

Analiza i Przetwarzanie Obrazów

DOKUMENTACJA PROJEKTU

05.07.2023

Dominik Dziuba

Łukasz Wajda

Mariusz Biegański

Mateusz Niepokój

Przemysław Rodzik

Szymon Pawelec

1. Cele i założenia projektu

Celem projektu było stworzenie programu pozwalającego użytkownikowi na wyszukanie najkrótszej drogi pomiędzy zaznaczonymi na mapie punktami. Powinien on zapewniać odpowiedni interfejs, który umożliwia wczytanie wybranej przez siebie mapy, ustawienie punktu początkowego i końcowego, a algorytm wyszukujący trasę powinien brać pod uwagę szerokość dostępnych dróg. Efektem oczekiwanym działania programu ma być wizualizacja na przekazanym obrazku optymalnej z punktu widzenia algorytmu trasy.

2. Opis zrealizowanego rozwiązania

Projekt został zrealizowany jako aplikacja desktopowa (.exe) z graficznym interfejsem użytkownika. Umożliwia ona wczytanie wybranego przez użytkownika obrazu, jak również konfigurację parametrów odpowiadających za odpowiednie przekształcenie obrazu i wyszukanie optymalnej ścieżki. Ustawienie punktów początkowego i końcowego szukanej trasy możliwe jest poprzez zaznaczenie wybranych miejsc na wczytanej mapie.

3. Generowanie danych

Na potrzeby stworzenia algorytmu wyszukującego najkrótszą drogę, wygenerowano zbiór obrazów przy pomocy programu Easy Diffusion. Jest to narzędzie oparte na sztucznej inteligencji, podobne do Stable Diffusion, które ma za zadanie przekształcać tekst na obrazy.

W pierwszym etapie procesu wybrano fragment mapy bez etykiet ze strony <https://snazzymaps.com/style/24088/map-without-labels>. Ten konkretny obraz został wykorzystany jako podstawa do generowania nowych obrazów przy użyciu narzędzia Easy Diffusion. Z wygenerowanych obrazów wybrano jeden, który został wykorzystany do stworzenia końcowego zbioru obrazów. Zbiór ten użyty został do testowania algorytmu wyznaczania trasy.

Ostateczny prompt, który służył do tego celu, prezentuje się następująco:

„map with white roads, only roads are white, city, road with difference width, very wide road, clear roads, all roads connected, black and white”.

Zdefiniowany został również negative prompt:

„label, text, characters, many colors, colored roads, color blending”.

Dodatkowe parametry ustawione w generatorze to:

- Seed: 1
- Number of Images: 32
- Model: sd-v1-4
- Image size: 1280x768
- Inference Steps: 25
- Guidance Scale: 28
- Prompt Strength: 0,82
- Image Quality: 95

4. Szczegółowy opis algorytmu

Algorytm wyszukiwania ścieżki można podzielić na kilka etapów. W pierwszym z nich, sprawdzany jest poprawny wybór punktów i dostępność wczytanego przez użytkownika obrazu. Ustawiane są również parametry takie jak kolor znalezionej ścieżki, czy jej szerokość. Następnie obraz jest odpowiednio przygotowywany do wykonania procesu wyszukiwania drogi.

Najpierw obraz mapy poddawany jest przekształceniu do skali szarości, dzięki czemu do każdego piksela przypisana jest pojedyncza wartość liczbowa. Następnie wartości te są pobierane z każdego zaznaczonego przez użytkownika punktu. Dzięki określonej w parametrze tolerancji, obliczana jest minimalna i maksymalna wartości, czyli zakres reprezentujący kolor drogi. Dla każdego tak zdefiniowanego zakresu przeprowadzane jest progowanie obrazu, które scalane jest w jeden obraz wynikowy za pomocą maski bitowej *bitwise_or*.

W kolejnym kroku, przy pomocy filtra uśredniającego wyznaczane są lokalne szerokości dróg. Szum usuwany jest dzięki nałożeniu maski *bitwise_and*, co równocześnie pozwala na zachowanie oryginalnych wartości pikseli tam, gdzie w operacji progowania nie znajdowała się droga. Ponadto, ustawiana jest minimalna wartość pikseli na obrazie.

Jako ostatnie przekształcenie, obraz zapisywany jest w postaci tablicy, a wszystkie wartości pikseli podnoszone są do 3. potęgi, co zapobiega przeszkakiwaniu znalezionej ścieżki przez obszary niebędące drogą.

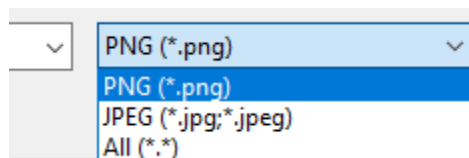
Dla tak przygotowanych danych, przeprowadzany jest zmodyfikowany algorytm Dijkstry. Polega on na wyszukaniu optymalnej ścieżki pomiędzy 2 punktami w taki sposób, jakby szerokie drogi miały mniejszy koszt przejścia, gdzie każdy piksel jest traktowany jak wierzchołek grafu.

Otrzymana w ten sposób trasa jest następnie rysowana i nakładana na obraz wejściowy.

5. Interfejs użytkownika

Opis przycisków:

- *Load Photo* – służy do wczytania obrazu. Domyślnie ustawione jest rozszerzenie .png, jednak możliwe jest wybranie innego - .jpg, .jpeg lub dowolnego *.*.




Scr. 2: Wczytywanie obrazu

- *Choose Path Color* – otwiera panel wyboru koloru, w jakim będzie rysowana ścieżka
- *Find Path* – uruchamia algorytm wyszukiwania ścieżki pomiędzy zaznaczonymi punktami.

Opis parametrów:

- *Max Image Width* – określa maksymalną szerokość obrazka.
- *Max Image Height* – określa maksymalną wysokość obrazka. Podobnie jak *Max Image Width*, jeśli wczytany obraz jest mniejszy, nie zostanie przeskalowany do podanego rozmiaru.
- *Minimum Pixel Weight* – określa minimalną wartość pikseli. Zbyt niska wartość może spowodować tworzenie się 'pętli' w wyszukanej ścieżce.
- *Step Value* – określa odległość pomiędzy kolejnymi pikselami, dla których rysowany będzie odcinek ścieżki.
- *Custom Error* – parametr tolerancji wykorzystany do tworzenia możliwego zakresu kolorów dróg. Domyślna wartość 0,05 oznacza, że wartość minimalna, czyli dolna granica wykorzystana do operacji progowania jest o 5% mniejsza od wartości piksela, a wartość maksymalna 5% większa. Dzięki temu parametrowi możliwe jest uwzględnienie pikseli o minimalnie mniejszej lub większej wartości w algorytmie wyszukiwania ścieżki.
- *Filter Size* – określa rozmiar filtra uśredniającego.
- *Path Width* – rozmiar rysowanej na obrazie ścieżki.

 AiPO — □ ×

Max Image Width:

Max Image Height:

Load Photo

Minimum Pixel Weight:

Step Value:

Choose Path Color

Find Path

Custom Error

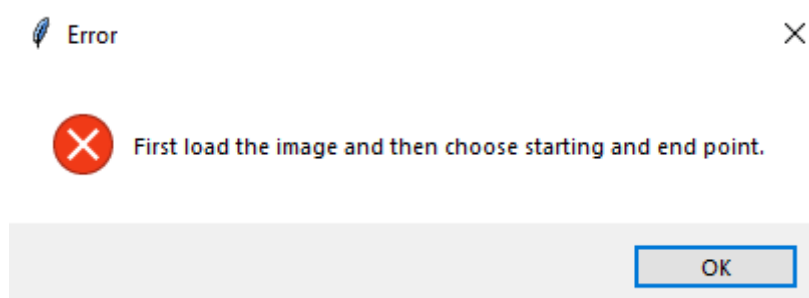
Filter Size

Path Width:

Scr. 3: Interfejs użytkownika

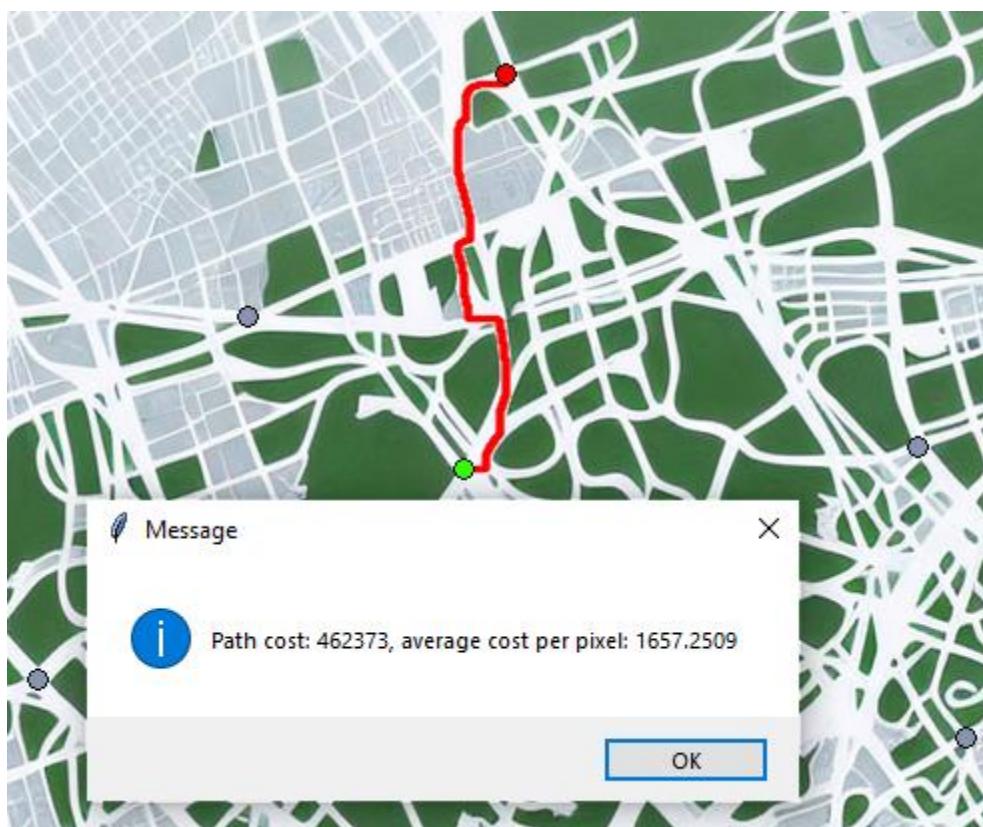
6. Instrukcja obsługi aplikacji

Aplikacja uruchamiana jest jako plik wykonywalny .exe. Po uruchomieniu, wyświetla się główne okno aplikacji – interfejs użytkownika. Możliwa jest modyfikacja parametrów opisanych w poprzednim punkcie dokumentacji. Aby przeprowadzić wyszukiwanie ścieżki, należy najpierw wczytać obraz. Naciśnięcie przycisku 'Find Path' bez wczytanego obrazu wyświetli na ekran użytkownika odpowiedni komunikat.



Scr. 4: Komunikat błędu

Po załadowaniu obrazu nie jest możliwe jego skalowanie. Kolejnym krokiem jest wybór punktów na obrazie. Użytkownik ma możliwość zaznaczenia punktów przez naciśnięcie w odpowiednim miejscu na mapie lewego przycisku myszy. Pierwszy wybór oznaczony kolorem zielonym to miejsce startu, drugi wybór oznaczony kolorem czerwonym to punkt końcowy. Możliwe jest wybranie kolejnych punktów oznaczonych kolorem szarym, dzięki temu w przypadku dróg oznaczonych innym kolorem niż punkty startu i końca, zakres możliwych z punktu widzenia algorytmu kolorów dróg zostanie rozszerzony. Wyczyszczenie zaznaczonych punktów możliwe jest poprzez naciśnięcie na mapie prawego przycisku myszy. Uruchomienie algorytmu bez zaznaczenia przynajmniej 2 punktów – czyli początku i końca – również spowoduje wyświetlenie komunikatu błędu. Po poprawnym uruchomieniu algorytmu, wyświetlony zostanie komunikat informujący użytkownika o łącznym koszcie ścieżki i koszcie średnim w przeliczeniu na pojedynczy piksel.



Scr. 5: Informacje o znalezionej ścieżce

W celu wyszukania kolejnej ścieżki należy wyczyścić zaznaczone punkty przy pomocy PPM i wybrać nowe.

Aby zmienić obraz, należy użyć przycisku 'Load Photo'. Obecnie otwarta mapa zostanie zastąpiona nową.

7. Wykorzystane biblioteki

W projekcie wykorzystane zostały następujące biblioteki:

- tkinter – odpowiada za interfejs graficzny użytkownika
- numpy – wykorzystanie tablic do reprezentacji obrazu i filtra uśredniającego
- cv2 – umożliwia przeprowadzenie transformacji obrazu takich jak filtry, zmiana przestrzeni kolorów, jak również rysowanie wyznaczonej ścieżki
- heapq – przydatna w reprezentacji i sortowaniu pikseli jako wierzchołków grafu w algorytmie Dijkstry
- numba – zwiększenie wydajności algorytmu Dijkstry
- PIL – obsługa plików graficznych

8. Podział ról w grupie

Dominik Dziuba – stworzenie koncepcji rozwiązania, przekształcenia na obrazie, generalna implementacja algorytmu wyszukiwania ścieżki

Łukasz Wajda – implementacja GUI

Mateusz Niepokój – generowanie obrazów testowych

Mariusz Biegański – utworzenie plików wykonywalnych programu na środowiska windows i linux

Przemysław Rodzik – GUI, implementacja algorytmu Dijkstry

Szymon Pawelec – dokumentacja