Гироскоп на основе монокулярной камеры

Кудеров П.В.

ИУ9-121

**Оглавление**

[Введение 3](#_Toc420044688)

[1 Аналитическая часть 4](#_Toc420044689)

[1.1 4](#_Toc420044690)

[2 Научно-исследовательская часть 5](#_Toc420044691)

[3 Проектно-конструкторская часть 6](#_Toc420044692)

[4 Технологическая часть 7](#_Toc420044693)

[4.1 Методология разработки и используемые средства 7](#_Toc420044694)

[4.2 Руководство пользователя 7](#_Toc420044695)

[4.2.1 Приложение Gyrocam 7](#_Toc420044696)

[5 Организационно-экономическая часть 10](#_Toc420044697)

[5.1 Введение 10](#_Toc420044698)

[5.2 Организация и планирование процесса разработки программы 10](#_Toc420044699)

[5.2.1 Техническое задание 10](#_Toc420044700)

[Заключение 11](#_Toc420044701)

[Приложение 1. Диаграмма Ганта выполненных работ 13](#_Toc420044702)

# Введение

Последние десятилетия наблюдается высокий уровень интереса к автономной самоуправляемой технике, и с каждым годом он продолжает расти. Робототехническим системам можно найти применение в условиях военных конфликтов, экстремальных для человека или несопоставимых с его габаритами, в качестве транспортных средств для пассажиро- и грузоперевозок, для проведения масштабных и/или простых шаблонных работ и так далее. Одной из наиболее проблемных подсистем самоуправляемой техники является система навигации, и, в частности, подсистема позиционирования.

[*image: nav systems relation*]

Среди популярных применяемых в автономной робототехнике решений – использование инерциальных систем навигации (ИНС). Подобные системы содержат акселерометры для определения параметров линейного ускорения, а также гироскоп (или акселерометры, измеряющие центробежное ускорение) для определения углов поворота и наклона. На основе данных этих датчиков производится последующее вычисление вектора скорости и координат объекта. Преимущества методов инерциальной навигации состоят в автономности, помехозащищенности и возможности полной автоматизации всех процессов навигации. Основная же проблема – наличие дрейфа, то есть накопление ошибки со временем работы. Различными техниками можно уменьшить величину ошибки, но не избавиться вовсе.

Не имеют дрейфа глобальные (спутниковые) системы навигации (GPS, GLONASS, GALILEO и другие их аналоги). Для существующих систем данного класса характерна погрешность позиционирования порядка 5-15 метров (***данные википедии***). Дополнительные проблемы связаны с диапазоном рабочей частоты сигнала – уровень приема сигнала от спутников может серьезно ухудшиться под плотной листвой деревьев, из-за очень большой облачности, в условиях плотной городской застройки. Практически невозможно определить свое точное местонахождение внутри помещений, в тоннеле, причем даже профессиональными геодезическими приемниками (***практически*** ***цитата из ненаучной статьи***). Но и в благоприятных условиях точности в несколько метров для определенного круга задач может быть недостаточно. В таких случаях глобальные системы навигации используются лишь для локализации нахождения объекта, а уточнение координат объекта ведется с помощью других систем.

Среди систем локального позиционирования распространено использование технологий построения трехмерной карты окружающей среды на основе данных эхолокации инфракрасными или ультразвуковыми датчиками.

Набирает популярность использование методов компьютерного зрения. Видеодатчики имеют маленький размер, энергопотребление, цену, поэтому есть интерес использовать их в системах позиционирования и навигации. Для тестирования методов можно использовать даже камеры мобильных телефонов. Системы позиционирования на основе анализа изображений не имеют дрейфа, при этом может достигаться довольно высокая точность, сравнимая с ИНС потребительского класса. Большинство из предлагаемых методов основано на обнаружении базисных элементов изображения и слежения за ними в потоке изображений.

Среди распространенных – метод одновременной навигации и картирования (simultaneous localization and mapping, SLAM) , а также метод определения структуры объекта в процессе движения (structure from motion, SfM). Оба этих метода в той или иной степени пытаются построить 3D модель окружающей среды, относительно которой происходит движение камеры.

[*image: slam & sfm examples*]

\* SLAM – используются Particle filters / Sequential Monte Carlo(SMC) или расширенный фильтр Калмана, bundle adjustment (корректировка узлов?)

\* SfM (Structure from motion) - видимо, то же, что и 3D reconstruction from 2D images, распространен алгоритм Scale-invariant feature transform (SIFT), другой распространенный алгоритм - Speeded Up Robust Features (SURF).

Одна из подзадач, возникающих при построении системы позиционирования с использованием алгоритмов компьютерного зрения – определение углов наклона и поворота камеры, то есть создание аналога гироскопа.

В данной работе реализован метод, предложенный Вилле Хуттуненом и Робертом Пише [**Link1**], который заключается в определения трехмерной ориентации монокулярной камеры с использованием точек схождения перспективы (ТСП), обнаруженных на изображениях.

[*image: mahattan world example, indoor, city outdoor, non-city outdoor*]

Данный метод подходит для использования внутри помещений, в условиях городской застройки, а также мест, где под влиянием деятельности человека четко прослеживаются правильные геометрические примитивы (man-made world, Manhattan world). В таких условиях изобилие линейных объектов правильной геометрической формы позволяет достигнуть высокой точности.

Плюсы конкретно данного метода (***ниже почти дословная цитата из******Link1***):

* в условиях "Manhattan World" изобилие линейных объектов правильной геометрической формы позволяет достигнуть высокой точности
* число интересующих нас ТСП ограничено <= 3
* ТСП не зависят от положения камеры - только от ее ориентации
* робастность относительно случайных нестационарных объектов, попадающих в кадр (люди, транспортные средства и т.п.)

Методы на основе определения ТСП позволяют решать задачи:

* навигации автономных транспортных средств, роботов
* 3D реконструкции окружающей среды
* калибровки камер и коррекции изображений

Метод был протестирован на наборе изображений YorkUrbanDb [**Link3**], созданного на базе York University Centre for Vision Research, а также на случайных изображениях, взятых из интернета или сделанных вручную.

[*image: our result image*]

Набор YorkUrbanDb состоит из 47 изображений внутри помещений и 55 изображений городских сцен с вычисленными значениями ТСП.

Существует целый ряд работ, посвященных вычислению ТСП на снимках городских сцен и внутри помещений, то есть в условиях так называемого «Manhattan World» [**Link2**]..

Промежуточная цель работы состоит в исследовании применимости метода Хуттунена и Пише в условиях съемок внутренних помещений и городских пейзажей, обнаружение слабых мест и ознакомление с аналогичными работами по данной тематике, предложение вариантов улучшения метода, нахождение удобного способа тестирования (?).

Конечная цель – создание библиотеки / приложения, которое позволяет эффективно и с точностью, сравнимой с гироскопами потребительского класса, вычислять углы наклона и поворота камеры по входному потоку изображений в условиях «Manhattan World assumption» в реальном времени..

# Аналитическая часть

## 

# Научно-исследовательская часть

Поставленную задачу можно разделить на следующие этапы:

Используемый в данной работе метод нахождения углов поворота и наклона состоит из следующих шагов:

* выделение сегментов линий на изображении
* выделение наилучших кластеров СЛ, сходящихся в одной точке (или бесконечности), используя алгоритм RANSAC
* (не работает) приведение координат сегментов линий в нормализованные координаты путем домножения слева на обратную матрицу калибровки
* вычисление вектора направления ТСП на основе полученных кластеров СЛ путем приближенного решения задачи минимизации Ax = 0 методом SVD декомпозиции матрицы A
* ортогонализация матрицы, составленной из векторов направлений ТСП
* (нет) получение углов (крен, тангаж, рыскание) из матрицы поворота
* (нет) связывание матриц поворота и кормление их расширенному фильтру Калмана

Входные данные алгоритма - изображение с условием "Manhattan World".

Первым этапом является выделение сегментов линий на изображении. Для решения данной задачи в оригинальной статье предлагается использовать метод Джиои и ... (Link4). (проверить, что написано во введение самой статьи Джиои) Это довольно новый алгоритм, отличающийся высокой скоростью за счет линейной зависимости сложности от размеров изображения по сравнению с другими алгоритмами, основанных на анализе связанных компонент градиента изображения. По быстродействию он уступает алгоритмам, основанным на преобразовании Хафа (Hough), но позволяет достичь более высокого качества.

В своем приложении я использовал реализацию алгоритма Джиои в библиотеке opencv версии 3.0.0. Ему отвечает класс cv::LineSegmentDetector (LSD), который принимает на вход изображение в градациях серого (в opencv тип CV\_8UC1), и различные параметры настройки алгоритма. В своем приложении в качестве параметров настройки я использовал рекомендованные по умолчанию.

Результатом работы модуля LSD является список найденных сегментов линий. Для каждого сегмента дается следующая информация:

\* координаты концов в пикселах в виде четверки целых чисел (тип данных opencv Vec4i)

\* ширина линии

\* точность, с которой он найден

\* число ложных срабатываний (number of false alarms) в области сегмента линии в виде логарифмической шкалы качества детектирования

Получив результат работы детектора сегментов линий, я провожу фильтрацию сегментов по длине, отбрасывая слишком короткие меньше 20 символов. Во-первых, это позволяет значительно ускорить работу на последующих этапах. Во-вторых, мною было замечено, что короткие сегменты чаще относятся к ошибочным направлениям (не к искомым ТСП). Те же, что относятся к искомым ТСП за счет своей длины ухудшают точность вычислений, т.к. погрешность детектора в обнаружении одного из концов отрезка в 1 пиксел ведет к достаточно большой итоговой угловой погрешности.

Далее я провожу подготовку структур данных к последующим этапам работы алгоритма, создавая на основе каждого отрезка, представленного четверкой целых чисел, объект структуры LineSegment, вычисляю и сохраняю следующие поля:

\* оригинальную четверку координат концов, полученных в виде объекта структуры Vec4i

\* координаты точек концов отрезка в нормализованных координатах и удобном формате типа cv::Point3d

\* середину отрезка в нормализованных координатах

\* уравнение линии, которую задает сегмент, в нормализованных координатах. Уравнение вычисляется как векторное произведение точек концов отрезка

Далее, я провожу кластеризацию сегментов линий, используя адаптивный алгоритм RANSAC. На каждом прогоне алгоритма вычисляется самый большой оставшийся кластер сегментов, линии которых пересекаются в одной точке с некоторой допустимой погрешностью. Полученный кластер объявляется соответствующим некоторой ТСП. Всего производится 3 запуска алгоритма.

Алгоритм состоит из последовательности итераций, состоящих из следующих шагов:

\* случайным образом выбирается пара сегментов линий

\* вычисляется точка их пересечения, которая объявляется потенциальной ТСП

\* далее проводится каждого сегмента линии на предмет принадлежности его потенциальной ТСП путем вычисления функция расстояния до ТСП по формуле:

Point3d l = lineThroughPoints(segment.middle, vp);

double d = incidence(l, segment.from) / norm12(l);

return abs(d) <= distanceEpsilon

&& abs(asin(d / norm(segment.middle - segment.from))) <= angleEpsilon;

, где

static const double distanceEpsilon = 2; // 2 pixel

static const double angleEpsilon = 0.04; // ~1% pi or ~2 degree

Таким образом для имеющейся потенциальной ТСП определяется множество соответствующих ей сегментов. Назовем содержащиеся в нем сегменты внутренними, а все остальные - внешними по отношению к данной vp. Обозначим также их отношение как r. Чем выше число r, тем более подходящей считается ТСП.

Число итераций алгоритма определяется адаптивно следующим образом. Определим число итераций k, необходимых, чтобы гарантировать с вероятностью p, что хотя бы одна потенциальная ТСП, случайно выбранная алгоритмом RANSAC, не содержит внешних сегментов линий. Вероятность выбора k ТСП по крайней мере с одним внешним сегментом в каждой из них составляет (1 - r^2)^k (подтвердить). Число итераций, необходимых, чтобы гарантировать вероятность ниже порога 1 - p:

k >= log(1 - p) / log (1 - r^2)

Хотя истинное значение r и неизвестно, его можно аппроксимировать снизу наилучшим из найденных за предыдущие итерации r.

После каждого запуска алгоритма RANSAC внутренние сегменты для найденной ТСП удаляются из выборки и в последующих запусках не участвуют. В итоге после 3х последовательных запусков алгоритма мы имеем 3 кластера сегментов линий, каждому из которых соответствует довольно грубая оценка ТСП.

## Геометрия перспективных изображений

Проективное преобразование и однородные координаты, модель булавочного отверстия, ТСП

Теория:

1. Геометрия ТСП

Теория точек схождения перспективы (ТСП) рассматривается обычно в терминах проективной геометрии, изучающей геометрические свойства, являющихся инвариантными относительно проективных преобразований, а также сами эти преобразования. Проецирование трехмерной сцены на двухмерную плоскость изображения, осуществляемое фото- или видеокамерой, – одно из таких преобразований.

Одной из интересующих нас особенностей проективного преобразования является тот факт, что параллельность прямых не является инвариантом относительно него.

1.1. Проективное преобразование и однородные координаты, модель булавочного отверстия, ТСП

В проективной геометрии точки представлены в однородных координатах. Например, рассмотрим точку в двухмерном евклидовом пространстве (x, y) @ R2. Чтобы представить ее на проективной плоскости, необходимо лишь добавить третью компоненту z, равную 1, т.е. (x, y, 1) @ P2. Более того, для любого ненулевого a (ax, ay, a) = (x, y, 1). В случае нулевого a координаты точки вырождаются в точку (0, 0, 0), которая не включена в P2.

Рассмотрим уравнение прямой на двухмерной евклидовой плоскости:

ax + by + c = 0

Точки (x, y), и только они, удовлетворяющие данному уравнению являются лежащими на данной прямой. Теперь заметим, что уравнение (\*) можно переписать следующим образом:

ax + by + c\*1 = aX + bY + cZ = 0

Таким образом мы определили все точки (x, y, 1) = (X / Z, Y / Z, 1) = (X, Y, Z) проективного пространства, лежащие на прямой. Заметим также, что aX + bY + cZ = l \* p = 0, где l = (a, b, c), p = (X, Y, Z), a \* - скалярное произведение. Во-первых, важным свойством проективной геометрии является то, что уравнение линии задается вектором той же размерности, что и точки. Во-вторых, имеет место весьма красивое и простое выражение связывающее точки и проходящие через них линии.

Не вдаваясь в подробности, запишем это и другие важные уравнения проективной геометрии, которые так или иначе понадобятся нам в работе:

\* проверка принадлежности точки прямой incidence(p1, p2) = p1 \* p2

\* прямая через две точки ltp(p1, p2) = p1 x p2, где x - векторное произведение

\* точка пересечения двух прямых intersection(l1, l2) = l1 x l2

Чтобы вернуться из P2 обратно в R2, достаточно поделить координаты точки на z-координату, то есть (X, Y, Z) = (X / Z, Y / Z, 1) = (x, y, 1) @ P2 ~ (x, y) @ R2. Из данной процедуры сразу видно, что P2 ( R2, так как содержит элементы с z = 0. Эти точки составляют довольно важно подмножество P2 и называются идальными. Еще их называют точками в бесконечности, так как они соответствуют предельным точкам, лежащим бесконечно далеко от начала координат. Несмотря на свой особый вид, данные точки никаким специальным образом не обрабатываются, то есть рассматриваются абсолютно также как и обычные. Все идеальные точки лежат на идеальной прямой или прямой в бесконечности.

Также заметим, что точкам p(X, Y, Z) = (x, y, 1) @ P2 можно поставить в соответствие прямую, проходящую через начало координат и точку p`(x, y, 1) @ R3 с выколотой точкой (0, 0, 0). Таким же образом линия l(a, b, c) на проективной плоскости может быть визуализирована в R3 плоскостью, образованной началом координат и перпендикуляром к l`(a, b, c). Тогда точкам с координатами (x, y, 1) в R3 соответствует плоскость Z = 1, идеальным точкам соответствуют точки на Z = 0, а идеальной прямой сама плоскость Z = 0.

Данная связь между R3 и P2 может быть легко продолжена до связи между P3 (добавлением к точкам R3 четвертой координаты 1) и P2. Такая связь очень хорошо подходит для описания преобразования проецирования трехмерной сцены на двухмерную плоскость изображения.

Также заметим, что точкам p(X, Y, Z) = (x, y, 1) @ P2 можно поставить в соответствие прямую, проходящую через начало координат и точку p`(x, y, 1) @ R3 с выколотой точкой (0, 0, 0). Таким же образом линия l(a, b, c) на проективной плоскости может быть визуализирована в R3 плоскостью, образованной началом координат и перпендикуляром к l`(a, b, c). Тогда точкам с координатами (x, y, 1) в R3 соответствует плоскость Z = 1, идеальным точкам соответствуют точки на Z = 0, а идеальной прямой сама плоскость Z = 0.

Данная связь между R3 и P2 может быть легко продолжена до связи между P3 (добавлением к точкам R3 четвертой координаты 1) и P2. Такая связь очень хорошо подходит для описания преобразования проецирования трехмерной сцены на двухмерную плоскость изображения.

Рассмотрим точку в R3 p(X, Y, Z). Для того, чтобы прибавить к ней вектор t(Tx, Ty, Tz), мы можем воспользоваться следующим матричным выражением:

[p + t] = [ E t] \* [p]

[1 ] [ 0 1] [1]

Похожим образом выражается поворот точки p и умножение каждой из его координат на независимый коэффициент d(dx, dy, dz):

[p'] = [R 0] \* [p]

[1 ] [0 1] [1]

[dx \* X] [dx 0 0 0]

[dy \* Y] = [0 dy 0 0] \* [p]

[dz \* Z] [0 0 dz 0] [1]

[1 ] [0 0 0 1]

Можно заметить, что в данных выражениях точка p @ R3 представлена в нормализованных координатах P3 (X, Y, Z, 1). Также очевидно, что данные выражения остаются верными, если и для точкек вида alpha \* p = (alpha X, alpha Y, alpha Z, alpha):

alpha \* (M \* p) = M \* (alpha \* p).

Интересно рассмотреть как данные преобразования влияют на идеальные точки q(X, Y, Z, 0). Простой подстановкой проверяется, что:

\* перенос на вектор t оставляет идеальную точку на месте

\* поворот действует на идеальную точку абсолютно также, как и на конечную

\* масштабирование на вектор d(dx, dy, dz) действует аналогично действию на конечную точку

Теперь рассмотрим связь между системами координат камеры и мировой системой координат. Пусть в мировой системе координаты камеры представлены точкой t, а матрица поворота R связывает соответствующие оси систем, тогда выражения связи имеет вид:

pw -> R(pw - t) = pc

, где pw - точка в мировой системе, а pc - в координатах системы камеры.

pc = Rpw - Rt

Если же воспользоваться нормализованными координатами, получится следующее:

[pc] = [R -Rt] \* [pw] = [R 0] \* [I -t] \* [pw]

[1 ] [0 1] [1 ] [0 1] [0 1] [ 1]

Матрица M, определенная как:

M = [R 0] \* [I -t]

[0 1] [0 1]

, задает матрицу перехода между системами координат. Матрицы R и t задают внешние (extrinsic) параметры камеры - ориентацию и позицию - в мировых координатах.

Одна из самых простых и обычно используемых моделей для конечной проективной камеры - модель булавочного отверстия или модель камеры обскуры (pinhole camera), в которой точки p(X, Y, Z) @ R3 проецируются на двухмерную плоскость по правилу:

(x, y) = (fX / Z, fY / Z)

или в нормализованных координатах:

(x, y, 1) = (f X/Z, fY / Z) = |при Z != 0| = (X, Y, Z / f) = (fX, fY, Z)

Данное правило может быть записано в виде матрицы проекции:

[fX] [f 0 0 0] [X]

[fY] = [0 f 0 0] = [Y]

[Z ] [0 0 1 0] [Z]

[1]

Очевидно также, что вектор [0 0 0 1]T - является нуль-вектором нуль-пространства данного преобразования.

Работая с реальными камерами и точками на полученных с них изображениях, удобнее иметь дело с координатами, выраженными в пикселах нежели, например, в миллиметрах. Перевод координат требует информации о линейных размерах пиксела (например, в миллиметрах) и координат главной точки (principal point), которой соответствует центр изображения (точка пересечения оптической оси камеры с плоскостью изображения), так как в общем случае она может не совсем точно совпадать с центром матрицы камеры (более того, довольно часто центр координат в пикселах определяется одним из углов изображения).

За такого рода перевод координат отвечает так называемая матрица калибровки камеры K, которая задает внутренние (intrinsic) параметры камеры и предполагается неизменяемой во времени:

K = [ f/mx s px ]

[ 0 f/my py ]

[ 0 0 1 ]

, где

\* f - фокусное расстояние камеры в некоторой единице длины (обычно в мм или дюймах),

\* mx, my - линейные размеры пиксела, выраженные в той же единице длины, что и f. Таким образом f / mx и f / my имеют размерность пикселов (мм / (мм / px) = px),

\* s - коэффициент ассиметрии камеры

\* px, py - координаты главной точки в пикселах.

Большинство цифровых камер на основе приборов с зарядовой связью (ПЗС) имеют квадратные пикселы (mx = my), нулевую ассиметрию (s = 0) и главную точку, расположенную близко к центру изображения.

! возможно стоит сказать про дисторсию, что она есть, но в данной работе мы ей принебрегаем

Теперь мы готовы выразить проективное преобразование, отвечающее отображению трехмерной сцены на двухмерную плоскость изображения:

F: P3 -> P2, F(p) = p' = KR[I | -t] p

Рассмотрим прямую в P3, заданную как X(alpha) = A + alpha \* D, где А - точка на этой прямой, D(d, 0) - направляющий вектор и d @ R3, alpha @ R. Проекция этой прямой:

x(alpha) = P(X(alpha)) = P(A) + alpha \* P(D) = P(A) + alpha \* KR[I3, -t]D = P(A) + alpha \* KRd

## Обнаружение ТСП

**Обнаружение СЛ**

Вкратце преобразование Хо (Хаф, Hough Transform) и анализ связанных компонент ориентаций градиента изображения, возможно про Canny детектор, кратко про LSD (pdf)

Выбрали анализ ... почему?

Lsd - O(img.size()), but slower than Hough

**Кластеризация СЛ**

RANSAC, можно (нужно! но, возможно, не здесь, а дальше в "что можно улучшить") заикнуться про J-Linkage (введение в статье pdf - сравниваются разные способы кластеризации)

RANSAC выбрали, потому что популярное решение.

Расписать функцию расстояния d(vp,l), как в статье, как делаю я.

Расписать как выбирается число итераций алгоритма (вывод формулы)

На данном этапе выработана грубая оценка ТСП

**Уточнение ТСП**

Описать математически текущую ситуацию - пучок линий сходится примерно в одной точке. Какое уравнение связывает их в идеале.

Значит решаем Ax = 0 приближенно. Что делать, если система не переопределена (у меня сейчас просто упадет программа!)?

Какие существуют способы приближенного решения данной задачи? LSQM, SVD, smth else

Почему мы не решаем ее тут, а переходим в нормализованные координаты изображения? Что это такое вообще, зачем? В честь кого это делается? [Cipolla]

Получили нулевые направления, что дальше?

**Определение ориентации по ТСП**

Есть матрица D = [d1, d2, d3]. Если только две приблизительно ортогональны (или 3ей не нашли вообще), 3ю вычисляем на основе их через векторное произведение.

Объясняю, что D скорее всего не ортогональна => можно ортогонализовать. Как? SVD! D' = u \* vt

Приводим матрицу нулевых направлений

Матрица относительного поворота R = D2 D1t

At = A-1, т.к. ортогональная => alpha(R) = alpha(D2) + alpha(D1t) = alpha(D2) - alpha(D1)

# Проектно-конструкторская часть

# Технологическая часть

## Методология разработки и используемые средства

## Руководство пользователя

### Приложение Gyrocam

Результатом работы является Win32 консольное приложение, реализующее алгоритм нахождения точек схождения перспективы (ТСП) на изображении, а также векторов единичных направлений, соответствующих им.

Для использования данного приложения требуется установленная на компьютере пользователя библиотека компьютерного зрения opencv версии 3.0.0. Описание процесса установки и необходимые файлы можно найти по адресу [http://docs.opencv.org/3.0‑alpha/doc/tutorials/introduction/windows\_install/windows\_install.html](http://docs.opencv.org/3.0alpha/doc/tutorials/introduction/windows_install/windows_install.html).

Приложение запускается со следующими аргументами:

1. строка абсолютного или относительго пути к файлу изображения, которое следует обработать. Поддерживается такие форматы как bmp, jpg, jpeg,tiff, png.
2. строка абсолютного или относительного пути выходного файла изображения. Формат выходного изображения будет выбран на основе расширения в имени файла данного пути. Поддерживаемые форматы те же, что и во входном аргументе.

Для изображения, заданного входным аргументом, приложение рассчитывает единичные векторы ТСП и выводит окно. В данном окне отображается само исходное изображение с наложенными поверх него прямыми линиями разных цветов. Пример окна приложения показан на рисунке (**Рисунок 1**).

Сначала поверх изображения накладываются линии, соответствующие трем наилучшим (с вероятностью >= 95%) кластерам сегментов линий, сходящихся в какой-либо одной точке. Для них используются темные варианты трех цветов – темно-синий, темно-зеленый и темно-красный цвета в порядке выделения каждого кластера.

Далее поверх полученного изображения накладываются линии, соответствующие кластерам сегментов линий, сходящихся в вычисленных ТСП. Данные линии отображаются яркими вариантами трех цветов – синим, зеленым и красным (для сравнения, например, если темный вариант красного – bgr(0, 0, 127), то яркий – bgr(0, 0, 255)). Кластеры одного цвета, но разной степени яркости в общем случае могут как полностью совпадать, так и частично различаться. Связь между ними следующая – сегменты темного цвета как бы продуцируют некоторую ТСП, а яркого – индуцированы этой же ТСП. В случае полного совпадения кластеров одного цвета – поверх кластера темного цвета будет наложен кластер яркого цвета и он не будет виден.

Выведенное пользователю изображение также сохраняется на диск в выходной файл. Путь, название и расширение файла определяется соответствующим (вторым) входным аргументом командной строки.



Рисунок 1 – Окно с обработанным изображением, поверх которого наложены сегменты линий, соответствующие найденным ТСП. На нем виден случай, когда кластеры темных вариантов цветов полностью покрыты кластерами ярких цветов.

Помимо описанного выше окна с изображением после вычисления ТСП приложение выводит в консоль некоторую служебную информацию от модуля opencv и затем две матрицы 3х3 с числами с плавающей запятой (**Рисунок 2**). Каждая матрица имеет формат как в примере ниже:

[0.99999738, 0.0022501017, 0.00047868371;

0.93076748, 0.36539984, -0.012442469;

-0.10985497, -0.99394757, -0.00042265668]

Границы матрицы задаются квадратными скобками, строки матрицы разделяются точкой с запятой с последующим переводом строки, числа в рамках одной строки разделяются запятыми.

Первой матрице соответствуют единичные векторы найденных ТСП. Каждый вектор представляет собой строку матрицы. Порядок соответствия векторов ТСП наложенным на изображение линиям следующий – синий, зеленый, красный.

Вторая матрица – ближайшая к первой ортонормированная матрица в смысле нормы Фробениуса. Данная матрица является искомой матрицей поворота системы координат камеры.

Описанный выше вывод двух матриц также сохраняется на диск текстовый файл в том же формате. Полный путь данного файла получается прибавлением расширения «.txt» к строке пути второго входного аргумента (пути к выходному изображению). Например, если при запуске приложения вторым аргументом на вход передать относительный путь «../TestSamples/output.jpg», то матрицы будут записаны в файл по следующему пути: «../TestSamples/output.jpg.txt».



Рисунок 2 – Вывод в консоли матриц направлений ТСП (неортогональной и ортогональной, соответственно) на тестовом запуске.

Описанное консольное приложение было протестировано на наборе изображений YorkUrbanDb (The York Urban Line Segment Database). Набор состоит из 47 изображений внутри помещений и 55 изображений городских сцен. Для каждого изображения из базы приведены данные по точкам схождения перспективы, что позволяет оценить уровень точности работы приложения. Описание и сам набор изображений можно найти на странице базы <http://www.elderlab.yorku.ca/YorkUrbanDB/>.

Также может представлять интерес для тестирования набор изображений в составе так называемой Eurasian Cities Database. Описание и сам архив базы для скачивания доступен на <http://graphics.cs.msu.ru/en/research/projects/msr/geometry>.

# Организационно-экономическая часть

## Введение

Разрабатываемое в рамках дипломной работы программное обеспечение является.

Целью данного раздела является расчет трудоемкости, продолжительности разработки программного обеспечения и сметы затрат.

## Организация и планирование процесса разработки программы

### Техническое задание

.

# Заключение

В данной работе были

Библиография

Link1. Huttunen & Piché, 2012

Link2. Coughlan & Yuille, 1999, 2003

Link3. The York Urban Line Segment Database

1. Диаграмма Ганта выполненных работ