Malowanie za pomocą ruchu ciała

-Dokumentacja projektu

Czerwiec 2018

Grupa TI-1

Mateusz Pilarczyk Numer indeksu:112670

Spis treści

1

Opis zadania

Celem projektu jest napisanie programu, który umożliwi użytkownikowi ry- sowanie za pomocą ciała (bez urządzeń wskazujących).

2

Zasada działania

Zanim uruchomimy oprogramowanie należy się zaopatrzyć w niewielki przedmiot, najlepiej w innym kolorze niż otoczenie jakie będzie widoczne dla kamery. Przechwytywania zaczyna działać w momencie naciśnięcia przycisku start. Następnie należy zaznaczyć na obrazie z kamery za pomocą kursora myszy przedmiot, który trzymamy w ręce. Od tej chwili oprogramowanie zaczyna śledzić zaznaczony przedmiot. Przestrzeń kolorów każdej klatki obrazu zostaje przekonwertowana na HSV(Hue-barwa Saturation-nasycenie Value-wartość). W celu rozpoczęcia śledzenia obiektu użytkownik zaznacza myszką obiekt, który chcemy śledzić. Po wykonaniu tej czynności na bazie zaznaczonego obszaru zostaje utworzony histogram reprezentujący zakres barw wydzielonego obszaru. W nstępnej kolejności w oparciu o histogram zostaje utworzona mapa konturowa całego obrazu.W oparciu o mapę konturową zostaje wywołany algorytm camshift odpowiedzialny za określanie położenia śledzonego obiektu.

3

Założenia realizacyjne

Do zrealizowania projektu zostało użyte:

- Visual Studio 2015,
- OpenCV 3.1,
- Język programowania: C++.
- Biblioteka Windows forms

Powyższe narzędzia wybrałem ze względu na znajomość środowiska Visual Studio i języka C++. Dodatkowo projekt został wykonany z pomocą biblioteki Windows forms co umożliwiło stworzenie prostego i przejrzystego interfejsu graficznego umożliwiając proste użytkowanie programu. OpenCV posiada już skompilowane biblioteki, które można dołączyć do projektu w Visual Studio co w znaczący sposób ułatwiło i przyśpieszyło pracę. Dodatkowo biblioteki te pozwoliły na zrealizowane potrzebnych nam funkcjonalności takich jak:

- przechwytywanie obrazu z kamery,
- przetwarzanie przechwyconego obrazu,
- Operacje morfologiczne,

Projekt został oparty na wyodrębnianiu kolorów z zarejestrowanego przez kamerę obrazu, a następnie śledzenie zaznaczonego obiektu. Po przechwyceniu obrazu przez program zostaje on przekonwertowany z formatu RGB na HSV. Konwersja ta pozwala w łatwy sposób operować na zakresie akceptowalnych barw ponieważ wszystkie znajdują się na jednej warstwie obrazu i są oznaczone wartościami od 0 do 360. Do redukcji szumu występującego w obrazie stosujemy filtry:

- Filtr gaussa: Jest to operacja rozmywania pomagająca usunąć zakłócenia na obrazie typu.

- Erode (erozja - usuwa ostre krawędzie): Erozja jest techniką pomniejszania obiektów poprzez usunięcie pikseli granicznych. Tą operację można rozumieć jako odwrócenie operacji dylacji.

- Dylacja: Dylacja jest techniką zwiększenia / rozszerzania obiektów, zwykle we wszystkich kierunkach jednocześnie

4

Wymagania

Do uruchomienia oprogramowania wymagana jest:

- OpenCV 3.1
- Komputer z systemem operacyjncym Windows

5

Instrukcja użytkownika

5.1 Potrzebne urządzenia

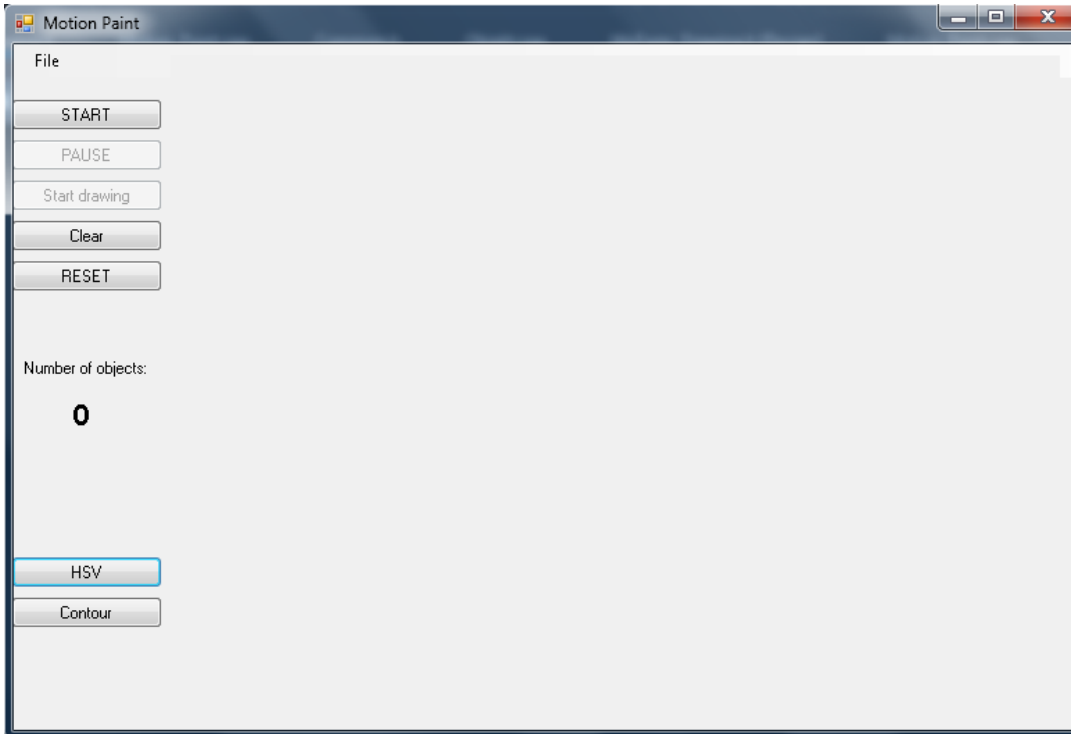
- Komputer z wgranym oprogramowaniem,
- Kamera,
- Klawiatura,

5.2 Stanowisko pracy

Oprogramowanie najlepiej radzi sobie z śledzeniem obiektu gdy ten wyróżnia się kolorem w obrazie przechwyconym przez kamerę. Najlepszy efekt został otrzymany w pomieszczeniu w którym panuje półmrok, a obiekt świecił na zielono.

5.3 Korzystanie z oprogramowania

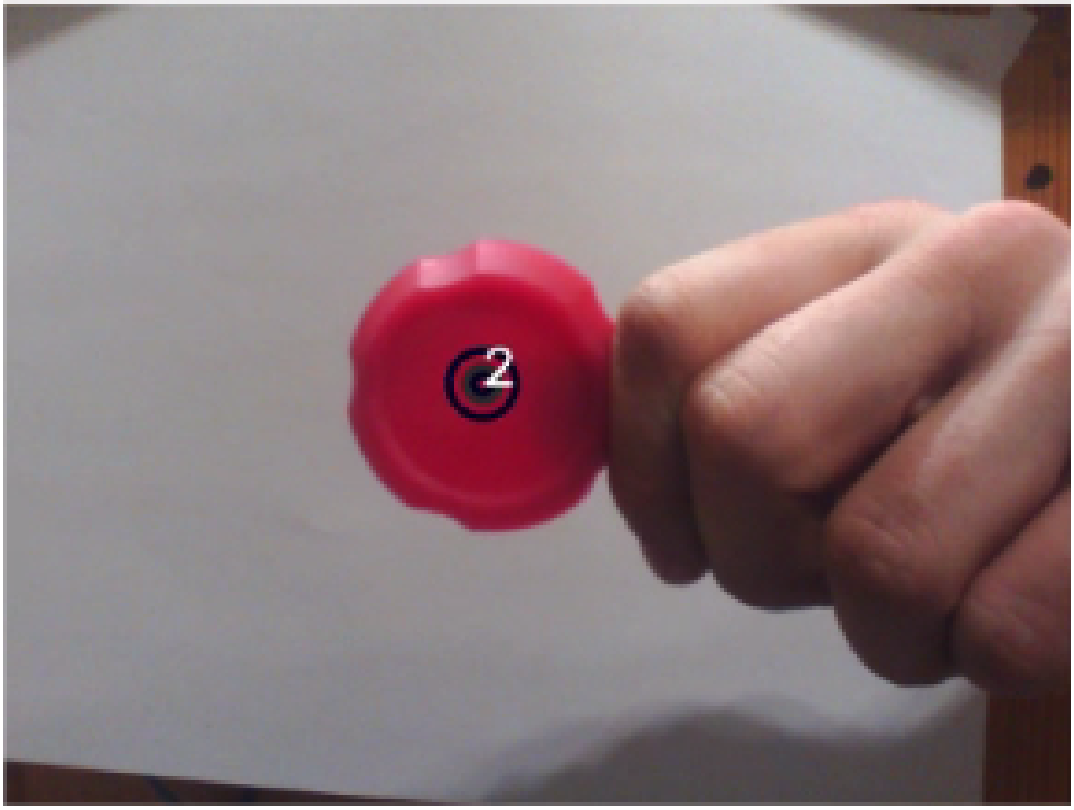
- Po uruchomieniu oprogramowania na ekranie wyświetli się okno główne aplikacji
- W celu rozpoczęcia przechwytywania obrazu należy należy nacisnąć przycisk start znajdujący się po lewej stronie w oknie aplikacji:



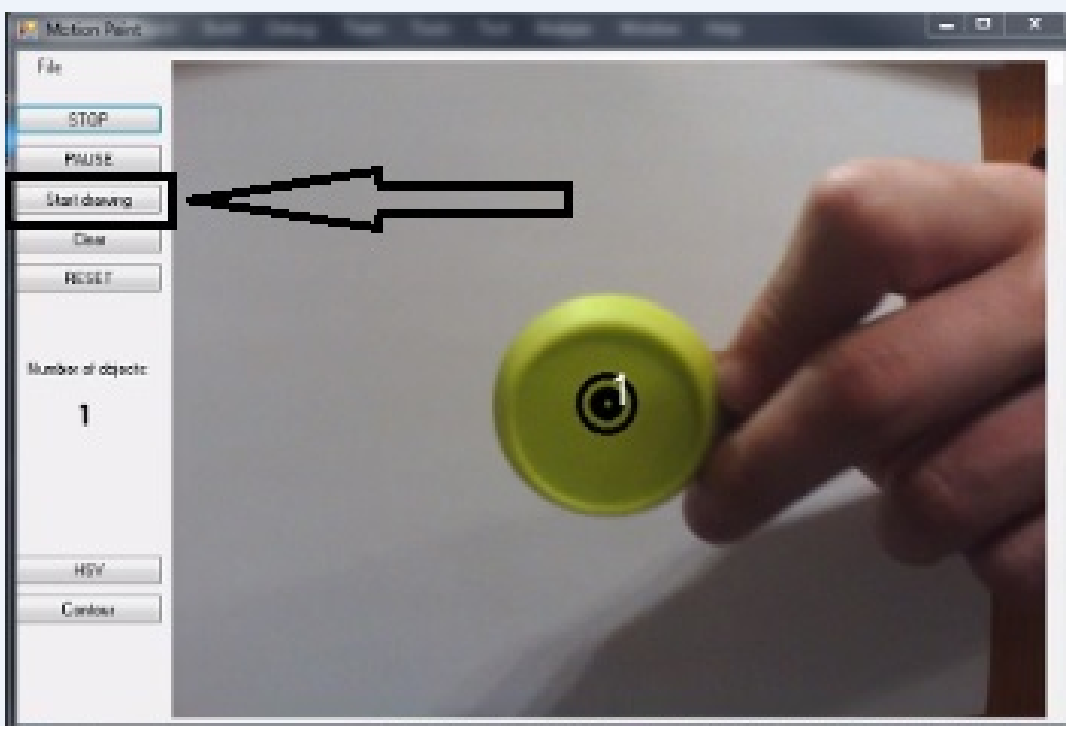
- Aby zacząć śledzić obiekt należy w oknie z przechwytywanym obrazem należy zaznaczyć kursorem obiekt, który chcemy śledzić.
- Na przechwytywanym obrazie, na zaznaczonym obiekcie powinien pojawić okrąg wraz z identyfikatorem śledzonego obiektu . Jest to wirtualny kursor informujący nas w jakim położeniu zaczniemy rysować. Jeżeli kursor się nie pojawił należy powtórzyć czynność.

- 

- Po zaznaczeniu obiektu myszką po lewej stronie pojawia się informacja ile obecnie obiektów jest śledzonych.
- W celu śledzenia więcej niż jednego obiektu równocześnie należy powtórzyć czynność w poprzednim punkcie. Zaleca się żeby kolejny obiekt do śledzenia był innego koloru niż poprzedni.

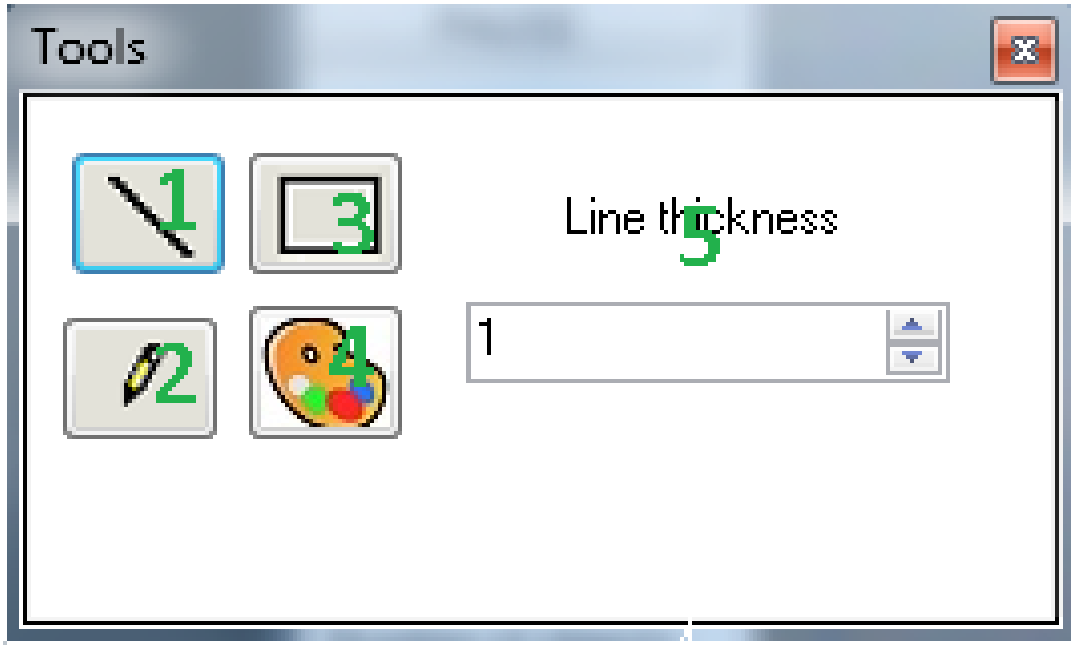
- 

- W celu rozpoczęcia rysowania należy nacisnąć przycisk Start drawing:

- 

- Po wykonaniu akcji pojawi się dodatkowe białe okienko pełniące funkcję planszy do rysowania oraz okienko z interfejsem wyboru narzędzia do rysowania

- Pojawi się dodatkowe białe okienko pełniące funkcję planszy do rysowania oraz okienko z interfejsem wyboru narzędzia do rysowania



- Opis interfejsu z narzędziami do rysowania:

1. Rysowanie linii
2. Standardowe malowanie na obrazie
3. Rysowanie prostokąta
4. Wybór koloru rysowanej linii
5. Wybór grubości rysowanej linii

5.4 Okno z ustawieniami

Suwaki:

- Thresh lb - dolny zakres akceptowanych kolorów (zmiany można podejżeć po wciśnięciu H),
- Thresh ub - górny zakres akceptowanych kolorów (zmiany można podejżeć po wciśnięciu H),
- Blur - filtr powodujący rozmycie obrazu,
- Dilate - filtr powodujący “rozszerzanie”,
- Erode - filtr powodujący usuwanie ostrych krawędzi,

5.5 Opis implementacji

- Implementacja operacji morfologicznych

```
1   void Operation_filter_Blur(cv::Mat &thresh, int i) {
        if (obiekt.at(i).blur < 1)
3       {
            obiekt.at(i).blur = 1;
5       }
        cv::GaussianBlur(thresh, thresh, cv::Size(2 * obiekt.at(i).blur +
     1, 2 * obiekt.at(i).blur + 1), 0.3 *(((2 * obiekt.at(i).blur + 1) -
     1)*0.5 - 1) + 0.8);
7     }
      void Operation_filter_Erode(cv::Mat &thresh, int i) {
9       if (obiekt.at(i).erode < 1)
        {
11          obiekt.at(i).erode = 1;
        }
13      cv::Mat erodeElement = cv::getStructuringElement(cv::MORPH_RECT,
     cv::Size(obiekt.at(i).erode, obiekt.at(i).erode));

15      erode(thresh, thresh, erodeElement);
      }
17    void Operation_filter_Dilate(cv::Mat &thresh, int i) {
        if (obiekt.at(i).dilate < 1)
19      {
            obiekt.at(i).dilate = 1;
21      }
        cv::Mat dilateElement = cv::getStructuringElement(cv::MORPH_RECT,
     cv::Size(obiekt.at(i).dilate, obiekt.at(i).dilate));
23
        dilate(thresh, thresh, dilateElement);
25    }

27    void Operation_filter(cv::Mat &thresh, int i) {
        Operation_filter_Dilate(thresh, i);
29      Operation_filter_Erode(thresh, i);
        Operation_filter_Blur(thresh, i);
31    }
```

- Funkcja redukucjąca drganie

```
1 bool Drawing_Radius_move(Drawing_Position actualposition, int
    movestabilize, int Radiustabilize)
      {
3       bool stabilize = false;
```

```
        if (abs(actualposition.point.x - position.point.x) >
    movestabilize)
        {
            position.point.x = actualposition.point.x;
            stabilize = true;
        }
        if (abs(actualposition.point.y - position.point.y) >
    movestabilize)
        {
            position.point.y = actualposition.point.y;
            stabilize = true;
        }
        if (abs(actualposition.r - position.r) > Radiustabilize)
        {
            position.r = actualposition.r;
            stabilize = true;
        }
        return stabilize;
      }
```

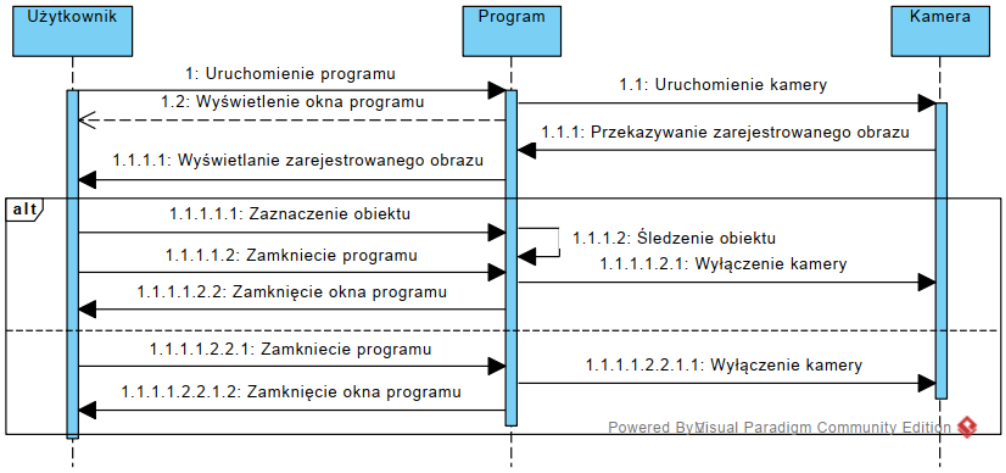
- Inicjalizacja przechwytywania video

```
mat_card = cv::Mat(480, 640, CV_8UC3, cv::Scalar(255, 255, 255));

        button_pauses->Enabled = true;

        Operation_Deactivate();
        Is_Original_active = true;
        button_start->Text = "STOP";
        Is_start_active = true;
        capture.release();
        capture.set(CV_CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
        capture.set(CV_CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
        Operation_Deactivate();
        if (video == "0")capture = cv::VideoCapture(0);
        else capture = cv::VideoCapture(video);


        Image_Original->Show();
        obiekt.clear();
        Timer_Capture->Start();
```

- Fragment kodu odpowiedzialny za wyznaczanie położenia obiektu do śledzenia

```
1
cv::inRange(mat_hsv_split[0], obiekt.at(i).Hmin, obiekt.at(i).Hmax, obiekt
    .at(i).mat_contour);
3                Operation_filter(obiekt.at(i).mat_contour, i);
                 cv::calcBackProject(&mat_hsv_split[0], 1, 0, obiekt.at(i).
    gethistogram(), obiekt.at(i).mat_backproj, &histogram_pointer_Zasieg);
5                obiekt.at(i).mat_backproj &= obiekt.at(i).mat_contour;
                 obiekt.at(i).setRectangle_tracked(cv::CamShift(obiekt.at(i)
    .mat_backproj, obiekt.at(i).rectangle
7                    , cv::TermCriteria(cv::TermCriteria::EPS | cv::
    TermCriteria::COUNT, 10, 5)));
                 obiekt.at(i).setTracking_Points(Drawing_Position(obiekt.at(
    i).getRectangle_tracked().center, Drawing_Radius_get(obiekt.at(i).
    getRectangle_tracked().size.width, obiekt.at(i).getRectangle_tracked()
    .size.height)));
```

5.6 Diagram UML



-

6

Harmonogram pracy

	Początek	Koniec	Liczba dni
Zapoznanie się z OpenCV	3-03	14-03	11
Integracja z windows forms	15-03	27-03	13
Implementacja wykrywania i śledzenia więcej niż jednego obiektu równocześnie	28-03	11-04	16
Testowanie aplikacji	12-04	26-04	15
Implementacja dodatkowych funkcji	27-04	10-05	15

7

Dlaczego temat został wybrany

Temat projektu został wybrany ,ponieważ biblioteka OpenCV oferuje bardzo duże spektrum możliwości w przetwarzaniu obrazu. Projekt implementuje śledzenie obiektu w związku z tym istnieje bardzo szeroki zakres rozbudowy programu co umożliwia wykorzystanie aplikacji.