ОПИСАНИЕ СИСТЕМЫ

Программное обеспечение «Система поиска CVS КОЗ 1 Автономный поиск»

Аннотация:

В документе представлено описание ПО «Система поиска CVS КОЗ 1 Автономный поиск».
Система предназначена для автоматизации процесса автономного поиска.

Листов 13

Москва 2024

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

Сокращение	Детальная расшифровка
«Система»	Система поиска CVS КОЗ 1 Автономный поиск
ПО	Программное обеспечение
БПЛА	Беспилотный летательный аппарат

ВВЕДЕНИЕ

Документ представляет собой подробное описание программного обеспечения «Система поиска CVS КОЗ 1 Автономный поиск». В нём представлены Описание общих сведений, функции программы, её структура, принципы работ.

Для работы с системой необходимы технические специалисты с высшим образованием и квалификацией не ниже инженера.

1 Общие сведения

Основная функция системы поиска CVS КОЗ 1 Автономный поиск – автоматическое детектирование людей на изображениях, полученных с беспилотников.

1.1 Ключевые особенности

– Программа способна принимать и обрабатывать изображения различных размеров.

– Программа сама производит препроцессинг входного изоображения, включая нормализацию, масштабирование и преобразование в формат, подходящий для нейросетевого алгоритма.

– Программа оптимизирована для работы на бортовых нейросетевых вычислителях семейства Nvidia Jetson.

– Результаты обработки предоставляются в открытом формате.

– Программа поддерживает пакетную обработку.

– Использует нейросетевые архитектуры YOLO v8 и YOLO v11 для детектирования людей и других объектов.

1.2 Преимущества системы

– Автоматический поиск людей на изображениях значительно сокращает время и трудозатраты на анализ данных, полученных с БПЛА.

– Оптимизация под архитектуру Nvidia Jetson обеспечивает обработку изображений на ограниченных вычислительных ресурсах.

– Использование современных нейросетевых архитектур YOLO v8 и YOLO v11 гарантирует высокую точность детектирования объектов.

– Программа работает с изображениями разных размеров и форматов, обеспечивая широкую область применения.

– Пакетная обработка сокращает общее время обработки.

2 Функциональные характеристики

2.1 Назначение программного обеспечения

– Программное обеспечение, которое анализирует данные, полученные с помощью БПЛА, используя нейросетевые алгоритмы для нахождения людей.

3 Описание назначения функций

Описание назначения функций (синтаксических единиц в файлах с исходным кодом системы), их входных и выходных данных, которое должно быть размещено в самих файлах с исходным кодом

3.1 Класс cvs_cpp_detector_module.Detector

```
detector = cvs_cpp_detector_module.Detector("","11m_2_3k.eng",
conf, iou)
```

Входные данные при инициализации:

– путь к ONNX-файлу модели (в данном случае "");

– путь к заранее сгенерированный файлу модели в формате TensorRT engine (в данном случае "11m_2_3k.eng"). Если по указанному пути engine-файл отсутствует или произошла ошибка инициализации, выполняется попытка инициализации из ONNX-файла. После успешной инициализации скомпилированный engine сохраняется по указанному пути;

– conf – (Confidence Score) устанавливает минимальный порог уверенности для обнаружений (в данном случае 0.35). Объекты, обнаруженные с достоверностью ниже этого порога, будут проигнорированы. Настройка данного значения может помочь уменьшить количество ложных срабатываний;

– iou – порог Intersection Over Union (IoU) для не максимального подавления (NMS) (в данном случае 0.2). Более низкие значения приводят к уменьшению количества обнаружений за счет устранения перекрывающихся ячеек, что полезно для уменьшения количества дубликатов.

– Назначение:

– инициализирует детектор и выполняет предсказания на входных изображениях через метод infer.

3.2 Функция predict

```
def predict(images: Union[List[np.ndarray], np.ndarray]) ->
List[List[dict]]:
```

Входные данные:

– images: список изображений или одно изображение в формате np.ndarray.

Выходные данные:

– список списков словарей с результатами предсказаний, где каждый словарь содержит координаты объектов, метки и вероятности.

Назначение:

– выполняет инференс модели на переданных изображениях и возвращает результаты детектирования.

3.3 Класс Detector:

Назначение:

– представляет обёртку для использования YOLO детектора на TensorRT API из Python.

Конструктор:

```
Detector(const std::string& model_path,
const std::string& engine_path,
float conf_thr, float iou);
```

Входные данные:

– model_path — путь к ONNX-модели;

– engine_path — путь к TensorRT-engine;

– conf_thr — порог уверенности для обнаружений;

– iou — порог IoU для NMS.

Назначение:

– инициализирует TensorRT-детектор;

– создаёт экземпляр через интерфейс IDetectorTRT.

Исключения:

– std::runtime_error, если инициализация не удалась.

3.4 Метод infer:

```
std::vector<std::unordered_map<std::string, double>> infer(const
py::array_t<uint8_t>& image);
```

Входные данные:

– image — трёхмерный массив NumPy, представляющий RGB-изображение.

Выходные данные:

– список словарей, содержащих результаты обнаружения (координаты, метки, вероятности).

Назначение:

– выполняет инференс, используя метод detect из IDetectorTRT;

– конвертирует результаты в удобный формат (Python-список словарей).

Исключения:

– std::runtime_error, если входное изображение не является 3D.

3.5 Класс cvs::DetectorTRT:

Назначение:

– является реализацией интерфейса IDetectorTRT для выполнения детектирования с использованием TensorRT.

3.6 Метод initialize:

```
Status initialize(const std::string& model_path,
const std::string& engine_path,
float conf_thr, float iou) noexcept override;
```

Входные данные:

– model_path — путь к ONNX-модели.

– engine_path — путь к TensorRT-движку.

– conf_thr — порог доверия.

– iou — порог IoU.

Выходные данные:

– возвращает статус успешности инициализации.

Назначение:

– загружает движок из engine_path или создаёт из model_path.

3.7 Метод detect:

```
std::vector<DetectItem> detect(
const ImageView& image_view) override;
```

Входные данные:

– image_view — представление входного изображения.

Выходные данные:

– список объектов типа DetectItem, каждый из которых содержит координаты, метку и вероятность.

Назначение:

– выполняет детектирование объектов.

3.8 Метод preprocess_image:

```
Status preprocess_image(const ImageView& image,
const gpuprocess::ROI& roi,
PreprocessParams& preprocess_params);
```

Входные данные:

– image — объект ImageView для входного изображения.

– roi — регион интереса (ROI).

– preprocess_params — параметры предварительной обработки.

Назначение:

– выполняет масштабирование, добавляет паддинги и преобразует изображение в формат, совместимый с TensorRT.

3.9 Метод preprocess_image_LetterBox:

```
Status preprocess_image_LetterBox(const ImageView& image,
const gpuprocess::ROI& roi,
PreprocessParams& preprocess_params);
```

Назначение:

– аналогично preprocess_image, но дополнительно выполняет letterbox-трансформацию для сохранения пропорций.

3.10 Метод preprocess_image_LetterBox_opencv:

```
Status preprocess_image_LetterBox_opencv(
const ImageView& image,
const gpuprocess::ROI& roi,
PreprocessParams& preprocess_params);
```

Назначение:

– аналогично preprocess_image_LetterBox, но использует OpenCV ресайз вместо NPP.

3.11 Метод infer:

```
Status infer() noexcept;
```

Назначение:

– запускает инференс, используя загруженный engine.

Выходные данные:

– статус выполнения (успех или ошибка).

3.12 Метод postprocess:

```
std::vector<DetectItem> postprocess(const ImageView& image,
const PreprocessParams& preprocess_params);
```

Входные данные:

– результаты инференса.

Назначение:

– применяет фильтрацию подавлением немаксимальных значений (NMS) и преобразование координат.

3.13 Метод validate_output_dims:

```
Status validate_output_dims(const nvinfer1::Dims& dims) const;
```

Назначение:

– проверяет корректность выходных размеров модели YOLO.

3.14 Метод init_buffers:

```
Status init_buffers();
```

Назначение:

- инициализирует входные и выходные буферы для TensorRT API;
- аллоцирует память на GPU и CPU.

4 Структура программного обеспечения

На рисунке 1 показана структура проекта «Системы»

```
|-- cvs_detector
|   |-- CMakeLists.txt
|   |-- api
|   |   |-- core_status.h
|   |   |-- defines.h
|   |   |-- image_view.h
|   |   `-- itrt_detector.h
|   |-- pybind11
|   `-- src
|       |-- TLogger.h
|       |-- cuda_helpers.cpp
|       |-- cuda_helpers.h
|       |-- cvs_detector_pybind.cpp
|       |-- detection_postprocess.h
|       |-- gpu_image.cpp
|       |-- gpu_image.h
|       |-- image_view.cpp
|       |-- trt_detector.cpp
|       |-- trt_detector.h
|       `-- utils
`-- solution
    |-- 11m_2_3k.eng
    |-- cvs_cpp_detector_module.cpython-310-aarch64-linux-gnu.so
    |-- metadata.json
    `-- solution.py
```

Рисунок 1- структура проекта

4.1 Сущности ПО

– Исходные данные - трёхмерный массив NumPy, представляющий RGB-изображение.

– Нейронная сеть- Обработка изображений различных размеров

– Выходные данные- Список словарей с результатами предсказаний, где каждый словарь содержит координаты объектов, метки и вероятности