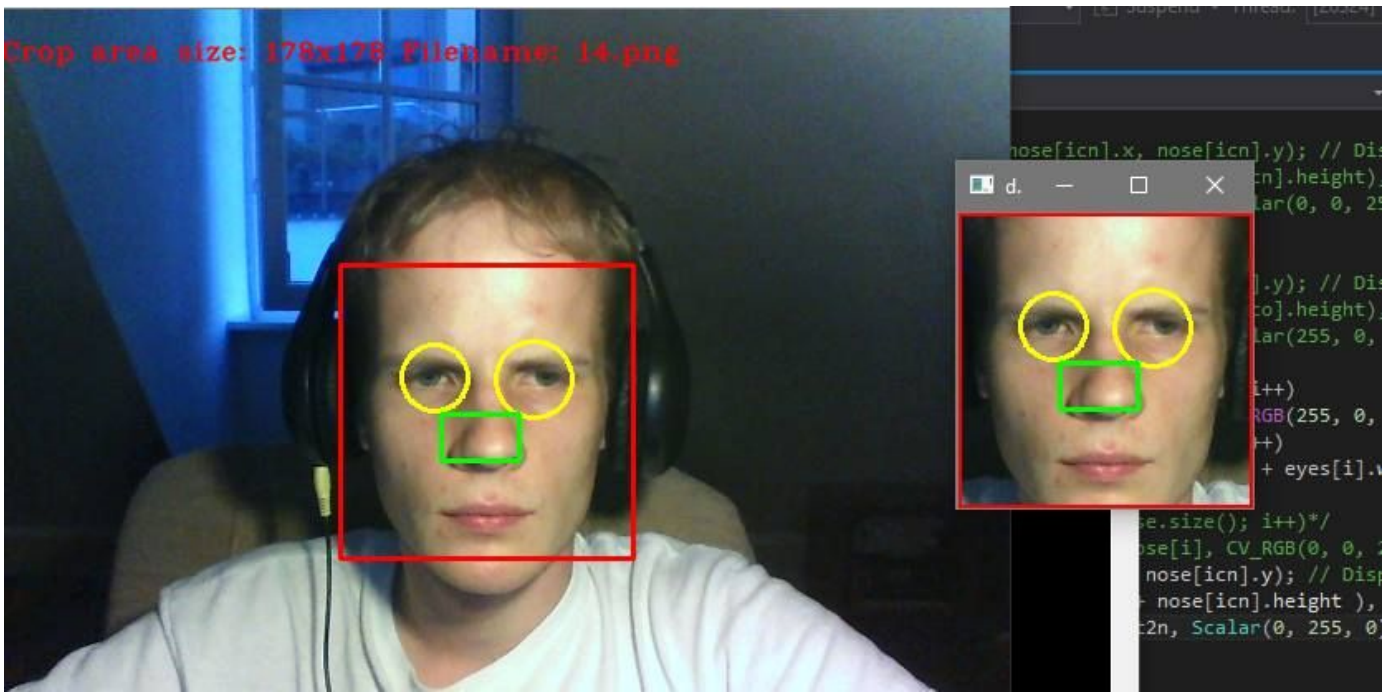


Rys. 21 Wizualizacja działania metody kontrastu wykorzystywanej do śledzenia ruchu



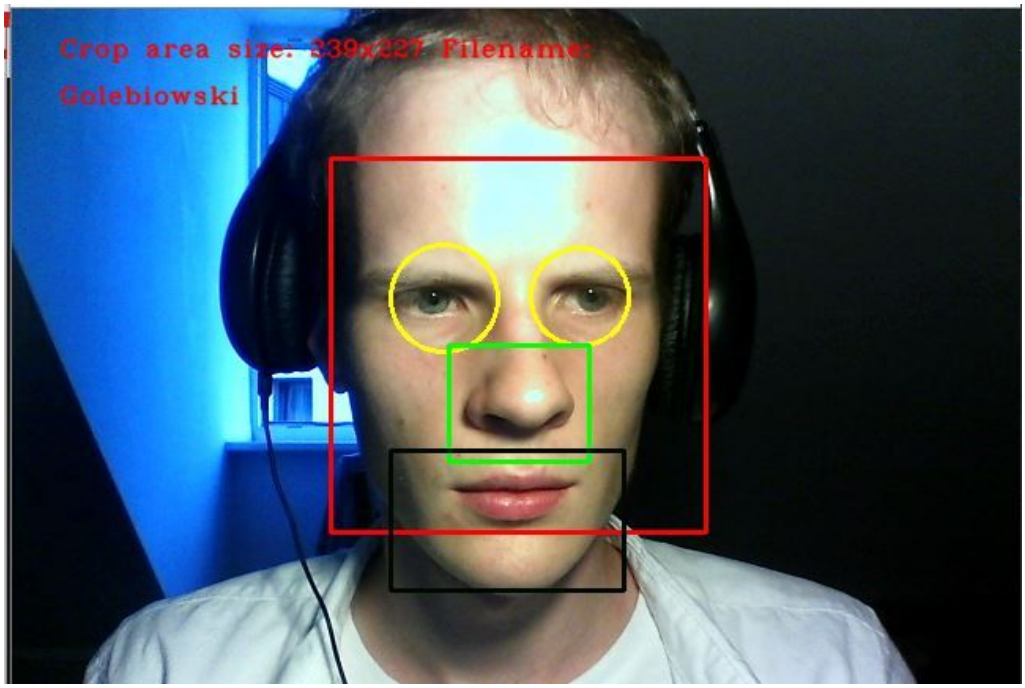
Rys. 22 Wizualizacja działania metody kontrastu wykorzystywanej do śledzenia ruchu

➢ iteracja trzecia zawiera moduł wykrywający twarz:

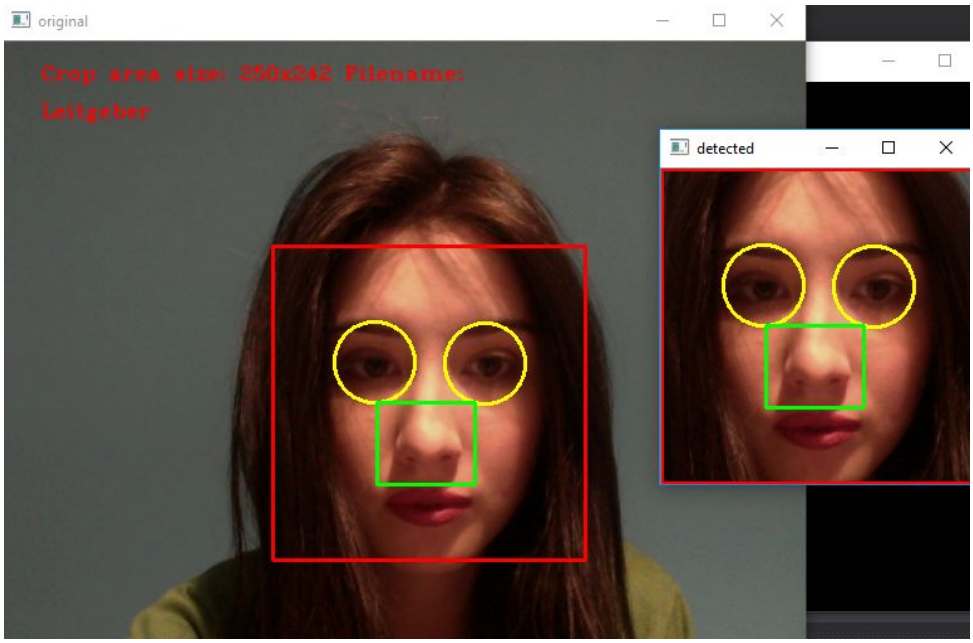


Rys. 23 Wykrywanie poszczególnych elementów twarzy

➢ iteracja czwarta zawiera rozpoznawanie twarzy:



Rys. 24 Identyfikacja osoby której twarz została wykryta



Rys. 25 Identyfikacja osoby której twarz została wykryta

7. Przypadki użycia oraz diagramy UML

- **Uruchomienie programu:**

Główny scenariusz:

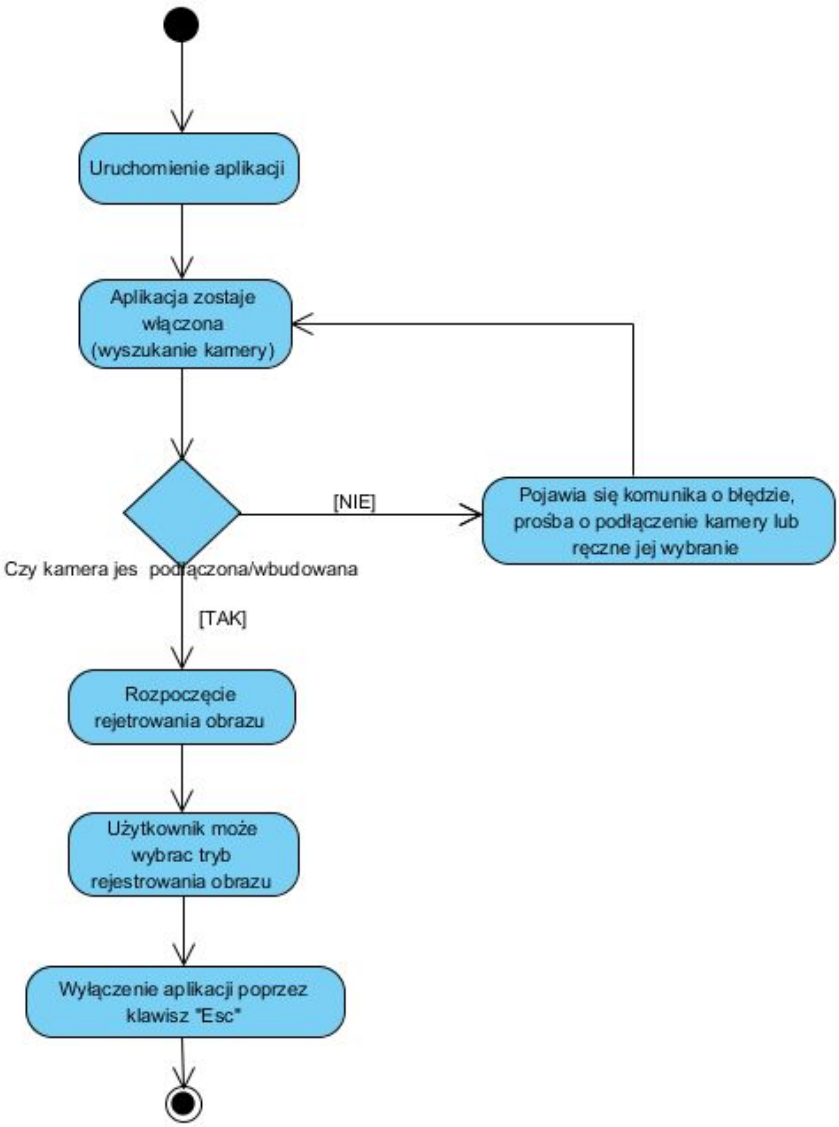
1. Uruchomienie aplikacji.
2. Automatycznie zostaje włączona kamera.
3. Rozpoczyna się rejestrowanie obrazu.
4. Użytkownik aplikacji może wybrać tryb rejestrowania obrazu.
5. Aby zakończyć działanie aplikacji użytkownik naciska klawisz Esc.

Rozszerzenia:

2.1 Brak wbudowanej lub zewnętrznej kamery.

- Pojawi się komunikat o błędzie dotyczącym niewskazaniu domyślnej kamery.

Diagram aktywności:



- **Uruchomienie trybu debug:**

Główny scenariusz:

1. Podczas gdy aplikacja jest włączona, użytkownik może włączyć tryb debug, w tym celu należy nacisnąć klawisz z literą “d”.

2. Użytkownik może zatrzymać (klawisz “p”) lub odwołać działanie trybu (klawisz “d”).

Rozszerzenia:

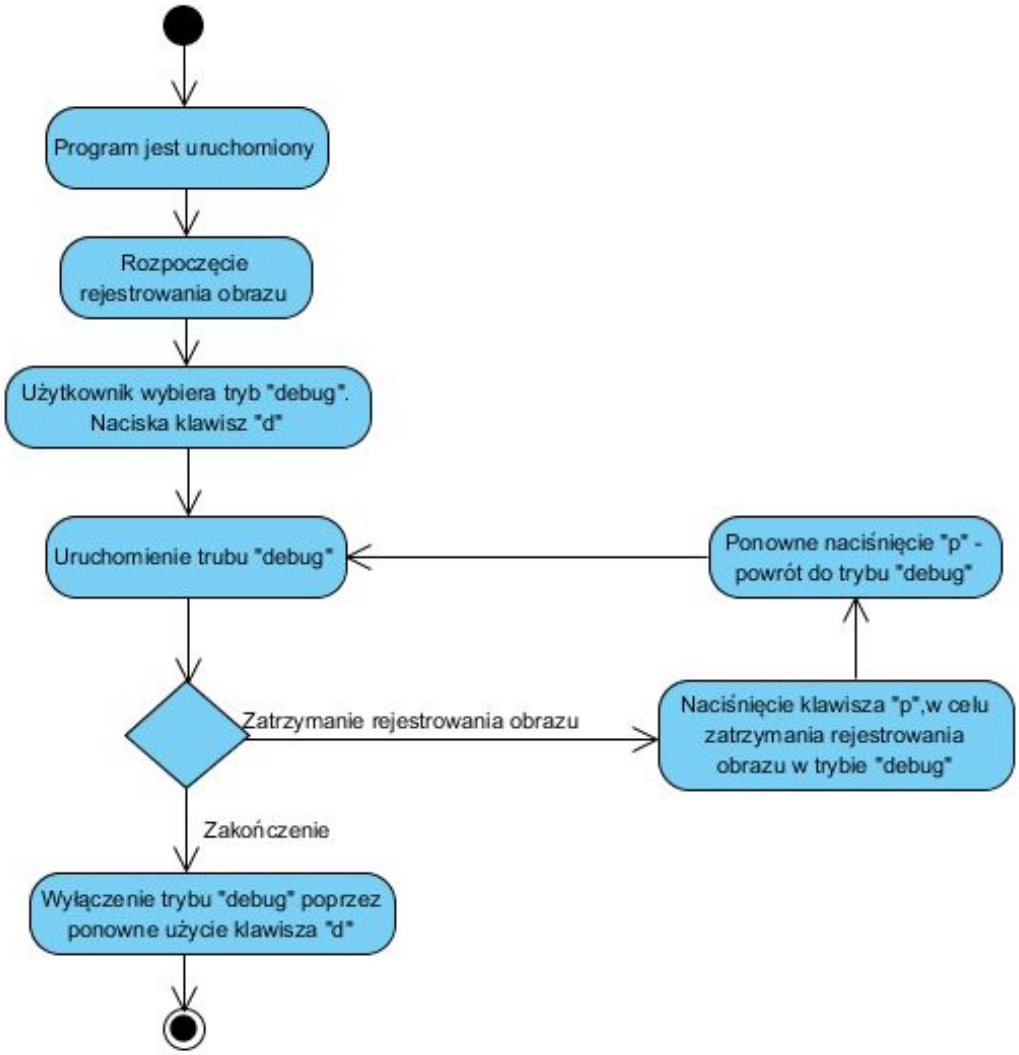
2.1 Tryb debug nie był włączony.

- Próba odwołania działania trybu nie powiedzie się. Komunikat poinformuje o nieaktywnym trybie.

2.2 Włączony jest tryb tracking.

- Komunikat o nieprawidłowej operacji.

Diagram aktywności:



- **<u>Uruchomienie trybu tracking:</u>**

<u>Główny scenariusz:</u>

1. Uruchomienie trybu *tracking*, poprzez zastosowanie klawisza “t”.

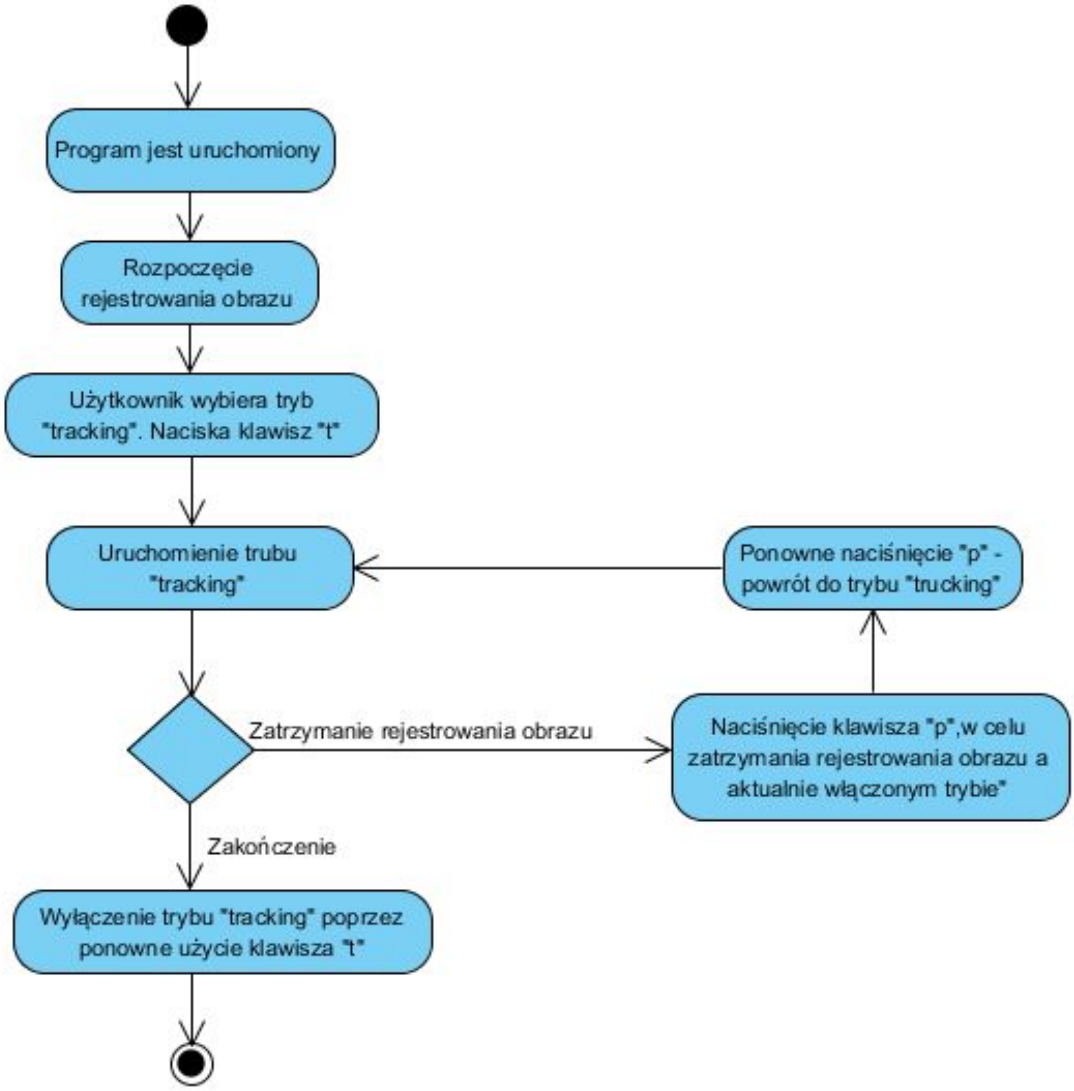
2. Aplikacja pokazuje pozycje obiektu na ekranie wraz ze współrzędnymi (śledzenie ruchomego obiektu).
3. Wyłączenie trybu tracking, poprzez ponowne zastosowanie klawisza “t”.

Rozszerzenia:

3.1 Tryb debug nie był włączony.
- Próba odwołania działania trybu nie powiedzie się. Komunikat poinformuje o nieaktywnym trybie.

3.2 Włączony jest tryb tracking.
- Komunikat o nieprawidłowej operacji.

Diagram aktywności:



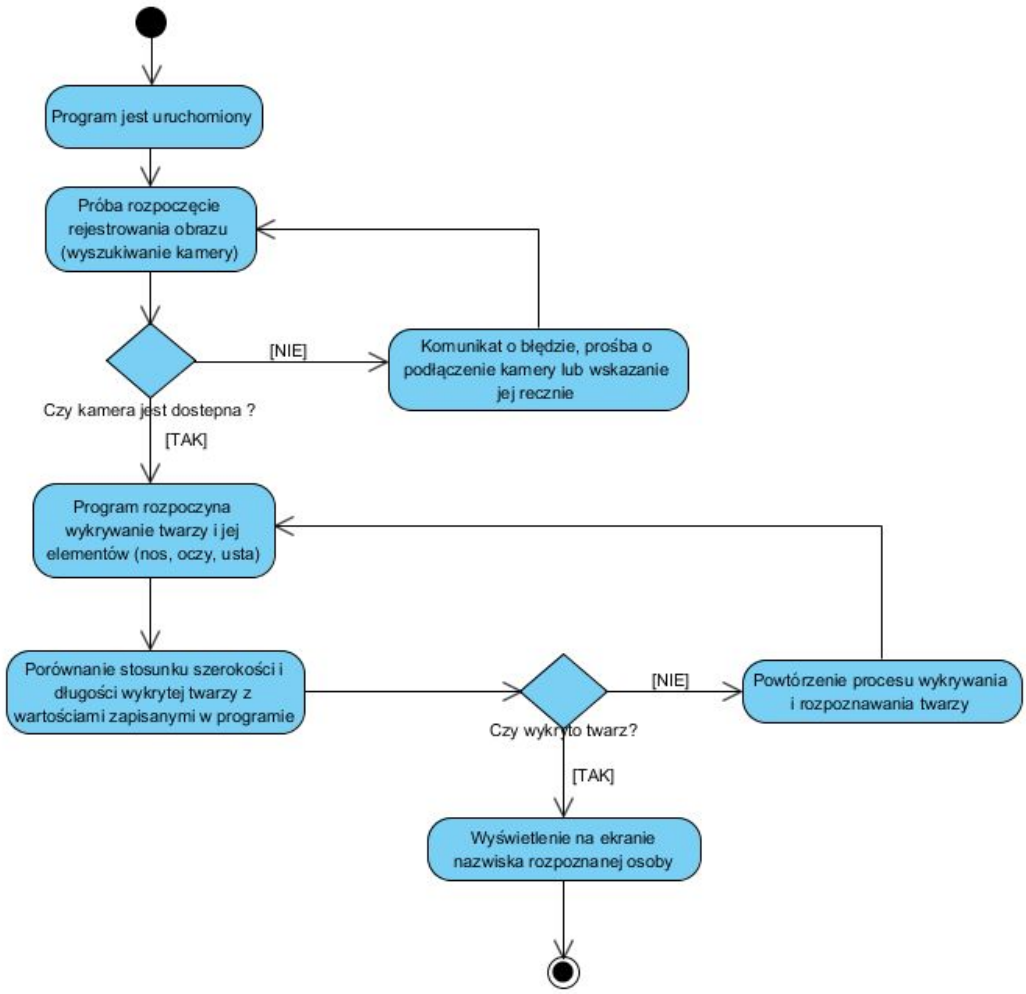
- **<u>Wykrywanie i rozpoznawanie twarzy:</u>**

  <u>Główny scenariusz:</u>

  1. Uruchomienie programu.

2. Automatyczne włączenie kamery i rozpoczęcie rejestrowania obrazu. Program wykrywa twarz, oczy, nos i usta. Na podstawie stosunku długości boków prostokąta wyznaczającego twarz rozpoznawana jest osoba przed kamerą.
3. Na ekranie zostanie wyświetlone nazwisko rozpoznanej osoby, po wcześniejszym porównaniu parametrów.

Diagram aktywności:



8. Najważniejsze fragmenty kodu

I. Wykrywanie twarzy

```
string nazwisko1 = "Golebiowski";
double wspGolebiowski = 0.9359;
double widthFace = (double)faces[ib].width;
double heightFace = (double)faces[ib].height;
Factor = heightFace / widthFace;
            if (Factor > wsp - (wsp*0.005) && Factor > wsp +
(wsp*0.005))
            {
                  sstm1 << nazwisko1;
                  text1 = sstm1.str();
                  putText(frame, text1, cvPoint(30, 60),
                  FONT_HERSHEY_COMPLEX_SMALL, 0.8, cvScalar(0,
0, 255), 1, CV_AA);
            }

string nazwisko2 = "Leitgeber";
double wspLeitgeber = 0.9367;
double widthFace = (double)faces[ib].width;
double heightFace = (double)faces[ib].height;
Factor = heightFace / widthFace;
            if (Factor > wsp - (wsp*0.005) && Factor > wsp +
(wsp*0.005))
            {
                  sstm1 << nazwisko2;
                  text1 = sstm1.str();
                  putText(frame, text1, cvPoint(30, 60),
                  FONT_HERSHEY_COMPLEX_SMALL, 0.8, cvScalar(0,
0, 255), 1, CV_AA);
            }
```

Z obiektu faces poprzez wywołanie funkcji długość i szerokość obiektu cv::Rect pobrane zostają wartości opisujące wymiary wykrytej twarzy. Obliczany zostaje współczynnik dla wykrytego obiektu i jeżeli współczynnik ten zgadza się z wartością oczekiwaną dla współczynnika określonej osoby na ekranie zostaje wyświetlone jej nazwisko. Dopuszczalny błąd pomiaru przyjmujemy jako 0,5%

II. Śledzenie ruchu

- Wykrywanie ruchu

```
void searchForMovement(Mat thresholdImage, Mat &cameraFeed)
{
     bool objectDetected = false;
     Mat temp;
     thresholdImage.copyTo(temp);
     vector< vector<Point> > contours;
     vector<Vec4i> hierarchy;

findContours(temp,contours,hierarchy,CV_RETR_CCOMP,CV_CHAIN_
APPROX_SIMPLE );
     if (contours.size()>0)objectDetected = true;
     else objectDetected = false;

     if (objectDetected)
     {
```

```
            vector< vector<Point> > largestContourVec;

largestContourVec.push_back(contours.at(contours.size() -
1));
            objectBoundingRectangle =
boundingRect(largestContourVec.at(0));
            int xpos = objectBoundingRectangle.x +
objectBoundingRectangle.width / 2;
            int ypos = objectBoundingRectangle.y +
objectBoundingRectangle.height / 2;
            theObject[0] = xpos, theObject[1] = ypos;
      }
      int x = theObject[0];
      int y = theObject[1];
      circle(cameraFeed, Point(x, y), 30, Scalar(0, 255, 0),
2);
      line(cameraFeed, Point(x, y), Point(x, y - 30),
Scalar(0, 255, 0), 2);
      line(cameraFeed, Point(x, y), Point(x, y + 30),
Scalar(0, 255, 0), 2);
      line(cameraFeed, Point(x, y), Point(x - 30, y),
Scalar(0, 255, 0), 2);
      line(cameraFeed, Point(x, y), Point(x + 30, y),
Scalar(0, 255, 0), 2);
      putText(cameraFeed, "(" + intToString(x) + "," +
intToString(y) + ")", Point(x, y), 1, 1, Scalar(255, 0, 0),
2);
}
```

W celu przefiltrowania obrazu wykorzystana jest, zaczerpniętej z OpenCV, funkcja `findContours`, która pobiera dwa wektory, funkcja zwraca wszystkie kontury.

Jeżeli wektor `contours` nie jest pusty, oznacza to że został wykryty obiekt. Zakładamy ze największy kontur jest szukanym przez nas obiektem (znajduje się na końcu wektora `contours`).
Aktualizowanie położenia obiektu dokonywana jest poprzez wprowadzenie zmian w wartości tablicowej `theObject`.
Następnie rysowane są celowniki obiektu, o promieniu 30, gdzie x, y to środek wykrytego obiektu (punkt ciężkości).

9. Open CV

OpenCV (ang. Open Source Computer Vision Library) jest biblioteką do przetwarzania obrazów w czasie rzeczywistym. Obecnie udostępniana jest na licencji BSD i wspiera wszystkie najważniejsze platformy programowe. Napisana jest C, ale istnieją nakładki na inne języki. Możliwości biblioteki są ogromne, począwszy od prostych operacji na pojedynczych pikselach przez zaawansowane algorytmy przetwarzania obrazów, po algorytmy uczenia maszynowego, wykorzystywane w takich zagadnieniach jak detekcja twarzy. Niewątpliwą zaletą jest możliwość wykonywania wszystkich operacji “w locie”, bezpośrednio na strumieniu wideo.

Biblioteka składa się z czterech głównych komponentów:

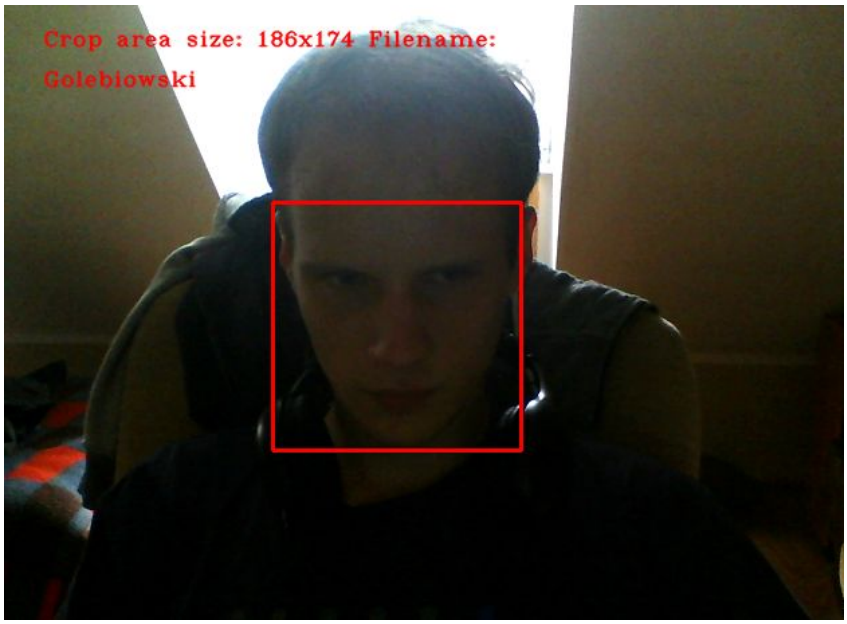
- CV – przetwarzanie obrazów i algorytmy wizyjne.
- MLL – klasyfikatory i narzędzia grupujące.
- HighGUI – GUI oraz wejście/wyjście dla obrazów i wideo.
- CXCORE – podstawowe struktury danych i algorytmy.

Wykrywanie obiektów na obrazie, należy do zadań, dla których ciężko jest stworzyć algorytm. Problem ten jest bardzo złożony. Dlatego też do jego rozwiązania stosuje się elementy inteligencji obliczeniowej. Dziedzina ta zajmuje się między innymi uczeniem maszynowym, czyli grupą algorytmów które uczy się pewnych zadań na podstawie wzorców.
W obecnym czasie do rozpoznawania obiektów, szeroko-stosowane są dwa rozwiązania. Pierwsze z nich to eigenface, a drugie to haar. Biblioteka korzysta z tego drugiego rozwiązania. Metoda ta została opracowana przez Paula Viol'a. Opiera się ona na wirtualnym oknie, które przesuwa się po obrazie, w celu znalezienie pasujących regionów.

10. Ograniczenia programu

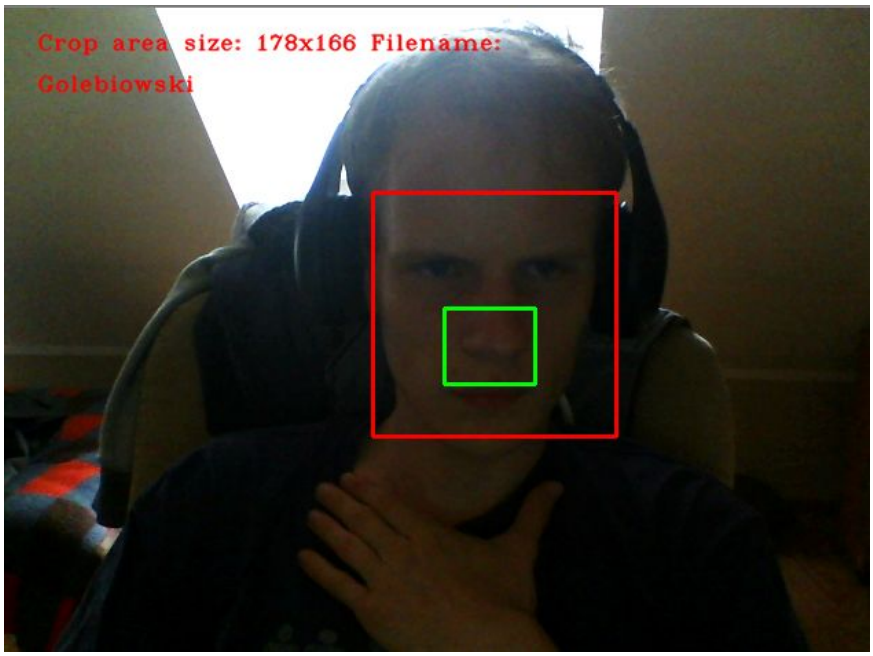
Ograniczenia wynikają głównie z powodu nieodpowiedniego oświetlenia obiektów badanych, a także ze stosunkowo niskiej jakości kamery dla której wykonywane

zostały testy aplikacji



Zdj. 1 Negatywne skutki złego oświetlenia

Na zdjęciu XX główne źródło światła znajduje się za twarzą co drastycznie zmniejsza kontrast twarzy i co za tym idzie poszczególnych elementów twarzy co sprawia, że nie są wykrywane przez program oczy czy nos.

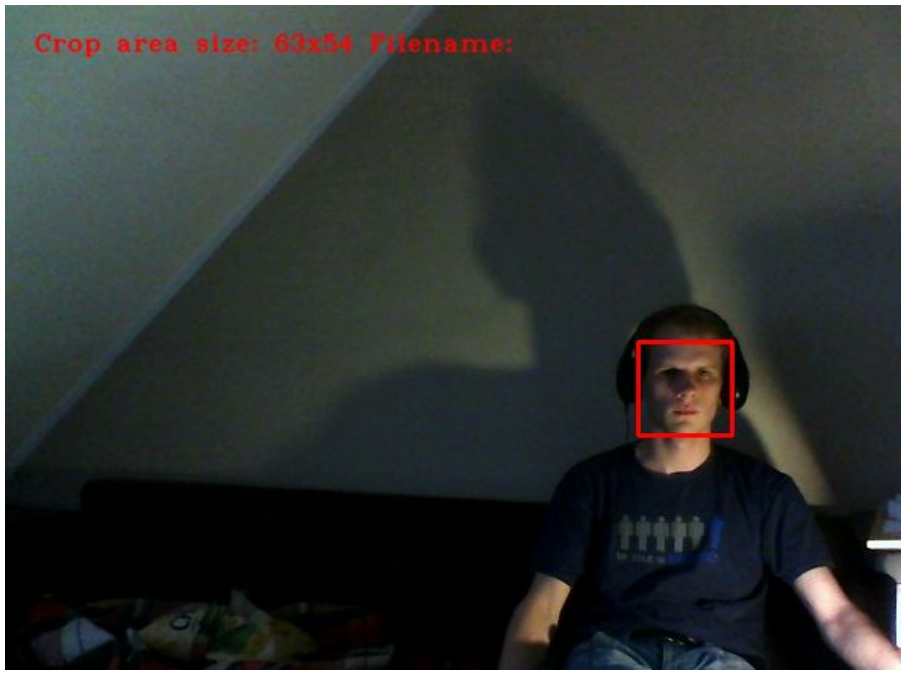


Zdj. 2 Negatywne skutki złego oświetlenia

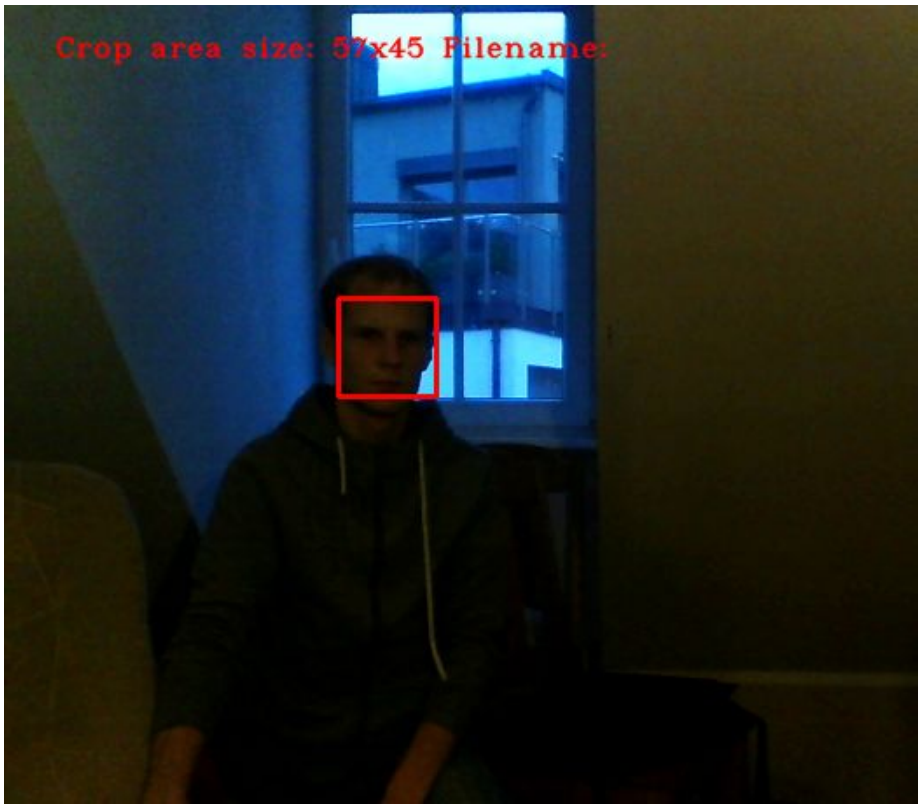
Niskiej jakości oświetlenie powoduje wahania działania programu i niektóre większe elementy twarzy są wykrywane sporadycznie. Nie jest to stałe odwzorowanie jak widać na zdjęciach 1 i 2. Element nosa został zarejestrowany tyko na jednej z tych klatek.

Zdj. 3 Negatywne skutki złego oświetlenia

W skrajnych przypadkach kiedy główna wiązka promieniowania słonecznego jest skierowana w stronę kamery nie jest wykrywana nawet sama twarz.



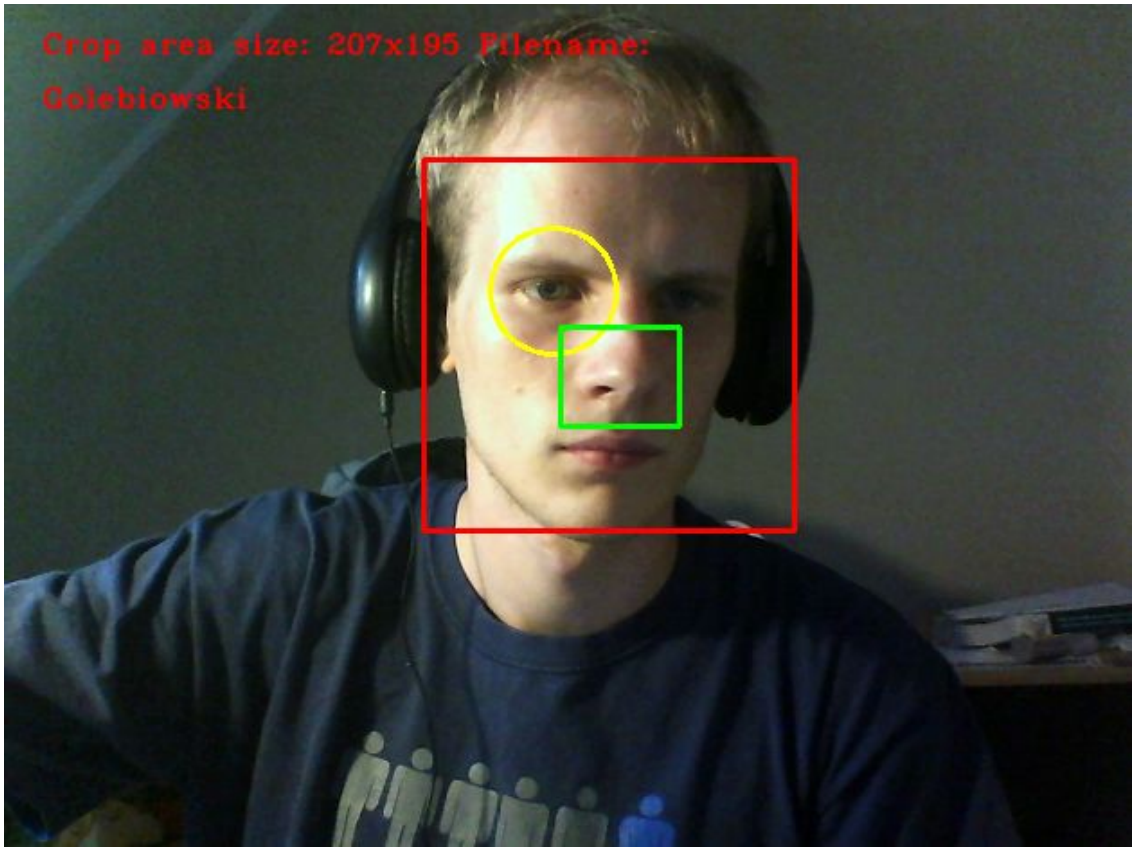
Zdj. 4 Stosunkowo duża odległość od kamery



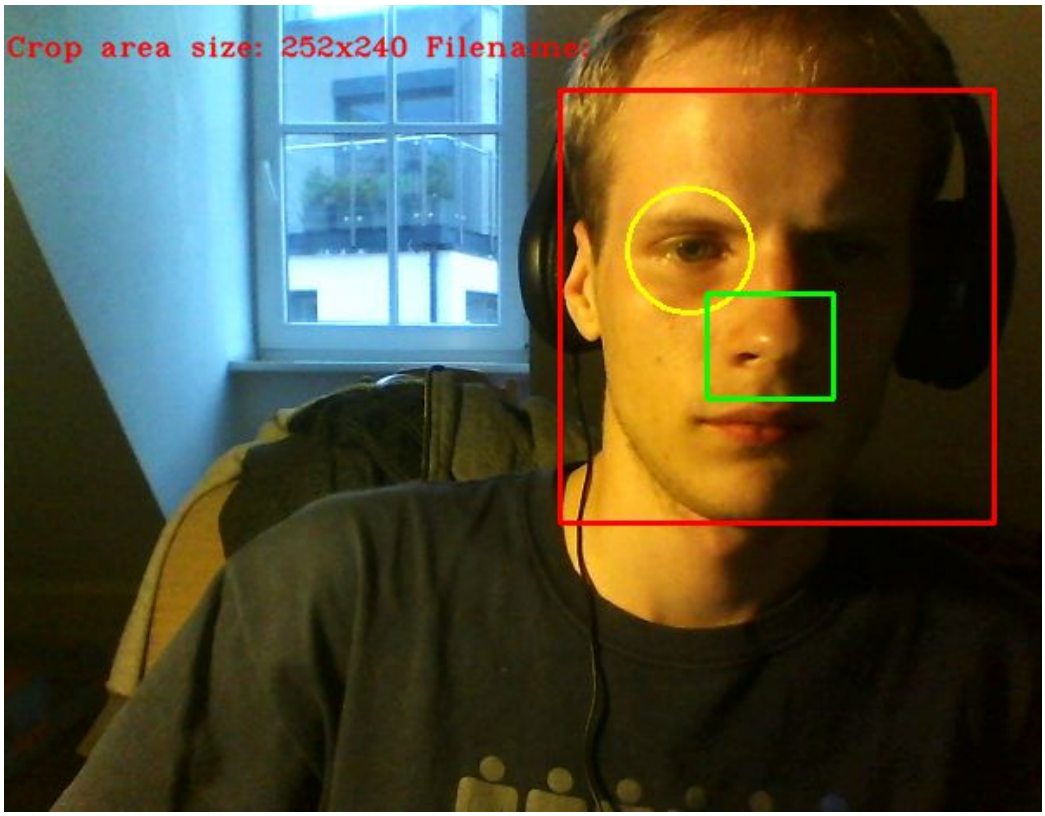
Zdj. 5 Stosunkowo duża odległość od kamery

W sytuacji kiedy badany obiekt znajduje się w stosunkowo znacznej odległości od kamery rozmiary rozpoznawanej twarzy ulegają lekkim wahaniom co czasami

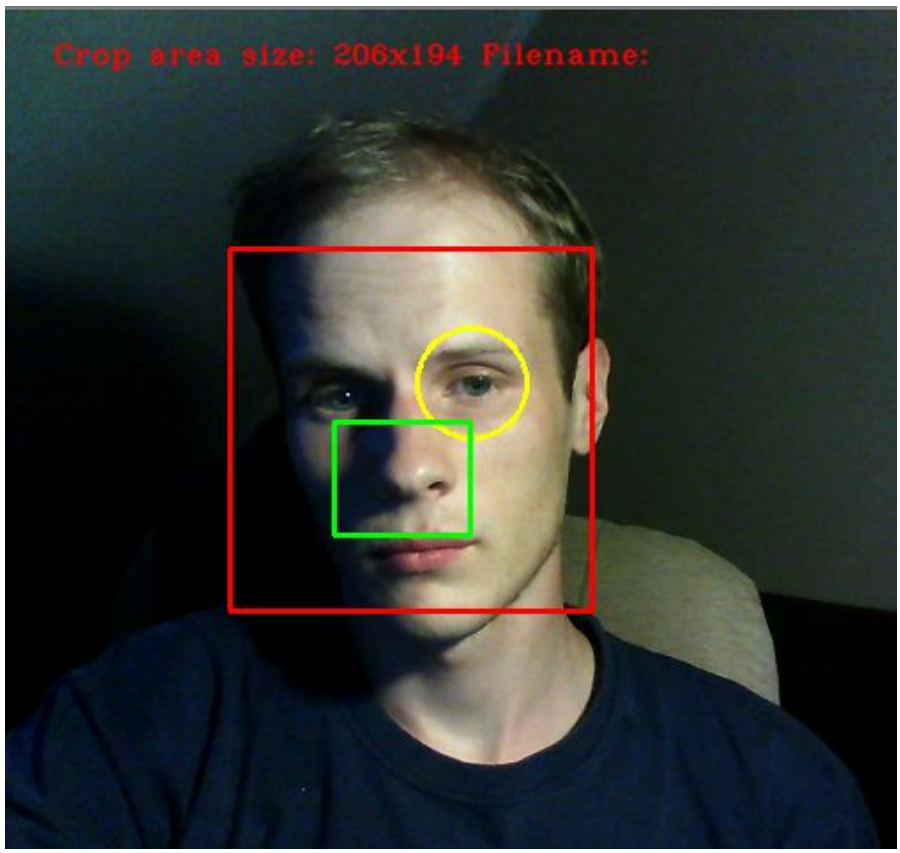
wpływa na pozorną zmianę proporcji wykrywanej twarzy co za tym idzie określenia kim jest wykryta osoba.



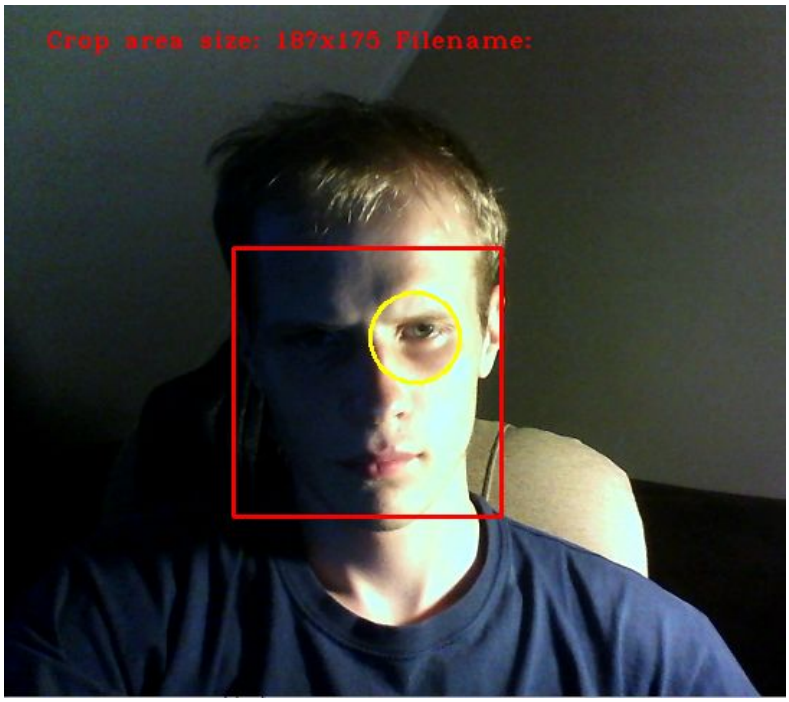
Zdj. 6 Skutki silnego bocznego źródła światła



Zdj. 7 Skutki silnego bocznego źródła światła

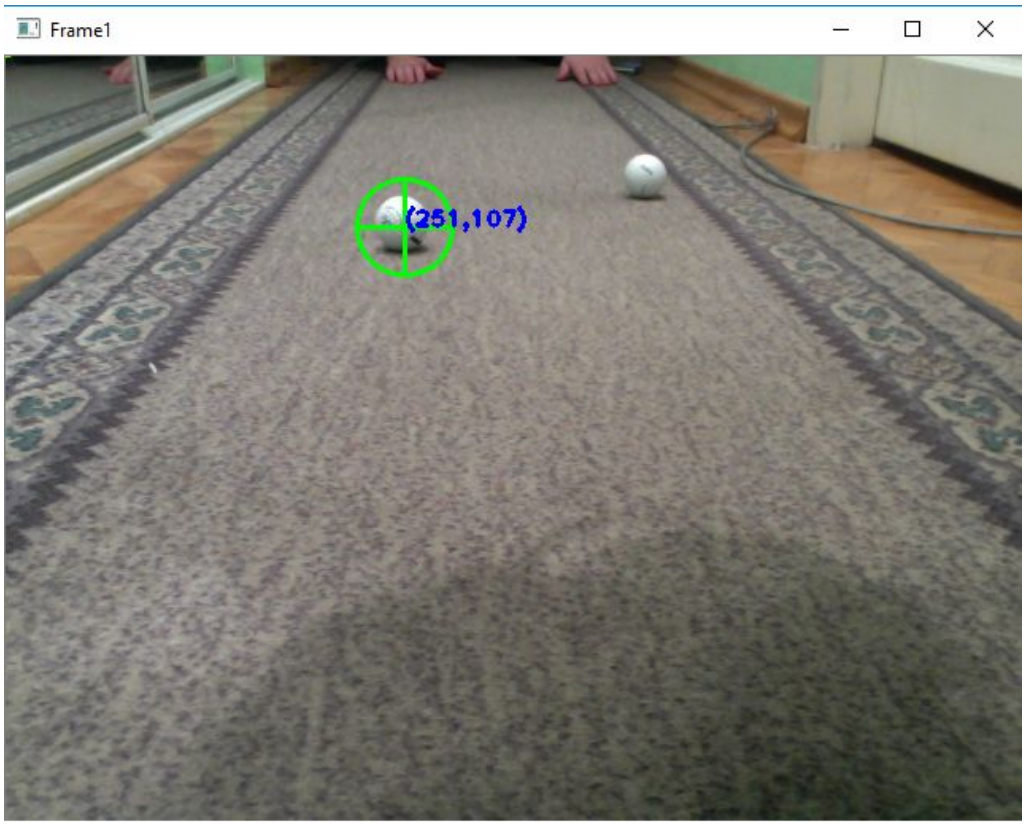


Zdj. 8 Skutki silnego bocznego źródła światła



Zdj. 9 Skutki silnego bocznego źródła światła

W sytuacji kiedy źródło światła oświetla tylko jedną część twarzy algorytm wyznaczający położenie oczu nie wykrywa drugiego oka co jest spowodowane małym kontrastem pomiędzy okiem a twarzą wkoło niego



Zdj. 10 Śledzenie jednego obiektu

11. Wnioski

Pomimo wielu napotkanych problemów udało się zakończyć projekt. Aplikacja realizuje podstawowe założenia projektowe. Została utworzone zgodnie z harmonogramem prac. Program działa poprawnie, jednakże występują opóźnienia i w słabo oświetlonych miejscach występuje problem z wykryciem twarzy i jej elementów.
Program do wykrywania ruchu działa poprawnie dla wykrywania jednego obiektu na raz.

Dokumentacja zawiera opis i testy aplikacji oraz jej poszczególne części składowe.

12. 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)
•Computer Vision and Pattern Recognition, 1991. Proceedings CVPR ’91., IEEE Computer Society Conference on In Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1991), pp. 586-591
•Paul Viola and Michael J. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” IEEE CVPR, 2001.
•Jaesik Choi,Realtime On-Road Vehicle Detection with Optical Flows and Haar-like feature detector
•Install OpenCV in windows (http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html#windows-installation)
•OpenCV download (https://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.4.10/opencv-2.4.10.exe/download)

13. Planistyka

- stworzenie wersji beta dla detekcji twarzy
- prezentacja i zreferowanie wyników prac
- redundancja pracy
- rozpoczęcie implementacji śledzenia obiektów
- wstępna prezentacja osiągniętych wyników
- detekcja poszczególnych części twarzy
- prezentacja wyników na zajęciach
- wykrywanie twarzy na podstawie stworzonego algorytmu detekcji twarzy
- oddanie projektu
- dokumentacja