|  |  |  |  |
| --- | --- | --- | --- |
| WZWI - projekt |  | | |
| Projekt | Wykrywanie i śledzenie ruchu obiektów zarejestrowanych kamerą szybką z wykorzystaniem języka Python | | |
| Grupa lab.: 2 | 1. Zuzanna Heller 2. Kamil Kondziołka | | |
| Zespół: |
| Data wykonania ćwiczenia: | | Data oddania sprawozdania: | Ocena: |

**WSTĘP**

**Wyjaśnienie pojęcia**

Wykrywanie obiektów (ang. *object detection*) polega na znalezieniu obiektu na obrazie i wskazanie jego położenia za pomocą np. markera lub prostokąta (ang. *bounding box*). Jest ono potrzebne do wskazania przedmiotów do śledzenia.

Śledzenie ruchu (ang. *motion tracking*, *video tracking*) to monitorowanie zmiany w położeniu danego obiektu. Umożliwia obserwację poruszania się obiektu i jego analizę lub nagranie i późniejsze wykorzystanie.

Zastosowanie kamery szybkiej pozwala na uchwycenie krótkich, często niewidocznych dla oka zmian w jakimś zjawisku – w tym przypadku w ruchu obiektu – dzięki przechwytywaniu wysokiej ilości klatek na sekundę.

Śledzenie odbywa się za pomocą uprzedniego wykrycia i oddzielenia obiektu od tła przy pomocy algorytmu Canny’ego do detekcji krawędzi. Na każdej klatce nagrania przeprowadzane są kolejne kroki algorytmu, które tworzą obraz o czarnym tle i białych obrysach śledzonego przedmiotu. Następnie wokół krawędzi rysowany jest prostokąt wskazujący położenie przesuwającego się przedmiotu.

**Rodzaje urządzeń**

Istnieje szeroka gama modeli kamer szybkich, które różnią się parametrami nagrania (maksymalne rozdzielczości, czasy nagrań, ilość klatek na sekundę), rozmiarem czy dostosowaniem do określonych warunków pracy (np. wzmocnione kamery do zastosowań przemysłowych).



Rysunek 1. Kamera szybka z możliwością nagrywania w 4K

Źródło: <https://www.ects.pl/produkt/kamera-szybka-phantom-flex4k/>

Nowoczesne technologie pozwalają na tworzenie kamer o niekiedy ekstremalnych parametrach.



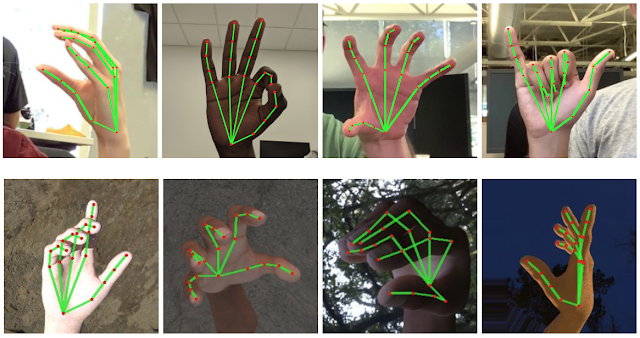
Rysunek 2. Niewielka kamera szybka o wymiarach 32mm x 32mm x 28mm

Źródło: <https://www.ects.pl/produkt/kamera-szybka-phantom-miro-n5/>

**Wybrane sposoby detekcji obiektu**

Śledzenie ruchu przedmiotów jest ściśle powiązane z detekcją przedmiotów. Nagranie jest cyklem zarejestrowanych klatek, dlatego komputer w rzeczywistości nie obserwuje faktycznego ruchu przedmiotu, lecz porównuje jego położenie między kolejnymi klatkami. Dlatego program śledzący powinien być w stanie rozpoznać interesujący nas przedmiot na każdej klatce, na której się pojawia.

Można wykorzystać do tego sztuczną inteligencję, która po odpowiednim „wyszkoleniu” danymi przykładowymi będzie mogła wyróżnić podobny obiekt w obrazie. Wiąże się to z pojęciem rozpoznawania przedmiotów (ang. *object recognition*), które oznacza rozpoznanie i opisanie nie gdzie, a co znajduje się na obrazie.



Rysunek 3. Rozpoznawanie i śledzenie ruchu dłoni

Źródło: <https://google.github.io/mediapipe/solutions/hands.html>

Do śledzenia i często rejestrowania ruchu (ang. *motion capture*) wykorzystuje się specjalne markery, które rozpoznawane są przez przetwarzający nagranie program a następnie aplikowane na wybranym modelu 3D.



Rysunek 4. Motion capture z użyciem markerów

Źródło: https://www.vulture.com/2018/12/andy-serkis-on-his-breakthrough-motion-capture-performances.html

W przypadku statycznego ustawienia kamery oraz jednolitego tła wykorzystać można techniki wykrywania krawędzi. Polegają one na wyodrębnieniu zarysu przedmiotu poprzez odpowiednie przekształcenia matematyczne. Otrzymany obraz może zostać użyty do określenia położenia przedmiotu.



Rysunek 5. Obraz przed i po zastosowaniu algorytmu wykrywania krawędzi

Źródło: <https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_Image_Canny_Edge_Detection.php>

Wszystkie metody mają swoje zalety, wady i ograniczenia; śledzenie markerów niekiedy wymaga odpowiedniego do tego sprzętu, sztuczna inteligencja może mieć problemy z rozpoznaniem przedmiotu zniekształconego lub częściowo zakrytego a wykrywanie krawędzi wymaga odpowiedniego przygotowania warunków nagrania.

**Znaczenie metody oraz jej zastosowanie**

Technologia śledzenia ruchu wykorzystywana jest w wielu dziedzinach, od medycyny po branżę rozrywkową.

Jednym z najczęściej spotykanych jej zastosowań są kamery monitoringu (śledzenie ruchu ulicznego, przechodniów, ewentualnych intruzów na danej posiadłości). Oprócz nich możemy wyróżnić również analizy sportowe (śledzenie poruszania się i rozmieszczenia graczy na boisku, analiza ruchu sportowca w celu oceny jego pracy) czy medyczne (rozpoznawanie ruchu tkanek czy ciał obcych w organizmie).

Śledzenie ruchu wykorzystywane jest również przy obróbce filmów (dodawanie efektów specjalnych) czy wymienionym wcześniej *motion capture* (nagranie ruchu aktora w celu animacji postaci na potrzeby m.in. filmów czy gier).

**Wyjaśnienie celu projektu**

Celem projektu jest obserwacja ruchu obiektów w zwolnionym tempie oraz śledzenie ich przy pomocy obramowania (*bounding box*).

Badanie takie może wskazać m.in. niedoskonałości badanego przedmiotu dzięki wykorzystaniu zwolnionego tempa i ujawnienie niewidocznych gołym okiem drobnych zmian w trajektorii ruchu.

**CZĘŚĆ DOŚWIADCZALNA**

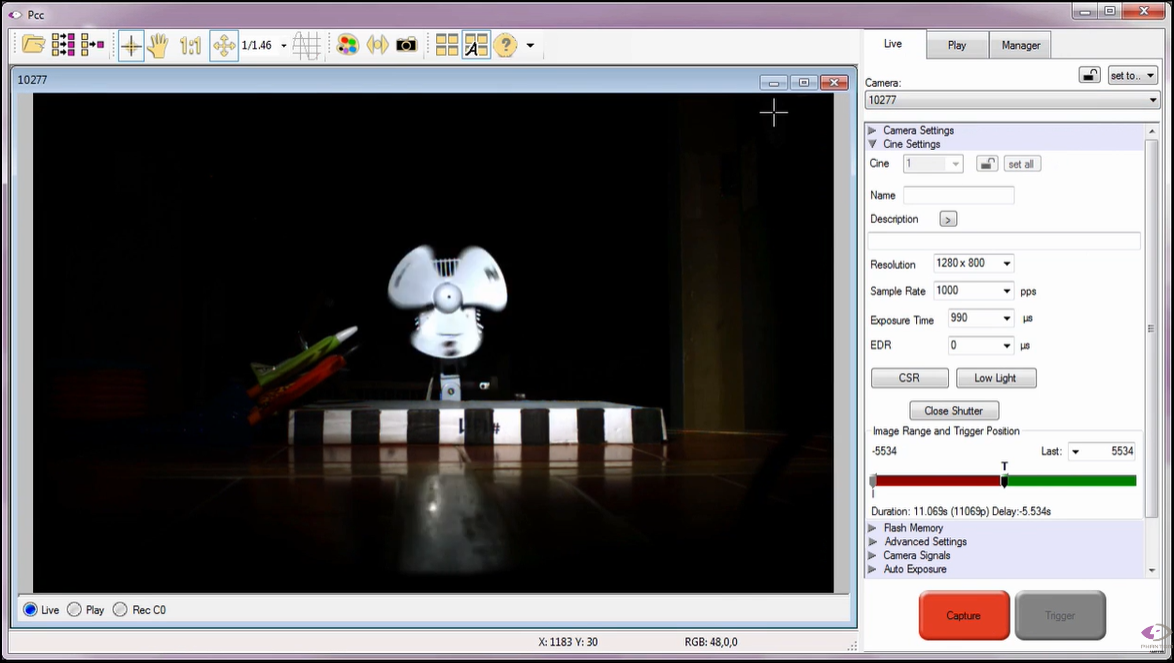
**Ogólny opis doświadczenia**

Kamerą szybką Phantom Miro M310 zarejestrowano ruch dwóch przedmiotów: piłki do tenisa stołowego oraz dwuzłotówki. Do detekcji i śledzenia napisane zostały skrypty w języku Python.

Nagrania przetwarzane są najpierw przez skrypt wykonujący algorytm wykrywania krawędzi Canny’ego. Skrypt tworzy wersję nagrania z krawędziami, które może zostać następnie przetworzone przez skrypt rysujący *bounding box* wokół krawędzi.

**Zarejestrowanie ruchu przedmiotów**

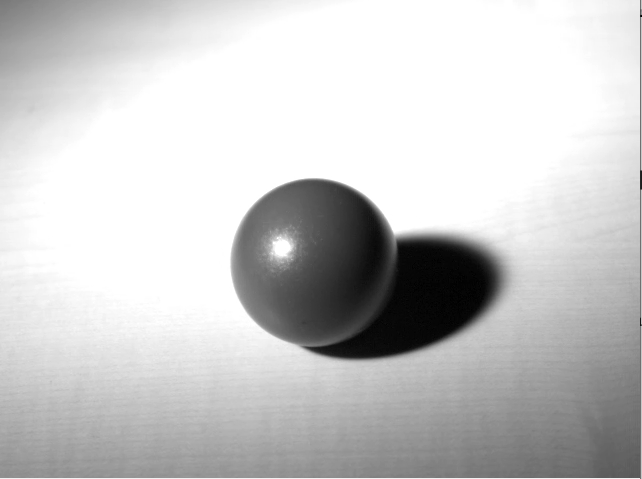
Do nagrań wykorzystano kamerę szybką Phantom Miro M310 dostępną na uczelni, podpiętą do laptopa. Nagrywaniem sterowano za pomocą dedykowanego oprogramowania od producenta.



Rysunek 6. Interfejs oprogramowania sterującego nagrywaniem

Źródło: <https://youtu.be/VXGDpxqZHX8?list=PL0TccwBnZqgQySikYXrLWq8ahv7eFJWyb>

Nagrano niewielki obszar biurka oświetlony reflektorem o dużej mocy. Nagrania wyeksportowano do formatu .mp4, w formacie 640x480, 30fps.



Rysunek 7. Nagranie ruchu piłki

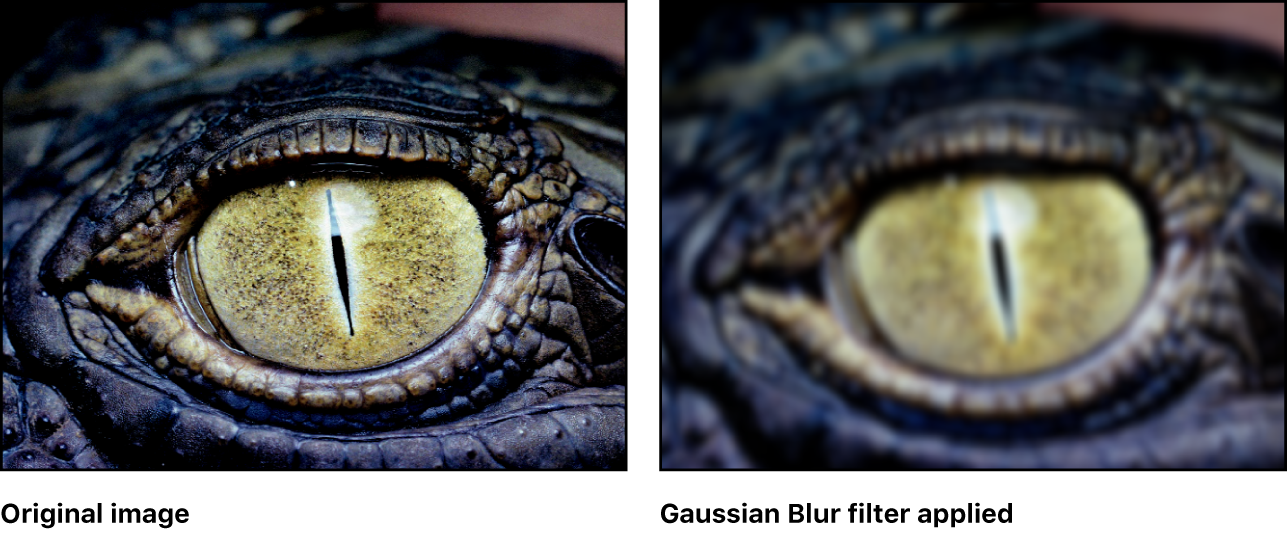


Rysunek 8. Nagranie ruchu monety

**Detekcja krawędzi Canny’ego**

Algorytm wykrywania krawędzi stworzony przez Johna F. Canny’ego w 1986 r. na wejściu przyjmuje obraz w szarościach i podzielony jest na kilka etapów:

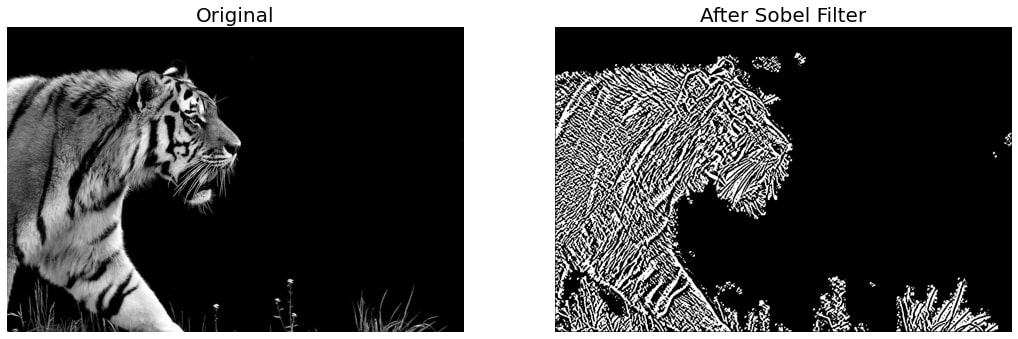
1. **Nałożenie na obraz filtru Gaussa** w celu rozmycia go i usunięcia szumów



Rysunek 9. Obraz przed i po rozmyciu filtrem Gaussa

Źrodło: <https://support.apple.com/pl-pl/guide/motion/motn169fe6e6/mac>

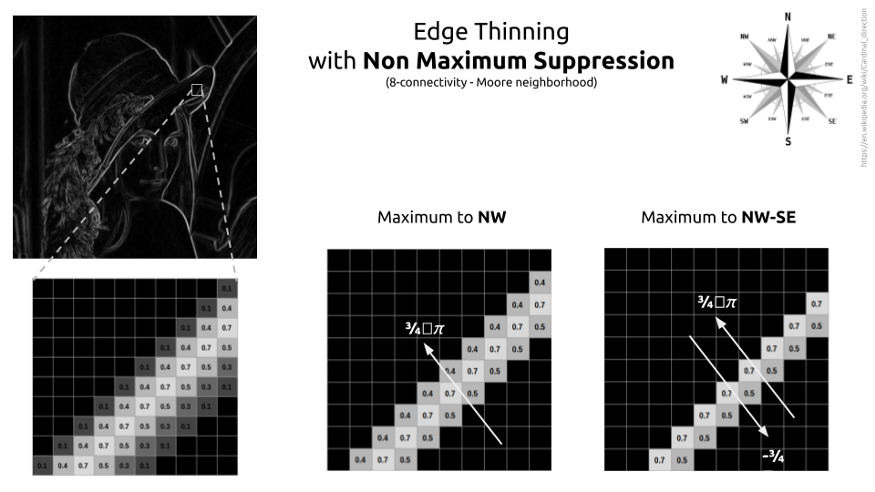
1. **Zastosowanie filtrów konturowych Sobela**, które obliczają pionową i poziomą mapę gradientów obrazu (zmiany natężeń pikseli); po ich połączeniu otrzymuje się obraz z konturami



Rysunek 10. Obraz przed i po zastosowaniu filtrów Sobela

Źródło: <https://learnopencv.com/edge-detection-using-opencv/>

1. **Obliczenie kierunków krawędzi**; otrzymuje się mapę kierunków w jakich skierowana jest krawędź dla każdego piksela
2. **Eliminacja niemaksymalnych pikseli** (ang. *non-maximum suppression*); porównywane są sąsiednie piksele o tym samym kierunku i zostawiany jest jedynie ten o najsilniejszym natężeniu



Rysunek 11. Przedstawienie działania NMS

Źródło: <https://towardsdatascience.com/implement-canny-edge-detection-from-scratch-with-pytorch-a1cccfa58bed>

1. **Wykrywanie krawędzi „słabych” i „mocnych”** (ang. *double threshold*); po ustaleniu górnego i dolnego progu natężenia dla każdego piksela przypisywana jest wartość silna lub słaba: jeśli piksel ma natężenie większe od progu górnego, przypisywana jest mu wartość silna, jeśli znajduje się pomiędzy progami – wartość słaba, jeśli jest poniżej dolnego progu – piksel uznawany jest jako nieznaczący



Rysunek 12. Obraz z jedynie dwoma wartościami pikseli – „mocnymi” i „słabymi”

Źródło: <https://justin-liang.com/tutorials/canny/>

1. **Konstrukcja krawędzi**; następuje sprawdzenie, czy słabe piksele są wystarczająco znaczące aby uznać je za część krawędzi – następuje to poprzez szukanie silnych pikseli wokół danego słabego: jeśli choć jeden piksel sąsiedni jest silny, piksel słaby jest kwalifikowany jako krawędź.



Rysunek 13. Obraz po wyszukaniu wszystkich znaczących krawędzi

Źródło: <https://justin-liang.com/tutorials/canny/>

**SKRYPT WYKONUJĄCY WYKRYWANIE KRAWĘDZI (main.py)**

**Parametry wejściowe:** Nagranie poruszającego się przedmiotu w szarościach

**Wynik:** Nowe nagranie będące przetworzoną wersją nagrania wejściowego z wykrytymi krawędziami

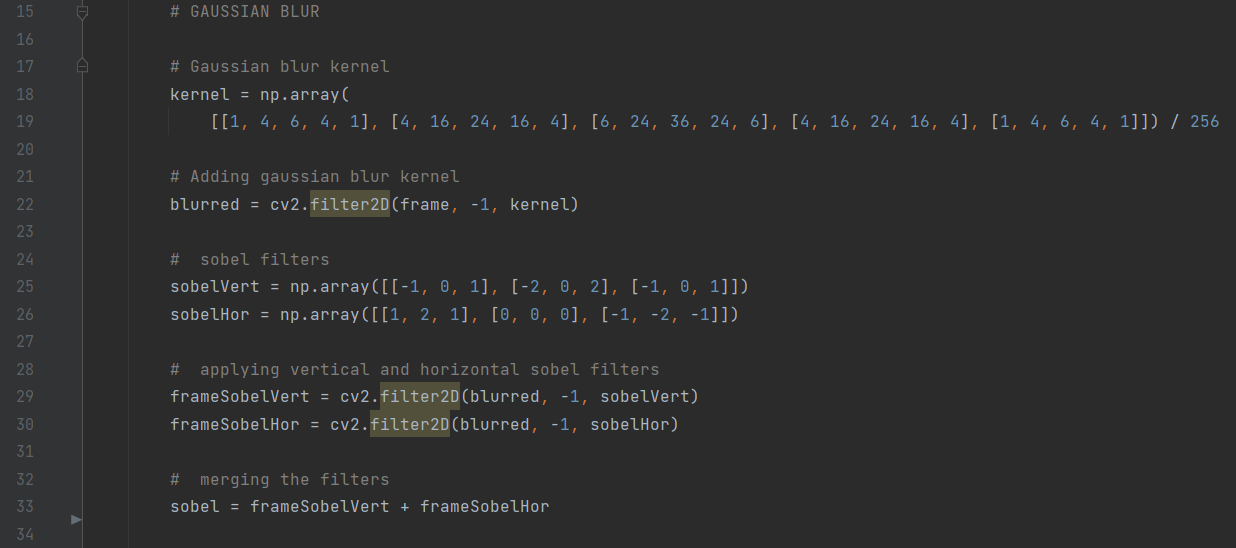
**Działanie:**

Skrypt wykorzystuje biblioteki OpenCV oraz NumPy. Wczytuje film i w pętli wykonuje operacje dla każdej klatki nagrania. Wszystkie operacje na pikselach opierają się na wartościach z zakresu RGB (0 – 255).



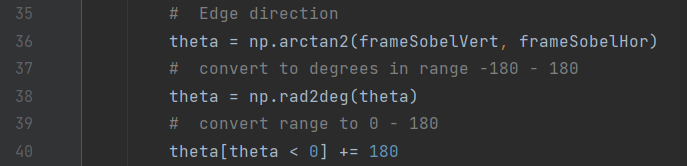
Rysunek 14. Przygotowanie nagrania i wczytywanie kolejnych klatek

Najpierw na klatkę nakładane jest rozmycie za pomocą filtru Gaussa. Następnie nakładane są filtry Sobela – pionowy i poziomy, które są łączone w jeden obraz.



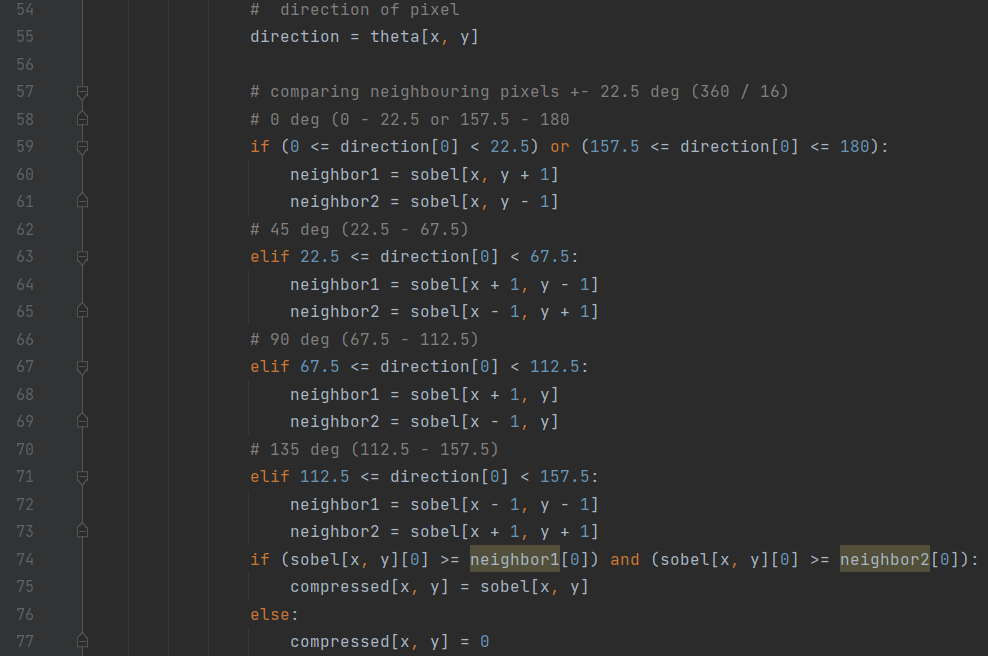
Rysunek 15. Filtr Gaussa oraz filtry Sobela

Następnie obliczane są kierunki krawędzi (obliczane wg. wzoru ), które przekształcane są z radianów na stopnie w zakresie 0 do 180.



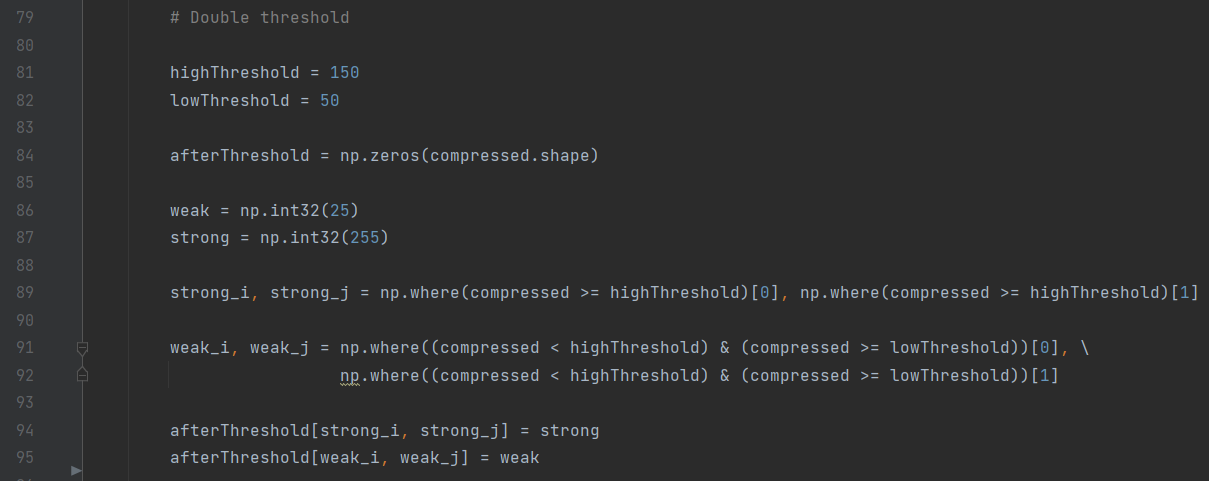
Rysunek 16. Obliczanie kierunków krawędzi

Do zastosowania non-maximum suppression tworzony jest nowy pusty obraz. W pętli dla każdego piksela sprawdzany jest jego kierunek i w zależności od niego porównywani z nim są odpowiedni „sąsiedzi”- piksele o zbliżonym kącie (kąty sprawdzane są w przedziałach +/- 22,5 stopnia od danej osi układu współrzędnych). Jeśli piksel jest intensywniejszy od jego sąsiadów jest on przenoszony na pusty obraz.



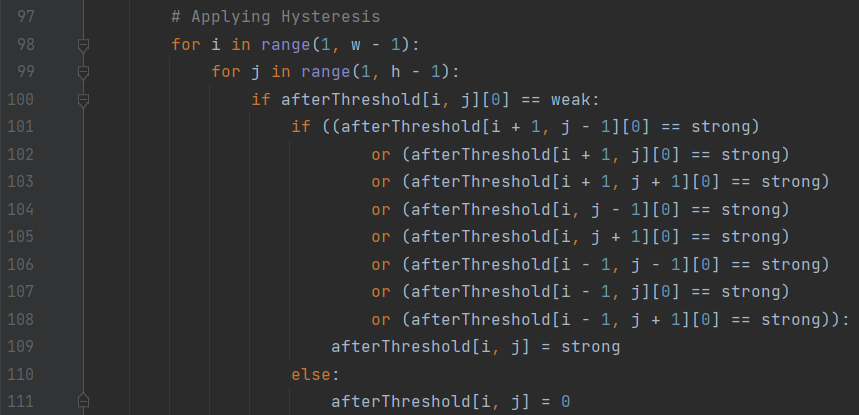
Rysunek 17. Wnętrze pętli porównującej sąsiednie piksele o tym samym zwrocie

Następnie ustawiane są wartości progów: górnego (150) i dolnego (10). Ponownie tworzony jest pusty obraz. Dodawane są wartości słabych pikseli (25) oraz mocnych (255). Wybierane są współrzędne pikseli silnych (równe lub powyżej progu górnego) oraz słabych (poniżej górnego oraz większych lub równych progowi dolnemu). Odpowiednie wartości silnych i słabych pikseli są przenoszone w obliczone wcześniej współrzędne pustej klatki.



Rysunek 18. Wykrywanie silnych i słabych pikseli

Następnie dla każdego słabego piksela dokonywana jest decyzja o należeniu do krawędzi lub nie; sprawdzane są wszystkie sąsiednie piksele i jeżeli choć jeden z nich jest silny to bieżący piksel również staje się silny. W przeciwnym wypadku piksel jest zaczerniany.



Rysunek 19. Sprawdzanie sąsiednich pikseli dla tych słabych

Na konsoli wypisywany jest numer kolejnej przetworzonej klatki. Gotowa klatka dodawana jest do nagrania wynikowego a otwarte nagrania są zwalniane.

**SKRYPT ŚLEDZĄCY OBIEKT (tracker.py)**

**Parametry wejściowe:** nagranie obiektu do śledzenia oraz jego wersja z samymi krawędziami

**Wynik:** wyświetlenie nagrania z ramką śledzącą obiekt

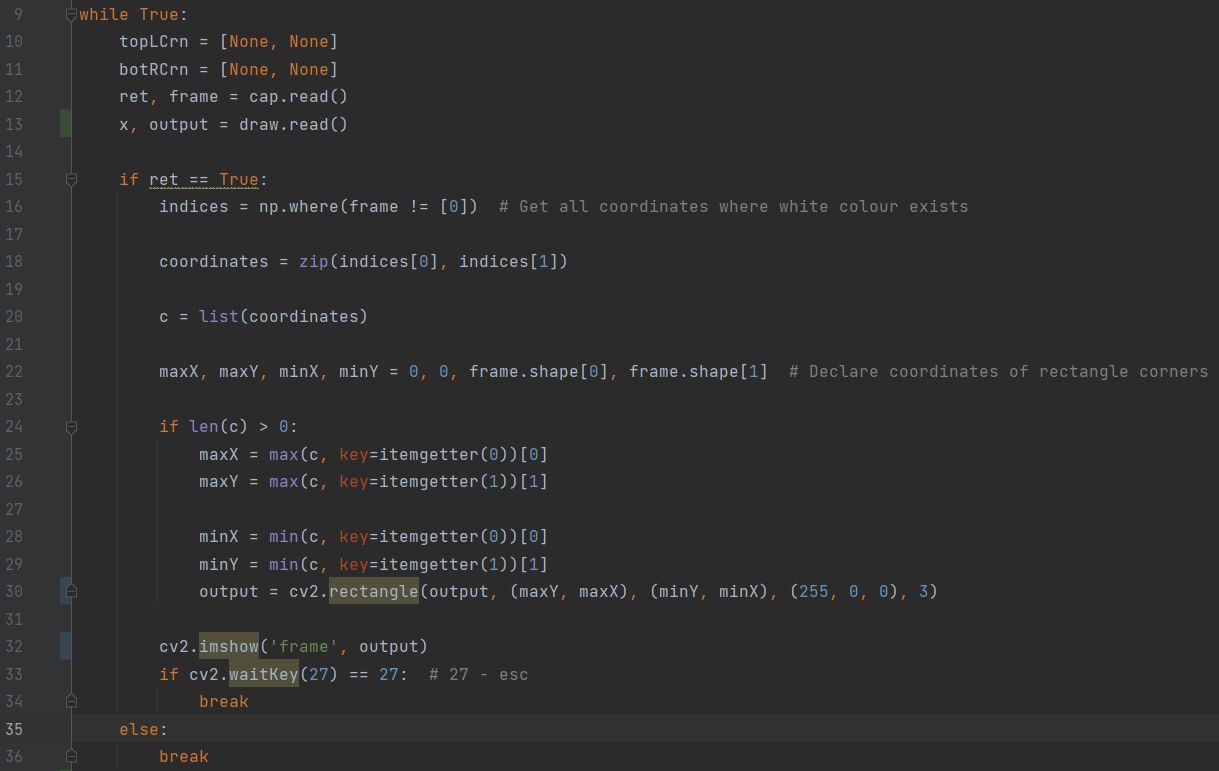
**Działanie:**

Skrypt wykorzystuje biblioteki OpenCV i NumPy oraz z biblioteki Operator funkcję itemgetter. Wczytuje oba nagrania i w pętli wykonuje operacje dla każdej klatki. Wszystkie operacje na pikselach opierają się na wartościach z zakresu RGB (0 – 255).

Program najpierw szuka wszystkich pikseli, które mają biały kolor (są to piksele krawędzi). Jeśli takie się pojawiły (sprawdzane jest to za pomocą długości listy wykrytych współrzędnych) oblicza współrzędne minimalne i maksymalne spośród wykrytych białych pikseli, na podstawie których narysuje ramkę wokół wykrytego obiektu.

W nowym oknie wyświetlane są kolejne ramki oryginalnego nagrania z prostokątem zawierającym w sobie wykryty obiekt. Odtwarzanie można zatrzymać za pomocą klawisza esc.

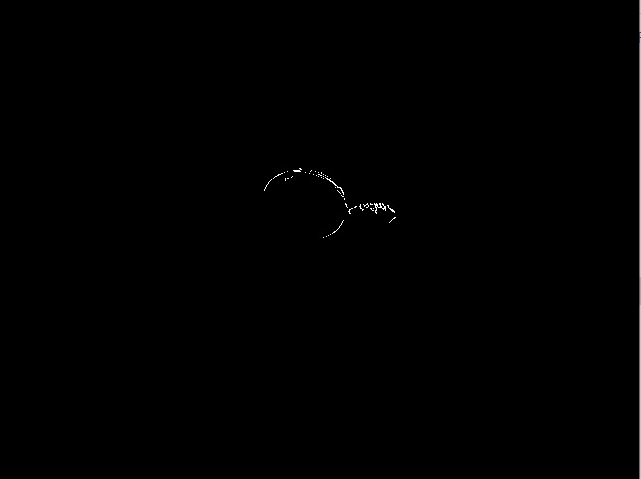
Otwarte nagrania są zwalniane.



Rysunek 20. Główna część skryptu tracker

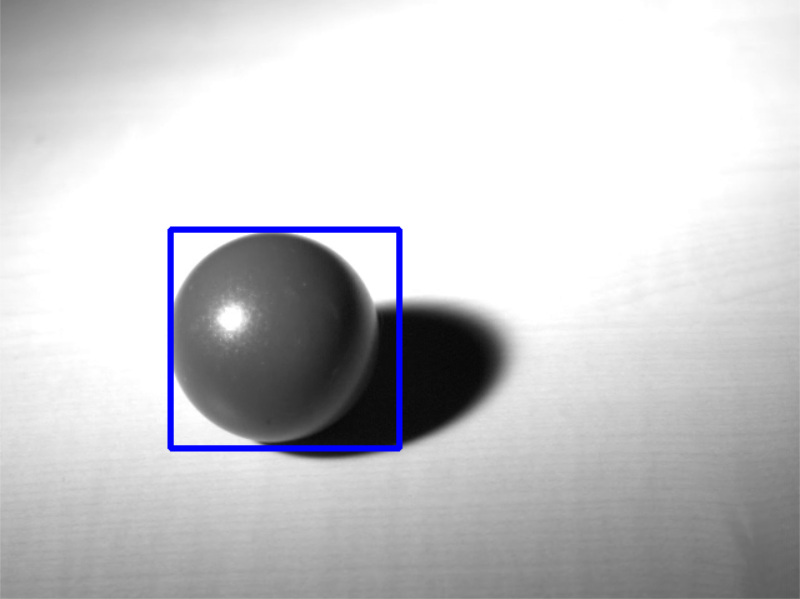
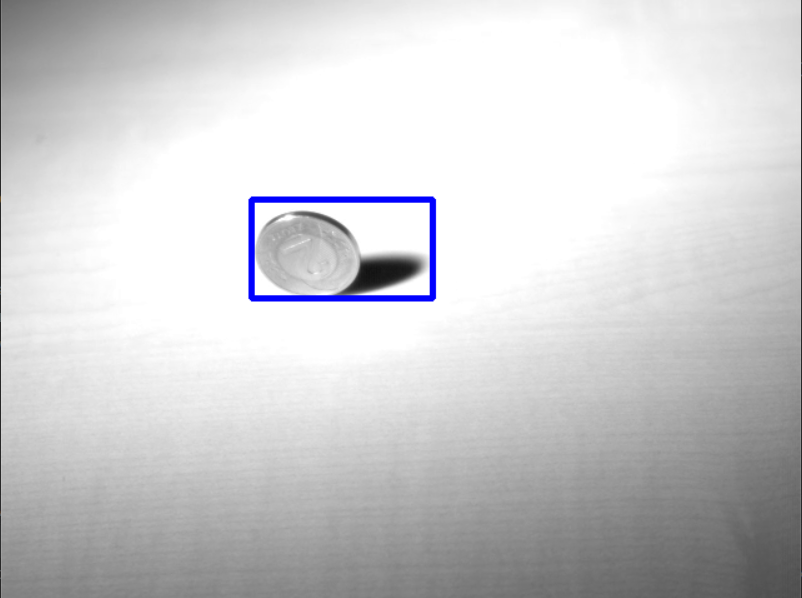
**Przedstawienie wyników**

Nagranie zwrócone przez skrypt wykrywające krawędzie ma postać czarnego obrazu z białymi obrysami śledzonego przedmiotu. Obrysy nie muszą idealnie odzwierciedlać przedmiotu – wybierane są najistotniejsze linie przybliżające objętość.

Rysunek 21. Nagrania- krawędzie

Nagrania z ramką śledzącą odtworzone w środowisku PyCharm:

Rysunek 22. Wynik końcowy działania programu

**Podsumowanie całości badania i wnioski**

Wykonano nagrania przy użyciu kamery szybkiej. Dokonano implementacji algorytmu Canny’ego do wykrywania krawędzi na obrazie. Informacje o krawędziach wykorzystano do narysowania ramki wokół śledzonego obiektu.

Wykorzystanie kamery szybkiej pozwoliło na dokładniejsze śledzenie nagranych obiektów; większa ilość klatek pozwala zauważyć więcej niż na standardowym nagraniu lub podczas obserwacji gołym okiem.

**Załączniki**

Projekt programu PyCharm z plikami źródłowymi:

1. Pliki ze skryptami (main.py oraz tracker.py)
2. Nagraniami obiektów (w folderze assets: pilka.mp4 oraz moneta.mp4)
3. Nagrania po zastosowaniu wykrywania krawędzi (result\_pilka.mp4 oraz result\_moneta.mp4)