Техническое задание 1

Техническое задание 2

Календарный план

РЕФЕРАТ

Расчётно-пояснительная записка 78 с., 24 рис., 2 табл., 40 ист., 1 прил.

В работе представлена разработка метода компьютерной картографии помещения на основе визуальной одометрии.

Проведён анализ предметной области и существующих методов компьютерной картографии помещения. Выявлены преимущества и недостатки каждого метода. Обозначены возможные проблемы исследования.

Разработанный метод реализуется в виде программного обеспечения, проводится его апробация.

КЛЮЧЕВЫЕ СЛОВА: картография помещений, визуальная одометрия, SLAM, MEMS, LIDAR, точки интереса, кластеризация.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Потребность в навигации и картографировании местности вокруг себя всегда была присуща человеку. Изначально составление карты требовало высокой квалификации, точных математических расчётов вручную. В настоящее же время для составления различных карт и схем местности активно используются возможности компьютеров: специальное программное обеспечение, различные камеры и датчики. Стоит отметить, что компьютерная (или цифровая) картография не является самостоятельным разделом, собственно, картографии. Она просто изменила способы визуализации данных. Если раньше это были чернила на бумаге, то сейчас это полностью экран компьютера.

С возникновением автономных роботов остро встала проблема их ориентации в пространстве (открытом или закрытом). Роботу-пылесосу, например, необходимо определить расстояние до какого-либо препятствия, на основании этого построить свой маршрут и запомнить его. Также при возникновении чрезвычайной ситуации в здании человеку нужно знать актуальную схему помещения, которая, зачастую, не всегда есть.

Целью данной работы является разработка метода картографирования помещения на основе визуальной одометрии. Для её достижения необходимо решить нижеперечисленные задачи:

- провести обзор предметной области и существующих методов картографирования помещений с помощью ЭВМ;
- разработать основные положения предполагаемого метода картографирования помещений на основе визуальной одометрии;
- спроектировать и реализовать программное обеспечение для демонстрации разработанного метода;
- провести апробацию метода картографирования помещений на основе визуальной одометрии.

1 Аналитический раздел

В данном разделе описана предметная область. Также проанализированы существующие решения и произведено их сравнение. Описаны методы локализации камеры в помещении и его картографирования, методы и алгоритмов кластеризации набора данных.

1.1 Предметная область

Картография (по Большой российской энциклопедии) — область науки, техники и производства, охватывающая создание, изучение и использование географических карт и других картографических произведений (в т. ч. карт других планет). [1] Картография изучает различные геоинформационные системы и их взаимное местоположение в пространстве, методы создания карт и других моделей. Из-за повсеместного развития компьютерных технологий картография стала толковаться как создание не только физических моделей (собственно, карт и глобусов), но также моделей компьютерных и виртуальных. К последним относят не только карты, полученные или хранящиеся в ЭВМ, но и всевозможные компьютерные анимации карт и базы данных, собираемых для их построения.

Стоит отметить, что неотъемлемой частью картографии, а вместе с ней и карты как таковой, являются различные знаки и обозначения, вводящиеся чтобы картографическую информацию можно было считывать. Именно семиотика, то есть «язык» карты, позволяет отличить один её тип от другого.

Карты могут отличаться по способу их построения, масштабу, тематике и т. д. Из них стоит рассмотреть типы самых распространённых:

— общегеографическая карта содержит основные элементы местности;

— топографическая карта содержит информацию о высотах точек на местности;

— тематические карты (климатические, экономические, биологические и т. п.);

— рельефные карты дают объёмное трёхмерное изображение поверхности;

— цифровые и электронные карты используются для формирования баз данных, а затем визуализируются различным программным обеспечением с учётом проекций и условных обозначений.

Из этого списка представляют интерес, очевидное карты цифровые и электронные, так как именно их проектируют автономные роботы и другие аппаратные и программные средства. Их примеры приведены на рисунках 1 и 2 соответственно.

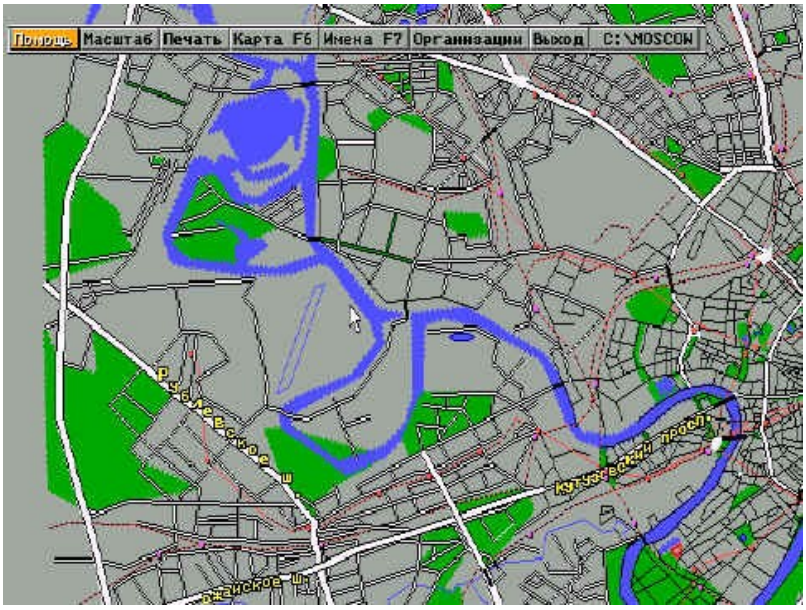

Рисунок 1 — Образец цифровой карты города Москва в специализированном ПО

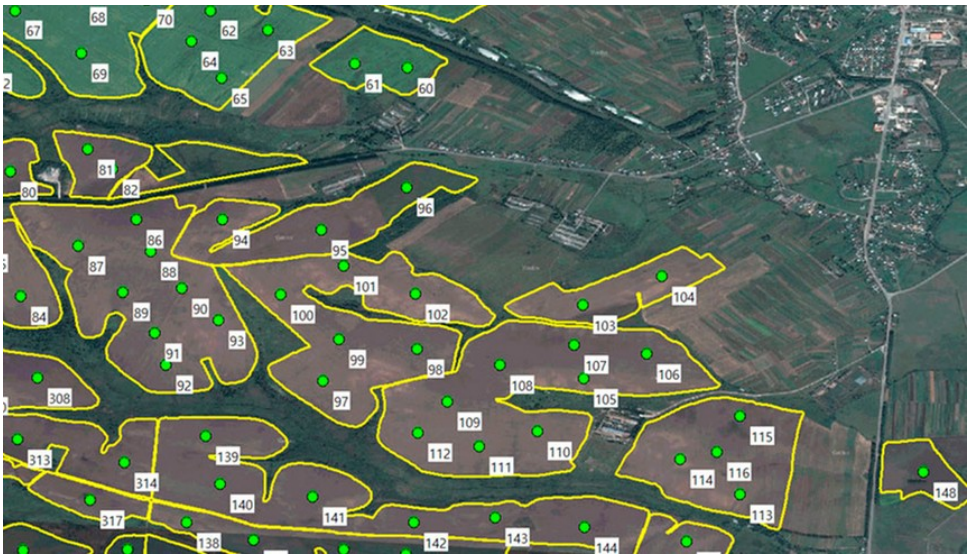

Рисунок 2 — Электронная карта сельскохозяйственных полей

Далее карты классифицируют по масштабу, то есть по тому, насколько различаются размеры построенного изображения модели и реального объекта.

— планы – 1:5000 и крупнее;

— крупномасштабные – 1:10000–1:200000;

— среднемасштабные – 1:300000–1:1000000 включительно;

— мелкомасштабные – мельче 1:1000000.

Именно карты-планы целесообразно строить для картографирования помещения или любого другого закрытого пространства.

Важными элементами карты являются не только само по себе картографическое изображение, но и её легенда. Она представляет собой таблицу со знаками, которые наносятся на модель местности, и соответствующим им понятиям.

1.2 Обзор методов локализации в помещении и его картографирования

Методы распознавания окружающей среды и локализации в ней камеры поверхностно и очень грубо можно разделить на визуальные и невизуальные. Первые основаны на визуальной одометрии (SLAM и SPLAM). В них применяется компьютерное зрение и различного типа камеры *видимого* диапазона для восстановления геометрии пространства. В невизуальных же используются датчики иного рода: MEMS-сенсоры, стационарные RFID-метки, светоотражающие маяки и т. д.

Ниже будут рассмотрены оба типа методов построения карт.

1.2.1 Визуальная одометрия

Визуальная одометрия (ВО) основана на обработке изображений, поступающих с камеры [2]. Так как эти изображения содержат информацию о таких важных характеристиках, как форма и положение объектов в пространстве и относительно себя, то данный метод позволяет позиционировать камеру в пространстве без каких-либо контактов между ним и окружающей средой.

Традиционный способ обработки изображения в ВО предполагает выделение таких ключевых элементов в кадре, как линии, углы, и их перемещение от кадра к кадру. Эта информация оценивается с помощью фильтра Кальмана. Ещё один способ обработки вычисляет яркость пикселей и оптических потоков, смещающихся кадр за кадром.

Вкупе с визуальной одновременной локализацией и сопоставлением (SLAM) датчик помимо, собственно, определения своего положения в пространстве строит карту окружения со своей траекторией [3]. В общем виде принцип работы SLAM складывается из двух этапов: определение датчиком своего местоположения, ориентируясь на своё окружение, и сверка со своей строящейся картой для её исправления. При анализировании изображения сравниваются координаты и временные характеристики. Алгоритмы, которые основаны на отслеживании и сравнении интересующих точек в кадре, обладают явным недостатком: не всегда удаётся распознать эти точки, особенно если с камеры поступает изображение низкого качества.

С математической точки зрения очередная позиция камеры рассчитывается в конце пройденной траектории. Для этого необходимы следующие данные [4]:

$$u_{\{1:t\}}=\{u_1,u_2,u_3,\dots,u_t\}, \tag{1}$$

где u — управление камеры в момент времени t,

$$z_{\{1:t\}}=\{z_1,z_2,z_3,\dots,z_t\}, \tag{2}$$

где z — информация об окружении в момент времени t, m — построенная карта, и:

$$x_{\{1:t\}}=\{x_1,x_2,x_3,\dots,x_t\}, \tag{3}$$

где x — полученное местоположение камеры в момент времени t.

В итоге формула для расчёта траектории камеры выглядит следующим образом:

$$p(x_{\{1:t\}}, m | z_{\{1:t\}}, u_{\{1:t\}}), \quad (4)$$

Преимуществом этого метода является быстрота его работы. Также обозреваемая местность, чья карта будет строиться, не требует предварительной подготовки.

Основным недостатком данного способа является отсутствие метода корректировки расчётной координаты камеры, что приводит к нарастанию погрешности локализации в процессе движения, так как данные одометрии имеют значительную погрешность. Ещё одним минусом визуальной одометрии является сильная зависимость точности карты от качества входных данных, в том числе и от разрешающей способности камеры, техногенным факторам (большое количество пыли в воздухе, низкая освещённость помещения).

1.2.2 Разновидности алгоритмов визуальной одометрии

Согласно общепринятым терминам [5], существуют следующие методы оценки параметров модели:

- косвенные;
- прямые;
- разреженные;
- плотные;
- гибридные.

В косвенных методах используются так называемые «ключевые точки» (или, «точки интереса») для выявления степени изменения положения камеры между изображениями, а равно и наличия движения. Такими точками могут послужить любые знаковые очертания объектов на снимках, будь то углы помещения, окна или другой большой предмет.

Прямые методы анализируют такую информацию, как яркость отдельных пикселей, трёхмерные координаты точки и другие фотометрические характеристики.

Отличие разреженного и плотного алгоритмов заключается в количестве анализируемых точек. Первому алгоритму требуется небольшое количество точек, в результате чего конструируется разреженная модель — каркас. В случае плотного алгоритма анализируется как можно большее количество точек изображения, что обеспечивает максимально детализированный результат. Здесь важно отметить, что каждая точка изображения рассматривается не отдельно от соседних с них, а вкупе с ними, образуя в некоторой степени, окрестность. Иначе плотная модель не будет корректна и точна в достаточной степени.

Гибридные методы в разной степени сочетают вышеописанные подходы.

1.2.3 Разновидности современных реализаций подхода SLAM

Для облегчения использования алгоритмов SLAM в большом количестве программного обеспечения (в том числе и для различного вида роботов) используются библиотеки и пакеты с открытым исходным кодом: Gmapping [6], Cartographer [7], Rtabmap [8]. Все три находят активное применение вместе с ROS — Robot Operating System (с английского — «Операционная система для роботов»).

Gmapping в качестве основного инструмента для входных данных использует LIDAR (лазерный радар). После сканирования пространства вокруг пакет определяет местоположение робота с помощью адаптивного алгоритма локализации Монте-Карло [9]. Предполагая множество возможных местоположений робота, алгоритм сопоставляет «очертания» построенной карты и данные с датчика (лидара). Таким образом множество мест, где может быть робот в конкретный момент времени, сводится к одной точке, которая, в конечном итоге, и будет считаться истинным положением робота в пространстве. Результат работы пакета представлен на рисунке 3.

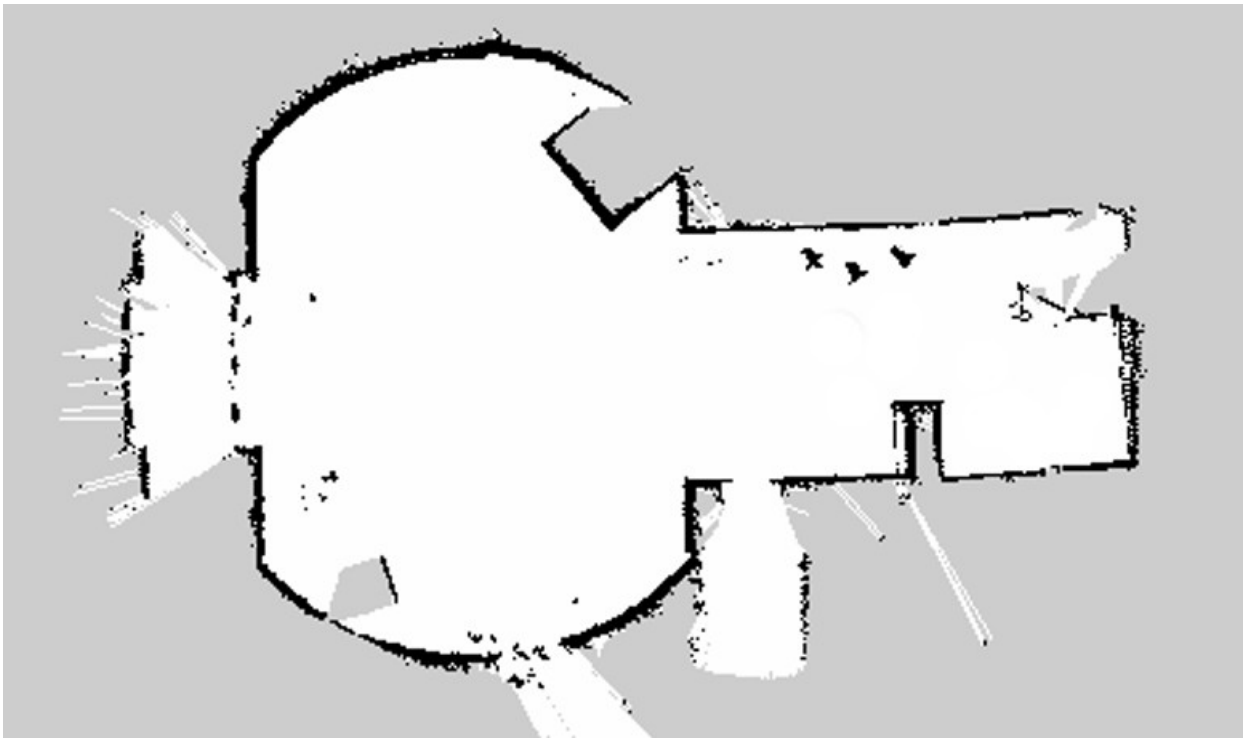

Рисунок 3 — Пример карты, построенной пакетом Gmapping

Пакет Google Cartographer схож с Gmapping в способности по данным лидара и одометрии, но обладает более широкими возможностями. Например, из-за более сложной организации строящейся карты местности — она основана на так называемых «ячейках» — вероятность пропуска какого-либо объекта на карте значительна меньше, чем у аналога. Пример готовой карты показан на рисунке 4.

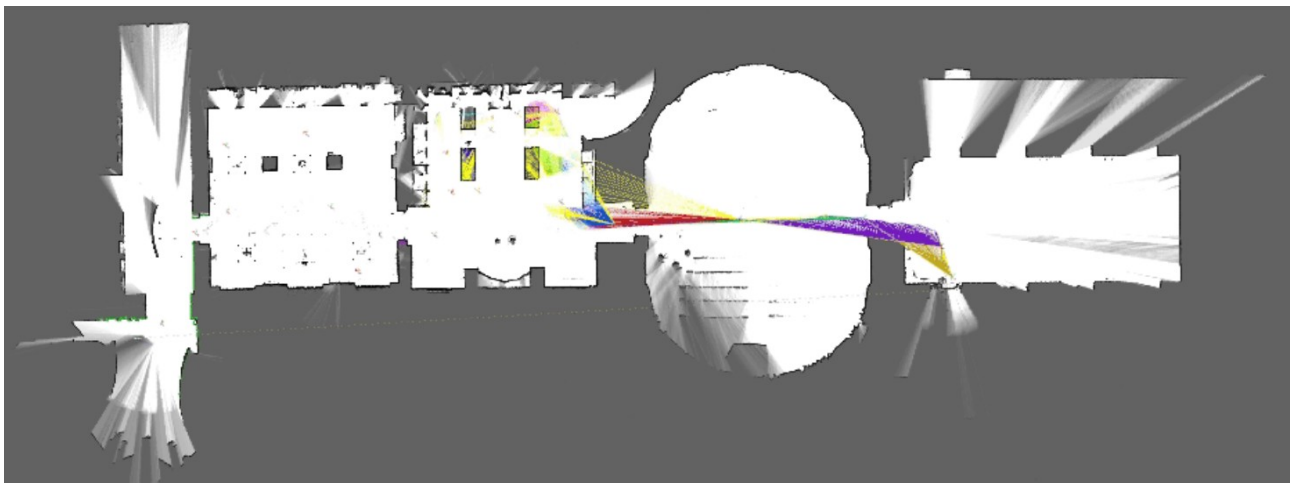

Рисунок 4 — Карта Cartographer

Основная особенность данного метода заключается в том, что он состоит из локального SLAM и глобального SLAM. Локальный SLAM создаёт отдельные небольшие части карты. Затем глобальный SLAM-алгоритм ищет совпадения между частями карт, а также между данными сенсоров и всеми картами, созданными при помощи локального SLAM в параллельных процессах. В результате глобальный SLAM формирует единую карту окружающего пространства.

Rtabmap основан на поиске и сопоставлении соответствий данных сенсоров с использованием памяти, в которой хранится база данных визуальных образов в соответствии с данными о местоположении робота. На каждом поступающем изображении выделяются ключевые точки, это же изображение сопоставляется с текущим местоположением робота и с другими уже имеющимися изображениями. Так составляется база данных визуальных образов и координат, где они были получены. Сравнивая образ с хранящимися в базе, вычисляется их коэффициент соответствия. При высоком значении оного будет считаться, что робот уже был в этом месте и, при необходимости, скорректирует одометрию. Всё это нивелирует накапливающуюся ошибку по мере роста базы данных изображений и передвижения робота.

Стоит также отметить широкую совместимость Rtabmap с различными типами сенсоров (лидар, стерео- и RGBD-камеры) и одометрий (встроенная, одометрия колёс и т. д.).

Google Cartographer обладает большей гибкостью в плане настроек и выбора сенсоров. Возможность использования не только лидаров, но также камер глубины позволяет получить большее количество информации. Также, Cartographer позволяет без проблем составлять карты больших по площади помещений.

В отличие от GMapping и Cartographer, SLAM библиотеки Rtabmap строит не только 2D-карту, но и создаёт базу данных визуальных образов. Таким образом, Rtabmap ищет глобальные совпадения по визуальным образам, в то время как составленная карта помещения может использоваться лидаром для локальной корректировки местоположения робота в пространстве.

1.2.4 Метод бинокулярного зрения

В данном методе робот строит карту помещения и решает задачу локализации, используя особую систему бинокулярного зрения (она представлена на рисунке 5).

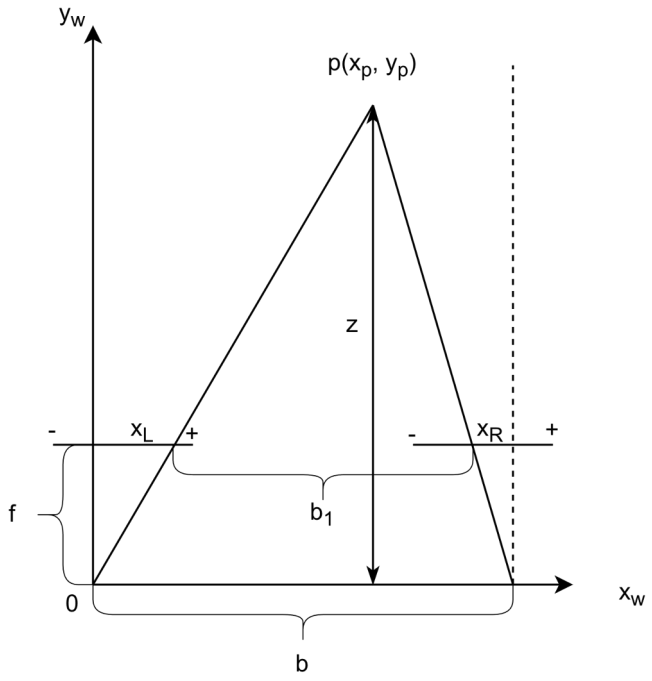

Рисунок 5 — Система бинокулярного зрения

Для этого используются одновременно две камеры, с помощью которых вычисляются координаты точки пространства, расположенной на некотором расстоянии. Таким образом, достаточно найти такой фрагмент уже имеющейся

карты, который соответствует местности, через взаимно корреляционную функцию. [10]

Говоря более подробно, первоначальные координаты положения камеры должны быть известны. Затем он с помощью обеих камер получает так называемое «облако точек» — координаты точек всего препятствия перед ним. Это может быть стена, стул или любой другой предмет интерьера неизвестного заранее помещения. Расстояние до некоторой точки в системе координат камер вычисляется по следующей формуле.

$$\begin{cases} \frac{z}{b}=\frac{z-f}{b_1}, \\ b_1=b-x_L+x_R, \end{cases} \tag{5}$$

где:

— z — расстояние от точки пространства P до оптической оси телекамер;

— f — фокусное расстояние объективов;

— b — оптическая ось телекамер, т. е. базы системы;

— x_L и x_R — координаты левого и правого изображений.

Учитывая параллакс точки $d=x_L-x_R$, получаем:

$$z=\frac{bf}{d}=\frac{bf}{x_L-x_R}. \tag{6}$$

Но начало системы рассматриваемой системы координат совпадает с оптической осью левой камеры, и оно смещено на половину относительно центра камеры, то в итоге координаты точки (x, y) в системе бинокулярной камеры следующие:

$$\begin{cases} y=\frac{bf}{d}, \\ x=\frac{-b}{2}-\frac{x_L y}{f}. \end{cases} \tag{7}$$

Для корректного преобразования координат при перемещении камеры в расчётах используется несколько матриц.

1. Матрица вращения вида

$$R=\begin{bmatrix} -\sin\theta & \cos\theta & 0 \\ \cos\theta & \sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}; \tag{8}$$

2. Матрица сдвига

$$S=\begin{bmatrix} \Delta x \\ \Delta y \\ 0 \end{bmatrix}; \tag{9}$$

3. Матрица преобразования

$$T=\begin{bmatrix} R & S \\ 0 & 1 \end{bmatrix}. \tag{10}$$

Здесь угол θ соответствует углу поворота вокруг оси z при смещении камеры вдоль оси Δx и Δy.Однако окончательная формула преобразования между мировой системой координат и системой координат камеры сформируется после того, как мы примем во внимание вектор L координат точки P в системе координат камеры C и вектор L' координат точки в мировом пространстве W.

В итоге:

$$\begin{bmatrix} L' \\ 1 \end{bmatrix}=\begin{bmatrix} R & S \\ 0 & 1 \end{bmatrix}\times\begin{bmatrix} L \\ 1 \end{bmatrix}\triangleq T\times\begin{bmatrix} L \\ 1 \end{bmatrix}. \tag{11}$$

Что же касается описания полученного облака точек, то для этого применяется метод наименьших квадратов. Стена помещения хорошо описывается линейная модель $y=kx+b$.

Её параметры рассчитываются по следующей формуле:

$$\begin{cases} k=\dfrac{\sum\limits_{i=1}^{n} x_i y_i - n\,\overline{x}\,\overline{y}}{\sum\limits_{i=1}^{n} x_i^2 - n(\overline{x})^2}, \\ b=\overline{y}-k\,\overline{x}, \end{cases} \tag{12}$$

где $\overline{x}$, $\overline{y}$ — среднее значение параметров.

Переход к взаимно корреляционной функции обусловлен необходимостью представления облака точек дискретно. То есть карта помещения состоит из облака точек $[A_1, A_2, \ldots, A_N]$, а неизвестный фрагмент карты — $[B_1, B_2, \ldots, B_M]$. И чтобы определить, где находится на карты наблюдаемое пространство, оценивается степень совпадения точек обоих массивов данных. Максимально значение этого коэффициента (в формуле 6) корреляции и будет однозначно определять положение нового фрагмента на карте помещения.

$$\max R_i(t) = \max \int_{i}^{i+M-1} \varphi_1(A_i)\varphi_2(B_i)\,di, i \in [1, N-M+1], \tag{13}$$

где:

— φ_1 — карта помещения;

— φ_2 — фрагмент стены, которую «наблюдает» камера.

Данный метод решения локализации мобильного робота достаточно универсален, так как не зависит от вида картируемой местности, обладает высокой точностью. Также нельзя не отметить использование лишь одной

бинокулярной камеры. Из минусов можно отметить весьма затратные с точки зрения времени вычисления математические операции.

1.2.5 MEMS-сенсор

Метод SLAM предполагает навигацию и составление маршрута робота посредством высокоточных камер и лазеров. К сожалению, они достаточно дорогие и энергозатратные, что вынуждает искать альтернативные решения. Одним из таких является ориентирование по датчику магнитного поля Земли MEMS (Micro Electro Mechanical System).

Реализация алгоритма с MEMS экспериментального подвижного робота подробно описана в [11]. В данной работе предложена и реализована система навигации мобильного робота на основе смартфона Samsung GT-S7710, оснащённого сенсором MEMSHSCDTD008A [12], с помощью которого была построена магнитная карта помещения и оценена возможность навигации по ней.

В картируемом помещении в определённых точках были проведены измерения геомагнитного поля. Затем полученные замеры были представлены в формате JSON и записаны на карту памяти смартфона для обработки специальным ПО. В течение эксперимента мобильный робот с установленным на борту сенсором двигался по заранее известному маршруту, а данные его замеров затем были сопоставлены и после предобработки наложены на карту с учётом ориентации. В результате, изменения величины магнитного поля H по осям X, Y и Z повторяют его реальную траекторию. Таким образом, в результате была показана возможность использования магнитного поля Земли для навигации внутри помещений.

Очевидно, что данный метод ориентирован на невизуальные данные, получение которых не зависит от таких девайсов, как камеры или лазерные дальномеры. Они дорогостоящие, сильно нагреваются и шумят при работе. По магнитному же полю Земли успешно ориентируются многие представители животного мира.

Недостаток данного метода — предварительная подготовка местности: измерение в выбранной точке магнитного фона, который может отличаться в разных областях земной поверхности из-за достаточно неочевидных факторов, например залежей магнитных руд. Магнитные поверхности и намагниченные предметы также сильно влияют как на предварительные измерения, так и на ориентацию самого робота.

1.2.6 Комбинированный

Необходимо подчеркнуть, что способы локализации робота с использованием визуальной одометрии обладает существенным недостатком. Нарастающую погрешность в определении координат зачастую трудно исправить. Для повышения точности определения местоположения робота через его координаты могут использоваться специальные метки RFID.

Метка RFID (англ. radio frequency identification) — двухкомпонентное изделие из интегральной схемы для обработки информации и преобразования радиосигнала и антенны для приёма и передачи этого сигнала. [13]

В помещении заранее расставляются светоотражающие маяки. С помощью них затем будет производиться ориентация робота и построение статической карты. Методом SLAM (simultaneous localization and mapping — одновременная локализация и картирование) вычисляется первоначальная координата местоположения робота, а RFID-метки позволяют скорректировать её. После этого происходит детектирование контуров объектов камерой глубины и построение подкарты препятствий в конкретной позиции, что держит в актуальном состоянии информацию о слоях построенной карты в текущей позиции. На этом этапе применяются LIDAR (лазерный радар) и дальномеры на задней части корпуса робота для привязки его позиции к уже пройденному маршруту. Они же формируют трёхслойную структуру карты. Когда же робот переместится на определённого расстояние от предыдущей координаты, строится новая подкарта на основе карт препятствий, статической и глобальной.

Такая совокупность технических средств нужна для постоянной корректировки и обновления пройденного маршрута и карты помещения, так как методы построения карт с использованием SLAM, RFID или LIDAR по отдельности приводит к значительным погрешностям.

Комбинированный метод достаточно надёжен. Он использует различные методы, основанные на относительно независимых входных данных: SLAM, RFID и LIDAR, — что повышает точность определения координат робота.

Однако снова необходима предварительная подготовка местности. Для корректной ориентации робота требуется расстановка специальных светоотражающих маяков. Нельзя не отметить большое количество камер, нужных для многократной перепроверки выходных данных на каждом этапе алгоритма.

1.2.7 SPLAM-алгоритм с применением графов

Simultaneous Planning, Localization and Mapping (SPLAM) — расширение метода SLAM (simultaneous localization and mapping), позволяющее однозначно локализовать робота в пространстве и построить карту этого пространства, но и корректно объединять её с топологическими картами большего масштаба. [14]

Метод преобразует сохраняет ранее собиравшуюся информацию о положении робота для составления карты, а затем преобразует её элементы в ограничения и последовательность так называемых «поз». В результате строится граф поз (pose graph). [15] Его пример показан на рисунке 6.

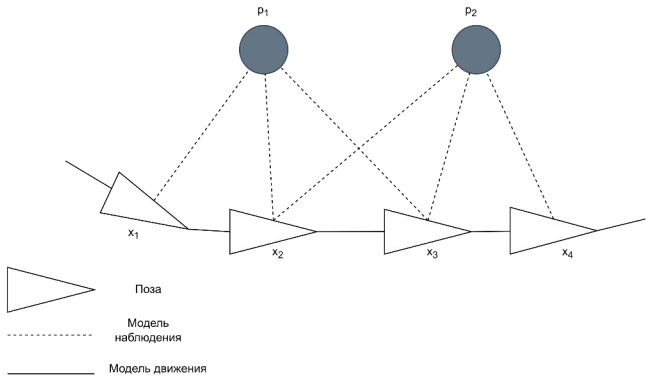

Рисунок 6 — Граф позы алгоритма SPLAM с оптимизацией графа

Рассматриваемый метод имеет структуру, разделённую на две части: внутреннюю и внешнюю. Внешняя часть отвечает за построение графа (посредством сканирования и обнаружения замкнутого контура), а внутренняя — за, собственно, оптимизацию. Оптимизация графа поз необходима, так как информация, поступающая с датчиков, не всегда точна и обладает ошибкой, а значит, не согласована.

Для создания интерфейса оптимизации графа устанавливается связь между узлами поз через лазерное сканирование. Лазерное сканирование сопоставляет позы либо между двумя последовательными кадрами, либо между одним кадров и картой. Затем, после обнаружения относительных изменений кадры объединяются. [16]

Замкнутый цикл (или петля) представляет собой ранее уже посещённую роботом местность. Считать, что подобная петля обнаружена, можно тогда, когда два набора точек из входного облака совпадают. В этом случае граф поз подвергается перестроению, что значительно его упрощает, сокращает время обращения к нему и уменьшает выделенную под него память.

Главным преимуществом алгоритма SPLAM с оптимизацией графа является экономия используемой памяти и уменьшение количества обрабатываемой информации и увеличение быстродействия. Также не требуется подготовка картируемой местности.

1.3 Критерии сравнения методов локализации и картографирования помещения

Очевидно, что все вышеперечисленные методы локализации автономного робота (строго говоря, лишь камеры) и картографирования на основе его перемещений помещения в тех или иных ситуациях на практике бывают не всегда применимы в чистом виде. Очень часто приходится выбирать между высокой точностью определения местоположения робота и предварительными действиями для подготовки картируемой местности и относительно быстрой работой самого алгоритма локализации, но неточностью построенной этим алгоритмом картой. Также принимают во внимание побочные факторы от

использования технических устройств, установленных на роботе для решения поставленной задачи, такие как стоимость необходимого оборудования, его вес. Чтобы узнать среднюю стоимость оборудования для каждого из методов, была получена выборка из 10 предметов по запросам «веб-камеры», «промышленные, бинокулярные камеры», «лидары», «RFID» в трёх профильных российских интернет-магазинах (`dns-shop.ru`, `onpad.ru`, `baslerweb.com`), а затем высчитана их средняя цена. Для метода визуальной одометрии средняя стоимость веб-камеры составила 4029 руб. 20 коп. Средняя стоимость бинокулярной промышленной камеры — 30890 руб., в то время как стоимость всей необходимой аппаратуры (камера глубины, радиометки, лидар) для комбинированного метода в среднем — 133468,15 руб.

Для наглядности в таблице 1 приведены особенности рассмотренных ранее способов решения задачи определения местоположения и картографирования помещений.

Таблица 1 — Сравнение алгоритмов по выделенным критериям

Метод	Требование подготовки помещения	Входные данные	Средняя цена оборудования, руб.	Вес оборудования, г
Метод визуальной одометрии	Нет	Последовательность изображений	4029,20	От 30 до 200
Метод с использованием бинокулярной камеры	Нет	Координаты точек препятствий	30890	От 350 до 1000
Комбинированный метод	Да (расстановка светоотражающих маяков и RFID-меток)	Данные от RFID-сенсоров, от лидаров, RGB-D-камеры	133468,15	От 1000 и больше

1.4 Возможные проблемы

Вполне естественно, что задача построения схемы помещения на основе визуальной одометрии сопряжена со множеством трудностей. Точность считывания информации с датчиков (например, камеры) зависит её разрешения, фокуса, различных фильтров. Положение камеры может рассчитаться некорректно на любом этапе построения карты помещения, если окружение измениться и прежние ориентиры будут каким-либо образом изменены. Поэтому необходимо конкретизировать и ограничить набор задач, которые будут решены в этой выпускной квалификационной работе.

В данной работе будет рассмотрено построение двумерных схем-карт помещения с проекцией объектов интерьера на плоскость пола. Это направление было выбрано в соответствие с несколькими соображениями. Во-

первых, построение трёхмерных карт уже реализовано и не без применения различных «продвинутых» средств, зачастую трудно доступных в обычных бытовых условиях. Во-вторых, построение трёхмерных карт, которые гораздо более информативны, чем двумерные, требует большое количество места для хранения на ЭВМ (телефоне или компьютере). В-третьих, полученные двумерные карты-схемы имеют применения не только в виде цифровых данных, в отличие от трёхмерных вариантов. Например, такую карту можно использовать в качестве схемы эвакуации из помещения.

В таком случае, непосредственно к теме картирования помещения относятся вопросы о порядке входных изображений, о том, с какого ракурса сняты объекты на них. Для всесторонней прорисовки объектов необходимо несколько раз пройти одним и тем же маршрутов, фотографируя объект. Поэтому, важно распознавать уже отснятые участки, корректировать их. Предполагается, что окружение, в котором находится камера замкнутое, поэтому о наличии четырёх «несущих» стен системе известно.

Однако, проблемы разрешающей способности фотооборудования, фокусировки кадра и всего, что касается качества входных изображений, останутся за пределами обсуждений. Будет считаться, что изображения, подающиеся на вход алгоритма обладают достаточным разрешением, должным образом сфокусированы, а также не имеют искажений и какой-либо цветокорректировки.

1.5 Обзор методов кластеризации набора данных

Алгоритмы кластеризации используются для различных задач, например, обнаружение аномалий, выбросов в данных, сжатие данных, обнаружение структуры данных, восстановление недостающих данных и т. д. Некоторые реальные приложения кластеризации обнаруживают мошенничества в страховании, сегментируют клиентов, проводят анализ землетрясений или городского планирования. Алгоритмы кластеризации используются в биологии, например, для описания и проведения пространственных и временных сравнений комплексов организмов.

Дальнейший обзор метрик, подходов к кластеризации приведён в соответствии с [17].

1.5.1 Обзор метрик

Кластеризуя множество объектов, исследователь разбивает эти объекты на группы (кластеры) по принципу «схожести», обычно это расстояние. Есть большое количество определить метрику расстояния, однако основные из них:

— евклидово расстояние — геометрическое расстояние в многомерном пространстве ρ между точками x и x':

$$\rho(x,x')=\sqrt{\sum_{i}^{n}(x_i-x'_i)^2}; \tag{14}$$

— квадрат евклидова расстояния. Применяется для придания большего веса более отдалённым друг от друга объектам:

$$\rho(x,x')=\sum_{i}^{n}(x_i-x'_i)^2; \tag{15}$$

— манхэттенское расстояние. Эффект от применения этой метрика зачастую ничем не отличается от расстояния Евклида, однако для этой меры влияние отдельных больших разностей уменьшается (т. к. они не возводятся в квадрат):

$$\rho(x,x')=\sum_{i}^{n}|x_i-x'_i|; \tag{16}$$

— расстояние Чебышёва. Полезна, если нужно определить два объекта как «различные», если они различаются по какой-либо одной координате:

$$\rho(x,x')=\max(|x_i-x'_i|); \tag{17}$$

— степенное расстояние. Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Его формула:

$$\rho(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x'_i)^p}, \tag{18}$$

где параметр p ответственен за постепенное взвешивание разностей по отдельным координатам, r ответственен за прогрессивное взвешивание больших расстояний между объектами.

1.5.2 Обзор алгоритмов кластеризации

Иерархические алгоритмы строят систему вложенных друг в друга кластеров, образуя структуру дерева. Среди алгоритмов иерархической кластеризации выделяются два основных типа: восходящие и нисходящие алгоритмы. Нисходящие алгоритмы работают по принципу «сверху вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на всё более мелкие кластеры. Более распространены восходящие алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во всё более крупные, пока все объекты выборки не будут содержаться в одном кластере. Пример входных данных для иерархического алгоритма и дендрограмма (дерево кластеров) изображены на рисунке 7.

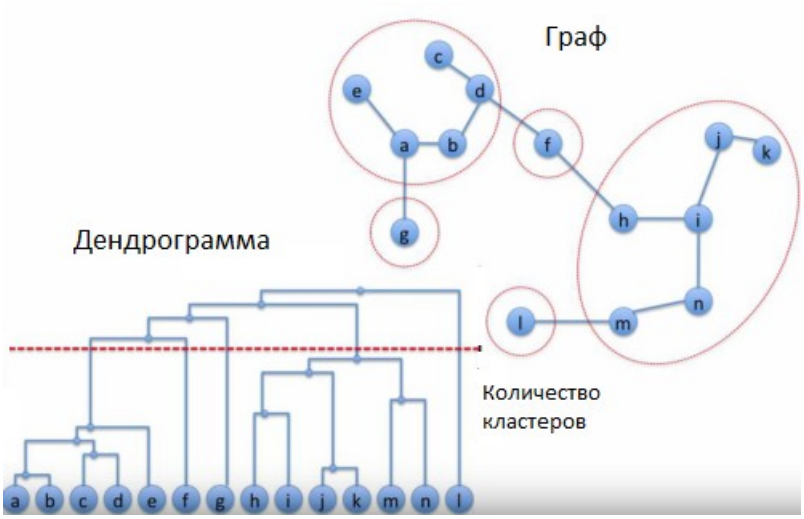

Рисунок 7 — Дендрограмма данных, кластеризованных с помощью иерархического алгоритма.

Недостатками иерархических алгоритмов является получившаяся полная система разбиений, не всегда подходящая для решаемой задачи.

Плоские алгоритмы кластеризации подразумевают собой организацию кластеров на одном уровне, без вложений. К такой категории алгоритмов относится алгоритмы квадратичной ошибки, а именно — метод k-средних. Вообще, минимизация среднеквадратичной ошибки отвечает требованию построения оптимального количества кластеров:

$$e^2(X, L) = \sum_{i=1}^{k} \sum_{j=1}^{N_i} \|x_j - c_i\|^2, \qquad (19)$$

где c_i — центроиды j-того кластеров; k — количество кластеров; N_i — количество объектов в кластере; x_j — объекты в кластере.

К минусам метода k-средних можно отнести необходимость заранее знать количество кластеров.

Чёткие (или непересекающиеся) алгоритмы каждому объекту выборки ставят в соответствие номер кластера, т. е. каждый объект принадлежит только одному кластеру. Нечёткие (или пересекающиеся) алгоритмы каждому объекту ставят в соответствие набор вещественных значений, показывающих степень отношения объекта к кластерам. Таким образом, каждый объект принадлежит к каждому кластеру с некоторой вероятностью. К непересекающимся алгоритмам относятся вышерассмотренные иерархический и k-средних. Одним из

представителей пересекающихся алгоритмов является алгоритм c-средних. Этапы работы данного алгоритма:

1. Выбрать начальное нечёткое разбиение n объектов на k кластеров путём выбора матрицы принадлежности U размера $n \times k$.
2. Используя матрицу U, найти значение критерия нечёткой ошибки $\mathrm{E}^2(X,U)=\sum_{i=1}^{N}\sum_{k=1}^{K} U_{ki}\|x_i^{(k)}-c_k\|^2$, где c_k — центроид нечёткого кластера k.
3. Перегруппировать объекты с целью уменьшения значения критерия нечёткой ошибки.
4. Возвращаться в п. 2 до тех пор, пока изменения матрицы U не станут незначительными.

Для работы алгоритма требуется количество выделяемых кластеров, что может не удовлетворять некоторым задачам.

Таблица 2 — Сравнение алгоритмов кластеризации

Алгоритм кластеризации	Форма кластеров	Входные данные	Вычислительная сложность	Результаты
Иерархический	Произвольная	Число кластеров или порог расстояния для усечения иерархии	$\mathrm{O}(n^2)$	Бинарное дерево кластеров
k-средних	Гиперсфера	Число кластеров	$\mathrm{O}(nkl)$, где k – число кластеров, l — число итераций	Центры кластеров
c-средних		Число кластеров, степень нечёткости		Центроиды, матрица принадлежности

Исходя из задачи построения двумерной карты помещения целесообразно выбрать иерархический алгоритм кластеризации, использующий евклидово

расстояние в качестве метрики, так как нужно учитывать вложенность контуров предметов в пространстве.

1.6 Диаграмма проектируемого метода

На рисунке 8 представлена IDEF0-диаграмма метода картографии помещений на основе визуальной одометрии.

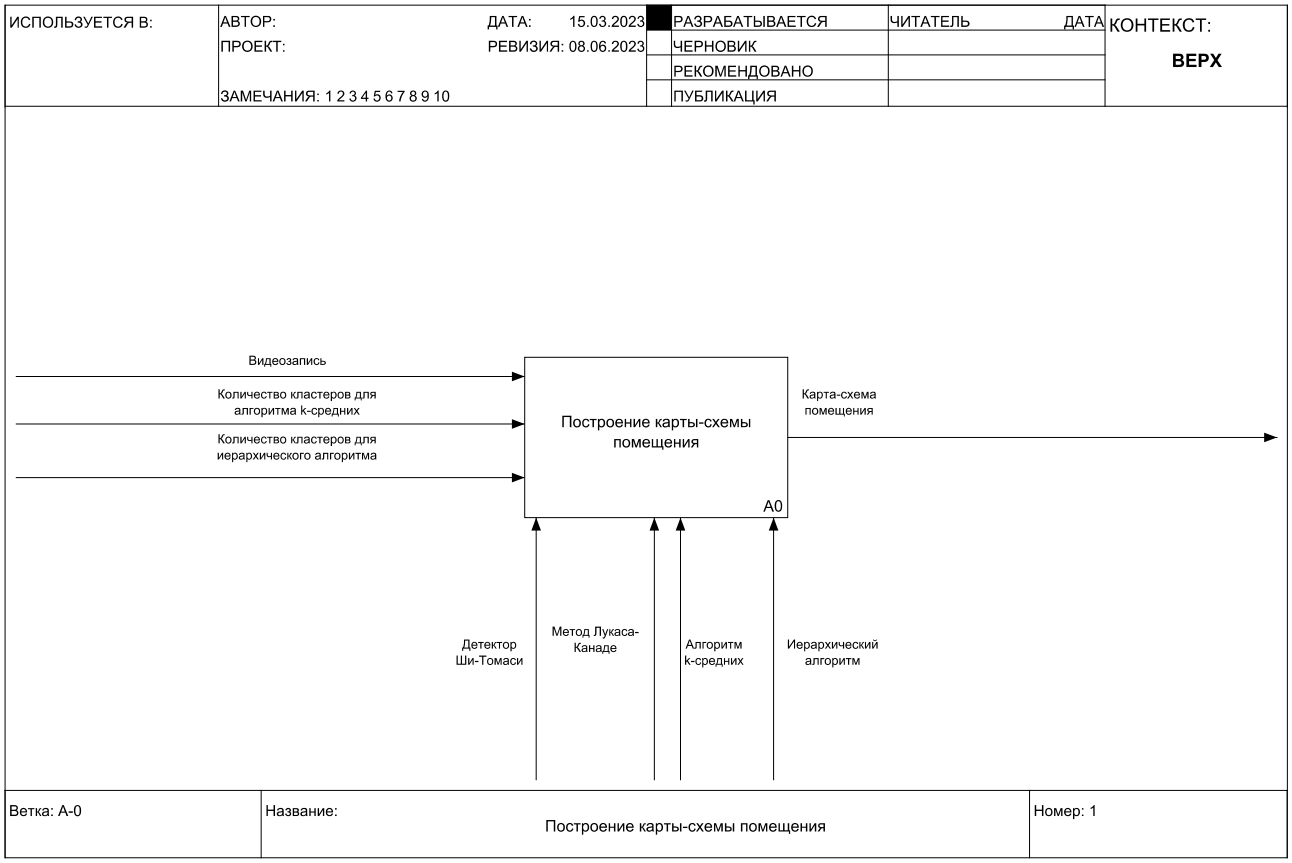

Рисунок 8 — Задача построения карты помещения

1.7 Выводы

В данном разделе была описана предметная область. Также были проанализированы существующие решения и произведено сравнение описанных решений: методов локализации камеры в помещении и его картографирования, критерии сравнения этих методов. Был приведён сравнительный анализ методов и алгоритмов кластеризации набора данных.

2 Конструкторский раздел

В данном разделе приведены IDEF0-диаграммы, проведён анализ входных данных, выходных данных и их ограничений, требуемая функциональность метода, особенности его реализации.

2.1 Диаграмма IDEF0

На рисунках 9 и 10 изображены этапы построения карты помещения в виде диаграммы IDEF0.

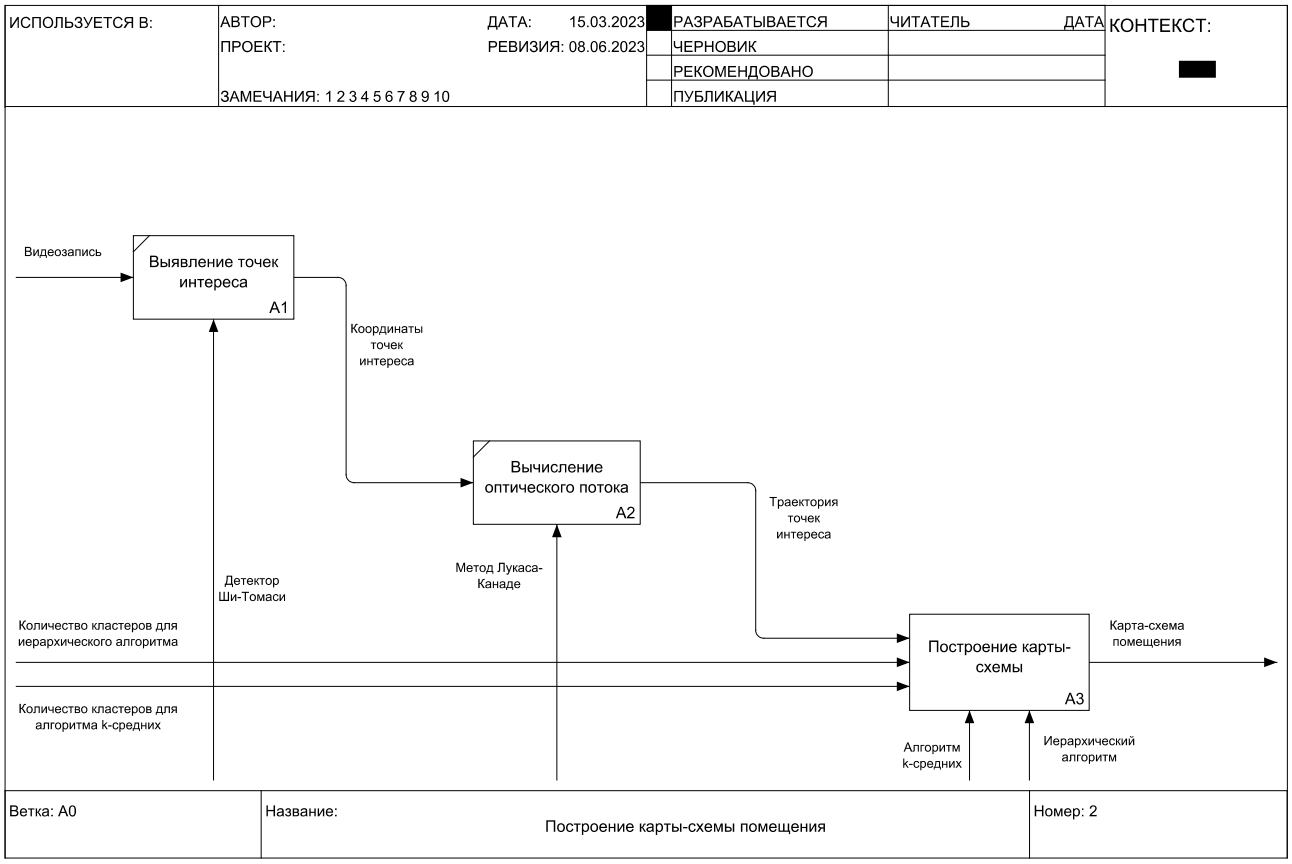

Рисунок 9 — Этапы построения карты помещения

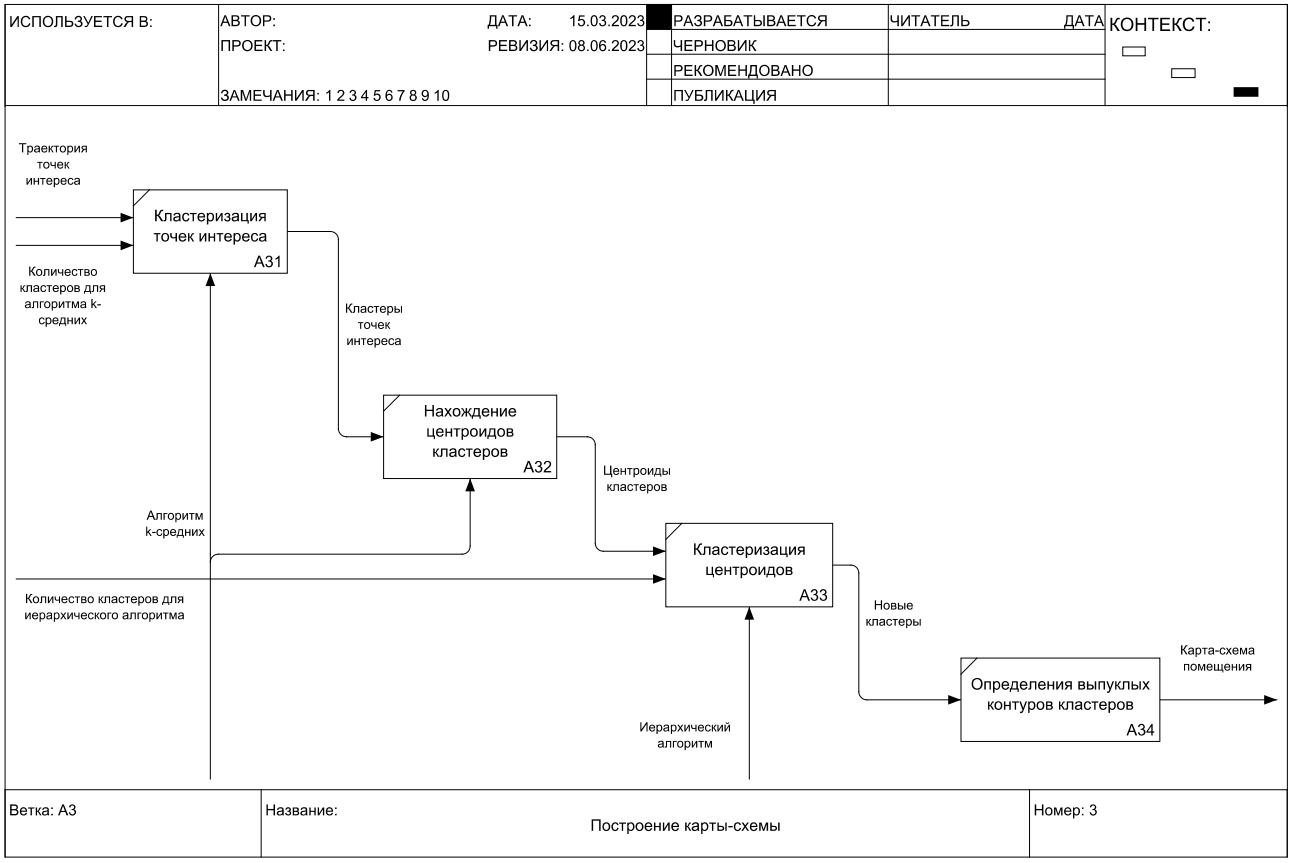

Рисунок 10 — Этапы формирования карты помещения

2.1.1 Выявление точек интереса

Чтобы отслеживать характерные признаки кадра и его индивидуальные особенности, на нём выделяются так называемые точки интереса, или «углы». В «углах», где определяется граница между двумя объектами в кадре, интенсивность пикселя меняется в пределах заданной окрестности относительно её центра, таким образом, в области вокруг угла градиент изображения можно определить через частные производные двух его направлений и изменение интенсивности.

Для решения задачи данной задачи выделяется несколько этапов [18]:

1. Вычисление собственных чисел и собственных векторов блоков изображения. Для каждого пикселя p определяется некоторая окрестность $S(p)$ заданного размера. Затем вычисляется ковариационная матрица производных функции яркости изображения I по координатам x и y:

$$M = \begin{bmatrix} \sum_{S(p)} \left(\frac{dI}{dx}\right)^2 & \sum_{S(p)} \frac{1}{dxdy} \\ \sum_{S(p)} \frac{1}{dxdy} & \sum_{S(p)} \left(\frac{dI}{dy}\right)^2 \end{bmatrix}, \quad (20)$$

где: x и y — координаты пикселя в системе координат кадра. Из матрицы M рассчитывается минимальное собственное значение, называемое картой точности $\mathrm{Q}(x, y)$. Каждую точку такого изображения называют коэффициентом точности *qualityLevel*.

2. Подавляются немаксимумы изображения, причём локальные максимумы в окрестности 3×3 сохраняются.
3. Углы с собственным значением меньше, чем произведение $qualityLevel \cdot \max_{x,y}(\mathrm{Q}(x, y))$, отбрасываются. Здесь *qualityLevel* — заданный коэффициент точности.
4. Оставшиеся углы (они же точки интереса) сортируется по соответствующим им коэффициентам точности в порядке убывания.
5. Отбрасывается каждый угол, для которого существует более точный угол на расстоянии меньше заданного минимального.

Таким образом формируется вектор координат углов (точек интереса) исходного изображения.

2.1.2 Вычисление оптического потока

Для решения задачи нахождения оптического потока широко используется дифференциальный алгоритм Лукаса — Канаде. Он позволяет найти смещение каждого пикселя между двумя кадрами с помощью частных производных по двум направлениям изображения. [19]

Допустим, что в окрестности пикселя значение оптического потока практически неизменно, то есть значения пикселя переходят из одного кадра в другой без каких-либо изменений. Тогда возможно записать функцию яркости пикселя I от его координат и времени:

$$I(x,y,t)=I(x+u_x,y+u_y,t+1), \tag{21}$$

где: x и y — координаты пикселя в системе координат кадра, t — номер кадра в последовательности, причём «расстояние» между кадрами равно единице. Таким образом, вектор оптического потока (V_x,V_y) пикселя p равен:

$$\begin{cases} I_x(q_1)V_x+I_y(q_1)V_y=-I_t(q_1) \\ I_x(q_2)V_x+I_y(q_2)V_y=-I_t(q_2), \\ \cdots \\ I_x(q_n)V_x+I_y(q_n)V_y=-I_t(q_n) \end{cases} \tag{22}$$

где q_1, q_2, …, q_n — пиксели внутри окна с центром в пикселе p, а $I_x(q_i)$, $I_y(q_i)$, $I_t(q_i)$ — частные производные изображения по координатам и времени в точке q_i . Переписав (19) в матричной форме и применив метод наименьших квадратов, получим вектор оптического потока:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix}=\begin{bmatrix} \sum\limits_i I_x(q_i)^2 & \sum\limits_i I_x(q_i)I_y(q_i) \\ \sum\limits_i I_x(q_i)I_y(q_i) & \sum\limits_i I_y(q_i)^2 \end{bmatrix}^{-1}\begin{bmatrix} -\sum\limits_i I_x(q_i)I_t(q_i) \\ -\sum\limits_i I_y(q_i)I_t(q_i) \end{bmatrix}. \tag{23}$$

Одним из недостатков обычного алгоритма Лукаса — Канаде является его *локальность*, то есть невозможно определить смещение пикселей внутри таких участков кадра, размер которых больше окрестности пикселя.

Пирамидальный вариант этого алгоритма призван устранить эту проблему [20]. При обработке изображения размерности $N\times N$ строится так называемая гауссовская пирамида изображений — упорядоченное множество L -ый элемент которого это матрица $2^{M-L}\times 2^{M-L}$ пикселей, где $L\in[0,M]$, $M=\log_2 N$. Последующий элемент этой пирамиды формируется фильтрацией и прореживанием предыдущего в два раза по строке и столбцу.

На рисунке 11 показана схема пирамидального алгоритма Лукаса — Канабе.

Рисунок 11 — Схема пирамидального алгоритма Лукаса — Канабе

2.1.3 Построение схемы

Построение схемы является многоэтапным процессом.

Зная координаты перемещения точек интереса от кадра к кадру входной последовательности изображений можно соединить их определённым образом и так построить контуры объектов на фотографиях. Эти контуры и будут в дальнейшем считаться картой-схемой.

Однако отсутствие должной стабилизации камеры и тряска значительно осложняют эту задачу — слишком близко расположенные углы образуют «кучи» точек, через которые невозможно провести предполагаемый контур предмета. Для устранения эффекта тряски было решено алгоритмы кластеризации и определения так называемых центроидов.

Кластер — это подмножество объектов в выборке, которые похожи друг на друга в пространстве. Метрику сходства необходимо выбирать под конкретную задачу, но чаще всего используется евклидово расстояние.

Для этапа «усреднения» значений точек интереса будет применён алгоритм кластеризации k-средних (k-means). Этот алгоритм, относящийся к числу плоских алгоритмов кластеризации, строит заданное число кластеров, расположенных как можно дальше друг от друга. Одни и те же объекты можно по-разному разбить на кластеры, это зависит от начального положения центроидов. Чаще всего выбирается k случайных объектов из выборки, которые становятся центроидами. Целью алгоритма является минимизация дисперсии внутри кластера. Алгоритм k-средних:

1. Берутся случайные точки в пространстве, которые принимаются за центры кластеров (центроиды c_i).
2. Для каждого объекта в выборке находится ближайший к нему центроид.
3. Каждому центроиду соответствует множество ближайших объектов. Формула для его нахождения:

$$c_i = \frac{\sum_{j=1}^{N_i} x_j}{N_i}, \tag{24}$$

где N_i — количество объектов в кластере.

4. Центроиды переходят в найденный центр кластера. Далее алгоритм повторяется, пока центроиды не перестанут менять положение.

Функция стоимости алгоритма L выглядит следующим образом:

$$\mathrm{L}(c_1, c_2, \ldots, c_k) = \sum_{i=1}^{k} \sum_{j=1}^{N_i} \|x_j - c_i\| \to \min, \tag{25}$$

где c_i — центроиды кластеров; k — количество кластеров; N_i — количество объектов в кластере; x_j — объекты в кластере.

Схема работы алгоритма k-средних изображена на рисунке 12.

Рисунок 12 — Схема работы алгоритма k-средних.

Иерархическая агломеративная кластеризация будет использована для того, чтобы объединить центроиды, найденные на предыдущем этапе, в новые кластеры. «Агломеративная» (то есть, «восходящая») означает, что алгоритм начнёт работу с одного объекта и постепенно объединит их в кластеры большего размера. Иерархия контуров предметов на видео означает иерархию соответствующих им кластеров, ведь вложенность очертаний предметов в друг друга при взгляде сверху очевидна.

Выпуклую оболочку получившихся кластеров будем считать контуром соответствующего объекта на видео, а совокупность таких контуров — картой-схемой. Вообще, выпуклой оболочкой некоторого множества точек является такое выпуклое множество (в нём все точки отрезка между двумя точками множества также должны принадлежать этому множеству), которое содержит заданную фигуру и которое само содержится в любом другом выпуклом множестве [21]. Построение выпуклой оболочки будет осуществляться алгоритмом быстрой оболочки, в основе которого лежит быстрая сортировка Хоара [22]. Метод быстрой оболочки предполагает следующие геометрические построения:

1. Пусть дано некоторое множество точек. Выберем такие две точки этого множества — A и B, которые будут крайней левой и крайней правой соответственно. Проведём прямую через эти точки и обозначим подмножество выше или на прямой как M_1, а ниже или на ней — M_2.
2. Теперь рассмотрим M_1. Выберем точку P такую, что расстояние между ней и прямой AB будет максимальным. Эта точка является вершиной выпуклой оболочки множества. Проведём отрезки AP и BP. Точки внутри образовавшегося треугольника APB не будут рассматриваться в дальнейшем, так как находятся внутри границы оболочки.
3. Аналогично для множеств точек слева от AP и справа от BP находим самую удалённую точку от этих прямых и строим выпуклую оболочку.
4. Аналогично для M_2.

2.2 Требования к программному обеспечению

Разработанное программное обеспечение должно предоставлять следующие возможности:

- загрузку видеозаписи;
- просмотр промежуточных результатов (распределения точек по кластерам) в виде изображений;
- сохранение промежуточных результатов (распределения точек по кластерам) в виде изображений;
- просмотр получившейся карты-схемы;
- сохранение получившейся карты-схемы.

На рисунке 13 представлена диаграмма вариантов использования.

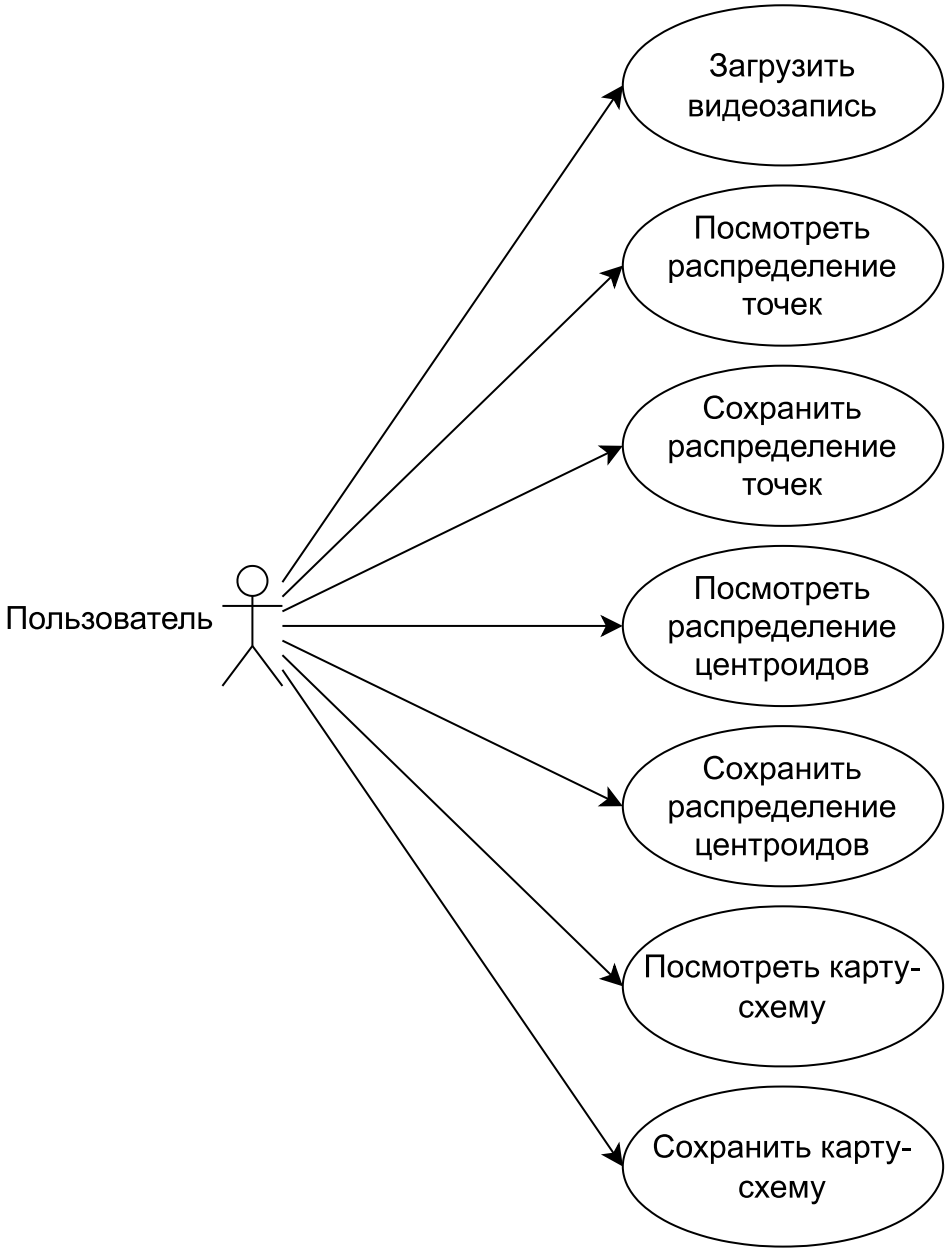

Рисунок 13 — Диаграмма вариантов использования

2.3 Входные и выходные данные

На вход программе подаётся видеозапись, формат которой может быть `AVI` или `MP4`. Ниже перечислены рекомендации для того, чтобы получившаяся карта-схема была как можно более точной:

— камера должна быть направлена вниз и охватывать как можно большее пространство картируемого помещения;

— быть неподвижной;

— длительность съёмки — 5 секунд.

Программные проверки, например, на предмет и длительность съёмки не планируются.

На выходе будет формироваться изображение, представляющее собой карту-схему сцены на входной видеозаписи. Оно имеет такие же размеры как и исходное видео, в нём на чёрном фоне отмечены контуры объектов (выпуклые оболочки кластеров), чей цвет соответствует цвету своих кластеров.

2.4 Выводы

В данном разделе был разработан и описан метод картографии помещений на основе визуальной одометрии. Также определены основные компоненты разрабатываемого программного обеспечения и их взаимосвязи, декомпозиция этих компонентов, построение структурных иерархий и проектирование компонентов. Приведены схемы алгоритмов и ограничения входных данных для решения поставленной задачи.

3 Технологический раздел

В данном разделе обоснован выбор языка программирования и среды разработки. Также описана реализация разработанного метода, инструкции для запуска, работа с графическим интерфейсом.

3.1 Выбор языка программирования и среды разработки

Для реализации разработанного метода был выбран язык Python [23] по следующим причинам:

— будучи языком с динамической типизацией, позволяет лаконично описывать различные математические преобразования, освобождая программиста от действий, не связанных с стоящей перед ним задачей;
— не являясь чисто объектно-ориентированный языком программирования, он позволяет описывать поведение проектируемой системы без лишних абстракций;
— наличие библиотеки OpenCV для работы с изображениями [24];
— наличие математических библиотек для инженерных расчётов Scipy [25], Sklearn [26], Numpy [27], необходимых с большими объёмами данных.

В качестве среды разработки была выбрана Visual Studio Code [28] по следующим соображениям:

– полностью бесплатна;
– имеет огромное количество расширений, в том числе и для работы с Python [29];
– данная среда доступна на большинстве операционных систем, а значит обладает кроссплатформенностью и легко переносима.

3.2 Реализации алгоритмов

Поскольку язык Python не навязывает определённую структуру разрабатываемого на нём ПО, то в состав разработанной программы вошли как классы, так и отдельные функции.

Поиск точек интереса в кадрах и вычисление оптического потока выделены в класс `App`. Он имеет несколько полей:

— `track_len`. Представляет собой целое число, равное длине списка точек для вычисления оптического потока. При инициализации экземпляра класса равно 5;
— `detect_interval`. Представляет собой целое число, является шагом для нахождения точек интереса в кадре. Таким образом, для каждого n-ого кадра будут найдены точки интереса. При инициализации экземпляра класса равно 5;
— `tracks`. Представляет собой список координат точек интереса, чьё местоположение меняется от кадра к кадру. Каждый элемент такого списка — это кортеж из абсциссы и ординаты точки интереса в пространстве кадра. При инициализации экземпляра класса список создаётся пустым;
— `cam`. Это поле хранит дескриптор входного файла с видео формата, который указан в пункте 2.3;
— `frame_idx`. Представляет собой целое число, равное номеру текущего кадра. Поскольку поиск точек интереса и вычисление оптического потока происходит в цикле по всем кадрам видеозаписи, целесообразно хранить индекс кадра. При инициализации экземпляра класса равно 0.

Единственным методом класса `App` (помимо конструктора `__init__`, который единственным явным аргументом принимает исходный видеофайл) является метод `run`, реализующий функции поиска необходимых точек интереса. Этот метод возвращаемого значения не имеет, как и явных аргументов.

3.2.1 Выявление точек интереса

Получение последовательных изображений осуществляется посредством класса `VideoCapture` библиотеки OpenCV [30], затем цветовое пространство каждого кадра с помощью функции `cvtColor` [31] из традиционного (красный, зелёный, синий) в режим оттенков серого. За непосредственно определение интересующих точек отвечает функция

`goodFeaturesToTrack`, в которую, помимо прочих, передаются такие параметры, как: максимальное количество обнаружаемых точек интереса (или, углов), характеристика минимального показателя качества обнаруженных углов, размер окрестности пикселя, в которой ведётся поиск угла и т. д. [32]. В последствие, точки интереса будут обнаруживаться через наперёд заданное количество кадров, так как функция не проверяет, насколько корректны следующие ключевые точки. Поэтому даже если какая-либо точка исчезает на изображении, есть вероятность, что оптический поток найдёт следующую точку, которая может выглядеть близко к ней.

3.2.2 Вычисление оптического потока

Имея два последовательных кадра видеозаписи и координаты точек интереса на них, можно, сравнив их, вычислить оптический поток. Достигается это засчёт использования функции `calcOpticalFlowPyrLK` [33], вычисляющей оптический поток по пирамидальному алгоритму Лукаса — Канаде (его принцип действия был разобран в пункте 2.1.2), причём двоекратного. Вычисляется оптический поток точек интереса как от предыдущего кадра к последующему, так и в обратном направлении. Такие меры нужны, чтобы минимизировать возможную ошибку, когда новое положение точки интереса вычислить невозможно.

Координаты найденных таким образом точек заносятся в CSV-файл [34] (comma-separated values, с англ. «значения, разделённые запятыми»), а по самим точкам отрисовывается их траектория в видео. Отрисовка различных линий, окружностей реализована средствами всё той же библиотеки OpenCV [35].

3.2.3 Построение схемы

Помимо вышеобозначенного класса `App`, в программе имеется функция `draw_map`. Её аргументами являются переменные `file` и `example_frame`, а возвращаемым значением — `None`. Первый аргумент `file` представляет собой CSV-файл, хранящий координаты точек интереса в каждом обработанном кадре. Второй аргумент — первый кадр из видео, на размерах и цветовом профиле которого будут все промежуточные изображения и конечная

карта-схема. Назначение данной функции — отрисовка карты-схемы объектов в видео.

Стоит отметить, что библиотеки Scipy и Scikit основаны на Numpy, поэтому типы выходных данных их методов и функций представляют типы данных из Numpy, например: тип данных `ndarray` (от англ. *n*-dimensional array — *n*-мерный массив). [36]

Первым этапом из файла выгружаются координаты точек в двумерный список. Таким образом данные становятся более удобными для их последующей обработки.

Вторым этапом осуществляется сама кластеризация точек. За кластеризацию отвечает класс `KMeans` из библиотеки `scikit-learn`. [37] Явные параметры, использовавшиеся при его инициализации: `n_clusters` равен числу, введённому пользователем, `n_init='auto'`, `max_iter=300`. Число `n_init` означает количество итераций алгоритма *k*-средних для разных центроидов. В данном случае оно равно 1, так как используется не классический вариант алгоритма, а «жадный». В «жадном» варианте *k*-средних несколько раз выбирается «лучший» центроид в терминах вероятностного распределения точек. Параметр `max_iter` отвечает за максимальное количество итераций за один проход алгоритма.

Так как пользователю дана возможность просматривать изображения, формирующегося на каждом этапе, то — для наглядности — каждый кластер окрашен в разный цвет. Этого удалось достичь засчёт ассоциативного массива (он же «словарь»), в котором сопоставлены номера кластеров и их цвет в модели КЗС. Результирующее изображение составляется посредством функции `circle` из библиотеки OpenCV. Здесь следует добавить, что для того, чтобы хранить информацию о том, какая точка относится к какому кластеру, у экземпляра класса `KMeans` есть атрибут `labels_`, чей тип данных — `ndarray` и который хранит номера кластеров в порядке соответствующих им точек в исходном наборе данных.

Следующим этапом построения карты-схемы является кластеризация центроидов имеющихся точек. В библиотеке Scikit тоже есть подходящий класс для иерархической восходящей кластеризации — `AgglomerativeClustering`. [38] В разработанной программе он явно инициализируется следующими параметрами: `n_clusters` и `linkage`. Если первый отвечает за количество обнаружаемых кластеров и задаётся пользователем, то второй указывает, какой тип связи, т. е. какое расстояние между наборами данных будет использоваться при построении матрицы сходства. Наиболее подходящим значением будет `'average'` — среднее расстояние между элементами этих кластеров. Набором же данных для текущего этапа будет набор центроидов, получаемых из атрибута `cluster_centers_` объекта `KMeans`. Визуализация строится аналогично визуализации на предыдущем этапе, однако перед передачей координат точек функции circle, нужно преобразовать их из типа `float` (число с плавающей запятой) в тип `int` (целое число).

Для нахождения выпуклых оболочек кластеров были использованы два ассоциативных массива. Первый хранит пары «кластер – набор входящих в него точек», а второй «кластер – его выпуклая оболочка». За построения последней отвечал объект `ConvexHull` библиотеки Scipy [39]. Для его инициализации нужен список типа `ndarray` из точек, для которых и строится выпуклая оболочка. Используя возможности OpenCV, а именно функцию `line`, можно построить контуры объектов, снова преобразовав рациональные координаты в целочисленные.

3.3 Тестирование отрисовки карты-схемы

Отрисовка карты-схемы по вычисленным координатам является такой же важной частью программы, как и само их вычисление. При виде непонятных линий, не соответствующих чертам объектов в видео, кажется, что возможная ошибка может где угодно, но только не в тривиальной операции прорисовки

граней выпуклой оболочки. Именно поэтому важно устранить любое неожиданное поведение функции отрисовки карты-схемы.

Можно выделить следующие классы эквивалентности:

— множества, содержащие меньше, чем 3 точки. В этом случае оболочки будут вырожденными. В случае одной точки — собственно, в точку, двух точек — в отрезок. Более того, построение таких «оболочек» не допускается объектом `ConvexHull`.

— множества, содержащие больше, чем 2 точки и не имеющие внутренних. Здесь оболочка не будет являться вырожденной построить корректно.

— множества, содержащие больше, чем 2 точки и имеющие внутренние точки, не попадающие на границу множества.

Была написана отдельная программа, рисующая выпуклую оболочку заданного множества точек. Её листинг представлен ниже (листинг 1). Рисунок 14 представляет собой получившееся изображение. Проверка количества точек множества предотвращает ошибку программы во время построения оболочки, а взятие координат вершин оболочки через обращение к объекту позволяет в случае задания двух разных контуров в одном наборе данных корректно её изобразить.

Листинг 1 — Программа для тестирования

```
import cv2 as cv
import numpy as np
from scipy.spatial import ConvexHull
def draw_str(dst, target, s) → None:
    x, y = target
      cv.putText(dst, s, (x+1, y+1), cv.FONT_HERSHEY_PLAIN, 1.0, (0, 0, 0),
thickness=2, lineType=cv.LINE_AA)
     cv.putText(dst, s, (x, y), cv.FONT_HERSHEY_PLAIN, 1.0, (255, 255, 255),
lineType=cv.LINE_AA)
zeros_like = cv.imread('1.png')
colored_features_img = np.zeros_like(zeros_like)
points1 = np.array([[10, 10]])
points2 = np.array([[10, 10], [500, 10]])
points3 = np.array([[15, 15], [505, 15], [505, 505]])
points4 = np.array([[600, 600], [910, 600], [910, 910], [600, 910]])
points5 = np.array([[25, 600], [515, 600], [515, 900], [25, 900], [245, 745]])
points = [points1, points2, points3, points4, points5]
for array in points:
    for point in array:
          cv.circle(colored_features_img, (point[0], point[1]), 0, (255, 255,
255), 10)
        draw_str(colored_features_img, (point[0]+5, point[1]+5), f'{point[0]};
{point[1]}')
    if array.shape[0] > 2:
        hull = ConvexHull(array)
        for simplex in hull.simplices:
            x1 = int(hull.points[simplex[0]][0])
            y1 = int(hull.points[simplex[0]][1])
            x2 = int(hull.points[simplex[1]][0])
            y2 = int(hull.points[simplex[1]][1])
            cv.line(colored_features_img, (x1, y1), (x2, y2), (0, 255, 0))
cv.imwrite('1.png', colored_features_img)
```

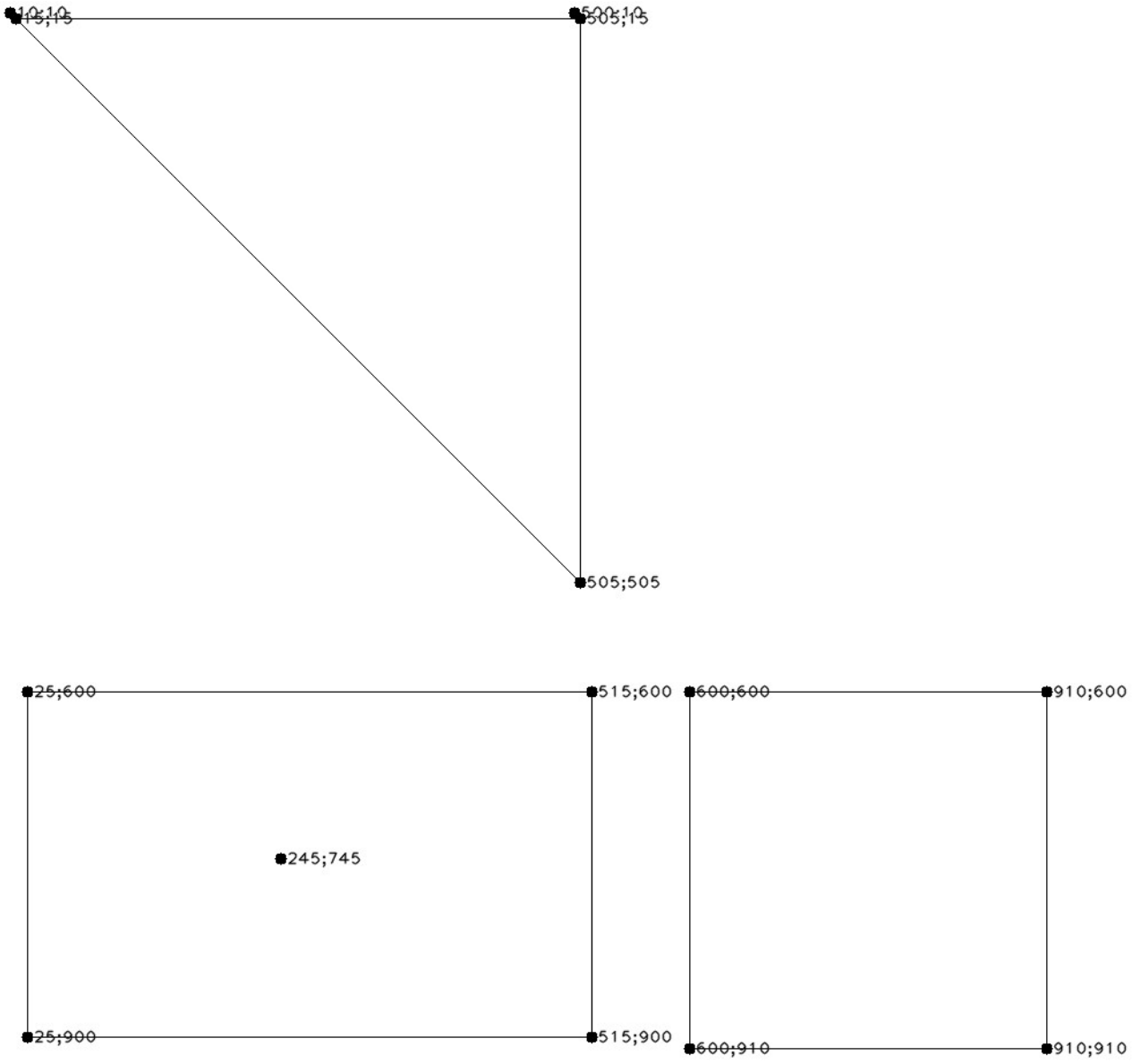

Рисунок 14 — Результаты тестирования

3.4 Взаимодействие с программой

Взаимодействие с программой осуществляется через графический пользовательский интерфейс, созданный с помощью библиотеки Tkinter [40]. Единственное окно представлено на рисунке 15.

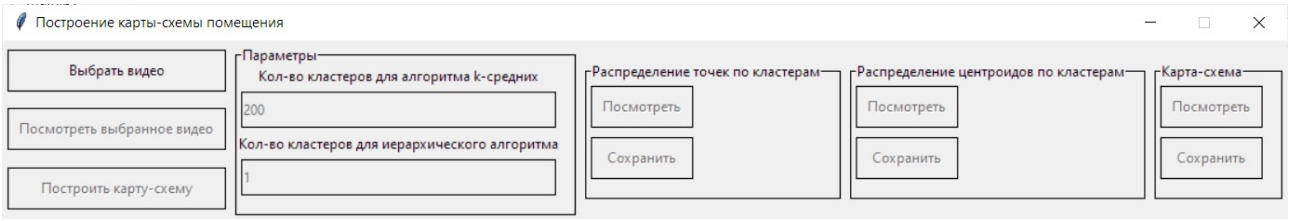

Рисунок 15 — Пользовательский интерфейс

Окно можно поделить на 5 частей. В самой левой находятся кнопки для загрузки видео в программу, его просмотра и, собственно, построение по видео карты-схемы. Справа находится раздел с двумя параметрами, которые влияют на степень правдоподобности карты. Первый параметр задаёт количество кластеров для алгоритма *k*-средних и должен быть больше 2. Второй параметр задаёт количество кластеров для иерархической агломеративной кластеризации и должен быть равен хотя бы 1. Далее располагаются кнопки для просмотра и сохранения изображений, полученных на последующих этапах анализа точек интереса.

3.5 Инструкции для запуска

Для запуска разработанной программы необходимо установить интерпретатор языка Python не ниже версии 3.8. Также необходимо установить через пакетный менеджер `pip` следующие пакеты: `opencv-python`, `numpy`, `scipy`, `scikit-learn`.

Для непосредственно запуска разработанной программы необходимо запустить интерпретатор языка Python, передав ему в качестве аргумента путь к единственному файлу `main.py`: `py main.py`.

3.6 Пример работы

Пусть пользователь после запуска программы выбрал некоторую видеозапись с учётом требований пункта 2.3 и значения параметров: кол-во кластеров для алгоритма *k*-средних — 100, кол-во кластеров для иерархического алгоритма — 2. После нажатия на кнопку «Построить карту-схему» появляется окно, в котором воспроизводится исходное видео с отмеченными на нём зелёным цветом точками интереса и которое закрывается

автоматически после окончания видео. Пример этого окна изображён на рисунке 16.

Рисунок 16 — Вычисленный оптический поток точек интереса (отмечены зелёным цветом)

Кадр из входного видео представлен на рисунке 17 (а). На рисунке 17 (б) изображено распределение точек интереса из всех кадров входного видео по кластерам (алгоритм k-средних). На рисунке 17 (в) изображено распределение найденных центроидов в соответствии иерархического аггломеративного алгоритма. Рисунок 17 (г) — это получившаяся карта-схема объектов из видеозаписи. Как уже было сказано, программа не проверяет наличие на видео именно помещения, поэтому фактически в видео может оказаться всё, что угодно.

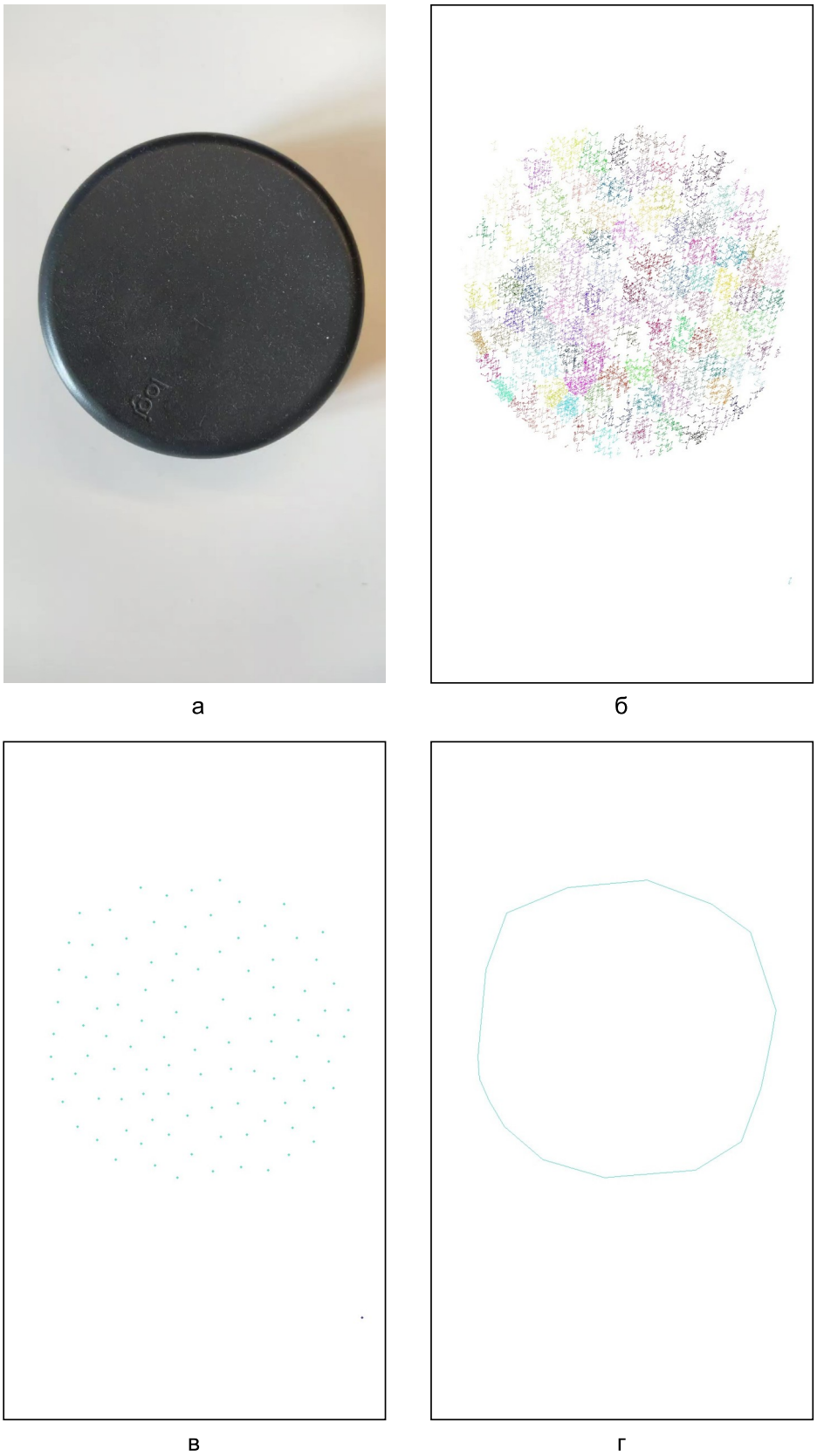

Рисунок 17 — Входное (а) и выходные (б, в, г) изображения

3.7 Выводы

В данном разделе был выбран и обоснован выбор языка программирования и среды разработки. Также была описана реализация разработанного метода. Были описаны инструкции для запуска, работа с графическим интерфейсом.

4 Исследовательский раздел

В данном разделе проводится исследование эффективности разработанного метода картографирования на основе визуальной одометрии, также изучено влияние конфигурации, освещённости сцены в видео на строящуюся карту.

Значение результатов этих исследований сложно недооценивать: освещённость сцены и количество объектов на ней имеют большое влияние на её качество (или, правдоподобность). В свою очередь правдоподобность карты-схемы критична в самых разных условиях эксплуатации разработанного метода. К ним относятся, например, оценка качества уборки роботом-пылесосом. Поскольку устройство подобных роботов подходит только для сбора пыли, то, проезжая, по жидкости или веществу вязкой консистенции на полу остаётся след, а нижняя часть робота, включая ходовую, повреждается. Предложенный метод картирования пола, где будет находится робот, позволит своевременно предупредить опасную ситуацию для механизмов автономного аппарата. Также карта может быть полезна в аддитивных технологиях и трёхмерной печати.

4.1 Характеристики компьютера

Характеристики компьютера, на котором будет проводиться исследование:

— операционная система: Windows 10 Домашняя для одного языка 22H2, 64-разрядная;

— центральный процессор: Intel® Core™ i7-8550U 1.80ГГц 1.99 ГГц;

— количество ядер: 4;

— количество потоков: 8;

— объём оперативного запоминающего устройства: 8 ГБ.

4.2 Характеристики камеры

Характеристики камеры, на которую велась видеосъёмка:

— сенсор: 48 мегапикселей (Sony IMX 586);

— оптическая стабилизация;

— диафрагма: $f/1.7$.

4.3 Исследование зависимости количества угловых точек от контрастности сцены

В этом исследовании контрастность сцены во входном видео разделена экспертным методом на три уровня: низкая, средняя и высокая. На сцене низкой контрастности объекты на переднем плане мало или совсем неотличимы от заднего плана. При средней контрастности объекты на переднем плане возможно различить, однако у них длинная нечёткая тень, накладывающаяся на другие объекты сцены. На сцене же высокой контрастности теней от объектов почти нет, либо они короткие и чёткие.

Результаты опроса экспертов в количестве 10 человек представлены на рисунке 18 (рисунки 18 (а), 18 (е) и 18 (ж) инвертированы).

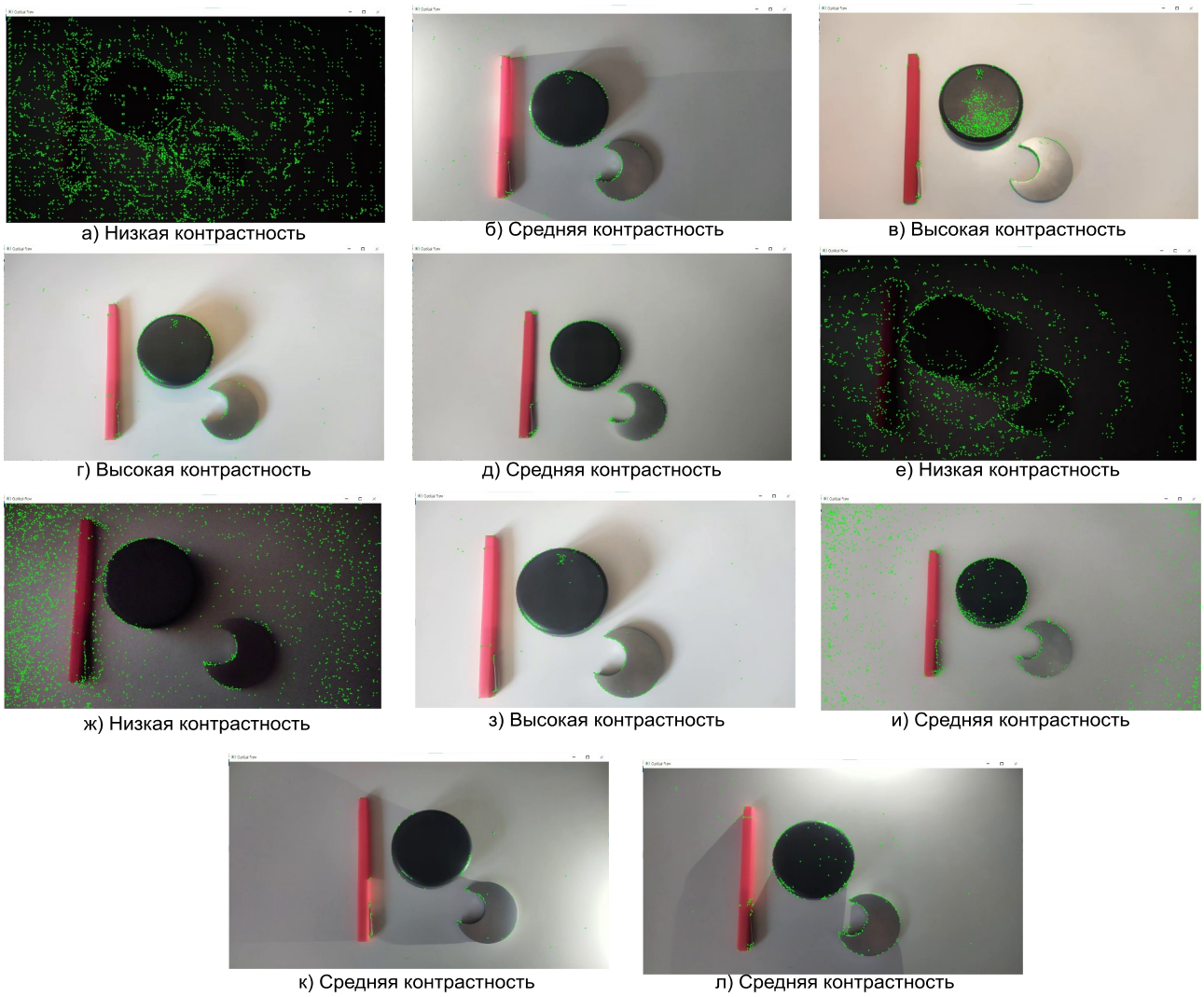

Рисунок 18 — Кадры высокой, средней и низкой контрастности

Рисунок 19 демонстрирует результаты исследования. На видео с высокой контрастностью ожидаемо было найдено больше точек, чем на видео с низкой: более отчётливые контуры объектов и различные особенности их поверхности означают более низкую интенсивность пикселей объектов переднего плана и более высокую интенсивность заднего белого фона. Однако же видео с низкой контрастностью сцены неожиданно показало самое большое количество точек интереса. Это связано не с недостатками алгоритма вычисления оптического потока, а с многочисленными артефактами съёмки в тёмном помещении: зернистость, экспозиция и т. д. Для проверки этого предположения было вручную создано видео с абсолютно чёрным фоном и абсолютно белым фоном и проанализированы на предмет точек интереса,. Обработка обоих видео со сплошным фоном (чёрным или белым) дала результат в 0 обнаруженных точек

— на кадрах попросту нет пикселей, чья интенсивность не равнялась бы 0 (чёрный цвет) или 255 (белый цвет), а значит градиент функции интенсивности всюду равен 0.

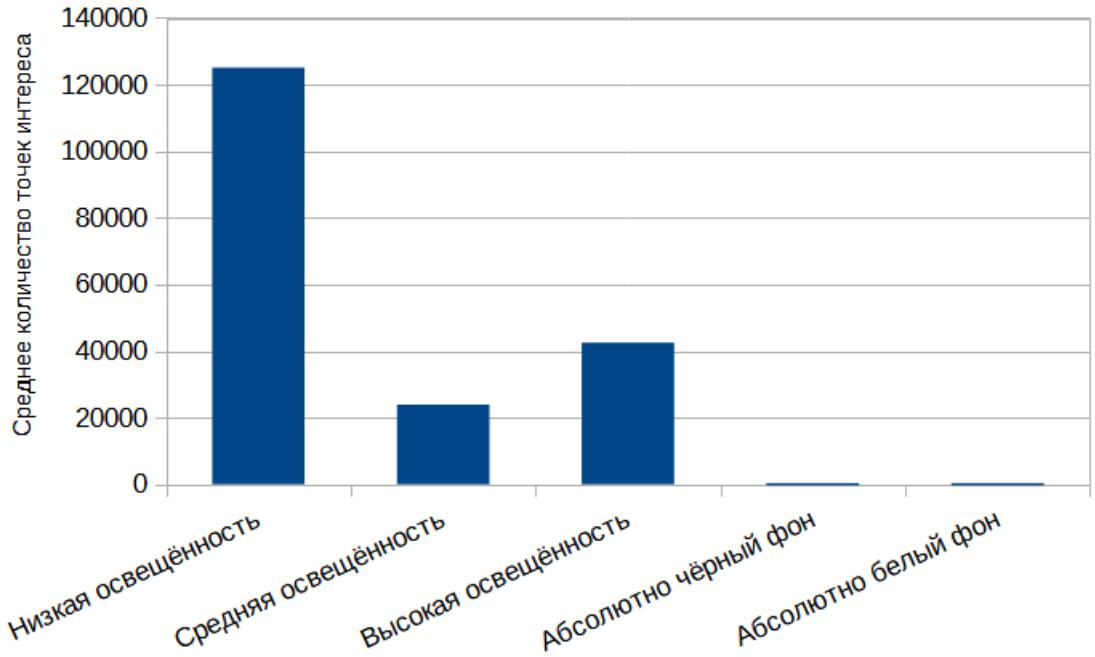

Рисунок 19 — Зависимость среднего количества точек интереса от контрастности сцены из видео

4.4 Исследование влияния на правдоподобность карты-схемы задаваемого количества кластеров

Поскольку назначение разработанного программного обеспечения — апробация метода построения карты-схемы помещений с помощью визуальной одометрии (т. е. по видеозаписи) и поскольку *правдоподобность* зависит от двух выявленных в ходе разработки метода параметров, то было бы целесообразно узнать, насколько значения этих параметров влияют на карту-схему. Правдоподобностью карты-схемы помещения будем считать экспертную оценку факта соответствия контуров объектов непосредственно в помещении и контуров на карте-схеме.

Предварительно были выделены несколько классов сцен входного видео:

— однородная сцена. Сцена представляет собой сплошной фон без выделяющихся объектов. Допустимы различные тени или блики света.

— сцена низкой сложности. На сцене расположены от одного до пяти крупных, хорошо выделяющихся объектов несложной формы (круг, прямоугольник и т. д.).
— сцена средней сложности. В этом случае на сцене расположены один или несколько самопересекающихся контуров объектов, например, скрученные провода.
— сцена высокой сложности. На такой сцене могут быть расположены в большом количестве мелкие объекты как и примитивной, так и неправильной формы: клавиатура, бумага с текстом, цветная узорчатая ткань. Допустимы неконтрастные передние и задние планы изображения.

Факт правдоподобности получившейся карты-схемы оценивали 10 экспертов. Каждому были предоставлены 12 изображений высокой контрастности по три на каждый из 4 классов сцены (однородная, низкой, средней или высокой сложности): кадр из входного видео, карта-схема, построенная с параметрами, значения которых основаны на предположении, что параметр «количество кластеров для алгоритма k-средних» соответствует в каком-то смысле степени стабилизирования видео, а «количество кластеров для иерархического алгоритма» — количеству объектов на сцене, третьей будет карта-схема, построенная со значениям, опровергающими выдвинутое предположение. Карты-схемы из опроса представлена на рисунке 20.

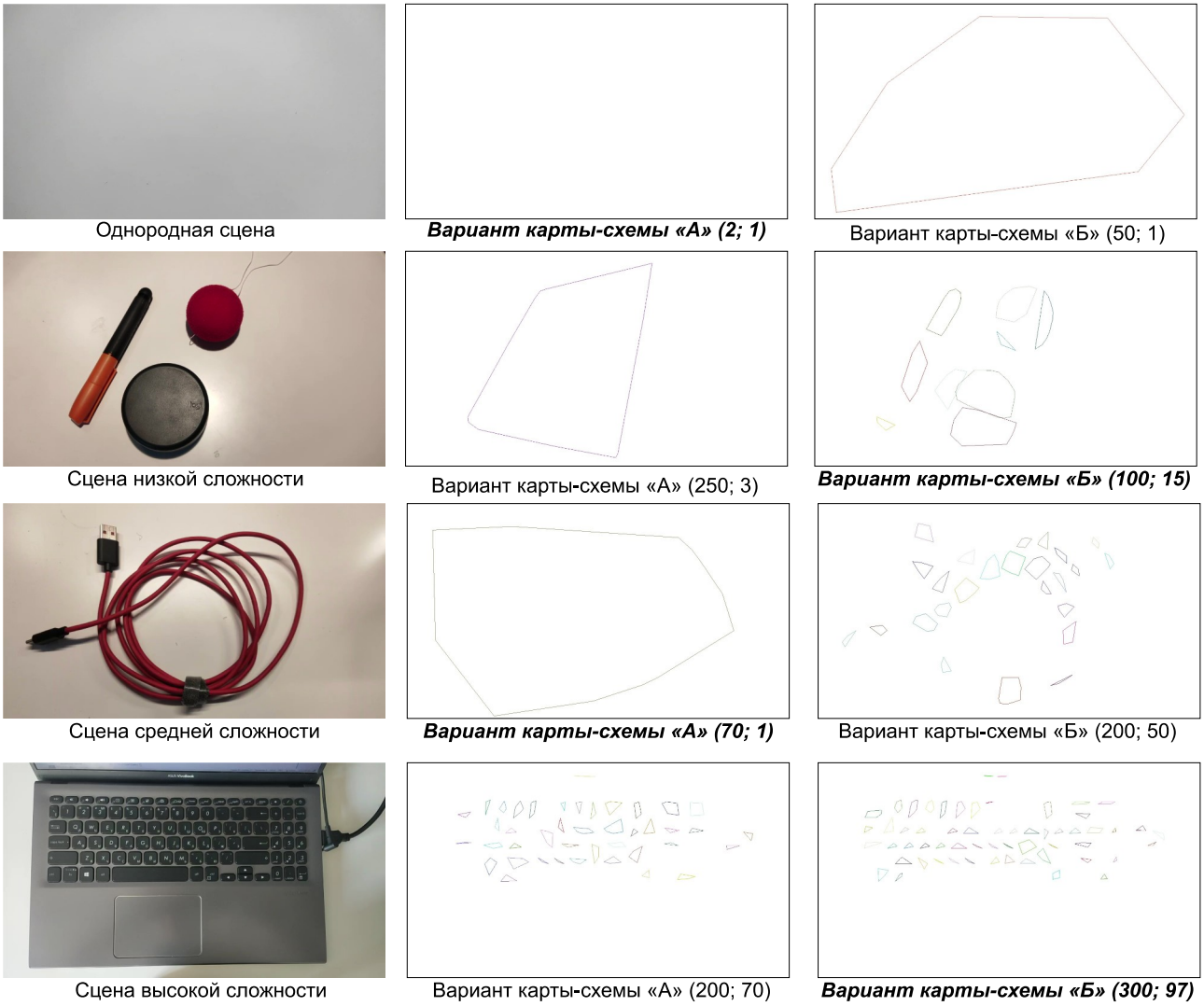

Рисунок 20 — Материал для экспертного опроса

Опрос показал, что эксперты назвали правдоподобными следующие варианты:

— для однородной карты вариант «А» (количество кластеров для алгоритма k-средних: 2; количество кластеров для иерархического алгоритма: 1);

— для сцены низкой сложности конфигурации: вариант карты «Б» (количество кластеров для алгоритма k-средних: 100; количество кластеров для иерархического алгоритма: 15);

— для сцены средней сложности конфигурации: вариант карты «А» (количество кластеров для алгоритма k-средних: 70; количество кластеров для иерархического алгоритма: 1);

— для сцены высокой сложности конфигурации: вариант карты «Б» (количество кластеров для алгоритма k-средних: 300; количество кластеров для иерархического алгоритма: 97).

В трёх из четырёх случаях карты, построенные с параметрами, значения которых основаны на предположении, что параметр «количество кластеров для алгоритма k-средних» соответствует в каком-то смысле степени стабилизирования видео, а «количество кластеров для иерархического алгоритма» — количеству объектов на сцене, были оценены как более правдоподобные, чем карты, построенные на основе противоположного суждения, следовательно, выдвинутое предположение подтвердилось.

4.5 Исследование влияния на правдоподобность карты-схемы конфигурации сцены

По каждым результатам исследования приведено 4 изображения. Рисунок «а» демонстрирует кадр входного видео (оно, согласно рекомендациям из пункта 2.3, статично), «б» — выявленные алгоритмом Лукаса — Канаде точки интереса, «в» — обнаруженные алгоритмом «k-средних» центроиды кластеров, «г» — карта-схема, получающаяся методом вычисления выпуклой оболочки кластеризованных точек из изображения «в».

4.5.1 Однородная сцена

Введя минимальные допустимые параметры (количество кластеров для алгоритма k-средних: 2, количество кластеров для иерархического алгоритма: 1), получаем следующие изображения (рисунок 21). Из рис. 21 (а) ясно, что сцена представляла из себя пустую поверхность стола белого цвета с некоторыми неровностями и отблесками света. Как и ожидалось, точек интереса было найдено немного (755), что вместе с заданным количеством кластеров сделало невозможным отрисовку какого-либо контура.

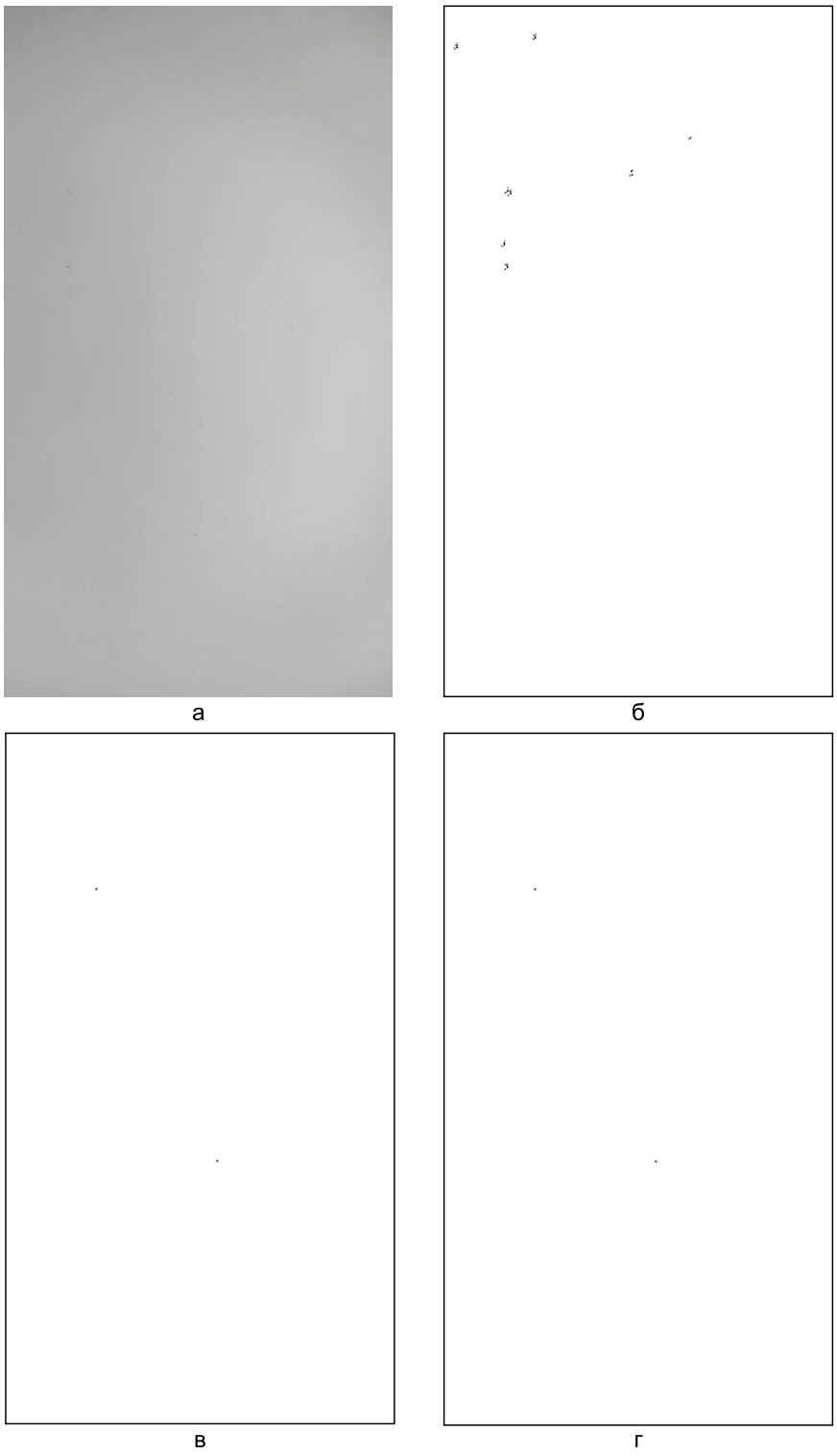

Рисунок 21 — Карта-схема однородной сцены с параметрами 2 и 1

4.5.2 Сцена низкой сложности

Сцена низкой сложности, а также получившиеся изображения представлены далее (рисунок 22). Из рис. 22 (а) видно, что на хорошо освещённой сцене расположены три ясно различимых предмета правильной формы. Окончательные значения параметров 200 и 4 были выбраны методом

подбора, так как с ними карта-схема получается наиболее правдоподобной. Неровность очертаний круга и «поглощение» контура ручки справа можно объяснить отсутствием на их теневых сторонах необходимых точек интереса, ведь в области тени яркость соседних пикселей меняется постепенно, что критично для алгоритма Лукаса — Канаде.

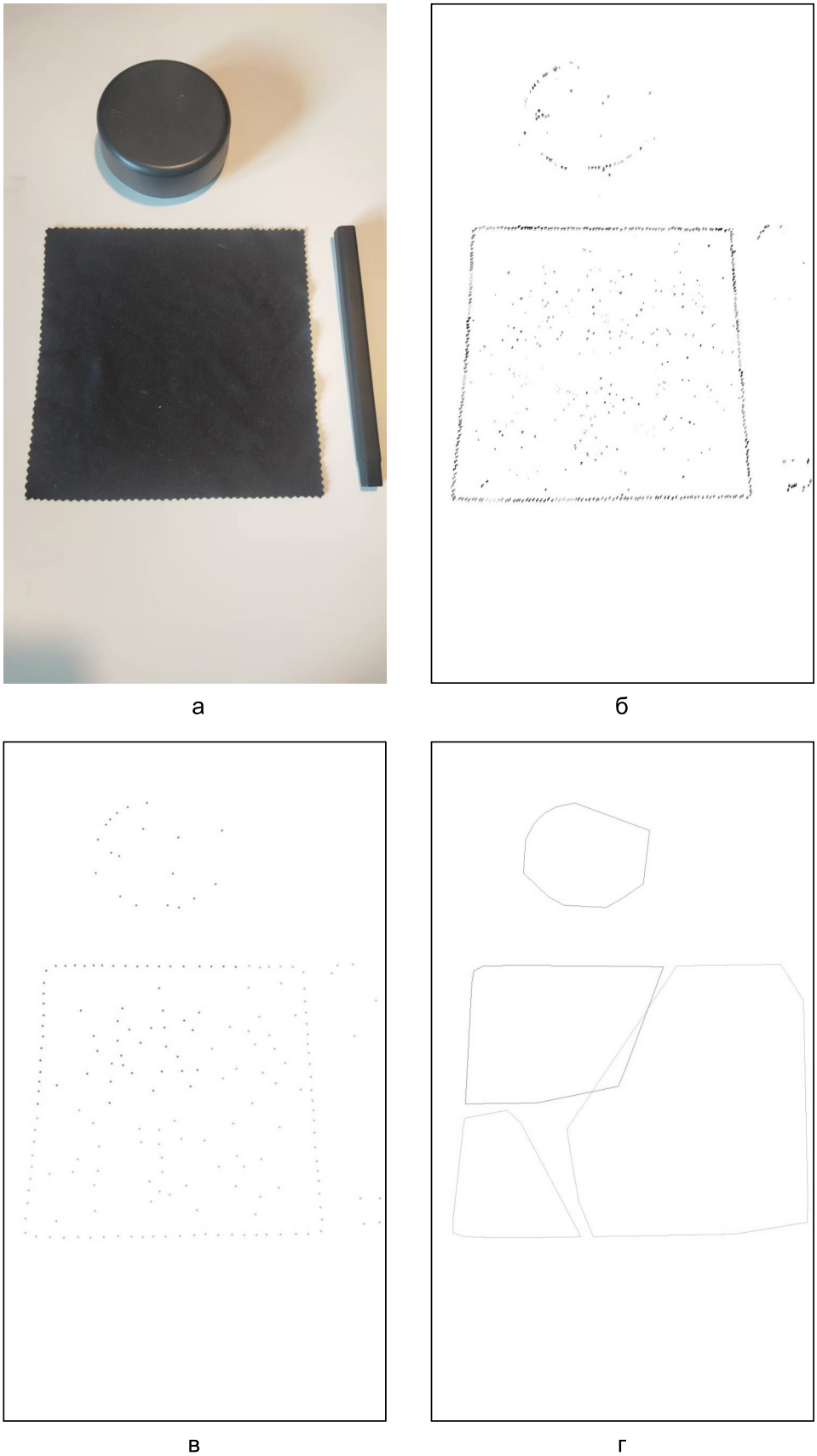

Рисунок 22 — Карта-схема сцены низкой сложности с параметрами 200 и 4

4.5.3 Сцена средней сложности

Сцена средней сложности предполагает наличие пересекающихся контуров предметов, объектов неконтрастных цветов в отличие от предыдущих испытаний, где использованы только чёрные и белые цвета. Результаты можно увидеть на рисунке 23. Значения, использованные при построении карты в данном исследовании (200 и 50), — яркие примеры того, что задаваемое количество объектов в видео может далеко не соответствовать реальным (очевидно, что объект в реальности один, а не 50).

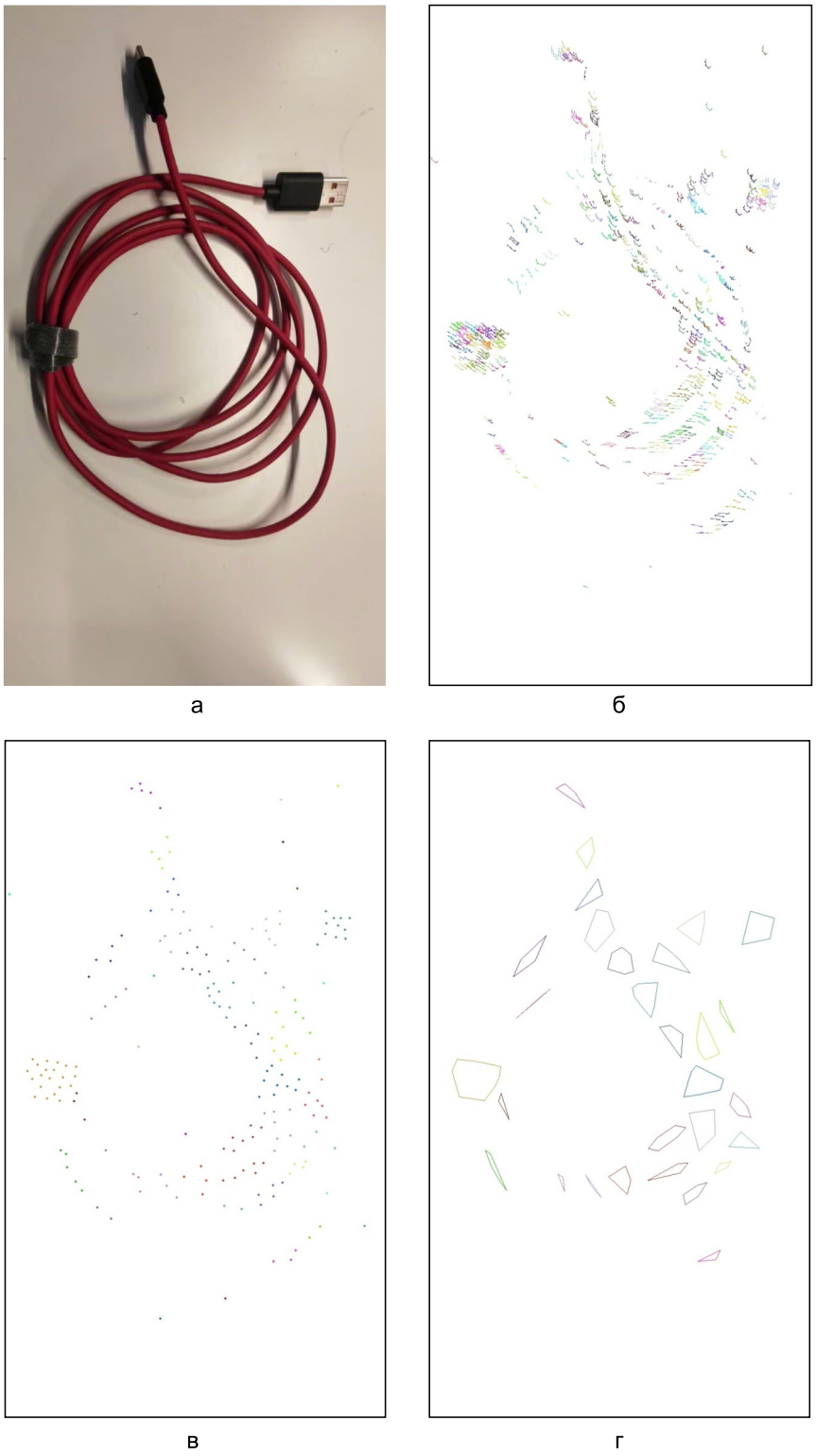

Рисунок 23 — Карта-схема сцены средней сложности с параметрами 200 и 50

4.5.4 Сцена высокой сложности

В этом исследовании на видео будет показана клавиатура — объект сложной формы с вложенными контурами, т. е. контурами клавиш, иногда с подсветкой. Особую сложность представляют надписи на клавишах — обычно

имеющие белый цвет с чёрным фоном. Все эти мелкие детали требуют тщательного подбора двух входных параметров, чьи значения — как было показано в предыдущем исследовании — не обязательно могут совпадать с реальными. Кадр из входного видео и результаты исследования продемонстрированы на рисунке 24.

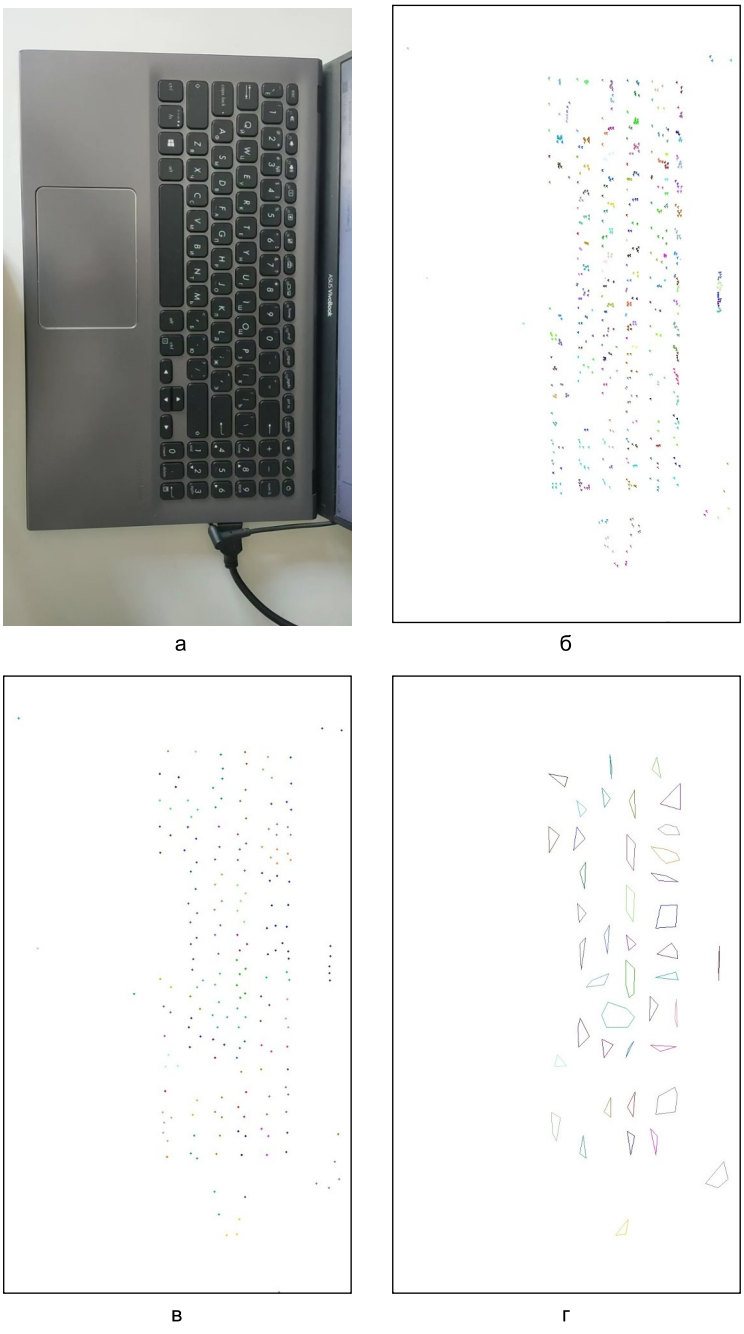

Рисунок 24 — Карта-схема сцены высокой сложности с параметрами 200 и 70

Программа распознала большое количество многоугольных контуров-клавиш, однако общие черты прямоугольного корпуса ноутбука не

угадываются: точки интереса по краям корпуса находятся слишком далеко для объединения в один кластер иерархическим алгоритмом.

4.6 Выводы

В разделе были исследованы важные для разработанного метода зависимости: количества точек интереса от контрастности сцены и точности построения карты-схемы от конфигурации сцены и особенностей предметов на ней. В результате было выявлено, что:

- чем выше контрастность сцены, тем больше будет распознано точек интереса (на 48,65%, т. е. почти на половину), поскольку очертания объектов и их теней чётче и градиенты интенсивности в точках интереса будет иметь больший модуль;
- при отсутствии подходящей настройки камеры для условий съёмки точки интереса будут детектироваться даже в сильно затемнённом или засветлённом помещении, и из-за их большого количества и «хаотического» выявления в результате построенная карта-схема не будет правдоподобна;
- к наиболее правдоподобным контурам на построенных картах-схемах приводят значения для однородных сцен — минимальные (2; 1);
- к наиболее правдоподобным контурам на построенных картах-схемах приводят значения для сцен низкой сложности — k-кластеров: 100±50, количество иерархических кластеров равно количеству предметов на сцене;
- к наиболее правдоподобным контурам на построенных картах-схемах приводят значения для сцен средней сложности, на которых, в частности, расположены скрученные провода либо верёвки, — k-кластеров: 100±50, количество иерархических кластеров равно количеству предметов на сцене;
- к наиболее правдоподобным контурам на построенных картах-схемах приводят значения для сцен высокой сложности — k-кластеров: 300±100,

количество иерархических кластеров равно количеству предметов на сцене;

— чем больше предметов расположено на сцене, чем чаще пересекаются их контуры, тем сложнее добиться правдоподобности карты-схемы: в случае близко расположенных предметов кластеризация их отцентрированных точек интереса невозможна иерархическим алгоритмом, ввиду их близости;

— варьирование двух входных параметров позволяют в некоторой степени устранить несоответствия, отмеченные в предыдущем пункте.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы был разработан метод картографирования помещений на основе визуальной одометрии. В процессе выполнения были выполнены все поставленные задачи:

- проведён обзор предметной области и существующих методов компьютерной картографии помещений;
- разработаны основные положения предполагаемого метода картографирования помещений на основе визуальной одометрии;
- спроектировано и реализовано программное обеспечение для демонстрации разработанного метода;
- проведена апробация метода картографирования помещений на основе визуальной одометрии.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Берлянт А. М. КАРТОГРАФИЯ // Большая российская энциклопедия. Том 13. – Москва – 2009 – с. 244.
2. Aqel M.O.A. et al. Review of visual odometry: types, approaches, challenges, and applications // SpringerPlus – 2016.
3. Jacob Engel, Thomas Schops, Daniel Crempes. LSD-SLAM: Large-Scale Direct Monocular SLAM // European Conference on Computer Vision (ECCV). – 2014.
4. E. Pedrosa, Autonomous Navigation with Simultaneous Localization and Mapping in/outdoor / E. Pedrosa, L. Reis, C. M. D. Silva and H. S. Ferreira. – 2020.
5. Engel J., Koltun V., Cremers D. Direct Sparse Odometry // ArXive-prints. – 2016. – 1607.02565.
6. Gmapping [Электронный ресурс]. Режим доступа: http://wiki.ros.org/gmapping (дата обращения: 21.04.2023).
7. Google Cartographer ROS [Электронный ресурс]. Режим доступа: https://google-cartographer-ros.readthedocs.io/en/latest/# (дата обращения: 21.04.2023).
8. RTAB-Map, Real-Time Appearance-Based Mapping [Электронный ресурс]. Режим доступа: http://introlab.github.io/rtabmap/ (дата обращения: 21.04.2023).
9. Adaptive Monte Carlo localization [Электронный ресурс]. Режим доступа: http://wiki.ros.org/amcl (дата обращения: 25.04.2023).
10. Бай. Система бинокулярного зрения для построения карты помещения и локализация мобильного робота // Политехнический молодежный журнал. – 2018. – № 11(28). – с. 11.
11. Герасюто, С. Л. Построение навигационной карты внутри помещений по величине магнитного поля земли MEMS сенсором мобильного робота /

С. Л. Герасюто, Г. А. Прокопович, В. А. Сычев // Экстремальная робототехника. – 2014. – Т. 1. – № 1. – с. 372–375.

12. Compact Geomagnetic Sensor with Wide Dynamic Range. ZETTLER electronics GmbH [Электронный ресурс]. Режим доступа: http://www.zettlerelectronics.com/pdfs/sonstige/ALPHS_HSCD_E_Flyer_2013.pdf (дата обращения: 08.10.2022).
13. Что такое RFID-метки. Технология радиочастотной идентификации [Электронный ресурс]. Режим доступа: https://www.atol.ru/blog/chto-takoe-rfid-metki-tekhnologiya-radiochastotnoy-identifikatsii/ (дата обращения: 21.11.2022).
14. Козлов, Д. А. Разработка автономной робототехнической системы с использованием SPLAM алгоритма, основанного на применении графов / Д. А. Козлов // Информационные технологии и нанотехнологии (ИТНТ-2021): Сборник трудов по материалам VII Международной конференции и молодежной школы, Самара, 20–24 сентября 2021 года / Под редакцией В.В. Мясникова. – Самара: Самарский национальный исследовательский университет имени академика С.П. Королева – 2021.
15. Карлеварис-Бьянко Н., Юстис Р. М. Общая маргинализация узлов на основе факторов и разбивка ребер для SLAM с графом поз. // Международная конференция IEEE по робототехнике и автоматизации. IEEE – 2013 – с. 5748–5755.
16. Салви Дж., Матабош С., Фофи Д. и др. Обзор последних методов регистрации изображений диапазона с оценкой точности. // Изображение & Компьютерное зрение – 2007 – с. 578–596.
17. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. // ACM Comput. Surv. – 1999 – с. 264–323.
18. Jianbo Shi and Tomasi. Good features to track. // Proceedings of IEEE Conference on Computer Vision and Pattern Recognition – Seattle, WA, USA. – 1994 – с. 593-600.

19. Fleet D., Weiss Y. Optical Flow Estimation. In: Paragios N., Chen Y., Faugeras O. // Handbook of Mathematical Models in Computer Vision. – Springer – Boston, MA – 2006.
20. Смирнов С. А. РАЗРАБОТКА И ОПТИМИЗАЦИЯ АЛГОРИТМА СЛЕЖЕНИЯ ЗА ТРАНСПОРТНЫМ СРЕДСТВОМ В ВИДЕОПОТОКЕ НА ОСНОВЕ ПИРАМИДАЛЬНОГО МЕТОДА ЛУКАСА — КАНАДЕ / Смирнов С. А., Бабаян П. В., Ершов М. Д., Муравьев В. С. // ВК. 2020. № 2 (38). Режим доступа: https://cyberleninka.ru/article/n/razrabotka-i-optimizatsiya-algoritma-slezheniya-za-transportnym-sredstvom-v-videopotoke-na-osnove-piramidalnogo-metoda-lukasa (дата обращения: 02.05.2023).
21. Половинкин Е. С, Балашов М. В. Элементы выпуклого и сильно выпуклого анализа. // М.: Физматлит – 2004.
22. Jonathan Scott. A Proof for a QuickHull Algorithm // Electrical Engineering and Computer Science – Technical Reports. № 65. https://surface.syr.edu/eecs_techreports/65 (дата обращения: 02.05.2023).
23. Welcome to Python.org. [Электронный ресурс]. Режим доступа: https://www.python.org/ (дата обращения: 15.05.2023).
24. Home – OpenCV. [Электронный ресурс]. Режим доступа: https://opencv.org/ (дата обращения: 15.05.2023).
25. SciPy. [Электронный ресурс]. Режим доступа: https://scipy.org/ (дата обращения: 15.05.2023).
26. scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation. [Электронный ресурс]. Режим доступа: https://scikit-learn.org/stable/ (дата обращения: 15.05.2023).
27. NumPy. [Электронный ресурс]. Режим доступа: https://numpy.org/ (дата обращения: 15.05.2023).
28. Visual Studio Code - Code Editing. Redefined. [Электронный ресурс]. Режим доступа: https://code.visualstudio.com/ (дата обращения: 15.05.2023).

29. Python - Visual Studio Marketplace. [Электронный ресурс]. Режим доступа: https://marketplace.visualstudio.com/items?itemName=ms-python.python (дата обращения: 15.05.2023).
30. OpenCV: cv::VideoCapture Class Reference [Электронный ресурс]. Режим доступа: https://docs.opencv.org/4.x/d8/dfe/classcv_1_1VideoCapture.html
31. OpenCV: Color Space Conversions. [Электронный ресурс]. Режим доступа: https://docs.opencv.org/4.x/d8/d01/group__imgproc__color__conversions.html#ga397ae87e1288a81d2363b61574eb8cab (дата обращения: 15.05.2023).
32. OpenCV: Feature Detection [Электронный ресурс]. Режим доступа: https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga1d6bb77486c8f92d79c8793ad995d541 (дата обращения: 15.05.2023).
33. OpenCV: Object Tracking [Электронный ресурс]. Режим доступа: https://docs.opencv.org/4.x/dc/d6b/group__video__track.html#ga473e4b886d0bcc6b65831eb88ed93323 (дата обращения: 15.05.2023).
34. RFC 4180 - Common Format and MIME Type for Comma-Separated Values (CSV) Files [Электронный ресурс]. Режим доступа: https://datatracker.ietf.org/doc/html/rfc4180 (дата обращения: 15.05.2023).
35. OpenCV: Drawing Functions [Электронный ресурс]. Режим доступа: https://docs.opencv.org/4.x/d6/d6e/group__imgproc__draw.html (дата обращения: 15.05.2023).
36. numpy.ndarray — NumPy v1.24 Manual [Электронный ресурс]. Режим доступа: https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html (дата обращения: 15.05.2023).
37. sklearn.cluster.KMeans — scikit-learn 1.2.2 documentation [Электронный ресурс]. Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans (дата обращения: 15.05.2023).

38. sklearn.cluster.AgglomerativeClustering — scikit-learn 1.2.2 documentation [Электронный ресурс]. Режим доступа: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering (дата обращения: 15.05.2023).
39. scipy.spatial.ConvexHull — SciPy v1.10.1 Manual [Электронный ресурс]. Режим доступа: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html (дата обращения: 15.05.2023).
40. tkinter — Python interface to Tcl/Tk — Python 3.11.3 documentation [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/tkinter.html (дата обращения: 15.05.2023).

ПРИЛОЖЕНИЕ А