

Ćwiczenie 5

ŚLEDZENIE RUCHU ZA POMOCĄ PRZEPLYWU OPTYCZNEGO

Zakres pracy

W ramach ćwiczenia należy do dostarczonego interfejsu, umożliwiającego wyświetlanie obrazów przechwytywanych z kamery, dodać możliwość śledzenia ruchu przy użyciu dwóch algorytmów:

1. przepływu optycznego wykorzystującego metodę Farnebacka (jest to tzw. gęsty przepływ optyczny, w którym wektor ruchu wyznaczany jest dla każdego piksela),
2. przepływu optycznego wykorzystującego metodę Lucasa-Kanade (najpierw raz znajdujemy na obrazie punkty, które da się śledzić z dużą dokładnością, następnie na każdej nadchodzącej klatce wyznaczamy nowe położenia tych punktów, stosując dodatkowo piramidę rozdzielczości, aby zwiększyć precyzję obliczeń).

W przypadku 1 na obraz z kamery należy nakładać:

- wektory reprezentujące kierunki ruchu (z krokiem 8 pikseli wzdłuż każdej osi),
- piksele, w których wykryto niezerowy ruch (intensywność koloru ma być proporcjonalna do długości wektora ruchu).

W przypadku 2 na obrazie z kamery należy zaznaczać aktualne położenia punktów charakterystycznych oraz ślad ich ruchu z 10 ostatnich klatek (intensywność śladu powinna maleć z upływem czasu). Należy również sprawdzać skuteczność śledzenia (tzw. status) i w przypadku problemów ponownie uruchamiać funkcję wyszukującą punkty.

Wskazówki implementacyjne

1.

Do realizacji zadania wykorzystujemy bibliotekę OpenCV.

Kamery dostępne w laboratorium zwracają obrazy o rozdzielczości **640×480**.

Zasadniczą część implementacji należy zawrzeć w kodzie metody

```
COpticalFlowDlg::OnTimer .
```

W wersji *Debug* działanie programu może być bardzo spowolnione, należy więc przygotować również wersję *Release*.

2.

Utworzenie pełnej kopii macierzy **A** i przypisanie jej do zmiennej **B** (podstawienie **A = B** nie przeprowadza kopiowania danych) :

```
Mat B = A.clone();
```

Zamiana macierzy **C** zawierającej obraz kolorowy na macierz **M** reprezentującą obraz w odcieniach szarości (**M** nie musi być zainicjowana):

```
cvtColor(C, M, CV_BGR2GRAY);
```

Utworzenie punktu o współrzędnych zmiennoprzecinkowych $\{x, y\}$:

```
Point2f pt = Point2f(x, y);
```

Narysowanie na obrazie (macierzy) `mObr` czerwonej linii o grubości 1 piksela, przebiegającej od punktu `p0` do punktu `p1` (`p0` i `p1` mogą być typu `Point2f`):

```
line(pImage, p0, p1, CV_RGB(255,0,0), 1);
```

3.

Funkcje używane do wyliczania przepływu optycznego zwracają współrzędne wektorów ruchu oraz śledzonych punktów w postaci macierzy, których elementy można traktować jako obiekty typu `Point2f`.

Należy pamiętać, że jeśli dane opisujące wartości przepływu optycznego dla poszczególnych pikseli zawarte są w obiekcie `Mat`, to chcąc pobrać informacje dotyczące piksela o współrzędnych $\{x, y\}$ musimy odwołać się do elementu macierzy o indeksach (y, x) . Natomiast rysując linię na obiekcie typu `Mat` współrzędne punktów podajemy tak jak dla normalnego obrazu.

Do zaimplementowania śladów ruchu przydatne mogą okazać się szablony STL `vector` i `queue`.

4.

Funkcja obliczająca przepływ optyczny metodą Farnebacka:

```
void calcOpticalFlowFarneback(  
    InputArray prev,           //poprzednia klatka (obiekt Mat)  
    InputArray next,          //aktualna klatka (obiekt Mat)  
    InputOutputArray flow,    //WYJŚCIE: wektory ruchu (obiekt Mat)  
    double pyr_scale,         //mnożnik piramidy rozdzielczości (podać 0.5)  
    int levels,                //liczba poziomów piramidy (podać 3)  
    int winsize,              //rozmiar okna uśredniania (podać 15)  
    int iterations,           //liczba iteracji na jednym poziomie (podać 3)  
    int poly_n,                //rozmiar bloku wokół piksela (podać 5)  
    double poly_sigma,        //stopień wygładzania (podać 1.2)  
    int flags)                 //dodatkowe flagi (podać 0)
```

- Obie analizowane klatki powinny być obrazami 8-bitowymi 1-kanalowymi (w odcieniach szarości).
- Macierz `flow` nie musi być inicjowana. Jej rozmiar będzie taki sam jak rozmiar `prev`.

5.

Funkcja wyznaczająca punkty, które można śledzić z dużą dokładnością (narożniki):

```
void goodFeaturesToTrack(  
    InputArray image,          //obraz źródłowy (8-bitów na p., 1 kanał)  
    OutputArray corners,      //WYJŚCIE: macierz punktów charakterystycznych  
    int maxCorners,           //maksymalna liczba punktów do znalezienia  
    double qualityLevel,      //minimalna jakość punktu (ustawić 0.01)  
    double minDistance)       //minimalna odległość punktów (podać 10)
```

Funkcja obliczająca przepływ optyczny metodą Lucasa-Kanade z wykorzystaniem piramid rozdzielczości:

```
void calcOpticalFlowPyrLK(  
    InputArray prev,           //poprzednia klatka (obiekt Mat)  
    InputArray next,          //aktualna klatka (obiekt Mat)
```

```
InputArray prevPts,           //poprzednie położenia punktów charakt.  
InputOutputArray nextPts,    //WYJŚCIE: nowe położenia punktów  
OutputArray status,          //WYJŚCIE: status śledzenia punktów  
OutputArray err,             //WYJŚCIE: błędy śledzenia punktów  
Size winSize)                //rozmiar bloku - podać Size(16,16)
```

- Zmienne (macierze) wyjściowe nie muszą być inicjowane.
- W jednokolumnowej macierzy `status` wartość 1 oznacza, że udało się znaleźć nowe położenie punktu, 0 wskazuje na błąd śledzenia.
- Elementy jednokolumnowej macierzy `err` zawierają wartości błędów śledzenia poszczególnych punktów.