

Разработка алгоритма детектирования поверхности дороги с помощью стереокамеры

Содержание

1. Цели задания	1
2. Содержание задания	1
3. Теоретическое введение	2
3.1. Основные понятия	2
3.2. Алгоритмы построения карты диспаратности	9
3.3. Алгоритм нахождения свободных для движения регионов	14
3.4. KITTI Benchmark Suite	15
4. Выполнение задания	17
4.1. Формат данных	17
4.2. Нахождение карт диспаратности	17
4.3. Нахождение регионов доступных для движения	18
4.4. Оценка результатов работы алгоритма	19
4.5. Выводы	19
Список литературы	20
Приложение. Исходный код программ	27

1. Цели задания

- Ознакомиться с основами языка программирования Python и библиотеки OpenCV.
- Ознакомиться с базовыми алгоритмами цифровой обработки изображений и технического зрения.
- Разработать алгоритм детектирования поверхности дороги с помощью стереокамеры.
- Сформировать навыки написания технических отчетов.

2. Содержание задания

- Установить и настроить Python и OpenCV.
- Скачать со страницы The KITTI Vision Benchmark Suite – Road/Lane Detection Evaluation 2013 следующие данные:
 - left color images,
 - calibration and training labels,
 - right color image extension,
 - development kit.

- По необходимости ознакомиться с основами языка программирования Python, библиотекой для работы массивами NumPy и библиотекой OpenCV.
- Ознакомиться с базовыми принципами обработки стереоизображений. Получить карту диспаритетности для скачанных ранее изображений с помощью алгоритма StereoSGBM и, качественно оценивая результат, выбрать подходящие параметры алгоритма.
- Ознакомиться со статьей [1] и реализовать алгоритм нахождения поверхности, доступной для движения. В качестве исходных карт диспаритетности можно использовать следующие данные:
 - полученные на предыдущем шаге с помощью алгоритма StereoSGBM карты диспаритетности,
 - карты диспаритетности, рассчитанные с помощью алгоритма ELAS,
 - карты диспаритетности, рассчитанные с помощью алгоритма SGM.
- Ознакомиться со статьей [2] и файлами из development kit и по предложенной методике оценить результат разработанного алгоритма.
- Результаты работы представить в виде отчета и исходного кода.

3. Теоретическое введение

3.1. Основные понятия

В данном разделе содержатся базовые сведения о понятиях используемых в стерео зрении. Теоретический материал основан на различных источниках, в том числе на [5, 6, 7].

Проективная геометрия. В геометрии стерео зрения значительную роль играет проективная геометрия. К проективной геометрии существует несколько подходов: геометрический (подобно Евклидовой геометрии, вводятся понятия геометрических объектов, аксиомы и из этого выводятся все свойства проективного пространства), аналитический (всё рассматривается в координатах, как в аналитическом подходе к Евклидовой геометрии), алгебраический (всё рассматривается как некоторая структура над алгебраическим полем).

Однородные координаты. В то время как на обычной Евклидовой плоскости точки описываются парой координат $(x, y)^T$, на *проективной плоскости* точки описываются трехкомпонентным вектором $(x, y, w)^T$. При этом для любого ненулевого числа a , векторы $(x, y, w)^T$ и $(ax, ay, aw)^T$ соответствуют одной и той же точке. А нулевой вектор $(0, 0, 0)^T$ не соответствует никакой точке и не рассматривается. Такое описание точек плоскости называется *однородными координатами (homogeneous coordinates)*.

По аналогии с проективной плоскостью, *точки трехмерного проективного пространства* определяются четырехкомпонентным вектором однородных координат $(x, y, z, w)^T$. Опять же для любого ненулевого числа a , координатные вектора $(x, y, z, w)^T$ и $(ax, ay, az, aw)^T$ соответствуют одной и той же точке.

Между точками трехмерного Евклидова пространства и трехмерного проективного пространства можно установить соответствие. Вектору однородных координат $(x, y, z, w)^T$ при $w \neq 0$ соответствует точка Евклидова пространства с координатами $(x/w, y/w, z/w)^T$. А про точку с вектором однородных координат вида $(x, y, z, 0)^T$ говорят, что она лежит в бесконечности.

Проективное преобразование. С геометрической точки зрения, *проективное преобразование* (*homography, projective transformation*) — это обратимое преобразование проективной плоскости (или пространства), которое переводит прямые в прямые. В координатах проективное преобразование выражается в виде невырожденной квадратной матрицы H , при этом координатный вектор x связан с координатным вектором x' уравнением $x' = Hx$.

Модель проективной камеры. Современные камеры с CCD матрицей хорошо описываются с помощью модели, называемой *проективной камерой* (*projective camera, pinhole camera*). Существуют и другие модели.

Проективная камера определяется следующими параметрами:

- центром камеры C ;
- главной осью Cr — лучом, начинающимся в центре камеры и направленным туда, куда камера смотрит;
- плоскостью изображения — плоскостью, на которую выполняется проецирование точек;
- системой координат на плоскости изображения.

В такой модели, произвольная точка пространства X проецируется на плоскость изображения в точку x , лежащую на отрезке CX , который соединяет центр камеры C с исходной точкой X (см. Рис. 1).

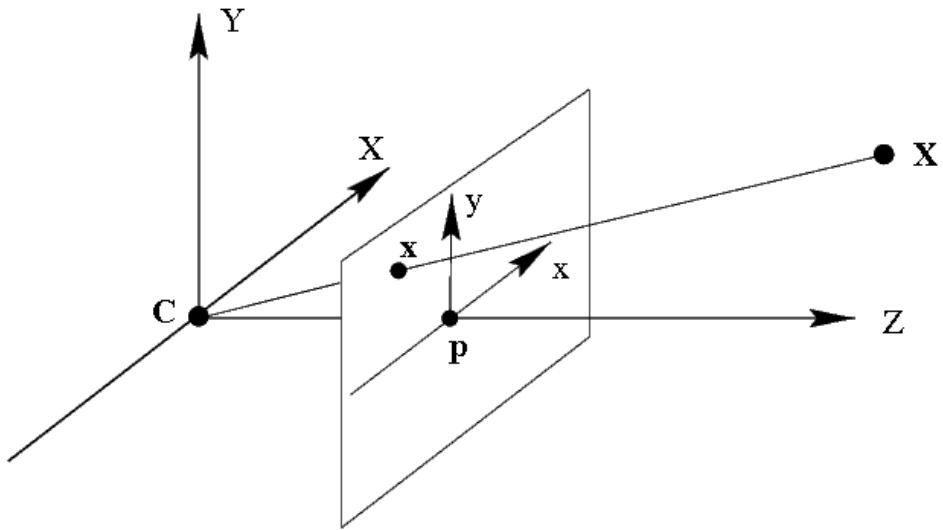


Рис. 1. Модель камеры.

Формула проецирования для идеальной камеры имеет простую математическую запись в однородных координатах:

$$x = PX$$

где X — однородные координаты точки пространства, x — однородные координаты точки плоскости, P — матрица камеры размера 3×4 . Однако, в случае более сложных камер, искажения, вносимые линзами, могут сильно повлиять на результат. В результате функция проецирования принимает более сложный вид и часто записывается как последовательность преобразований.

Дисторсия. В силу неидеальности оптики, на изображениях, полученных с камер, присутствуют искажения-дисторсии (*distortion*). Данные искажения нелинейны, однако могут быть вычисленны.

В результате дисторсии в действительности прямые линии становятся кривыми на изображении, кроме линий, лежащих в одной плоскости с главной осью. Дисторсии не зависят от расстояния до объекта, а зависят только от координат точек, в которые проецируются пиксели объекта. Следовательно, для компенсации дисторсий обычно выполняется преобразование исходного изображения полученного с камеры. Это преобразование будет одним и тем же для всех изображений, полученных с камеры, при условии постоянства фокусного расстояния.

Калибровка. В ситуации, когда известны внутренние параметры камеры и коэффициенты дисторсии, говорят, что камера откалибрована. Матрица камеры P неявно содержит в себе все эти параметры.

Калибровка камер обычно выполняется за счет многократной съемки некоторого калибровочного шаблона, на изображении которого можно легко выделить ключевые точки, для которых известны их относительные положения в пространстве. Далее составляются и решаются (приближенно) системы уравнений, связывающие координаты проекций, матрицы камер и положения точек шаблона в пространстве. Таким образом находятся все необходимые параметры.

Существует несколько различных подходов к решению задачи калибровки:

- Классический подход — алгоритм Roger Y. Tsai [10]. Он состоит из двух этапов: на первом определяются параметры внешней калибровки; на втором — внутренней калибровки и дисторсии. *Параметры внутренней калибровки*: фокусное расстояние, угол наклона пикселей и *принципиальная точка* (точка пересечения плоскости изображения с оптической осью, совпадающая с центром фотографии). В реальных камерах, как правило, бывает немного смещена из-за оптических искажений). *Параметры внешней калибровки* задают положение камеры в мировой системе координат. Параметры внешней калибровки связаны непосредственно с фотографируемой сценой, поэтому (в отличие от параметров внутренней калибровки) каждой фотографии соответствует свой набор этих параметров.
- Новая гибкая технология калибровки камеры, которая была разработана Zhengyou Zhang [9] и основана на использовании плоского калибровочного объекта в виде шахматной доски.
- Автокалибровка — получение калибровочных данных непосредственно по изображениям, причём в сцене не требуется присутствие специальных калибровочных объектов [11].

Существуют общедоступные реализации алгоритмов калибровки, например, Matlab Calibration toolbox. Библиотека OpenCV также включает в себя алгоритмы калибровки камер и поиска калибровочного шаблона на изображении. Более подробно о калибровке см. [5].

Пара камер. Об определении трехмерных координат наблюдаемых точек можно говорить, когда есть как минимум две камеры.

Если имеются две камеры, заданные своими матрицами P и P' в некоторой системе координат, в таком случае говорят, что имеется пара откалиброванных камер. Если центры камер не совпадают, то эту пару камер можно использовать для определения трехмерных координат наблюдаемых точек.

Эпиполярная линия. Перед тем как перейти к описанию метода вычисления трехмерных координат точек, рассмотрим некоторые важные геометрические свойства, связывающие положения проекций точки трехмерного пространства на изображениях с различных камер.

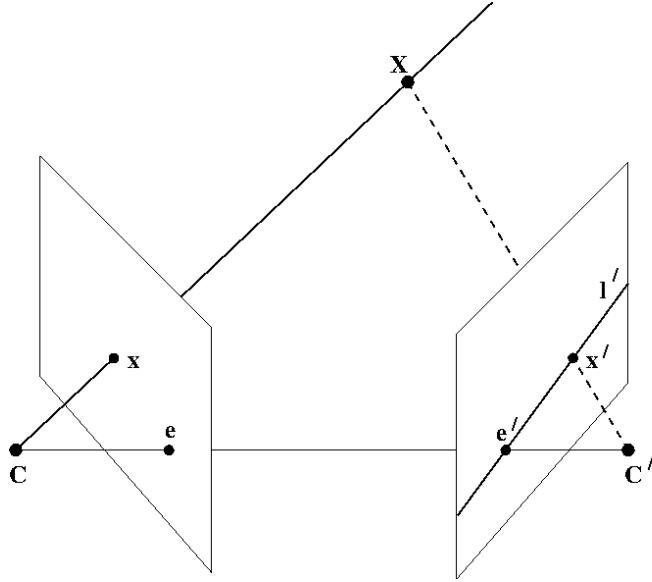


Рис. 2. Эпиполярная геометрия

Пусть имеются две камеры, как изображено на Рис. 2: C — центр первой камеры, C' — центр второй камеры. Точка пространства X проецируется в x на плоскость изображения левой камеры и в x' на плоскость изображения правой камеры. Изображения, полученные левой и правой камерами, будем называть *базовым изображением* и *парным изображением* соответственно.

Прообразом точки x на изображении левой камеры является луч xX . Этот луч проецируется на плоскость второй камеры в прямую l' , называемую *эпиполярной линией (epiline)*. Образ точки X на плоскости изображения второй камеры обязательно лежит на эпиполярной линии l' .

Таким образом, каждой точке x на базовом изображении соответствует эпиполярная линия l' на парном изображении. При этом пара для x на парном изображении может лежать только на соответствующей эпиполярной линии. Аналогично, каждой точке x' на парном изображении соответствует эпиполярная линия l на базовом.

Эпиполярную геометрию используют для поиска стереопар, и для проверки того, что пара точек может быть *стереопарой* (т.е. проекцией некоторой точки пространства).

Фундаментальная матрица. Пусть имеется пара откалиброванных камер, и пусть x — однородные координаты точки на изображении одной камеры, а x' — на изображении второй. Существует такая матрица F размера 3×3 , что пара точек x, x' является стереопарой тогда и только тогда, когда $x'^T F x = 0$.

Матрица F называется *фундаментальной матрицей (fundamental matrix)*. Ее ранг равен двум, и она зависит только от матриц исходных камер P и P' . В главе «Computation of Fundamental Matrix» книги [5] можно найти различные алгоритмы вычисления F по набору точек.

С помощью фундаментальной матрицы вычисляются уравнения эпиполярных линий. Для точки x , вектор, задающий эпиполярную линию, будет иметь вид $l' = Fx$, а уравнение самой эпиполярной линии: $l'^T x' = 0$. Для точки x' , вектор, задающий эпиполярную линию, будет иметь вид $l = F^T x'$.

Существенная матрица. Помимо фундаментальной матрицы, существует еще такое понятие, как *существенная матрица* (*essential matrix*) (часто обозначают E). По существенной матрице можно восстановить положение и поворот второй камеры относительно первой, поэтому она используется в задачах, в которых нужно определить движение камеры.

Триангуляция точек. Процесс определения трехмерной координаты точки по координатам ее проекций называется *триангуляцией* (*triangulation*).

Для нахождения трехмерной координаты точки необходимо решить следующую систему уравнений:

$$\begin{cases} x = PX, \\ x' = P'X. \end{cases}$$

Ректификация. Как уже говорилось, парную точку x' для точки x нужно искать на эпиполярной линии. Соответственно, для упрощения поиска изображения выравнивают так, чтобы все эпиполярные линии были параллельны сторонам изображения (обычно горизонтальны). Более того, изображения выравнивают так, чтобы для точки с координатами (x_0, y_0) (в системе координат изображения) соответствующая ей эпиполярная линия задавалась уравнением $y = y_0$, тогда для каждой точки соответствующую ей парную точку нужно искать в той же строке на изображении со второй камеры. Такой процесс выравнивания изображений называют *ректификацией* (*rectification*). Ректификация позволяет сузить двумерное пространство поиска решений для проблемы соответствия точек до одного измерения. Другими словами: после проведения ректификации, все пиксели изображения с правой камеры, которые могут соответствовать данной точке изображения с левой камеры, будут находиться в одной строке.

Карта диспаритности. Рассмотрим Рис. 3. Пусть X – наблюдаемый объект; B – расстояние между камерами (*стереобаза*); f – фокусное расстояние камер (будем полагать, что одно и то же для двух камер); x_{pr}, x'_{pr} – евклидовы координаты проекций точек x, x' на ось OO' ; z – расстояние от точки X до прямой OO' . Воспользовавшись подобием треугольников $OXO' \sim xXx'$ легко получить:

$$B - (x'_{pr} - x_{pr}) = \frac{Bf}{z}. \quad (1)$$

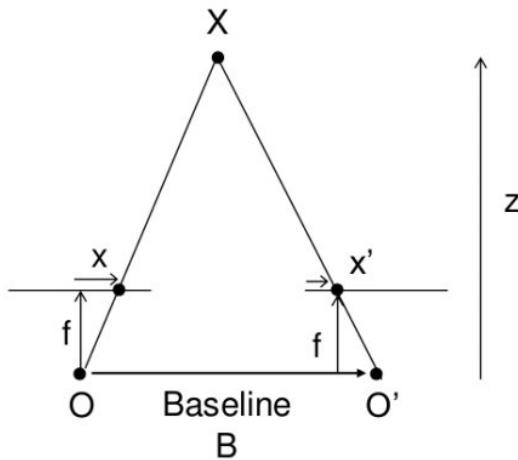


Рис. 3. Поясняющий рисунок

Если x_{img} , x'_{img} – евклидовы координаты точки X в системе координат изображений с левой и правой камеры соответственно, то, очевидно, что они следующим образом связаны с координатами проекций на ось OO' :

$$\begin{aligned}x_{img} &= x_{pr}, \\x'_{img} &= - (B - x'_{pr}).\end{aligned}$$

Подставляя последние выражения в (1), получим:

$$(x_{img} - x'_{img}) = \frac{Bf}{z}. \quad (2)$$

*Диспаратностью, смещением или диспаритетом (*disparity*)* называют величину d , которая задается выражением:

$$d = x_{img} - x'_{img}. \quad (3)$$

Исходя из этого, выражение (2) можно переписать следующим образом:

$$d = \frac{Bf}{z}. \quad (4)$$

Очевидно, что точки, расположенные дальше от плоскости стереопары, имеют меньшее значение диспаратности, и наоборот. Также с увеличением расстояния B между камерами, значение диспаратности увеличивается. Из-за обратной зависимости глубины z и смещения d , разрешающая способность систем стерео зрения, которые работают на основе данного метода, лучше на близких расстояниях, и хуже на далеких.

Если найти точечные соответствия между изображениями, полученными с разных камер, то с помощью формулы (3) без труда можно найти диспаратность d для каждой точки на изображении.

После ректификации изображений, выполняют поиск соответствующих пар точек. Существуют различные способы поиска точечных соответствий [12]. Самый простой способ состоит в следующем. Для каждого пикселя левой картинки с координатами (x_0, y_0) выполняется поиск пикселя на правой картинке. При этом предполагается, что пиксель на правой картинке должен иметь координаты $(x_0 - d, y_0)$, где d – диспаратность. Поиск соответствующего пикселя выполняется путем вычисления максимума функции отклика, в качестве которой может выступать, например, корреляция окрестностей пикселей. В результате находится диспаратность и строится карта смещений.

*Картой диспаратности или картой смещений (*disparity map*)* называется изображение, полученное в результате вычисления диспаратности для каждой точки на исходных изображениях. Значение яркости в каждой точке такого изображения будет равно вычисленной диспаратности в этой точке.

На карте диспаратности могут существовать точки, в которых не удалось найти диспаратность, такие точки называются *точками разрыва диспаритета*. Области, в которых диспаритет терпит разрыв, обычно заполняют. Например, находят ближайшие к точке разрыва «значащие» пиксели слева и справа вдоль строки и присваивают точке разрыва наименьший из двух диспаритетов, т.е. значение диспаритета в области фона. После заполнения пустот карту диспаритета можно сгладить, например, медианным фильтром

Карта глубины. *Карта глубины (*depth map*)* – это изображение, на котором в каждом пикселе, вместо цвета, храниться расстояние до камеры. Карта глубины может быть получена

с помощью специальной камеры глубины (например, сенсор Kinect является своего рода такой камерой), а также может быть построена по стереопаре изображений [6].

По известной карте диспаратности с помощью выражения (4), можно найти расстояние z до точек на сцене, фотографии которой рассматриваются. Однако для этого требуется знать расстояние B между камерами и фокусное расстояние камер f , которое можно найти при помощи калибровки. Причем, поскольку фокусное расстояние и расстояние между камерами остаются постоянными для всех точек на фотографии, то диспаратность можно использовать как относительную глубину точек.

Один из возможных алгоритмов вычисления расстояния до наблюдаемого объекта можно записать следующим образом [8]:

- 1) Производится сбор изображений, на которых изображен шаблон шахматной доски в различных положениях, с левой и правой камер стереопары. Для корректной работы реализации алгоритма необходимо, чтобы шаблон шахматной доски полностью попадал в область видимости камер стереопары и занимал как можно большую площадь кадра.
- 2) Производится поиск узлов шаблона шахматной доски на изображениях полученных на шаге 1. Поиск узлов шаблона можно осуществить при помощи функции библиотеки OpenCV `cv::findChessboardCorners()`. Координаты узлов шаблона сохраняются для дальнейшего использования.
- 3) Производится калибровка каждой из стереопар по отдельности. Калибровка камер стереопары осуществляется при помощи функции `cv::calibrateCamera()`.
- 4) Производится калибровка всей стереопары. Данные, полученные на предыдущем шаге подаются на вход функции `cv::stereoCalibrate()`, которая считает внутренние параметры стереопары. После процедуры стерео калибровки мы получаем фокусное расстояние камер, координаты принципиальных точек камер и величину стереобазы.
- 5) Производится ректификация изображений, полученных на шаге 1. Ректификация заключается в выравнивании деформированных изображений, таким образом, чтобы эпиполярные линии изображений с левой камеры совпадали с эпиполярными линиями изображений с правой камеры. Данную процедуру выполняет функция `cv::stereoRectify()`.
- 6) Производится устранение дисторсии на изображениях шахматной доски, полученных на шаге 5. Устранение дисторсии производится при помощи функций `cv::initUndistortRectifyMap()` и `cv::remap()`. После выполнения данного шага, все прямые линии наблюдаемых объектов на сцене становятся прямыми линиями на изображении.
- 7) Для каждой пары соответствующих изображений с левой и правой камер производится построение карты диспаратностей по изображениям, полученным на шестом шаге. Каждый пиксель карты диспаратностей, в действительности, содержит в себе информацию о том, сколько пикселей по оси OX находится между соответствующими пикселями изображений объекта с левой и правой камер. Построение карты диспаратностей может быть осуществлено на основе алгоритма SGM [4] при помощи класса `cv::StereoSGBM`.
- 8) Производится вычисление расстояний до наблюдаемых на сцене объектов (см. формулу (4)). Вычисление расстояния до объекта напрямую связано с информацией, которую несет в себе карта диспаратностей, построенная на шаге 7, и информацией полученной в результате калибровки на шаге 6.

3.2. Алгоритмы построения карты диспаратности

В данном разделе содержатся базовые сведения об алгоритмах построения карты диспаратности.

Для построения карты диспаратности необходимо найти точечные соответствия между изображениями с разных камер. Только лишь эпиполярного ограничения (ректификации) и предположения о соответствии цветов соответствующих точек недостаточно, так как задача остается некорректно поставленной, потому что:

- Решение может не существовать (перекрытия);
- Решение может быть не уникально (однородные области);
- Небольшие изменения входных данных существенно влияют на результат (шум).

По этой причине существуют различные методы, которые используют дополнительные ограничения и предположения.

Различают два достаточно широких класса алгоритмов для решения проблемы соответствия: *локальные и глобальные алгоритмы*.

Вычислительная сложность локальных алгоритмов невысока, однако результат работы может содержать ошибки, которые, в свою очередь, могут оказаться неприемлемыми для некоторых приложений. Алгоритмы называются локальными, так как диспаритет в каждой точке зависит только от ее локальной окрестности.

Глобальные алгоритмы, как правило, показывают хорошие результаты, но из-за высокой вычислительной сложности использовать их в реальном времени практически невозможно. Алгоритмы называются глобальными, так как диспаритет в каждой точке вычисляется при помощи некоторой глобальной процедуры оптимизации, то есть зависит не только от локальной окрестности.

В сумме эти два класса алгоритмов объединяют большинство из существующих решений. Большинство алгоритмов выполняют, частично или полностью, следующие шаги [13]:

- 1) Вычисление меры соответствий.
- 2) Суммирование мер соответствий.
- 3) Вычисление диспаратностей на основе вычисленных мер.
- 4) Улучшение карты диспаратности.

Реальная последовательность шагов зависит от конкретного алгоритма. Под *мерой соответствий* понимают величину, которая характеризует схожесть между двумя областями на изображениях с разных камер. Мера соответствий используется для того, чтобы уменьшить количество неправильно найденных соответствий. Меру соответствий также называют *стоимостью соответствий*. Существуют различные подходы вычисления меры, многие из которых основаны на значениях интенсивности пикселей. Например, такие способы, как нахождение абсолютной разности интенсивностей (Absolute Intensity Differences, AD) и квадрата разности интенсивностей (Squared Intensity Differences, SD).

Локальные алгоритмы. Этот класс алгоритмов подсчитывает диспаратность каждого пикселя в отдельности, используя окна фиксированного или адаптируемого размера для корреляции. Выбор формы окна и его размеров является непростой задачей.

Для локальных алгоритмов мера соответствия (т.е. первый шаг работы алгоритма) определяется как схожесть между двумя областями, одна из которых находится в базовом изображении, а другая – в парном. Форма этих областей зависит от конкретного алгоритма. Стандартные подходы, появившиеся на заре исследований проблемы соответствия, используют прямоугольное окно фиксированного размера. Размеры этого окна определяются опытным путем. Объем вычислений, в этом случае, сильно сокращается по сравнению с современными методами, но существует ряд проблем, избежать которых, используя фиксированный размер окна, практически невозможно. Маленькие окна приводят к недостаточной информации о текстуре области, в связи с чем возникает большая чувствительность к шумам. При использовании большого окна возникает эффект «раздувания» объектов переднего плана (*foreground fattening*) при однотонных фонах. Существуют алгоритмы, решающие эту проблему с помощью адаптивного размера окна. Развитие локальных методов на сегодняшний день идет в различных направлениях. Среди которых можно выделить: направление адаптивных окон; алгоритмы, которые используют сегментацию изображения и исходят из предположения, что области разрыва диспаритета совпадают с краями (границами сегментов) на изображении; алгоритмы, которые используют вероятностные модели, например алгоритм ELAS [26].

После того, как для каждого пикселя подсчитана мера соответствия, локальный алгоритм, обычно, переходит к шагу суммирования мер. Стандартный подход состоит в суммировании или усреднении мер соответствия в некоторой области. Усреднение может производиться с помощью свертки с каким-нибудь ядром, чаще всего Гауссовым.

Финальным шагом является вычисление диспаратности на основе просуммированных или усредненных мер соответствий. Диспаратность для данного пикселя базового изображения – это значение, на котором достигается минимальное значение меры (или максимальное, в зависимости от выбранной меры). Проблема, возникающая при таком подходе, состоит в том, что каждой точке выбирается уникальное соответствие на парном изображении, но на самом же деле, одной точке может соответствовать сразу несколько. К сожалению, эта проблема возникает не только у локальных алгоритмов, но и у подавляющего большинства методов, целью которых является построение карты диспаратности.

Глобальные алгоритмы. В отличие от локальных алгоритмов, где нахождение диспаратности происходит для каждого пикселя отдельно, целью глобального подхода является поиск наилучшей карты диспаратности для всего изображения сразу. Глобальные алгоритмы практически всю работу выполняют на шаге вычисления диспаратностей, часто пропуская шаг суммирования мер соответствия. Чаще всего глобальные алгоритмы решают проблему соответствия путем минимизации некоторого функционала энергии. Глобальные алгоритмы – наилучшие по качеству методы на сегодняшний день. Количество ошибок, допускаемых ими, относительно невелико. Тем не менее, большинство глобальных алгоритмов непригодны для работы в реальном времени [13].

Как правило, глобальные алгоритмы формулируются в терминах разметки графа и минимизации энергии. Рассматривается граф (решетка, см. Рис. 4), где метки – значения диспаритета, узлы – пиксели. Осуществляется поиск разметки D , минимизирующей функцию энергии $E(D)$:

$$E(D) = E_{data}(D) + E_{smooth}(D),$$

где $E_{data}(D)$ – унарный потенциал, который характеризует степень согласованности конфигурации с парой входных изображений и вычисляется с помощью мер соответствия; $E_{smooth}(D)$

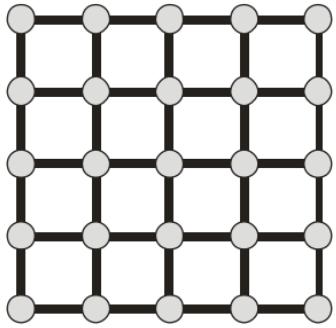


Рис. 4. Схема графа решетки.

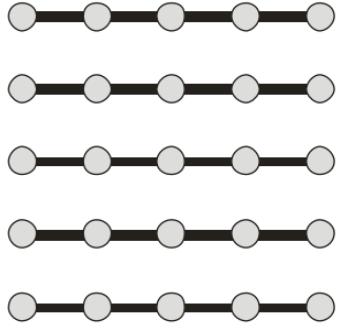


Рис. 5. Схема графа в [14].

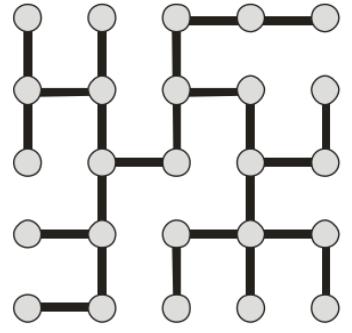


Рис. 6. Схема графа в [15].

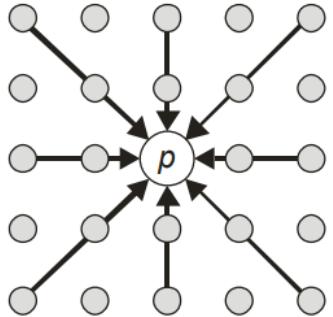


Рис. 7. Схема графа в SGM [4].

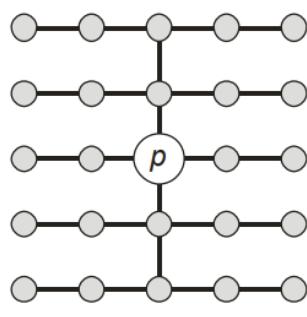


Рис. 8. Схема графа в [17].

– парный потенциал, который явным образом реализует предположение о гладкости, т.е. накапливает величину штрафа, налагаемого на данную конфигурацию при нарушении гладкости диспаратностей. Обычно при вычислении парного потенциала учитываются соседние пиксели. В результате минимизации функционала находится карта диспаратности D .

Развитие глобальных алгоритмов заключается в уходе от графов общего вида и переходу к деревьям. Отсутствие циклов в деревьях позволяет использовать метод динамического программирования и повысить скорость работы алгоритма.

Существует также разделение глобальных алгоритмов по способу минимизации энергии. Чаще всего используется либо динамическое программирование, либо нахождение минимального разреза графов (graph cuts). Алгоритмы разреза графов также получили название двумерных. Их производительность ниже, но результаты имеют более высокую точность. В ходе использования одномерных алгоритмов строки изображений обрабатываются независимо друг от друга. Вследствие этого их скорость выше, но возникает эффект «гребенки» – рассогласование значений диспаратности на уровне отдельных строк.

Так, например, в алгоритме Scanline Optimization [14] удаляются все вертикальные ребра (Рис. 5). Алгоритм работает, однако рассогласованность строк между собой (*horizontal streaking*) вносит в результат заметные искажения. Алгоритм, предложенный в работе [15], учитывает связь различных строк и основан на построении минимального покрывающего дерева (Рис. 6). Результат улучшается, однако некоторая рассогласованность всё-таки остается.

Другой популярный алгоритм – SGM, описан в работе [4]. Подход не совсем глобальный, но и не локальный. Алгоритм SGM дает практически такие же результаты, как и глобальные алгоритмы, однако скорость его работы гораздо выше. В каждом пикселе строится свое дерево (Рис. 7). Оптимизация производится вдоль лучей, исходящих из пикселя. Рассогласованности нет, но могут присутствовать «изолированные» пиксели, резко отличающиеся от соседних с

ними пикселей. Такие пиксели могут быть удалены различными способами на этапе постобработки.

Алгоритм Simple Tree, предложенный в [17], лишен недостатка SGM. В нем предлагается в каждом пикселе строить два дерева (Рис. 8), совместно покрывающих все изображение.

Сравнительный анализ эффективности алгоритмов показывает, что метод SGM показывает наилучшие результаты с точки зрения быстродействия, качественной и количественной оценки полученных карт диспаритности [18].

StereoBM. В библиотеке OpenCV [3] реализованы алгоритмы StereoSGBM и StereoBM.

Алгоритм StereoBM производит некоторую предобработку изображений. Построочно анализирует базовое и парное изображение. На каждой строке алгоритм берет точку или их набор, и пытается найти похожий набор точек на этой же строкке на правом изображении. После этого он считает, насколько эти наборы точек друг относительно друга смешены. На последнем шаге проводится некоторая пост обработка и в результате получается карта диспаритета.

На вход функции `cv::StereoBM` подаются базовое и парное изображения и два параметра – `ndisparities` и `SADWindowSize`. Первый параметр `ndisparities` указывает, насколько "далеко" алгоритм должен просмотреть строчку парного изображения в поисках нужного ему фрагмента. Если указать `ndisparities = 16`, то это будет означать, что алгоритм будет "заглядывать" левее на 16 пикселей. Параметр `SADWindowSize` – это размер окна, который алгоритм будет использовать для вычисления меры соответствия.

StereoBM – алгоритм, работающий в реальном времени (может обрабатывать изображение 1920×1080 в миллисекундах). StereoBM работает быстрее, но является менее точным по сравнению с StereoSGBM. Как показывают эксперименты, некоторая предобработка исходных изображений улучшает качество работы алгоритма и приближает его к качеству алгоритма StereoSGBM (см. [12]).

StereoSGBM. StereoSGBM является более точным алгоритмом, но работает медленней (несколько секунд, для изображения 1920×1080).

Реализация алгоритма StereoSGBM в библиотеке OpenCV несколько отличается от оригинального алгоритма SGM, описанного в [4], а именно:

- По умолчанию алгоритм рассматривает 5 направлений суммирования стоимостей соответствий вместо 8. Для рассмотрения 8 направлений необходимо установить параметр `mode=StereoSGBM::MODE_HH`.
- Алгоритм сравнивает блоки, а не пиксели. Для сравнения пикселей необходимо установить параметр `blockSize=1`.
- Вместо предлагаемой в работе меры соответствия использована более простая мера Birchfield-Tomasi [16], которая устойчива к погрешности сэмплирования (когда непрерывный сигнал может быть сэмплирован по-разному, из-за чего цвета соответствующих пикселей на разных изображениях могут не совпасть).
- Используется некоторая пре- и пост- обработка, как и в реализации алгоритма StereoBM.

Параметры алгоритма StereoSGBM:

- `minDisparity` – минимально возможное значение диспаритета. Обычно значение равно нулю, однако иногда процедура ректификации может смещать изображения, поэтому этот параметр можно соответствующим образом отрегулировать.

- `numDisparities` – модуль разности максимального и минимального значения диспаритета. В текущей реализации значение данного параметра должно делиться на 16.
- `blockSize` – размер сравниваемых блоков. Нечетное число больше 1. Обычно размер выбирается в промежутке 3 … 11.
- `P1` – первый параметр, контролирующий сглаживание диспаритета.
- `P2` – второй параметр, контролирующий сглаживание диспаритета. Чем больше значение, тем больше сглаживание. `P1` – это величина штрафа в случае, если диспаритет изменяется на ± 1 между соседними пикселями в окне заданного размера. `P2` – это величина штрафа в случае, если диспаритет изменяется более чем на 1 между соседними пикселями в окне заданного размера. Алгоритм требует выполнения условия $P2 > P1$. Относительно хороший результат дают следующие значения:

$$P1 = 8 \cdot \text{number_of_image_channels} \cdot \text{SADWindowSize}^2,$$

$$P2 = 32 \cdot \text{number_of_image_channels} \cdot \text{SADWindowSize}^2.$$

- `disp12MaxDiff` – максимально допустимая разность (в единицах целочисленного пикселя) в диспаритете для левого-правого изображений. Отрицательное значение отключает такую проверку.
- `preFilterCap` – значение порога усечения для предварительно фильтрованных пикселей изображения. Алгоритм сначала вычисляет производную по x на каждом пикселе и фиксирует это значение на интервале $[-\text{preFilterCap}, \text{preFilterCap}]$. Значения результата передаются в функцию расчета меры соответствий Birchfield-Tomasi.
- `uniquenessRatio` – запас в процентах, по который лучшее (минимальное) значение меры соответствий должно «выиграть» у второго лучшего значения, чтобы считать найденное совпадение правильным. Обычно значение в пределах диапазона 5-15 достаточно хорошее.
- `speckleWindowSize` – Максимальный размер областей гладких регионов диспаритета для поиска и удаления шумовых пятен. Значение 0 отключает фильтрацию пятен. Обычно рабастные значение лежат в пределах диапазона 50-200.
- `speckleRange` – максимальное различие в каждой области. При выполнении фильтрации пятен параметр имеет положительное значение, которое будет неявно умножено на 16. Обычно устанавливают значения 1 или 2.
- `mode` – режим работы. Возможные значения: `StereoSGBM::MODE_SGBM`, `StereoSGBM::MODE_HH`, `StereoSGBM::MODE_SGBM_3WAY`, `StereoSGBM::MODE_HH4`. Значение `StereoSGBM::MODE_HH`, запускает полномасштабный двухпроходный алгоритм динамического программирования. Алгоритм будет потреблять $O(W \cdot H \cdot \text{numDisparities})$, которые являются большими для изображений 640x480 и огромны для HD-изображений. По умолчанию установлено значение `False`.

3.3. Алгоритм нахождения свободных для движения регионов

В этом разделе рассмотрим алгоритм, предложенный в работе [1] и использующий *u-v*-диспаритет.

Детекция регионов, доступных для движения, является важной задачей для навигации мобильных роботов и беспилотных транспортных средств. Препятствия, которые делают регион недоступным для движения, можно разделить на два основных класса: 1) геометрические препятствия; 2) не геометрические препятствия (связанны, например, со свойством поверхности: вязкость, малая плотность, неровность). Данная задача может решаться на основе данных полученных с разных источников: лазерного сканера (LIDAR), стерео камер, 2D датчиков глубины и других.

Можно выделить два основных подхода к детекции регионов, доступных для движения, на основе данных, полученных со стерео камер:

- 1) Проведение 3D реконструкции по облаку точек, полученному на основе карты диспаритета, и с последующее использование детекции углов, поиска плоскостей и других характеристик для нахождения регионов, доступных для движения, и регионов с препятствиями [20, 21, 22]. Преимуществом такого подхода является хорошая точность. Недостатком – высокая сложность вычислений и, как следствие, невозможность работы в приложениях реального времени.
- 2) Использование *u-v*-диспаритета [23, 24, 25]. Преимуществом такого подхода является скорость вычислений.

v-диспаритет. Пусть v представляет собой в координату пикселя в (u, v) системе координат изображения карты диспаритета, d – значение диспаритета. Карта v -диспаритета строится таким образом, что каждый её пиксель с координатами (d, v) имеет значение интенсивности равное количеству пикселей с параметрами d, v на карте диспаритета. Другими словами, v -диспаритет получается на основе карты диспаритета, подсчетом количества пикселей в строке v , имеющих одно и то же значение диспаритета d .

Так, например, тот факт, что для вертикального препятствия, удаленного на некоторое расстояние, диспаритет остается постоянным, на карте v -диспаритета будет отражен в виде вертикальной линии, значение интенсивности которой будет характеризовать ширину препятствия. Ровная поверхность, такая как дорога, на карте v -диспаритета также будет отражена в виде линии, только уже не вертикальной, а наклоненной. Такая линия называется *линией корреляции земли* и она соответствует региону, доступному для движения.

Таким образом, чтобы найти регион, доступный для движения, нужно найти линию корреляции земли и пиксели на карте диспаритета, которые ей соответствуют. Это не сложно сделать с помощью преобразования Хафа, однако присутствие на карте v -диспаритета линий соответствующих препятствиям значительно ухудшает результат. В работе [1] предлагают использовать *u*-диспаритет для предварительного поиска препятствий, чтобы они не учитывались при построении v -диспаритета и, как следствие, не мешали определению линии корреляции земли.

u-диспаритет. Аналогично v -диспаритету можно определить *u*-диспаритет. Карта *u*-диспаритета строится таким образом, что каждый её пиксель с координатами (u, d) имеет значение интенсивности равное количеству пикселей с параметрами u, d на карте диспаритета. Другими словами, *u*-диспаритет получается на основе карты диспаритета, подсчетом количества пикселей в столбце u , имеющих одно и то же значение диспаритета d .

Препятствиям на карте u -диспаритета будут соответствовать горизонтальные линии, интенсивность которых будет характеризовать высоту препятствий. Ровные поверхности, такие как дорога, в данном случае будут иметь малую интенсивность на карте u -диспаритета.

С использованием u -диспаритета могут быть найдены линии соответствующие препятствиям. Пиксели на карте диспаритета, соответствующие этим линиям, будут пикселями препятствий и могут не рассматриваться при построении карты v -диспаритета.

Алгоритм. Изложим кратко предложенный в работе [1] алгоритм:

- 1) Построить карту u -диспаритета для исходной карты диспаратности.
- 2) Построить бинарное изображение u -диспаритета. Для каждого пикселя (u, d) при условии, что значение интенсивности u -диспаритета больше некоторого порогового значения, найти пиксели в u столбце на исходной карте диспаратности, для которых диспаритет равен d . Найденные пиксели пометить как пиксели препятствия (присвоить значение интенсивности 1), остальные – как потенциальные пиксели региона свободного для движения (присвоить значение интенсивности 0).
- 3) Применить морфологическую операцию замыкания для связывания областей с небольшими разрывами. После чего удалить небольшие изолированные участки, т.к. они не пригодны для движения.
- 4) На исходной карте диспаратности найти пиксели, которые соответствуют пикселям препятствия (значение интенсивности 1). Найденные пиксели образуют новую карту диспаратности – *карту препятствий*. Остальные пиксели образуют *карту не-препятствий*.
- 5) Построить карту v -диспаритета для карты диспаратности не-препятствий.
- 6) Бинаризовать изображение v -диспаритета с помощью некоторого порога. С помощью преобразования Хафа на бинаризованном изображении найти линию корреляции земли. Для всех пикселей, лежащих на найденной линии, найти соответствия на карте не-препятствий. Найденные соответствия пометить как область, доступную для движения (присвоить значение интенсивности 1), всем остальным присвоить значение интенсивности 0. Таким образом получается *карта регионов доступных для движения*.
- 7) На найденной карте регионов, доступных для движения, удалить небольшие изолированные участки, т.к. они не пригодны для движения.

3.4. KITTI Benchmark Suite

В этом разделе рассмотрим бенчмарк для оценки производительности методов нахождения дорожного полотна.

The KITTI Vision Benchmark Suite – данные и бенчмарк для разработки и оценки производительности методов и алгоритмов для автономных транспортных средств. Данные получены с автомобиля оснащенного стерео камерами, мобильным лазерным сканером и GPS. Также предлагаются бенчмарки для сравнения производительности различных алгоритмов на предоставляемых данных. Среди них есть бенчмарк Road/Lane Detection Evaluation 2013 для оценки производительности алгоритмов определения дороги и полосы движения [2].

Таблица 1. Структура KITTI-ROAD Dataset.

аббревиатура	тренировочные	тестовые	описание
UU	98	100	городские дороги без дорож.разметки
UM	95	96	городские двухполосные с д.разметкой
UMM	96	94	городские многополосные с д.разметкой
URBAN	289	290	все три вместе

KITTI-ROAD Dataset. Содержит два набора изображений с левой и правой камер: один тренировочный, с эталонными результатами и калибровочными данными камер; другой тестовый, в котором эталонные результаты отсутствуют. Тестовый и тренировочный наборы содержат изображения трёх классов дорог (см. Табл. 1). Кроме того, предоставляются данные мобильного лазерного сканера и GPS, которые могут быть использованы для получения результатов.

KITTI-ROAD Benchmark. Производительность алгоритмов предлагается оценивать с помощью "вида сверху" (BEV). В работе [2] показано, что использование "вида сверху" даёт более корректные результаты, чем использование "перспективного вида". Результаты (в нашем случае карта регионов доступных для движения) преобразовываются к "виду сверху" с помощью обратного перспективного преобразования изображения. Далее в пространстве BEV вычисляются характеристики, дающие оценку работы алгоритма (более подробно см. [2]):

$$\begin{aligned} Precision &= \frac{TP}{TP + FP}, \\ Recall &= \frac{TP}{TP + FN}, \\ F\text{-measure} &= (1 + \beta^2) \frac{Precision \cdot Recall}{\beta^2 Precision + Recall}, \\ Accuracy &= \frac{TP + TN}{TP + FP + TN + FN}, \\ F_{max} &= \max_{\tau} F\text{-measure}, \\ AveragePrecision &= \frac{1}{11} \sum_{r \in 0, 0.1, \dots, 1} \max_{\tilde{r}: \tilde{r} > r} Precision(\tilde{r}), \end{aligned}$$

где τ – порог бинаризации в случае если результатом (карты доступных для движения регионов) является не бинарное изображение, а изображение, интенсивность пикселей которого равна вероятности; TP , FP , TN , FN – количество истинно-положительных, ложно-положительных, истинно-отрицательных, ложно-отрицательных классификаций; β – коэффициент, который полагается равным 1; r – значение $Recall$, а $Precision(r)$ – соответствующее этому значению $Precision$.

Если результат классификации положительный, и истинное (эталонное) значение тоже положительное, то речь идет об истинно-положительном значении (true-positive).

Если результат классификации положительный, но истинное значение отрицательное, то речь идет о ложно-положительном значении (false-positive).

Если результат классификации отрицательный, и истинное значение тоже отрицательное, то речь идет об истинно-отрицательном значении (true-negative).

Если результат классификации отрицательный, но истинное значение положительно, то речь идет о ложно-отрицательном значении (false-negative).

F_{max} и AP (Average Precision) достаточно хорошо характеризуют точность алгоритма по нахождению дороги. Чем ближе они к единице (100%), тем точнее работает алгоритм. Бенчмарк предоставляет *development kit*, в котором реализован данный метод оценки производительности.

4. Выполнение задания

В этой части приведены описание и результаты экспериментов. Исходный код программ размещен на GitHub¹. Данные для экспериментов: KITTI-ROAD Dataset².

4.1. Формат данных

Базовые и парные изображения: цветные изображения в формате PNG, 8 бит (`uint8`), 3 канала (RGB). Имена файлов `<cat>_xxxxxx.png`, где `<cat>`: `uu`, `um` или `umm`.

Карты диспаритности: изображения в формате PNG, 16 бит (`uint16`), 1 канал. Значение 65535 является маркером для невалидных значений диспаритетов. Значения диспаритетов представлены в формате с фиксированной точкой, 4 бита отведено на дробную часть. Таким образом, чтобы получить значение диспаритета, необходимо после чтения изображения поделить результат на 16. Имена файлов соответствуют именам базовых изображений.

Карты доступных для движения регионов: изображения в формате PNG, 16 бит (`uint8`), 1 канал. Значение 255 является маркером регионов доступных для движения. Имена файлов `<cat>_<type>_xxxxxx.png`, где `<type>`: `road`.

Карты эталонных результатов: цветные изображения в формате PNG, 8 бит (`uint8`), 3 канала (RGB). Канал В содержит карту эталонных результатов (ground truth), канал R содержит допустимые области оценки (valid evaluation areas). Имена файлов соответствуют именам карт регионов доступных для движения.

4.2. Нахождение карт диспаритности

Карты диспаритета находятся для каждой пары базового и парного изображений помощью алгоритма `cv::StereoSGBM`. После качественной оценки результата, были выбраны следующие параметры алгоритма (подробнее о значениях параметров см. п. 3.2):

- `minDisparity = 0;`
- `numDisparities = 7 · 16;`
- `blockSize = 5;`
- $P1 = 8 \cdot \text{number_of_image_channels} \cdot \text{SADWindowSize}^2$;
- $P2 = 32 \cdot \text{number_of_image_channels} \cdot \text{SADWindowSize}^2$,
- где `SADWindowSize = 7`, `number_of_image_channels = 3`;
- `disp12MaxDiff = -1;`

¹<https://github.com/abramenko/traversable-region-detection>

²http://www.cvlibs.net/datasets/kitti/eval_road.php

- `preFilterCap = 1;`
- `uniquenessRatio = 5;`
- `speckleWindowSize = 200;`
- `speckleRange = 1;`
- `mode = StereoSGBM::MODE_SGBM_3WAY.`

Процесс вычисления карт диспаритета реализован в виде скрипта `img2disp.py` (см. Приложение, стр. ??). В результате работы алгоритма `cv::StereoSGBM` получается изображение типа `int16`. Полученное изображение преобразовывается в тип `uint16`, а все пиксели, значение диспаритета которых было меньше нуля, помечаются как невалидные (присваивается значение интенсивности равное 65535). Результат вычисления карты диспаритета для базового и парного изображения (см. Рис. 9, 10) показан на Рис. 11 на стр. 22. Результаты достаточно хорошие, но их можно улучшить, если провести ректификацию базового и парного изображений.

4.3. Нахождение регионов доступных для движения

Процесс нахождения доступных для движения регионов реализован в виде скрипта `find_traversable.py` (см. Приложение, стр. ??). Реализован алгоритм, описанный в п. 3.3. В реализации алгоритма используются следующие параметры:

- `u_disp_threshold = 3` – порог для бинаризации u -диспаритета;
- `v_disp_threshold = 3` – порог для бинаризации v -диспаритета;
- `morph_disk_radius = 9` – размер радиуса диска (в пикселях), который используется как структурообразующий элемент при морфологической операции замыкания.
- `small_obj_size = 500` – максимальный размер небольших изолированных объектов (в пикселях). Объекты, размер которых меньше заданного, будут удалены.
- `line_width=20` – ширина найденной линии корреляции земной поверхности.

В скрипте `find_traversable.py` нахождение линии корреляции земной поверхности реализовано следующим образом:

- 1) С помощью преобразования Хафа находятся параметры r, θ прямой, набравшей наибольшее количество голосов.
- 2) Считается, что пиксель лежит на линии корреляции земной поверхности, если его координаты удовлетворяют неравенству:

$$r - \frac{\text{line_width}}{2} \leq (d \cdot \cos \theta + v \cdot \sin \theta) \leq r + \frac{\text{line_width}}{2}$$

Необходимость такой реализации вызвана тем, что за счет различных факторов (ориентация стереокамер, погрешности возникающих при вычислении диспаритета, неровности земной поверхности), линия корреляции земли не является идеальной линией, а имеет форму клина (см. Рис. 15 на стр. 24). При использовании описанной выше реализации, считается, что пиксель лежит на линии корреляции земли, если он находится между двумя синими сплошными

линиями (см. Рис. 15), которые удалены на расстояние $\frac{\text{line_width}}{2}$ от красной пунктирной линии, найденной с помощью преобразования Хафа.

На Рис. 12 представлена бинаризованная карта u -диспаритета, рассчитанная для карты диспаритета базового изображения (см. Рис. 11). На Рис. 13, 14 представлены найденные с использованием u -диспаритета карты препятствий и не-препятствий, соответственно. На Рис. 15 представлена бинаризованная карта v -диспаритета и изображены границы линии корреляции земли. Регион доступный для движения, найденный с помощью линии корреляции земли, изображен на Рис. 16 (стр. 24).

Можно заметить, что на Рис. 16 не все плоские регионы (сравните с Рис. 18 на стр. 25) отмечены как регионы доступные для движения. Это связано с тем, что линия корреляции земли ищется в виде прямой заданной ширины, в то время как на самом деле она является клином. Эту проблему можно решить, если искать линию корреляции земли в виде клина. Также можно увеличить ширину линии корреляции, например положить $\text{line_width}=100$, тогда большинство точек клина будут попадать в область между сплошными линиями и регионом, доступным для движения, увеличится (см. Рис. 18).

4.4. Оценка результатов работы алгоритма

Оценим качество результатов работы алгоритма поиска свободных для движения регионов с помощью методики описанной в п.3.4. В качестве входных данных используется тренировочный набор данных KITTI-ROAD Dataset. Оценка результатов проводится в BEV пространстве.

В Таб. 2 на стр. 25 приведены результаты вычисления характеристик для карт диспаратности, полученных методами: StereoSGBM [4], ESGM [27], ELAS [26]. Можно видеть, что результаты отличаются незначительно, поэтому можно сделать вывод, что карты диспаритета, рассчитанные разными методами для одних и тех же данных, также отличаются незначительно. Для приложений реального времени целесообразно выбрать быстрый локальный метод ELAS, а в приложениях, где время не критично, можно использовать полу-глобальные методы StereoSGBM или ESGM.

В Таб. 3 на стр. 26 приводятся результаты работы алгоритма детекции доступных для движения регионов. Вычисления производились с использованием различных значений параметра line_width . Можно видеть, что из рассмотренных значений параметра наилучшая точность достигается при $\text{line_width} = 5$. А наибольшее значение F_{max} получается для значений line_width лежащих на отрезке $[5, 20]$

Полученные характеристики отражают точность и качество нахождения алгоритмом дорожного полотна. В целом алгоритм показывает неплохие результаты, если учитывать то, что он предназначен для обнаружения достаточно плоских участков, по которым теоретически можно проехать, а не только для обнаружения дорожного полотна.

4.5. Выводы

В ходе работы рассмотрены основные понятия стереозрения, описаны основные подходы к построению карты диспаритета. Реализовано нахождение карты диспаритета с помощью библиотеки OpenCV и алгоритма StereoSGBM. Кратко изложен алгоритм нахождения регионов, доступных для движения. Приведены результаты вычислений и оценка работы алгоритма на данных KITTI-ROAD Dataset.

Использование стерео камер позволяет в реальном времени решать проблемы навигации беспилотных транспортных средств и мобильных роботов. Однако стереозрение чувствительно к погодным и другим условиям, что приводит к необходимости использовать и данные с других

сенсоров. К таким сенсорам можно отнести 3D и 2D лазерные сканеры. Обработка данных, полученных с таких сенсоров занимает больше времени, что не всегда приемлемо с точки зрения безопасности движения. Однако совместное использование данных может помочь качественно улучшить результаты.

Список литературы

1. Zhu X. et al. *Stereo vision based traversable region detection for mobile robots using uv-disparity* // Control Conference (CCC), 2013 32nd Chinese. - IEEE, 2013. - pp. 5785-5790.
2. Fritsch J., Kuehnl T., Geiger A. *A New Performance Measure and Evaluation Benchmark for Road Detection Algorithms* // International Conference on Intelligent Transportation Systems (ITSC) – 2013.
3. Bradski, Gary. *The OpenCV Library* // Dr. Dobb's Journal: Software Tools for the Professional Programmer 25.11 – 2000 – pp. 120-123.
4. Heiko Hirschmuller. *Stereo processing by semiglobal matching and mutual information* // Pattern Analysis and Machine Intelligence, IEEE Transactions on – 2008 – 30(2) – pp. 328–341.
5. Hartley, Richard, and Andrew Zisserman. *Multiple view geometry in computer vision.* // Cambridge university press – 2003.
6. Bradski, Gary, and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* // "O'Reilly Media, Inc. 2008.
7. Основы стереовидения <https://habrahabr.ru/post/130300/>
8. Ильясов Э. С. *Вычисление расстояния до наблюдаемого объекта по изображениям со стереопарой* // Молодой ученый. – 2016. – №14. – С. 146-151.
9. Zhang Z. *A flexible new technique for camera calibration* // Pattern Analysis and Machine Intelligence, IEEE Transactions on. IEEE. – 2000. – 22(11). – pp. 1330–1334.
10. Tsai, Roger Y., and Thomas S. Huang. *Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces* // IEEE Transactions on pattern analysis and machine intelligence. – 1984. – 1. – pp. 13-27.
11. Frey, Markus. *Fully Automatic Calibration of Multiple Cameras to a Single World Coordinate System with Bundle Adjustment*
12. Parchami, Mostafa, and Gian-Luca Mariottini. *A comparative study on 3-D stereo reconstruction from endoscopic images* // Proceedings of the 7th International Conference on PErvasive Technologies Related to Assistive Environments. – ACM, 2014.
13. Scharstein, Daniel, and Richard Szeliski. *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms* // International journal of computer vision – 2002. – 47(1-3) –pp. 7-42.
14. Jung, Ho Yub, Kyoung Mu Lee, and Sang Uk Lee. *Stereo matching using scanline disparity discontinuity optimization* // International Conference on Advanced Concepts for Intelligent Vision Systems. – Springer, Berlin, Heidelberg, 2006.

15. Veksler, Olga. *Stereo correspondence by dynamic programming on a tree*// Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. – Vol. 2. – IEEE, 2005.
16. S. Birchfield, C. Tomasi. *A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling*. – PAMI, 1998.
17. M. Bleyer, M. Gelautz. *Simple but Effective Tree Structures for Dynamic Programming based Stereo Matching*// VISAPP. – 2008.
18. Пономарев, С. В. *Методика сравнения алгоритмов стереоизрения при восстановлении трехмерной модели лица человека*// Научно-технический вестник информационных технологий, механики и оптики – 2013. – 6 (88). – с. 40-45.
19. Geiger, Andreas, Martin Roser, Raquel Urtasun. *Efficient large-scale stereo matching*// Asian conference on computer vision. – Springer, Berlin, Heidelberg, 2010.
20. L. Li, R. Wang, and M. Zhang. *Study on Stereo Vision-based Cross-country Obstacle Detection Technology for Intelligent Vehicle*// in Proceedings of the Third International Conference on Natural Computation – 2007. – pp. 719-723.
21. A. Murarka, M. Sridharan, B. Kuipers. *Detecting Obstacles and Drop-offs using Stereo and Motion Cues for Safe Local Motion*// in Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems – 2008 – pp. 702-708.
22. T. Braun, H. Bitsch, K. Berns. *Visual Terrain Traversability Estimation using a Combined Slope/Elevation Model*// in KI 2008: Advances in Artificial Intelligence – LNCS 5243, 2008. – pp. 177-184.
23. R. Labayrade, D. Aubert, J.-P. Tarel. *Real Time Obstacle Detection in Stereovision on Non Flat Road Geometry Through "V-disparity"Representation*// in Proceedings of IEEE Intelligent Vehicle Symposium – 2002. – pp. 646-651.
24. A. Broggi, C. Caraffi, R. I. Fedriga, P. Grisleri, *Obstacle Detection with Stereo Vision for Off-Road Vehicle Navigation*// in IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops – 2005.
25. Z. Hu and K. Uchimura, *U-V-Disparity: An efficient algorithm for Stereovision Based Scene Analysis*// in Proceedings of IEEE Intelligent Vehicle Symposium/ – 2005. – pp. 48-54.
26. Geiger A., Roser M., Urtasun R. *Efficient Large-Scale Stereo Matching*// Asian Conference on Computer Vision (ACCV) – 2010.
27. Hirschmüller H. *Accurate and efficient stereo processing by semi-global matching and mutual information*// Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on – IEEE, 2005. –pp. 807–814



Рис. 9. Базовое изображение (uu_000093.png).



Рис. 10. Парное изображение.

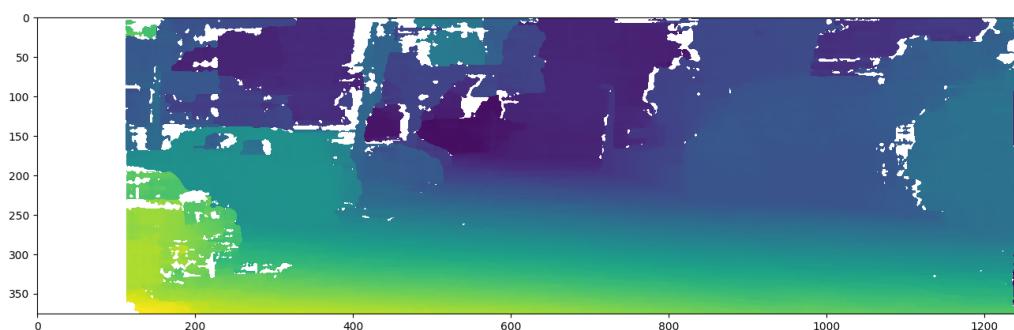


Рис. 11. Карта диспаритета в псевдоцветах для базового изображения полученная с помощью алгоритма cv::StereoSGBM.

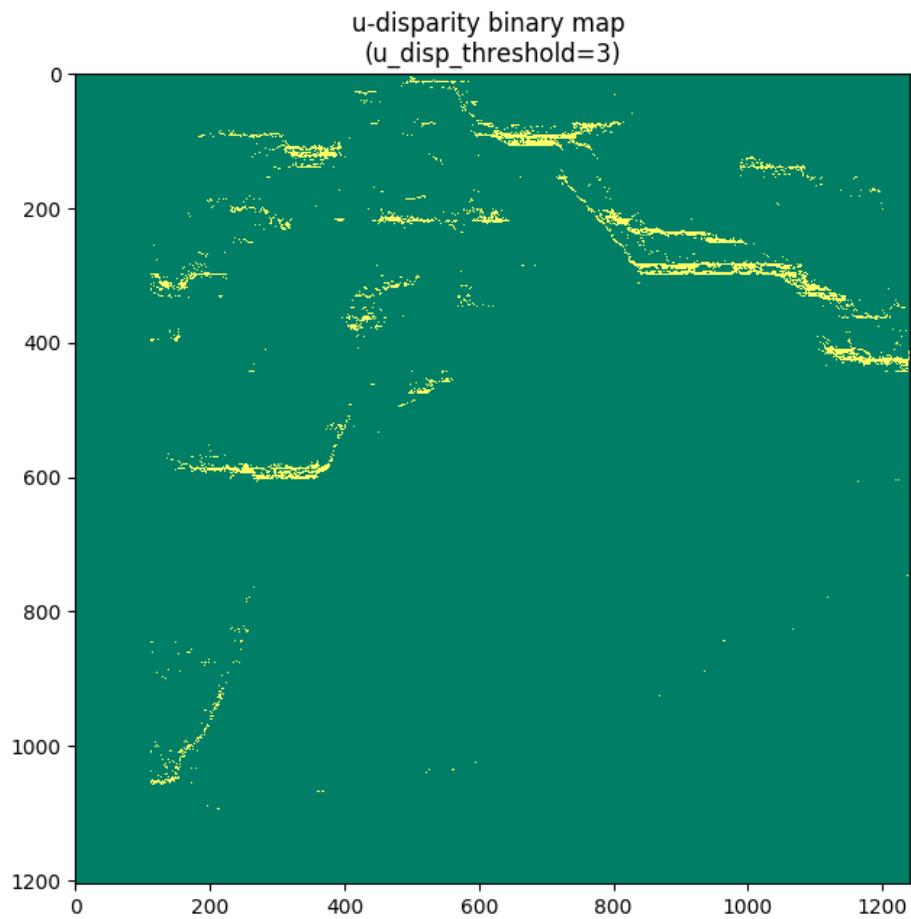


Рис. 12. Бинаризованная с помощью порога карта и-диспаритета.

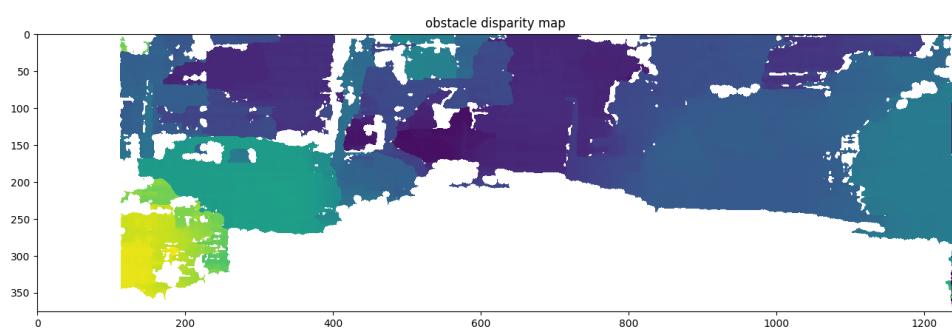


Рис. 13. Карта диспаритета препятствий.

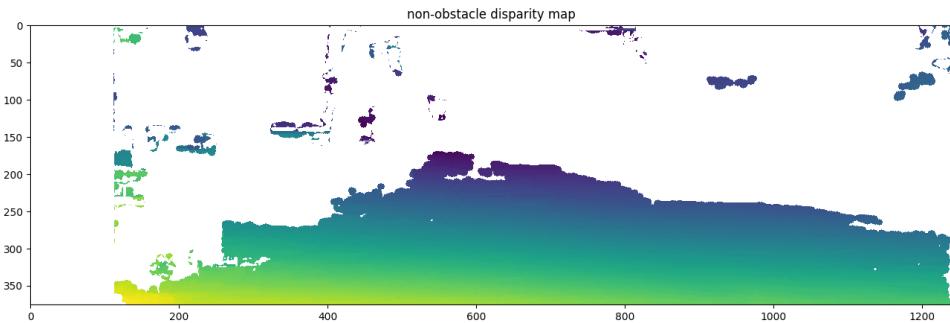


Рис. 14. Карта диспаритета не-препятствий.

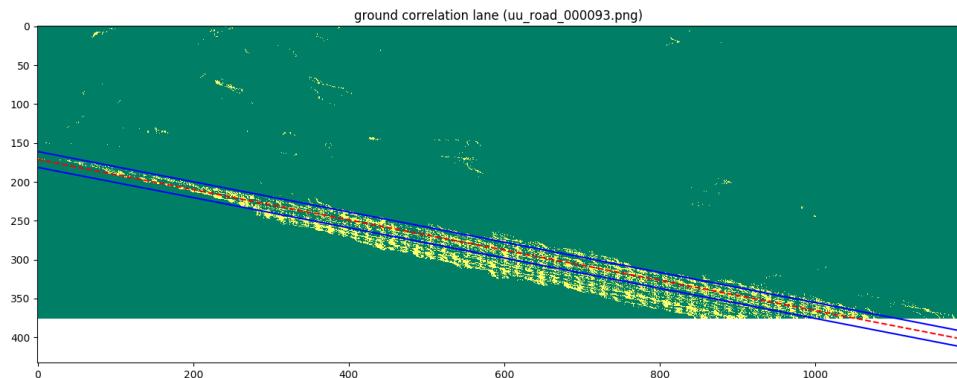


Рис. 15. Нахождение линии корреляции земли на карте v-диспаритета (line_width=20).

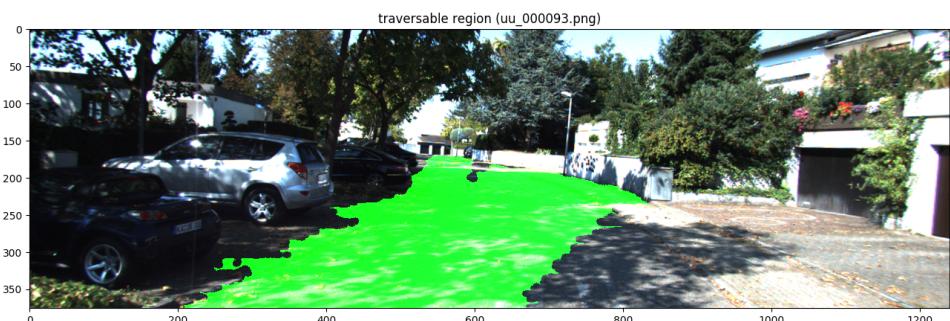


Рис. 16. Найденный регион для движения с использованием алгоритма (line_width=20).

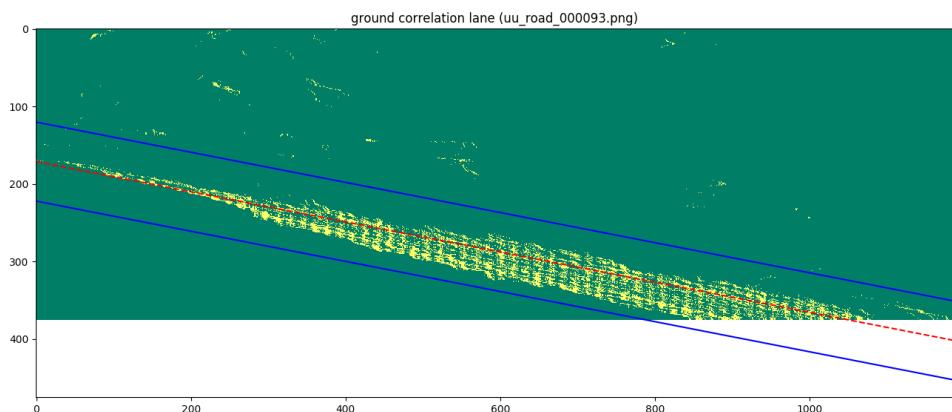


Рис. 17. Нахождение линии корреляции земли на карте v-диспаритета (line_width=100).

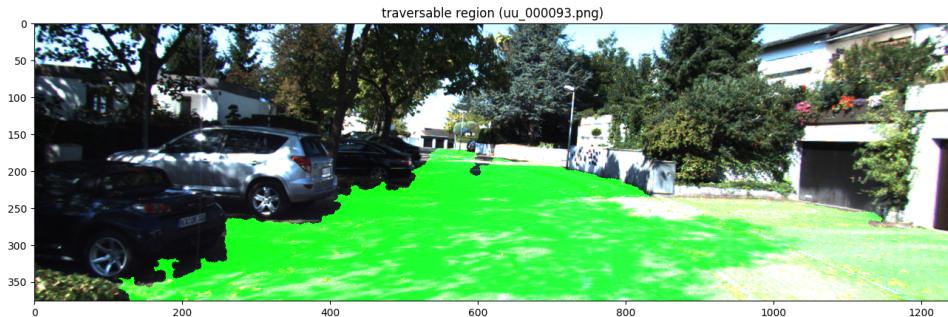


Рис. 18. Найденный регион для движения с использованием алгоритма (line_width=100).

Таблица 2. Характеристики работы алгоритма для карт диспаратности, полученных различными методами.

UM						
method	line_width	$F_{max}(\%)$	AvgPrec(%)	Prec. (%)	Recall(%)	Acc. (%)
StereoSGBM	20	64.57	48.63	50.34	90.03	68.88
ESGM	20	63.63	46.33	47.82	95.07	65.76
ELAS	20	63.56	47.06	48.62	91.76	66.86
UMM						
method	line_width	$F_{max}(\%)$	AvgPrec(%)	Prec. (%)	Recall(%)	Acc. (%)
StereoSGBM	20	78.25	67.08	68.57	91.11	73.58
ESGM	20	78.82	65.84	67.21	95.29	73.29
ELAS	20	78.45	66.38	67.81	93.06	73.33
UU						
method	line_width	$F_{max}(\%)$	AvgPrec(%)	Prec. (%)	Recall(%)	Acc. (%)
StereoSGBM	20	71.09	53.53	59.59	88.07	81.20
ESGM	20	71.96	55.57	58.50	93.47	80.89
ELAS	20	72.21	56.71	59.76	91.22	81.58

Таблица 3. Характеристики работы алгоритма для различных значений `line_width`.

<code><cat></code>	<code>line_width</code>	$F_{max}(\%)$	$AvgPrec(\%)$	$Prec.(\%)$	$Recall(\%)$	$Acc.(\%)$
UM	1	62.85	57.65	79.45	52.00	80.64
UM	5	72.61	59.97	66.30	80.26	80.93
UM	10	69.40	52.83	57.57	87.33	75.74
UM	15	66.59	49.14	53.06	89.39	71.75
UM	20	64.57	48.63	50.34	90.03	68.88
UM	30	62.39	46.16	47.63	90.42	65.67
UM	40	61.35	45.05	46.41	90.50	64.08
UM	50	60.83	44.51	45.81	90.51	63.29
UM	60	60.44	44.11	45.37	90.51	62.68
UM	70	60.17	43.83	45.07	90.51	62.26
UM	80	59.99	43.65	44.86	90.51	61.97
UM	90	59.86	43.51	44.71	90.51	61.76
UM	100	59.79	43.44	44.64	90.51	61.65
UMM	1	68.57	68.20	52.17	100.00	52.17
UMM	5	77.47	71.67	78.98	76.01	76.93
UMM	10	78.93	69.75	73.66	85.01	76.32
UMM	15	78.94	67.35	70.72	89.32	75.14
UMM	20	78.25	67.08	68.57	91.11	73.58
UMM	30	77.18	64.86	66.13	92.66	71.41
UMM	40	76.50	63.78	64.94	93.08	70.17
UMM	50	76.04	63.12	64.22	93.20	69.36
UMM	60	75.89	62.91	63.99	93.23	69.10
UMM	70	75.78	62.77	63.83	93.24	68.91
UMM	80	75.72	62.69	63.74	93.25	68.80
UMM	90	75.67	62.62	63.67	93.25	68.72
UMM	100	75.64	62.59	63.63	93.25	68.67
UU	1	56.79	51.35	81.48	43.58	82.60
UU	5	71.35	59.09	71.41	71.29	84.98
UU	10	72.82	58.06	65.13	82.58	83.82
UU	15	71.79	55.11	61.52	86.18	82.23
UU	20	71.09	53.53	59.59	88.07	81.20
UU	30	69.99	51.80	57.48	89.46	79.87
UU	40	69.47	53.77	56.52	90.10	79.22
UU	50	69.15	53.29	55.99	90.38	78.83
UU	60	68.86	52.90	55.56	90.52	78.52
UU	70	68.71	52.70	55.34	90.59	78.35
UU	80	68.61	52.57	55.20	90.64	78.24
UU	90	68.56	52.49	55.12	90.67	78.17
UU	100	68.52	52.44	55.06	90.67	78.14

Приложение. Исходный код программ

Исходный код скриптов на Python 3.6 доступен по адресу:
<https://github.com/abramenko/traversable-region-detection>

Структура расположения файлов и папок представлена на Рис. 19.

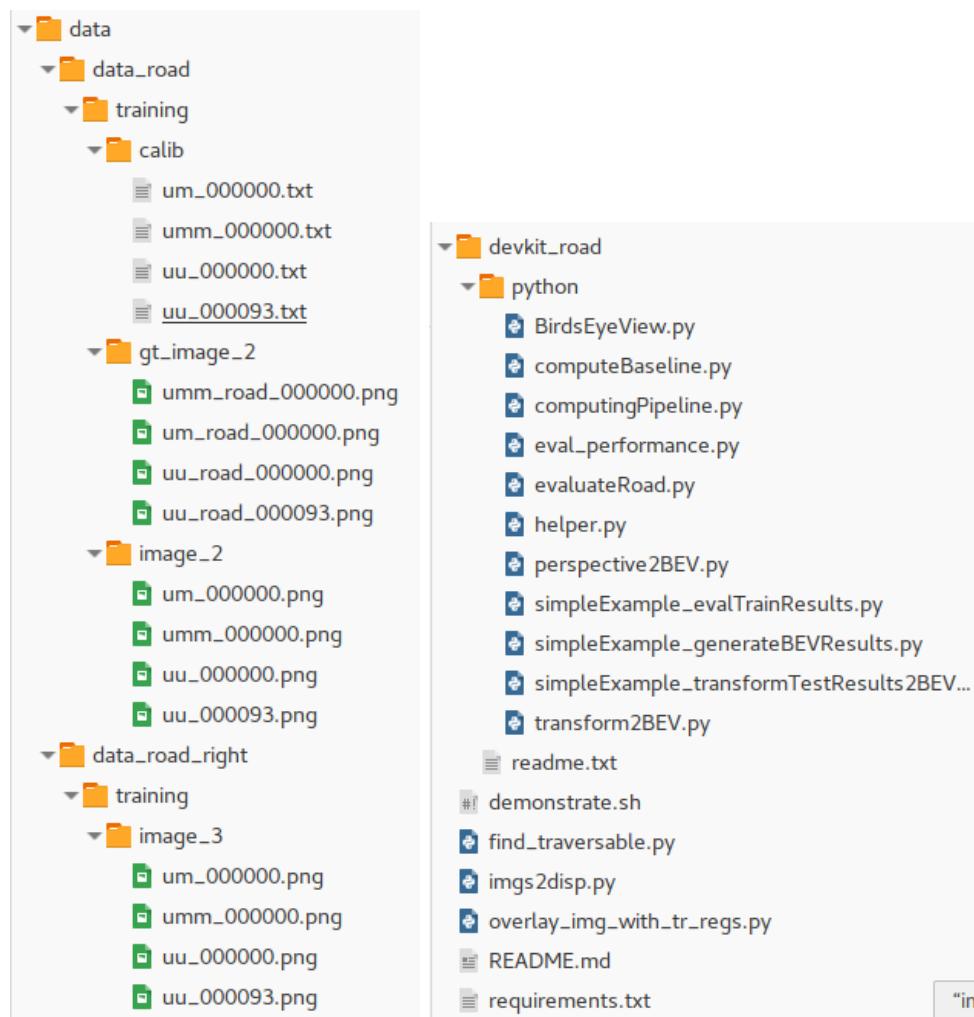


Рис. 19. Структура расположения файлов и папок.