MSU ROVER TEAM

ОТКРЫТЫЕ БИБЛИОТЕКИ ДЛЯ АВТОНОМНОЙ НАВИГАЦИИ ШЕСТИКОЛЕСНОГО ПРОТОТИПА МАРСОХОДА (РОВЕРА) ПО УМЕРЕННО ПЕРЕСЕЧЁННОЙ НЕЗНАКОМОЙ МЕСТНОСТИ С ВИЗУАЛЬНЫМ РАСПОЗНАВАНИЕМ ЦЕЛИ НАВИГАЦИИ.

СОГЛАСОВАНО: Научный эксперт проекта, к.ф.-м.н.	УТВЕРЖДАЮ: Руководитель проекта
_____ В.М. Буданов	_____ А.А. Смирнов
05.12.2025	05.12.2025

Открытая библиотека распознавания указателей направления движения, определения расстояния и направления к указателям.

Руководство программиста.

ЛИСТ УТВЕРЖДЕНИЯ

MSUROVERTEAM-CV-V1.0.0

(открытая библиотека в сети Интернет)

ИСПОЛНИТЕЛИ:

_____ Я.И. Шейпак
03.12.2025

_____ М.Д. Реппо
03.12.2025

2025

MSU ROVER TEAM

УТВЕРЖДЕНО

ОТКРЫТЫЕ БИБЛИОТЕКИ ДЛЯ АВТОНОМНОЙ НАВИГАЦИИ ШЕСТИКОЛЕСНОГО ПРОТОТИПА МАРСОХОДА (РОВЕРА) ПО УМЕРЕННО ПЕРЕСЕЧЁННОЙ НЕЗНАКОМОЙ МЕСТНОСТИ С ВИЗУАЛЬНЫМ РАСПОЗНАВАНИЕМ ЦЕЛИ НАВИГАЦИИ.

Открытая библиотека распознавания указателей направления движения, определения расстояния и направления к указателям.

Руководство программиста.

MSUROVERTEAM-CV-V1.0.0

(открытая библиотека в сети Интернет)

Листов 10.

2025

АННОТАЦИЯ.

Открытая библиотека распознавания указателей направления движения, определения расстояния и направления к указателям разработана в рамках проекта «Разработки открытых библиотек для автономной навигации шестиколесного прототипа марсохода (ровера) по умеренно пересечённой незнакомой местности с визуальным распознаванием цели навигации». Проект выполнен на средства выделенные «Фондом содействия развитию малых форм предприятий в научно-технической сфере» (Фонд содействия инновациям) по договору предоставления гранта № 64ГУКодИИС13-D7/102402 от 23 декабря 2024г.

Под полностью автономным режимом навигации (движения) в данном проекте понимается режим, при котором ровер с Аккермановой геометрией поворота самостоятельно, без команд оператора (человека), передвигается по умеренно пересечённой и незнакомой местности по указателям направления движения до указателя конечной цели, может выполнить заранее запрограммированные действия у каждого указателя и самостоятельно вернуться обратно к месту старта. При этом оператор может просматривать на своем мониторе видеоизображения и телеметрию, предаваемые с ровера.

В качестве указателя направления движения применяется знак (с белым фоном и с черной стрелкой), размером 300 х 200 мм, поднятый на высоту 100 - 150 мм над поверхностью. Размеры стрелки на знаке приведены в Приложении А к настоящему Руководству. В качестве указателя конечной цели используется оранжевый дорожный конус, с размерами соответствующим п.4.3.1.1 ГОСТ32758-2014.

«Открытая библиотека распознавания указателей направления движения, определения расстояния и направления к указателям» предназначена для автоматического обнаружения/распознавания указателей направления движения и конечной цели, автоматического определения расстояния до них и передачи информации о детектированных навигационных объектах в модуль локализации для принятия решений о движении ровера.

«Открытая библиотека распознавания указателей направления движения, определения расстояния и направления к указателям» разработана на языке программирования Python 3, с использованием нейросетевой модели YOLOv8 (Ultralytics), для платформы ROS2 Humble (Robot Operating System 2 версии Humble).

СОДЕРЖАНИЕ.

Общие сведения о программе.

«Открытая библиотека распознавания указателей направления движения, определения расстояния и направления к указателям» предназначена для автоматического обнаружения/распознавания указателей направления движения и конечной цели, автоматического определения расстояния до них и передачи информации о детектированных навигационных объектах в модуль локализации для принятия решений о движении шестиколесного прототипа марсохода (ровера) с Аккермановой геометрией поворота по умеренно пересечённой незнакомой местности.

В качестве указателя направления движения применяется знак (с белым фоном и с черной стрелкой), размером 300 х 200 мм, поднятый на высоту 100 - 150 мм над поверхностью. Размеры стрелки на знаке приведены в Приложении А к настоящему Руководству. В качестве указателя конечной цели используется оранжевый дорожный конус, с размерами соответствующим п.4.3.1.1 ГОСТ32758-2014.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ.

Платформа: ROS2 Humble (Robot Operating System 2 версии Humble).

Язык программирования: Python 3.

Нейросетевая модель: YOLOv8 (Ultralytics).

Зависимости:

- opencv-python (компьютерное зрение);
- numpy (численные вычисления);
- ultralytics (YOLO детекция);
- ROS2 (интеграция с роботом).

Измерения:

- калиброванное измерение расстояния методом кусочно-линейной интерполяции по таблице калибровки;
- измерение углов на основе модели камеры-обскуры.

Диапазон работы:

- расстояние: 1-10 метров (калиброванный диапазон);
- угол обзора: зависит от параметров камеры.

СПИСОК ОБЪЕКТОВ ДЛЯ ДОКУМЕНТИРОВАНИЯ.

1. DetectionResult - класс данных результата распознавания.
2. NavigationDetector - основной класс детектора.
3. detect_arrow() - функция распознавания стрелок (left/right).
4. detect_cone() - функция распознавания дорожных конусов.

Дополнительные служебные методы (не требуют отдельного документирования):

- __init__() - конструктор класса;
- detect_all() - комплексная детекция.

Структура программы.

Модуль: eureka_nav_lib.py

Публичные объекты библиотеки:

1. Класс DetectionResult (структура данных);
2. Класс NavigationDetector (основной класс детектора);
3. Функция detect_arrow() (распознавание стрелок);
4. Функция detect_cone() (распознавание дорожных конусов).

1. КЛАСС DetectionResult.

Описание: структура данных для хранения результата распознавания навигационного объекта (стрелки или дорожного конуса).

Тип: dataclass.

Поля (атрибуты).

- object_type: str
  Тип обнаруженного объекта
  Значения: "arrow" (стрелка), "cone" (дорожный конус)
- direction: str
  Направление объекта для навигации
  Значения: "left" (налево), "right" (направо), "none" (нет направления)
- distance_m: float
  Расстояние до объекта в метрах (калиброванное измерение)
- angle_deg: float
  Угол отклонения объекта от центра камеры в градусах
  Отрицательные значения - справа, положительные - слева
- confidence: float
  Уверенность детекции в диапазоне [0.0, 1.0]
- bbox: tuple[int, int, int, int]
  Ограничивающий прямоугольник детекции в формате (x1, y1, x2, y2)

Назначение: передача информации о детектированных навигационных объектах в модуль локализации для принятия решений о движении ровера.

2. КЛАСС NavigationDetector.

Описание: основной класс детектора навигационных объектов для автономного марсохода.

Обеспечивает распознавание стрелок и дорожных конусов с калиброванным измерением расстояния и угла.

Методы инициализации:

- __init__(weights_path: str)
  Инициализация детектора с загрузкой модели машинного обучения.
  Входные параметры:
  - weights_path: Путь к файлу весов YOLO модели (.pt файл)

  Возвращаемое значение: нет

Публичные методы:

- detect_arrow(image: np.ndarray) -> List[DetectionResult]
  Назначение: распознавание стрелок с определением направления движения (левая или правая стрелка).
  Входные параметры:
  - image: Изображение в формате BGR (numpy array, цветное изображение)

  Возвращаемое значение: список объектов DetectionResult, содержащих информацию о всех обнаруженных стрелках. Поле direction содержит "left" или "right".
  Применение: используется для определения направления движения ровера по стрелкам на местности.

- detect_cone(image: np.ndarray) -> List[DetectionResult]
  Назначение: распознавание дорожных конусов для определения целей навигации.
  Входные параметры:
  - image: Изображение в формате BGR (numpy array, цветное изображение)

  Возвращаемое значение: список объектов DetectionResult, содержащих информацию о всех обнаруженных конусах. Поле direction для конусов всегда "none".
  Применение: используется для обнаружения целевых точек навигации (конусов) на местности.

- detect_all(image: np.ndarray) -> List[DetectionResult]
  Назначение: распознавание всех навигационных объектов одновременно (стрелки и конусы).
  Входные параметры:
  - image: Изображение в формате BGR (numpy array, цветное изображение)

  Возвращаемое значение: список всех обнаруженных объектов DetectionResult (стрелки и конусы).
  Применение: комплексный анализ окружения для навигации ровера.

3. ФУНКЦИЯ detect_arrow().

- Полное имя: NavigationDetector.detect_arrow()
- Назначение: функция распознавания стрелок (левых и правых) на изображении с камеры ровера для определения направления движения.
- Входные данные:
  - image: Изображение с камеры ровера (numpy array, BGR формат).
- Выходные данные:
  - список результатов детекции (List[DetectionResult]);
  - каждый результат содержит:
    - тип объекта: "arrow",
    - направление: "left" (левая стрелка) или "right" (правая стрелка),
    - калиброванное расстояние в метрах,
    - угол отклонения в градусах,
    - уверенность детекции (0.0-1.0).
- Алгоритм:
  - детекция стрелок нейросетью YOLO;
  - анализ формы стрелки методом PCA с мажоритарным голосованием:
    - распределение массы пикселей,
    - градиент ширины контура,
    - остроконечность (поиск самого острого угла);
  - калиброванное измерение расстояния по таблице калибровки;
  - вычисление угла относительно центра камеры.
- Применение: основная функция для визуального распознавания направления движения ровера по стрелкам на местности.

4. ФУНКЦИЯ detect_cone().

- Полное имя: NavigationDetector.detect_cone()
- Назначение: функция распознавания дорожных конусов на изображении с камеры ровера для определения целей навигации.
- Входные данные:
  - image: Изображение с камеры ровера (numpy array, BGR формат).
- Выходные данные:
  - список результатов детекции (List[DetectionResult]);
  - каждый результат содержит:
    - тип объекта: "cone",
    - направление: "none" (для конусов не определяется),
    - калиброванное расстояние в метрах,
    - угол отклонения в градусах,
    - уверенность детекции (0.0-1.0).
- Алгоритм:
  - детекция конусов нейросетью YOLO;
  - фильтрация детекций по порогу уверенности;
  - подавление избыточных детекций (Non-Maximum Suppression);
  - калиброванное измерение расстояния по таблице калибровки;
  - вычисление угла относительно центра камеры.
- Применение: функция для обнаружения целевых точек навигации (конусов) на местности, к которым должен двигаться ровер.

Интеграция с модулем локализации.

Библиотека предоставляет набор объектов для передачи информации о навигационных объектах в модуль локализации «Библиотеки автономной навигации по распознанным указателям движения».

1. DetectionResult содержит все необходимые данные для локализации:
   a. Тип объекта (стрелка/конус);
   b. Направление движения (left/right);
   c. Расстояние до объекта (метры);
   d. Угловое положение (градусы);
   e. Уверенность детекции.
2. NavigationDetector.detect_arrow() - основная функция распознавания стрелок с выходом "левая стрелка" или "правая стрелка".
3. NavigationDetector.detect_cone() - функция распознавания дорожных конусов как целей навигации.

Пример использования библиотеки.

Python код:

```
from eureka_nav_simple import NavigationDetector, DetectionResult
import cv2

# Инициализация детектора
detector = NavigationDetector(weights_path="./weights/best.pt")

# Захват изображения с камеры
image = cv2.imread("camera_frame.jpg")

# Распознавание стрелок
arrows = detector.detect_arrow(image)
for arrow in arrows:
    print(f"Стрелка: {arrow.direction}")
    print(f"Расстояние: {arrow.distance_m:.2f} м")
    print(f"Угол: {arrow.angle_deg:.1f}°")

# Распознавание конусов
cones = detector.detect_cone(image)
for cone in cones:
    print(f"Конус на расстоянии {cone.distance_m:.2f} м")
    print(f"Угол: {cone.angle_deg:.1f}°")
```

ПРИЛОЖЕНИЕ А.

Указатель направления движения.