Projekt
Analiza i Przetwarzanie Obrazów

Opis zrealizowanego rozwiązania

Rozwiązywany problem

Aplikacja pozwala na wyznaczenie najszybszej drogi między podanymi punktami na wczytanym obrazku, przy założeniu że drogi są jaśniejsze od tła oraz prędkość poruszania się jest proporcjonalna do szerokości drogi.

Etapy działania

1. **Wczytanie obrazu**
2. **Wyznaczenie punktów startu i stopu**
3. **Preprocessing obrazu**
4. **Znalezienie najkrótszej ścieżki algorytmem mrówkowym**
5. **Wyświetlenie rozwiązania**

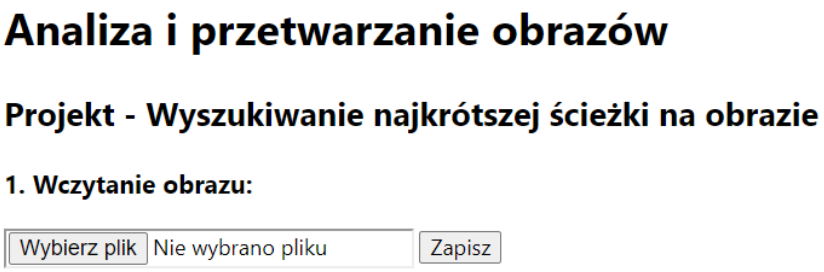
Instrukcja obsługi aplikacji

Opis uruchamiania

Projekt dostępny pod linkiem:
https://github.com/orgs/analiza-obrazow-projekt/repositories.
Składa się z aplikacji app uruchamianej przy pomocy polecenia:
python3 -m http.server następnie otwieranego w przeglądarce pod adresem http://localhost:8000/ oraz generatora obrazków w repozytorium generate_maps, które opisuje proces konfiguracji stable diffusion web ui wykorzystanego do stworzenia przykładowych map.

Instrukcja obsługi

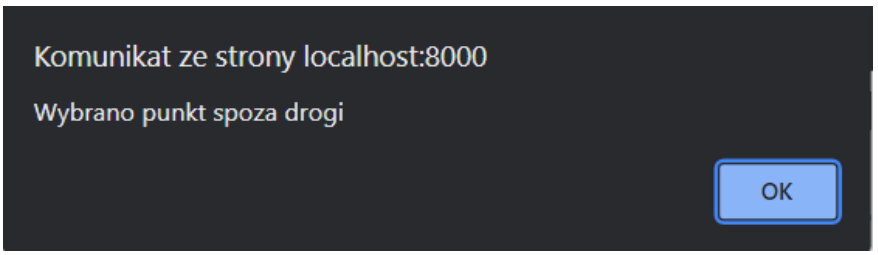
Po uruchomieniu aplikacji należy wczytać interesujący nas obrazek przy pomocy poniżej przedstawionego pola (Rys.1) oraz zatwierdzić swój wybór przyciskiem “Zapisz”, co spowoduje wyświetlenie obrazka w punkcie nr 2.



Rys. 1 Wygląd pierwszej części interfejsu

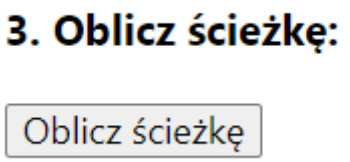
Następnie, w punkcie 2, użytkownik wybiera punkt początkowy oraz końcowy ścieżki poprzez kliknięcie w odpowiednim miejscu na obrazie. Każdy wybór zostaje poddany walidacji, aplikacja sprawdza, czy wskazany punkt znajduje się w obszarze drogi (biały kolor) i wyświetla jego współrzędne pod obrazem (Rys.2). Jeżeli wybrano punkt leżący poza drogą (kolor czarny), zostaje wyświetlony alert z taką informacją (Rys.3).

Rys. 2 Przedstawienie wyboru punktów na obrazie



Rys. 3 Komunikat o błędnym wyborze punktu na obrazie

W punkcie trzecim odbywa się uruchamianie algorytmu obliczającego najkrótszą ścieżkę po kliknięciu przycisku “Oblicz ścieżkę” (Rys.4). Obliczenia mogą trwać do 5 minut.



Rys. 4 Prezentacja punktu nr 3 aplikacji

Podział ról w grupie projektowej

Generowanie danych - Wiktor Urban
Obróbka obrazu - Mariusz Marszałek
Główna część algorytmu - Ryszard Borzych
Ustalenie szerokości trasy - Bartłomiej Bożek
Frontend - Monika Kidawska, Przemysław Rewiś
Dokumentacja - Tomasz Praszkiewicz

Szczegółowy opis algorytmów i narzędzi

Parametry i opis preprocessing danych

1. Konwersja obrazu na skalę szarości.
2. Rozszerzanie obrazu funkcją cv2.dilate z jądrem 5x5 w postaci jedynek.
3. Binaryzacja obrazu przy użyciu algorytmu otsu. Białe piksele interpretowane są jako droga, czarne natomiast jako miejsce gdzie nie ma drogi.

Parametry i opis algorytmu mrówkowego

Opis algorytmu mrówkowego

Cały obraz jest traktowany jako tablica wartości prawda-fałsz.

1. Ruch mrówek

Po wyborze punktu początkowego i końcowego na obrazie, w miejscu początkowym są tworzone obiekty klasy mrówka. Następnie cyklicznie odbywa się aktualizacja pozycji wszystkich mrówek. Warunkiem zakończenia algorytmu jest znalezienie liczby dróg zadanej z góry.

Na początku każda mrówka sprawdza, czy nie jest w odległości mniejszej niż pięć jednostek w każdej osi od celu. W takim przypadku ścieżka jest uważana za znalezioną (odległość ta jest niezerowa ze względu na różne możliwe wartości *speed*). Mrówka jest w takim wypadku teleportowana z powrotem na start.

W każdym kroku mrówka wybiera w sposób losowy położenie odległe o *speed* na siatce pikseli. Położenie może być diagonalnie, wertykalnie lub horyzontalnie od obecnego położenia mrówki. Odbywa się sprawdzenie, czy na dane pole można się poruszyć - czy nie leży poza obrazem i czy jest ono białe. Spośród dostępnych pól wybierane jest losowe z wykorzystaniem metody ruletki, gdzie wartość pola to wartość feromonów na danym polu - im wyższa tym wyższe prawdopodobieństwo wybrania pola. Pole jest zapamiętywane jako odwiedzone przez daną mrówkę.

Każda mrówka posiada pamięć pól odwiedzonych. Są one przedstawione w postaci macierzy o rozmiarze równym obrazowi zawierającej zmienne typu Boolean. Jest to główne ograniczenie pamięciowe algorytmu. Pamięć obliczana jest z następującego wzoru:

$$memory = width * height * ants * boolSize$$

Nie pozwala to na zbyt dużą liczbę mrówek. Prz przejściu przez dane pole, jego wartość jest oznaczana jako True. Zastosowano to rozwiązanie zamiast zapamiętywania kolejnych położeń jako tablica krotek, gdyż:

- mrówki wielokrotnie odwiedzają to samo miejsce - powodowałoby to niepotrzebny wzrost rozmiaru tablicy
- operacja append jest w Pythonie czasochłonna

Dodatkowo każda mrówka pamięta swoje ostatnie położenie. Stanowi to kolejne zabezpieczenie na wypadek, gdyby mrówka znalazła się na polu, z którego nie ma żadnego wyjścia. Wówczas wycofuje się ona na poprzednie odwiedzone przez siebie pole.

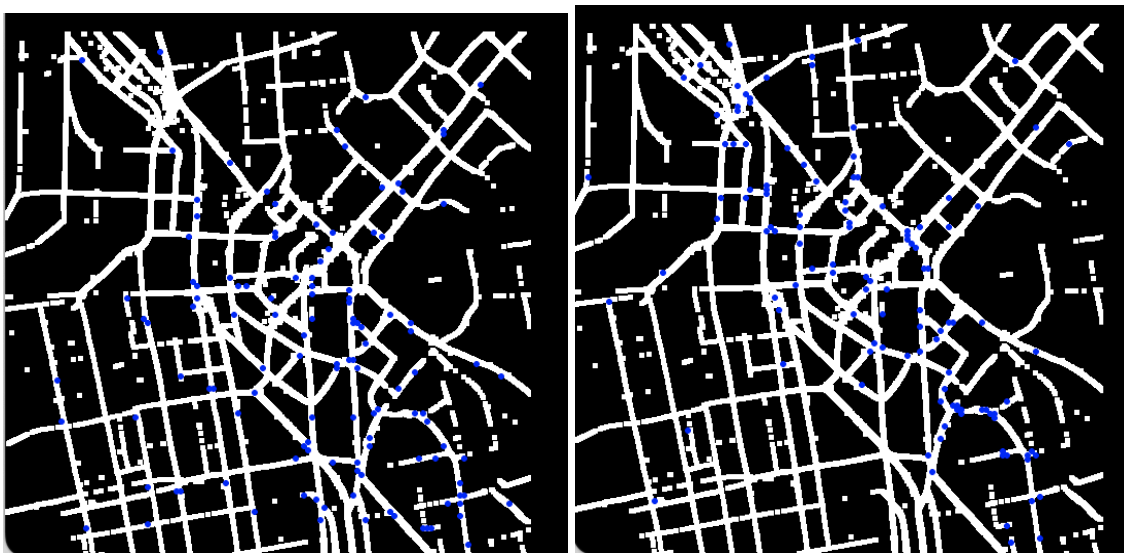
2. Feromony

Bazowa wartość feromonów na starcie algorytmu jest zgodna ze wzorem:

$$pheromones(i,j) = textureSize * \sqrt{2} - \sqrt{(i - goal_x)^2 + (j - goal_y)^2}$$

gdzie *i,j* to współrzędne danego piksela obrazu. Powoduje to, że ruch mrówek nie jest całkiem losowy, a w naturalny sposób są one "przyciągane" w stronę celu. Efekt ten jest relatywnie niewielki przez rozmiar tekstury.

Dodatkowo, w przypadku znalezienia celu, osobnik dodaje swoją drogę do śladu feromonalnego z dużą wagą. Sprawia to, że kolejne mrówki łatwiej odnajdują drogę do celu. Na poniższej ilustracji widać jak przy pierwszym odnalezieniu celu mrówki są rozłożone po labiryncie, a po kilkukrotnym odnalezieniu drogi zaczynają się układać w kierunku celu.



Rys. 5 Ruch mrówek po ścieżkach

Powyższe rysunki zostały wygenerowane przy pomocy wizualizacji algorytmu stworzonej przy użyciu biblioteki kivy.

Parametry:

liczba mrówek szukających celu = 125
rozmiar obrazu = 512
prędkość poruszania się mrówek = 4

Konfiguracja generatora przykładowych map

1. Należy zainstalować stable diffusion webui
   https://github.com/AUTOMATIC1111/stable-diffusion-webui
2. Uruchomić webui poleceniem
   ./webui.sh --precision full --no-half
3. Wczytać model stable_diffusion_checkpoint: mdjrny-v4.ckpt [5d5ad06cc2]
   (https://huggingface.co/prompthero/openjourney/blob/main/mdjrny-v4.ckpt)
4. W webui wprowadzić konfigurację, a następnie nacisnąć przycisk generate
   mode: img2img
   starting_image: starting_image.png
   positive_prompt: map, network of white roads, dark background, curvy, highways, streets, and avenues, infrastructure, aerial view 2d, monochrome, scheme, layout
   negative_prompt: colorful, river,
   sampling method: LMS
   sampling_steps: 66
   resize_mode: just_resize
   restore_faces: false
   tiling: false
   tab: resize_to
   width: 512
   height: 512
   batch_count: 100
   batch_size: 1
   cfg_scale: 7
   denoising_strength: 0.75
   seed: 2
   script: None