Politechnika Poznańska
Wydział Elektryczny
Informatyka

Podstawy Teleinformatyki

Malowanie obrazów biorąc pod uwagę ruch ciała

Gracjan Rutkowski 126808
gracjan.rutkowski@student.put.poznan.pl

Norbert Wołowiec 126799
norbert.wolowiec@student.put.poznan.pl

Daniel Lachowicz 126823
daniel.lachowicz@student.put.poznan.pl

Poznań, 28.06.2018

1. Wstęp

Tematem naszej pracy było “Malowanie obrazów biorąc pod uwagę ruch ciała”. Wybraliśmy tę tematykę ze względu na możliwość zmierzenia się z problemem rozpoznawania obrazu, zachęciły nas do tego wcześniejsze zajęcia z Przetwarzania obrazów i systemów wizyjnych.

Problematyka rozpoznania poszczególnych części ciała ludzkiego oraz wyodrębnienie ich na potrzeby sterowania programem wydała nam się wymagająca, ale wykonalna. W internecie można znaleźć wiele rozwiązań rozpoznających ręce, twarze czy ludzkie sylwetki.

W naszym projekcie skupiliśmy się na rozpoznawaniu dłoni oraz jej gestów. To właśnie dłoń zastępuje nam urządzenie sterujące kursorem i pozwala na malowanie po obrazie pobranym z kamery.

1.1. Podział pracy

Nad projektem pracowaliśmy w metodologii równoległej [6]. Podział pracy był następujący:

- Gracjan Rutkowski był odpowiedzialny za obsługę kamery, obrazu oraz pilnowanie ogólnej spójności projektu. W końcowej fazie połączenie modułu rysowania wraz z modułem wykrywanie dłoni i jej gestów
- Norbert Wołowiec pracował nad całym modułem rysowania, jego logiką oraz obsługą wydarzeń
- Daniel Lachowicz miał za zadanie przygotować czytelny interfejs, moduł wykrywania dłoni wraz z jej gestów

1.2. Harmonogram Pracy

Każdy z nas wywiązywał się z wcześniej ustalonych terminów.

Termin Końcowy	**Zadania**
5 kwietnia	faza projektowa, poszukiwanie najlepszych rozwiązań i bibliotek, testy, sprawna obsługa kamery
19 kwietnia	nauka wykrywania dłoni, prace nad logiką rysowania
3 maja	nauka wykrywania gestów dłoni, dalsza praca nad logiką rysowania
17 maja	połączenie modułu rysowania z częścią odpowiedzialną za wykrywanie dłoni i gestów
31 maja	poprawki końcowe, praca nad interfejsem

2. Teoria

Projekt stworzenia aplikacji pozwalającej na malowanie przy pomocy gestów i ruchów dłoni jest tematem mało opisanym w literaturze. W celu przygotowania własnego rozwiązania tego zagadnienia podzieliliśmy je na dwa oddzielne Moduły.

Pierwszy z nich to moduł odpowiedzialny za rysowanie na obrazie. Przykładem takiej aplikacji jest MS Paint. W naszej wersji posiada mniej funkcjonalności, dodatkowe możliwości można dodać w przyszłości.

Drugim modułem jest rozpoznawanie dłoni i jej gestów. Na ten temat można znaleźć obszerną literaturę. Przedstawia ona wiele możliwych rozwiązań tego problemu.

2.1. Moduł rysowania

Grafika komputerowa stała się integralną częścią życia każdego użytkownika urządzeń elektrycznych. Od ikonki aplikacji po menu wyborów. W dzisiejszych czasach grafiki cyfrowe stworzone przy pomocy urządzeń elektrycznych uznawane są za dzieła sztuki.

Komputerową obróbkę graficzną obrazu wykorzystuje się również przy różnych dziełach kultury. Efekty specjalne w filmach, okładki książek, tworzenie komiksów. Dzięki powstawaniu specjalistycznych narzędzi zwiększających precyzję, takich jak tablet graficzny, możemy tworzyć coraz bardziej dokładne grafiki.

2.2. Moduł rozpoznawania dłoni i jej gestów

Komunikacja z komputerem przy pomocy ruchów ciała lub jego poszczególnych części jest tematem ciekawym oraz zgłębianym już od wielu lat. Na przykład w 2010 roku na rynku pojawiły się dwa urządzenia pozwalające na taką komunikację.

Pierwszym z nich jest zaprezentowany przez firmę Sony Computer Entertainment kontroler PlayStation Move skierowane dla konsol PlayStation 3 oraz PlayStation 4. Działa on na zasadzie zczytywanie pozycji kontrolera poprzez obraz z kamery internetowej. Pozwala to na przemieszczanie kursora na ekranie, niestety jeżeli chcemy wykonać akcję, na przykład kliknięcia, musimy wcisnąć przycisk na kontrolerze.

Drugie urządzenie to kontrpropozycja od firmy Microsoft pierwotnie przeznaczone dla urządzeń XBox360 i XboX One. Kinekt bazuje na pobieraniu ruchów szkieletu ciała użytkownika i na jego podstawie odwzorowywaniu ruchów na ekranie telewizora. W 2011 roku firma Microsoft ogłosiła wydanie pakietu Kinekt for Windows SDK pozwalającego na połączenie urządzenia z komputerem posiadającym system operacyjnym Windows.

W celu ułatwienia używania wszelkich aplikacji oraz urządzeń funkcjonalnych coraz częściej korzysta się z naturalnych metod komunikacji, takich jak głos czy ruch

ciała. Coraz częściej przekazuje się informacje przy pomocy obrazów, spowodowane jest to posiadaniem przez każdego aparatu w telefonie. W dzisiejszym świecie technologicznym telefon z dwoma aparatami, jednym z przodu, drugim z tyłu stał się standardem. Komunikacja przy pomocy mowy oraz ruchu ciała jest bardzo dobrym rozwiązaniem dla osób niepełnosprawnych, pozwalającym im na łatwiejszą komunikację z urządzeniem.

Rozpoznawanie gestów samo w sobie jest procesem skomplikowanym. W szczególności ze względu na ilość stopni swobody dłoni [1]. Problem możemy podzielić na: wyekstrachowanie wektora cech i konturu z obrazu oraz klasyfikację gestów.

W naszej pracy testowaliśmy dwa podejścia. Pierwsze z nich polega na stworzeniu dwóch baz danych, testowej i referencyjnej, przy pomocy obrazów dłoni reprezentujących różne gesty [Fig. 1]. Jest to sposób popularniejszy w aplikacjach, które mają za zadanie tylko rozpoznać gest, bądź znaleźć podobny na obrazie.Tworzenie wektora cech oraz klasyfikacji gestów w tej metodzie ma wiele rozwiązań.

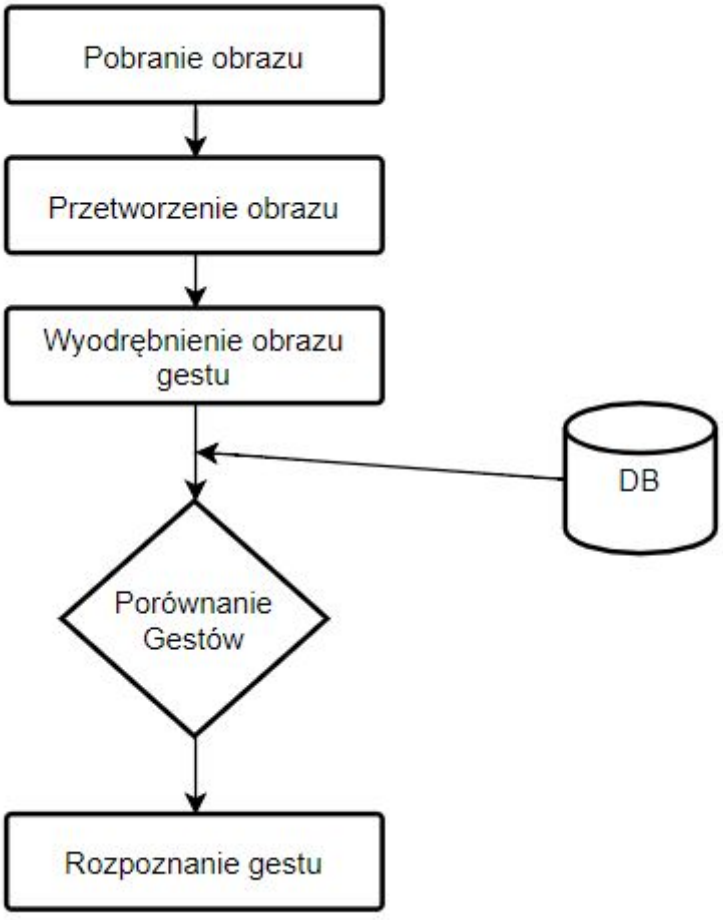


Fig. 1

Diagram wygląda przebieg rozpoznawania gestu dłoni począwszy od pobrania obrazu z kamery lub pliku i kończąc na informacji zwrotnej o geście. Baza danych, na diagramie oznaczona jako “DB”, zawiera wcześniej zapisane gesty.

Jedno z nich przygotował Romer Rosales wykorzystując dwie referencyjne bazy danych zawierające 9 000 i 750 000 obrazów referencyjnych. Pierwsza z nich zawierała obrazy gestów wykonanych w kierunku kamery. Nadgarse, opuszki palców, środek ciężkości oraz inne punkty charakterystyczne dłoni zostały zebrane przy pomocy specjalnej rękawicy nakładanej w trakcie wykonywania gestów. Rosales zastosował metodę dopasowania szablonu i udało mu się osiągnąć 70% skuteczności [2].

Inne podejście do metody opartej na bazy danych zaproponował Stenger stosując, dla każdego gestu, 40 obrazów wzorcowych. Następnie obrócił każdy dziesięciokrotnie, aby uzyskać obrazy odchylone od obrazu wzorcowego na od -10 do 10 stopni dzięki czemu otrzymał 400 obrazów wzorcowych pojedynczego gestu. Przy tworzeniu bazy danych również korzystał z wirtualnej rękawicy [3].

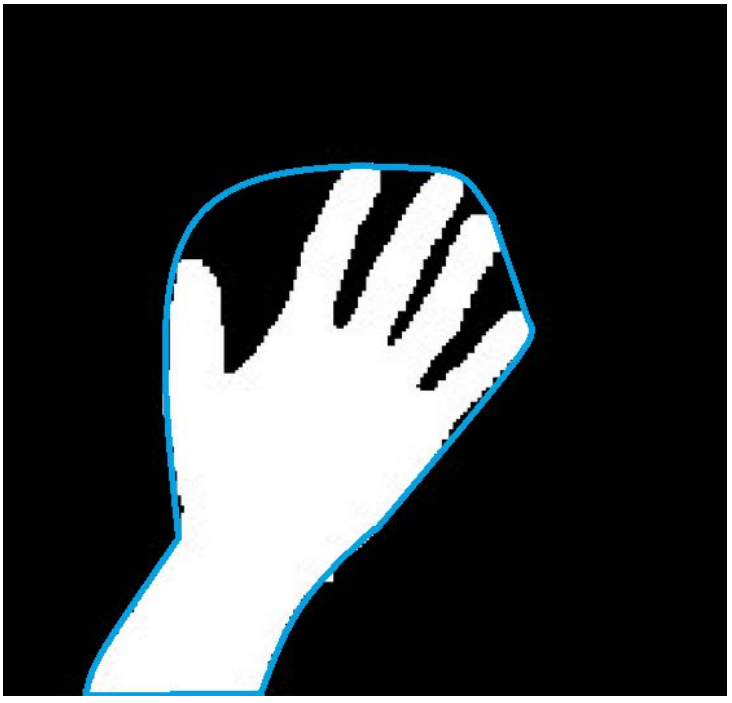
Poza wcześniej wymienionymi metodami stosuje się również samoorganizujące i samorosnące sieci, klasyfikowane następnie przy użyciu sieci neuronowych [4], oraz operacje na obrazie, na przykład zliczanie pikseli w określonych kierunkach od punktu ciężkości masy dłoni.

W internecie można znaleźć wiele przykładów wykorzystywania tej metody w celu tworzenia gier interaktywnych między człowiekiem, a komputerem. Przykładem jest rozegranie rozgrywki kamień, papier, nożyce przy pomocy tej metody i przy użyciu urządzenia Kinekt.

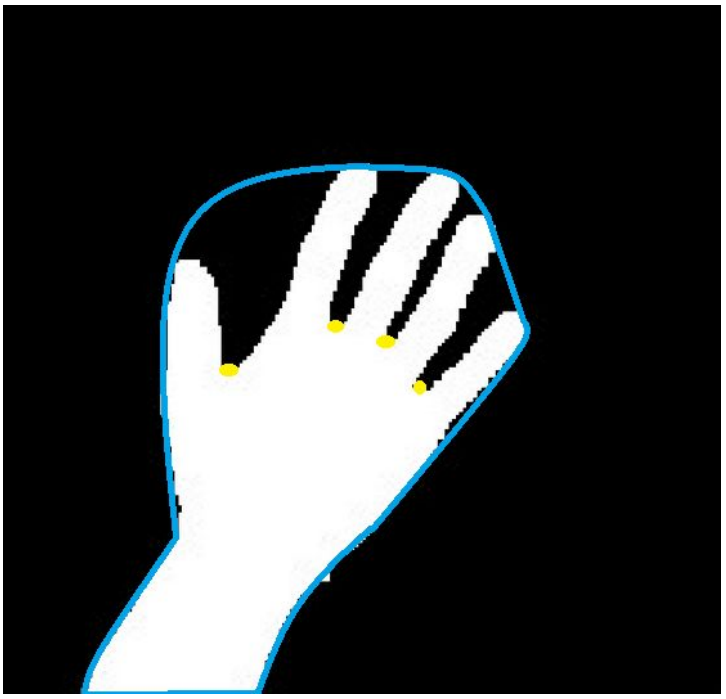
Porzuciliśmy tę metodę ze względu na brak możliwości wyznaczenia punktu przy pomocy gestu, jedyne co byłoby możliwe to wykonywanie akcji zależnie od pokazanego gestu.

Drugim podejściem jest rozpoznawanie dłoni poprzez kolor skóry, problemem tej metody jest wykrywanie twarzy, bądź innych części ciała z odkrytą skórą, jako dłoni. Ta metoda nie posiada żadnych metod uczenia się, ani żadnych baz danych. Wszystkie operacje opierają się o paletę kolorów obrazu [5]. Ta metoda pozwala wyznaczyć stały punkt dłoni mający być kursorem. Rysunki od Rys. 1 do Rys. 6 ukazują krokowe przetworzenie obrazu dłoni w celu uzyskania punktu dla kursora.

Rys. 1
Wykrywanie ręki przez odpowiednie dobranie wartości przy filtrowaniu w celu uzyskania maski dłoni. Biały rejon jest naszym wykrytym obrazem dłoni.

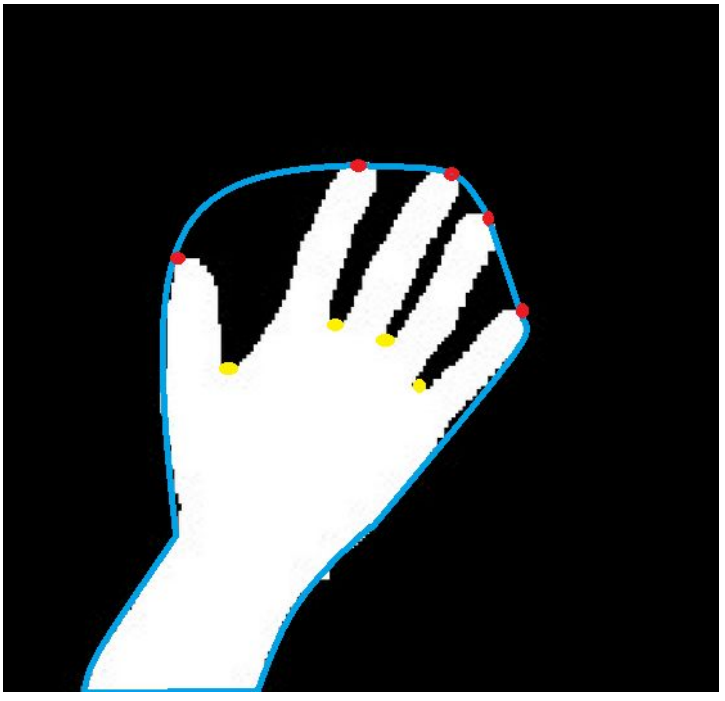


Rys. 2
Rysunek przedstawia wygląd konturu wokół dłoni zaznaczonego linią niebieską, Jest to działanie w początkowej fazie algorytmu, ma ono za zadanie znaleźć na obrazie dłoń w oparciu o kolor skóry. Następnie informacje o konturze przesyłane są do funkcji wyszukującej zagłębienia między palcami.



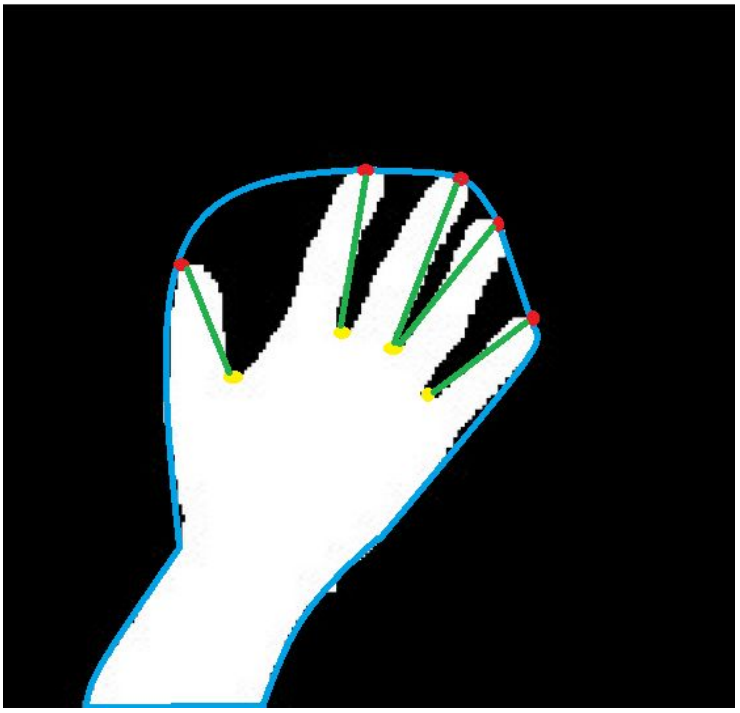
Rys. 3

Po otrzymaniu uzyskaniu konturu, aplikacja wyszukuje punkty zagłębień i oznacza je na żółto. Punkty te mają za zadanie wyznaczenie punktów dłoni między palcami. Informacje o zagłębieniach zostają następnie przesłane dalej w celu wyznaczenia całkowitego palców.



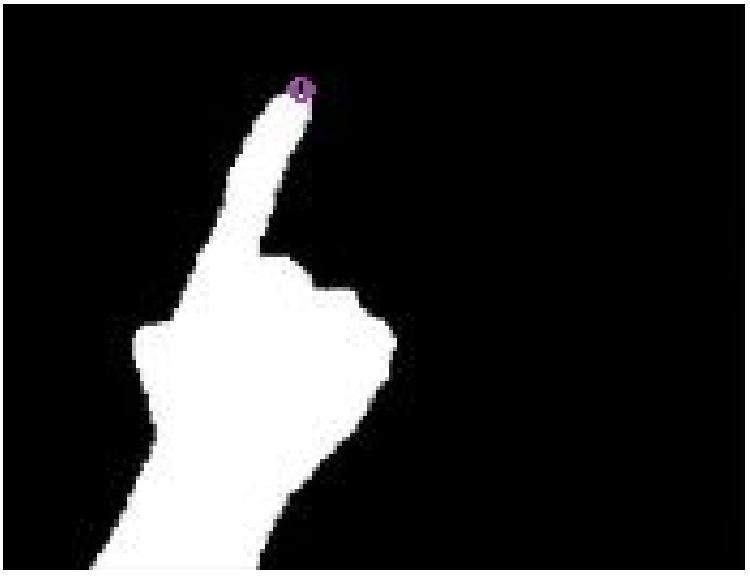
Rys. 4

Po otrzymaniu informacji o zagłębieniach wyszukujemy informacje o punktach szczytowych palców. Znajdują się one na odcinku konturu wcześniej uzyskanego i zawierają piksele w kolorze skóry. Oznaczane są czerwoną kropką.



Rys. 5

Posiadając wyznaczone punkty szczytowe oraz zagłębienia możemy wyznaczyć palce. Oznaczane są kolorem zielonym. Palce wyznaczamy łącząc punkt szczytowy z najbliższym punktem zagłębienia.



Rys. 6

Punkt przekazywany jako wskaźnik jest miejscem szczytowym dla najdłuższego palca. Przy okazji zliczamy ilość palców. Gesty rozpoznajemy przez ilość wyprostowanych palców.

Idealnym rozwiązaniem byłoby połączenie obu sposobów na rozpoznanie dłoni. Pozwoliłoby to na poszerzenie zakresu gestów wykonujących akcje, dzięki czemu możemy obsłużyć więcej akcji. Doprowadziłoby to do zmniejszenia ilości ikon na ekranie bez utraty funkcjonalności.

Każda z tych metod ma swoje mocne i słabe strony. Tab. 1 przedstawia porównanie obu metod ze strony użytkownika, uwzględniliśmy w tym porównaniu tylko kilka najistotniejszych, według nas, wartości.

Tab. 1
Poniższa tabela przedstawia porównanie czterech wartości, na które zwróciłby użytkownik przy wyborze dwóch identycznych aplikacji opartych o różne metody. Koszt wskazuje, czy poza samą aplikacją potrzebne są jakieś dodatkowe przedmioty. Kształt dłoni wskazuje czy możliwe są jakiekolwiek odstępstwa od kształtu dłoni osoby przygotowującej aplikację lub dłoni w bazie danych.

Metoda	Porównanie z baza danych	Rozpoznawanie w oparciu o kolor skóry
Koszt	Niższy	Wyższy
Wygoda użytkowania	Wyższa	Niższa
Kształt dłoni	Wyższe kryteria	Mniejsze kryteria
Kalibracja	Niekonieczna	Konieczna

Porównanie wskazuje, że metoda oparta o porównanie z bazą danych jest wygodniejsza z perspektywy użytkownika. Jednak w naszym projekcie wybraliśmy metodę drugą. Było to spowodowane głównie potrzebą wyznaczenia konkretnego punktu, wskazującego koordynaty dla kursora. Pierwsza metoda nie pozwalała na łatwe wykrycie takiego punktu. W drugiej było to dużo łatwiejsze, przy wykrywaniu skóry algorytm zaznaczał jako dłoń również inne części ciała, np twarz. W toku prac nad implementacją tej metody, w celu zniwelowania problemu wykrywania nadmiernych części ciała, uznaliśmy, że wymagana będzie rękawiczka jednokolorowa.

3. Założenia

Głównym założeniem projekty było stworzenie aplikacji pozwalającej na rysowanie przy pomocy gestów dłoni. W celu poprawy odczuć płynących z użytkowania oprogramowania dodano narzędzia urozmaicające obsługę. Zadbano o to by można było o zapisanie wyników pracy do pliku.

3.1. Funkcjonalne

- Sterowanie przy pomocy ruchów dłoni - poruszanie kursorem po ekranie przy użyciu dłoni jako kontrolera, kursor się przemieszcza w momencie, kiedy mamy wyprostowane wszystkie palce dłoni
- Rysowanie na ekranie - rozpoczynamy rysowanie prostując tylko jeden palec na dłoni, tak długo jak wyprostowany jest tylko jeden palec będziemy rysować na ekranie
- Wybór koloru - figury mogą być rysowane w różnych kolorach wybieranych z palety kolorów
- Wybór figury - przy pomocy odpowiednich ikon możemy wybrać poszczególne figury, które chcemy wykorzystać przy malowaniu na ekranie
    - Linia - prosta linia, zaznaczamy punkt początkowy i końcowy,
    - Prostokąt - kontur prostokąta, rysowany od na zasadzie wyboru punktu początkowego będącego narożnikiem, a następnie punktu końcowego będącego narożnikiem naprzeciw początkowego
    - Elipsa- elipsę rysujemy przy pomocy wyboru dwóch punktów, tak samo jak w przypadku prostokąta, elipsa powstanie jako okrąg wpisany w prostokąt przy założeniu, że musi mieć cztery miejsca stykowe z prostokątem
    - Pędzel - tak długo jak robimy dłonią gest rysowania tak długo będziemy rysować, linia będzie powstawać w miejscach nad którymi przesuniemy kursorem
- Zapisanie do pliku - zapisujemy postęp prac do pliku, zarówno to co narysowaliśmy jak i obraz z kamery
- Włączenie tła z pliku - wczytujemy plik graficzny do aplikacji w celu zastąpienia widoku z kamery własną grafiką
- Wybór koloru - wybieramy kolor z palety barw w kształcie koła, w momencie kiedy najedziemy kursorem na paletę powiększa się ona i możemy wybrać kolor

3.2. Niefunkcjonalne

- Wskazywanie kursora na względem położenia dłoni - aplikacja zczytuje położenie dłoni we współrzędnych obrazu, następnie wysyła je jako współrzędne kursora
- Aplikacja obsługuje ruch jednej ręki - jeżeli pojawią się dwie ręce lub więcej aplikacja nie będzie poprawnie działać, kursor losowo może przeskakiwać między obiema dłońmi
- Wymagana kamerka internetowa - aby aplikacja mogła pobrać informacje o położeniu ręki musi najpierw pobrać obraz z kamery internetowej.
- Urządzenia z systemem Windows - aplikacja przeznaczona jest na urządzenia posiadające system operacyjny Windows
- Wykrywanie dłoni - aplikacja poszukuje, na obrazie pobranym z kamery, dłoni użytkownika w celu obsłużenia programu
- Wykrywanie gestów - aplikacja sprawdza obraz z kamery w celu sprawdzenia gestu wykonywanego przez użytkownika

4. Wybrane technologie

W trakcie pracy nad naszą aplikacją korzystaliśmy z języka C#[jakiś odnośnik do bibliografii]. W naszym projekcie wykorzystaliśmy gotowy silnik Windows Presentation Foundation bazujący na .NET, ponieważ pozwala na szybkie przygotowanie szkieletu interfejsu aplikacji.

4.1. Biblioteki:

4.1.1. EmguCV

Jest to klasa opakowująca bibliotekę OpenCV, dzięki temu możemy korzystać z funkcjonalności tej biblioteki w języku C#

4.1.2. System.Windows.Shapes

Zapewnia dostęp do biblioteki kształtów, dzięki niej możemy rysować coś więcej niż proste linie

4.2. Narzędzia:

4.2.1. Microsoft Visual Studio 2017

Środowisko programistyczne wykorzystane przez nas w celu napisania całej aplikacji

4.2.2. Git

System kontroli wersji, który wykorzystywaliśmy w celu udostępniania w grupie postępów

5. Opis implementacji

Implementacja aplikacji trwała trzy miesiące, już w fazie projektowej powstawały pierwsze linie kodu przy testowaniu różnych języków i ich funkcjonalności. Pierwszym działaniem było wybranie języka programowania. Zastanawialiśmy się między Pythonem a C#. Pierwsza możliwość posiadała już wiele gotowych rozwiązań, o które moglibyśmy się oprzeć w pracy nad własną aplikacją. Nad wyborem drugiej jednak przemawiało nasze doświadczenie w pracy z tym językiem.

Obie możliwości zawierają dużą ilość bibliotek ułatwiających pracę przy implementacji zarówno rysowania na obrazie jak i rozpoznawania dłoni oraz jej gestów. Ostatecznie wybraliśmy drugą opcję. Dużym plusem przy pracy z językiem C# był silnik graficzny Windows Presentation Foundation pozwalający na łatwiejsze tworzenie interfejsu użytkownika. Fig. 2 - 8 przedstawiają zarys programu z podziałem na klasy.

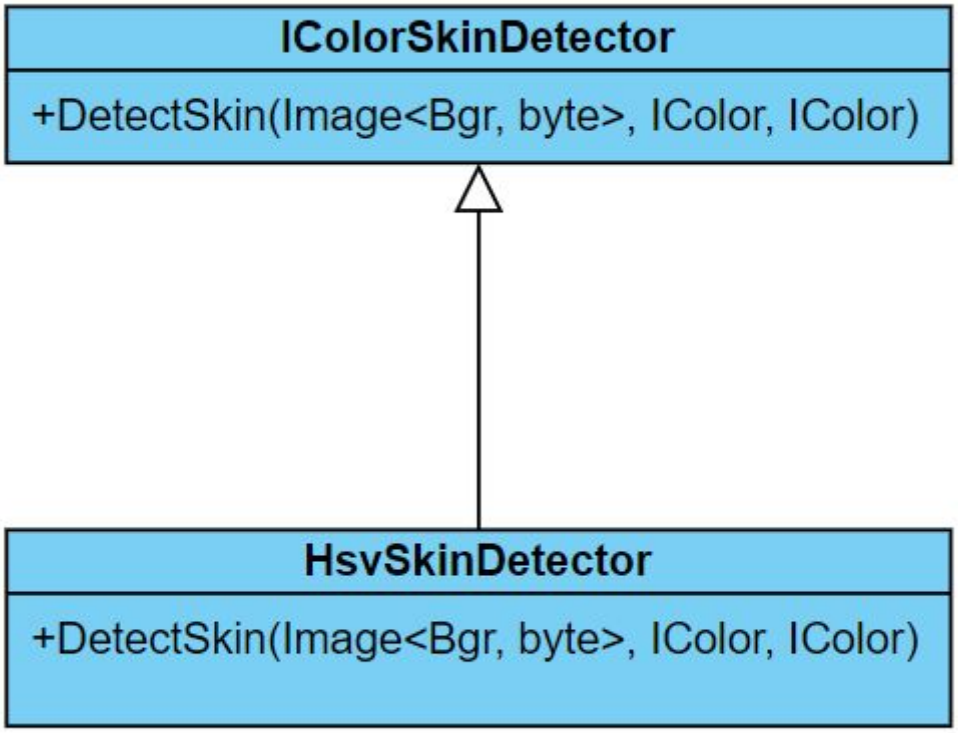


Fig. 2

Diagram przedstawia konstrukcję klasy HsvSkinDetector odpowiedzialną za wykrywanie na obrazie skóry dłoni. Dziedziczy ona po interfejsie IcolorSkinDetector. Klasa ta jest wykorzystywana przez klasę HandDetection.

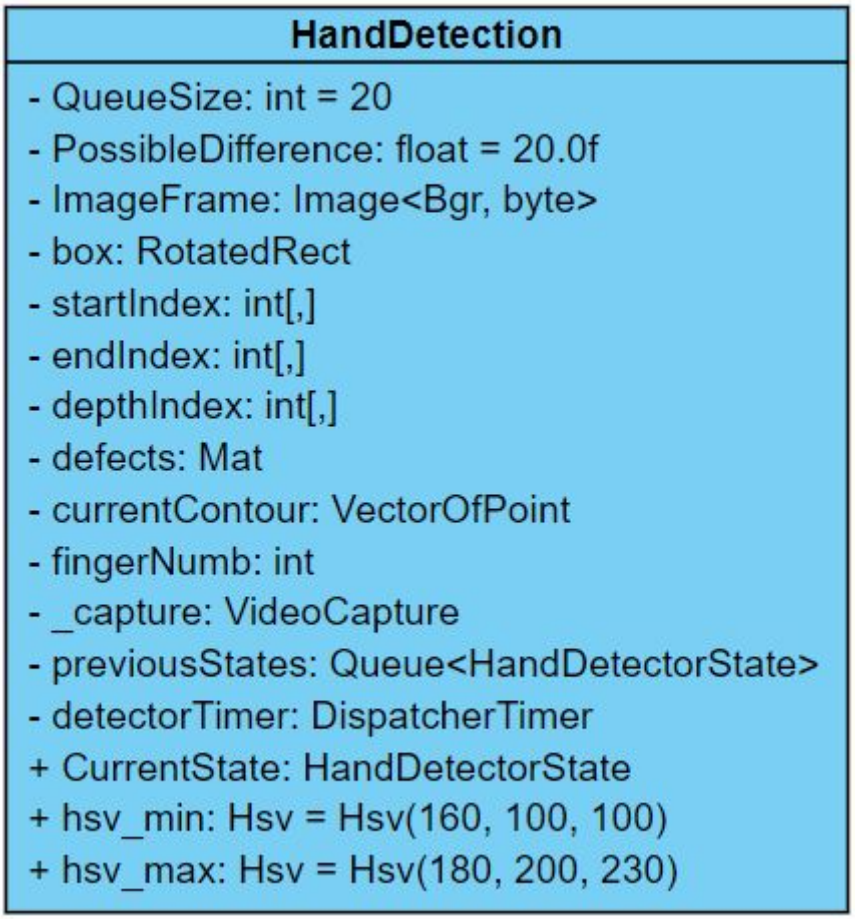


Fig. 3

Diagram przedstawia klasę HandDetection, a dokładnie jej zmienne. Klasa ta jest odpowiedzialna za obsługę wykrywania na obrazie dłoni oraz pobieranie z obrazu współrzędnych dla kursora.

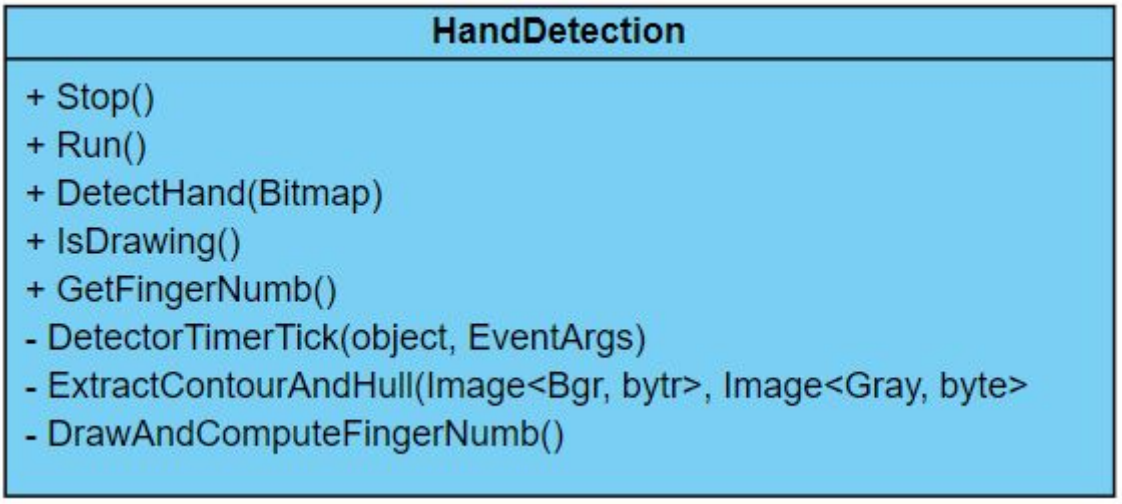


Fig. 4

Diagram przedstawia metody zawarte w klasie HandDetect. Metody te odpowiadają za rozpoczęcie pracy wątku, sprawdzenie obrazu za ręką, podanie ilości palców oraz informacji, czy wykonano gest rysowania.

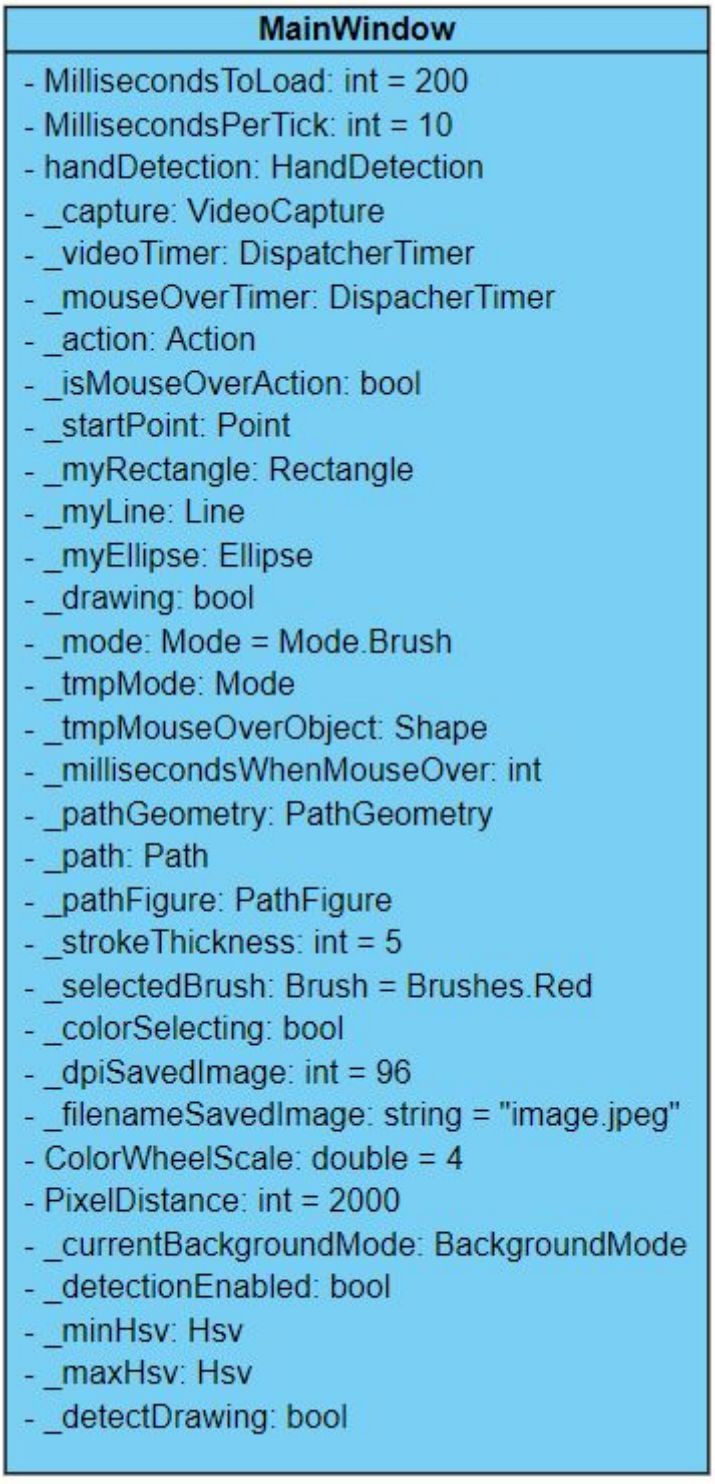


Fig. 5
Diagram przedstawia zmienne klasy MainWindow. Jest to klasa odpowiedzialna za moduł rysowania, zawiera również metody wykorzystywane przez interfejs użytkownika.

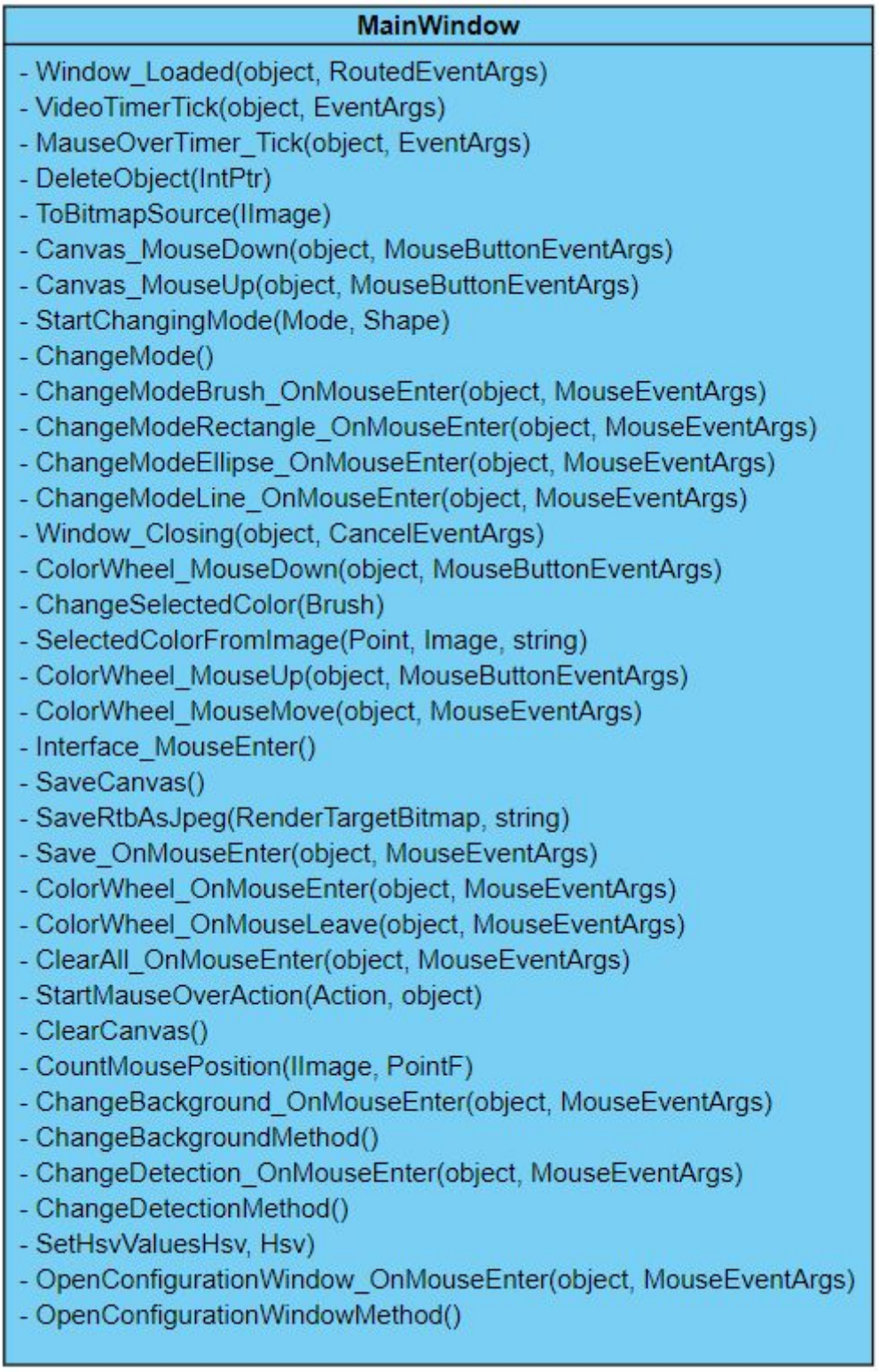


Fig. 6
Diagram klasy przedstawia wszystkie funkcje zawarte w klasie MainWindow. Funkcje te służą do obsługi interfejsu użytkownika oraz konfiguracji detekcji dłoni.

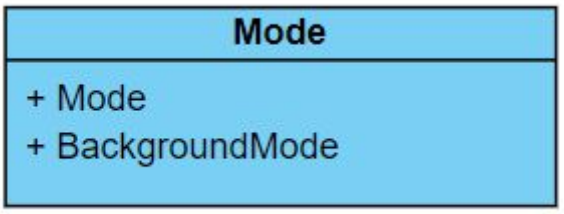


Fig. 7
Diagram przedstawia klasę przechowującą wartości Enum wykorzystywane w całym programie. Wybraliśmy to rozwiązanie, ponieważ te wartości wykorzystywane są w wielu klasach i czytelniejsze było wyodrębnić je do oddzielnej klasy.

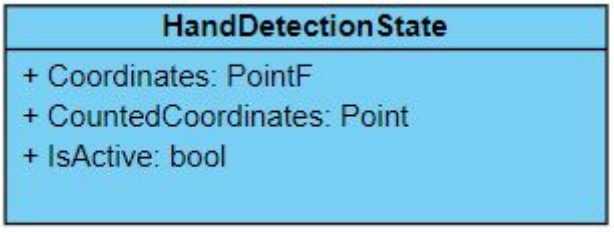


Fig. 8
Diagram przedstawia budowę struktury HandDetectionState. Jest ona odpowiedzialna za przechowywanie koordynatów pobranych z obrazu oraz przefiltrowanej wartości tych koordynatów.

5.1. Architektura rozwiązania

Aplikację w części implementacyjnej podzieliliśmy na dwie części, pierwsza to moduł rysowania, druga to moduł rozpoznawania dłoni oraz jej gestów. Połączenie ich polega na przesyłaniu wartości z funkcji detekcji ręki do funkcji ruchu kursorem oraz sprawdzenia czy wykonywany jest gest rysowania, czy nie. Oba moduły działają na oddzielnych wątkach, dzięki czemu aplikacja działa szybciej.

5.1.1. Moduł rysowania

Moduł rysowania odpowiada za obsługę malowania i tworzenia figur geometrycznych w miejscach wskazanych przez kursor. Do obsługi kursora możemy wykorzystać dwa kontrolery, pierwszy to mysz, drugi to nasza dłoń wraz z rękawiczką. Do wyboru mamy cztery rodzaje narzędzia do rysowania, pędzel, prostokąt, elipsa oraz linia prosta.

W celu pozbycia się problemu skaczącego kursora, ze względu na minimalne przesunięcie się wektora odczytanego na obrazie, zaprojektowaliśmy filtr uśredniający tę wartość. Pobiera on ostatnie trzydzieści wartość i na ich podstawie wylicza wartość średnią położenia kursora.

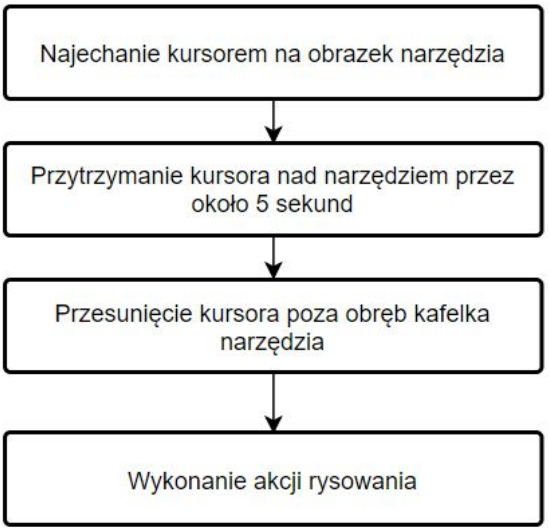


Fig. 2
Przebieg akcji wyboru narzędzia do rysowania. Diagram jest uniwersalny dla wszystkich czterech możliwych końcówek pędzla.

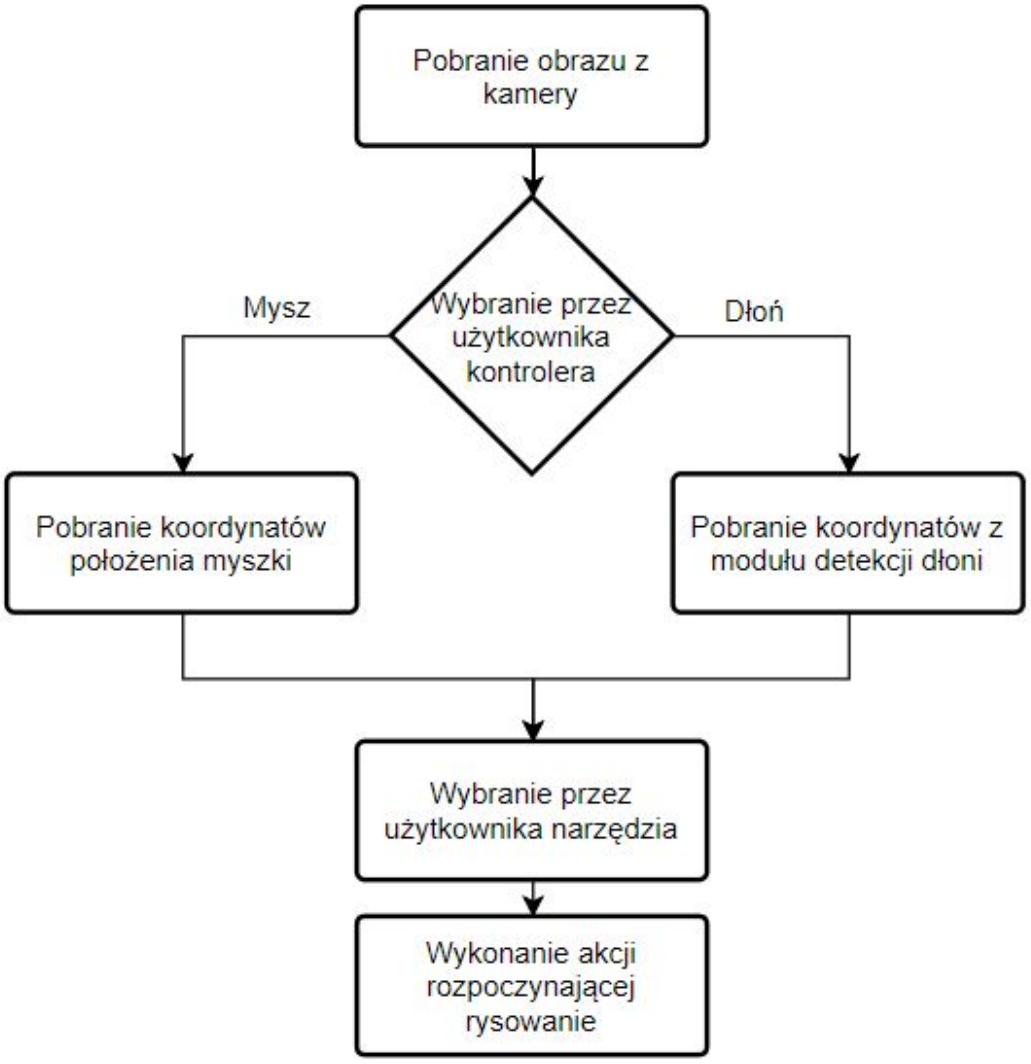


Fig. 2
Diagram przedstawia zarys działania modułu rysowania, użytkownik może wybrać kształt figur, które chce rysować oraz ich kolor. Możliwe jest również wybranie białego tła, czy przełączenie między trybami wprowadzania.

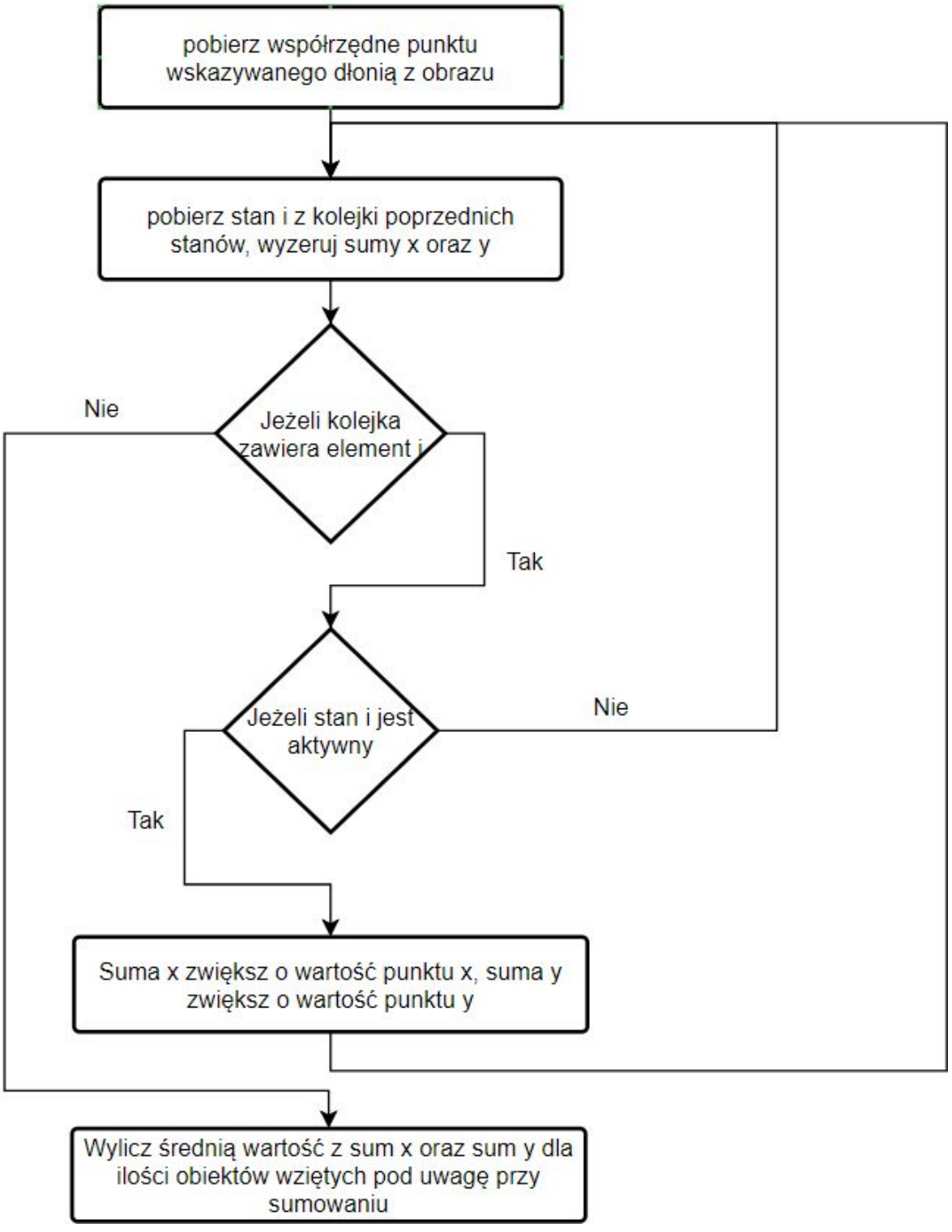


Fig. 3
Schemat algorytmu filtracji położenia kursora na ekranie. Wykorzystuje on kolejkę poprzednich stanów, którą następnie przegląda w celu uśrednienia położenia. Po uśrednieniu dodaje je do kolejki. Kiedy kolejka zostaje przepełniona ostatni element zostaje z niej zdjęty.

5.1.2. Moduł rozpoznawania dłoni oraz jej gestów

Moduł ten odpowiada za poszukiwanie na obrazie ręki, sprawdzanie jej gestu oraz przesyłanie współrzędnych wskaźnika na ręce. Pobiera z modułu kolorów dane na temat koloru rękawiczki, a następnie wyszukuje przy ich pomocy na obrazie kontrolera w postaci ręki w rękawiczce. Następnie przeszukuje obraz w poszukiwaniu punktów zagłębień i szczytowych. Na ich podstawie wyznaczany jest palec, a następnie po zliczeniu palców sprawdzany jest gest. Zależnie od ilości wyprostowanych palców wykrywa gest rysowania lub ruchu.

Jest to moduł podzielony na kilka pomniejszych klas. Klasa główna odpowiada za obsługę całej logiki rozpoznawania dłoni, jej gestów oraz wyznaczania punktu kursora. Pomniejsze klasy odpowiadają za rozpoznawanie koloru skóry lub rękawiczki w oparciu o podany zakres kolorystyczny.

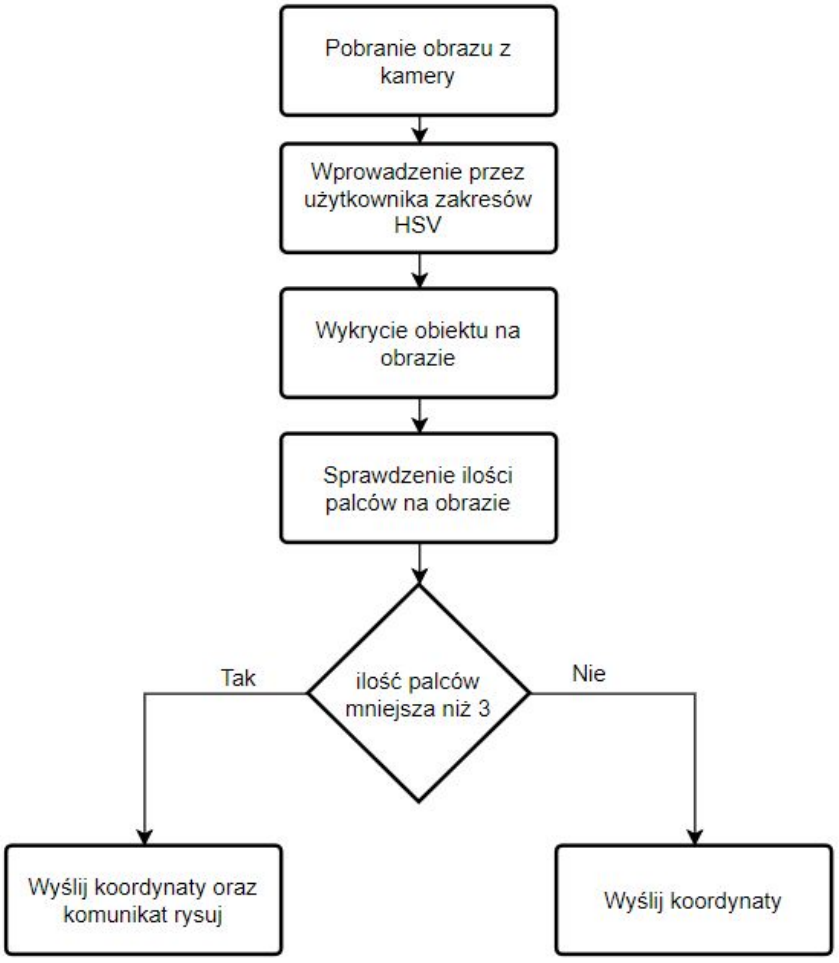


Fig. 3
Diagram przedstawia zamysł przebiegu pracy modułu od pobrania obrazu do wysłania informacji do modułu rysowania. Informacje o zakresie kolorów HSV wprowadza użytkownik po uruchomieniu programu.

5.2. Interesujące problemy i ich rozwiązania

5.2.1. Problem określania punktu

W pierwotnym zamyśle programu chcieliśmy wykorzystać biblioteki uczące się w celu rozpoznawania dłoni na obrazie, dałoby to nam możliwość na większy zakres gestów obsługiwanych przez aplikację, niestety nie pozwalało to na jednoznaczne określenie punktu wskazywanego przez dłoń wykonującą gest.

Rozwiązaliśmy ten problem przez zmianę podejścia z bibliotek zdeklarowanych gestów na rozpoznawanie dłoni poprzez kolory. Pozwoliło to nam na określenie jednego punktu będącego lokalizacją wskaźnika, niestety zawęziło ilość gestów obsługiwanych do dwóch, pokazanie wszystkich palców prosto powoduje tylko przesunięcie kursora, dłoń z wyprostowanym tylko jednym palcem powoduje rozpoczęcie rysowania lub wybór na pasku narzędzi. Punkt kursora pokrywa się z punktem wskazywanym przez wyprostowany palec.

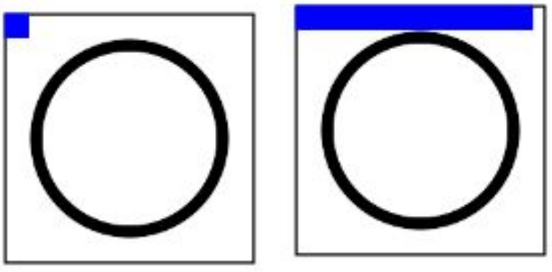
5.2.2. Problem rozpoznawania skóry

Drugim z problemów które napotkaliśmy było rozwiązanie problemu czytania barwy skóry, jeżeli w zasięgu kamery pojawił się element ciała z odkrytą skórą, nie będący dłonią, powodowało to jego wykrycie, aplikacja przez to gubiła współrzędne kursora i przesuwała go w miejsce przez nas niechciane.

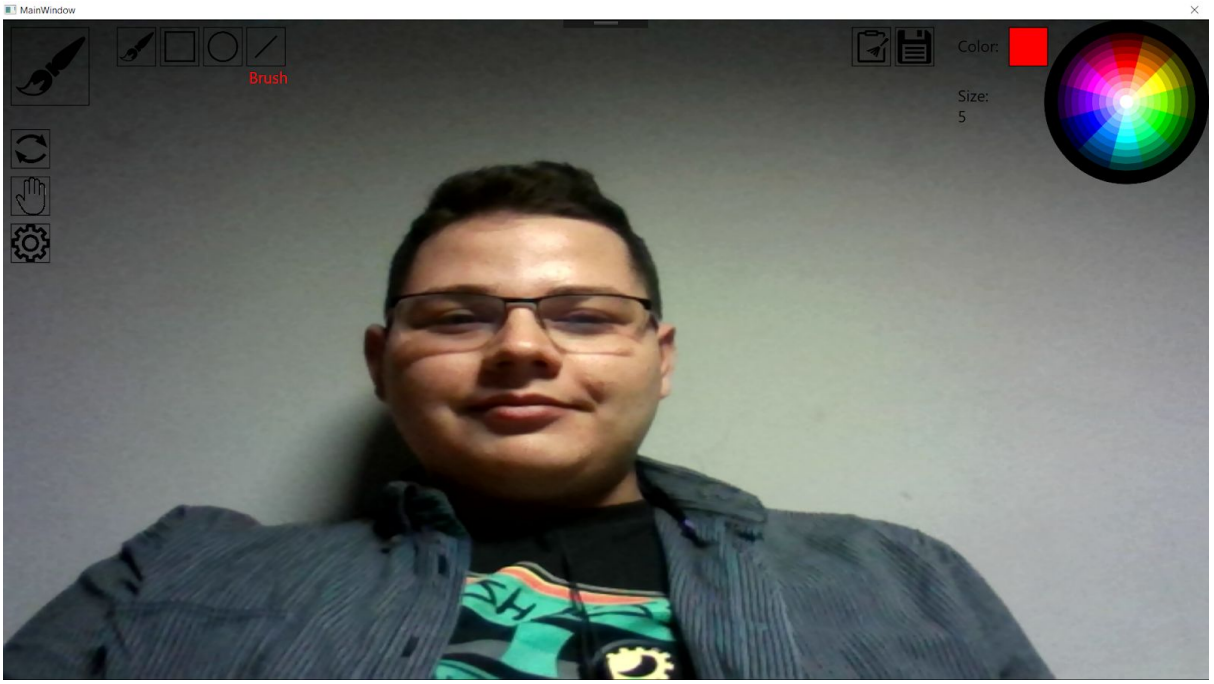
Z tym problemem poradziliśmy sobie przy pomocy niebieskiej rękawiczki, zakładamy ją na dłoń, a program poszukuje kolory z nią związane, dzięki temu kolor skóry nie jest już wykrywany jako dłoń.

5.3. Instrukcja użytkowania

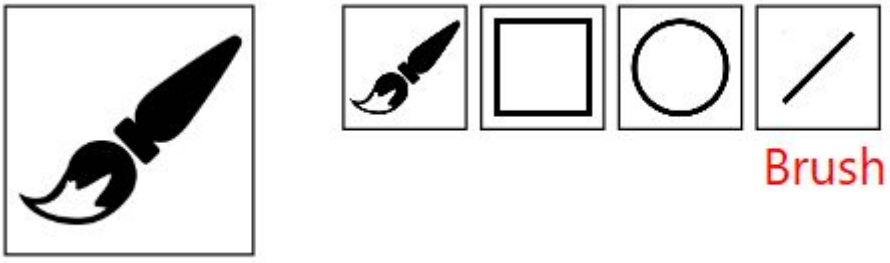
Nasza aplikacja wymaga posiadania jednolitej kolorystycznie rękawiczki, zalecamy kolor niebieski, w celu sterowania dłonią. Obszarem na którym rysujemy jest albo obraz pobrany z kamerki sprzętowej komputera lub na białej kartce. Wszelkie działania dłonią odczytywane są z każdą klatką obrazu, rysowanie na obrazie z kamerki odbywa się na na bieżąco pobieranych obrazach. W celu wybrania jakiejś opcji należy najechać kursorem nad nią i przytrzymać go przez kilka sekund, aż pasek ładowania się nie zapełni.



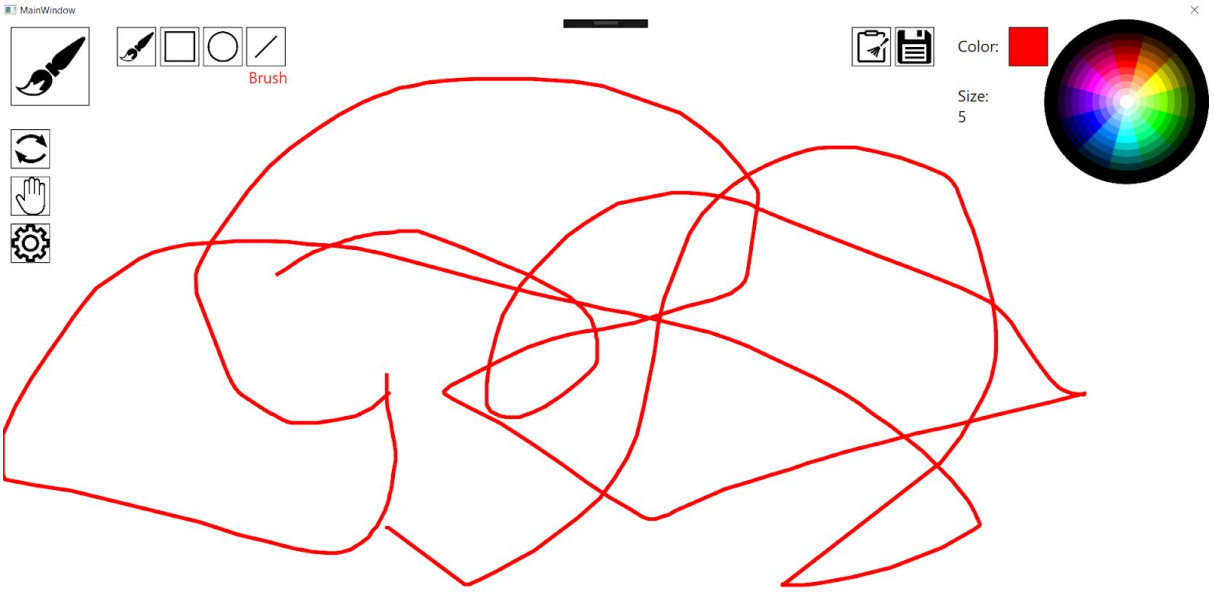
Rys. 7
Lewa strona rysunku przedstawia pasek ładowania w początkowej fazie pojawiający się po najechaniu na ikonę jednego z przyborów. Prawa strona przedstawia ten sam pasek w końcowej fazie ładowania.



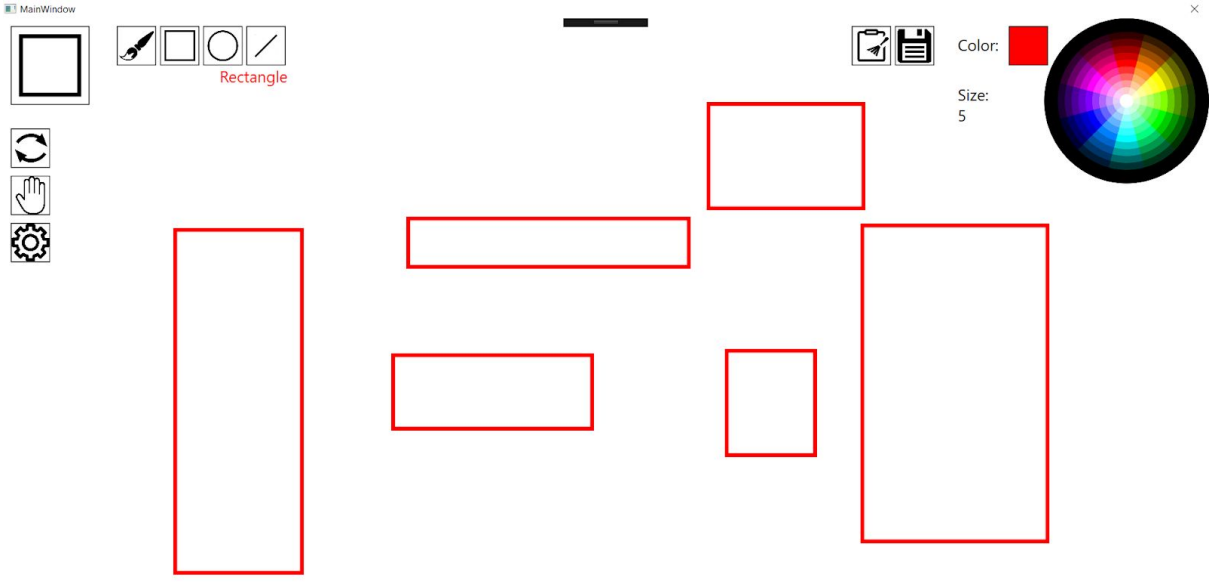
Rys. 8
Widok ekranu po uruchomieniu aplikacji. W górnej części widzimy pasek z możliwościami aplikacji. Po lewej stronie znajdują się końcówki rysownicze, po prawej opcje dodatkowe.



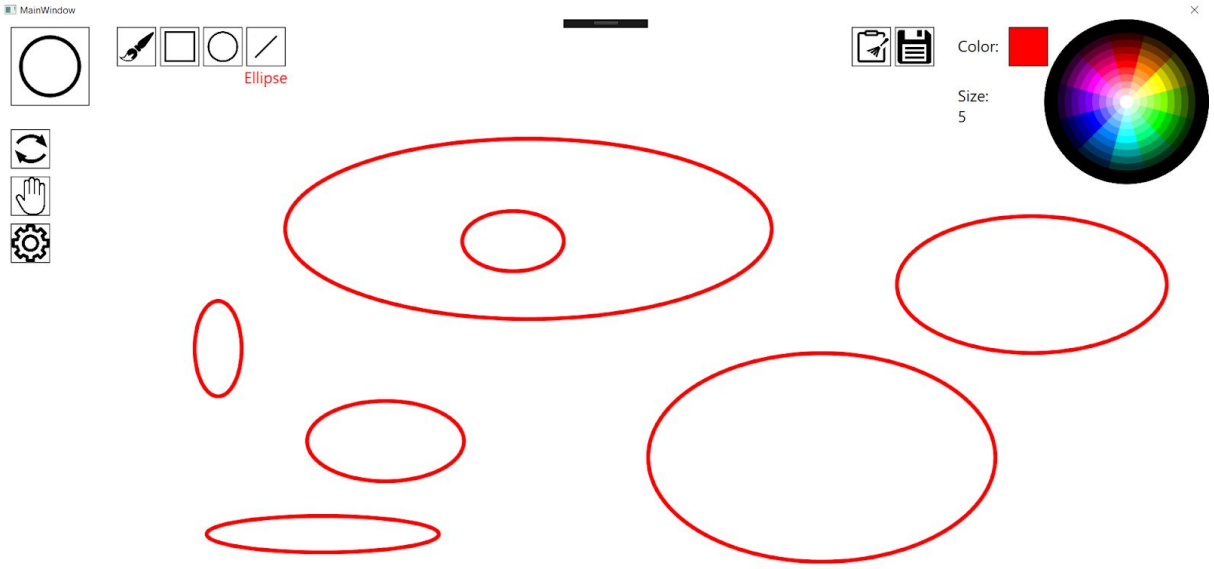
Rys. 9
Lewa górna część paska narzędzi. Największy kafelek przedstawia aktualnie wybrane narzędzie do rysowania. Mniejsze przedstawiają możliwe narzędzia rysownicze. Kolejno od prawej: linia prosta, elipsa, prostokąt oraz pędzel. Napisz na czerwono informuje użytkownika o nazwie aktualnie wybranego narzędzia.



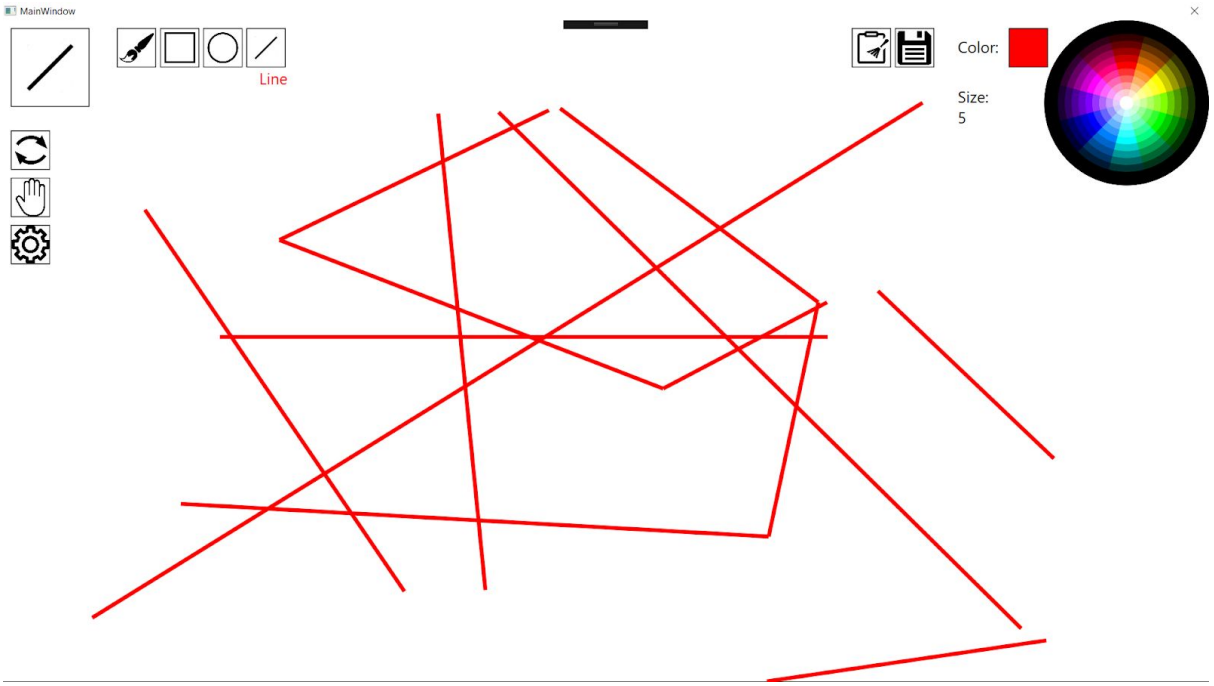
Rys. 10
Przykładowy rysunek przy pomocy narzędzia pędzla. Pozwala ono na rysowanie w miejscach nad którymi przeciągniemy kursor. Minusem tego przyboru jest trudność narysowania linii prostej.



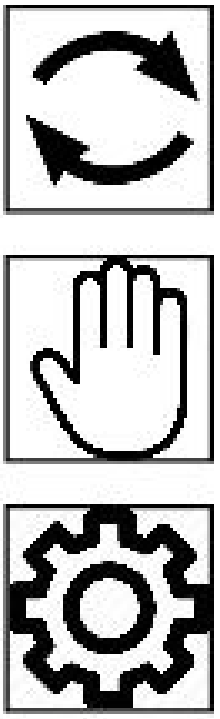
Rys. 11
Przykładowy rysunek wykonany przy pomocy narzędzia rysującego prostokąty. Jest to narzędzie przydatne przy tworzeniu figur o prostych bokach i kątach 90 stopni pomiędzy nimi.



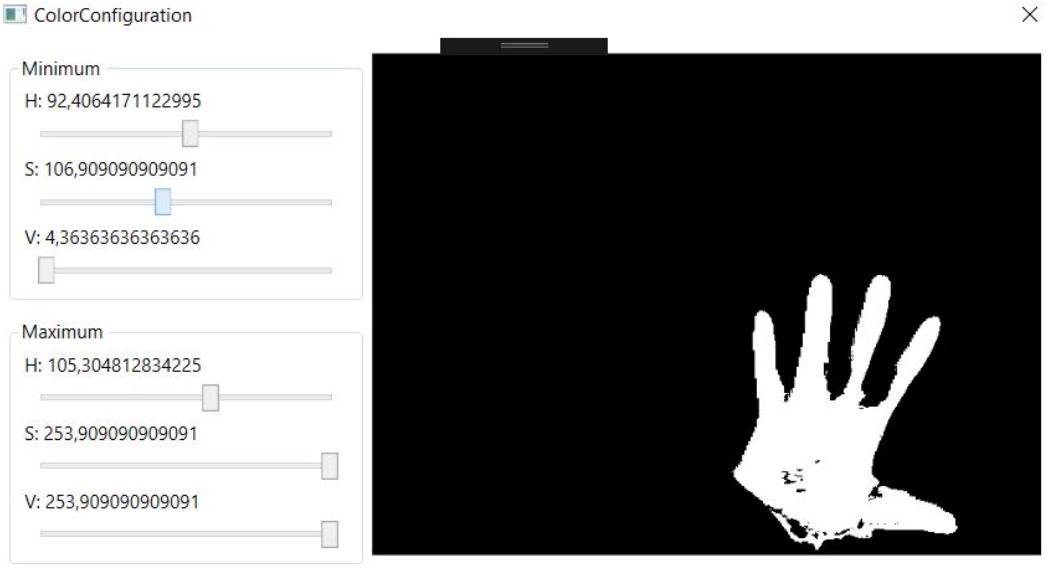
Rys. 12
Przykładowy rysunek przy wykonany przy pomocy narzędzia “elipsa”. Jak sama nazwa wskazuje możemy tworzyć przy jego pomocy elipsy. Figury bardzo trudne do osiągnięcia przy pomocy pędzla.



Rys. 13
Ostatnim narzędziem do rysowania jakie udostępnia nasza aplikacja jest linia. Pozwala ona na tworzenie odcinków poprzez wyznaczenie punktu początkowego i końcowego. Przy jego pomocy możemy rysować figury nieudostępnione jako przybory.



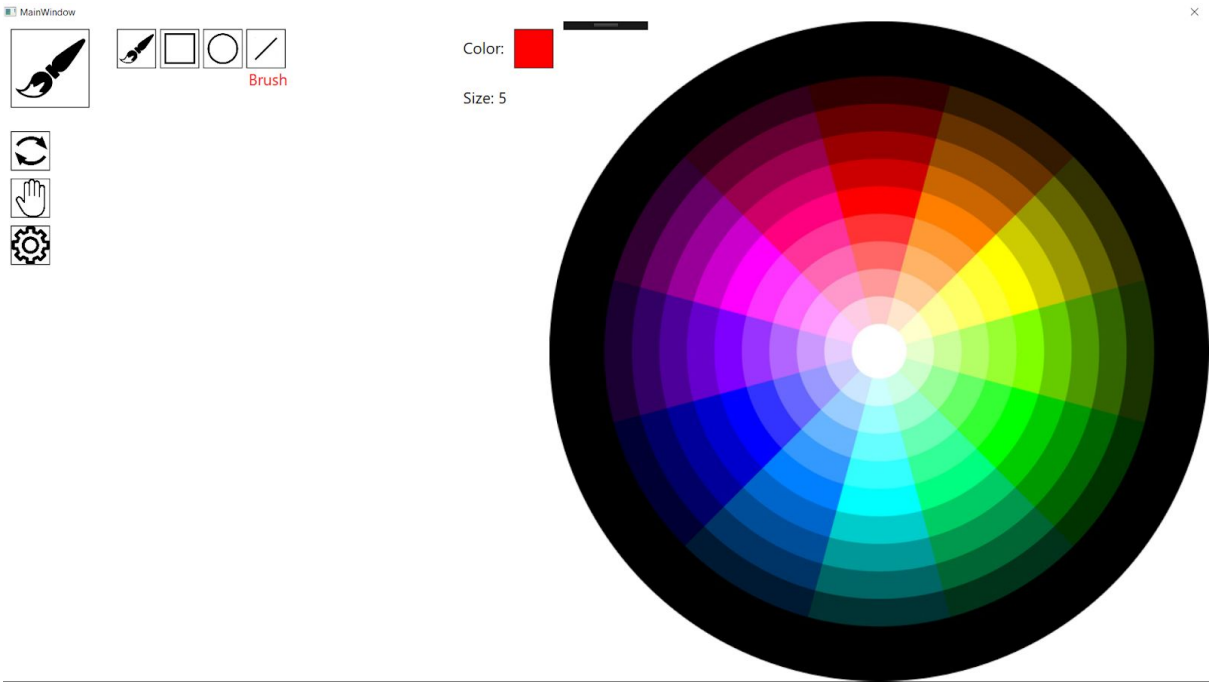
Rys. 14
Lewa dolna część paska narzędzi. Kafelki służą do konfiguracji wstępnej aplikacji, pierwszy od góry służy do otwierania nowego okna, w którym dobieramy współczynniki HSV dla rozpoznawania dłoni. Drugi służy do przechodzenia między trybem sterowania myszką i dłonią.



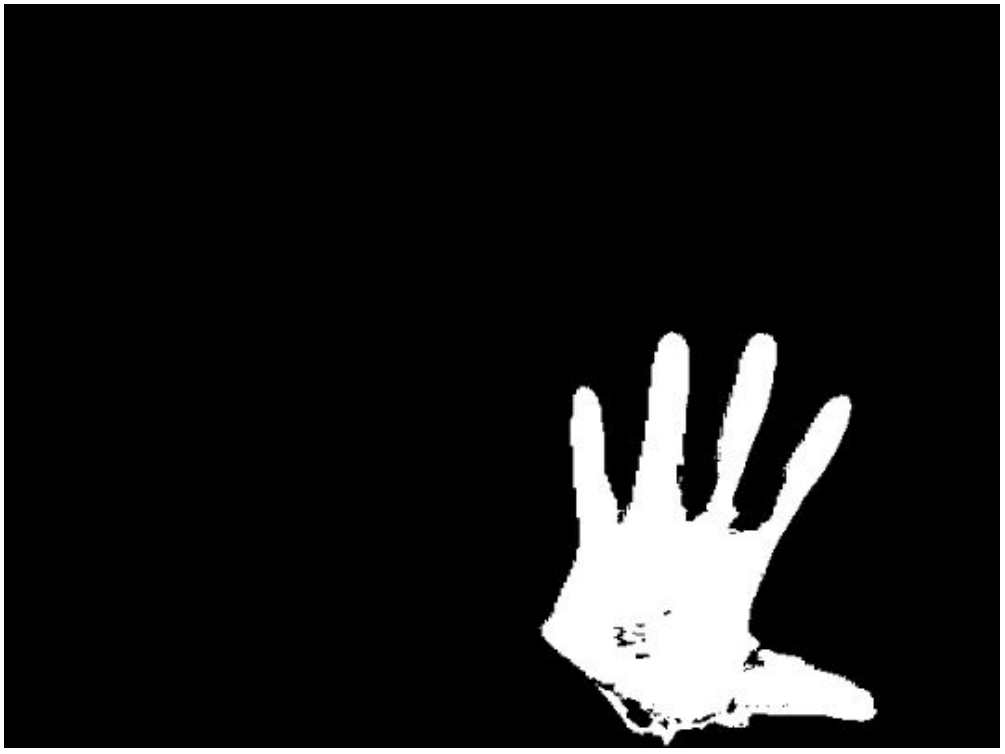
Rys. 15
W celu rozpoczęcia używania programu przy pomocy modułu rozpoznawania dłoni należy skonfigurować paletę kolorów HSV, aby rozpoznawała jedynie dłoń. Problematyczne mogą być kolory czerwone ze względu na zawieranie się ich na obu końcach suwaka.

Rys. 16
Widok aplikacji po wybraniu opcji białego tła. Ponowne wybranie tej opcji powoduje powrót do widoku obrazu z kamerki.

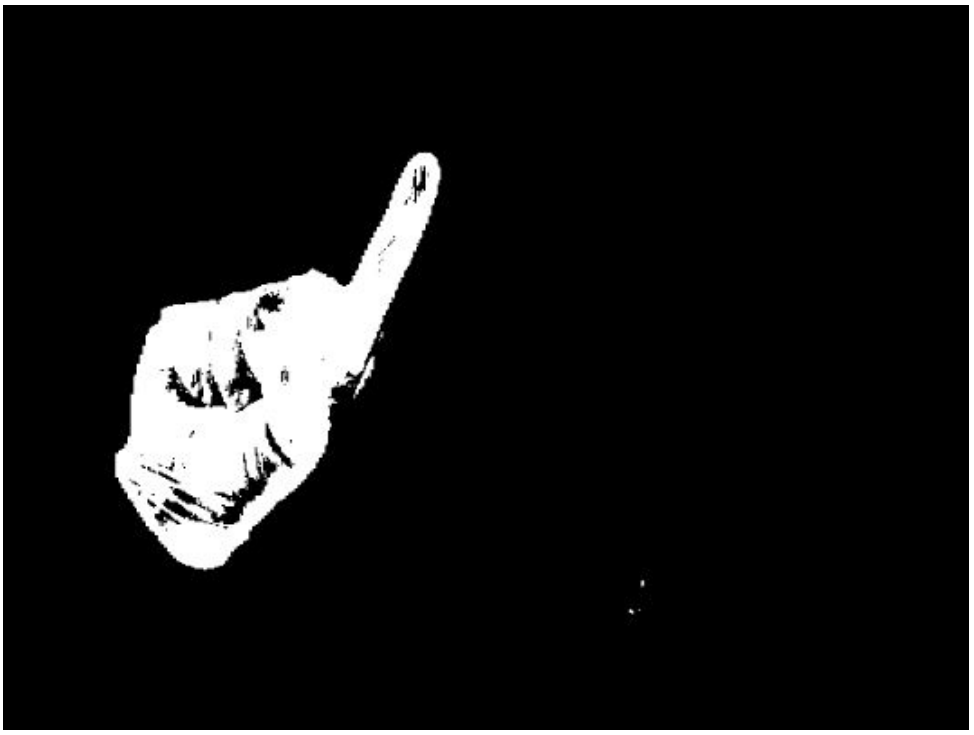
Rys. 17
Prawa część paska narzędzi. Koło pozwala na wybór koloru, po najechaniu na nie powiększa się. Przy napisie “Size” pojawia się wielkość narzędzia. Kafelek przy napisie “Color” przedstawia aktualnie wybrany kolor. Kafelek z ikoną dyskietki obsługuje zapisywanie wyniku prac, zaś w kształcie kartki i pędzelka służy do czyszczenia okna z tego co namalowaliśmy.



Rys. 18
Widok wyboru koloru. Koło palety kolorów powiększa się po najechaniu na nie, następnie w celu wybraniu koloru najeżdżamy na wybrany segment i wykonujemy gest rysowania lub klikamy myszką.



Rys. 19
Sterowanie odbywa się przy pomocy dłoni, rozróżniane są dwa gesty, pierwszy to dłoń z wyprostowanymi wszystkimi palcami. W momencie wykonywania tego gestu kursor się przemieszcza bez rysowania.



Rys. 20

Drugim jest dłoń z wyprostowanym palcem, niezależnie którym, szczyt wyprostowanego palca to miejsce, w którym znajduje się kursor. Ten gest jest znakiem dla programu, że chcemy zacząć rysować.

6. Uwagi końcowe

Projekt zajął nam trochę więcej czasu niż początkowo zakładaliśmy, lecz pomimo tego udało nam się dotrzymać terminów ustalonych na początku prac w harmonogramie. Prace postępowały w miarę sprawnie, czasem pojawiał się problemy, ale udawało się je usunąć w krótkim czasie. Najwięcej problemów sprawiło nam przygotowanie odpowiedniego modułu rozpoznawania dłoni.

Projekt posiada jeszcze wiele możliwości poprawy, na przykład można by usprawnić mechanizm rozpoznawania gestów, dodać rozpoznawanie gestów w oparciu o bazy danych, co poszerzyłoby funkcjonalności czy dodać obsługę wielu dłoni jednocześnie. Aplikacja jest na poziomie zadowalającym pod względem założeń początkowych projektu, pomimo małych zmian, które musieliśmy wprowadzić w celu ukończenia projektu.

Praca w grupie przebiegała sprawnie, wszyscy wywiązywali się z terminów w mniejszym lub znaczniejszym stopniu, ale w ogólnym zamyśle harmonogram pracy był utrzymywany. Czasem pojawiały się poślizgi ze względu na święta państwowe, na przykład 3 maja, ale nie powodowały one problemów przy pracy innym członkom grupy.

Praca nad tym projektem pozwoliła nam na lepsze poznanie tematyki systemów wizyjnych, ich obsługi oraz rozpoznawania części ciała przy pomocy kamerki internetowej. Temat po zagłębieniu okazał się bardzo ciekawy, a zarazem nie tak łatwy jak wydawało się nam na początku.

7. Bibliografia

[1] A. C. Pickering, „The search for a safer driver interface: a review of gesture recognition Human Machine Inteface.,” *Computing and Controling Engeering,* pp. 34-40, 25 Kwietnia 2005.

[2] R. Rosales, „Combining Generative and Discriminative Models in Framework for Articulated Pose Estimation.,” *Internal Journal of Computer Vision,* pp. 251-276, Maj 2006.

[3] B. Stenger, „Template-Based Hand Pose Recognition Using Multiple Cues,” *Computer Vison - ACCV,* pp. 551-560, 2006.

[4] E. Stergopoulou i N. Papamarkos, „Hand gesture recognition using a neural network shape fitting techniqu.,” *Engeering Applications of Artificial Intelligence,* pp. 1141-1158, Grudzień 2009.

[5] Y.-R. Wang, W.-H. Lin i L. Yang, „A novel real time hand detection based on skin-color,” w *2013 IEEE Interenational Symposium on Consumer Electronics (ISCE)*, Hsinchu, Taiwan, 2013.

[6] „it-consulting,” 12 czerwiec 2018. [Online]. Available: http://it-consulting.pl/autoinstalator/wordpress/tag/metodologie-analizy-i-projektowania/.