САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Математико-механический факультет

Кафедра системного программирования

Мухаматулин Максим Садриевич

Hyperlapse. Стабилизация видео с Помощью 3D-моделирования сцены

Курсовая работа

Научный руководитель:

доц. Пименов А. А.

Санкт-Петербург

2015

Оглавление

1 Введение

При съемке видео без использования специального оборудования или техники всегда наблюдается тряска, которая становится особенно заметной при ускорении видеофрагмента. Стандартные методы стабилизации (на постобработке) заключаются в слежении за положением характерных точек и их последующей фиксации или сглаживании (имеется в виду выравнивание скорости и траектории движения). Таким образом характерные точки на кадре становятся неподвижными или их траектории станут достаточно гладкими и предсказуемыми.

Эти методы демонстрируют приемлемые результаты для видео с незначительным перемещением по сцене. В случае же съемки в движении возникает эффект параллакса. Из-за этого характерные точки двигаются относительно друг друга, и сглаживание положения конкретной или средней точки становится неприемлемым методом для подобных видео. Также, из-за движения, точки постепенно выходят за границы кадра, и появляется необходимость в поиске новых характерных точек.

Собственно, hyperlapse - это и есть метод замедленной киносъемки в движении, названный по аналогии с timelapse - статичной замедленной киносъемки. Такая техника съемки часто используется в аэросъемке и в автомобильных регистраторах. Но основная область применения этой технологии является интерактивное искусство. Особенно данная методика популярна при съемке архитектуры, так как hyperlapse позволяет быстро дать представление о форме здания любого размера. На видеохостингах можно найти hyperlapse видео почти каждого города мира, знакомящее зрителя с основными его достопримечательностями.

2 Постановка задачи

В Microsoft Research предложили [1] метод стабилизации видео в движении, основанный на (1) реконструкции 3D сцены и положений камеры для каждого кадра в течении всего видео с последующим (2) сглаживанием траектории движения и ориентации камеры и (3) склейки выходного изображения из избыточного количества имеющихся кадров (так как видеофрагмент ускоряется, входных кадров значительно больше, чем будет на выходе). Данный метод не зависит от ориентации камеры и скорости появления новых точек в кадре, а также от величины тряски и её частоты (разве что это может привести к смазыванию кадра изображения). Также для восстановления сцены подразумевается, что камера не калибрована, то есть мы не знаем её внутренних параметров.

Цель моей работы заключается в реализации первого этапа метода стабилизации видео, предложенного Microsoft Research, на котором можно достаточно полно исследовать возможные улучшения и потенциал использования в робототехнике и других областях компьютерного зрения. То есть в реализации реконструкции 3D сцены и вычисление положений и ориентаций камеры для каждого кадра видеофрагмента для дальнейшей стабилизации траектории её движения.

3 Существующие решения

Существует 2 решения этой задачи, специально предназначенные для стабилизации видео в движении, но входные данные и подход этих решений несколько различается.

3.1 Photosynth.net

Сервис Photosynth [2] занимается тем, что создает интерактивные анимации для последовательности фотографий, количество которых явно недостаточно для получения плавной анимации. То есть, помимо стабилизации, сервис занимается тем, что генерирует промежуточные изображения из существующих таким образом, что они плавно перетекают друг в друга.

3.2 Microsoft Hyperlapse

Приложение Microsoft Hyperlapse [3] является закрытой реализацией метода, предложенного Microsoft Research. Оно работает ровно с противоположенными входными данными, в отличие от photosynth. Приложение получает видеофрагмент, снятый в обычном режиме, и ускоряет его. Стабилизация же происходит с помощью сглаживания траектории движения камеры и склейки избыточных кадров видеоролика таким образом, что этот кусок кадра для данного положения камеры (которого в самом видеофрагменте могло и не быть, но траектория сгладилась так, что в отстабилизированной траектории камера находится именно в этом положении).

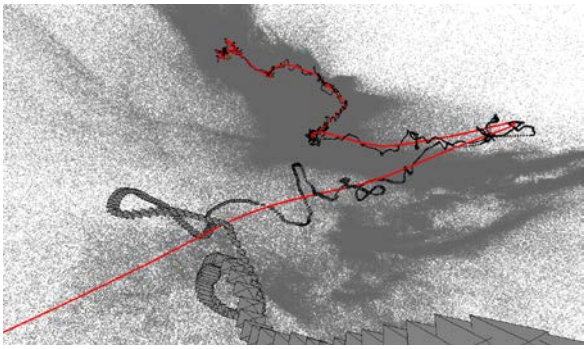

Рис. 1: Сглаживание траектории движения камеры

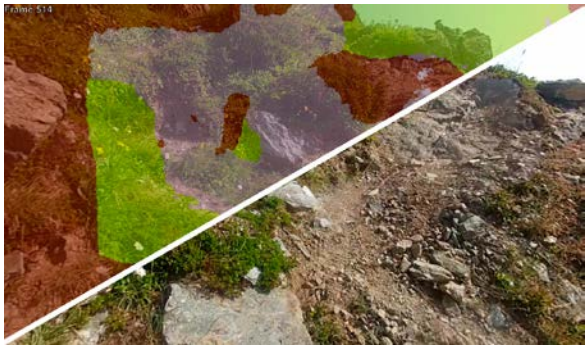

Рис. 2: Склейка избыточных кадров в выходное изображение

3.3 Другие решения

Решения, вроде Adobe Warp Stabilizer, Deshaker и proDAD Mercalli [4] работают на основе анализа изображения и деформации каждого кадра. A Hyperlapse from Instagram [5] похож на существующие алгоритмы стабилизации видео в том, что он также деформирует каждый видеокадр, чтобы удалить небольшое дрожание камеры, но, чтобы определить необходимое количество вращения для каждого кадра, полагается не на анализ изображения, а на встроенный гироскоп камеры. Но эти решения, как уже было сказано выше, не дают приемлемого результата при ускорении видеофрагмента.

4 Решение

В рамках данной работы реализуется алгоритм генерации 3d сцены по последовательности изображений, в которой камера меняет свое местоположение. Для реализации решения было решено использовать такие средства разработки программного обеспечения, как Microsoft Visual Studio 2013 Ultimate, C++. Для обработки изображений используется открытая библиотека компьютерного зрения OpenCV. В последующем, для решения задачи bundle adjustment применяется библиотека PBA [6].

4.1 Наивный метод стабилизации

Первым делом для ознакомления с инструментами был реализован наивный метод стабилизации, заключающийся в том, что объемная природа сцены игнорируется и кадр рассматривается как плоскость, в котором не возникает эффекта параллакса. Примерно таким образом работают решения вроде Adobe Warp Stabilizer.

4.1.1 Получение характерных точек изображения

Первым шагом в реализации этого метода является нахождение на изображении характерных точек. Для этого был использован детектор SIFT [7], обеспечивающий инвариантность нахождения одних и тех же характерных точек относительно преобразований изображений, то есть смещения, поворота, масштабирования и изменения яркости. Мною была применена реализация, входящая в поставку библиотеки OpenCV.

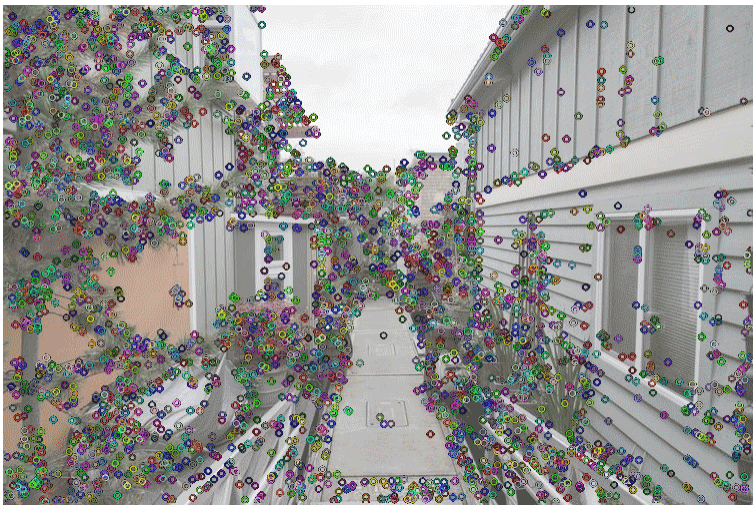

Рис. 3: Выделение характерных точек, найденных алгоритмом SIFT

Чуть более чем 3000 характерных точек выделено на изображении.

На этом этапе для каждого изображение получается огромное количество характерных точек для каждого кадра, и несколькими следующими шагами будет их фильтрация на основе нахождения соответствий характерных точек на соседних кадрах последовательности.

4.1.2 Получения соответствий для характерных точек соседних изображений

Первым этапом фильтрации будет использование стандартного средства OpenCV – Flann. В результате будет получено множество соответствий между характерными точками на соседних кадрах. Характерные точки, оставшиеся без соответствия, отметаются.

На данном этапе производилась только одна фильтрация, но в следующей реализации будет добавлена еще фильтрация на основе эпиполярного ограничения.

4.1.3 Поиск и применение матрицы гомографии

После получения множества соответствий для соседних кадров в наивном методе стабилизации наступает этап поиска такого преобразования между кадрами, что оно переводит все характерные точки из первого кадра максимально близко к соответствующих им характерным точкам со второго кадра. Это преобразование определяется матрицей H размера 3x3, называемой матрицей гомографии. Выражается она уравнением

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \\ 1 & 1 & \cdots & 1 \end{pmatrix} = H \cdot \begin{pmatrix} x_1' & x_2' & \cdots & x_n' \\ y_1' & y_2' & \cdots & y_n' \\ 1 & 1 & \cdots & 1 \end{pmatrix}$$

где (x_1, y_1) и (x_1', y_1') – координаты соответствующих характерных точек на первом и втором кадре соответственно. Эта матрица ищется методом наименьших квадратов, а потом уточняется с помощью алгоритма Левенберга-Марквардта. Для её поиска были использованы средства OpenCV.

При применении этой матрицы ко второму кадру, характерные точки второго кадра будут почти соответствовать характерным точкам первого кадра, а само изображение накладываться на первое. После нахождения матриц гомографий для каждой пары кадров, для получения матрицы гомографии между первым и n-ым кадром достаточно взять произведение всех матриц гомографий между первым и n-ым кадром.

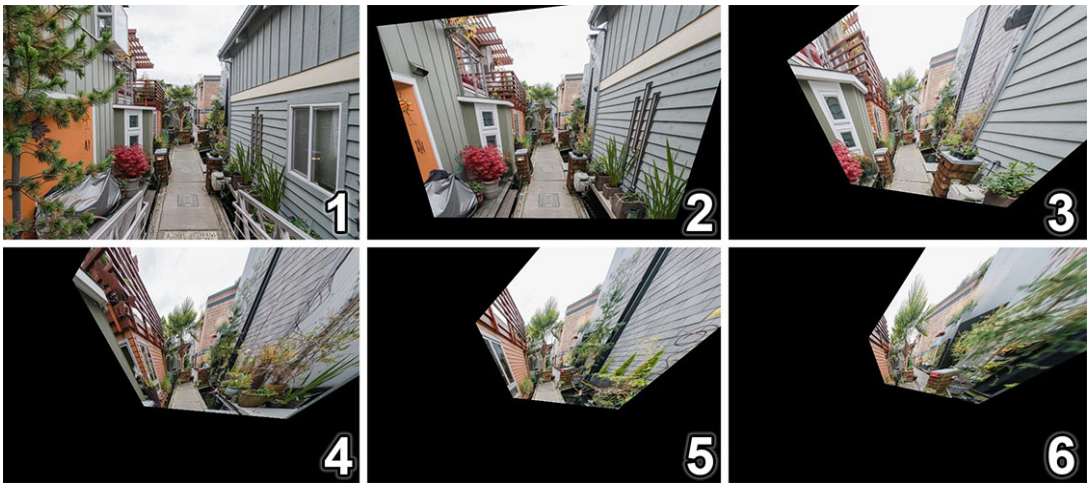

Рис. 4: Последовательное применение матрицы гомографии

Для получения матрицы гомографии, которое будет переводить второе изображение в первое, достаточно найти обратную матрицу к уже найденной матрице гомографии.

Следующим этапом была генерация промежуточных кадров для последовательности, для получения схожего с сервисом Photosynth эффекта. Идея заключается в применении неполной матрицы гомографии к текущему и следующему изображению и наложении их друг на друга с полупрозрачностью.

Неполная матрица гомографии выражается следующим образом

$$(1 - p) \cdot H + p \cdot E$$

где E – единичная матрица, H – матрица гомографии, а p – прогресс от текущего к следующему кадру. Нижний (текущий кадр) всегда остается непрозрачным, а верхний (следующий) имеет прозрачность, соответствующую прогрессу p.

Рис. 5: Начальный кадр, $p = 0$

Рис. 6: Промежуточный сгенерированный кадр, $p = 0{,}5$

Рис. 7: Конечный кадр, $p = 1$

Следующим этапом было применение морфинга для того, чтобы участки, на кадре которые при наложении не накладывались, хотя бы плавно перетекали друг в друга. Для получения карты морфинга использовался алгоритм Farnebäck [8], дающий в результате карту оптического потока (то есть для каждого пикселя – его смещение в следующем кадре). Для её получения использовалась стандартная реализация OpenCV.

Потом карта оптического потока передавалась функции remap, которая смещала каждый пиксель изображения согласно указанному смещению. Результат не оправдал ожидания.

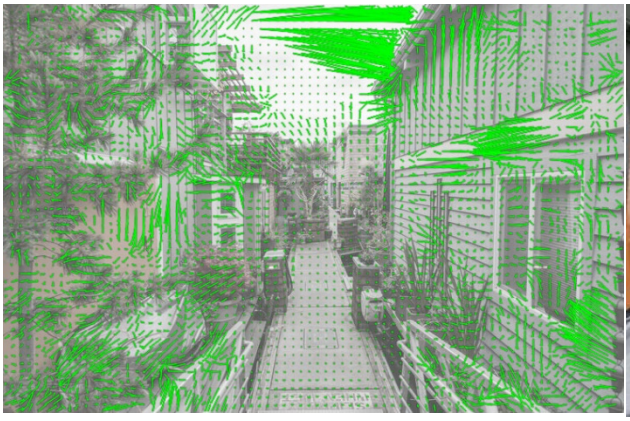

Рис. 8: Карта оптического потока, полученная с помощью алгоритма Farnebäck

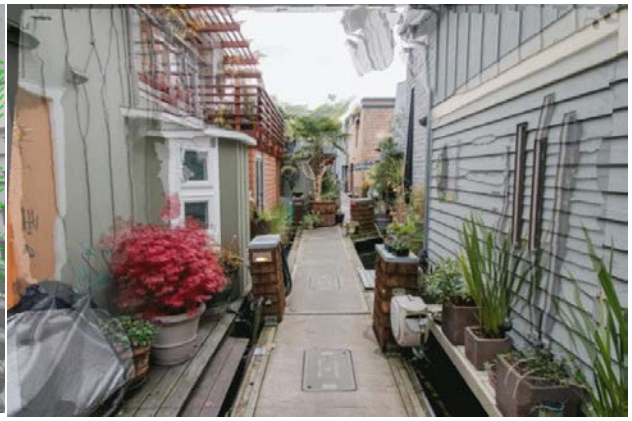

Рис. 9: Результат применения морфинга, "крыша поехала"

На этом этапе стало ясно, что игнорировать объемную природу сцены для получения приемлемых результатов невозможно. Поэтому следующим этапом был переход к воссозданию 3d сцены из изображений.

4.2 Метод воссоздания 3d сцены

Реализация наивного метода не прошла безрезультатно. Во-первых, были изучены инструменты, предоставляемые OpenCV, а во-вторых, первые этапы воссоздания 3d сцены, а именно поиск характеристических точек на изображениях и построение соответствий, совпадают с теми, что были реализованы в наивном методе. Поэтому ими можно воспользоваться.

4.2.1 Фильтрация соответствий с помощью эпиполярного ограничения

Здесь следующим этапом после первичной фильтрации будет вторичная фильтрация соответствий с помощью эпиполярного ограничения [9]. Его геометрический смысл заключается в том, что соответствующие точки должны лежать на прямых, имеющих одну точку пересечения. Эти прямые называются эпиполярными линиями. На практике это заключается в поиске такой матрицы F размером 3x3, называемой фундаментальной матрицей, для которой для большинства точек будет выполняется равенство $x_1^{\mathrm{T}} F x_2 = 0$, где x_1 и x_2 – однородные координаты характерных точек первом и втором изображении соответственно.

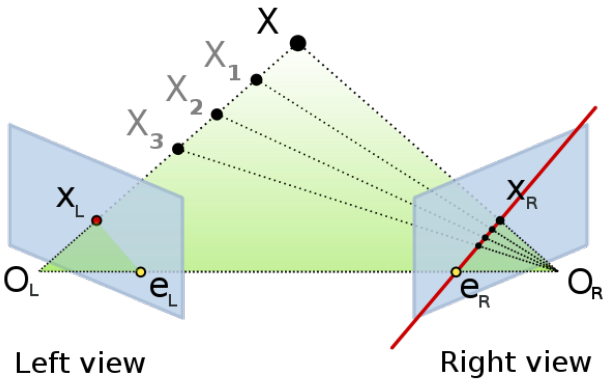

Рис. 10: Геометрический смысл эпиполярного ограничения

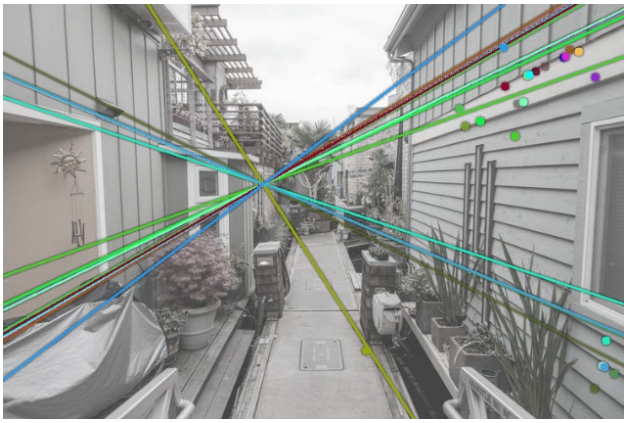

Рис. 11: Эпиполярные линии на изображении

Фундаментальная матрица ищется с помощью алгоритма RANSAC, то есть нахождении наилучшей случайной выборки. После её получения мы отфильтровываем соответствия, не вписывающиеся в заданный порог. Опытным путем было установлено, что хорошая фильтрация получается при таком пороге, при котором количество соответствий уменьшается в среднем в 3 раза. В случае, показанном ниже, количество соответствий упало со 180 до 50.

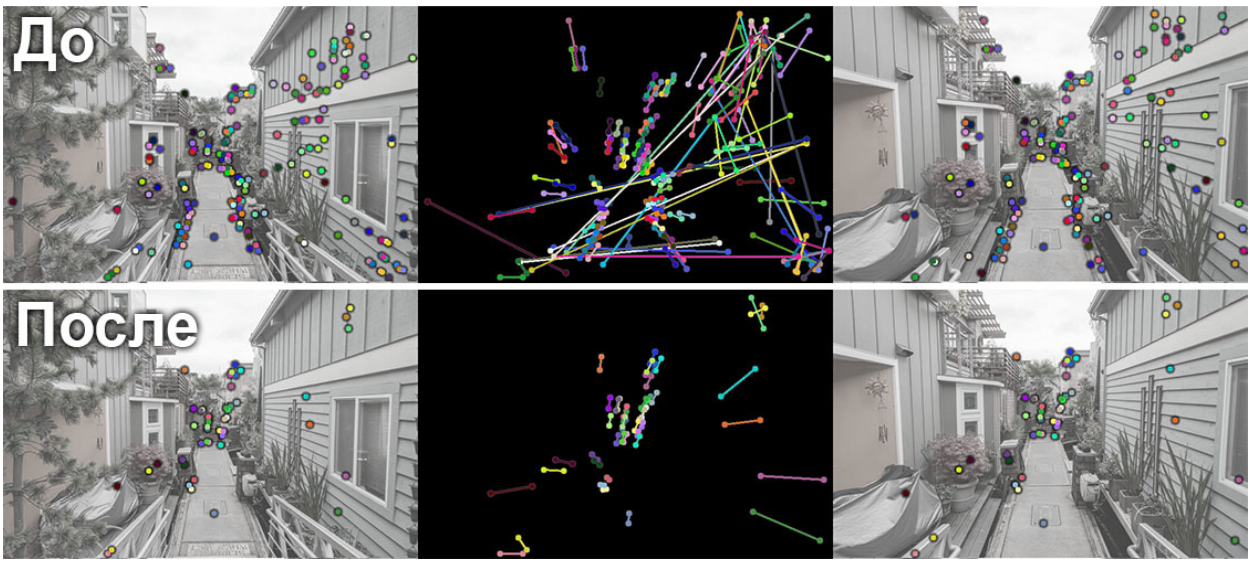

Рис. 12: Фильтрация соответствий с помощью эпиполярного ограничения

4.2.2 Извлечение внутренних параметров камеры из фундаментальной матрицы

Следующим шагом было извлечение черновых внешних параметров камеры из фундаментальной матрицы, полученной на предыдущем этапе. Для этого фундаментальная матрица преобразовывалась к особому виду, называемому существенной матрицей, с помощью равенства

$$E = K_1^{\mathrm{T}} F K_2$$

где E – существенная матрица, K_1 и K_2 – матрицы внутренних параметров камеры первого и второго кадра соответственно. Но так как мы считаем, что съемка всей последовательности велась одной камерой, то

$$K_1 = K_2 = K = \begin{pmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

где α_x и α_y – фокусное расстояние камеры по осям, а x_0 и y_0 – центральная ось камеры. Так как α_x и α_y нам не известны, на этом этапе они прикидываются и позже уточняются. А за x_0 и y_0 берется центр изображения.

Так как $E = R[t]_\times$, где

$$[t]_\times = \begin{pmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{pmatrix}$$

то существенная матрица однозначно раскладывается на внутренние параметры камеры, то есть матрицу поворота и координаты в пространстве. Для этого используется сингулярное разложение. Я использовал стандартную реализацию из OpenCV.

4.2.3 Решение задачи bundle adjustment

Следующим этапом было получение координаты положения в пространстве каждой характеристической точки изображения и уточненное положение камеры. Эта задача представляется в виде системы нелинейных уравнений

$$\begin{pmatrix} w_1u_1 & w_2u_2 & w_3u_3 & \cdots & w_nu_n \\ w_1v_1 & w_2v_2 & w_3v_3 & \cdots & w_nv_n \\ w_1 & w_2 & w_3 & \cdots & w_n \\ w_1'u_1' & w_2'u_2' & w_3'u_3' & \cdots & w_n'u_n' \\ w_1'v_1' & w_2'v_2' & w_3'v_3' & \cdots & w_n'v_n' \\ w_1' & w_2' & w_3' & \cdots & w_n' \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & x_0 & 0 & 0 & 0 \\ 0 & \alpha_y & y_0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha_x & 0 & x_0 \\ 0 & 0 & 0 & 0 & \alpha_y & y_0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ r_{11} & r_{12} & r_{13} & -t_1 \\ r_{21} & r_{22} & r_{23} & -t_2 \\ r_{31} & r_{32} & r_{33} & -t_3 \end{pmatrix} \cdot \begin{pmatrix} X_1 & X_2 & X_3 & \cdots & X_n \\ Y_1 & Y_2 & Y_3 & \cdots & Y_n \\ Z_1 & Z_2 & Z_3 & \cdots & Z_n \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}$$

где (u_i, v_i) и (u_i', v_i') – координаты i-ой характерной точи на пером и втором кадре соответственно, $\begin{pmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$ – матрица внутренних параметров (на предыдущем этапе мы брали черновые значения, здесь имеет смысл брать такие же), $\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$ и $\begin{pmatrix} -t_1 \\ -t_2 \\ -t_3 \end{pmatrix}$ – внешние параметры камеры (получили черновые значения на предыдущем этапе) и (X_i, Y_i, Z_i) – координаты i-ой характерной точки в пространстве.

Такая задача называется bundle adjustment, и решается она с помощью оптимизации входных данных посредством минимизации ошибки. Для решения этой задачи я использовал библиотеку PBA. Полученная на предыдущем этапе матрица внешних параметров используется как первое приближение. А в качестве первого приближения для координат характерных точек в пространстве используются их двумерные координаты, удаленные от камеры на значительное расстояние.

На данном этапе, из-за большого количества шумов и наличия ложных соответствий (даже после применения эпиполярного ограничения), результат применения алгоритма.

Для визуализации облака точек на данном этапе производится вывод в формат ply. Реализация вывода взята из библиотеки corecvs [10]. Приводить изображение 3d точек в данном формате не имеет большого смысла, так как в таком случае не удастся удостовериться в том, что они находятся на разных расстояниях в пространстве, так как облако точек получается очень разряженным.

Заключение

В рамках выполнения данной работы были реализованы следующие этапы метода стабилизации изображения с помощью 3D-моделирования сцены:

- Найдены характерные точки для каждого кадра последовательности с помощью алгоритма SIFT
- Найдены соответствия между характерными точками в соседних кадрах
- Реализован алгоритм фильтрации соответствий характерных точек с помощью эпиполярного ограничения
- Реализована архитектура для удобного поиска и хранения множества характерных точек, нахождения соответствий между этими множествами и фильтрации этих соответствий с помощью эпиполярного ограничения
- Реализовано извлечение черновых внешних параметров камеры из фундаментальной матрицы
- Создана обертка для pba для возможности использовать структуры библиотеки OpenCV для решения задачи bundle adjustment
- Получены координаты характерных точек в пространстве и положения камер для каждого кадра последовательности изображений

Список литературы

1. Kopf J., Cohen M., Szeliski R. First-person Hyperlapse Videos 2014. URL: http://research.microsoft.com/en-us/um/redmond/projects/hyperlapse/
2. Photosynth - Capture your world in 3D. [Электронный ресурс] URL: https://photosynth.net/
3. // Microsoft Hyperlapse - Microsoft Research: [сайт]. URL: http://research.microsoft.com/en-us/um/redmond/projects/hyperlapseapps/
4. Мерьков С. Стабилизация видео: сравнение трёх инструментов 2011. URL: http://www.ixbt.com/divideo/stabilization-1.shtml
5. // Difference between Microsoft's and Instagram's Hyperlapse. 2014. URL: http://research.microsoft.com/en-us/um/redmond/projects/hyperlapse/igcomp/
6. Wu C., Agarwal S., Curless B., Seitz S.M. Multicore Bundle Adjustment 2011. URL: http://grail.cs.washington.edu/projects/mcba/
7. Lowe D.G. Distinctive Image Features 2004. URL: http://www.cs.berkeley.edu/~malik/cs294/lowe-ijcv04.pdf
8. Farneb¨ack G. Two-Frame Motion Estimation Based on 2003. URL: http://www.diva-portal.org/smash/get/diva2:273847/FULLTEXT01.pdf
9. Hartley R., Zisserman A. Epipolar Geometry and the Fundamental Matrix 2004. URL: http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook1/HZepipolar.pdf
10. // corecvs: [сайт]. URL: https://github.com/PimenovAlexander/corecvs/blob/master/core/geometry/mesh3d.cpp#L266