Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт математики и фундаментальной информатики
Базовая кафедра вычислительных и информационных технологий

УТВЕРЖДАЮ
Заведующий кафедрой
_____ / В.В. Шайдуров

«16» мая 2017 г.

БАКАЛАВРСКАЯ РАБОТА

Направление 02.03.01 Математика и компьютерные науки

РЕКОНСТРУИРОВАНИЕ 3D-МОДЕЛИ ПО ГРАФИЧЕСКИМ ИЗОБРАЖЕНИЯМ

Научный руководитель
кандидат физико-математических наук,
доцент _____ / Д.А. Цыганок
16.06.17

Выпускник _____ /А.А. Кубраков
16.06.17

Красноярск 2017

РЕФЕРАТ

Выпускная квалификационная работа по теме «Реконструирование 3D-модели по графическим изображениям». Содержит 24 страницы текста, 1 приложение, 11 использованных источников.

ИЗОБРАЖЕНИЕ, ПРОЕКТИВНАЯ КАМЕРА, МАТРИЦА ВНУТРЕННИХ ПАРАМЕТРОВ КАМЕРЫ, ГИСТОГРАММА, ОСОБАЯ ТОЧКА, ДЕСКРИПТОР, ФУНДАМЕНТАЛЬНАЯ МАТРИЦА, СУЩЕСТВЕННАЯ МАТРИЦА, ВОССТАНОВЛЕНИЕ ТРЕХМЕРНЫХ КООРДИНАТ.

Цель работы – создание WEB-приложения для реконструирования 3D модели по графическим изображениям.

В результате мной были изучен и реализован алгоритм 3D реконструкции, разработан сервер и пользовательский интерфейс для WEB-приложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время в связи с быстрым развитием 3D моделирования, применяемых в 3D печати, робототехнике, игровой индустрии и прочих компьютерных технологиях существует необходимость создания фотореалистичных трёхмерных моделей реальных объектов с высокой точностью.

Поскольку создание подобной трёхмерной модели вручную довольно трудоёмкий процесс, требующий от специалиста высоких навыков, появилась задача автоматизации процесса 3D моделирования. Одним из решений этой задачи является создание модели по набору фотографий реального объекта.

На сегодняшний день существует ряд программных решений реконструирования трехмерных моделей по изображениям. В то время как одна группа этих решений, таких как: Agisoft Photoscan, Cinema 4D R18, PhotoModeler, показывают качественные и стабильные результаты, но обладают высокой ценой (от 100$ за базовые версии и порядка 1000$ за расширенные), другая группа представляет собой свободное программное обеспечение, которое не обладает высокой точностью и стабильностью реконструирования.

Помимо прочего, данные решения работают на определённой платформе, как правило, Microsoft Windows. Наиболее же удобным вариантом как для использования, так и для разработки является формат WEB приложения.

Цели и задачи работы. Целью данной работы является разработка WEB приложения для реконструкции 3D модели по набору графических изображений.

Для достижения цели были поставлены и решены следующие задачи.

1. Изучить алгоритм 3D реконструкции.
2. Программно реализовать данный алгоритм.
3. Разработать серверную часть приложения.
4. Разработать пользовательский интерфейс.

Структура и объем бакалаврской работы. Работа состоит из введения, трех разделов, заключения, списка литературы. Общий объем составляет 32 страницы, включая приложение; иллюстративный материал представлен 12 рисунками (из них 6 в приложениях); список литературы содержит 11 наименований.

1 Алгоритм реконструкции 3D модели по набору изображений

В данной главе будут использоваться определения из работы [1].

Определение 1.1. Изображением будем называть прямоугольную матрицу с элементами a_{ij}. Если a_{ij} – вектор вида (I_1, I_2, I_3), где $0 < I_i < 255$ – целые числа, выражающие интенсивность красного, зеленого и синего цветовых каналов соответственно, изображение называется цветным. Если a_{ij} – целые числа, принимающие значения от 0 до 255, изображение называется монохромным.

Будем обозначать изображение через функцию вида $I(x, y)$, значение которой соответствует элементу матрицы с индексами (x, y). Через $I_r(x, y)$, $I_g(x, y)$, $I_b(x, y)$ обозначим значение цветовых каналов в точке (x, y).

В работе изображение будет рассматриваться как часть Евклидовой плоскости, при этом номера строк и столбцов будут рассматриваться как её координаты.

Определение 1.2. Однородными координатами называется вектор вида $(x, y, w)^T$, $w \neq 0$.

Однородным координатам можно сопоставить вектор Евклидовой плоскости с координатами $(x/w, y/w)^T$. Обратное преобразование в однородные координаты производится добавлением единицы к вектору: $(x, y)^T \rightarrow (x, y, 1)^T$.

Замечание 1.1. Для любого ненулевого числа a, векторы $(x, y, w)^T$ и $(ax, ay, aw)^T$ соответствуют одной и той же точке.

Аналогичным образом вводится понятие однородных координат для трехмерных точек.

Для алгоритма реконструирования понадобится описать модель камеры изображения. Для этого хорошо подходит модель, называемая проективной камерой. Модель позволяет проецировать точки трехмерного

пространства на плоскость изображения.

Геометрия модели представлена на рисунке 1.1.

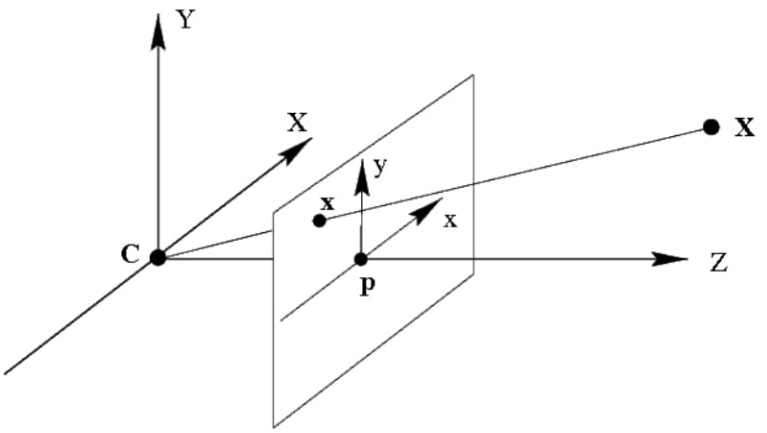

Рисунок 1.1 – геометрия проективной камеры

Проективная камера задается точкой C – центром камеры, лучом Cp – главной осью камеры, плоскостью изображения — плоскостью на которую выполняется проецирование точек, и системой координат на этой плоскости. В такой модели, произвольная точка пространства X проецируется на плоскость изображения в точку x лежащую на отрезке CX, который соединяет центр камеры C с исходной точкой X.

Формула проецирования имеет простую математическую запись в однородных координатах:

$$x = PX, \tag{1.1}$$

где X - однородные координаты точки пространства, x — однородные координаты точки плоскости, P — матрица камеры размера 3 × 4.

Матрица P выражается следующим образом:

$$P = KR[I\,|\text{-}\mathbf{c}] = K[R|\mathbf{t}], \tag{1.2}$$

где K – верхняя треугольная матрица внутренних параметров камеры

размера 3 × 3 (конкретный вид приведен ниже), R – ортогональная матрица размера 3 × 3, определяющая поворот камеры относительно глобальной системы координат, I – единичная матрица размера 3 × 3, вектор $\boldsymbol{c}$ — координаты центра камеры, а $\boldsymbol{t} = -R\boldsymbol{c}$.

Стоит отметить, что в силу замечания 1.1 матрица камеры определена с точностью до постоянного ненулевого множителя, который не изменит результатов проецирования точек по формуле (1.1). Однако этот постоянный множитель обычно выбирается так, чтобы матрица камеры имела вышеописанный вид.

В самом простейшем случае, когда центр камеры лежит в начале координат, главная ось камеры сонаправлена оси Cz, оси координат на плоскости камеры имеют одинаковый масштаб (что эквивалентно квадратным пикселям), а центр изображения имеет нулевые координаты, матрица камеры будет равна $P = K[I|0]$, где

$$K = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{1.3}$$

У реальных CCD камер пикселы обычно незначительно отличаются от квадратных, а центр изображения имеет ненулевые координаты. В таком случае матрица внутренних параметров примет вид:

$$K = \begin{pmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{1.4}$$

Коэффициенты f, α_x, α_y — называются фокусными расстояниями камеры.

Помимо этого, в силу неидеальности оптики, на изображениях,

полученных с камер, присутствуют искажения-дисторсии. Данные искажения имеют нелинейную математическую запись:

$$\begin{aligned} x'' &= x'(1+k_1r^2+k_2r^4+k_3r^6)+2p_1x'y'+p_2(r^2+2x'^2), \\ y'' &= y'(1+k_1r^2+k_2r^4+k_3r^6)+2p_2x'y'+p_1(r^2+2y'^2), \end{aligned} \tag{1.5}$$

где k_1, k_2, p_1, p_2, k_3 — коэффициенты дисторсии, являющиеся параметрами оптической системы; $r^2 = x'^2 + y'^2$; (x', y') — координаты проекции точки относительно центра изображения при квадратных пикселях и отсутствии искажений; (x'', y'') — искаженные координаты точки относительно центра изображения при квадратных пикселях.

Определение 1.3. В ситуации, когда известны внутренние параметры камеры и коэффициенты дисторсии говорят, что камера откалибрована.

На вход алгоритма поступают цифровые изображения формата JPEG, PNG или PGM. На выходе алгоритм возвращает массив реконструированных трехмерных координат модели и массив цветов, соответствующих каждой координате.

Алгоритм реконструирования трехмерной модели по набору изображений состоит из следующих этапов.

1. Предобработка входных изображений.
2. Поиск особых точек на каждом изображении и их сопоставление с особыми точками на других изображениях.
3. Определение трехмерных координат камер, соответствующих каждому изображению, по найденным особым точкам.
4. Восстановление трехмерных координат соответствующих друг другу особых точек.
5. Минимизация вычислительной ошибки.

1.1 Предварительная обработка входных изображений

Так как изображения, поступающие на вход алгоритма, получены при съемке с разных ракурсов с разными условиями освещения, то для улучшения результатов второго этапа алгоритма необходима предварительная обработка, включающая перевод изображений в монохромные и корректировку контрастности изображений.

Перевод изображения в монохромный режим необходим, так как для работы алгоритма достаточно одного цветового канала. Пусть дано изображение $I = I(x, y)$. Тогда точки монохромного изображения являются средним значением трех цветовых каналов цветного изображения и определяются по формуле

$$I_{gray}(x,y) = \frac{I_r(x,y) + I_g(x,y) + I_b(x,y)}{3}. \quad (1.6)$$

Так как поиск особых точек на втором шаге алгоритма может затруднится при наличии на изображениях областей с низкой контрастностью, то необходимо повысить контрастность таких областей.

Введем понятие гистограммы изображения.

Определение 1.4. Гистограммой изображения будем называть дискретную функцию H, определённую на множестве значений $[0, 255]$. При этом значением $H[i]$ является отношение числа точек на изображении со значением i к 255.

Очевидно, что для повышения контрастности изображения, необходимо приблизить гистограмму изображения к равномерному распределению.

Добиться этой цели можно с помощью процедуры называемой эквализацией гистограммы. Далее изложена её суть.

Пусть $H[i]$ вычисленная гистограмма монохромного

изображения *I(x, y)*. Вычислим новые значения гистограммы $H_{экв}$ по формуле:

$$H_{экв}[i] = H[i-1] + H[i], \qquad i = \overline{1..255}. \tag{1.7}$$

Теперь изображение будет преобразовано по формуле:

$$I_{экв}(x,y) = H[I(x,y)]. \tag{1.8}$$

Пример 1.1

Рассмотрим работу вышеизложенной процедуры на примере изображения на рисунке 1.2

Рисунок 1.2 – тестовое изображение.

Гистограмма изображения будет иметь график (рисунок 1.3):

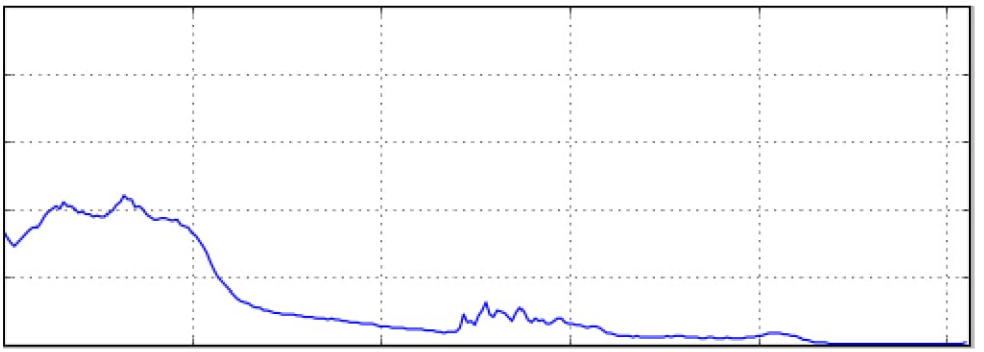

Рисунок 1.3 – гистограмма исходного изображения

После процедуры эквализации гистограммы график примет вид (рисунок 1.4):

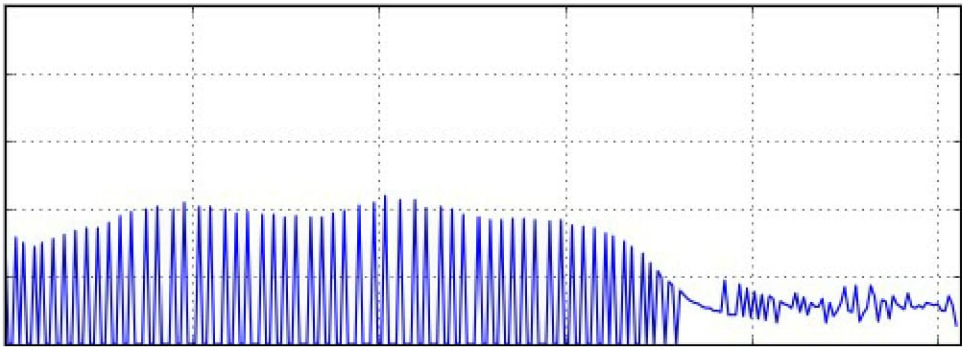

Рисунок 1.4 – гистограмма после процедуры эквализации.

После преобразования изображения, оно будет иметь вид (рисунок 1.5).

В данном подразделе был описан простой и эффективный метод предварительной обработки изображений с целью повышения стабильности второго этапа алгоритма.

Рисунок 1.5 – предобработанное изображение

1.2 Поиск и сопоставление особых точек

Для определения трехмерных координат объекта необходимо установить некоторые точки на исходных изображениях, соответствующие одним и тем же реального мира. Такими точками будут служить *особые точки.*

Определение 1.5. Особой точкой называется точка изображения, окрестность которой можно отличить от окрестности любой другой точки изображения в некоторой большей окрестности.

Так как изображения сняты с разных ракурсов с разным освещением, алгоритм поиска особых точек должен быть устойчив к повороту камеры, к изменению освещения, присутствия шумов на изображении.

На сегодняшний день наиболее популярным алгоритмом, отвечающим этим требованиям является SIFT (Scale-invariant feature transform), предложенный David Lowe. Описание алгоритма содержится в оригинальной работе [2]. Так как алгоритм запатентован, его свободное использование невозможно, поэтому он не удовлетворяет целям настоящей

работы.

Альтернативой алгоритму SIFT является, алгоритм ORB (Oriented FAST and rotated BRIEF), представленный Ethan Rublee в 2011м году. Подробное описание алгоритма приведено в [3].

Алгоритм ORB принимает на вход монохромное изображение и находит особые точки с помощью модифицированного алгоритма FAST [3,4]. Далее, для того чтобы сопоставить особые точки, соответствующие одной и той же трехмерной точке реального объекта, используется дескриптор rBRIEF [3,5]. Дескриптор особой точки описывает окрестность этой точки таким образом, что при повороте камеры и изменении освещения дескриптор остается неизменным.

Алгоритм возвращает список особых точек и их дескрипторов для каждого изображения.

Предполагается, что исходные изображения поступают в порядке увеличения угла поворота камеры относительно объекта реконструкции. Пусть $I_1, I_2, \ldots, I_N$ – исходные изображения. Особые точки будут сопоставляться между изображениями $(I_1, I_2), (I_2, I_3), \ldots, (I_{N-1}, I_N)$.

Для обеспечения большей точности, сопоставление пары изображений производится полным перебором и сравнением всех дескрипторов особых точек. Необходимым условием работы алгоритма является взаимооднозначность найденных соответствий: точка одного изображения может соответствовать только одной точке другого изображения и наоборот.

1.3 Определение координат камер

Для определения трехмерных координат реального объекта необходимо знать координаты камер исходных изображений. Для определения положения камер будет использоваться понятие *фундаментальной матрицы.*

Определение 1.6. Фундаментальной матрицей F называется матрица размера 3 × 3, такая, что для любой пары соответствующих точек $x \leftrightarrow x'$ двух изображений выполняется условие:

$$x'^T Fx = 0. \tag{1.9}$$

Из работы [1, с. 241-247] известно, что ранг F равен 2 и она определена с точностью до ненулевого множителя и зависит только от матриц исходных камер. Помимо фундаментальной матрицы, существует такое понятие, как существенная матрица.

Определение 1.7. Пусть дана пара изображений с фундаментальной матрицей F и матрицами внутренних параметров K, K'. Существенной матрицей называется матрица

$$E = K'^T FK. \tag{1.10}$$

По существенной матрице можно восстановить положение и поворот второй камеры, если матрицу поворота R первой камеры принять за единичную, а центр первой камеры расположить в начале координат (принять вектор сдвига $\boldsymbol{t}$ за нулевой). Матрица поворота второй камеры R' и её вектор сдвига $\boldsymbol{t}$ определяются из сингулярного разложения матрицы E.

Пусть дано сингулярное разложение матрицы E:

$$E = U \sum V^T, \tag{1.11}$$

где U и V – ортогональные матрицы размера 3 × 3, а матрица Σ имеет вид:

$$\sum = \begin{pmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{1.12}$$

Диагональные элементы Σ являются её сингулярными числами, два из которых, в силу ограничений на существенную матрицу, описанных в [1, с. 257-259], должны совпадать, а оставшееся равняться нулю.

Пользуясь результатами [1, с. 253-257], имеем выражения для R', $\boldsymbol{t}$:

$$\begin{aligned} R &= UW^{-1}V^T, \\ [\mathbf{t}]_\times &= UW\sum U^T, \end{aligned} \tag{1.13}$$

где

$$\begin{aligned} W &= \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \\ [\mathbf{t}]_\times &= \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}. \end{aligned} \tag{1.14}$$

Таким образом, для нахождения положения камер исходных изображений необходимо знать фундаментальные матрицы для пар изображений и матрицы внутренних параметров.

В данной работе предполагается, что для съемки объекта

используется одна камера с постоянным фокусным расстоянием. На данном шаге задается приближенное значение матрицы внутренних параметров; при минимизации вычислительной ошибки (подраздел 1.5) вычисляется близкое к точному значение матрицы. Если разрешение исходных изображений n × m, то для приближенного значения допустимо использовать матрицу

$$K = \begin{pmatrix} 0{,}8n & 0 & 0{,}5n \\ 0 & 0{,}8n & 0{,}5m \\ 0 & 0 & 1 \end{pmatrix}. \qquad (1.15)$$

Для вычисления фундаментальной матрицы используется нормализованный восьмиточечный алгоритм описанный в [6]. Для его работы необходимо минимум 8 пар соответствующих точек. Используемая в данной работе реализация, рассматриваемая в разделе 3, отсеивает пары соответствующих точек, не удовлетворяющие с заданной точностью условию (1.9), что позволяет снизить ошибку вычисления.

Полностью рассмотрим работу данного этапа алгоритма. Из второго этапа получены точечные соответствия для пар изображений (I_1, I_2), (I_2, I_3), … , (I_{N-1}, I_N). Считаем, что центр камеры для I_1 находится в начале координат, а её матрица поворота единичная. Используя вышеизложенные выкладки находим матрицу поворота и вектор сдвига для камеры I_2, и далее последовательно находим матрицы оставшихся камер, учитывая уже найденные матрицы.

1.4 Восстановление трехмерных координат

Зная положения камер можно восстановить трехмерные координаты соответствующих друг другу точек. Пусть $I_1, I_2, \ldots, I_N$ – исходные изображения, и проекции точки реального мира X присутствуют на $I_{k1}, I_{k2}, \ldots, I_{ks}$. Рассмотрим пару таких изображений. В идеальных условиях, лучи, проведенные от центров камер данных изображений через соответствующие проекции точки X , будут пересекаться в точке X.

Однако на практике, в силу неидеальности оптики и присутствия шумов, лучи не будут пересекаться.

В работе используется метод восстановления трехмерной координаты, в котором предполагается, что точка X, лежит на середине кратчайшего отрезка, соединяющего два луча (Рисунок 1.6).

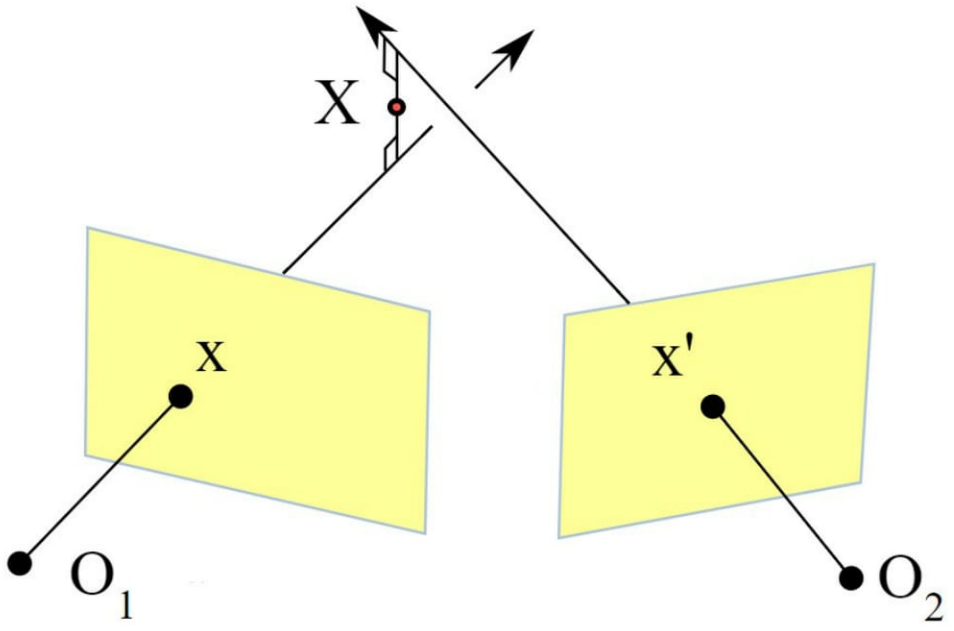

Рисунок 1.6 – восстановление трехмерной координаты

Таким образом, если даны два изображения с проекциями x, x' точки X, матрицами поворота R, R', векторами сдвига $\boldsymbol{t}, \boldsymbol{t'}$ и общей

матрицей внутренних параметров K, то точка X восстанавливается следующим образом:

Методом наименьших квадратов решается система трех линейных уравнений относительно двух параметров a_1, a_2 :

$$a_1 R^{-1} K^{-1} x - a_2 R'^{-1} K^{-1} x' = \mathbf{t} - \mathbf{t}'. \quad (1.16)$$

Восстанавливается точка X:

$$X = a_1 R^{-1} K^{-1} x - \mathbf{t} + a_2 R'^{-1} K^{-1} x' - \mathbf{t}'. \quad (1.17)$$

Замечание 1.2. в вышеизложенном методе точки x, x', X в однородных координатах.

Таким образом, проводя восстановление точки X для пар изображений I_{k1}, I_{k2}, ... , I_{ks} получим $s-1$ вычисленное значение X. В качестве решения выбирается среднее из них.

1.5 Минимизация вычислительной ошибки

На предыдущем этапе алгоритма получено начальное приближение восстановленных точек. На данном этапе производится минимизация вычислительной ошибки, значения восстановленных точек и параметров всех камер приближаются к точным. Задача минимизации выглядит следующим образом:

$$\min_{a_j,\, b_i} \sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} d(Q(a_j, b_i), x_{ij})^2, \quad (1.18)$$

где n – количество трехмерных точек, проекции которых присутствуют на

m изображениях, $\boldsymbol{x}_{ij}$ – проекция i-й точки на j-м изображении, $d(\cdot, \cdot)$ – расстояние в трехмерном пространстве, $\boldsymbol{a}_j$, $\boldsymbol{b}_i$ – вектора параметров камеры и трехмерных точек соответственно, $Q(\boldsymbol{a}_j, \boldsymbol{b}_i)$ – вычисленная проекция i-й точки на j-м изображении, а v_{ij} равен 1, если проекция i-й точки присутствует на j-м изображении, и 0 в противном случае.

Заметим, что вектор параметров камеры $\boldsymbol{a}_j$ так же содержит коэффициенты дисторсии.

Проекция Q вычисляется по формуле (1.1).

Данная задача представляет задачу нелинейной оптимизации и для её решения применяется метод наименьших квадратов.

2 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

В данной главе рассмотрены инструменты для реализации алгоритма и примеры их использования.

Основным языком разработки был выбран Python версии 3.5. Это современный язык высокого уровня, который в сочетании с C++ библиотеками обладает высокой скоростью и при этом удобен для научных вычислений. Помимо этого, код, написанный на Python, совместим с серверной частью приложения, разобранной ниже в данной главе.

2.1 Реализация алгоритма 3D реконструкции

Для реализации алгоритма 3D реконструкции были использованы следующие библиотеки:

- OpenCV 3 – библиотека компьютерного зрения. В ней реализована работа с разными форматами изображений, операции предобработки изображений, поиска особых точек, фундаментальной матрицы и матриц камеры. Данная библиотека не имеет альтернатив по функциональности и производительности.
- NumPy 1.12 – библиотека линейной алгебры. Используется для операций с матрицами и векторами.
- SciPy 0.19 – библиотека для научных вычислений. Используется для решения задач оптимизации. В частности, в данной работе используется реализованный в ней метод наименьших квадратов.

Данные библиотеки являются открытыми, написаны на языке C++ и имеют интерфейс для языка Python. Это обеспечивает высокую скорость работы вычислений.

Рассмотрим основные функции данных библиотек, используемые в

программе.

Для загрузки изображений используется функция библиотеки OpenCV cv2.imread(). Она принимает два параметра – путь к изображению и режим чтения изображения. В данной программе в качестве режима чтения выбран флаг cv2.IMREAD_GRAYSCALE, который позволяет считывать изображение в монохромном режиме. Изображение загружается в виде объекта numpy.array библиотеки NumPy, который представляет из собой матрицу с целыми значениями от 0 до 255.

Для экзализации гистограммы используется функция cv2.equalizeHist(), которая принимает на вход изображение и изменяет его.

Далее для поиска особых точек с помощью алгоритма ORB применяется объект класса cv2.ORB. Он создается при помощи функции cv2.ORB_create() и имеет метод detectAndCompute, который принимает на вход изображение и возвращает массив особых точек и массив их дескрипторов.

Для сопоставления точек полным перебором используется объект cv2.BFMatcher. Его конструктор имеет параметр crossCheck, который принимает булево значение. Если crossCheck = True, то сопоставление производится с требованием взаимооднозначности.

Для нахождения фундаментальной матрицы используется функция cv2.findFundamentalMat(). Она принимает несколько параметров: cv.FindFundamentalMat(points1, points2, method). Здесь points1, points2 – массивы особых точек, при этом, точки с одинаковыми индексами соответствуют. Параметр method определяет метод нахождения фундаментальной матрицы. В данном алгоритме он принимает значение cv2.CV_FM_8POINT. Для определения матриц поворота и векторов сдвига используется функция cv2.recoverPose(), которая принимает в качестве параметров существенную матрицу и два массива особых точек для пары изображений.

В алгоритме так же используются функции numpy.linalg.lstsq() для решения системы линейных алгебраических уравнений при восстановлении трехмерных координат и scipy.optimize.least_squares для решения задачи нелинейной оптимизации (1.18).

2.2 Реализация WEB-приложения.

Для реализации серверной части используется фреймворк Flask. Сервер на данном фреймворке отличается простой реализации, при этом имеет большой функционал и устойчив к высоким нагрузкам.

Для разработки сервера была использована официальная документация Flask [8].

Пользовательский интерфейс был реализован стандартными методами с использованием языка Javascript и таблиц стилей Bootstrap.

Так же, для отображения результатов реконструкции в виде трехмерной модели была использована библиотека three.js.

2.3 Архитектура WEB-приложения.

Приложение состоит из нескольких модулей. Рассмотрим модули, main.py, load_images.py, match_images.py, image.py отвечающие за реализацию алгоритма реконструкции.

Модуль image.py представляет собой класс изображения, который содержит оригинальное изображение, уменьшенную с целью оптимизации работы монохромную копию изображения, списки особых точек и их дескрипторов для изображения и список точечных соответствий со следующим изображением. Конструктор класса Image выглядит следующим образом:

```
def __init__(self, img, gray, kp, des):
    self.img = img
    self.gray = gray
    self.kp = kp
    self.des = des
    self.matches = {}.
```

Здесь img – оригинальное изображение, gray – уменьшенная монохромная копия, kp и des – списки особых точек и их дескрипторов, matches – список точечных соответствий, который заполняется на следующих этапах работы программы.

Модуль load_images.py отвечает за загрузку и предобработку изображений. Он загружает изображения на сервере при помощи функции cv2.imread(), эквализирует их гистограммы функцией cv2.equalizeHist() и находит особые точки при помощи объекта класса cv2.ORB функцией detectAndCompute(). Модуль создает объекты класса Image для хранения изображений и информации об особых точках.

Модуль match_images.py последовательно находит точечные соответствия для пар изображений при помощи объекта класса cv2.BFMatcher и дополняет объекты класса Image, созданные в модуле load_images.py.

Основным модулем является модуль main.py, который последовательно вызывает процедуры модулей load_images.py, match_images.py для загрузки, обработки изображений и поиска точечных соответствий. Далее последовательно для пар изображений находятся фундаментальная и существенная матрицы, восстанавливаются положения камер и трехмерных координат с использованием функций из подраздела 2.2. В конце работы модуль main.py производит минимизацию вычислительной ошибки.

Серверная часть полностью реализована в модуле view.py и её

основные функции представлены в методе index() с декоратором вида: @app.route('/', methods=['POST', 'GET']). При GET запросе к корневому каталогу сервера, приложение возвращает пользователю страницу с полем для выбора и загрузки изображений. При помощи технологии ajax, изображения отсылаются на сервер, реконструируется трехмерная модель и отсылается пользователю обратно через ajax.

3 ПРИМЕРЫ РАБОТЫ ПРИЛОЖЕНИЯ

Для тестирования WEB-приложения был использован набор из 16 фотографий объекта с разных ракурсов (рисунок 3.1). Для съемки была использована камера смартфона модели Xiaomi Redmi Note 3 Pro. Тестирование проводилось на сервере, запущенном на персональном компьютере с процессором Intel Core I7 3517U, 4GB RAM, под управлением операционной системой Ubuntu 14.04. Время выполнения алгоритма составило 121,56 секунд.

Интерфейс WEB-приложения имеет поле загрузки фотографий с устройства и окно отображения результата в виде трехмерной модели (рисунок 3.2). Трехмерную модель можно рассматривать с разных ракурсов, вращая камеру левой кнопкой мыши и перемещая правой (Рисунки 3.3, 3.4, 3.5, 3.6). Из рисунков видно пример результата работы приложения. Некоторые области трехмерной модели не реконструировались, из чего можно сделать вывод, что для лучшей работы алгоритма текстура объекта не должна быть однородной, так как однородная текстура не содержит особых точек. Для улучшение работы алгоритма можно использовать нанесение текстуры на модель.

ЗАКЛЮЧЕНИЕ

В работе получены следующие результаты:

1. Изучен алгоритм 3D реконструкции.
2. Программно реализован данный алгоритм.
3. Разработана серверную часть приложения 3D реконструкции по набору изображений.
4. Разработан пользовательский интерфейс приложения.

Данное приложение является достаточно эффективным аналогом платных решений задачи 3D реконструкции.

Так же, данная работа служит опорой для дальнейших исследований с целью повышения точности реконструирования и возможности наложения текстур на 3D модель.

Список литературы

1. Richard Hartley and Andrew Zisserman. Multiple View Geometry in computer vision / R. Hartley, Zisserman. A - Cambridge University Press, 2003 - 655p.
2. David, L. G. Distinctive Image Features from Scale-Invariant Keypoints. / L. G. David // Journal of Computer Vision, 2004 - 28p.
3. ORB: an efficient alternative to SIFT or SURF (PDF). / E. Rublee, V. Rabaud, K. Konolige, G. Bradski. - IEEE International Conference on Computer Vision (ICCV), 2011 – 8p.
4. Edward Rosten, Tom Drummond. Machine learning for high-speed corner detection (PDF). / E. Rosten, T. Drummond. - European Conference on Computer Vision, 2006 – 14p.
5. BRIEF: Binary Robust Independent Elementary Features / M. Calonder, V. Lepetit, C. Strecha, P. Fua - 11th European Conference on Computer Vision (ECCV), Heraklion, Crete, 2010. – 14p.
6. Richard I. Hartley. In Defense of the Eight-Point Algorithm. / R. I. Hartley - IEEE Transaction on Pattern Recognition and Machine Intelligence, 1997 – 14p.
7. Richard I. Hartley, Peter Sturm. Triangulation. / R. I. Hartley, P. Sturm. – Computer vision and image understanding Vol. 68, 1997 – 12p.
8. Официальная документация Flask 0.12 [Электронный ресурс]: режим доступа - http://flask.pocoo.org/docs/0.12/
9. Официальная документация OpenCV 3 [Электронный ресурс]: режим доступа – http://docs.opencv.org/3.2.0/
10. Официальная документация NumPy 1.12 [Электронный ресурс]: режим доступа - https://docs.scipy.org/doc/numpy-1.12.0/reference/
11. Официальная документация SciPy 0.19 [Электронный ресурс]: режим доступа - https://docs.scipy.org/doc/scipy/reference/

Приложение А

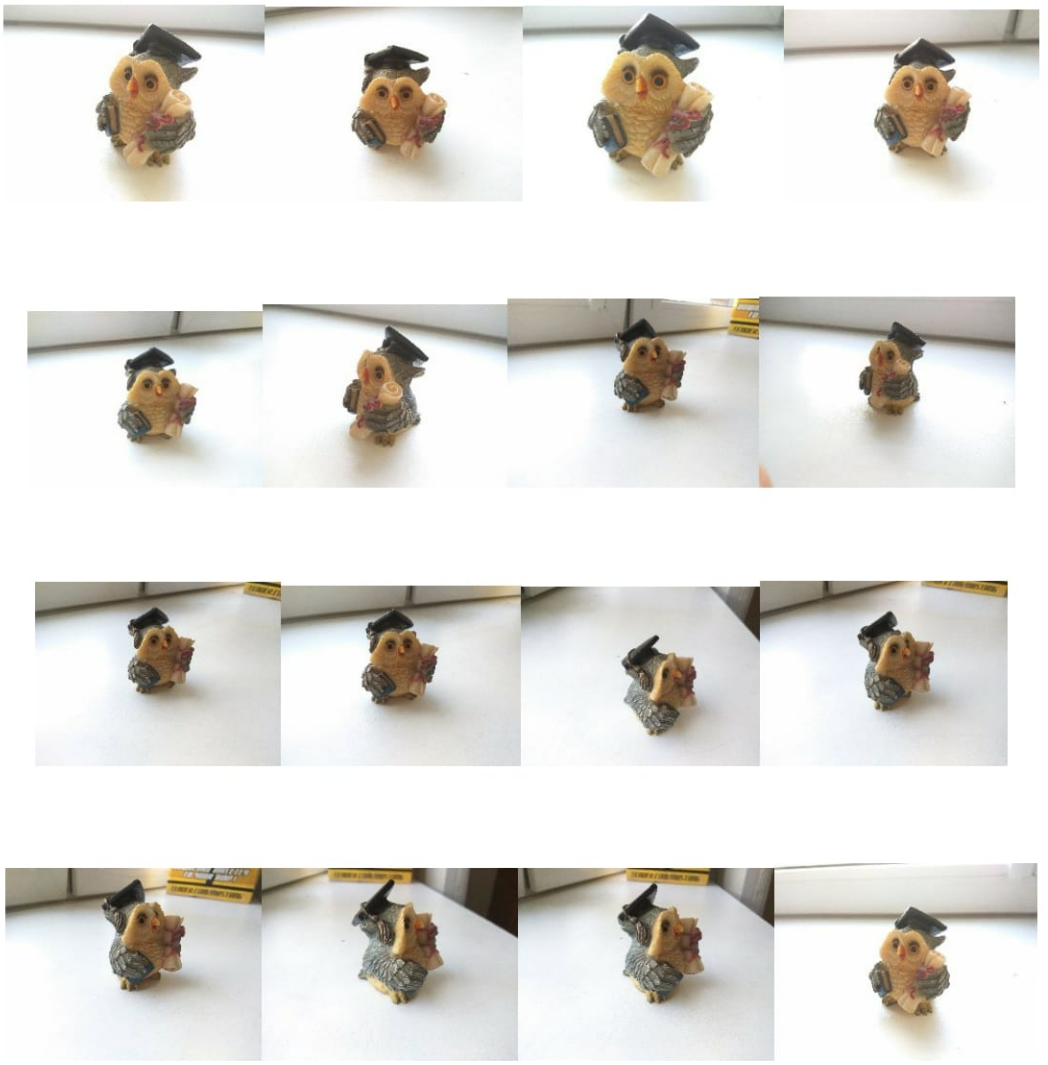

Рисунок 3.1 – набор фотографий реконструируемого объекта

Рисунок 3.2 – процесс загрузки и обработки фотографий

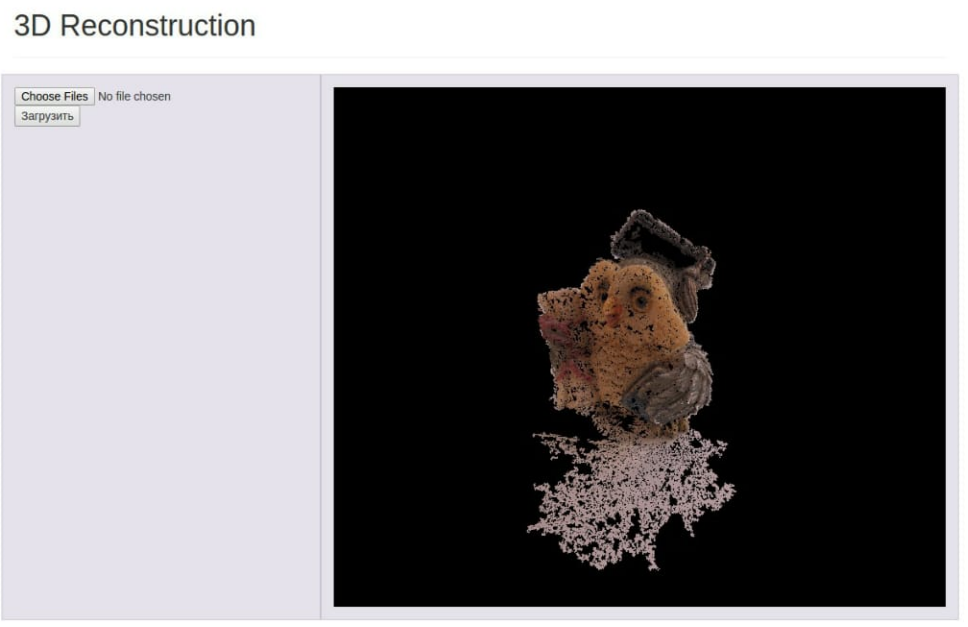

Рисунок 3.3 – результат реконструкции

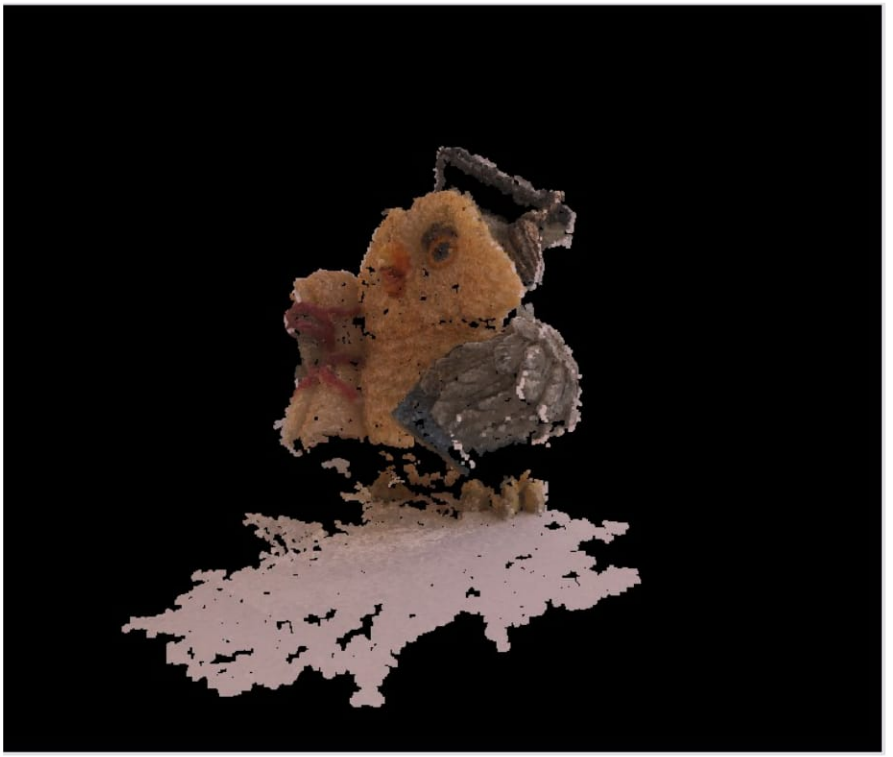

Рисунок 3.4 – результат реконструкции

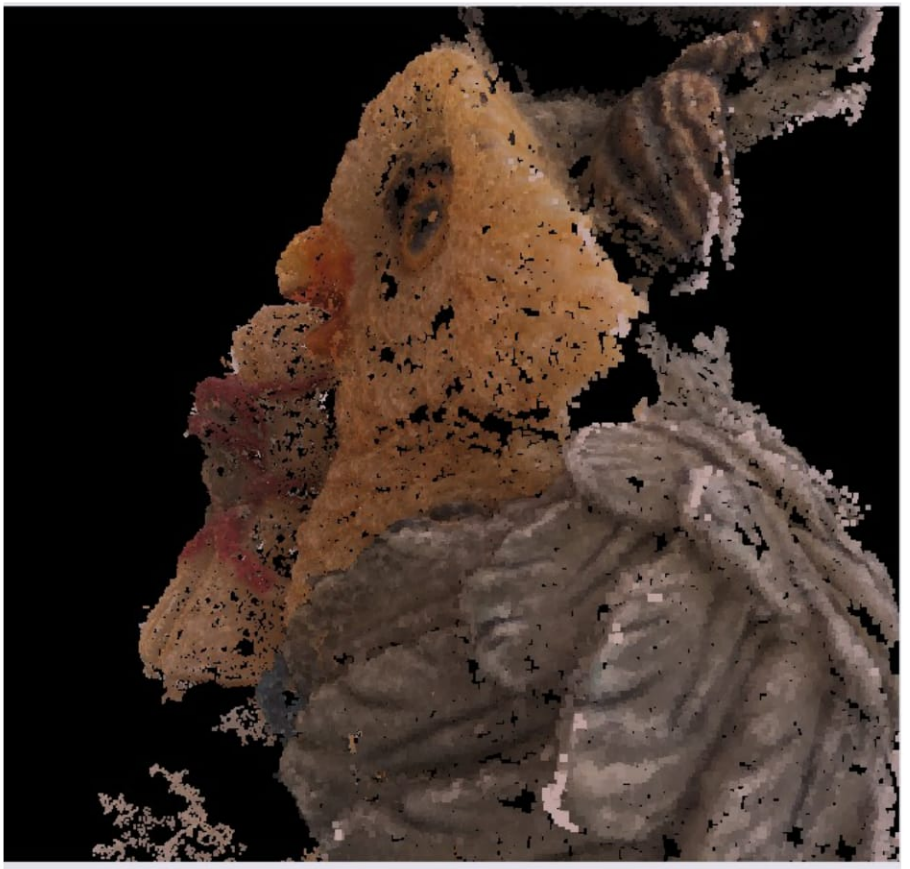

Рисунок 3.5 – результат реконструкции

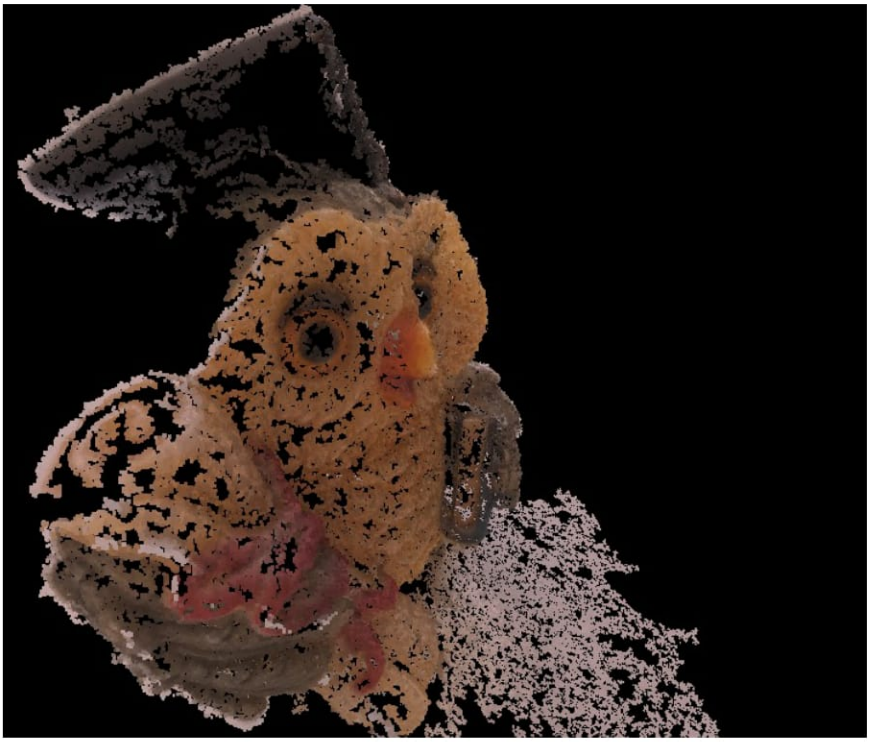

Рисунок 3.6 – результат реконструкции