Описание технической архитектуры

программного обеспечения «Орбис КОЗ 1 Экспедиция.DS»

Москва, 2026

СОДЕРЖАНИЕ

Термины и определения 3

1. Общие сведения 4

2. Описание архитектуры приложения 5

2.1. Описание функций и файлов 5

2.2. Диаграмма последовательностей 9

3. Описание архитектурного стиля 10

Термины и определения

НТИ – национальная технологическая инициатива;

ПО – программное обеспечение;

1. Общие сведения

Настоящий документ содержит описание техническое архитектуры программного обеспечения «Орбис КОЗ 1 Экспедиция.DS».

Исключительные права на программное обеспечение принадлежат Фонду НТИ (далее – Компания).

Настоящий документ подлежит размещению на официальном сайте Компании в сети Интернет по адресу: https://nti.fund/about/activity/information.php (далее – официальный сайт).

ПО не имеет пользовательского графического интерфейса, управление осуществляется с использованием консоли.

Эффективность распознавания оптимизирована для работы с данными высокого и сверхвысокого разрешения. Для цифровых моделей рельефа (ЦМР), полученных по данным лидара, рекомендуется плотность точек > 10-15 pts/m², а для ортофотопланов — пространственное разрешение лучше 0.5 м/пиксель.

2. Описание архитектуры приложения

2.1. Описание функций и файлов

ПО «Орбис КОЗ 1 Экспедиция.DS» состоит из следующих файлов и функций:

- inference/solution.py: Основной исполняемый модуль. Отвечает за весь конвейер обработки: загрузку изображений, препроцессинг, запуск инференса ансамбля моделей, постобработку детекций, векторизацию и сохранение результатов в формате GeoJSON.
- ensemble_config.yaml: Конфигурационный файл, определяющий состав, веса и параметры ансамбля нейросетевых моделей. Позволяет гибко управлять процессом обнаружения без изменения кода.
- models/: Директория, содержащая веса предобученных моделей YOLO для обнаружения различных классов объектов (поселения, дороги, укрепления, поля, архитектура и др.).
- requirements.txt: Файл со списком всех необходимых Python-зависимостей и их версий для гарантии воспроизводимости среды выполнения.
- Dockerfile: Инструкция для автоматической сборки изолированного и переносимого Docker-контейнера со всем окружением, необходимым для работы ПО.

Пример запуска:

Локальный запуск (без Docker) после установки зависимостей:

bash

python inference/solution.py /путь/к/вашим/tiff_файлам /путь/для/сохранения/результатов

#Запуск с пользовательскими параметрами

python inference/solution.py dataset/test/ results/ --tile-size 1280 --overlap 128 --scale-factor 0.25

Запуск через Docker:

bash

Собрать Docker-образ

docker build -t orbis-kos:latest .

Запустить контейнер с обработкой данных

docker run --rm --gpus all \

  -v "$(pwd)/dataset/test":/app/input \

  -v "$(pwd)/results":/app/output \

  orbis-kos:latest \

  /app/input /app/output --tile-size 1280 --overlap 128

Ключевые особенности запуска:

При локальном запуске требуется предварительная установка Python 3.10+ и всех зависимостей из requirements.txt. Docker-вариант автоматически создает изолированную среду со всеми необходимыми компонентами

Описание содержания файлов:

– inference/metadata.json: Служебный файл, содержащий метаинформацию о решении, такую как версия ПО, авторство, описание входных/выходных форматов и системные требования.

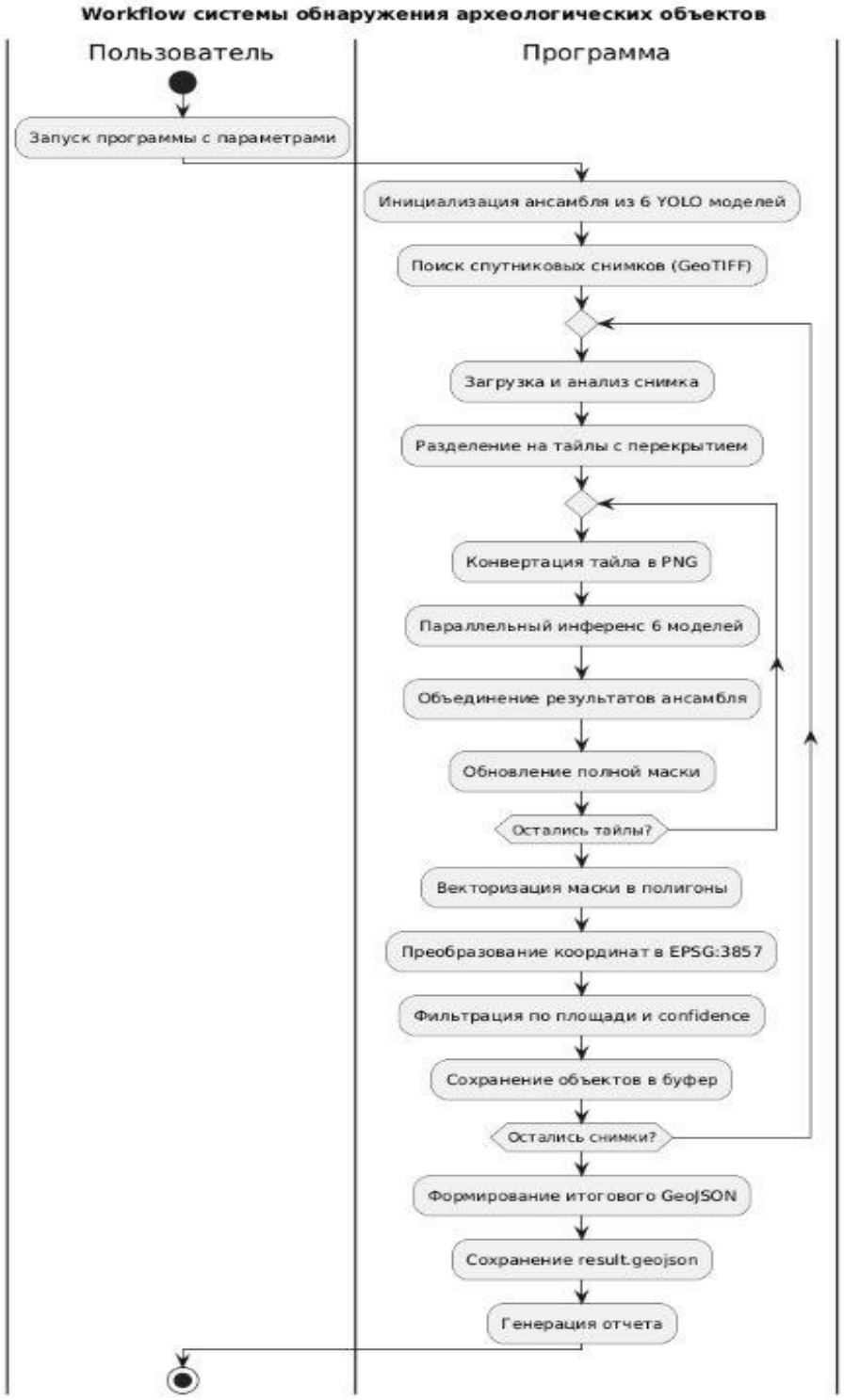
– train_dataset/: Директория с окончательными наборами данных (изображения и разметка), использованными для обучения финальных версий моделей.
– utils/dataset.py: Утилита для подготовки и анализа данных. Создает структурированный датасет, обрабатывая пути к TIFF-изображениям и связанным с ними файлам разметки.
– utils/yolo_files_30_rez.py: Скрипт для конвертации исходных данных и разметки в формат, пригодный для непосредственного обучения моделей YOLO (например, создает файлы train.txt, val.txt и аннотации в соответствующем формате).
– notebooks/ensemble_yolo.ipynb: Основной Jupyter-ноутбук для исследовательского анализа данных (EDA), их разделения на тренировочную/валидационную/тестовую выборки и пошагового обучения ансамбля моделей YOLO.
– notebooks/augment_*.ipynb: Серия ноутбуков, реализующих методы аугментации данных (например, повороты, отражения, изменение яркости/контраста) для увеличения размера и разнообразия обучающей выборки по конкретным, менее представленным классам объектов (поселения, укрепления, караваны, архитектура). Это необходимо для повышения сбалансированности датасета и улучшения качества детекции редких классов.
– notebooks/visual_mask.ipynb: Инструмент для визуализации и проверки корректности масок разметки, накладывая их на исходные изображения. Используется для контроля качества данных на этапах разметки и предобработки.
– changed/: Директория, содержащая дополнительную и исправленную разметку для TIFF-изображений. Часто сопровождается документацией (README или комментариями) с пояснениями о внесенных

изменениях, что обеспечивает прозрачность и отслеживаемость истории подготовки данных.

– README.md: Корневой файл документации проекта. Обычно содержит общее описание, инструкции по быстрому старту (установка зависимостей, запуск), объяснение структуры проекта и другую важную информацию для пользователей и разработчиков.

2.2. Диаграмма последовательностей

Диаграмма последовательности программного обеспечения «Орбис КОЗ 1 Экспедиция.DS» приведена на рисунке:



3. Описание архитектурного стиля

ПО «Орбис КОЗ 1 Экспедиция.DS» реализовано в модульном архитектурном стиле с элементами конвейерной (pipeline) обработки и принципами объектно-ориентированного программирования (ООП). Данный стиль выбран для обеспечения высокой связности компонентов, отвечающих за конкретные задачи, и их слабой связанности, что упрощает поддержку, тестирование и масштабирование системы.

Ключевые характеристики архитектурного стиля:

1. Модульность (Modularity): Кодовая база разделена на четко определенные классы и функции с единой ответственностью.

2. Конвейерная обработка (Pipeline Processing): Архитектура следует линейному потоку данных «ETL-типа» (Extract, Transform, Load):

– Извлечение (Extract): Чтение GeoTIFF-файлов и их геопространственных метаданных.

– Трансформация (Transform): Последовательная обработка через цепочку операций: масштабирование, тайлирование, инференс моделей, постобработка, векторизация.

– Загрузка (Load): Сериализация и сохранение итоговых векторных объектов в стандартный формат GeoJSON.

3. Использование паттернов проектирования:

– Стратегия (Strategy): Разные алгоритмы обработки (например, тайлирование) инкапсулированы в отдельные классы, что позволяет гибко менять их, не затрагивая основную логику.

– Фасад (Facade): Основной скрипт solution.py предоставляет упрощенный интерфейс (фасад) для запуска сложной внутренней системы,

скрывая детали работы ансамбля моделей, тайлинга и трансформации координат.

4. Конфигурационное управление: Критические параметры системы (пороги уверенности, минимальные площади, состав ансамбля моделей) вынесены в отдельные конфигурационные структуры (CLASS_THRESHOLDS, MIN_AREAS_M2, ensemble_config.yaml). Это позволяет адаптировать работу ПО под различные сценарии без изменения исходного кода.

5. Слоистая обработка данных: Архитектура предполагает последовательное преобразование данных между различными представлениями: растр (GeoTIFF) → тензор (для нейросети) → пиксельная маска → векторный полигон (GeoJSON). Каждый слой абстрагирован от деталей предыдущего.

Адрес электронной почты для связи по вопросам о программе при необходимости дополнительной консультации по установке: **ntifundsoft@nti.fund**