Санкт-Петербургский Государственный Университет
Математическое обеспечение и администрирование информационных систем
Системное программирование

Терехов Михаил Андреевич

Визуальная одометрия для широкоугольных камер: прямой подход

Курсовая работа

Научный руководитель:
Пименов А. А.

Консультант:
Корчёмкин Д. А.

Санкт-Петербург
2018

Оглавление

1. Введение

Компьютерное зрение на раннем этапе своего развития оптимистично рассматривалось как набор легких задач, которые необходимо решить, чтобы приступить к более продвинутым проблемам в области искусственного интеллекта. Известная история гласит, что в 60-е годы один профессор из MIT предложил своему студенту задание на лето: тот должен был "подключить камеру к компьютеру и заставить компьютер описывать, что он увидел"[1].

К сожалению, студент не справился. Задача "описания увиденного" включает в себя восстановление геометрии пространства по набору снимков. В случае с неорганизованными коллекциями фотографий задачу принято называть *Structure from Motion (SfM)*, и в этой области исследователи добились небывалых результатов: плотные трехмерные модели восстанавливаются по фотографиям, загруженным из интернета, без какой-либо информации о камерах [2, 3].

Однако развитие вычислительных мощностей позволило пойти еще дальше, и стали появляться работы с подобной обработкой видео в реальном времени. Поток кадров обладает своей спецификой в сравнении с неорганизованной коллекцией снимков: всегда одна и та же камера, гладкое движение и, как следствие, возможность достаточно точного предсказания положения камеры в пределах нескольких следующих кадров, но появились ограничения жесткого реального времени. Задача отслеживания положения камеры и восстановления окрестностей в режиме online в английской литературе носит название *visual odometry (VO)*. Применения включают дополненную реальность [4], навигацию беспилотных летательных аппаратов [5] и наземных транспортных средств [6]. На последние нацелена и данная работа.

В рамках системы помощи водителю *ADAS (Advanced Driver Assistance*

[1] В оригинале "spend the summer linking a camera to a computer and getting the computer to describe what it saw". История взята из книги [1].

System) многие современные автомобили оснащаются набором датчиков, позволяющих анализировать ситуацию на дороге и состояние водителя. Этот набор включает ультразвуковые датчики для помощи при парковке, радары и лидары (LIDAR, Light Identification and Ranging) для обнаружения препятствий на дороге и, наконец, камеры. Использование информации, полученнной с этих камер, для решения задачи VO, позволит определять расстояния до объектов альтернативными способами и избавит нас от необходимости установки остальных, более дорогостоящих датчиков.

2. Задачи

Целью данной работы является создание системы VO для работы с широкоугольной камерой. При создании подобной системы могут использоваться различные методы, и авторами предлагается сконцентрировать внимание на *прямом* подходе, при котором потребуется решить следующие подзадачи:

- Обзор существующих подходов и реализаций, выявление полезных идей.
- Рассмотрение различных моделей камеры, выбор наиболее подходящей.
- Реализация какого-либо подхода к инициализации системы.
- Реализация Direct Image Alignment для отслеживания положения поступающих кадров.
- Получение глубин точек методом поиска вдоль эпиполярной кривой, оценка дисперсии результата.
- Совместная оптимизация поз и глубин на нескольких ключевых кадрах для повышения точности построенной модели.

Предполагается, что в дальнейшем проект получит развитие как система, работающая с группами камер, однако подобное развитие выходиит за рамки представленной курсовой работы.

3. Обзор существующих решений

Современные методы решения задач VO во многом берут своё начало в работе PTAM (Parallel Tracking and Mapping)[4], авторы которой решили определять положения только у точек, находящихся на подмножестве всех кадров (такие кадры стали называться *ключевыми*). Благодаря разряженности множества ключевых кадров появилась возможность использовать метод, ранее неприменимый к задачам реального времени — совместную оптимизацию (в оригинале *bundle adjustment, BA*).

BA предполагает минимизацию некоторой целевой функции для повышения консистентности модели. При этом как в упомянутой выше системе PTAM, так и в других появившихся вскоре успешных работах, основывающихся на сходных идеях (например, ORB-SLAM[7][2]), природа оптимизируемой функции остается геометрической: на ключевых кадрах выбираются *ключевые точки*, и высказываются предположения о соответствиях этих точек на разных кадрах, после чего позиции кадров и точек в пространстве приводятся в соответствие с этими предположениями. Такие подходы в дальнейшем стали называться *непрямыми (indirect)*, так как информация об изображении заменяется на информацию о выбранных ключевых точках на этапе препроцессинга, и дальнейшая работа ведется только с ними.

Следующей идеей, позволившей уплотнить получаемое множество точек, стало изменение природы оптимизируемой функции. В *прямых (direct)* подходах не высказывается никаких предположений о соответствиях точек на разных кадрах. Положения пикселей переносятся с одного кадра на другой, после чего накладывается требование фотометрической консистентности — интенсивность исходного пикселя должна сопадать с интенсивностью перенесенного, так что

[2] ORB — аббревиатура-название алгоритма поиска и соотнесения ключевых точек [8]; SLAM, или Simultaneous Localization and Mapping — устоявшееся название расширения алгоритмов VO функциональностью *замыкания циклов*, корректировки полученной модели на основании повторного появления в месте, посещенном ранее.

различия в них и составляют основу оптимизируемой функции. Эта идея была успешно использована в работах LSD-SLAM (Large Scale Direct SLAM)[9, 10] и DSO (Direct Sparse Odometry)[11].

При переходе к широкоугольным камерам непрямые подходы теряют в привлекательности — алгоритмы опредедения и соотнесения ключевых точек [8] начинают работать хуже из-за искажений, вызванных fisheye-линзами. Прямые подходы, наоборот, переносятся на этот случай довольно прямолинейно. У алгоритмов LSD-SLAM и DSO уже появились соответствующие аналоги — [12] и [13].

Несмотря на описанные выше проблемы сочетания indirect подходов с fisheye-объективами, пока только эти подходы были расширены на случай системы нескольких широкоугольных камер. У PTAM появился аналог MCPTAM (Multi-Camera PTAM)[14], у ORB-SLAM — MultiCol-SLAM[15]. Представленная работа может послужить основой для заполнения образовавшейся ниши.

Модель камеры

Ключевая особенность данной работы — использование широкоугольных камер. Угол обзора больше 180° позволяет охватить большую часть сцены, и совершенно необходим в случае со стандартным расположением камер в ADAS-системах. На рис. 1 видно, что не-широкоугольная камера, смонтированная в стандартном положении, не захватывает ценных деталей сцены.

(*а*) Оригинал (*б*) $\alpha = 88\ ^\circ$ (*в*) $\alpha = 53\ ^\circ$

Рис. 1. Оригинальное изображение и как оно же выглядело бы для стандартной камеры с различным горизонтальным углом обзора α

Модель камеры должна предоставлять отображение проекции $\pi : \mathbb{R}^3 \to \Omega$ множества точек пространства на область изображения $\Omega \subset \mathbb{R}^2$ и отображение «обратной» проекции π^{-1}. Геометрическим местом точек пространства, которые π отображает в определенную точку на изображении, является луч. В большинстве случаев достаточно использовать т. н. *центрально-проективные* модели, в которых все такие лучи имеют общее начало, называемое *центром* камеры. Тогда удобно записать «обратную» проекцию в формате $\pi^{-1} : \Omega \to S^2$, как отображение, сопоставляющее точке на изображении пересечение луча, ей соответствующего, и единичной сферы S^2. В дальнейшем будет использоваться именно эта нотация.

Существует несколько распространенных центральных моделей широкоугольных камер. В рамках данной работы использовалась довольно общая *полиномиальная* модель, введенная в [16]. Эта модель использует предположение о симметричности оптической системы камеры относительно главной оптической оси. В таком случае расстояние от точки на изображении до центра зависит только от вертикального наклона луча, который проецируется в эту точку. Формально говоря, в этой модели π^{-1} записывается как

$$\begin{gathered} \pi^{-1}(\mathbf{p}) = \frac{\mathbf{x}}{\|\mathbf{x}\|} \\ \mathbf{x} = \left(\tilde{\mathbf{p}}^\intercal, f\left(\|\tilde{\mathbf{p}}\|\right)\right)^\intercal \\ \tilde{\mathbf{p}} = s(\mathbf{p} - \mathbf{c}) \\ f(\rho) = a_0 + a_2\rho^2 + a_3\rho^3 + a_4\rho^4 + \cdots \end{gathered} \tag{1}$$

Таким образом, точка $\mathbf{p}$ на изображении сначала преобразуется к нормализованным координатам $\tilde{\mathbf{p}}$ с центром $\mathbf{c}$, после чего переносится вдоль оси Oz на $f\left(\|\tilde{\mathbf{p}}\|\right)$, и луч, который проецируется камерой в точку $\mathbf{p}$, задается направлением $\left(\tilde{\mathbf{p}}^\intercal, f\left(\|\tilde{\mathbf{p}}\|\right)\right)^\intercal$. На практике функция f задается конечным числом коэффициентов[3] $a_0, a_2, a_3, \ldots, a_N$ разложения в ряд Маклорена.

[3] $a_1 = 0$, т. к. требуется $f'(0) = 0$, чтобы π^{-1} было гладким в центре.

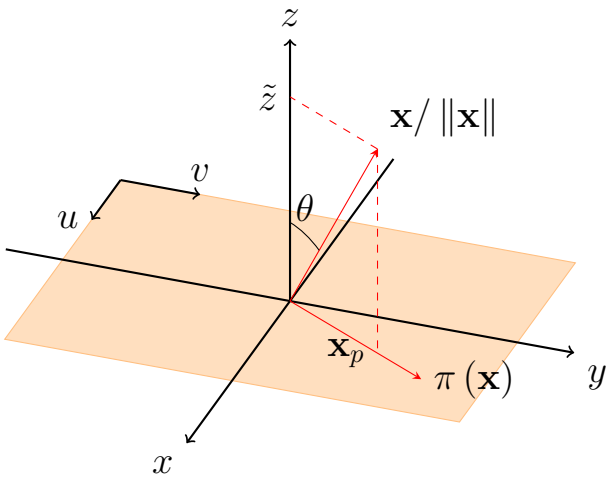


Рис. 2. Проецирование точки $\mathbf{x}$ на изображение. В первом случае $\|\pi_1(\mathbf{x})\| = g_1(\theta)$, во втором $\|\pi_2(\mathbf{x})\| = g_2(\tilde{z})$. (u, v) — координаты на изображении.

У предложенной модели есть один недостаток — для того, чтобы спроецировать точку, требуется по значению $f\left(\|\tilde{\mathbf{p}}\|\right)$ восстановить $\|\tilde{\mathbf{p}}\|$, то есть решить уравнение степени N. Более того, в дальнейшем системе потребуются производные от корня этого уравнения по коэффициентам. Всё это слишком ресурсозатратно для применения в задачах реального времени. Поэтому имеет смысл последовать за [14] и отказаться от явного вычисления обратной функции, заменив отображение π близкой моделью. Авторами для этой цели были рассмотрены два варианта (см. рис. 2), один из которых — *модель Канналы-Брандта*[17]. В рамках этой модели π записывается так:

$$\begin{gathered}\tilde{\mathbf{p}} = \pi_1(\mathbf{x}) = g_1(\theta)\frac{\mathbf{x_p}}{\|\mathbf{x_p}\|} \\ \tilde{z} = \left[\frac{\mathbf{x}}{\|\mathbf{x}\|}\right]_3 \\ \theta = \arccos(\tilde{z}) \\ g_1(\theta) = c_0 + c_1\theta + \cdots + c_M\theta^M\end{gathered} \tag{2}$$

Вторая использованная модель схожа с описанной выше, но полиномиальной полагается зависимость расстояния до центра на изображении от $\tilde{z}$:

$$\begin{gathered}\tilde{\mathbf{p}} = \pi_2(\mathbf{x}) = g_2(\tilde{z})\frac{\mathbf{x}_p}{\|\mathbf{x}_p\|} \\ g_2(\tilde{z}) = b_0 + b_1\tilde{z} + \cdots + b_M\tilde{z}^M\end{gathered} \tag{3}$$

Таблица 1. Средние ошибки репроекции для двух рассматриваемых методов

M	2	4	6	8	10	12	14
e_1, пикс.	6.99	1.50	0.36	0.14	0.049	0.014	0.005
e_2, пикс.	37.33	15.91	8.44	5.52	3.94	2.88	2.39

где $\mathbf{x}_p$ означает проецирование $\mathbf{x}$ на Oxy.

4. Описание компонентов системы

4.1. Отображение проекции

По описанным выше причинам, отображение, обратное к π^{-1} не может быть вычислено явно, поэтому авторы рассматривали функции π_1 и π_2 в качестве способов приближения отображения проекции. Для обеих функций g_i и для различных степеней M с помощью π^{-1} находилось по 2000 пар (аргумент, значение), после чего коэффициенты b_i и c_i оценивались линейной (по коэффициентам) регрессией. Далее выбирались $N_0 = 10^4$ случайных точек p_i на изображении, и для $\pi_{1,2}$ вычислялась среднеквадратичная ошибка репроекции

$$e_{1,2} = \left(\frac{1}{N_0} \sum_{i=1}^{N_0} \left\| p_i - \pi_{1,2} \left(\pi^{-1} \left(p_i \right) \right) \right\|^2 \right)^{\frac{1}{2}} \tag{4}$$

Получившаяся зависимость e_i от M приведена в таблице 1. Видно, что первый метод превосходит второй. Более того, учитывая, что сами изображения даны нам с гранулярностью в 1 пиксель, ошибка репроекции менее 0.5 может считаться приемлемой, так что первый метод пригоден к использованию.

4.2. Инициализация

В обычном режиме визуальная одометрия работает, опираясь на данные, полученные ранее. Например, чтобы оценить положение нового кадра, в данной работе используется информация о точках с предыдущего. Для самых первых

кадров ничего подобного не предоставлено, и поэтому их обработка должна производиться отдельно.

Классический для компьютерного зрения подход к определению относительного положения двух кадров — использование ключевых точек, и представленная инициализация, в противоположность остальной системе, является примером непрямого метода. Причина заключается в том, что нам хотелось как можно раньше получить работающие оценки для расположения точек и камер, которые можно было бы сравнивать с результатами работы основного алгоритма.

В данной реализации этот подход работает в два этапа: глобальная оценка движения между кадрами и глубин ключевых точек, а затем интерполяция глубин контрастных точек. Рассмотрим подробнее, что эти этапы из себя представляют.

Глобальная оценка позы

Сначала на двух первых ключевых кадрах выделяются и сопоставляются ключевые точки ORB. Из-за наличия соответвствий-*аутлаеров*, то есть равномерно распределенных ошибок, требуется метод отделения верно полученных пар от неверных. Для этого применяется широко распростараненный вероятностный метод RANSAC (Random Sample Consensus)[18], выбирающий наилучшую *минимальюную модель*. Оказывается, с точностью до неоднозначности в 40 вариантов, можно определить относительное движение камеры между двумя снимками по пяти соответствиям точек на них ([19], глава 6). Построенная таким образом оценка движения и называется минимальной моделью. Для модели проверяется, сколько всего соответствий с ней согласуются, то есть у скольки геометрическая ошибка репроекции не превосходит определенного значения.

Ошибка вычисляется по формулам

$$\begin{aligned} err\left(p_1, p_2, \xi\right) &= \min\left(R\left(p_1, p_2, \xi\right), R\left(p_2, p_1, \xi^{-1}\right)\right) \\ R\left(p_1, p_2, \xi\right) &= \left\| p_1 - \pi\left(pr_{e_2(p_2,\xi)}\left(\pi^{-1}\left(p_1\right)\right)\right)\right\|^2 \end{aligned} \tag{5}$$

где p_1 и p_2 — точки соответствия на первом и на втором кадрах, ξ — отображение координат первого кадра на координаты второго, pr_{e_2} — функция проекции на плоскость, проходящую через центры камеры и луч $\pi^{-1}\left(p_2\right)$ (т. н. *эпиполярная* плоскость). Поскольку для p_1 и p_2 системе неизвестны *глубины* точек (расстояния до центра камеры), они считаются соответствующими модели, если центры камеры и два луча в направлениях p_1 и p_2 лежат на общей эпиполярной плоскости. Описанная ошибка репроекции и есть количественная мера близости к этому факту.

Полученное с помощью RANSAC движение уже достаточно близко к действительности, чтобы определить, какие соответствия являются инлаерами, но ему не хватает точности для дальнейшего использования. Для решения этой проблемы в данной работе производится усреднение позы по инлаерам. Конкретнее, алгоритмом Левенберга-Марквардта минимизируется общая ошибка репроекции

$$\sum_{(p_1,p_2)\in M} R\left(p_1, p_2, \xi\right) + R\left(p_2, p_1, \xi^{-1}\right) \tag{6}$$

где M — множество соответствий-инлаеров. Для минимизации используется ceres-solver[20].

Интерполяция

До завершения инициализации системы теперь остается еще один шаг. Представленный выше способ оценки движения между кадрами выдает набор ключевых точек и глубины для них. Но остальная часть системы опирается на положения *контрастных* точек — тех, в которых градиент изображения доста-

точно велик. Использование глубин именно в таких точках критично для прямых подходов: в противном случае основанные на дифференцировании методы оптимизации просто не будут работать. Было решено воспользоваться интерполяцией на нерегулярной сетке, для которой была использована триангуляция Делоне (см. рис. 3).

При этом триангуляция производится по набору ключевых точек, и оценка на глубину в контрастной (проколотой на рисунке) точке считается линейно по трем ключевым точкам, в треугольник из которых попала эта контрастная точка. Можно заметить искривление ребер сетки — оно вызвано тем, что рёбра являются прямыми в трехмерном пространстве, а не на изображении.

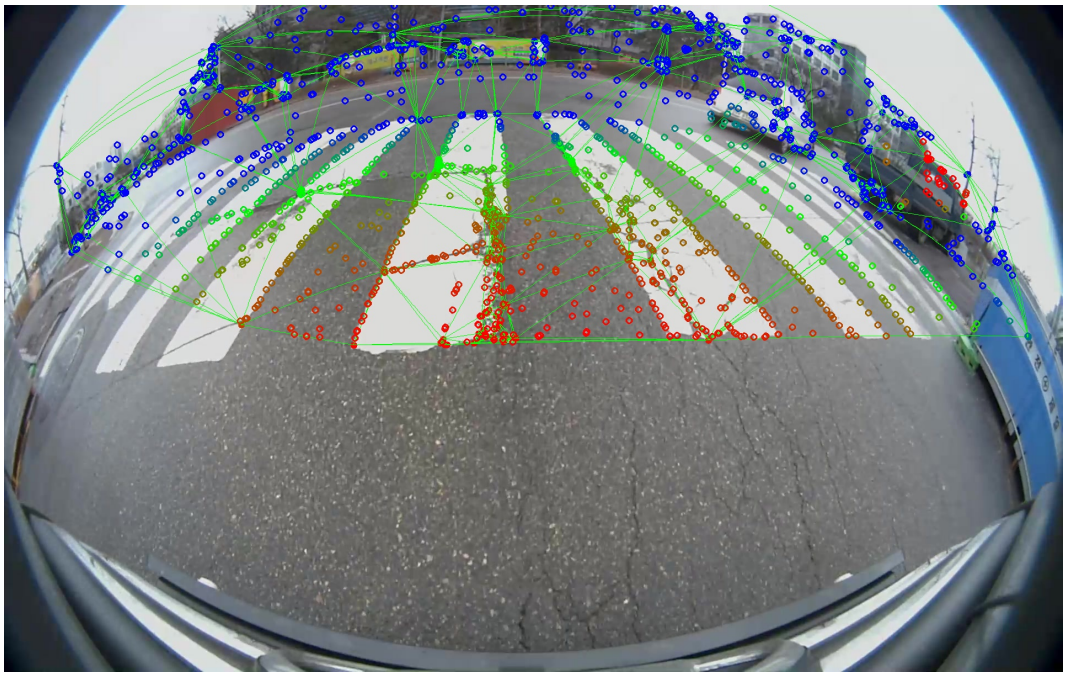


Рис. 3. Пример успешной инициализации представленной системы

4.3. Direct image alignment

Теперь, когда у системы откуда-либо появилась приблизительная карта глубин в контрастных точках кадра, появляется возможность давать точные оценки на последующие движения камеры. Техника, которой было предложено для этого воспользоваться, основывается на статье [21].

Прежде, чем формулировать задачу нелинейной оптимизации, сделаем важные замечания. Во-первых, чтобы direct-метод работал, требуется учесть автоматическую регулировку выдержки и/или размера диафрагмы на камере во время съемки. В данной работе, вслед за [11], эти эффекты математически моделируются с помощью *аффинного преобразования яркости*. Это значит, что в оптимизационной задаче у каждого кадра появляются дополнительные параметры a и b, и «эталонная» яркость пикселя p на кадре I становится равной $\mathrm{e}^{-a} I(p) - b$. Экспонента — тоже предложение из [11], и используется для удобства параметризации. Во-вторых, для представления изображения как дифференцируемой функции $I : \Omega \to \mathbb{R}$, требуется какая-либо интерполяция. В данной работе используется бикубическая интерполяция [22], встроенная в ceres-solver[20].

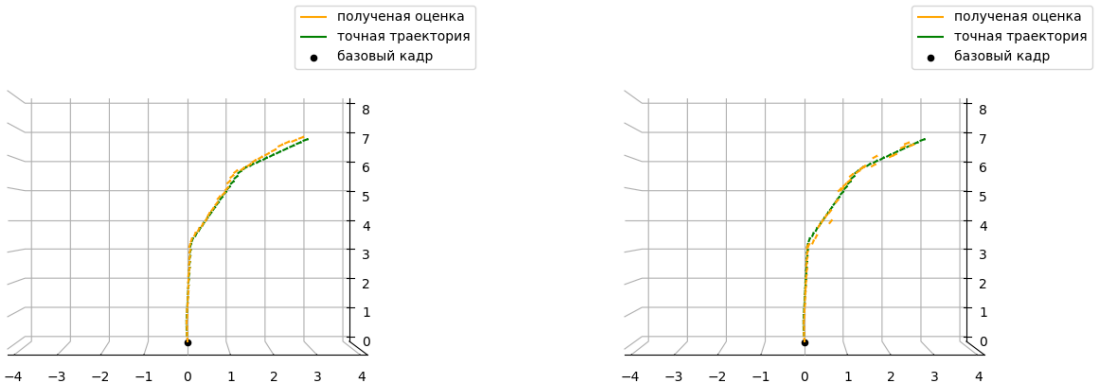
Итак, при поиске движения, произошедшего между кадрами I_1 и I_2, минимизируется фотометрическая ошибка репроекции

$$\begin{gathered} \sum_{p \in P} \rho \left(I_1(p) - \mathrm{e}^{a} \left(I_2\left(p'\right) + b \right) \right) \\ p' = \pi \left(\xi \cdot \left(d_p * \pi^{-1}(p) \right) \right) \end{gathered} \tag{7}$$

где I_1 — базовый кадр, P — множество контрастных точек p с глубинами d_p на нем; I_2 — кадр, положение которого определяется, p' — точка p, перенесенная с I_1 на I_2. ρ — *функция потерь*. Из-за того, что некоторые точки I_1 при репроецировании на I_2 попадут на новые объекты (из-за эффекта параллакса, или из-за движения в кадре), приходится учитывать наличие аутлаеров. Для уменьшения их влияния на систему и используется функция потерь.

$$\rho(x) = \begin{cases} \left(\frac{x}{x_{max}} \right)^2 & x \leq x_{max} \\ 2\frac{x}{x_{max}} - 1 & x > x_{max} \end{cases} \tag{8}$$

Для того, чтобы корректно определять движения при существенном отклонении начальных данных от истины, было предложено воспользоваться идеей



(*а*) Точные глубины (*б*) Глубины с 5% шумом

Рис. 4. Отслеживание положений 40 кадров относительно общего базового

из [21]. Все изображения изначально подаются алгоритму в низком разрешении, и, после очередного этапа минимизации, разрешение повышается.

Для тестирования данного алгоритма независимо от остальной системы использовался датасет-симуляция движения автомобиля с широкоугольной камерой по городу Multi-FoV[23]. Поскольку датасет синтетический, у авторов была возможность предоставить точные глубины в каждом пикселе. Для обработки выбирались базовые кадры на моменте поворота, и в контрастных точках использовались истинные глубины с 5% шумом. Одна из оценённых траекторий представлена на рис. 4. Видно несколько мест, где алгоритм сбивался с траектории, но возвращался на следующем же кадре.

4.4. Отслеживание точек вдоль эпиполярной кривой

С помощью метода, описанного выше, оценивается взаимное расположение между более старым кадром и самым новым, поступившим в систему. Из полученной оценки определяется геометрическое место точек на новом кадре, куда могла переместиться точка p со старого — т. н. *эпиполярная кривая*. Смещение точки вдоль этой кривой называется *диспаритетом*, из которого можно определить глубину точки.

Для определения диспаритета требуется найти на эпиполярной кривой

(*а*) Точка на старом кадре

(*б*) Эпиполярная кривая на новом

Рис. 5. Отслеживание точки вдоль эпиполярной кривой

максимально похожую точку. Для этого алгоритм перебирает точки p' с шагом в 1 пиксель и выбирает ту, которая минимизирует ошибку репроекции:

$$p'_{best} = \arg\min \sum_{v \in N_p} \rho \left(I_1 \left(p + v \right) - I_2 \left(p' + \mathbf{R} v \right) \right) \tag{9}$$

Где N_p — *паттерн разреженности*. Он определяет набор из нескольких точек в окрестности p, отслеживающихся вдоль кривой. На новом кадре паттерн поворачивается и масштабируется, в (9) это отражено умножением на $\mathbf{R}$.

На рис. 5 показан пример поиска вдоль эпиполярной кривой. Цвет кодирует значение ошибки репроекции (красный для меньших значений), чёрная засечка — выбранный минимум.

Оценка погрешности

Для отсеивания некачественно определённых глубин точек требуется метод оценки погрешности результата отслеживания вдоль эпиполярной кривой. Было решено воспользоваться способом, представленным в статье [9]. Этот метод предполагает приблизительное выражение искомого диспаритета через параметры с известной ошибкой, и использование *распространения неопределенности* для выражения ошибки диспаритета. Авторами [9] предлагается учитывать два фактора — ошибку в размещении эпиполярной кривой σ_p и ошибку в интенсивности пикселей σ_i. Формулы для расчёта ошибок диспаритета:

$$\sigma_{\lambda,l} = \frac{\| g \| \sigma_l}{\langle g, l \rangle} \qquad \sigma_{\lambda,i} = \frac{\sigma_i}{\langle g, l \rangle} \tag{10}$$

Где g — градиент нового изображения в найденной точке, l — нормированный вектор касательной к эпиполярной кривой. Общая ошибка вычисляется как $\sigma_\lambda^2 = \sigma_{\lambda,l}^2 + \sigma_{\lambda,i}^2$. Здесь σ_l — ожидаемая ошибка в положении эпиполярной кривой, а σ_i — ожидаемая ошибка в интенсивности. Обе подбираются как эмпирические константы. Однако, при разумных значениях σ_l и σ_i оказывается, что вклад $\sigma_{\lambda,i}$ несущественнен, поэтому в данной работе использовалась только компонента $\sigma_{\lambda,l}$.

Субпиксельное отслеживание

Линейная природа поиска не позволяет определить новое положение точки с погрешностью менее ± 0.5 пикселя. Это ограничение можно ослабить, если после выбора минимума дополнительно уточнить его с помощью какого-либо алгоритма оптимизации одномерной функции. В данной работе для этого использовался алгоритм Гаусса-Ньютона. При этом изображение, опять же, должно быть проинтерполировано. Сравнение работы алгоритма со включённым суб-

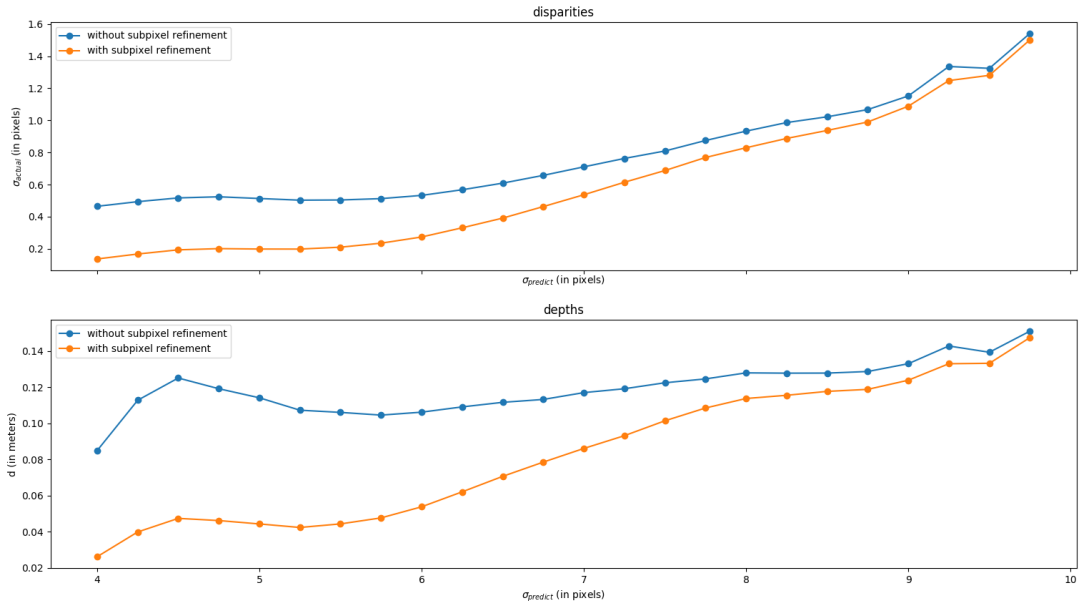


Рис. 6. Ошибки в диспаритетах (верх) и глубинах (низ) на участке датасета Multi-FoV в зависимости от предсказанных ошибок в диспаритетах

пиксельным отслеживанием и без него представлено на рис. 6.

На графиках по оси абсцисс отложены предсказанные ошибки в диспаритете ($\sigma_{predict}$). Для каждого значения $\sigma_{predict}$ рассматривались точки с предсказанной ошибкой из $\left[\sigma_{predict} - \frac{1}{8}, \sigma_{predict} + \frac{1}{8}\right]$. Для выбранных точек определялось медианное значение реальной ошибки, которое и попадало на график.

Предлагается обратить внимание на следующие аспекты графиков:

1. Увеличение ошибок диспаритета при увеличении предсказанной ошибки. Это показывает осмысленность выбранной модели оценки ошибок.

2. Без уточнения ошибка в диспаритете не опускается ниже 0.5.

3. Субпиксельное уточнение даёт существенный выигрыш для оценок реальных глубин. При небольшой предсказанной ошибке глубины с субпиксельным отслеживанием точнее в 2 раза.

4.5. Совместная оптимизация

Авторы системы DSO сумели повысить точность своего алгоритма, применив метод *совместной оптимизации*, сформулировав его для прямого подхода. В представленной работе было решено последовать их примеру.

Когда у системы есть набор из ключевых кадров и выбраны особенно хорошие оценки глубин на них, становится возможным дальнейшее уточнение положений и глубин с помощью минимизации прямой функции ошибок:

$$E = \sum_{I_1 \in F} \sum_{p \in P(I_1)} \sum_{I_2 \in obs(p)} r\left(I_1, I_2, p\right)$$

$$r\left(I_1, I_2, p\right) = \sum_{q \in N_p} \rho\left(I_2\left(reproj\left(q, d\left(p\right), \xi_2 \xi_1^{-1}\right)\right) + b_1 - \mathrm{e}^{a_2 - a_1}\left(I_1\left(q\right) + b_2\right)\right)$$

$$q' = \pi\left(\xi_2 \xi_1^{-1} \cdot \left(d_p * \pi^{-1}\left(q\right)\right)\right) \tag{11}$$

где F — текущее множество ключевых кадров, $P(I_1)$ — множество контрастных точек на кадре I_1, $obs(p) \subseteq F \setminus \{I_1\}$ — множество кадров, из которых видна точка p, N_p — описанный выше паттерн разреженности.

Для того, чтобы время работы совместной оптимизации не росло со временем, требуется жёстко ограничить размер множества ключевых кадров F сверху. Предполагается, что в F попадают недавние кадры, каждый из которых, однако, должен охватывать существенно отличную от остальных часть сцены. Алгоритм отбора ключевых кадров ещё предстоит реализовать, сейчас выбирается каждый k-й кадр ($k = 18$).

5. Программная реализация

Все описанные выше компоненты были реализованы и объединены в систему визуальной одометрии. На данный момент система обрабатывает видео не в реальном времени, так как целью работы было возможно скорейшее доказательство работоспособности используемых идей для одометрии.

5.1. Используемые технологии

Разработка ведётся на языке C++. Выбор языка обусловлен, прежде всего, наличием ряда удобных математических open-source библиотек. Конкретнее, в данной работе используются:

- Eigen[24] — Библиотека структур и алгоритмов линейной алгебры.
- Sophus[25] — Реализация геометрических групп Ли ($SO(3)$, $SE(3)$).
- ceres-solver[20] — библиотека для удобного построения и автоматического решения затач нелинейной оптимизации. Её возможности включают различные алгоритмы минимизации, автоматическое дифференцирование, работу с разреженными матрицами и пр.

В связи с выбором C++ в проекте также используются:

- CMake[26] — кроссплатформенная система сборки.
- OpenCV[27] — библиотека для работы с изображениями. OpenCV содержит реализацию определения ключевых точек и вычисления дескрипторов ORB, которые используются в фазе инициализаци представленной системы.
- GFlags[28] — удобный парсинг флагов коммандной строки.
- GLog[29] — библиотека для логирования.

5.2. Архитектура системы

Основные элементы алгоритма визуальной одометрии представлены на рис. 7.

Главный класс системы называется `DsoSystem`. Он связывает её компоненты и реализует алгоритм с рис. 7. Все настройки системы представлены в виде иерархической структуры (класс `Settings`), которая передаётся в `DsoSsystem`, и части которой распределяются по компонентам системы. При этом

- За инициализацию системы отвечает класс `DsoInitializer`
- За отслеживание положений новых кадров — `FrameTracker`
- За отслеживание точек — `ImmaturePoint`
- За совместную оптимизацию — `BundleAdjuster`

Логика, связанная с выводом, отделена от логики системы. Для этого использовался паттерн "Наблюдатель". У части компонентов системы (`DsoSystem`, `FrameTracker`, `DsoInitializer`) созданы интерфейсы наблюдателей, которые позволяют, среди прочего

Рис. 7. Структура алгоритма одометрии

- Сохранять определившуюся траекторию в файл (в дальнейшем она может быть отрисована с помощью скрипта на Python)
- Сохранять полученное облако точек
- Выравнивать правильные траектории и облака точек для сравнения с определившимися
- Показывать отладочный вывод — глубины точек на текущем кадре, предполагаемые ошибки в них и пр.

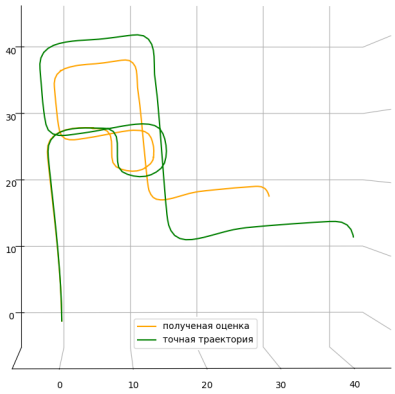


Рис. 8. Отслеженная траектория в сравнении с ground truth

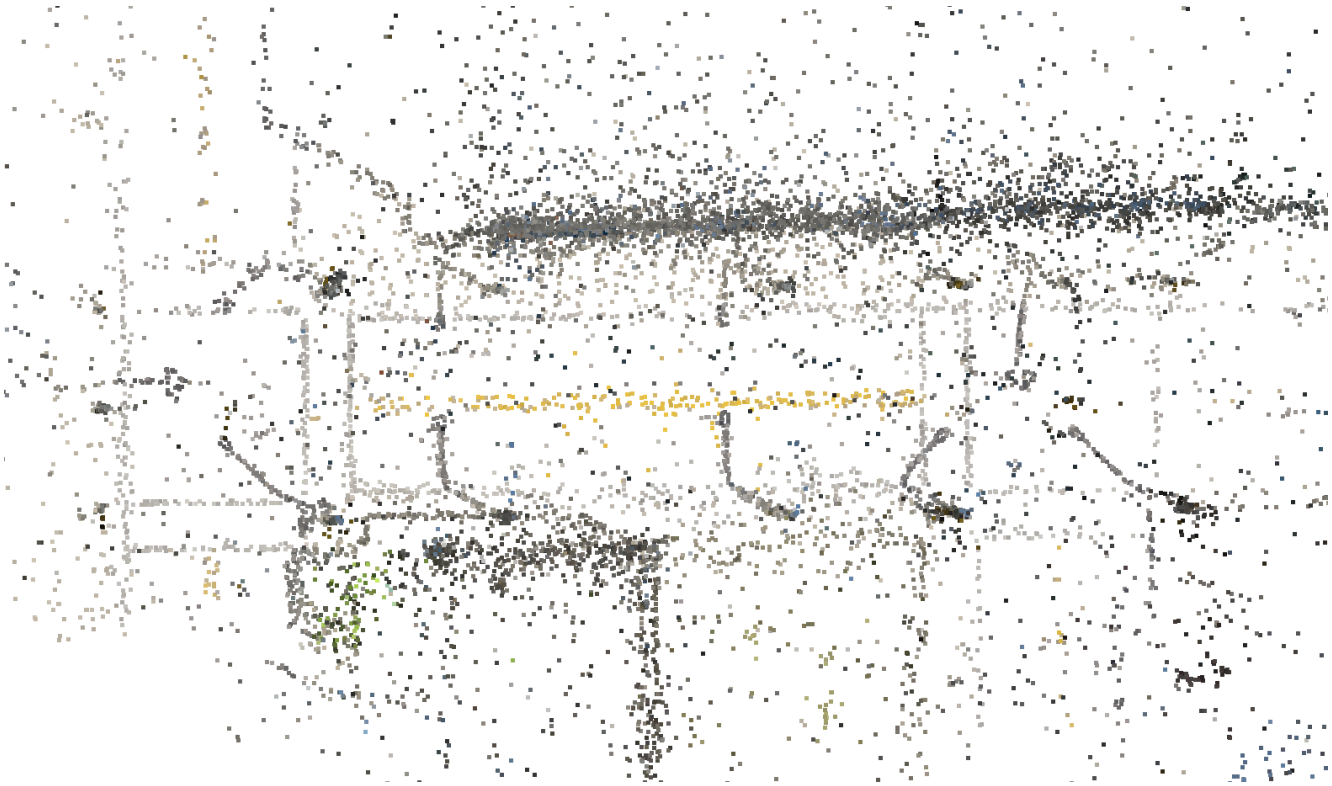
6. Результаты

Была построена система визуальной одометрии для широкоугольной камеры. При этом были решены следующие задачи:

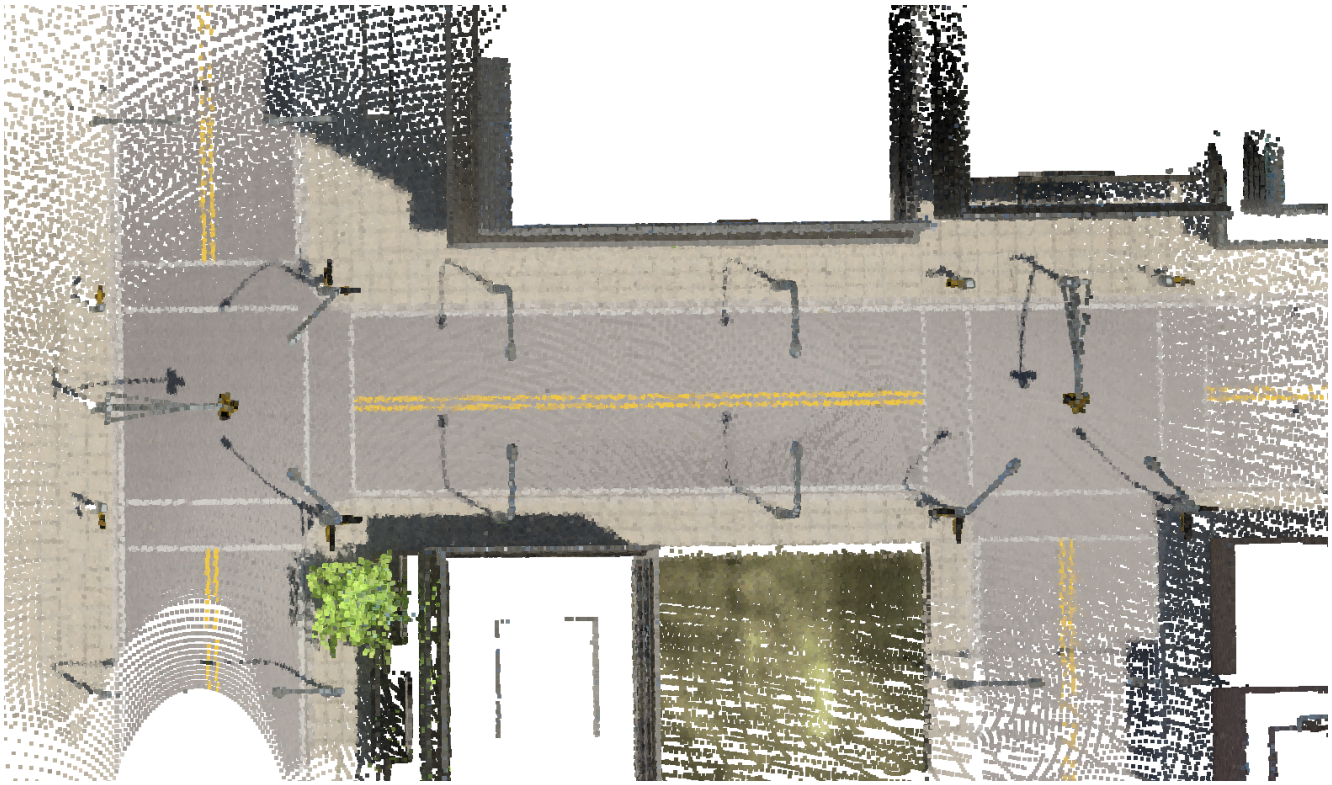
- Проведен обзор существующих решений задачи VO.
- Выбрана и реализована подходящая модель камеры.
- Реализован подход к инициализации системы.
- Реализован подход Direct Image Alignment для отслеживания положения поступающих кадров.
- Реализовано отслеживание точек вдоль эпиполярной кривой.
- Реализован прямой подход для совместной оптимизации нескольких ключевых кадров.

Система обработала синтетический датасет Multi-FoV[23] целиком. Сравнение полученной траектории с ground truth представлено на рис. 8.

Кроме того, на рис. 9 представлено сравнение полученного облака точек с точным облаком при запуске системы на участке датасета с двумя поворотами.



(*а*) Полученное облако точек



(*б*) ground truth

Рис. 9. Полученное облако точек в сравении с ground truth

Список литературы

1. Szeliski R. Computer Vision: Algorithms and Applications. — Springer, 2010.
2. Frahm J. et al. Building rome on a cloudless day // Computer Vision – ECCV 2010. — Springer, 2010. — P. 368–381.
3. Reconstructing the world* in six days *(as captured by the yahoo 100 million image dataset) / J. Heinly, J. Schönberger, E. Dunn, JM. Frahm // CVPR. — 2015.
4. Klein G., Murray D. Parallel tracking and mapping for small ar workspaces // 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. — IEEE, 2007.
5. Engel J., Sturm J., Cremers D. Camera-based navigation of a low-cost quadrocopter // 2012 IEEE/RSJ International Conference on Intelligent Robots and System. — IEEE, 2012.
6. Maimone M., Cheng Y., Matthies L. Two years of visual odometry on the mars exploration rovers // Journal of Field Robotics. — 2007. — Vol. 24, no. 3.
7. Mur-Artal R., Montiel J., Tardos J. Orb-slam: A versatile and accurate monocular slam system // Transactions on Robotics. — 2015. — Vol. 31, no. 5. — P. 1147–1163.
8. Orb: An efficient alternative to sift or surf / E. Rublee, V. Rabaud, K. Konolige, G. Bradski // European Conference on Computer Vision. — 2011. — P. 2564–2571.
9. Engel J., Sturm J., Cremers D. Semi-dense visual odometry for a monocular camera // Proceedings of the IEEE international conference on computer vision. — IEEE, 2013. — P. 1449–1456.
10. Engel J., Schöps T., Cremers D. Lsd-slam: Large-scale direct monocular slam // European Conference on Computer Vision. — Springer, 2014. — P. 834–849.
11. Engel J., Koltun V., Cremers D. Direct sparse odometry // IEEE transac-

tions on pattern analysis and machine intelligence. — 2018. — Vol. 40, no. 3. — P. 611–625.

12. Caruso D., Engel J., Cremers D. Large-scale direct slam for omnidirectional cameras // IROS. — 2015.
13. Omnidirectional dso: Direct sparse odometry with fisheye cameras / H. Matsuki, L. von Stumberg, V. Usenko et al. // IEEE Robotics and Automation Letters and Int. Conference on Intelligent Robots and Systems (IROS). — IEEE, 2018.
14. Harmat A., Trentini M., Sharf I. Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments // Journal of Intelligent and Robotic Systems. — 2015. — Vol. 78, no. 2. — P. 291–317.
15. Urban S., Hinz S. Multicol-slam - a modular real-time multi-camera slam system. — 1610.07336.
16. Scaramuzza D., Martinelli A., Siegwart R. A flexible technique for accurate omnidirectional camera calibration and structure from motion // Fourth IEEE International Conference on Computer Vision Systems (ICVS'06). — 2006. — P. 45.
17. Kannala J., Brandt S. S. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses // IEEE transactions on pattern analysis and machine intelligence. — 2006. — Vol. 28, no. 8. — P. 1335–1340.
18. Fischler M. A., Bolles R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography // Communications of ACM. — 1981. — jun. — Vol. 24, no. 6. — P. 381–395.
19. Stewenius H. Gröbner Basis Methods for Minimal Problems in Computer Vision : Ph. D. thesis / H. Stewenius ; Lund University. — Centre for Mathematical Sciences, Lund University, 2005. — P. 183.
20. Agarwal S., Mierle K. et al. Ceres solver. — Access mode: http://ceres-solver.org (online; accessed: 2019-05-16).
21. Kerl C., Sturm J., Cremers D. Robust odometry estimation for rgb-d cam-

eras // Robotics and Automation (ICRA), 2013 IEEE International Conference on / IEEE. — 2013. — P. 3748–3754.

22. Keys R. Cubic convolution interpolation for digital image processing // IEEE transactions on acoustics, speech, and signal processing. — 1981. — Vol. 29, no. 6. — P. 1153–1160.
23. Benefit of large field-of-view cameras for visual odometry / Zichao Zhang, Henri Rebecq, Christian Forster, Davide Scaramuzza // Robotics and Automation (ICRA), 2016 IEEE International Conference on / IEEE. — 2016. — P. 801–808.
24. Eigen — c++ template library for linear algebra. — Access mode: http://eigen.tuxfamily.org/ (online; accessed: 2019-05-16).
25. Sophus — c++ implementation of lie groups using eigen. — Access mode: http://eigen.tuxfamily.org/ (online; accessed: 2019-05-16).
26. Cmake — cross-platform build system. — Access mode: https://cmake.org (online; accessed: 2019-05-16).
27. Opencv — open source computer vision library. — Access mode: https://opencv.org (online; accessed: 2019-05-16).
28. Gflags — c++ library that implements commandline flags processing. — Access mode: https://github.com/gflags/gflags (online; accessed: 2019-05-16).
29. Glog — c++ implementation of the google logging module. — Access mode: https://github.com/google/glog (online; accessed: 2019-05-16).