Слайд 1:

Добрый день, уважаемые члены экзаменационной комиссии. Моя выпускная квалификационная работа называется “Применение точечных детекторов в алгоритмах обработки изображений ландшафтных сцен”.

В последние 20 лет применение систем технического зрения (СТЗ) в анализе изображений ландшафтов для решения разнообразных задач достигло уровня успешного применения. Среди различных работ в данной области можно выделить работы, связанные с обработкой изображений ландшафтов, полученных как в результате съёмки с воздуха (картографирование местности с помощью летательных аппаратов с бортовыми камерами, Определение компонент движения посадочного модуля марсианских автоматических станций Spirit и Opportunity), так и с бортовой камеры наземного РТК в естественной среде.

В работах по сопоставлению изображений друг с другом и с картами существенным допущением является то, что на изображениях ландшафтов как правило отсутствуют объекты с заранее известной моделью, то есть в основу работы таких систем необходимо поставить точечные признаки. В качестве первых этапов обработки в рассматриваемых системах выполняется алгоритм, состоящий из двух логических частей:

1. детектор точечных признаков

2. метод прослеживания

Важной характеристикой такого алгоритма является количество кадров, на которых можно проследить один признак. В частности, для определения компонент движения посадочных модулей станций Spirit и Opportunity было проанализировано три последовательных кадра.

Слайд 2:

Таким образом, цель данной работы — для каждого из двух типов ландшафтных изображений (наземная и орбитальная съёмки) выбрать алгоритмы, которые удовлетворяли бы требованиям скорости и надёжности: обрабатывать не менее 25 кадров в секунду и стабильно отслеживать признаки не менее чем на трёх последовательных кадрах.

Слайд 3:

Для достижения цели нужно выполнить следующие задачи:

1. выбрать детекторы точечных признаков, соответствующие поставленной цели;
2. Выбрать методы прослеживания;
3. Выбрать вычислители, для которых есть реализации различных детекторов точечных признаков;
4. Оценить работу алгоритмов на видеопоследовательностях, представленных в исходных данных, и вычислителях по количеству обрабатываемых в секунду кадров и надёжности отслеживания признаков;
5. На основании результатов оценки выбрать алгоритмы, удовлетворяющие цели работы.

Слайд 4:

Первой частью искомого алгоритма является детектор точечных признаков. Точечный признак — точка, отличающаяся от соседних точек в некоторой окрестности по определенному критерию. Детектор точечных признаков должен обеспечивать инвариантность нахождения одних и тех же особых точек относительно преобразований изображений. Различные детекторы используют различные техники поиска и имеют свои особенности, достоинства и недостатки.

Далее будут рассмотрены следующие дететкоры: Харриса, Shi-Tomasi, Fast, Star

Слайд 5:

на слайде представлена функция отклика дететкора Харриса и иллюстрация принципа его работы. Точка считатеся угловой, если оба собственных значения матрицы достаточно велики.

Слайд 6:

Детектор ши-томаси – это модификация дететкора углов харриса, где в качестве функции отклика используется минимальное из собственных значений. Опытным путём выяснилось, что такая функция отклика возвращает более устойчивые для прослеживания точки.

Слайд 7:

детектор Star был разработан специально для задач визуальной одометрии и SLAM, так как детекторхаррис а и shi-Tomasi не являются инвариантными относительно масштаба. Star использует пространство масшьабов, и на данном слайде показано, каким образом работа с пространством масштабов была реализована эффективно. Разность гауссиан аппрокисмируется бокс-фильтром, который вычисляется с помощью интегральных изображений.

Слайд 8:

В отличие от предыдущих детекторов, детектор FAST сравнивает яркости точки и ряда точек из её окрестности. Выполняется гораздо быстрее предыдущих, но в силу своей специфики часто возвращает признаки очень кучно даже при высоком пороге и подавлении немаксимумов, что можно видеть на слайде.

Слайд 9:

Дескриптор – вектор, описывающий окрестность особой точки и позволяющий однозначно выделить её среди других точек, используется для сопоставления точечных признаков, принадлежащих разным изображениям. Как и детекторы точек, генераторы дескрипторов реализуют различные подходы и имеют свои сильные и слабые стороны. В данной работе рассматриваются дескрипторы BRIEF и FREAK.

Слайд 10:

Генератор дескрипторов BRIEF (Binary Robust Independent Elementary Features) составляет дескриптор из серии тестов: в окрестности особой точки случайным образом выбирается несколько (обычно 128, 256 или 512) пар пикселей, между каждой из которых производится сравнение интенсивности, результат сравнения i-ой пары (1 или 0) записывается в дескриптор на место i-ого бита. Таким образом, дескриптор может занимать 16, 32 или 64 байт памяти. Для уменьшения шумов и увеличения надёжности дескриптора к изображению предварительно применяется фильтр Гаусса. На слайде изображён пример случайно сгенерированных пар точек.

Слайд 11:

Генератор дескрипторов FREAK (Fast Retianl Keypoint) основан на том же принципе, что и BRIEF – серия тестов и запись результатов в вектор, для уменьшения чувствительности к шуму используется фильтр Гаусса. Главное различие состоит в том, что в алгоритме FREAK сравниваются не интенсивности пикселей, а взвешенные по Гауссу суммы неких подобластей. На слайде изображена схема сравниваемых подобластей дескриптора. Можно видеть, что чем дальше подобласти от центра, тем больше радиусы окружностей. Эта идея отражает идею рецептивного поля зрительной системы человека: чем ближе точка к центру поля зрения, тем больший отклик она создаёт.

Слайд 12:

Существует класс алгоритмов, объединяющих в себе как детектор точечных признаков, так и генератор дескрипторов. Преимущество таких методов в том, что, объединив два алгоритма в один, возможно избежать большого количества накладных коммуникационных расходов. Также такой подход позволяет объединить ускоспециализированные детектор точек и генератор дескрипторов, которые по отдельности не представляли бы ценности. В данной работе рассмотрены следующие алгоритмы: SIFT, SURF, ORB, BRISK

Слайд 13:

детектор SIFT хоть и не используется в данной работе, потому что является слишком медленным для систем реального времени, но является важным алгоритмом как сам по себе, так и потому, что на его основе разработан рассмотренный далее SURF.

Детектор точек — это разность гауссиан, которая даёт хороший отклик на угловые точки, поиск локальных максимумов и отбрасываник большого количества точек, принадлежащих контурам, но не являющихся хорошими углами. В пространстве масштабов каждому слою применяется свёртка с ядром фильтра Гаусса, затем слои попарно вычитаются, получается пространство разностей гауссиан, в котором ищутся локальные максимумы.

Слайд 14:

дескриптор в SIFT — это гистограмма градиентов в окрестности признака. Для каждой подобласти вычисляются локальные градиенты, которые попадают в один из восьми диапазонов. Полчается эффективно, но долго, поэтому SIFT используется в задачах, не требующих работы в реальном времени, например, при составлении панорам из нескольких снимков.

Слайд 15:

детектор SURF был разработан для ускорения SIFT за счёт снижения качества функции отклика. Построение пространства разностей гауссиан осуществляется свёрткой с упрощённым ядром (представлено на слайде), что позволяет использовать интегральные изображения и таким образом значительно ускорить обработку. В результате во многих случаях локализуемость признаков возрастает по сравнению в SIFT в силу простоты функции отклика (снижение операционной погрешности).

Слайд 16:

для построения гистограммы используются взвещенные суммы по прямоугольным подокнам, окружающим особую точку.

Слайд 17:

детектор ORB использует модификацию дететкора FAST: точки не просто сравниваются, а ищутся n/2+1 точек, котореы последовательно темнее или ярче исходной точки. Также строится пространство масштабов, в котором с помощью функции отклика дететкора Харриса выделяются наиболее характерные точки. Затем вычисляются ориентация признака.

Слайд 18:

в качестве дескриптора используется BRIEF, который комбинируется с вычисленным ранее направлением. Получается дескриптор, инвариантный относительно масштаба и поворотов.

Слайд 19:

детектор точек в BRISK также использует модификацию FAST, инвариантную относительно масштаба, но для уточнения значения функции отклика для максимума на i-ом слое вычисляются его значения на i-1-ом и i+1-ом слоях, затем эти три точки интерполируются параболой, максимум которой принимается за значение функции отклика.

Слайд 20:

построение дескриптора производится путём проведения серии тестов между областями, показанными на слайде. Каждая окружность — подобласть, к которой применяется свёртка с ядром Гаусса. Также для каждой области задано значение диаметра окружности d, внутри которой нахдоятся области, сравнения с которыми участвуют в построении дескриптора. Вне этой окружности находятся области, сравнения с которыми участвуют в вычислении ориентации дескритора. Получается инвариантный относительно масштабов и поворота дескриптор.

Слайд 21:

все рассмотренные дететкоры и дескрипторы находят и описывают интересуемые точки, далее их нужно отслеживать между кадрами. Для этого можно выделить два метода прослеживания: оптический поток и сопоставление признаков. Алгоритмы оптического потока используют непосредственно особые точки и ищут их на соседних кадрах, опираясь на некоторые допущения:

1. перемещение точки достаточно мало
2. все точки перемещаются в одном направлении

поиск точек производится в подобластях, размер которых параметризуется.

Сопоставление признаков – метод, использующий дескрипторы как уникальные идентификаторы точек. Среди двух множеств дескрипторов, принадлежащих соседним кадрам, сравнением выбираются соответствующие друг другу дескрипторы. Сущетствуют два подхода: полный перебор и поиск ближайшего соседа. Полный перебор выполняется слишком долго для работы в реальном времени, поэтому в данной работе он не используется. Вместо него будет использоваться Approximate Nearest Neighbour – приближенный поиск ближайшего соседа

Слайд 22:

в итоге, если скомбинировать все детекторы и методы прослеживания, которые были рассмотрены, получается список алгоритмов, которые вы видите на слайде

Слайд 23:

в работе были использованы следующие вычислители

слайд 24-26: в качестве исходных данных представлены две видеопоследовательности: одна представляет наземную съемку, другая – орбитальную

слайд 27:

далее – критерии оценки результатов запусков алгоритмов

слайд 28:

методика оценки алгоритмов прослеживания сводится к следующему: для каждой тройки алгоритм – архитектура – вычислитель, если это возможно, вычислить время работы в миллисекундах для каждого кадра, количество найденных признаков и количество прослеженных признаков

слайды 29-32:

детекторы харриса и shi-tomasi возвращают точки в слишком большом диапазоне, часто меньше пяти точек, поэтому прослеживание на исходных видеопоследовательностях не представляется надёжным. Для улучшения работы необходимо реализовывать выорку наиболее пригодных точек и подавление немаксимумов самостоятельно, что замедлит алгоритмы ещё сильнее.

Слайды 33-34: детектор Star, в отличие от предыдущих, работает стабильнее и для второй видеопоследовательности показывает хорошие результаты. Возможно, сыграли роль два фактора: инвариантность относительно масштаба и эффективный отбор наиболее хороших точек.

Слайды 35-36: Fast хоть и работает быстро, но, как было упомянуто ранее, возвращает прищнакои очень кучно, и поэтому результаты отслеживани сильно зашумлены, а графики количества прослеженных признаков не являются репрезентативными

Слайды 37-40: на данных графиках можно наблюдать, что дескрипторы BRIEF и FREAK работают достаточно быстро и на результат существенного влияния не оказывают

слайды 41-54: упомянуть для каждого слайда время работы алгоритма и выделить ORB и BRISK.

Слайд 55:

на основании данных, полученных в результате тестов, можно сделатьв ывод о том, что для первой видеопоследовательности наиболее подходящими оказались детектор BRISK на CPU и ORB на CUDA. Хоть BRISK и не управляется с работой в реальном времени (нужно быть быстрее 30мс), он наиболее лизок к этому результату. ORB на CUDA ближе, но его применение потребует добавление видеокарты от Nvidia в состав РТК, что увеличит стоимость и уменьшит надёжность, так как видеокарту достаточно просто повредить механически, а наземные РТК подвергаются сильным механическим воздействиям. Также важно учитывать, что размер кадра в первом видео 4000 на 2500 пикселей. Уменьшение размера стереорпары как уменьшит количество данных для обработки, так и повысит надежность, так как шумных деталей будет меньше.

Слайд 56:

Аналогично первому видео, хорошие результаты показали BRISK на CPU и ORB на CUDA. Ограничения также аналогичны. Но в данном случае BRISK полностью удовлетворяет всем требованиям. Также относительно хорошие результаты показал дететкор Star, 80мс.

Слайд 57: спасибо за внимание.