Szymon CIEMAŁA, Krzysztof JASKOT
Politechnika Śląska

SYSTEM ROZPOZNAWANIA GESTÓW

Streszczenie. Praca skupia się na stworzeniu wygodnej w użyciu oraz powszechnie dostępnej biblioteki umożliwiającej wykorzystanie gestów, pozycji dłoni, obrotu dłoni oraz odległości miedzy wybranymi palcami jako elementów sterujących w dowolnym projekcie napisanym w języku Python. Do realizacji zadania wykorzystano biblioteki języka Python o otwartym kodzie źródłowym, głównie OpenCV, MediaPipe oraz SciKit-Learn.

GESTURE RECOGNITION SYSTEM

Summary. The work focuses on creating a convenient and widely available library enabling the use of gestures, hand position, hand rotation and the distance between selected fingers as control elements in any project written in Python. Open source Python libraries were used to complete the task, mainly OpenCV, MediaPipe and SciKit-Learn.

1. Wstęp

Rozwój technologii w ostatnich latach przyczynił się do coraz częstszego wykorzystywania wizji komputerowej i metod uczenia maszynowego do rozpoznawania oraz klasyfikacji rożnego typu obiektów, w tym części ludzkiego ciała. Pozwala to na interakcje człowieka z aplikacjami, często w sposób bardziej naturalny i intuicyjny [13, 1].

Celem pacy było stworzenie biblioteki w języku Python, która pozwoli na przystępne wykorzystanie algorytmów rozpoznawania gestów oraz ruchu dłoni. Biblioteka powinna oferować gotowe rozwiązania, na przykład przygotowane modele matematyczne pozwalające na rozpoznawanie alfabetu języka migowego oraz podstawowych gestów (otwarta, zamknięta dłoń). Dodatkowo powinna pozwolić na wyznaczenie pozycji dłoni, jej typu oraz innych własności takich jak, odległości miedzy końcówkami wybranych palców czy kąta obrotu dłoni. Założenia jakie zostały przyjęte podczas realizacji zadania to:

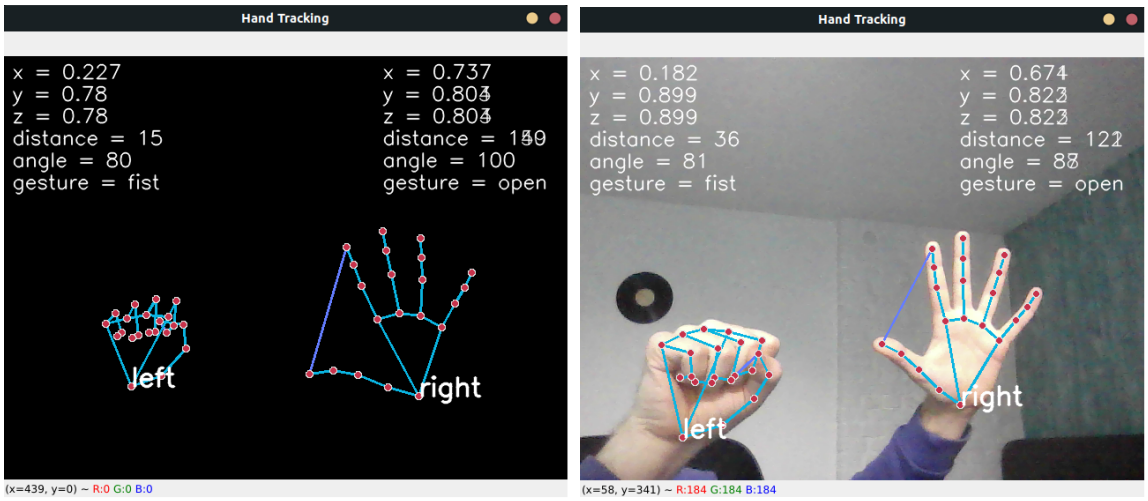
- Łatwość użytkowania – Poprzez łatwość użytkowania można rozumieć oprogramowanie zapisane zgodnie z powszechnie stosowanymi standardami („Zen of Python”) oraz przygotowaną dokumentacje wraz z przykładami wykorzystania.
- Dostępność – Biblioteka powinna być dostępna poprzez menedżer **pip**, dzięki czemu użytkownik ma możliwość instalacji biblioteki z repozytorium przy użyciu jednej komendy.
- Możliwość dodawania do biblioteki własnych elementów – Użytkownik powinien

mieć możliwość stworzenia własnego modelu rozpoznającego gesty, na przykład korzystając z interaktywnej instrukcji rozbudowy biblioteki.

2. Wymagania sprzętowe i programowe

Prezentowane rozwiązanie do prawidłowego działania wymaga zastosowania kamery, która pozwoli na pozyskanie obrazu. Rozdzielczość matrycy kamery powinna być wystarczająco duża, aby pozwolić na rozpoznanie dłoni. Nie ma tutaj minimalnych wymagań, większa rozdzielczość, czy też lepsza praca w warunkach niskiego oświetlenia kamery pozwoli na poprawniejsze działania algorytmów identyfikujących dłoń. Podobnie ma się liczba klatek na sekundę, która powinna być wystarczająco wysoka, tak aby można było sprawnie korzystać z możliwości oprogramowania. Uniwersalność prezentowanego rozwiązania pozwala na wykorzystanie dowolnej kamery podłączonej do komputera. Mogą to być kamery wbudowane w urządzenia przenośne takie jak laptopy lub komputery stacjonarne z podłączoną zewnętrzną kamerą USB. Niskie wymagania sprzętowe pozwalają również uruchomić oprogramowanie na systemach takich jak Raspberry PI 3B+ i wyżej.

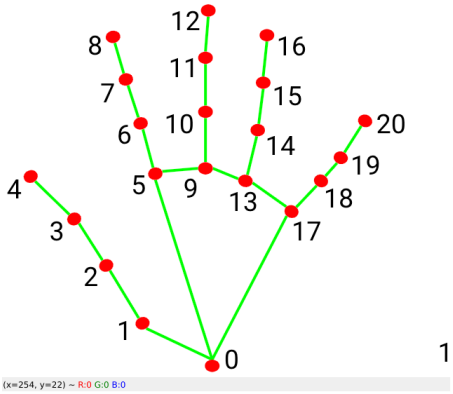
W celu realizacji zadania wykorzystano system operacyjny Linux (dystrybucja uBuntu wersja 20.04LTS). Jako język programowania posłużył Python. Dzięki swojej popularności oraz dostępności Python stał się językiem powszechnie stosowanym w pracy związanej z zagadnieniami uczenia maszynowego, analizy danych oraz przetwarzania obrazów. Na platformie PyPi można znaleźć wiele narzędzi pozwalających na pracę właśnie w tych dziedzinach, jak i również wielu innych. Dodatkowymi elementami wykorzystanymi w celu realizacji zadania są: iPython - interaktywna powłoka dla języka Python, która rozszerza jego działanie o introspekcję, czyli możliwość wykonywania poprzednich części programu, Jupyter Notebook oraz wspomniana platforma PyPi zawierająca różne moduły dla języka Python. Dodatkowo wykorzystano biblioteki OpenCV (wizja komputerowa), MediaPipe (uczenie maszynowe), SciKit-Learn (uczenie maszynowego, w tym algorytmy klasyfikacji, regresji oraz analizy skupień). Na rysunku 1 przedstawiono wyniki działania biblioteki OpenLeap.



Rys. 1. Zrzut ekranu programu testowego wykorzystujący bibliotekę OpenLeap

3. Algorytm rozpoznawania gestów

Algorytmy rozpoznawania gestów dłoni i języka migowego to złożone systemy, które wykorzystują różne techniki z zakresu przetwarzania obrazu, uczenia maszynowego i głębokiego uczenia. Stworzenie modułu wymagało napisania klasy wykorzystującej analizę obrazu i pozwalającej na rozpoznanie elementów charakterystycznych dłoni. Źródłem danych do analizy jest kamera internetowa podłączona przez złącze USB lub kamera wbudowana jak ma to miejsce w niektórych komputerach przenośnych. Obraz z kamery jest przetwarzany z wykorzystaniem funkcji dostępnych poprzez bibliotekę OpenCV [2]. Przetworzony obraz jest używany przez metody biblioteki MediaPipe, która pozwala na rozpoznanie elementów charakterystycznych dłoni [9, 12], co zostało przedstawione na rysunku 2. W napisanej klasie zostały zaimplementowane metody, które pozwalają na rozpoznanie typu dłoni (prawa, lewa), odległości miedzy paliczkiem palca wskazującego oraz kciuka, oraz obrotu dłoni. Kolejnym elementem jest wygenerowanie modelu matematycznego klasyfikującego gesty wykonywane przez dłonie. W tym celu została wykorzystana biblioteka SciKit-Learn wraz z dostępnymi poprzez nią algorytmami uczenia maszynowego [6]. Pierwszym krokiem w rozpoznawaniu gestów dłoni



Rys. 2. Elementy charakterystyczne dłoni [12]

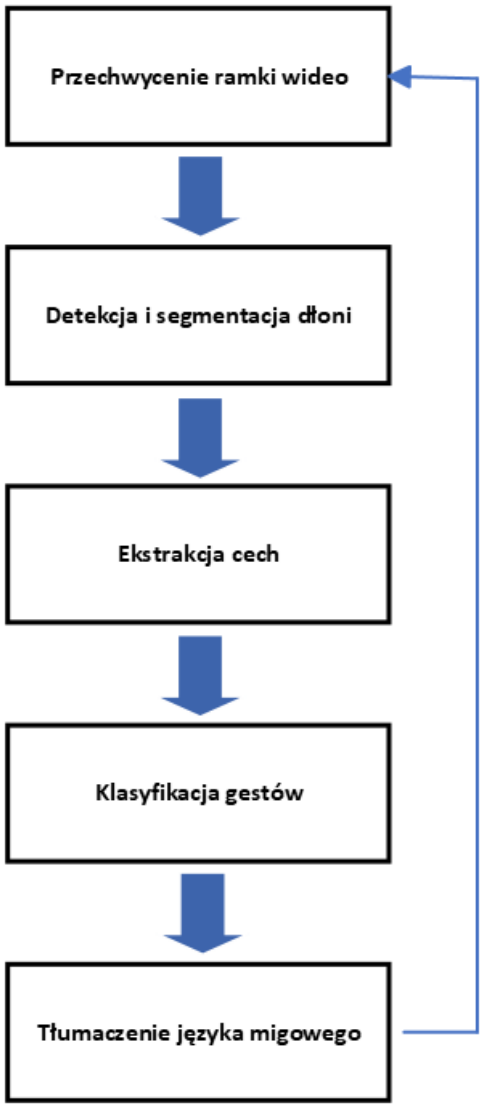
jest pozyskanie obrazów dłoni za pomocą kamer. Przetwarzanie obrazu obejmuje następujące etapy:

- Detekcja dłoni - Wykrywanie dłoni w obrazie przy użyciu metod takich jak wykrywanie krawędzi, segmentacja kolorów skóry, lub zaawansowane modele detekcji obiektów (np. YOLO, Faster R-CNN [10]).
- Segmentacja - Oddzielenie dłoni od tła, co pozwala na lepsze zrozumienie gestów.
- Śledzenie - Monitorowanie pozycji i ruchu dłoni w czasie rzeczywistym.

Z otrzymanego obrazu dłoni pozyskuje się cechy, które są istotne dla rozpoznawania gestów. W tym celu możemy się posłużyć metodą poszukiwania kluczowych punktów na dłoni (np. stawów palców) za pomocą algorytmów takich jak Mediapipe (rys. 2). Można również skorzystać z analizy konturów dłoni i ich kształtu oraz z deskryptorów takich jak HOG (Histogram of Oriented Gradients [5]) czy SIFT (Scale-Invariant Feature Transform [11]). Na podstawie pozyskanych cech gesty są klasyfikowane przy użyciu modeli uczenia maszynowego. Powszechnie stosowane techniki to:

- Klasyfikatory SVM (Support Vector Machine [4]) - Używane do klasyfikacji gestów na podstawie zebranych cech.
- Sieci neuronowe - Zwłaszcza konwolucyjne sieci neuronowe (CNN) oraz rekurencyjne sieci neuronowe (RNN), które mogą lepiej radzić sobie z sekwencyjnym charakterem gestów [8].
- Modele hybrydowe - Łączące różne podejścia, np. CNN do ekstrakcji cech i RNN do klasyfikacji sekwencji gestów.

Nowoczesne systemy rozpoznawania gestów coraz częściej wykorzystują głębokie sieci neuronowe, które automatycznie uczą się cech z danych wejściowych [3]. Wykorzystuje się również wcześniej wytrenowane modele na dużych zbiorach danych (np. ImageNet) i dostosowuje się je do specyficznego zadania rozpoznawania gestów. Pozwala to na skrócenie czasu treningu i poprawę dokładności modelu. Biorąc powyższe założenia algorytm rozpoznawania gestów można przedstawić jak na rysunku 3.



Rys. 3. Algorytm rozpoznawania gestów

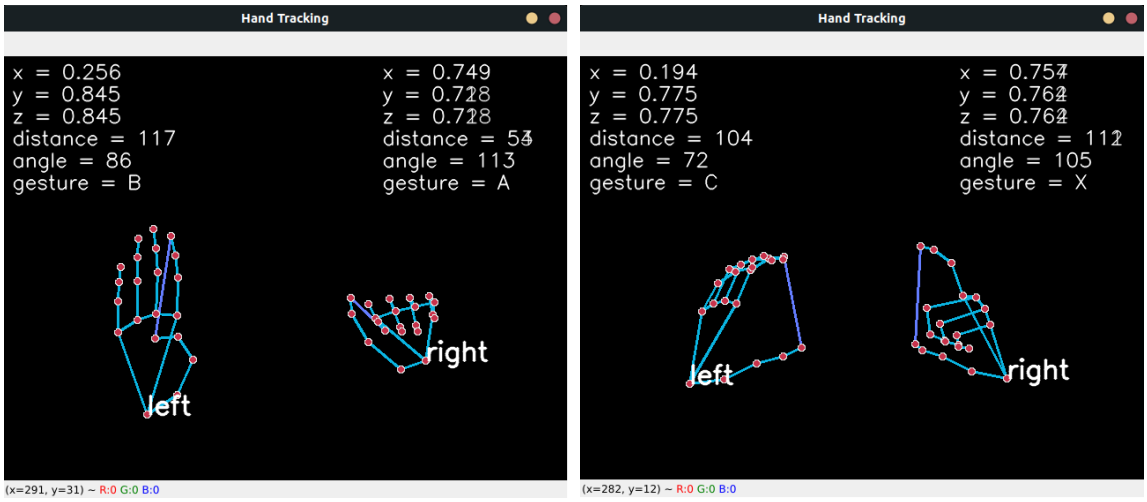
4. Wyniki

Na rysunku 4 przedstawiono stanowisko testowe, w którego skład wchodzi komputer przenośny z wbudowaną kamerą oraz uruchomione oprogramowanie do rozpoznawania gestów. Rozwiązanie takie umożliwia testowanie oprogramowania, które wykorzystując możliwości prezentowanych bibliotek pozwala na rozpoznawanie języka migowego. Dostęp do pakietu zawierającego wszystkie potrzebne elementy do działania znajdują się pod adresem $https://pypi.org/project/openleap/$ a ich instalacje możemy przeprowadzić przy użyciu komendy *pip install openleap*. Pod wskazanym adresem znajduje się również pełna dokumentacja. Natomiast na rysunku 5 pokazano zrzuty

Rys. 4. Wygląd stanowiska testowego do rozpoznawania gestów

ekranu z działającej aplikacji. Uruchomiona aplikacja miała za zadanie rozpoznawać litery języka migowego. Prezentowane rozwiązanie może posłużyć do dowolnych zadań związanych z rozpoznawaniem gestów w zależności od przygotowanej bazy uczącej (baza gestów). Prezentowane rozwiązanie można wykorzystać oprócz rozpoznawania

języka migowego do obsługi kiosków interaktywnych, sterowania komputerem, systemami audio w samochodach [7].



Rys. 5. Rozpoznawanie języka migowego litery A i B oraz C i X

5. Podsumowanie

Celem pracy było stworzenie biblioteki, która pozwoli na rozszerzenie aplikacji pisanych w języku Python o możliwość rozpoznawania gestów, sterowania dłońmi. Stworzona biblioteka charakteryzuje się otwartym kodem źródłowym, co oznacza, że można z niej bezpłatnie korzystać oraz każdy może dołączyć do jej rozbudowywania. Bibliotekę można ulepszyć poprzez napisanie jej w języku C++, który jest kompilowany. Pozwoliłoby to na zwiększenie wydajności oprogramowania. Język Python zezwala na korzystanie z modułów napisanych właśnie w języku C i C++, więc jest to język jak najbardziej kompatybilny. Zaprezentowane rozwiązanie poprawnie radzi sobie z rozpoznawaniem liter alfabetu migowego co można wykorzystać przy budowie małych systemów, które przekształcałby rozpoznane gesty w tekst lub mowę co umożliwiłoby komunikację z osobami niesłyszącymi. Dzięki uniwersalności prezentowanego rozwiązania można zbudować wydajny system korzystając na przykład z modułów Raspberry PI oraz dedykowanego akceleratora AI Hailo 8L.

LITERATURA

1. Badiola-Bengoa A, Mendez-Zorrilla A.: A Systematic Review of the Application of Camera-Based Human Pose Estimation in the Field of Sport and Physical Exercise. Sensors. 2021.
2. Bradski G., Kaehler A.: Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc., 2008.
3. Chung E. A., Benalcázar M. E.: Real-Time Hand Gesture Recognition Model Using

Deep Learning Techniques and EMG Signals 2019 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2019, 1-5.

4. Cortes C., Vapnik V.: Support-vector networks, Machine Learning, 1991, vol.20(3), p. 273–297.
5. Dalal N., Bill Triggs B.: Histograms of Oriented Gradients for Human Detection. International Conference on Computer Vision and Pattern Recognition (CVPR '05), 2005, p. 886–893.
6. Garreta R., Moncecchi G., Hauck T., Hackeling G.: Scikitlearn:machine learning simplified: implement scikit-learn into every step of the data science pipeline. Packt Publishing Ltd, 2017.
7. Guo L., Lu Z., Yao L.: Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review. IEEE Transactions on Human-Machine Systems, 2021, vol. 51(4), p. 300-309.
8. Hoffmann J., Navarro O., Kastner F., Janßen B., Hubner M.: A Survey on CNN and RNN Implementations, The Seventh International Conference on Performance, Safety and Robustness in Complex Systems and Applications, 2017, p. 33-39.
9. Indriani, Harris M., Agoes A.S.: Applying hand gesture recognition for user guide application using mediapipe. Proceedings of the 2nd International Seminar of Science and Applied Technology (ISSAT 2021), Atlantis Press, 2021, p. 101-108.
10. Jiang P., Ergu D., Liu F., Cai Y., Ma B.: A Review of Yolo Algorithm Developments, Procedia Computer Science, 2022, vol. 199, p. 1066-1073.
11. Lowe D. G.: Distinctive image features from scale-invariant key-points, International Journal of Computer Vision, 2004, p. 91-110.
12. MediaPipe. Hand landmarks. 2024
13. Nandwana B., Tazi S., Trivedi S., Kumar D., Vipparthi S. K.: A survey paper on hand gesture recognition, 7th International Conference on Communication Systems and Network Technologies (CSNT), Nagpur, India, 2017, p. 147-152.