Руководство пользователя

Введение

Данный проект разработан в рамках хакатона "Лидеры цифровой трансформации". Сервис предназначен для определения географической привязки сцен спутниковых снимков и корректировки битых пикселей. Сервис полностью автоматизирован и не требует промежуточной настройки в процессе работы.

Возможности

- Высокоскоростной алгоритм обработки данных
- Автоматическое определение геопозиции
- Обнаружение и корректировка битых пикселей
- Поддержка вывода в нескольких форматах (CSV, GeoJSON, GeoTIFF)
- RESTful API для легкой интеграции

Использование

Вы можете использовать сервис как по API, так и с помощью скрипта.

Использование с помощью API

Для обработки изображений сервис предоставляет API для взаимодействия. Вы также можете использовать такие инструменты, как `curl` или Postman.

Пример

```
import requests

scene_path = 'path_to_your_image.tif'
layout_name = 'your_layout_image.tif'
url = 'http://localhost:8000/process'
with open(scene_path, 'rb') as f:
    files = {'file': (scene_path, f)}
    data = {'layout_name': layout_name}
    response = requests.post(url, data=data, files=files)
    print(response.json())
```

Ответ:

```
{'task_id': '2823d72a-0760-4219-a75b-e50e176a1287'}
```

Для получения результатов обработки используйте в запросах в качестве параметра полученный `task_id` :

```
task_id = '2823d72a-0760-4219-a75b-e50e176a1287'  # ID, полученный при запуске задачи
url = f"http://localhost:8000/coords?task_id={task_id}"

response = requests.get(url)
print(response.json())
```

Ответ:

{
"layout_name": "layout_2021-06-15.tif",
"crop_name": "crop_0_0_0000.tif",
"ul": "402343.0; 5796618.8",
"ur": "414068.6; 5796388.8",
"br": "413511.5; 5778375.4",
"bl": "401785.8; 5778605.4",
"crs": "EPSG:32637",
"start": "2024-06-15T11:52:35",
"end": "2024-06-15T11:52:53"
}

Использование с помощью скрипта

Чтобы обработать изображения с помощью скрипта, вы можете запустить его внутри Docker-контейнера или настроить локальную среду Python.

Использование скрипта в Docker-контейнере

Запустите следующую команду:

```
docker run --rm -v .:/app -v /layouts:<layouts_dir> nikolove18 python -m src.main \
    --layout_name <layout_name> --crop_name <path_to_crop_image_inside_project_dir>
```

Использование скрипта локально

1. Настройте окружение Python и установите зависимости:

   ```
   pip install -r requirements.txt
   ```

2. Запустите скрипт:

   ```
   python main.py --crop_name <path_to_crop_image> --layout_name <path_to_layout_image>
   ```

API

POST `/process`

Запускает задачу обработки изображения.

- **Параметры:**
  - `layout_name` (string): Имя файла подложки.
  - `file` (file): Файл изображения для обработки.
- **Ответ:**
  - `task_id` (string): ID задачи обработки.

GET `/coords`

Получает геопривязанные координаты обработанного изображения.

- **Параметры:**
  - `task_id` (string): ID задачи обработки.

- **Ответ:**
  - JSON-объект с координатами и другой информацией об обработке:
    {
    "layout_name": "имя подложки",
    "crop_name": "имя снимка",
    "ul": "координаты верхнего левого угла",
    "ur": "координаты верхнего правого угла",
    "br": "координаты нижнего правого угла",
    "bl": "координаты нижнего левого угла",
    "crs": "система координат",
    "start": "время начала обработки",
    "end": "время окончания обработки"
    }

GET `/bug_report`

Получает отчет о коррекции битых пикселей.

- **Параметры:**
  - `task_id` (string): ID задачи обработки.
- **Ответ:**
  - JSON-объект с деталями отчета.

GET `/download/geojson`

Скачивает результат в виде файла GeoJSON.

- **Параметры:**
  - `task_id` (string): ID задачи обработки.
- **Ответ:**
  - Файл GeoJSON.

GET `/download/geotiff`

Скачивает результат в виде файла GeoTIFF с геопривязкой.

- **Параметры:**
  - `task_id` (string): ID задачи обработки.
- **Ответ:**
  - Файл GeoTIFF.

GET `/download/corrected_pixels`

Скачивает откорректированные пиксели в исходной системе координат.

- **Параметры:**
  - `task_id` (string): ID задачи обработки.
- **Ответ:**
  - Файл GeoTIFF.

GET `/download/bug_report`

Скачивает отчет о коррекции битых пикселей в формате CSV.

- **Параметры:**
  - `task_id` (string): ID задачи обработки.
- **Ответ:**
  - CSV-файл.

Логи и результаты

Логи, результаты обработки и отчеты хранятся в директории tasks/{task_id}. В этой директории находятся:

- `process.log` - файл с логами процесса обработки.
- `coords.csv` - файл с координатами углов сцены и дополнительными параметрами обработки.
- `coords.txt` - файл с координатами углов сцены
- `coords.geojson` - GeoJSON с координатами углов сцены в системе координат подложки.
- `bug_report.csv` - отчет по восстановленным битым пикселям.
- `corrected.tif` - файл GeoTIFF с восстановленными битыми пикселями.
- `aligned.tif` - файл GeoTIFF сцены с географической привязкой к подложке.