Analiza Obrazów

Dokumentacja projektu

Temat:

Wykrywanie tekstu

27 stycznia 2022

Marta Dychała, Michał Domin, Damian Koperstyński

Informatyka Stosowana

Wydział Fizyki i Informatyki Stosowanej

Akademia Górniczo-Hutnicza w Krakowie

1. Opis projektu

Celem projektu jest stworzenie aplikacji umożliwiającej rozpoznanie tekstu z wczytanego obrazka. Program umożliwia również skopiowanie rozpoznanego tekstu oraz zapis do pliku tekstowego. Aplikacja napisana została w Matlabie.

2. Założenia wstępne

a) umożliwienie użytkownikowi wczytania obrazu w formacie JPG, JPEG lub PNG dowolnego rozmiaru,
b) umożliwienie przycięcia wczytanego obrazu,
c) udostępnienie możliwości skopiowania i zapisania rozpoznanego tekstu,
d) utworzenie *datasetu* na potrzeby wytrenowania sieci neuronowej,
e) stworzenie prostego, intuicyjnego GUI dla użytkownika.

3. Podział pracy

- Marta Dychała – stworzenie graficznej części aplikacji GUI, integracja backend-frontend, testowanie aplikacji,
- Michał Domin – stworzenie i trening sieci neuronowej, backend, testowanie aplikacji,
- Damian Koperstyński – dokumentacja, testowanie aplikacji.

4. Działanie algorytmu wykrywania tekstu

Dane treningowe przygotowane zostały jako zestaw liter dla różnych, popularnych czcionek. Do nauki sieci neuronowej wykorzystano funkcję *train*, w której jako jeden z argumentów wykorzystany był wynik funkcji *feedforwardnet.*

Po wciśnięciu przycisku „WYKRYJ TEKST" wywoływany jest plik *im2text*. Wczytywane są parametry wytrenowanej sieci neuronowej z katalogu *data* oraz dodawana jest ścieżka do plików z katalogu *Functions*, w którym zawarte są m.in. funkcje służące do wyliczenia wartości współczynników. Następnie wczytany obraz jest binaryzowany, czyli następuje zmiana zakresu wartości pikseli na zbiór od 0 do 1, zamiana na obraz w odcieniach szarości i na końcu binaryzacja. W kolejnym kroku dokonywana jest separacja linii w obrazie, który zawiera litery, a następnie separowane są pojedyncze wyrazy. Wynik tego kroku przechowywany jest w zmiennej *lines*, w której każdy z elementów jest wyrazem bądź literą – jeśli jest to „w", „i", „z" lub „o".

Następnie program działając w pętli, której liczba iteracji równa jest ilości wykrytych wyrazów (zmienna *lines*) pobiera wyseparowany wyraz, z którego separowane są kolejno pojedyncze litery poprzez funkcję *regionprops*. Ilość iteracji kolejnej – zagnieżdżonej – pętli uzależniony jest od ilości liter w wyseparowanym wyrazie. W pętli tej obraz, przedstawiający każdą z liter *przycinany* jest do wielkości 160 x 112 za pomocą metody *imresize*. Następnie dla każdej litery wyliczana jest ilość pikseli w każdym z 70 pól – obraz 160 x 112 dzielony jest na 70 pól – każde pole o rozmiarze 256 pikseli 16 x 16, w wyniku czego otrzymujemy wektor kolumnowy o 70, znormalizowanym uprzednio do zakresu 0-1, wartościach. W kolejnym kroku każda z liter jest obiektem działania ośmiu funkcji, które dostarczają parametry charakteryzujące badaną literę. Parametrami tymi są:

- Kształt
- Współczynnik Blair-Bliss'a – średnia odległość od środka dla każdego piksela
- Współczynnik Malinowskiej
- CircularityS, CircularityL – kolistość

- Współczynnik Danielsson'a – średnia odległość piksela od krawędzi
- Współczynnik Haralick'a – średnia odległość pikseli obwodu od środka
- Współczynnik Feret'a – stosunek średnic z BoundingBox'a – bok poziomy do boku pionowego (*promienie Feret'a* to inaczej rozpiętość figury w *x* i *y*).

W wyniku tych operacji otrzymuje się zmienną, która zawiera wartości opisujące każde z 70 pól na jakie podzielony został obraz z wyseparowaną literą, 8 wartości parametrów opisanych powyżej i 2 wartości opisujące wielkość litery jako stosunek jej wysokości do szerokości oraz jego odwrotność. Opisana zmienna zawierająca 80 parametrów podawana jest na wejście sztucznej sieci neuronowej dwu warstwowej:

- Pierwsza warstwa – 100 neuronów

Na podstawie 80 parametrów sieć neuronowa dokonuje klasyfikacji i w wyniku podawana jest litera, która została dopasowana. Algorytm powtarzany jest dla każdej kolejnej litery w wyrazie. Po dopasowaniu wszystkich liter w danym wyrazie – następuje przejście do kolejnego wyseparowanego uprzednio wyrazu.

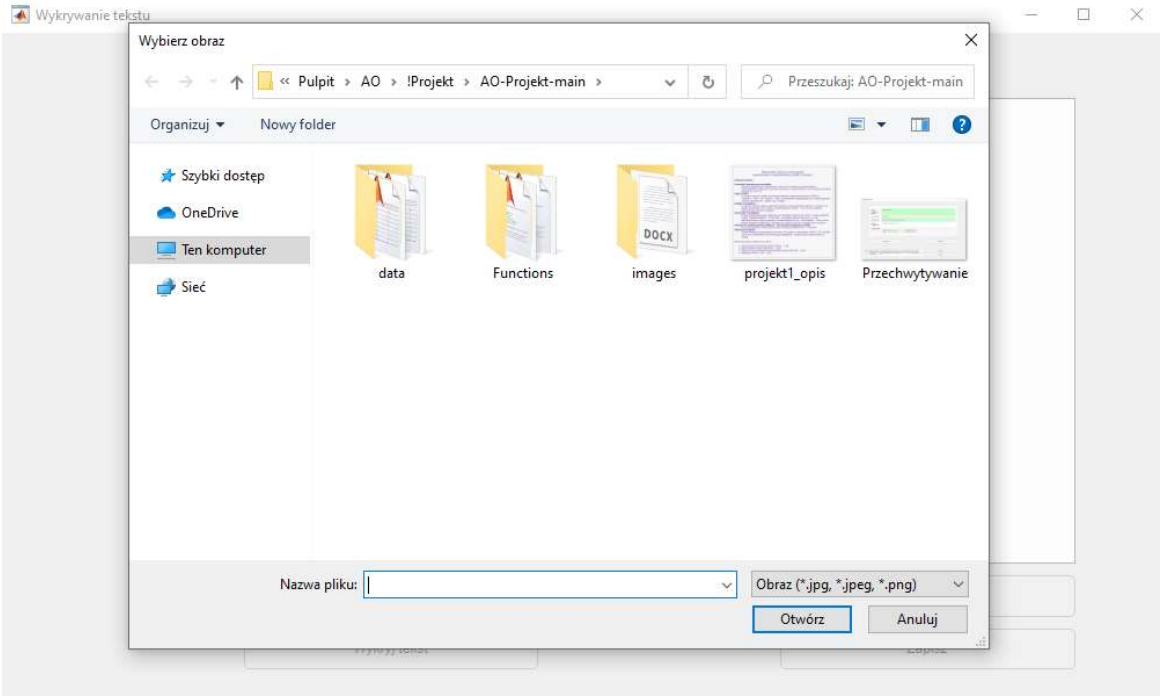
Zalecane jest, aby na wczytywanym obrazie występował duży kontrast pomiędzy tłem a literami – najlepiej jeśli są to czarne litery na białym tle, ponieważ w ten sposób zwiększa się prawdopodobieństwo prawidłowego dopasowania liter.

5. Interfejs użytkownika

Projekt zrealizowany został jako program z GUI, stworzonym przy użyciu aplikacji Matlab App Designer. Aby uruchomić aplikację najłatwiej jest otworzyć plik ***gui_app***. Można również otworzyć projekt w środowisku Matlab, a następnie z głównego katalogu projektu uruchomić plik ***main.m*** poprzez naciśnięcie przycisku 'Run'.

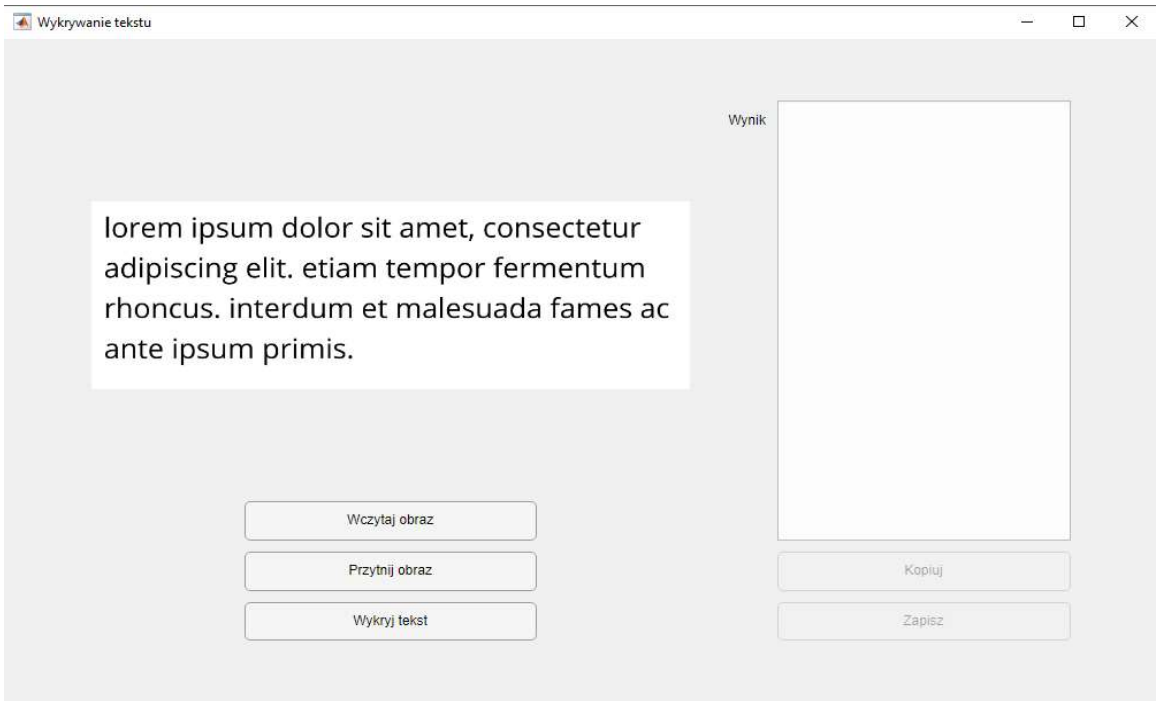
Rys. 1. Ekran główny aplikacji.

Rysunek powyżej (Rys.1.) przedstawia okno główne aplikacji. Początkowo aktywny jest jedynie przycisk umożliwiający użytkownikowi wczytanie obrazu z tekstem. Po wciśnięciu go pojawia się okno *eksploratora plików*, dzięki któremu w wygodny sposób można wybrać interesujący plik (Rys.2.).



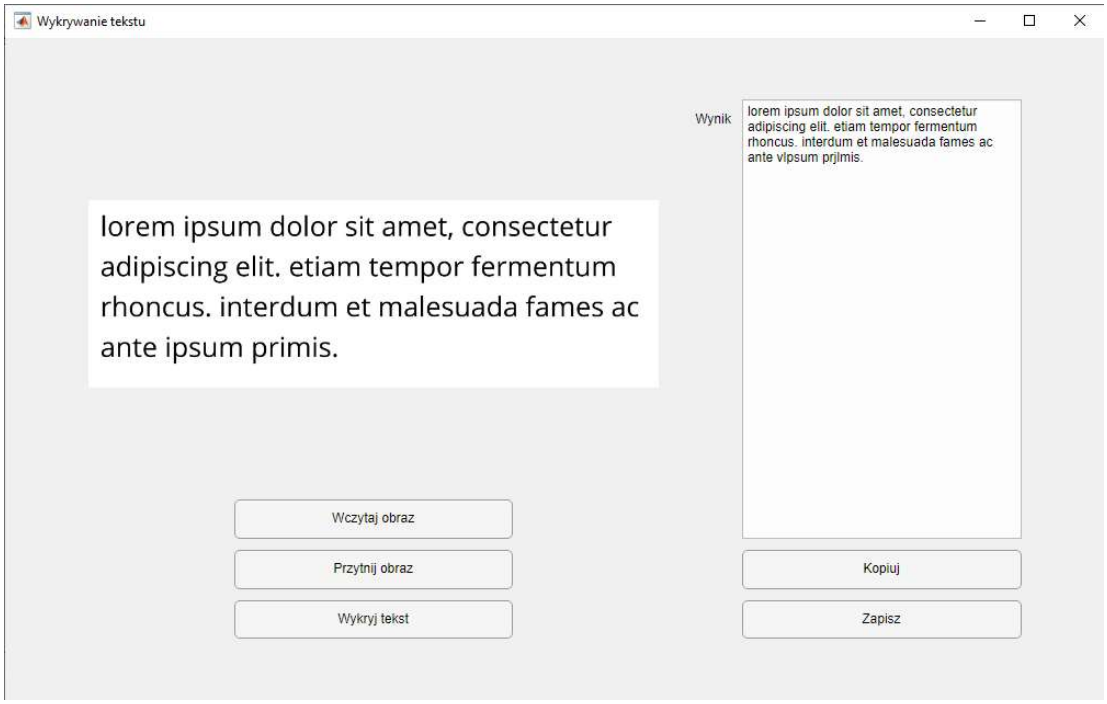
Rys. 2. Ekran wyboru obrazu do wczytania.

Po wybraniu oraz wczytaniu obrazka w lewej części okna głównego pojawia się jego podgląd oraz odblokowane zostają pozostałe przyciski w tej sekcji aplikacji – do operacji na wczytanym obrazie, co zaprezentowano na poniższym Rys. 3.

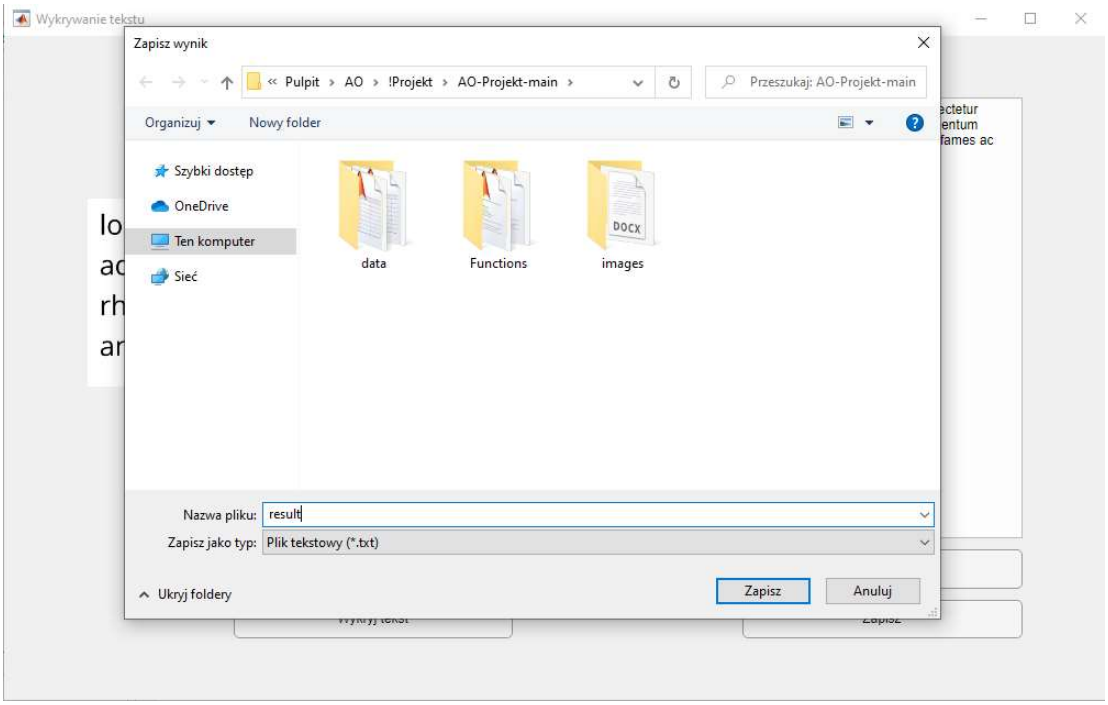


Rys. 3. Ekran aplikacji po wczytaniu obrazka.

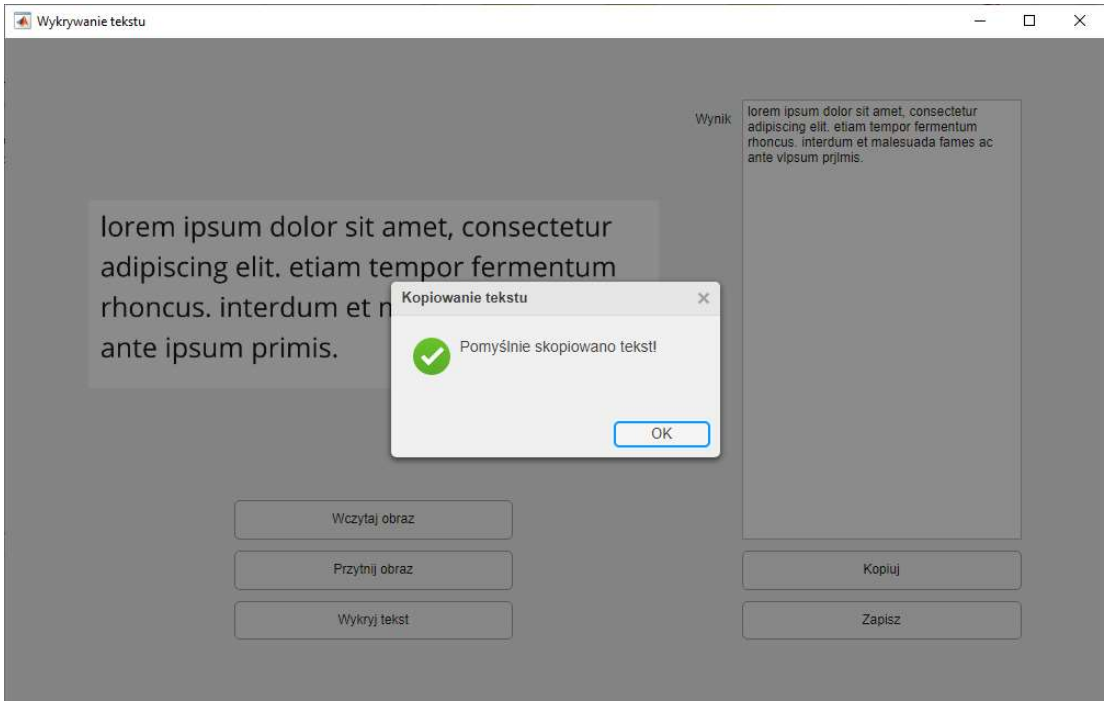
Przyciśnięcie przycisku *'Wykryj tekst'* uruchamia całą logikę opisaną w pkt. 4. niniejszej dokumentacji. Po zakończeniu działania algorytmu wykryty tekst pojawia się w prawej części okna aplikacji w sekcji *wynik*. Odblokowane zostają również przyciski umożliwiające skopiowanie tekstu do schowka systemowego oraz zapisanie go do pliku tekstowego (Rys. 4. i Rys. 4a).



Rys. 4. Ekran aplikacji po wykryciu tekstu z wcześniej wczytanego obrazka.

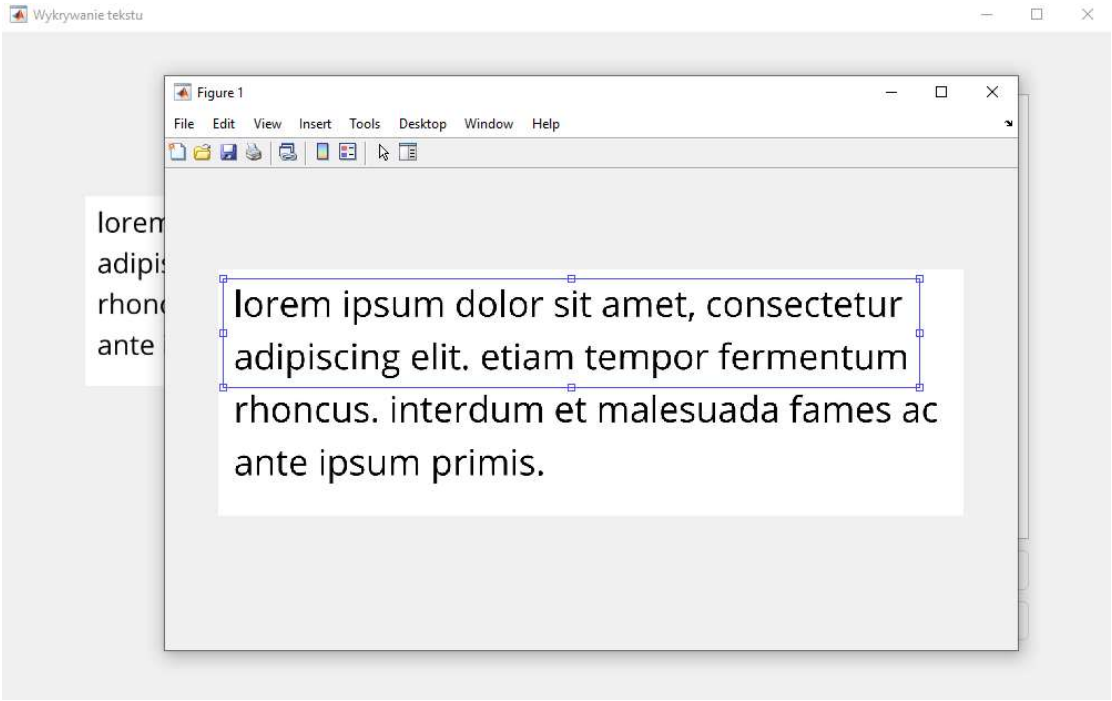


Rys. 4a. Zapis wykrytego tekstu do pliku tekstowego.



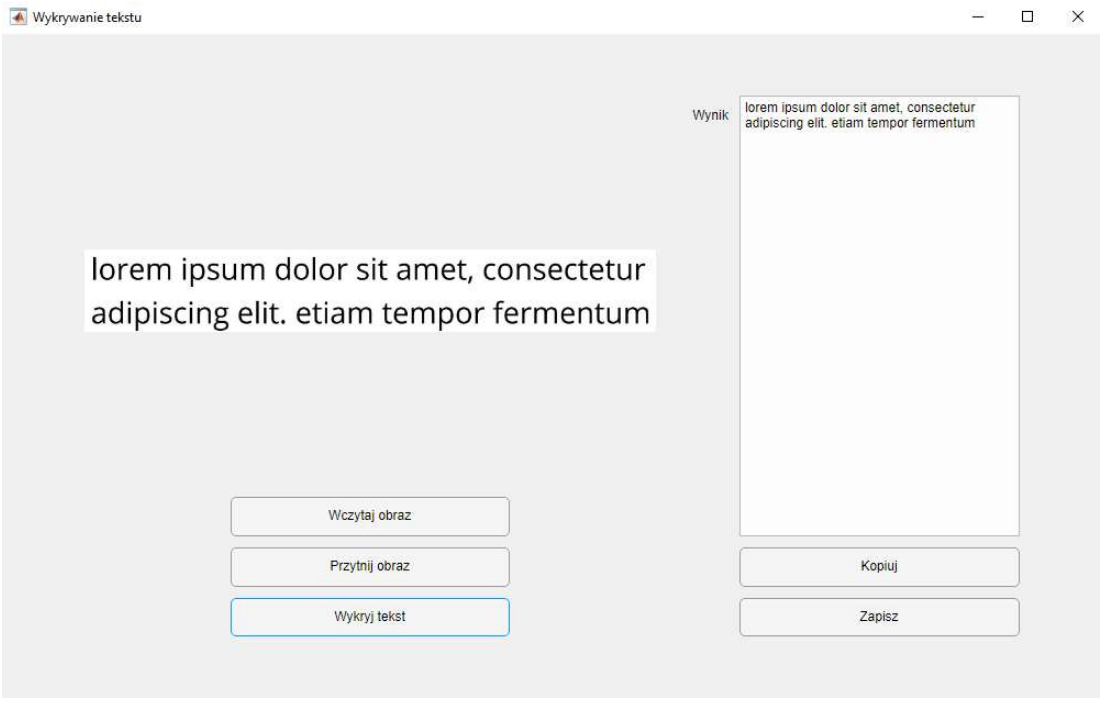
Rys. 4b. Skopiowanie wykrytego tekstu do schowka.

Ponadto, poza podstawową funkcjonalnością opisaną powyżej, zaimplementowano możliwość przycięcia obrazka. Aby skorzystać z tej opcji trzeba najpierw – zgodnie w powyższym opisem – wczytać obraz z pliku. Po tej operacji należy wybrać przycisk *'Przytnij obraz'*, w efekcie czego uruchomione zostanie okno umożliwiające wybranie interesującego fragmentu (Rys. 5.). W kolejnym kroku należy zaznaczyć fragment obrazka poprzez wciśnięcie LPM (lewego przycisku myszy) na początku obszaru, a następnie przesuwając mysz – z wciśniętym LPM – zaznaczyć interesujący nas zakres. Obszar zawierający się w niebieskiej ramce można edytować poprzez przesunięcia kwadracików znajdujących się na bokach ramki (zaznaczenia).



Rys. 5.Okno do wyboru fragmentu wczytanego obrazka.

Aby zatwierdzić wybranie danego fragmentu należy dwukrotnie kliknąć LPM w obszarze wewnątrz zaznaczenia. Powoduje to powrót do ekranu głównego aplikacji, ale w oknie podglądu widoczna jest teraz zaznaczona część obrazka (Rys. 6.). Oczywiście – podobnie jak poprzednio – po wykryciu tekstu jest możliwość skopiowania go do schowka oraz zapis do pliku.



Rys. 6. Wykryty tekst dla wybranego fragmentu obrazka.

6. Co nie działa?

Program rozpoznaje tylko litery o czcionkach, które zostały użyte do uczenia sieci neuronowej. W zbiorze uczącym nie zostały zawarte polskie znaki, a zatem aplikacja ich nie wykryje. Program ma problem z wykryciem tekstu umieszczonego na ciemnym tle. Dodatkowym ograniczeniem jest to, że litery na wczytanym obrazie powinny być czarne wówczas – w połączeniu w białym tłem – otrzymujemy najlepszy efekt. Ponadto jeśli chcemy wykorzystać aplikację do wykrycia tekstu, np. na zdjęciach zrobionych kartkom w książce, powinniśmy podzielić je na mniejsze fragmenty (np. 3), aby uzyskać zadowalający rezultat.

7. Co można usprawnić?

Dodanie do zbioru danych uczących liter innych czcionek. Uniezależnić poprawność wykrycia tekstu od koloru tła oraz czcionki.