**Analiza Obrazów**

**Dokumentacja projektu**

Temat projektu:

Aplikacja do kontroli przewozu drewna

Kateryna Andrusiak, Vadym Semkovych, Hleb Shypula

Informatyka Stosowana

Wydział Fizyki i Informatyki Stosowanej

Akademia Górniczo-Hutnicza w Krakowie

* **Opis projektu**

Celem projektu jest napisanie aplikacji do kontroli ilości drewna transportowanego z miejsca wyrębu lasu do zakładu obróbki. Aplikacja ma za zadanie policzyć ilość kłód załadowanych do ciężarówki oraz odnaleźć numer rejestracyjny samochodu. Zebrane dane są komponowane do wynikowego obrazu i zapisywane na dysk. Program jest napisany w MATLAB.

* **Algorytmy stosowane w projekcie**

**Algorytm do detekcji kłód załadowanych do ciężarówki**

Konwertujemy początkowy obraz do obrazu w odcieniach szarości.

Korzystamy z filtra medianowego oraz filtra Gaussa w celu usunięcia szumów (sól, pieprz i inne).

Przeprowadziliśmy korekcję kontrastu oraz gammy poprzez mnożenie oraz potęgowanie pikseli przez stałe.

Za pomocą funkcji *imfindcircles*, która korzysta z transformaty Hougha, znajdujemy kłody na obrazie. Mianowicie zapisujemy zwrócone wartości funkcji jako tablicę współrzędnych opisujących środki koł (kłod) oraz tablicę przechowującą promienie tych koł. Do funkcji przekazujemy zakres promieni koł do znalezienia jako [16, 50] pikseli.

Następnie korzystamy z napisanej przez nas funkcji DBSCAN, która korzysta z [algorytmu](https://en.wikipedia.org/wiki/DBSCAN) o tej samej nazwie. Jest to algorytm klasteryzacji oparty na gęstości - jeśli otrzyma zestaw punktów w jakiejś przestrzeni, algorytm grupuje punkty, które są blisko siebie (punkty z wieloma bliskimi sąsiadami), oznaczając punkty odstające, które są samotne na obszarach o małej gęstości (których najbliżsi sąsiedzi leżą daleko). Za pomocą tej funkcji filtrujemy odnalezione koła – usuwamy koła nienależące do cięrzarówki z drewnem. Robi się to przy pomocy parametrów przekazanych do funkcji – *minCircles* jako minimalna ilość bliskich sąsiedzi oraz *eps* jako odległość w zakresie której te sąsiedzi mają się znajdować.

Następnie rysujemy na obrazie odfiltrowane koła o odpowiednich współrzędnych oraz promieniach, które pokazują wykryte kłody.

**Algorytm do detekcji numeru rejestracyjnego**

Konwertujemy początkowy obraz do obrazu w odcieniach szarości.

Obliczamy próg za pomocą funkcji graythresh. Próg jest znormalizowany do zakresu [0, 1].

Konwertujemy obraz na obraz binarny za pomocą progu.

Wypełniamy dziury w obrazie binarnym za pomocą funkcji imfill.

Usuwamy obiekty zawierające mniej niż 100 pikseli za pomocą funkcji bwareaopen.

Obliczamy właściwości regionów na obrazie i otrzymujemy dane w tabeli używając regionprops. Zwrócone właściwości: pozycja i rozmiar najmniejszego prostokąta zawierającego region, orientacja, liczba Eulera, powierzchnia (ilość pikseli).

Analizując nasze obrazy, zakładamy, że numer znajduje się w dolnej części naszego obrazu, oraz że stosunek szerokości ramki numeru do wysokości jest w przedziale [3:1; 5:1], oraz że liczba Eulera równa się 1, orientację ustawiamy w przedziale [-5; 5]

Dla każdego regionu wyznaczamy pozycję oraz stosunek szerokości do wysokości, zatem ustalamy odpowiednie warunki dla detekcji potrzebnego regionu i zapisujemy takie regiony.

Dalej rysujemy wyznaczony przez algorytm kształt.

* **Analiza działania**

Po przeprowadzaniu testowania poprawności działania algorytmu wykryliśmy niedoskonałości związane z niedokładnością algorytmu.



Przykład 1

Na przykładzie 1 widzimy, że udało się nam dokładnie wyznaczyć numer rejestracyjny. Patrząc na kłody można zauważyć, że nie wszystkie zostały znalezione przez algorytm, mimo tego zdecydowana większość jest wyznaczona, zatem można wywnioskować, że dla tego obrazu algorytm działa poprawnie.



Przykład 2

Na przykładzie 2 nie ma numeru, dlatego algorytm go nie znalazł. Natomiast kłody udało się wyznaczyć prawie idealnie.



Przykład 3

Na przykładzie 3 udało się nam dokładnie wyznaczyć numer rejestracyjny. Kłody wyznaczane są dobrze, jednak widzimy wystąpienie obiektu wyznaczonego poza drewnem.



Przykład 4

Na przykładzie 4 udało się nam wyznaczyć miejsce gdzie ma znajdować się numer rejestracyjny. Algorytm dla detekcji kłód na tym przykładzie działa prawie idealnie, wyznaczone są wszystkie kłody gdzie kształt jest wyraźnie widoczny.

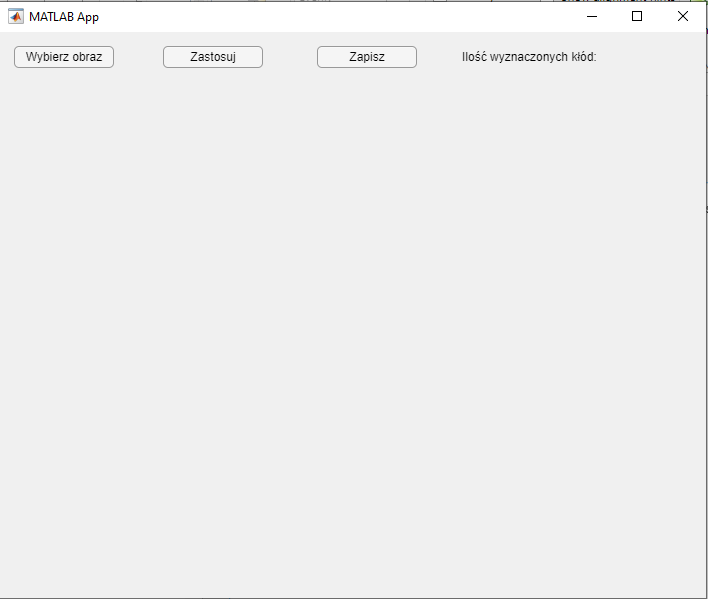


Przykład 5

Na przykładzie 5 udało się nam dokładnie wyznaczyć numer rejestracyjny. Natomiast są kłody, jakich algorytm nie znalazł, może to być spowodowane nieidealnym kształtem kłód.

Najlepiej nasza aplikacja działa w przypadku zdjęć zrobionych w jednakowych warunkach metearologicznych oraz technicznych (jakość obrazu, rozmiar obrazu, położenie ciężarówki  
z drewnem na obrazie oraz średnica kłod). W aktualnym stanie aplikacja wymaga osobnej konfiguracji dla każdego klienta.

* **Jak działa aplikacja?**

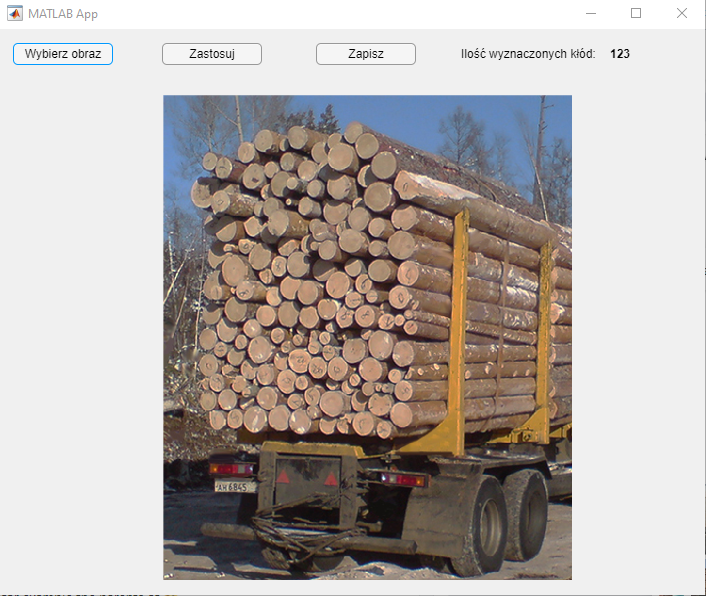


Rys. 1. Początkowe okno aplikacji.

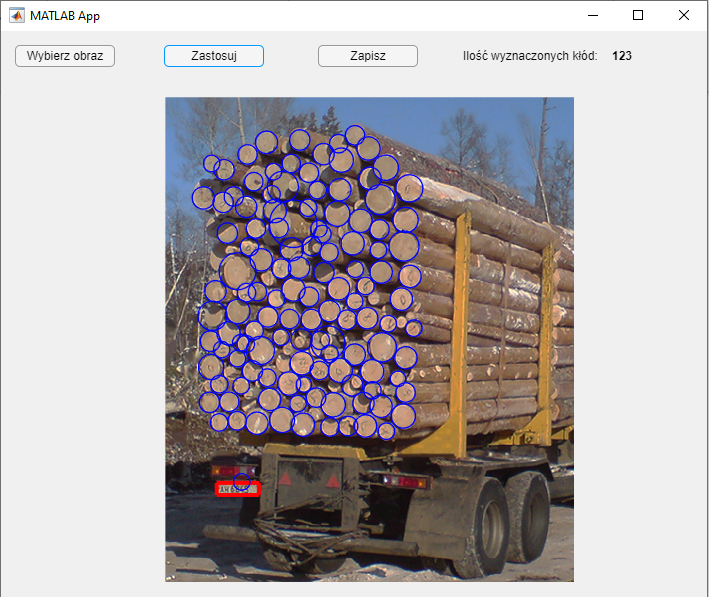


Rys. 2. Dostępna funkcjonalność aplikacji.

1. Przycisk pozwalający pobrać dowolny obraz \*.jpg, \*.png, \*.bmp.
2. Przycisk uruchamiający nasz algorytm.
3. Przycisk pozwalający zapisanie obrazu w dowolnym miejscu.
4. Miejsce, gdzie pojawia się ilość kłód wyznaczonych za pomocą algorytmu.



Rys. 3. Aplikacja po wybraniu obrazu.



Rys. 4. Aplikacja po zastosowaniu algorytmu dla wybranego obrazu.



Rys. 5. Zapisany obraz.