В разработанном алгоритме используются множество датчиков расстояния. Каждый датчик определяет расстояние на основе различных технологий, таких как ультразвук, инфракрасный лазер, оптическое или радио частотное излучение. Например, датчик определения расстояния VL53L0X работает на основе принципа лидара. Он использует инфракрасный лазер для измерения расстояния до объекта на основе времени, за которое отраженный лазерный сигнал возвращается обратно к датчику.

Когда лазерный сигнал выходит из датчика и попадает на объект, он отражается от него и возвращается обратно к датчику. Датчик затем измеряет время, за которое произошел этот процесс, и использует его для вычисления расстояния до объекта. Метод измерения расстояния довольно точен и позволяет датчику работать на расстоянии до нескольких метров.

Датчик VL53L0X также обладает функцией автоматической компенсации изменений окружающих условий, таких как освещенность и температура, что помогает обеспечить стабильную и точную работу в различных условиях.

Набор сигналов от всех датчиков дает представление о контуре одного "среза" сканируемой области. Считывание нескольких срезов позволяет получить трехмерное представление о контуре сканируемой области. Алгоритм может быть использован для подсчета количества объектов в сканируемой области, а также для вычисления объема объектов. Когда изобретение используется для подсчета объектов на поверхности, каждый объект должен быть окружен областью, которая отражает меньше света, чем сами объекты. Объекты могут касаться друг друга, если область, где они соприкасаются, отражает меньше света, чем остальные объекты.

Данные от всех датчиков объединяются для создания изображения среза сканируемой области. Путем считывания нескольких срезов можно получить трехмерное представление контура сканируемой области. Это устройство может использоваться для подсчета числа объектов в сканируемой области и определения их объема. При подсчете объектов, находящихся на поверхности, каждый объект должен быть окружен областью с более низким уровнем отражения света, чем сами объекты. Объекты могут соприкасаться друг с другом, если площадь контакта отражает меньше света, чем остальные объекты.

В качестве альтернативы датчиков расстояния могут использоваться датчики интенсивности отраженного света, которые измеряют уровень отраженного света от объекта и могут использоваться для определения расстояния до него.

Алгоритм идеально подходит для подсчета и определения размеров объектов, движущихся по движущейся конвейерной ленте, без необходимости размещения объектов в один ряд.

Рисунок N представляет собой вид в вертикальном разрезе платы с датчиками расположенного над лентой для переноса объектов.

Датчики с Na по Nh расположены на одинаковом расстоянии и установлены на плате N1. Плата N1 расположена над лентой N2.

Объекты N3 и N4 перемещаются по ленте N2 мимо датчиков с Na по Nh. Такие объекты не обязательно должны быть расположены в один ряд для того, чтобы измерения могли быть выполнены в соответствии с алгоритмом. Каждый датчик опрашивается и фиксирует показатели перпендикулярно плоскости датчика и в направлении ленты N2.

Каждый цикл опроса всех датчиков приводит к считыванию «среза области» под датчиками в определенное время. По мере перемещения объектов вдоль ленты N2 могут быть получены дополнительные "срезы" показаний. Таким образом, каждый датчик может рассматриваться как представитель координаты X, каждое последовательное считывание может рассматриваться как представитель координаты Y ля конкретной координаты X, а фактический сигнал датчика может рассматриваться как представитель координаты Z сканируемой области.

Рисунок N2 представляет собой график, показывающий показания для одного опроса каждого датчика ленты для переноса объектов, показанной на рисунке N. По оси X представлены датчики с N2a по N2h, а по оси Y – расстояния от датчиков до объектов.

Количество датчиков, выбранных для прохождения траектории для конкретного применения, зависит от нескольких характеристик, таких как ширина траектории, размер объектов, количество объектов, которые могут поместиться на траектории, и желаемое разрешение. Как правило, большее разрешение предпочтительно для вычисления объемов объектов, в то время как меньшее разрешение необходимо только для подсчета объектов.

Расстояние между каждым датчиком также важно, когда изобретение используется для подсчета объектов. Изобретение подсчитывает объект путем обнаружения смежной области. На рисунке N изображен контурный чертеж ленты из

-зависит от шума

-от размера объектов

-от технической спецификации

ОПИСАТЬ РАССТОЯНИЕ РАСПОЛОЖЕНИЯ ДАТЧИКОВ

Рисунок N2 также демонстрирует важность порога вхождения ОПИСАТЬ ВАЖНОСТЬ ПВ

На рисунке N3 представляет собой вид сверху типичного пути, содержащего объекты. На этом рисунке яйца 60, 61 и 62 лежат на ленте 63, которая движется мимо сканера (не показан). Яйца находятся выше (ближе к сканеру) в своих центральных частях, чем в периферийных частях. Яйца генерируют показания, показанные на рисунке N4.

На рисунке N4 представлен репрезентативный массив значений сигналов, указывающих на обнаруженный контур траектории, показанной на рисунке N3. Обнаружение объектов осуществляется путем обнаружения перепадов в отдельных датчиках и группировки датчиков, которые также демонстрируют изменения в показаниях.

Основной принцип работы алгоритма заключается в обнаружении и создании группы вокруг объекта, добавление датчиков в существующую группу, и поиск вершины путем выбора тройки датчиков с наименьшей суммой и определения центра этой тройки. Также алгоритм включает в себя отключение от группы на основе обнаружения другого объекта, корректировку принадлежности датчиков к группам путем проверки экстремумов, и отключение датчиков при прохождении объектов.

Блок-схема алгоритма представлена на рисунке n.

Рисунок n – Блок-схема алгоритма

Изначально все датчики выключены, что означает, что они не могут генерировать счет. Каждый датчик имеет отдельные поля.

* value – текущее значение;
* deltaY – описание;
* center – логическое значение, которое определяет является ли данный датчик центром тройки;
* listhen – логическое значение, которое отображает включен ли датчик;
* isFindTop – логическое выражение, которое определяет найдена ли вершина;
* isEggOne - – логическое выражение, которое определяет был ли у датчиков переход на другой объект;
* iterEggCol – количество показаний данной группы;
* idGroup – отображает номер группы;

Ниже приводится краткое пошаговое описание алгоритма в отношении показаний карты вершин, показанных на рисунке N.

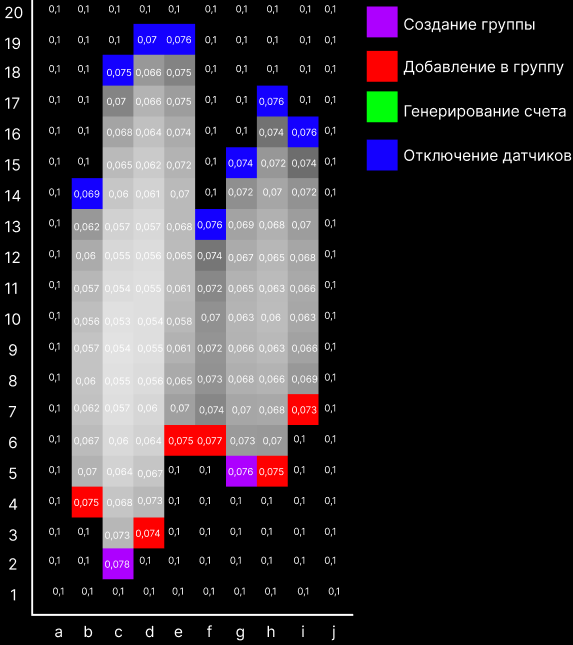


Рисунок N - Показания карты вершин.

Рисунок N представляет собой репрезентативный массив значений сигналов.

ПОШАГОВОЕ ОПИСАНИЕ КАК В ПАТЕНТЕ

Показание 1. Ни одно показание не превысило порог вхождения, все датчики остаются выключенными.

Показание 2. Показание датчика c превышает порог вхождения. Так как по соседству отсутствуют созданные группы, то создается новая группа (группа №1). Центр данной группы устанавливается на датчик c.

Показание 3. Показание датчика d превышает порог вхождения. Так как по соседству существует группа, то датчик d добавляется в данную группу (группа №1). Центр данной группы устанавливается на датчик c.

Показание 4. Показание датчика b превышает порог вхождения. Так как по соседству существует группа, то датчик b добавляется в данную группу (группа №1). Центр данной группы устанавливается на датчик c.

Показание 5. Показание датчика g превышает порог вхождения. Так как по соседству отсутствуют созданные группы, то создается новая группа (группа №2). Показание датчика h превышает порог вхождения. Так как по соседству существует группа, то датчик h добавляется в данную группу (группа №2). Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на на датчик h.

Показание 6. Показание датчика e превышает порог вхождения. Так как по соседству существует группа, то датчик e добавляется в группу №1. Показание датчика f превышает порог вхождения. Так как рядом находятся 2 группы, то датчик присоединятся к той, где показатель соседнего датчика меньше. Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на на датчик h.

Показание 7. Показание датчика i превышает порог вхождения. Так как рядом находится группа №2, то датчик присоединяется к ней. Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на на датчик h.

Показание 8-11. Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на на датчик h.

Показание 12. Значение тройки датчиков с наименьшей суммой группы 2 упало больше, чем на 5%. Это означает, что найдена вершина объекта и генерируется счет. Всем датчикам в этой группе устанавливается флаг, который запрещает генерировать счет. Центр группы 1 устанавливается на датчик с.

Показание 13. Показание датчика f упало ниже порога вхождения. Данный датчик отключается. Центр группы 1 устанавливается на датчик с.

Показание 14. Значение тройки датчиков с наименьшей суммой группы 2 упало больше, чем на 5%. Это означает, что найдена вершина объекта и генерируется счет. Всем датчикам в этой группе устанавливается флаг, который запрещает генерировать счет. Показание датчика b упало ниже порога вхождения. Данный датчик отключается.

Показание 15. Показание датчика g упало ниже порога вхождения. Данный датчик отключается.

Показание 16. Показание датчика i упало ниже порога вхождения. Данный датчик отключается.

Показание 17. Показание датчика h упало ниже порога вхождения. Данный датчик отключается.

Показание 18. Показание датчика c упало ниже порога вхождения. Данный датчик отключается.

Показание 19. Показание датчиков d, e упали ниже порога вхождения. Данные датчики отключаются.

# 1 Анализ состояния проблемы и постановка задачи исследования

Потоковые данные представляют собой непрерывный поток информации, поступающий в систему из внешнего источника в режиме реального времени. Они могут быть в виде различных типов данных, включая аудио и видеосигналы, сенсорные данные, текстовую информацию и т. д.

Значимость адаптивной обработки потоковых данных заключается в том, что она позволяет системам эффективно обрабатывать и анализировать непрерывно поступающую информацию в режиме реального времени. Это особенно важно в ситуациях, когда данные имеют высокую изменчивость и требуют оперативной обработки.

Адаптивная обработка потоковых данных предоставляет возможность системе адаптироваться к изменениям в потоке данных и оптимизировать процессы обработки на основе текущего состояния данных. Это может включать в себя динамическое изменение параметров алгоритмов обработки, выбор наиболее подходящих методов анализа, управление буферизацией данных и другие техники, которые позволяют достичь оптимальной производительности и результатов обработки.

Адаптивная обработка потоковых данных включает в себя несколько этапов:

* предварительная обработка данных данных;
* адаптация данных в зависимости от характеристики потока данных;
* модификация алгоритма детективарония объектов выпуклой формы;

В данной главе производится общее описание задачи и проводится анализ методов решения этой задачи

# 1.1 Общая характеристика проблемы

# 1.1.1 Определение адаптивной обработки потоковых данных

Потоковые данные представляют собой непрерывный поток информации, поступающий в систему из внешнего источника в режиме реального времени. Они могут быть в виде различных типов данных, включая аудио и видеосигналы, сенсорные данные, текстовую информацию и т. д.

Значимость адаптивной обработки потоковых данных заключается в том, что она позволяет системам эффективно обрабатывать и анализировать непрерывно поступающую информацию в режиме реального времени. Это особенно важно в ситуациях, когда данные имеют высокую изменчивость и требуют оперативной обработки.

Адаптивная обработка потоковых данных предоставляет возможность системе адаптироваться к изменениям в потоке данных и оптимизировать процессы обработки на основе текущего состояния данных. Это может включать в себя динамическое изменение параметров алгоритмов обработки, выбор наиболее подходящих методов анализа, управление буферизацией данных и другие техники, которые позволяют достичь оптимальной производительности и результатов обработки.

Примеры приложений адаптивной обработки потоковых данных включают в себя системы распознавания речи, видеонаблюдение, медицинскую диагностику, финансовый анализ и другие области, где требуется быстрая и точная обработка информации в режиме реального времени.

# 1.1.2 Описание проблемы обработки потоковых данных

Проблема адаптивной обработки потоковых данных заключается в необходимости эффективной обработки данных, поступающих в режиме реального времени из различных источников, таких как датчики, сенсоры, устройства IoT и другие. Особенностью потоковых данных является их непрерывный характер и возможность изменения характеристик данных со временем.

В контексте данной задачи, данные с датчиков представляют собой матрицу высот. Основной целью является разработка алгоритма, способного эффективно и точно обнаруживать эти яйца на карте вершин в реальном времени.

Одной из основных проблем является нестабильность и непостоянство данных, поскольку потоковые данные могут быть подвержены шуму, аномалиям и изменениям во времени. Это может привести к ложным срабатываниям или пропускам в обнаружении яиц, что снижает надежность и точность алгоритма.

Другой важной проблемой является необходимость адаптивности алгоритма к изменяющимся условиям работы и требованиям. В контексте данной задачи, например, ситуация может измениться, и новые типы яиц могут быть добавлены на карту вершин. Адаптивный алгоритм должен быть способен автоматически определять скорость потока данных и адаптироваться к ней без необходимости переписывания или переобучения всего алгоритма.

Таким образом, разработка адаптивного алгоритма обработки потоковых данных представляет собой сложную задачу, требующую учета нестабильности данных и способности к адаптации к изменяющимся условиям. Это обеспечит точность, надежность и эффективность обработки потоковых данных и обнаружения яиц на карте вершин.

# 1.2. Обзор существующих подходов к обработке потоковых данных

# 1.2.1 Описание традиционных методов обработки потоков данных их преимущества и недостатки

Традиционные методы обработки потоковых данных включают в себя различные техники и подходы, разработанные для эффективной обработки и анализа непрерывно поступающей информации. Примеры методов обработки потоковых данных:

* буферизация данных;
* потоковая обработка;
* алгоритмы обратной связи;
* методы семплирования и прогнозирования.

Буферизация данных включает в себя использование буферов для хранения и организации потоковых данных. Буферы могут использоваться для временного хранения данных перед их обработкой, а также для сглаживания колебаний в потоке данных. Буферы с разными размерами и стратегиями управления могут быть использованы для обеспечения оптимальной производительности и минимизации задержек в обработке.

Потоковая обработка включает выполнение операций обработки данных непосредственно по мере их поступления в систему. Вместо того, чтобы обрабатывать все данные одновременно, как в пакетной обработке, потоковая обработка позволяет анализировать и обрабатывать данные в реальном времени. Это позволяет системе реагировать на изменения в потоке данных и своевременно принимать решения. Этот подход особенно полезен для систем, где требуется непрерывная обработка данных, например, в финансовых системах, мониторинге сетей или системах реального времени.

Методы семплирования и прогнозирования в обработке потоков данных — это способы анализа и обработки непрерывно поступающей информации для выявления общих тенденций, проведения агрегации и прогнозирования будущих значений.

Семплирование в потоковых данных представляет собой метод, при котором из потока данных выбирается определенное подмножество элементов для анализа. Такой подход позволяет уменьшить объем данных, сохраняя при этом основные характеристики потока. Существуют различные методы семплирования, такие как случайное семплирование, семплирование с перекрытием, равномерное семплирование и другие.

Прогнозирование в потоковых данных — это процесс оценки будущих значений на основе анализа прошлых и текущих данных. Этот метод позволяет предсказать будущее состояние потока данных и использовать эту информацию для принятия решений. Прогнозирование может выполняться с использованием различных статистических и машинно-обучаемых методов, таких как методы временных рядов, регрессионный анализ, нейронные сети и др.

Эти методы позволяют анализировать и использовать только выборочные данные из потоков, что может значительно снизить требования к вычислительным ресурсам. Они также позволяют выполнять прогнозирование и предсказание на основе исторических данных, что может быть полезно для принятия оперативных решений.

Ограничения традиционных методов обработки потоковых данных:

* ограниченные вычислительные ресурсы;
* ограниченная масштабируемость;
* недостаток гибкости.

Ограниченные вычислительные ресурсы. Традиционные методы обработки могут столкнуться с ограничениями вычислительных ресурсов при обработке больших объемов данных или в условиях высокой скорости потока. Это может привести к задержкам в обработке и недостаточной пропускной способности системы;

Ограниченная масштабируемость. Некоторые традиционные методы обработки потоковых данных могут иметь ограничения на масштабируемость. При увеличении объема данных или потоков может потребоваться дополнительная архитектура или оптимизация для обеспечения эффективной обработки;

Недостаток гибкости. Традиционные методы могут быть ограничены в своей способности адаптироваться к изменяющимся требованиям и динамическим ситуациям. В случае изменения формата данных или требований к обработке.

Каждый из этих методов имеет свои преимущества и ограничения, и выбор конкретного метода зависит от требуемых результатов и характеристик потока данных. Однако, в целом, эти традиционные методы обработки потоковых данных являются основными строительными блоками для обработки и анализа непрерывно поступающей информации.

# 1.2.2 Анализ существующих решений по обработке потоковых данных

Пример алгоритма обработки потоковых данных представлен в патенте США №4,276,983. Данный патент описывает устройство и способ для обнаружения контура или формы одного или нескольких изделий с помощью отраженного света.

В настоящем изобретении используется множество пар светоизлучатель/детектор. Каждый детектор света определяет уровень света, отраженного от соответствующего светоизлучателя. Чем ближе объект находится к LEDP, тем больше света отражается от него и регистрируется детектором. Количество отраженного света преобразуется в сигнал и сохраняется. Набор сигналов от всех LEDP дает представление о контуре одного "среза" сканируемой области. Считывание нескольких срезов позволяет получить трехмерное представление о контуре сканируемой области. Изобретение может быть использовано для подсчета количества объектов в сканируемой области, а также для вычисления объема объектов. Когда изобретение используется для подсчета объектов на поверхности, каждый объект должен быть окружен областью, которая отражает меньше света, чем сами объекты. Объекты могут касаться друг друга, если область, где они соприкасаются, отражает меньше света, чем остальные объекты.[1]

Другой задачей изобретения является обеспечение способа подсчета объектов с использованием показаний света, который не требует калибровки или сравнения показаний света от соседних световых детекторов.

На рисунке 3 представлен вид сверху типичного пути, содержащего объекты.



Рисунок 3 - Вид сверху типичного пути

Рисунок 4 представляет собой репрезентативный массив значений сигналов, указывающих на обнаруженный контур траектории.



Рисунок 4 - Репрезентативный массив значений сигнала

Важным аспектом изобретения является то, что LEDP не нужно калибровать, поскольку соседние показания освещенности не нужно сравнивать. Обнаружение объектов осуществляется путем обнаружения изменений в отдельных датчиках и группировки смежных датчиков. Фактически, каждый LEDP может иметь порог вхождения, который отличается от других LEDP.

Следует также отметить, что могут наблюдаться незначительные колебания в показаниях LEDP даже при отсутствии значительных изменений в сканируемой области. Например, в показаниях 22-24 датчик 1 изменяет показания с 91 на 89 и обратно на 91, хотя в этот период под датчиком не проходит никакой объект. Эти незначительные колебания могут быть вызваны рядом факторов, например, вибрацией или загрязнением. Однако такие колебания не влияют на способ изобретения, поскольку они незначительны.

В примере, показанном на рисунке. 3 и 4, для всех датчиков предполагается порог вхождения, равный 99. Это означает, что, когда датчик достигает значения 99, под датчиком находится объект. Объект может находиться или не находиться под датчиком, когда показания света ниже порога вхождения, но, когда показания света находятся на уровне или выше ворога вхождения, под датчиком определенно находится объект. порог понижения равен 9 что означает, что часть объекта прошла мимо датчика, когда показания света, которые были выше порога вхождения, падают на 9.

Порог нового объекта равен 4, что означает, что, если показания света падают на 9, вызывая подсчет, но все еще выше порога вхождения, а затем поднимаются по крайней мере на 4, будет считаться, что обнаружен новый объект.

Группа формируется из включенных смежных датчиков. При обнаружении перепада равного порогу понижения (9) генерируется счет, при этом у всей группы отключается возможность генерации счета.

При падении показателей датчиков ниже порога вхождения, датчики отключаются.

Однако этот алгоритм обладает рядом недостатков, которые необходимо учесть при его применении.

Во-первых, алгоритм не учитывает шум, который может быть обнаружен датчиками. Он предполагает, что различия между датчиками обусловлены только различной калибровкой. Однако в реальных условиях существует возможность появления помех и шумов, которые могут искажать получаемые датчиками данные. Это может приводить к неточным результатам определения контура и подсчета объектов.

Во-вторых, алгоритм не учитывает большое расстояние между датчиками. Получаемые сигналы от датчиков могут иметь слишком большое друг к другу расстояние, что может затруднить точное определение контура объектов. Необходимо разработать специальные методы для учета этого малого расстояния между датчиками и минимизации возможных ошибок.

В-третьих, алгоритм не учитывает разную скорость движения объектов по конвейерной ленте. Это также может привести к неточным результатам при определении контура и подсчете объектов.

Пример способа обработки данных представлен в патенте RU №2309436 C2. Изобретение относится к способу обработки данных, в частности к обработке сейсмических данных, собранных с использованием отдельного датчика сейсмических данных.

Известный способ сбора сейсмических данных схематически иллюстрируется на рисунке 5. В известном способе сейсмические данные собираются матрицей 1, содержащей множество датчиков 2, 2'. Датчики 2, 2' на рисунке 5 скомпонованы в линейной матрице, но они могли бы быть альтернативно скомпонованы, например, в двумерной матрице. В линейной матрице 1 датчиков, показанной на рисунке 5, каждый датчик разнесен от смежных датчиков, по существу, на постоянное расстояние. Расстояние d между центром одного датчика и центром смежного датчика составляет, например, приблизительно 3,125 м, хотя известные варианты систем сбора сейсмических данных имеют широко варьирующееся расстояние разнесения датчиков.

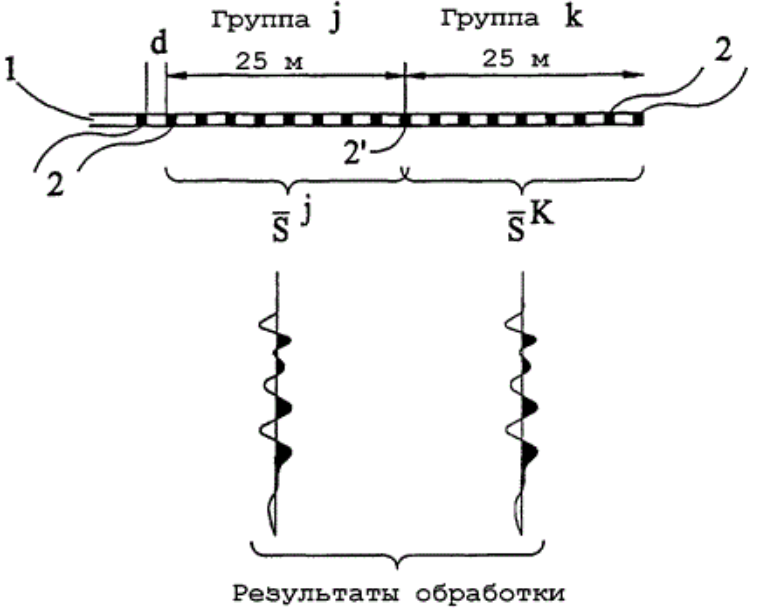


Рисунок 5 – схематическая иллюстрация известного способа сбора сейсмических данных

Типичная матрица сейсмических датчиков содержит большое число датчиков. Известная практика состоит в том, что датчики в матрице "жестко закреплены" в группы ближайших датчиков, где каждый датчик в группе принимает, по существу, одинаковую составляющую сигнала из подповерхностной мишени, которая должна быть визуализирована. Группирование предназначено для улучшения отношения сигнал/шум посредством электрического слияния аналоговых сигналов из каждого датчика в один сигнал. Этот один сигнал представляет оценку сигнала, который является общим для всех датчиков в группе, и является известным в основном как "общий сигнал".

Данное изобретение обеспечивает способ обработки данных, содержащий этапы: определения, из первой последовательности данных, собранных на первом датчике посредством дискретизации сигнала, имеющего ассоциированную неопределенность, и из второй последовательности данных, собранных посредством дискретизации по существу такого же сигнала, имеющего ассоциированную неопределенность, на втором датчике, пространственно разделенном от первого датчика, последовательности, индикативной по мере разброса первой и второй последовательностей данных, обеспечивая тем самым оценку исходной неопределенности сигнала.

На рисунке 6 представлен схематический вид системы сбора данных с отдельными сейсмическими датчиками, подходящей для использования со способом обработки согласно варианту осуществления данного изобретения.

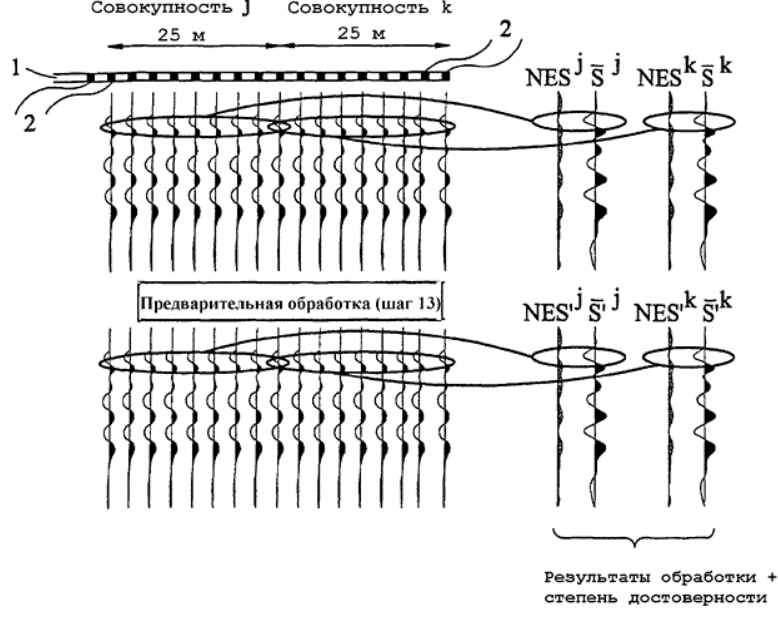


Рисунок 6 - схематический вид системы сбора данных с отдельными сейсмическими датчиками

На рисунке 7 изображена схематическая иллюстрация способа обработки данных согласно другому варианту осуществления настоящего изобретения

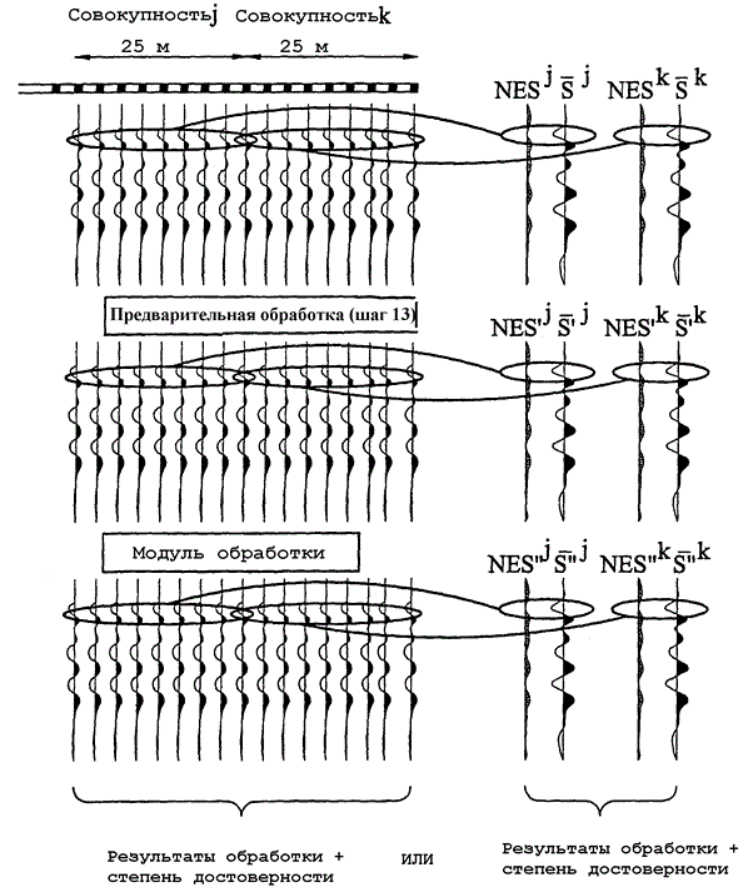


Рисунок 7 - иллюстрация способа обработки данных

На рисунке 8 изображена блок-схема, иллюстрирующая принципиальные шаги способа обработки согласно первому варианту воплощения настоящего изобретения. В указанном варианте воплощения оцениваются как мера разброса, так и общий сигнал.

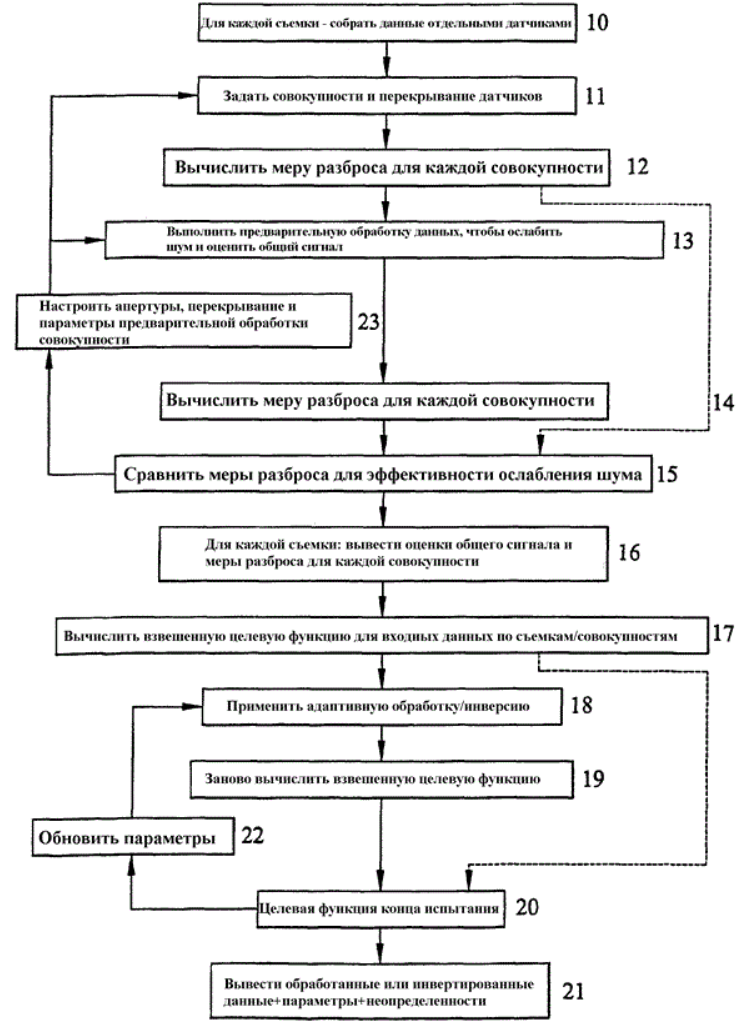


Рисунок 8– блок-схема, иллюстрирующая принципиальные шаги способа обработки

Отслеживание неопределенности в процессе обработки данных с использованием нелинейных операций обработки часто является не таким наглядным, как при использовании линейных операций. Поэтому в противоположность варианту осуществления, этап присваивания оценок общего сигнала, , чтобы представлять выходные сигналы из каждой совокупности, опускается. В варианте по рисунку 7 все последовательности данных, производимые индивидуальными датчиками, проходят через дополнительный модуль обработки в виде совокупностей. Эллипсы на рисунке 7 показывают, какие последовательности данных содержатся в j- и k-совокупностях. Последовательности, содержащие оценки общего сигнала и разброс данных внутри каждой совокупности, могут быть получены на любой желательной стадии в обработке данных, либо перед, либо после того, как данные были обработаны нелинейным модулем обработки. Еще в одном из вариантов осуществления мера разброса выходных сигналов датчиков в группе используется для выбора параметров предварительной обработки данных для уменьшения отношения сигнал/шум в пределах каждой совокупности. То есть параметры шагов предварительной обработки в блоках 11 и/или 13 выбираются на основе изменения нормализованной оценки разброса, вычисленной на этапах 12 и 14. Данная процедура схематически показана на рисунке 8 этапом 23 - перед тем как выходы датчиков воображаемо группируются. Следовательно, в данном варианте воплощения параметры этапов 11 и 13 предварительной обработки выбираются на основе нормализованной оценки разброса (или другой меры разброса) последовательностей данных, выводимых датчиками. В указанном варианте воплощения пробная выборка последовательностей данных, выводимых датчиками, воображаемо группируется в совокупности на этапе 11, предварительно обрабатывается на этапе 13, и нормализованные меры разброса вычисляются на этапах 12 и 14. Затем, согласно уровню улучшения отношения сигнал/шум разброса, заметного на этапе 15, варьируются параметры предварительной обработки на этапах 11 и 13 либо вручную, либо автоматически, и повторно вычисляются нормализованные меры разброса; шаг варьирования параметров этапов 11 и 13 предварительной обработки повторяется до тех пор, пока не будет минимизирована нормализованная мера разброса, взятая, чтобы представлять отношение сигнал/шум в предварительно обработанных сигналах. Указанный вариант осуществления показан схематически этапом 23.

# 1.2.3 Анализ методов фильтрации

Для выбора оптимального фильтра для поставленной задачи необходимо проанализировать существующие методы фильтрации. Обзор включает различные типы фильтров, такие как скользящее среднее, фильтр Гаусса, медианный фильтр, с их особенностями и ограничениями, которые необходимо учесть при выборе соответствующего фильтра.

При выборе определенного фильтра для поставленной задачи, необходимо учесть такие параметры, как частотный диапазон сигнала, требуемое подавление шума или выбросов, а также особенности конкретной системы или оборудования. Также следует учитывать вычислительные возможности доступных алгоритмов и время обработки данных.

Фильтр Гауса также может быть применен в обработке и фильтрации данных. В этом случае он используется для сглаживания шумов и выбросов в данных. Применение фильтра Гауса к данным основывается на том же математическом принципе, который используется для размытия изображений. Он основан на математической функции Гаусса, которая позволяет уменьшить шумы и удалить выбросы в данных. Передаточная функция гауссова фильтра высоких частот (ФВЧ Гаусса) с частотой среза, расположенной на расстоянии D0 от центра частотного прямоугольника, задается формулой

, (1)

Результат применения фильтра Гаусса представлен на рисунке 5.



Рисунок 5 - До и после применения фильтра Гауса

Медианный фильтр — это тип фильтра, применяемый для обработки сигналов и изображений. Для применения медианного фильтра к сигналу или изображению определяется окно определенного размера, которое перемещается по всему входному сигналу или изображению. В каждом положении окна пиксели (или значения входного сигнала) внутри окна сортируются, а затем выбирается медианное значение. Это медианное значение затем заменяет значение пикселя (или входного сигнала) в центре окна.

Фильтр Гаусса можно быть требовательным к ресурсам, особенно если он используется для обработки больших потоков данных. Это связано с тем, что применение фильтра Гаусса требует выполнения множества математических операций. Поэтому в случае работы с большими объемами данных или приложений реального времени может потребоваться значительное количество вычислительных ресурсов и времени для обработки.

Применение медианного фильтра позволяет эффективно уменьшать влияние выбросов и шумов на сигнал или изображение, сохраняя при этом краевые детали и структуру. Особенно медианный фильтр хорошо работает в случаях, когда есть значительные артефакты или шумы, которые необходимо удалить без размытия изображения. Пример представлен на рисунке 6. Результат фильтрации представлен на рисунке 7.



Рисунок 6 – Пример расчета медианы



Рисунок 7 - До и после применения медианного фильтра

Медианный фильтр является требовательным к ресурсам в силу необходимости сортировки значений в окне фильтрации. При применении медианного фильтра к изображению или сигналу, каждый пиксель или значение проходит через окно фильтрации, в котором выбирается медианное значение из всех значений в окне.

Для сортировки значений в окне фильтрации требуется проход по всему окну и выполнение операции сортировки. Это означает, что при каждом применении медианного фильтра к изображению или сигналу требуется большое количество вычислительных ресурсов, особенно при работе с большими окнами фильтрации или изображениями высокого разрешения.

Кроме того, медианный фильтр также требователен к памяти, так как для хранения всех значений в окне фильтрации необходимо выделить достаточно большой объем памяти.

Таким образом, медианный фильтр является требовательным к ресурсам из-за необходимости выполнения операции сортировки значений в окне фильтрации и выделения достаточного объема памяти для хранения этих значений.

Скользящее среднее — это метод фильтрации сигналов, который используется для сглаживания данных путем вычисления среднего значения по подмножеству точек данных, перемещающемуся по всему набору данных. Размер этого подмножества, называемого окном, определяется заранее.

Для применения скользящего среднего определяется размер окна, например, 5 точек данных для сигнала. Окно перемещается по всему сигналу, и для каждого положения окна вычисляется среднее значение точек данных внутри окна. Это среднее значение затем заменяет значение в центре окна.

Скользящее среднее полезно для сглаживания шумов и колебаний в сигнале, позволяя выявить общие тенденции и уменьшить влияние случайных изменений. Однако следует помнить, что скользящее среднее также может сглаживать резкие изменения и детали сигнала, что может быть нежелательным в некоторых ситуациях.

На рисунке 8 представлена матица весов. Результат фильтрации представлен на рисунке 9.



Рисунок 8 – Матрица весов



Рисунок 9 – До и после фильтрации

Необходимо учитывать, что выбор конкретного фильтра должен основываться на анализе сигнала и его особенностях. Для каждой ситуации требуется провести анализ спектра частот шумов и артефактов, а также принять во внимание желаемые параметры сигнала, такие как разрешение, стабильность и сохранение деталей.

Таким образом, для эффективной фильтрации данных с датчиков необходимо выбирать соответствующий фильтр, учитывая специфику сигнала и требования к итоговым данным. Подходящий выбор фильтра обеспечит эффективную фильтрацию шумов и артефактов, а также сохранение важных деталей сигнала, что в конечном итоге повысит качество и надежность результатов обработки сигналов.

При наличии шума или непостоянной скорости движения объектов, стандартные методы фильтрации могут давать неточные результаты.

# Постановка задачи исследования

# 2 Описание алгоритма

В данной главе производится описание разработанного робастного адаптивного алгоритма распознавания выпуклых объектов в потоковых данных инвариантный к уровню как собственных шумов оптических сенсоров, так и внешних помех. Разработанный алгоритм не требователен к вычислительным ресурсам и предназначен для реализации на базе SOC (system on chip).

# 2.1 Описание входных данных алгоритма

Входными данными разработанного алгоритма являются массив показаний датчиков определения расстояния. Данные датчики могут быть заменены, например, на датчики приближения.

На рисунке 1 представлен репрезентативный массив значений входных данных алгоритма, где по оси X отмечено положение десяти датчиков от a до j, а по оси Y измеренные построчно расстояния.

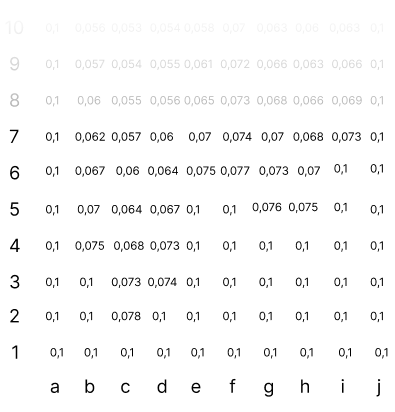


Рисунок 1 - Репрезентативный массив значений входных данных алгоритма

Примером датчика определения расстояния может служить VL53L0X, который работает на основе принципа лидара. Он использует инфракрасный лазер для измерения расстояния до объекта на основе времени, за которое отраженный лазерный сигнал возвращается обратно к датчику.

Набор сигналов от всех датчиков дает представление о контуре одного "среза" сканируемой области. Считывание нескольких срезов позволяет получить трехмерное представление о контуре сканируемой области.

# 2.2 Обзор алгоритма

Алгоритм предназначен для обработки данных от датчиков и обнаружения объектов в системе. Он включает в себя этапы фильтрации данных, контроля скорости прохождения объектов через датчики и обнаружения шарообразных объектов.

Первый этап - фильтрация данных. Все поступающие от датчиков показания проходят через специальный фильтр, который удаляет ненужные сигналы низкой интенсивности, шумы и помехи, оставляя только информацию о рассматриваемых объектах.

Второй этап - алгоритм скользящего окна с контролем скорости прохождения объектов через датчики. Датчики размещаются в определенном порядке и показания с них используются для определения скорости движения объектов в пределах системы. Для этого применяется скользящее окно, которое позволяет анализировать некоторый период времени и выяснить, сколько объектов проходит через датчики за данный промежуток времени. При этом высчитывается скорость прохождения.

Третий этап - алгоритм обнаружения объектов. После фильтрации данных и контроля скорости прохождения, происходит обнаружение объектов шарообразной формы.

# 2.3 Фильтрация данных

Для решения поставленной задачи фильтрации данных необходимо учесть ряд свойств и выбрать наиболее подходящий для данного случая фильтр. Анализ методов фильтрации, представленный выше, позволяет систематизировать подходы к обработке данных и выбрать оптимальный фильтр. Процесс фильтрации данных разделен на несколько этапов.

Первый этап – сбор данных. На данном этапе процесса фильтрации производится сбор и запись данных с датчиков для последующей обработки.

Далее следует этап предобработки данных. В процессе предобработки осуществляется фильтрация данных от возможных выбросов или шумов. Также учитываются физические ограничения, такие как предельные значения, и производится удаление данных с низкой интенсивностью. Этот этап позволяет подготовить данные к применению фильтра.

Последний этап – применение фильтра. Важным шагом в процессе фильтрации является выбор конкретного фильтра, который будет применяться к данным. Целью применения фильтра является удаление шумов, выбросов или несовершенств оборудования.

При выборе необходимого фильтра для поставленной задачи, необходимо учесть такие параметры, как частотный диапазон сигнала, требуемое подавление шума или выбросов, а также особенности конкретной системы или оборудования. Также следует учитывать вычислительные возможности доступных алгоритмов и время обработки данных.

Таким образом, для эффективной фильтрации данных с датчиков необходимо выбирать соответствующий фильтр, учитывая специфику сигнала и требования к итоговым данным. Подходящий выбор фильтра обеспечит эффективную фильтрацию шумов и артефактов, а также сохранение важных деталей сигнала, что в конечном итоге повысит качество и надежность результатов обработки сигналов.

Рисунок 2 демонстрирует важность порога вхождения (ПВ). Алгоритм требует установления ПВ, который фильтрует сигналы низкой интенсивности.

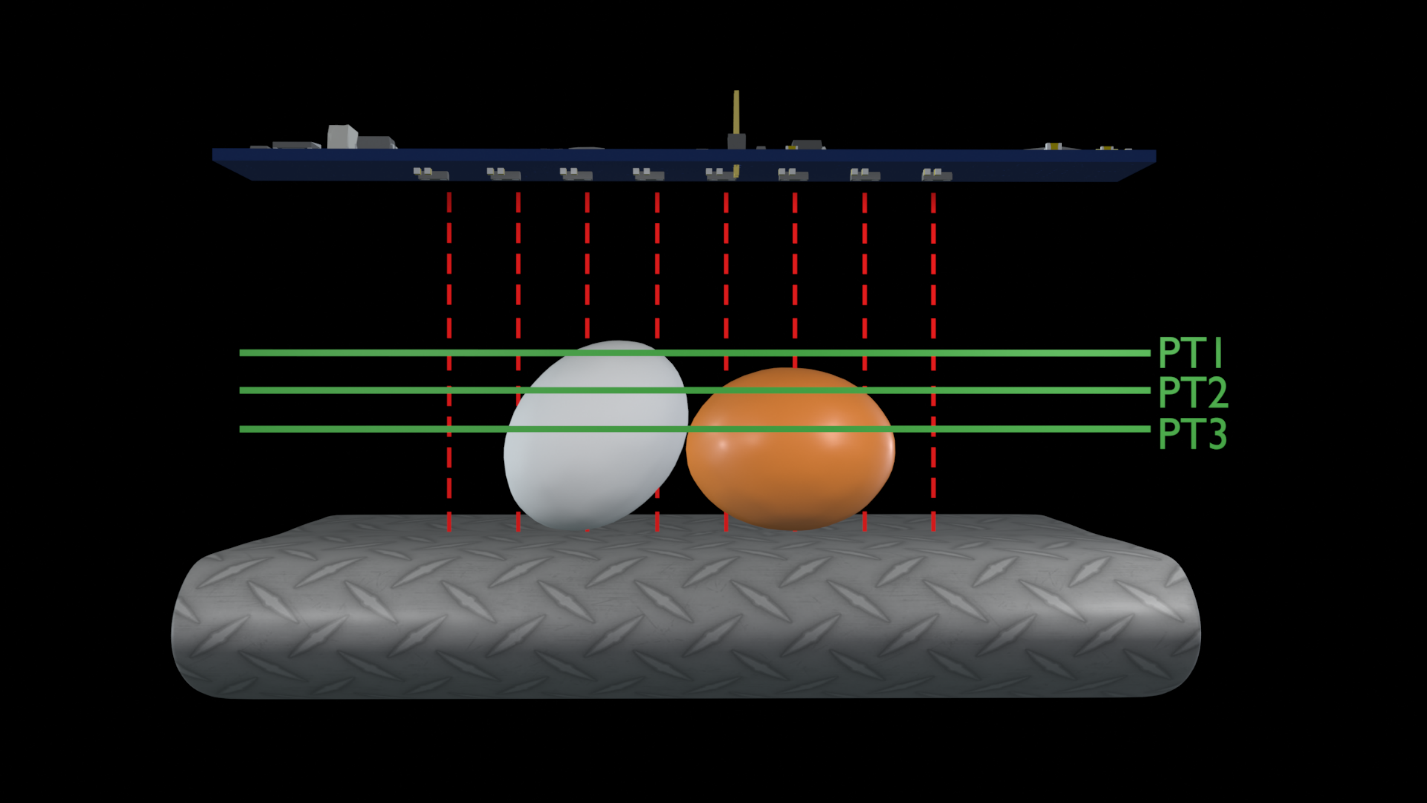


Рисунок 2 – Демонстрация важности ПВ

PT2 является показателем подходящего ПВ, поскольку он представляет собой уровень сигнала, превышающий меньшие части периметра яиц, и он не настолько высок, чтобы не указать на присутствие даже самого маленького яйца (или другого объекта), подлежащего сканированию. Теоретический ПВ на уровне, указанном PT1, был бы слишком низким, поскольку он был бы меньше, чем расстояние до самого маленького яйца. Поэтому это яйцо не будет подсчитано. Теоретический ПВ на уровне, обозначенном PT3 будет слишком высоким, так как при таком уровне ПВ невозможно обнаружить "края" объектов.

# 2.4 Описание алгоритма обнаружения объектов

Основной принцип работы алгоритма обнаружения объектов заключается в детектировании и создании группы вокруг объекта, добавление датчиков в существующую группу, и поиск вершины путем выбора тройки датчиков с наименьшей суммой и определения центра этой тройки. Также алгоритм включает в себя отключение от группы на основе обнаружения другого объекта, корректировку принадлежности датчиков к группам путем проверки экстремумов, и отключение датчиков при прохождении объектов.

Блок-схема алгоритма представлена на рисунке 3.

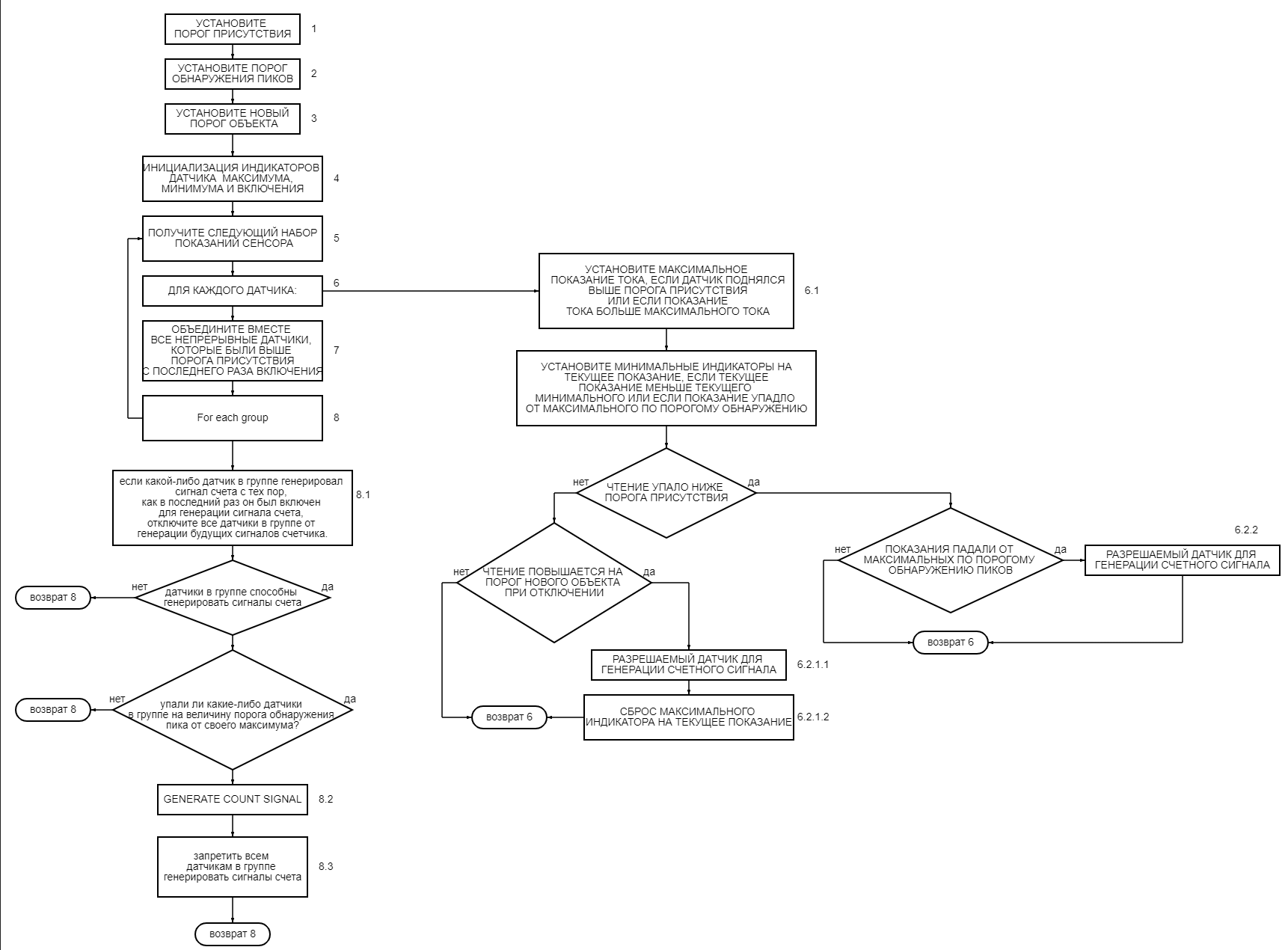


Рисунок 3 – Блок-схема алгоритма

Ниже приводится краткое пошаговое описание алгоритма в отношении показаний матрицы сигналов, показанных на рисунке 4.

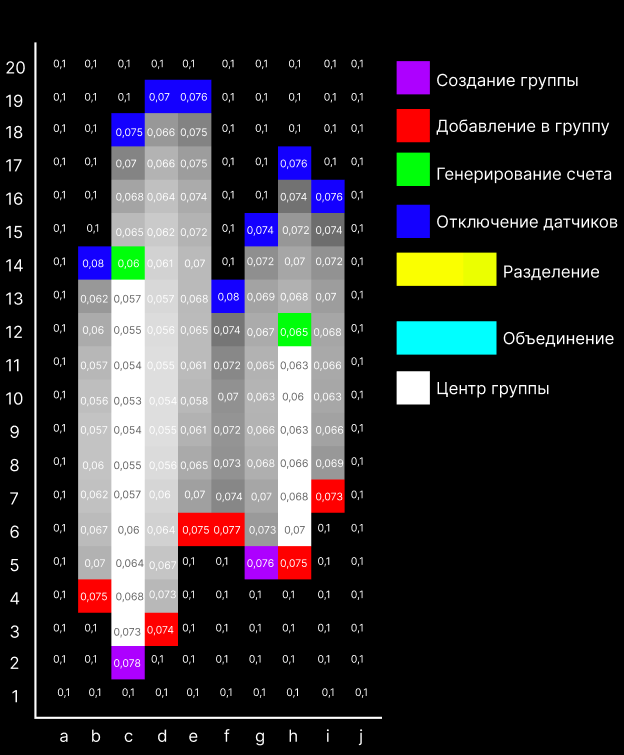


Рисунок 4 - Показания матрицы сигналов.

Показание 1. Ни одно показание не превысило, все датчики остаются выключенными.

Показание 2. Показание датчика c превышает ПВ. Так как по соседству отсутствуют созданные группы, то создается новая группа (группа №1). Центр данной группы устанавливается на датчик c.

Показание 3. Показание датчика d превышает ПВ. Так как по соседству существует группа, то датчик d добавляется в данную группу (группа №1). Центр данной группы устанавливается на датчик c.

Показание 4. Показание датчика b превышает ПВ. Так как по соседству существует группа, то датчик b добавляется в данную группу (группа №1). Центр данной группы устанавливается на датчик c.

Показание 5. Показание датчика g превышает ПВ. Так как по соседству отсутствуют созданные группы, то создается новая группа (группа №2). Показание датчика h превышает ПВ. Так как по соседству существует группа, то датчик h добавляется в данную группу (группа №2). Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на на датчик h.

Показание 6. Показание датчика e превышает ПВ. Так как по соседству существует группа, то датчик e добавляется в группу №1. Показание датчика f превышает ПВ. Так как рядом находятся 2 группы, то датчик присоединятся к той, где показатель соседнего датчика меньше. Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на датчик h.

Показание 7. Показание датчика i превышает ПВ. Так как рядом находится группа №2, то датчик присоединяется к ней. Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на на датчик h.

Показание 8-11. Центр группы 1 устанавливается на датчик с, а центр группы 2 устанавливается на на датчик h.

Показание 12. Значение тройки датчиков с наименьшей суммой группы 2 упало больше, чем на 5%. Это означает, что найдена вершина объекта и генерируется счет. Всем датчикам в этой группе устанавливается флаг, который запрещает генерировать счет. Центр группы 1 устанавливается на датчик с.

Показание 13. Показание датчика f упало ниже ПВ. Данный датчик отключается. Центр группы 1 устанавливается на датчик с.

Показание 14. Значение тройки датчиков с наименьшей суммой группы 2 упало больше, чем на 5%. Это означает, что найдена вершина объекта и генерируется счет. Всем датчикам в этой группе устанавливается флаг, который запрещает генерировать счет. Показание датчика b упало ниже ПВ. Данный датчик отключается.

Показание 15. Показание датчика g упало ниже ПВ. Данный датчик отключается.

Показание 16. Показание датчика i упало ниже ПВ. Данный датчик отключается.

Показание 17. Показание датчика h упало ниже ПВ. Данный датчик отключается.

Показание 18. Показание датчика c упало ниже ПВ. Данный датчик отключается.

Показание 19. Показание датчиков d, e упали ниже ПВ. Данные датчики отключаются.

# 2.5 Описание алгоритма скользящего окна с контролем скорости

При наличии шума или непостоянной скорости движения объектов, стандартные методы фильтрации могут давать неточные результаты.

Один из аналогов алгоритма скользящего окна с контролем может быть применен для решения подобных проблем. Этот алгоритм, основываясь на предположении о непрерывности скорости движения объектов, позволяет более точно определить их скорость и направление движения для корректной работы алгоритма обнаружения объектов.

Алгоритм использует окно заданного размера n, которое перемещается по временному ряду данных. Размер окна n зависит от количества объектов и максимальной скорости, чем больше объекты, тем больше значение n. В каждом положении окна происходит анализ данных для обнаружения объектов, на основе этого происходит оценка их скорости, что определяет размер окна m (1). При заполнении окна m данными вычисляется среднее арифметическое всех столбцов, которое затем используется для обработки алгоритмом поиска объектов.

*(1)*

где j – количество объектов, на основе которых производятся расчеты;

xi – количество детектированных показателей i-го объекта;

k – коэффициент пропорциональности, устанавливается экспериментально по результатам анализа соответствия формы объектов заданному образцу.

Основное преимущество данного алгоритма состоит в том, что он учитывает изменение скорости движения объектов во времени и способен дать более точные результаты при наличии шума или непостоянной скорости. Схема работы изображен на рисунке 5.

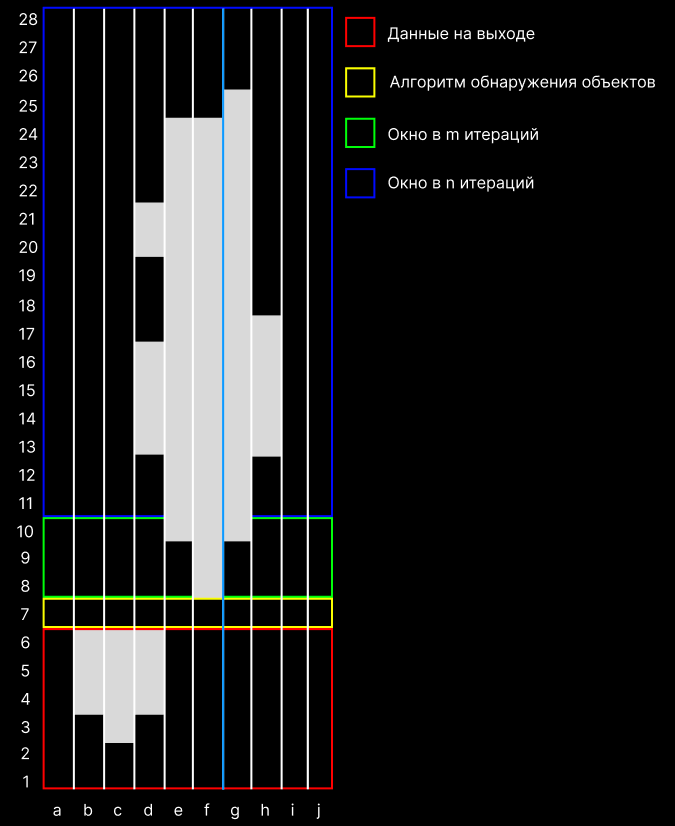


Рисунок 5 – Схема работы алгоритма классификации и скользящего окна

Блок-схема алгоритма представлена на рисунке 6.

Рисунок 6 – Блок-схема алгоритма

# 2.6 Выводы по разделу