Алгоритмы вычисления оптического потока в задаче определения собственного движения

Пономарев Е.С.[1,2], Григорьев А.С.[1,2]

[1] Институт Проблем Передачи Информации (ИППИ РАН)
[2] Московский Физико-Технический Институт (МФТИ)
evgps@ya.ru, me@ansgri.com

Аннотация Определение собственного движения камеры - важная задача в компьютерном зрении. В данной работе рассматривается совместное применение алгоритмов нахождения собственного движения (ego-motion estimation) и алгоритмов вычисления оптического потока(optical flow). Производится обзор ряда методов в каждой из составляющих задачи. Для численного сравнения строится метод, использующий алгоритмы нахождения плотного оптического потока с последующим вычислением по нему вращения камеры и уточнением с помощью RANSAC. Приводятся численные результаты его работы.

Исследование выполнено в ИППИ РАН за счет гранта Российского научного фонда (проект №14-50-00150)

1 Введение

Для роботизированных автономных систем, таких, как беспилотные наземные транспортные средства, задача определения собственного положения в пространстве в каждый момент времени крайне важна. Она может решаться с помощью разных средств - подсчета оборотов колеса, использования системы GPS/ГЛОНАСС, компаса, гироскопа, информации с камер или локаторов, закрепленных на таком ТС и некоторых прочих. Визуальный способ - один из наиболее часто используемых и перспективных. Его работа не зависит от изменения магнитного поля, пробуксовки колёс, уровня связи со спутниками. Кроме того, он пассивен, что повышает скрытность системы. Недостатками можно считать зависимость данного метода от освещенности, текстурированности сцены, вычислительную сложность.
В настоящее время существуют несколько подходов определения собственного движения с помощью камер. Наилучшие результаты, что вполне закономерно, получаются при использовании стереопары [7]. В данной работе речь пойдет о монокулярных способах, т.е. использующих последовательности изображений лишь с одной камеры.

В последние годы были построены достаточно точные монокулярные методы определения собственного движения камеры. Абсолютное большинство их основаны на нахождении и сопоставлении т.н. особых точек. В свою очередь, использование алгоритмов вычисления плотного оптического потока[3] имеет несколько важных потенциальных преимуществ:

1. Работоспособность на низкотекстурированных сценах. Там, где сложно найти достаточное количество особых точек.
2. Возможность без дополнительных действий получить детализированный рельеф наблюдаемого мира, так как при таком подходе сразу вычисляются видимые движения **всех** точек изображения.

Стоит заметить, что описаний методов, которые действительно работают по такому принципу, крайне мало в литературе и нет в таблице сравнения алгоритмов визуальной навигации Технологического Института Карлсруэ (KITTI/odometry) [7]. Однако, например в работе, посвященной подробнейшему сравнению и описанию алгоритмов нахождения собственного движения [10] можно найти, помимо прочего, результаты их работы как на эталонном (ground truth), так и на посчитанном методом Farnebäck [3] оптическом потоке. К сожалению, в [10] был рассмотрен только один алгоритм вычисления оптического потока, и сравнение было проведено на синтетических данных.
В то же время, существуют довольно обширные базы разнообразных способов вычисления плотного оптического потока. Например, таблицы алгоритмов вычисления оптического потока Миддлберийского колледжа (датасет Middlebury) [1] и Технологического Института Карлсруэ (датасет KITTI/flow) [7]. Сравнения описаны в различных статьях [6,1]. Чаще всего, такие сравнения производятся в метриках средней абсолютной или среднеугловой ошибки в нахождении векторов скорости точкек изображения на каких-нибудь эталонных примерах. Такое знание полезно, т.к. дает примерное представление о точности нахождения оптического потока, однако сложно проецируется на нашу задачу. Ведь совершенно не ясно, как различные по характеру ошибки влияют на точность конечного решения.

---

[3] ***Оптический поток*** *— это изображение видимого движения объектов, поверхностей или краев сцены, получаемое в результате перемещения наблюдателя (глаз или камеры) относительно сцены.*
Одна из возможных формализаций этого определения: если $I_0(x, y)$ - первый серый кадр видео (функция интенсивности от координаты точки на изображении), а $I_1(x, y)$ - второй кадр, содержащий точно те же самые точки, только смещенные(интенсивность сохранена, края изображения и наложение объектов не рассматриваются), то векторное поле

$$V(x, y) = (u(x, y), v(x, y)) \quad : \quad I_1(x, y) = I_0(x + u, y + v) \forall (x, y) \in \mathbb{I} \tag{1}$$

и есть оптический поток. „Плотный“ означает то, что найден он для всех точек изображения. Далее „плотный“ будем опускать.

2 Постановка задачи

Целью данной работы является исследование работы алгоритмов вычисления плотного оптического потока в задаче определения собственного движения.

Для достижения поставленной цели необходимо было решить следующие **задачи**:

1. Исследовать связь между видимым движением объектов неподвижного мира и собственным движением наблюдателя.
2. Рассмотреть существующие методы монокулярной визуальной одометрии.
3. Исследовать алгоритмы вычисления оптического потока.
4. Исследовать алгоритмы, позволяющие по найденным скоростям множества точек на изображении определить собственное движение камеры.
5. Построить метод, позволяющий оценить собственное движение камеры с помощью снятой ею последовательности кадров.
6. Реализовать построенный метод в виде программы/комплекса программ, обеспечив удобство тестирования.
7. Оценить точность реализованного метода на эталонных данных.
8. Сравнить различные алгоритмы вычисления оптического потока применительно к разработанному методу.

3 Обзор существующих работ

В данном разделе мы рассмотрим основные положения алгоритмов определения собственного движения по видеопоследовательности с одной камеры. Они работают в рамках определенной модели движения камеры, поэтому разумно начать с ее описания.

3.1 Геометрия задачи

Мы проделаем рассуждения, подобные тем, что рассматриваются в [2]. Пусть $\boldsymbol{X} = (X, Y, Z)$ - координаты точки трехмерного пространства в системе координат, связанной с камерой. Модель камеры – камера-обскура. Тогда проекция $\boldsymbol{X}$ на плоскость изображения: $\boldsymbol{x} = (x, y) = \frac{f}{Z}(X, Y)$. Где f – фокусное расстояние. Пусть $\boldsymbol{\nu} = (\nu_x, \nu_y, \nu_z)$ - переносная скорость точки O (начала системы координат, связанной с камерой, относительно мировой системы

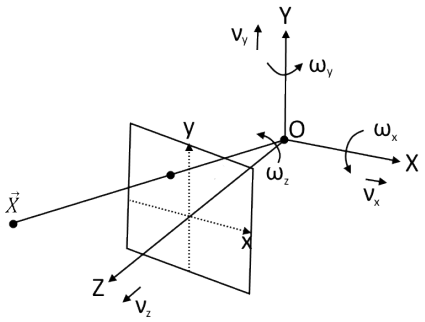


Рис. 1

координат). t_i - проекции этой скорости на оси системы координат камеры, $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ - вектор угловой скорости. В этих обозначениях движение $\boldsymbol{V}$ точки $\boldsymbol{X}$:

$$\begin{aligned} \boldsymbol{V} &= -\boldsymbol{\nu} - [\boldsymbol{\omega} \times \boldsymbol{X}], \\ X' &= -\nu_x - \omega_y Z + \omega_z Y, \\ Y' &= -\nu_y - \omega_z X + \omega_x Z, \\ Z' &= -\nu_z - \omega_x Y + \omega_y X, \end{aligned} \tag{2}$$

Тогда оптический поток $(u(x,y), v(x,y))$ в точке (x,y) :

$$\begin{aligned} u &= \frac{fX'}{Z} - \frac{fXZ'}{Z^2} = (-\frac{\nu_x}{Z} - \omega_y + \omega_z y) - x(-\frac{\nu_z}{Z} - \omega_x y + \omega_y x) \\ v &= \frac{fY'}{Z} - \frac{fYZ'}{Z^2} = (-\frac{\nu_y}{Z} - \omega_z x + \omega_x) - y(-\frac{\nu_z}{Z} - \omega_x y + \omega_y x) \end{aligned} \tag{3}$$

Напомним, что $y = \frac{Y}{Z}; x = \frac{X}{Z}$
В матричном виде:

$$\boxed{\boldsymbol{u}(x,y) = \frac{1}{Z(x,y)} A(x,y)\boldsymbol{\nu} + B(x,y)\boldsymbol{\omega}} \tag{4}$$

Где

$$\begin{aligned} A &= \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \\ B &= \begin{bmatrix} (xy)/f & -(f + x^2/f) & y \\ f + y^2/f & -(xy)/f & -x \end{bmatrix} \end{aligned} \tag{5}$$

Т.к. и $\boldsymbol{\nu}$ и $1/Z(x,y)$ неизвестны и перемножены в уравнении, то определить их лучше, чем с точностью до множителя, нельзя. Будем далее полагать, что $||\boldsymbol{\nu}|| = 1$.
В данной системе из двух уравнений 6 неизвестных (3 компоненты $\boldsymbol{\omega}$, 2 компоненты $\boldsymbol{\nu}$ (помним про условие $||\boldsymbol{\nu}|| = 1$) и глубина $Z(x,y)$). Первые пять неизвестных одинаковы для всех точек (x,y), глубина зависит от выбранной точки. Так что в идеальном случае потребуется записать 4 для 5 пар точек.

4 Декомпозиция задачи

Из геометрии видно, что в решении такой задачи три подпункта:

1. **Вычисление оптического потока.** Вычисление видимой скорости $\boldsymbol{u(x}$ некоторого множества точек на изображении по двум (или более) кадрам - см (1).

2. **Вычисление собственного движения по оптическому потоку.** Решение (4) на основе найденного в первом подпункте $\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x} \in \Omega$, где $\Omega \in \mathbb{I}$ - какое-то *подмножество* всех точек изображения.
3. **Выбор решения.** Различный выбор множества Ω может привести к различным(зачастую далеким от истинного) результатам. Голосованием на случайных выборках(RANSAC) из множества результатов работы алгоритма можно избежать ошибки.

4.1 Вычисление оптического потока

Для понимания принципов, на которых построены используемые в работе алгоритмы, приведем их краткое описание.

Основное уравнение дифференциальных методов вычисления оптического потока: Базовым предположением для вычисления оптического потока является **сохранение точкой своей интенсивности**:

$$I(\boldsymbol{x}, t) = I(\boldsymbol{x} - \boldsymbol{u}, t + 1); \tag{6}$$

где $I(\boldsymbol{x}, t)$ это интенсивность как функция от координаты пикселя на изображении $\boldsymbol{x} = (x, y)$ и времени(или номера кадра) t. Ею можно охарактеризовать видеопоток. $\boldsymbol{u} = (u, v)$ - смещение точки. Везде далее *смещение точки между двумя последовательными кадрами и её скорость мы будем отождествлять, т.е. считать, что между кадрами проходит единичный интервал времени.*
Конечно, сохранение яркости *(brightness constancy)* редко выполняется точно, потому что от кадра к кадру могут меняться, например, глобальные условия освещения и освещенность самого движущегося объекта. Масса проблем связана с этим допущением, но, как ни странно, оно достаточно хорошо работает на практике.

Следующее предположение о **малости $\boldsymbol{u}$**. Условимся искать лишь малые смещения. Если смещение большое, то уменьшив разрешение изображения, мы можем привести к данному случаю[4]. Запишем формулу Тейлора для функции I в точке (x, y) и пренебрежем членами степени выше 1 по u, v:

$$I(x - u, y - v, t) = I(x, y, t) - \boldsymbol{u} \cdot \nabla I(x, y, t) \tag{7}$$

В соответствии с (6):

$$I(x, y, t + 1) = I(x - u, y - v, t) = I(x, y, t) - \boldsymbol{u} \cdot \nabla I(x, y, t) \tag{8}$$

Между кадрами проходит единичный интервал времени, поэтому

$$I(x, y, t + 1) - I(x, y, t) = -\frac{\partial I(x, y, t)}{\partial t} \tag{9}$$

[4] Данный прием (multi-scaling) плох тем, что теряется большое движение малых объектов, которые просто исчезают при слишком маленьком разрешении.

Окончательно, получим уравнение переноса:

$$\boxed{\frac{\partial I(x,y,t)}{\partial t} + u \cdot \frac{\partial I(x,y,t)}{\partial t} + v \cdot \frac{\partial I(x,y,t)}{\partial t} = 0} \tag{10}$$

Данное уравнение является *основным уравнением дифференциальных методов вычисления оптического потока.*
Оно содержит две неизвестные и не может быть разрешено однозначно. Данное обстоятельство известно как *проблема апертуры.* Проблему решает наложение дополнительных ограничений — регуляризация[8]. В следующих подпунктах рассмотрим несколько гипотез о характере изменения изображения, решающих проблему апертуры наложением дополнительного условия.

Horn-Schunk: Алгоритм Horn–Schunck опирается на предположение о том, что на всем изображении оптический поток будет достаточно гладким.
Вводятся две функции, характеризующие ошибку:
Первая, характеризирующая несовпадение интенсивностей исходного и якобы смещенного пикселей,

$$E_d(\boldsymbol{x},\boldsymbol{u}) = E_d(I_0(\boldsymbol{x}) - I_1(\boldsymbol{x}+\boldsymbol{u})) = |I_0(\boldsymbol{x}) - I_1(\boldsymbol{x}+\boldsymbol{u})|^2 = |\boldsymbol{u}\cdot\nabla I(\boldsymbol{x},t) + \frac{\partial I(\boldsymbol{x},t)}{\partial t}|^2, \tag{11}$$

и вторая, характеризирующая добавочное требование на отсутствие резкого изменения скоростей/сдвигов[5].

$$E_s(\boldsymbol{u}) = E_s(\boldsymbol{u}, \nabla\boldsymbol{u}) = |\nabla\boldsymbol{u}|^2, \tag{12}$$

Минимизируется некоторый аналог энергии:

$$E = \int_{\boldsymbol{x}\in\mathbb{I}} E_d + \lambda E_s d\boldsymbol{x} = \int_{\boldsymbol{x}\in\mathbb{I}} |\boldsymbol{u}\cdot\nabla I(\boldsymbol{x},t) + \frac{\partial I(\boldsymbol{x},t)}{\partial t}|^2 + \lambda|\nabla\boldsymbol{u}(\boldsymbol{x})|^2 d\boldsymbol{x}, \tag{13}$$

где λ - неотрицательный весовой коэффициент.
Заметим, что здесь можем воспользоваться уравнением Эйлера-Лагранжа [15]. Запишем:

$$\begin{aligned} I_x^2 u + I_x I_y v &= \lambda\nabla^2 u - I_x I_t \\ I_x I_y u + I_y^2 v &= \lambda\nabla^2 v - I_y I_t, \end{aligned} \tag{14}$$

Откуда, приблизив лапласиан взвешенной суммой и выписав такие уравнения для всех точек изображения, найдем $\boldsymbol{u}(\boldsymbol{x})$.

[5] Мы помним, что условились считать сдвиг между кадрами и скорость одним и тем же.

TV-L1 (Total Variation in L1): Методы, основанные на методе наименьших квадратов работают хорошо, когда ошибка в сохранении интенсивности точки

$$e(x) = \boldsymbol{u}\nabla I(\boldsymbol{x}, t) + I_t(\boldsymbol{x}, t)$$

распределена примерно как $N(o, \sigma^2)$[5], что выполняется редко, т.к. интенсивность объектов меняется, например, вместе с тенями, зависящими от времени, мерцаниями освещения, изменением наклона поверхностей, появления бликов и проч. Они непригодны для случаев, когда $e(x)$ хорошо описывается распределениями с т.н. «тяжелыми хвостами», не устойчивы к выбросам [5]
В основе алгоритма DTV-L1 заложена та же идея, что и в алгоритм Horn-Schunk, однако с подынтегральными функциями в метрике L1, т.е. с

$$\begin{aligned} &E_d(\boldsymbol{u} \cdot \nabla I(\boldsymbol{x}, t) + \frac{\partial I(\boldsymbol{x}, t)}{\partial t}) = |\boldsymbol{u} \cdot \nabla I(\boldsymbol{x}, t) + \frac{\partial I(\boldsymbol{x}, t)}{\partial t}| \\ &E_s(\boldsymbol{u}, \nabla \boldsymbol{u}) = |\nabla \boldsymbol{u}| \end{aligned} \tag{15}$$

С применением функций такого вида связаны некоторые математические сложности, а именно не дифференцируемость модуля в 0 и, как следствие, неприменимость формулы Эйлера-Лагранжа для вариации функционала $\frac{\delta E}{\delta x}$.
Для решения данной проблемы обычно модифицируют подынтегральную функцию, приводя её к виду, удобному для оптимизации (см. [13]). Заметим, что существует множество модификаций данного алгоритма (см.[1]).

MDPOF (Motion Detail Preserving Optical Flow Estimation): Так, в одном алгоритме[12], использующем минимизацию функционала "энергии" в метрике L1, вводится дополнительное бинарное поле $\gamma(\boldsymbol{x}) \in \{0, 1\}, \boldsymbol{x} \in \mathbb{I}$, характеризующее заслонение большого объекта малым:

$$E_d(\boldsymbol{x}, \boldsymbol{u}, \gamma) = \gamma|I_1(\boldsymbol{x} + \boldsymbol{u}) - I_0(\boldsymbol{x})| + (1 - \gamma)|\nabla I_1(\boldsymbol{x} + \boldsymbol{u}) - \nabla I_0(\boldsymbol{x})| \tag{16}$$

$$E_s(\boldsymbol{x}, \boldsymbol{u}) = \omega(\boldsymbol{x})|\nabla \boldsymbol{u}|, \tag{17}$$

где $\omega(\boldsymbol{x}) = exp(-|\nabla I_0(\boldsymbol{x})|^\kappa)$, $\kappa \sim 0.8$

$$E(\boldsymbol{u}) = \int_{\boldsymbol{x} \in \Omega} E_d(\boldsymbol{x}, \boldsymbol{u}, \gamma(\boldsymbol{x})) + \lambda E_s(\boldsymbol{x}, \boldsymbol{u}) d\boldsymbol{x} \tag{18}$$

Алгоритм, по которому вычисляется бинарное поле $\gamma(\boldsymbol{x})$, можно прочитать в [12]. Стоит заметить, что это улучшение нацелено решить, наверное, самую главную проблему предыдущих методов - корректную обработку краев объектов, т.е. мест, где $\nabla \boldsymbol{u}(\boldsymbol{x})$ терпит разрыв.

4.2 Вычисление собственного движения

В данном пункте будет рассмотрена задача вычисления собственного движения по известным скоростям некоторого множества точек изображения. Главная проблема в вычислении собственного движения - разделить переносное и вращательные движения. Напомним формулу (4), связывающую виидимые скорости точек $\boldsymbol{u}$ с трансляцией $\boldsymbol{\nu}$ и поворотом $\boldsymbol{\omega}$:

$$\boldsymbol{u}(x,y) = A(x,y)\frac{\boldsymbol{\nu}}{Z(x,y)} + B(x,y)\boldsymbol{\omega}$$

Базируясь на довольно подробной работе [10], мы выбрали два алгоритма для наших исследований: Kanatani-B [9] и Zhang&Tomasi [14].

Kanatani: Приведем очень кратко идею алгоритма Kanatani [9]:
Введем единичную сферу с центром в оптическом центре камеры.

Тогда, движение произвольной точки трехмерного мира можно описать с помощью (см. рис 2): радиус вектора $\boldsymbol{m}$, направленного на данную точку, скалярной глубины $r(\boldsymbol{m})$ и видимой скорости $\dot{\boldsymbol{m}}$, направленной по касательной к поверхности сферы (т.е. перпендикулярно $\boldsymbol{m}$) Движение камеры с угловой скоростью $\boldsymbol{\omega}$ и поступательной скоростью $\boldsymbol{\nu}$ порождает видимое движение $\dot{\boldsymbol{m}}(\boldsymbol{m})$ точки трехмерного мира[6]:

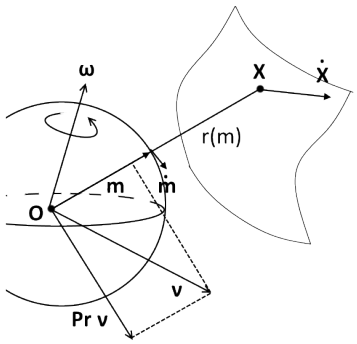


Рис. 2

$$\dot{\boldsymbol{m}}(m,y) = -[\boldsymbol{\omega}\times\boldsymbol{m}] - \frac{\boldsymbol{Pr_s\nu}}{r(\boldsymbol{m})} \qquad (19)$$

Тогда, цель алгоритма Kanatani, по известным $\dot{\boldsymbol{m}}(\boldsymbol{m})$ найти такие $\boldsymbol{\omega},\boldsymbol{\nu}$ и $r(\boldsymbol{m})$, что:

$$||\dot{\boldsymbol{m}}(m,y) + [\boldsymbol{\omega}\times\boldsymbol{m}] + \frac{\boldsymbol{Pr_s\nu}}{r(\boldsymbol{m})}||^2 \to \min_{\omega,\nu,r(\boldsymbol{m})} \qquad (20)$$

Дифференцируя по $\frac{1}{r(\boldsymbol{m})}$ и выражая $r(\boldsymbol{m})$, получим (учтя, что $|\dot{\boldsymbol{m}}| = 1$):

$$r(\boldsymbol{m}) = -\frac{\boldsymbol{Pr_s\nu}}{\dot{\boldsymbol{m}} + [\boldsymbol{\omega}\times\boldsymbol{m}]} = -\frac{\boldsymbol{\nu} - \boldsymbol{nn^T\nu}}{\dot{\boldsymbol{m}} + [\boldsymbol{\omega}\times\boldsymbol{m}]} = -\frac{\boldsymbol{\nu^T\nu} - \boldsymbol{\nu^T nn^T\nu}}{\boldsymbol{\nu^T}\dot{\boldsymbol{m}} + \boldsymbol{\nu^T}[\boldsymbol{\omega}\times\boldsymbol{m}]} = \frac{1-(\boldsymbol{n},\boldsymbol{\nu})^2}{(\boldsymbol{m},\boldsymbol{\omega},\boldsymbol{\nu}) - (\boldsymbol{\nu},\dot{\boldsymbol{m}})} \qquad (21)$$

Осталось найти $\boldsymbol{\omega}$ и $\boldsymbol{\nu}$.
Введем симметричную матрицу K:

$$K = (\boldsymbol{\omega},\boldsymbol{\nu})I - 0.5(\boldsymbol{\omega\nu^T} + \boldsymbol{\nu\omega^T}) \qquad (22)$$

[6] здесь используется приближение участка сферы частью плоскости ввиду малости смещения

Тогда, легко видеть, что

$$(\boldsymbol{m}, K\boldsymbol{m}) = (\boldsymbol{\omega}, Pr_s\boldsymbol{\nu}) = (\boldsymbol{\nu}, Pr_s\boldsymbol{\omega}) \tag{23}$$

$$(\boldsymbol{\nu}, K\boldsymbol{\nu}) = 0 \tag{24}$$

Матрица K имеет замечательный и отнюдь не очевидный вид, который следует из (22)[7]:

$$K = 0.5(tr(K))(I - \boldsymbol{\nu}\boldsymbol{\nu}^{\boldsymbol{T}}) + K\boldsymbol{\nu}\boldsymbol{\nu}^{\boldsymbol{T}} + \boldsymbol{\nu}\boldsymbol{\nu}^{\boldsymbol{T}}K \tag{25}$$

Введем т.н. twisted flow $\dot{\boldsymbol{m}}^* = [\boldsymbol{m} \times \dot{\boldsymbol{m}}]$:

$$\dot{\boldsymbol{m}}^* = [\boldsymbol{m} \times \dot{\boldsymbol{m}}] = -[\boldsymbol{m} \times [\boldsymbol{\omega} \times \boldsymbol{m}]] - \frac{[\boldsymbol{m} \times \boldsymbol{Pr_s\nu}]}{r(\boldsymbol{m})} = -\boldsymbol{Pr_s\omega} - \frac{[\boldsymbol{m} \times \boldsymbol{\nu}]}{r(\boldsymbol{m})} \tag{26}$$

Т.к. $r(\boldsymbol{m})$ - скаляр, то вектор $\dot{\boldsymbol{m}}^* + \boldsymbol{Pr_s\omega}$ должен быть параллелен $[\boldsymbol{m} \times \boldsymbol{\nu}]$, те

$$[(\dot{\boldsymbol{m}}^* + \boldsymbol{Pr_s\omega}) \times [\boldsymbol{m} \times \boldsymbol{\nu}]] = 0 \tag{27}$$

учитывая, что $(\boldsymbol{m}, \dot{\boldsymbol{m}}^*) = 0$ и $(\boldsymbol{Pr_s\omega}, \boldsymbol{\nu}) = 0$ условие 27 можно записать как:

$$[(\dot{\boldsymbol{m}}^*, \boldsymbol{\nu}) + (\boldsymbol{Pr_s\omega}, \nu)]\boldsymbol{m} = 0 \tag{28}$$

Таким образом, непосредственно решается задача

$$\boxed{J = \int_\Omega [(\dot{\boldsymbol{m}}^*, \boldsymbol{\nu}) + (\boldsymbol{Pr_s\omega}, \nu)]^2 d\Omega(\boldsymbol{m}) \to min} \tag{29}$$

J - квадратичная форма с девятью неизвестными параметрами (essential parameters) (6 у K + 3 у $\boldsymbol{\nu}$). Ω - телесный угол.
Ее минимизация является хорошо поставленной задачей, приведем результаты:
$\boldsymbol{\omega}$ можно выразить через необходимые(essential) параметры $\{K, \boldsymbol{\nu}\}$:

$$\omega = 0.5[tr(K) + 3(\nu, K\nu)]\nu - 2K\nu \tag{30}$$

Обозначим

$$\begin{aligned} L_{ij} &= \int_\Omega \dot{m}_i^* \dot{m}_j^* d\Omega(\boldsymbol{m}) \\ M_{ijk} &= \int_\Omega \dot{m}_i^* m_j m_k d\Omega(\boldsymbol{m}) \\ N_{ijkl} &= \int_\Omega m_i m_j m_k m_l d\Omega(\boldsymbol{m}) \\ N^{-1}_{ijkl} \quad &: \quad \sum_{k,l=1}^{3} N^{-1}_{ijkl} N_{klmn} = \delta_{im}\delta_{jn} \end{aligned} \tag{31}$$

[7] Покомпонентной проверкой можно убедиться, что это так

Где m_i - компоненты векторов, а δ_{ij} - символ Кронекера. Тогда, ν - собственный вектор матрицы $A = (A_{ij})$, соответствующий наименьшему собственному значению (см. [9], Theorem 3), где компоненты матрицы A есть:

$$A_{ij} = L_{ij} - \sum_{k,l,m,n=1}^{3} M_{ikl} N^{-1}_{klmn} M_{jmn} \tag{32}$$

4.3 Голосование на случайных выборках(RANSAC)

Из-за наличия большого количества выбросов, непосредственный подсчет компонент матрицы R не приведет к точному результату. Вполне стандартный метод для таких случаев - нахождение нескольких решений и применение голосования на случайных выборках(RANSAC - RANdom SAmple Consensus). Или, более подробно:

Из множества точек, для которых посчитан оптический поток, мы случайно выберем чуть больше, чем нужно для нахождения собственного движения. Например, 15 точек. По ним определим параметры матрицы поворота R[8] Пусть это r^k_{ij}, где k - номер случайной выборки точек. Тогда, получив K значений r^k_{ij}, а также задавшись некоторым порогом ϵ, мы будем считать инлайерами, или удовлетворяющими гипотезе $\hat{r_{ij}}$, те значения, которые отстоят от $\hat{r_{ij}}$ не более, чем на ϵ. Остальные считаем аутлайерами или выбросами. Тогда та гипотеза, для которой число инлайеров максимально, и будет избрана данным алгоритмом (для примера см. рис. 3) [4].

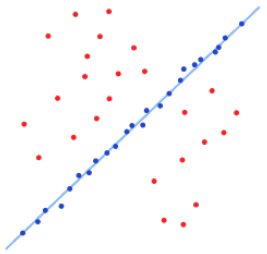


Рис. 3: Пример работы RANSAC

5 Описание метода

[8] Матрица R получается, если мы захотим посчитать, как преобразуется координата точки трехмерного неподвижного мира ($\boldsymbol{X_0} \to \boldsymbol{X_1}$) за единичное время при повороте камеры с угловой скоростью $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)$ и поступательной $\boldsymbol{\nu} = (\nu_x, \nu_y, \nu_z)$. Согласно (2), запишем:

$$\boldsymbol{X_1} = \dot{\boldsymbol{X}} \cdot 1 + \boldsymbol{X_0} = -[\boldsymbol{\omega} \times \boldsymbol{X_0}] - \boldsymbol{\nu} = R_1 0 \boldsymbol{X_0} - \boldsymbol{\nu}$$

Тогда, матрица R:

$$R = \begin{bmatrix} 1 & \omega_z & -\omega_y \\ -\omega_z & 1 & \omega_x \\ \omega_y & -\omega_x & 1 \end{bmatrix}$$

В подпункте, посвященной декомпозиции, мы показали, что рассматриваемая задача может быть естественным образом декомпозирована на три подзадачи, которые ценны и сами по себе:

1. Вычисление оптического потока.
2. Вычисление собственного движения наблюдателя по множеству известных скоростей точек.
3. Голосование на случайных выборках - выбор решения.

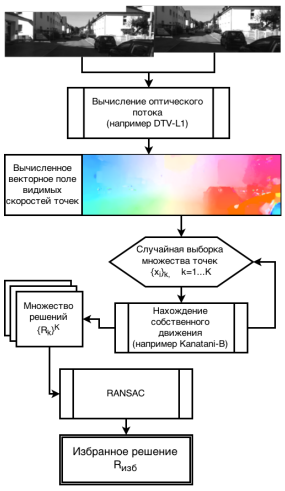


Рис. 4: Блок-схема метода

В совокупности они должны решать заглавную задачу. Мы объединили их в один метод, блок-схема которого представлена на рис. 4. Мы реализовали этот метод как программный комплекс на языках MATLAB и C++. Приведем описания интерфейсов реализованного комплекса. На первом шаге изображения поступают из внешнего мира на вход программы, находящей оптический поток. Для проведения экспериментов использовалась последовательность черно-белых кадров с исправленной дисторсией и известным движением. Здесь были использованы как собственные программы, написанные на C++ с использованием OpenCV (например, DTV-L1), так и программы мировых ученых, находящиеся в открытом доступе. Унифицированный вывод найденного оптического потока в принятом мировым сообществом формате файлов (.flo), позволяет исследовать множество алгоритмов, не прибегая к написанию собственных их реализаций.

Результаты работы алгоритма вычисления оптического потока обрабатываются написанной нами программой на языке MATLAB, векторное поле скоростей точек интерпретируется в удобный для дальнейшей работе вид. Далее, из множества всех точек изображения выбираются (случайным образом) подмножества небольшого размера, которые подаются на вход функции, реализующей следующий шаг метода - вычисление собственного движения.

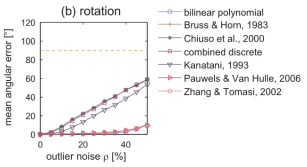


Рис. 5: График зависимости ошибки определения вращения от количества выбросов из [10]

Полученные матрицы $\{R_k\}_{k=1}^{K}$, которые можно интерпретировать как гипотезы о движении наблюдателя, повергаются процедуре голосования на случайных выборках (RANSAC) (визуализацию этого шага можно увидеть чуть ниже на рис. 6).

В конечном итоге, мы получим решение $\hat{R}$. Метод повторяется на следующей паре кадров. Из полученных на разных шагах R_{12} и R_{23} можно вычислить результирующую $R_{13} = R_{23} \cdot R_{12}$. Таким образом, мы можем вычислить

ориентацию камеры относительно ее начального положения. Анализируя алгоритмы вычисления оптического потока, понятно, что MDPOF должен находить оптический поток не хуже, чем DTV-L1, т.к. он использует те же предположения плюс обработку наложения объектов. Это подтверждается результатами их работы на датасете Middlebury и KITTI/flow [1,7]. Например, в первом он стоит на треть таблицы выше.
С другой стороны, опираясь на [10], видно, что, в свою очередь, различные алгоритмы вычисления собственного движения имеют, например, неодинаковую устойчивость к количеству выбросов и уровню нормального шума (см. рис. 5).

6 Численные результаты

а) б)

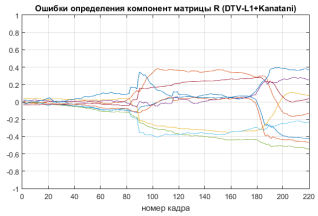
Рис. 6: Визуализация RANSAC для одной и той же пары кадров - значение r_{ij}^k в зависимости от номера выборки k. Результат работы показан черной линией. Слева для DTV-L1, справа - для MDPOF.

Описанный метод был опробован на тестовой последовательности кадров, снятых с реальной камеры, закрепленной на автомобиле. Для данной последовательности достаточно точно определено собственное движение (включая текущее направление камеры на каждом кадре относительно исходного), которое можно принять за эталонное (ground truth) (см рис. 7).

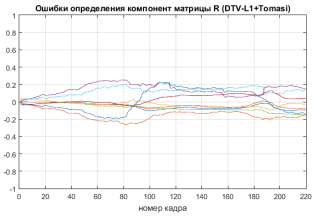
а) б)

Рис. 7: Эталонные элементы матрицы R (а) и угол поворота вокруг вертикальной оси (рад) (б)

Т.к. общий характер движения понятен (длинные прямые + 2 поворота), то делее приведем ошибки работы метода в зависимости от используемых алгоритмов (рис. 8 и рис. 9). Результаты:

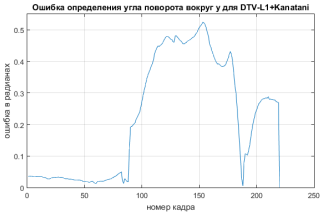


а)



б)

Рис. 8: Ошибки в вычислении: элементов матрицы R с помощью DTV-L1+Kanatani(а) и с помощью DTV-L1+Tomasi(б)



а)

б)

Рис. 9: Ошибки в вычислении угла поворота вокруг вертикальной оси (рад) с помощью DTV-L1+Kanatani(а) и с помощью DTV-L1+Tomasi(б)

7 Заключение

Основные результаты проделанной работы заключаются в следующем:

1. На основе анализа научных трудов, а также на основе собственного исследования, был построен метод вычисления собственного движения наблюдателя с помощью алгоритмов вычисления плотного оптического потока. Использование последних в исследуемой задаче является новым подходом.
2. Построенный метод был реализован в виде программного комплекса на языках MATLAB и C++. Он позволяет производить количественное сравнение различных реализаций алгоритмов вычисления оптического потока в данной задаче, использует унифицированные интерфейсы.

3. Было проведено количественное сравнение двух алгоритмов вычисления оптического потока в сочетании с двумя алгоритмами вычисления собственного движения на тестовой последовательности из датасета KITTI/odometry. Данная тестовая последовательность представляет собой кадры, снятые реальной камерой, закрепленной на автомобиле.
4. Лучший результат, достигнутый при сравнении, соответствует угловой ошибке 0.02 градуса/метр.

Список литературы

1. S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black, R. Szeliski.: A Database and Evaluation Methodology for Optical Flow. International Journal of Computer Vision, 1–31 (2011):
2. A. R. Bruss, B.K.P. Horn.: Passive Navigation. Computer Vision, Graphics, and Image Processing 21, 3-20 (1983)
3. G. Farneback.: Polynomial Expansion for Orientation and Motion Estimation. Link¨oping Studies in Science and Technology. Dissertations 790
4. M.A. Fisher, R.C. Bolles.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Austomated Cartography. Comm. of the ACM 24 (6), 381-395, (1981)
5. D.J. Fleet, Y. Weiss.: Optical Flow Estimation. Mathematical Models in Computer Vision: The Handbook, 239-258, (2005)
6. B. Galvin, B. McCane, K. Novins, D. Mason and S. Mills.: Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms. British Machine Vision Conference, 195-204 (1998)
7. A. Geiger, P. Lenz, C. Stiller, R. Urtasun.: Vision meets Robotics: The KITTI Dataset. International Journal of Robotics Research (IJRR), http://www.cvlibs.net/datasets/kitti/ (2013)
8. B.K.P. Horn, B.G. Schunck.: Determining Optical Flow. Artificial Intelligence, 185-202 (1980)
9. K. Kanatani.: 3-D Interpretation of Optical Flow by Renormalization. International Journal of Computer Vision, 267-282 (1993)
10. F. Raudies, H. Neumann.: A review and evaluation of methods estimating ego-motion. Computer Vision and Image Understanding 116, 606–633 (2012)
11. S. Song, M. Chandraker, C. Guest.: On the Second Order Statistics of Essential Matrix Elements. 36th German Conference, GCPR 2014, Münster, Germany, September 2-5, 2014, Proceedings, 547-557 (2014)
12. L. Xu, J. Jia.: Motion Detail Preserving Optical Flow Estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 34(9), 1744-1757, (2012)
13. C. Zach, T. Pock, H. Bischof..: A Duality Based Approach for Realtime TV-L 1 Optical Flow. Volume 4713 of Lecture Notes in Computer Science, chapter 22, 214–223, (2007)
14. T. Zhang, C. Tomasi.: On the consistency of instantaneous rigid motion estimation, International Journal of Computer Vision 51–79 (2002)
15. Л.Э. Эльсгольц: Дифференциальные уравнения и вариационное исчисление. Наука, (1969)