Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ФИЗИКИ

КАФЕДРА РАДИОФИЗИКИ

Направление: 10.03.01 Информационная безопасность
Профиль: Безопасность автоматизированных систем

КУРСОВАЯ РАБОТА

РЕШЕНИЕ ЗАДАЧИ ОБНАРУЖЕНИЯ ДВИЖУЩИХСЯ ОБЪЕКТОВ С ДАННЫХ КАМЕР БЕСПИЛОТНОГО АВТОМОБИЛЯ ДЛЯ ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ ЗАКРЫТЫХ ТЕРРИТОРИЙ

Студент 3 курса группы 06-751
«04» июня 2020 г. _____ (Галлямов А.З.)

Научный руководитель
к.т.н., доцент
«__» _____2020 г. _____ (Чикрин Д.Е.)

Казань-2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Обеспечение безопасности закрытых территорий всегда было актуальной задачей, но с развитием информационных технологий, с одной стороны, ее решение становится все более важным и критичным, с другой стороны, способы решения поставленной задачи расширяются. Одним из таких расширений является использование беспилотных транспортных средств и алгоритмов обнаружения движения с данных их камер с целью обеспечения защиты закрытых территорий различных предприятий и т.д. с целью возможного предотвращения различного рода утечек информации.

Сегодня активно разрабатываются различные методы обнаружения движущихся объектов, так как данная технология является очень перспективной в связи с тем, что спектр отраслей, нуждающихся в детектировании движущихся объектов, постоянно расширяется. Начиная от обеспечения информационной защиты территорий дата-центров, банков, заводов, хранящих большое количество персональной и коммерческой информации, обнаружения и возможного предотвращения несанкционированного доступа на них (в том числе и с использованием беспилотных транспортных средств), заканчивая активно развивающейся сегодня и достаточно перспективной сферой беспилотных транспортных средств, которым просто необходимо умение обнаруживать движущиеся объекты для эффективного и безопасного функционирования. Также технология распознавания движущихся объектов позволяет существенно ускорить работу таких алгоритмов, как HOG (histogram of oriented gradients) или SIFT (scale-invariant feature transform), нацеленных на определение определенного класса объектов (например, понять кто изображен: человек или автомобиль), путем сокращения количества областей на изображении, подаваемых на вход таких алгоритмов в качестве цели. Данные методы классификации объектов позволяют обеспечить более надежную защиту закрытых территорий. В связи с этим, представленная работа, посвященная

решению задачи определения движущихся объектов с данных камер является весьма актуальной.

Целью курсовой работы является решение задачи определения движущихся объектов с данных камер беспилотного автомобиля для обеспечения информационной защиты закрытых территорий.

Для того, чтобы достичь поставленной цели нам необходимо решить следующие задачи:

1. Рассмотреть существующие на сегодняшний день методы распознавания движущихся объектов с данных камер.
2. Изучить один из существующих готовых алгоритмов определения движущихся объектов с данных камер.
3. Использовать реализацию выбранного алгоритма распознавания движущихся объектов для тестирования его работы на тестовых видео.
4. Оценить работу выбранного алгоритма на тестовых видео.

ГЛАВА 1. МЕТОДЫ ОБНАРУЖЕНИЯ ДВИЖУЩИХСЯ ОБЪЕКТОВ С ДАННЫХ КАМЕР

Задача обнаружения движущихся объектов с данных камер существует довольно давно, но на сегодняшний день так и не имеется единственного и конкретного ее решения, в силу различных факторов таких, как, например, различные природные условия при съемке: туман, снег, дождь, различный уровень шума на входных изображениях, а также положение камеры: она может быть расположена статично либо же размещена на движущемся объекте. Так, в случае, если камера расположена статично, то все кадры с нее будут обладать довольно однообразным и схожим фоном с возможными отличиями освещения, в случае же, если камера движется, то возможна такая ситуация, когда камера снимает неподвижный объект, но в кадре он будет перемещаться. Все это приводит нас к тому, что в одних случаях различные методы распознавания движущихся объектов могут быть хорошо применимы, а в других демонстрировать неприемлемый результат.

Алгоритмам обнаружения движущихся объектов, работающих в автоматических системах обнаружения движения (например, системы обнаружения проникновения) необходимо обладать: невысокой вычислительной сложностью, способностью к корректной и стабильной работе при различных условиях освещения, погодных условий, а также способностью не срабатывать на объекты, не имеющие значения, такие как качающиеся ветки и листья. Рассмотрим некоторые наиболее распространенные методы детектирования движущихся объектов с данных камер.

1.1. Optical flow

Optical flow (или оптический поток) - это фундаментальный метод расчета движения интенсивности изображения, которая в дальнейшем может быть приписана движению различных объектов в сцене [10, 11, 12]. Данный

метод позволяет определить смещение каждой отдельно взятой точки в кадре, таким образом появляется возможность построить поле скоростей.

Как было сказано, оптический поток является чрезвычайно фундаментальной концепцией и данный метод в том или ином виде используется в большинстве алгоритмов обработки видео. Методы нахождения оптического потока основаны на вычислении оценок движения интенсивностей изображения во времени в видеопотоке. Полученные в результате поля потока могут быть проанализированы, с целью дальнейшей сегментации изображения на области, которые в свою очередь в дальнейшем могут быть связаны с движущимися объектами.

Кроме того алгоритмы оптического потока или оценки движения также могут использоваться для обнаружения и определения границ независимо движущихся объектов. Необходимо отметить, что методы, основанные на вычислении оптического потока, сложны в вычислительном отношении и, в связи с этим, требуют быстрых аппаратных и программных решений для реализации. Так как оптический поток является принципиально дифференциальной величиной, его оценка довольно сильно чувствительна к шуму, улучшение же чувствительности к шуму в свою очередь может означать увеличение сложности.

Отправной точкой для анализа метода оптического потока является предположение, что пиксели от кадра к кадру сохраняют свою интенсивность, т.е. значение яркости не меняться со временем.

Рассмотрим подробнее принцип работы метода вычисления оптического потока. Пусть у нас имеется изображение, которое непрерывно изменяется во времени. Обозначим яркость пикселя с координатами (x, y) через $I_t(x, y)$ в момент времени t. Если считать имеющееся динамическое изображение функцией времени и положения, то в следующий момент времени яркость пикселя можно расписать через разложение в ряд Тейлора:

$$I_{t+\Delta t}(x+\Delta x, y+\Delta y) = \tag{1.1}$$
$$= I_t(x,y) + \frac{\partial I_t}{\partial x}\Delta x + \frac{\partial I_t}{\partial y}\Delta y + \frac{\partial I_t}{\partial t}\Delta t + O(\partial^2),$$

где $\frac{\partial I_t}{\partial x}, \frac{\partial I_t}{\partial y}, \frac{\partial I_t}{\partial k}$ – это частные производные I_t. Считается, что смещение пикселя за маленький промежуток времени незначительно, т.е. $(\Delta x, \Delta y) \to 0$ при $\Delta t \to 0$, в этом случае можно считать, что значение яркости пикселя практически не изменяется, следовательно, можно пренебречь остаточным членом ряда Тейлора:

$$I_{t+\Delta t}(x+\Delta x, y+\Delta y) == I_t(x,y) - \frac{\partial I_t}{\partial t} == \frac{\partial I_t dx}{\partial x dt} + \frac{\partial I_t dy}{\partial y dt} \tag{1.2}$$

Уравнение (1.2) называется уравнением оптического потока. После этого необходимо построить вектор скорости пикселя $c = \left(\frac{dx}{dt}, \frac{dy}{dt}\right) = (u, v)$. Также вводится условие гладкости изменения скорости пикселя для полноты постановки задачи. В результате введения такого условия мы получаем, что наша исходная задача упрощается к процессу минимизирования квадратичной ошибки $\left(\frac{\partial I_k}{\partial x}u, \frac{\partial I_k}{\partial y}v, \frac{\partial I_k}{\partial k}\right)^2$ при таких ограничениях, как: $v_x^2 + v_y^2 = 0,\ u_x^2 + u_y^2 = 0$. В результате такой процедуры минимизации, которая применения к каждому отдельному пикселю текущего кадра видео, обеспечивается возможность построение поля векторов смещения для всех пикселей текущего изображения.

В действительности же условие постоянство яркости выполняется достаточно редко потому, что от кадра к кадру могут меняться как глобальные условия освещения, так и освещенность самого движущегося объекта. Несмотря на это, стоит отметить, что метод, основанный на предположении о постоянстве яркости, на практике показывает довольно хорошие результаты.

Нужно также отметить, что данный метод является локальным. Это означает, что в процессе определении смещения некоторого пикселя учитывается локальная окрестность этого пикселя, т.е. некоторая

область вокруг него. По этой причине невозможно определить точные смещения внутри равномерно окрашенных участков кадра, которые имеют размер больший, чем размер локальной области. Стоит сказать, что подобные участки на реальных кадрах видео в действительности встречаются не так уж и часто. Помимо этого возможны такие случаи, когда для некоторых текстур на изображении просто невозможно определить корректное смещение, т.е. получается такая ситуация, что движение вроде есть, но в какую именно сторону происходит это движение не понятно. Подобной проблеме подвержен не только рассмотренный метод. Например, человеческий глаз такое движение также может воспринять неоднозначно.

1.2. Отделение фона

Метод, основанный на вычитании фона, является более простым подходом к решению задачи обнаружения движущихся объектов. Предполагает, что для рассматриваемого видеофайла построена модель фона:

$$F = \{F(x, y), 0 \leq x \leq width, 0 \leq y \leq height\} \quad (1.3)$$

При этом нужно, чтобы данная модель постоянно обновлялась, для того, чтобы учитывать изменение положение камеры и изменения освещенности.

Работа метода вычитании фона состоит из двух шагов:

1. Сперва происходит вычитание из текущего кадра фонового изображения, т.е. происходит процесс попиксельного вычитания интенсивностей:

$$D_i(x, y) = abs\big(I_i(x, y) - F(x, y)\big), i = \overline{1, N}, \quad (1.4)$$

   где $I_i(x, y)$ – интенсивность i-ого кадра.

2. После этого происходит соотнесение пикселей либо фону, либо движущемуся объекту. Так, пиксель определяется как принадлежащий движущемуся объекту в том случае, если полученная разность интенсивности текущего кадра видео и фона для данного конкретного пикселя превосходит некоторое заданное пороговое число, в противном случае, пиксель считается принадлежащим фону.

$$M_i(x,y) = \begin{Bmatrix} 255\ , D_i(x,y) \geq \rho \\ 0\ , D_i(x,y) < \rho \end{Bmatrix}, i = \overline{1,N} \quad (1.5)$$

Способность к корректному обнаружению движущихся объектов в видео посредством такого метода, стоит отметить, весьма сильно зависит от модели фона изображения. Так, некорректно построенная модель фона может ошибочно посчитать за движущийся объект мельчайшие изменения освещенности или же, например, не включить в себя объекты, которые совершают небольшие движения, но не обладают информацией и не представляют какого-либо интереса (например, мелкие частицы в воздухе, качающиеся ветки и листья деревьев и т.п.), что, конечно же, не позволит нам получить приемлемый результат. Все большое множество самых различных методов построения фоновой модели или фонового изображения можно разделить на две большие группы. Рассмотрим их подробнее.

1.2.1. Нерекурсивные методы

Особенностью нерекурсивных методов является то, что для процесса обновления фоновой модели определенного видеокадра применяются данные об интенсивностях и цветах пикселей, во первых, самого текущего видеокадра, во вторых, некоторого определенного набора предыдущих фоновых моделей [17]. Рассмотрим принципы работы некоторых методов, которые относятся к группе нерекурсивных:

- *Метод, основанный на вычитании из текущего кадра видео предыдущий кадр*. Согласно данному методу, как можно догадаться, модель фона F_k для конкретного кадра I_k, который мы собираемся анализировать, совпадает с предыдущим кадром видео, т.е. мы получаем, что $F_k = I_{k-1}$. В таком случае на первом шаге метода вычитания фона вычисляется разница последовательно идущих кадров:

$$D_k(x,y) = abs\big(I_k(x,y) - F(x,y)\big) = \\ = abs\big(I_k(x,y) - I_{k-1}(x,y)\big), k = \overline{2,N} \quad (1.6)$$

- *Метод, основанный на усреднении определенного количества предшествующих кадров*. Обозначим в качестве s количество кадров, по которым будет происходить процесс построения фона. В этом случае модель фона F_k для текущего кадра I_k определяется следующим образом:

$$F_k(x,y) = \frac{1}{s}\sum_{j=0}^{s-1} I_{k-j}(x,y) \tag{1.7}$$

- *Метод, основанный на вычислении медианы фиксированного количества предшествующих кадров.* Здесь, как и в рассмотренном ранее случае, в качестве s обозначается количество кадров, на основании которых происходит процесс обновления модели фона. Модель фона F_k определяется следующим образом:

$$F_k(x,y) = med_{j=\overline{0,s-1}}\{I_{k-j}(x,y)\} \tag{1.8}$$

Основным преимуществом нерекурсивных методов вычитания фона является их простота реализации, а также скорость обновления моделей фона при переходе между кадрами в видео. Однако стоит отметить, что точность работы рассматриваемых методов в сильной степени зависит от скорости движущихся объектов. Так, объект определяется намного хуже, если скорость его передвижения невысока. Также рассмотренные методы являются довольно неэффективными при наличии в видео движущегося фона (например, движущиеся ветки и листья деревьев) и при изменении освещения в сцене. С целью сглаживания указанных эффектов возможно применение процедуры, называемой a-смешиванием. В данной процедуре обновленная модель фона текущего кадра I_k представляется выпуклой оболочкой модели фона F_{k-1}:

$$F_k(x,y) = aI_k(x,y) + (1-a)F_{k-1}(x,y) \tag{1.9}$$

1.2.2. Рекурсивные методы

Особенностью этой группы методов, предназначенных для построения фоновой модели, в отличие от рассмотренных нерекурсивных является то, что в процессе обновления фоновой модели видеокадра применяются данные об интенсивностях и цветах всех пикселей лишь текущего кадра. К группе рекурсивных методов относятся такие методы, как: метод, извлекающий визуальный фона, в зарубежной литература известен как Visual Background Extractor или ViBe [6], метод построения гистограммы [28], метод представления фоновой модели в виде смеси гауссовых распределений, известный также как Gaussian mixture model [4, 15, 18, 21], и метод «шифровальной» книги [5, 19]. Рассмотрим некоторые из этих методов:

- *Метод, основанный на извлечении визуального фона (Visual Background Extractor, ViBe).* В рассматриваемом методе для k-ого видеокадра модель фона представляется некоторым определенным набором множеств $M^k(p) = \{v_1, v_2, \dots, v_N\}$ для всех пикселей $p = (x, y)$ текущего видеокадра, где v_i – это цвет или интенсивность пикселя. С целью классификации пикселя p в цветовом пространстве изображения строится сфера $S_R(v(x))$ с радиусом R. После этого происходит процесс вычисления количества попадающих внутрь этой сферы векторов множества $M(p)$:

$$K = |S_R(v(x)) \cap M(p)| \tag{1.10}$$

При выполнении условия $K > T_{min}$, где T_{min} – это фиксированное пороговое значение, считается, что пиксель p принадлежит фону, в противном же случае, пиксель p считается принадлежащим движущемуся объекту. На первом этапе необходимо инициализировать множества $M^0(p)$ для всех пикселей p согласно формуле:

$$M^0(p) = \{v^0(y), y \in N_G(p)\}, \tag{1.11}$$

где $N_G(p)$ – это окрестность пикселя p, размер которой 3x3 (9 клеток, включая сам текущий пиксель); y выбирается случайным образом N раз. Обновление модели фона для I_k – ого кадра происходит в два шага:

1) в случае, если p считается пикселем, принадлежащим фону, то из множества $M^k(p)$ случайным образом выбирается компонента, заменяющаяся значением $v(p)$;
2) из окрестности $N_G(p)$ случайным образом выбирается один соседний пиксель, для которого будет выполняться предыдущий шаг.

- *Метод построения гистограммы*. Этот метод основан на процессе разбиения всего цветового пространства видеокадра на бины. Добавим, что для цветного изображения или в нашем случае видеокадра цветовое пространство представляет из себя трехмерный куб, а для изображения, переведенного в оттенки серого, цветовым пространством является отрезок изменения интенсивности. При применении данного метода происходит процесс построения гистограммы распределения цветов для всех пикселей видеокадра. При последующем анализе множества всех пикселей изображения в зависимости от того, какой цвет наблюдается в пикселе соответствующий бин построенной гистограммы увеличивается на единицу. Таким образом, получается, что каждый пиксель обладает некоторой гистограммой, столбцы которой показывают то, как часто пиксель принимает то или иное значение цвета. Например, у гистограммы, относящейся к пикселю, который представляет в кадре небо, наибольшую высоту будет иметь столбец, отвечающий за голубой цвет. Если значение определенного пикселя не превышает заданного фиксированного порогового значения, т.е. соответствующий текущему цвету пикселя столбец гистограммы ниже определенного

значения, то рассматриваемый пиксель считается не принадлежащим фону и определяется как принадлежащий движущемуся объекту в кадре.

- *Метод представления фоновой модели в виде смеси гауссовых распределений (Gaussian Mixture Model или GMM).* Гауссова смесь - это функция, состоящая из нескольких гауссианов, каждый из которых идентифицируется как $i \in \{1, \dots, k\}$, где k – это общее число гауссовых распределений. Каждый пиксель в видеокадре модулируется смесью из k гауссовых распределений. В этом методе при построении фоновой модели мы полагаем, что для любого пикселя видеокадра нам известна его история изменений цвета на всех предыдущих видеокадрах $\{X_1, X_2, \dots, X_D\} = \{I_i(x_0, y_0), i = \overline{1, D}\}$, где I_i – текущий кадр, (x_0, y_0) – координаты пикселя. Вероятность того, что определенный пиксель примет значение X_i в момент времени i может быть определена как:

$$P(X_i) = \sum_{j=1}^{k} \omega_j \, N(X_i \mid \theta_j), \tag{1.12}$$

где ω_j – вес j-ой компоненты гауссовой смеси, k – общее количество гауссовых распределений, $N(X_i \mid \theta_j)$ – это функция плотности нормального распределения, которая выражается формулой:

$$N(X_i \mid \theta_j) = \frac{1}{(2\pi)^{\frac{D}{2}} \left|\sum_j\right|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_i - \mu_j)^T (\sum_j)^{-1} (X_i - \mu_j)}, \tag{1.13}$$

где $\theta_j = \{\mu_j, \sum_j\}$, μ_j - математическое ожидание, $\sum_j$ – среднеквадратичное отклонение, D – размерность вектора наблюдений.

Цветовые компоненты пикселя считаются независимыми, следовательно, они обладают идентичным среднеквадратичным отклонением. В этом случае матрица ковариации будет определяться следующим способом:

$$\sum_j = \sigma_j^2 E, \tag{1.14}$$

где E – единичная матрица. Благодаря такому предположению вычислительная сложность метода существенно снижается. Причиной этого служит отсутствие необходимости в вычислении обратной матрицы к матрице ковариации $\sum_j$ в формуле (1.13). В итоге мы получаем, что для каждого пикселя видеокадра заданы распределения множества наблюдаемых значений цвета. Если пиксель примет новое значение цвета, то этот цвет будет представляться одной из основных компонент уже построенной гауссовой смеси и использоваться в дальнейшем для адаптации параметров вероятностной модели.

Построенные гауссовы распределения упорядочиваются следующим образом:

$$r_j = \frac{\omega_j}{\sigma_j} \tag{1.15}$$

Благодаря этому пикселю, принадлежащему фону видеокадра, в соответствие будет ставиться компонент гауссовой смеси с небольшой дисперсией σ_j и большим весом ω_j.

Кроме всего вышесказанного имеется еще одно не мало важное предположение, согласно которому первые B распределения используются в качестве модели фона сцены, если выполняется условие:

$$B = \arg min_b \left\{ \sum_{j=1}^{k} \omega_j > T \right\}, \tag{1.16}$$

где T – это задаваемое пороговое значение (параметр модели), определяющее минимальную долю фоновой модели.

В процессе анализа следующего видеокадра принадлежность пикселя к определенному распределению, определяется по правилу (1.17) (используется расстояние Махаланобиса [24]):

$$\sqrt{\left(X_{i+1} - \mu_j\right)^T (\sigma_j)^{-1} (X_{i+1} - \mu_j)} < 2{,}5\ \sigma_j \tag{1.17}$$

Вследствие этого возможны два случая:

1) допустим, что условие (1.17) для конкретного пикселя выполняется. Тогда в зависимости от того, относится ли гауссово распределение, к которому принадлежит рассматриваемый пиксель, в группу фоновых распределений B или нет, пиксель считается принадлежащим либо фоновой модели видеокадра, либо движущемуся объекту в кадре;
2) если условие (1.17) для рассматриваемого пикселя не выполняется ни с одним из имеющихся распределений, пиксель определяется в качестве принадлежащего движущемуся объекту.

Перед обработкой следующего видеокадра необходимо выполнить процедуру обновления всех параметров имеющихся распределений. Адаптация (обновление) параметров выполняется различным способом. Это зависит от того, нашлось ли для цвета текущего пикселя соответствующее распределение:

1) так, если условие об обнаружении соответствия выполняется, то новые параметры распределений, которым соответствует наблюдение цвета x_{i+1}, определяются по следующим формулам:

$$\omega_j^{new} = (1 - a)\omega_j + a, \tag{1.18}$$

$$\mu_j^{new} = (1 - \rho)\mu_j + \rho X_{i+1}, \tag{1.19}$$

$$(\sigma_j^{new})^2 = (1 - \rho) * \left(\sigma_j\right)^2 + \rho\left(X_{i+1} - \mu_j^{new}\right)\left(X_{i+1} - \mu_j^{new}\right)^T, \tag{1.20}$$

где a – некоторая заданная константа, $\rho = aN(X_{i+1} \mid \theta_j)$.

Параметры же всех остальных распределений, которым x_{i+1} не соответствует, не меняются. Для них лишь происходит пересчет весовых коэффициентов ω_j^{new} по формуле:

$$\omega_j^{new} = (1 - a)\omega_j \tag{1.21}$$

2) если же условие об обнаружении соответствия не выполняется, то крайняя компонента гауссовой смеси согласно (1.15) заменяется новым распределением уже с новыми параметрами:

$$\mu_j^{new} = X_{i+1}, \tag{1.22}$$

$$(\sigma_j^{new})^2 = max, \tag{1.23}$$

$$\omega_j^{new} = min, \tag{1.24}$$

т.е. в качестве математического ожидания выбирается текущее значение пикселя, дисперсия принимает максимально возможное значение, а значение веса, наоборот, минимальное.

Общее или максимальное число гауссовых распределений гауссовой смеси выбирается в зависимости от доступных вычислительных ресурсов, а также от степени сложности фона видеокадра. Для первоначальной инициализации параметров компонентов смеси может применяться либо EM-алгоритм [2, 3, 8, 9, 19, 23], либо метод k-средних.

ГЛАВА 2. КЛАСТЕРИЗАЦИЯ

2.1. Основные понятия

Кластеризация (или кластерный анализ) - процесс группировки различных экземпляров или объектов исходных данных в некоторые подмножества или группы таким образом, что идентичные или схожие между собой экземпляры группируются вместе друг с другом, а объекты, которые отличаются друг от друга, принадлежат различным группам (кластерам). Таким образом, мы получает эффективное представление множества всех элементов наших данных, которые распределены в различные группы по степени их схожести друг с другом [25, 26].

Структуру кластеризации можно представить в виде некоторого набора подмножеств или групп $G = G_1, \dots, G_m$ общего множества исходных данных D таких, что $D = \bigcup_{i=1}^{m} G_i$ и $G_i \cap G_j = \emptyset$ для $i \neq j$. Таким образом, мы получаем, что любой элемент из множества данных D входит в одну и только одну группу.

2.1.1. Меры расстояний

Так как кластеризация – это процесс разделения объектов на различные группы по степени их схожести друг с другом, то необходимо обладать каким-либо инструментом для измерения этой самой степени близости объектов, и благодаря которому мы сможем точно сказать, являются ли два объекта схожими либо разными. Существует два основных таких инструмента, используемых для оценки этого отношения: меры расстояния и меры подобия. Большинство методов кластеризации используют различные меры расстояния, с целью отыскания сходства либо различия между любой из имеющихся парой объектов. Именно поэтому рассмотрим более подробно различные меры расстояний.

Предположим, что у нас есть два p-размерных элемента из множества исходных данных $z = z_1, \dots, z_p$ и $x = x_1, \dots, x_p$ и нам необходимо узнать

степень их сходства. В таком случае определить расстояние между этими двумя объектами мы можем воспользовавшись метрикой Минковского, а именно:

$$d(z,x) = \left(|z_1 - x_1|^g + |z_2 - x_2|^g + \cdots + \left|z_p - x_p\right|^g\right)^{1/g}, \quad (2.1)$$

где $d(z,x)$ – это расстояние между элементами z и x. Если положить $g = 2$, то мы получим евклидово расстояние. Отметим, что эта мера расстояния используется чаще других. При возведении полученного евклидового расстояния в квадрат мы можем добиться такого эффекта, когда наиболее отдаленные друг от друга экземпляры обретут больший вес. При $g = 1$ мы получаем расстояние, называемое метрикой Манхэттена, для которой характерно то, что вес отдаленных друг от друга элементов уменьшается по сравнению с евклидовым расстоянием. Если устремить g в бесконечность, т.е. $g = \infty$, то мы получим метрику Чебышева. Данная мера расстояния позволяет определить степень схожести или различия между объектами по какой-либо одной координате, различие которой для объектов принимает максимальное значение.

Выбор метрики для определения расстояния между двумя объектами является отдельной задачей, так как от достаточно сильно зависят результаты процесса кластеризации, которые могут существенно различаться.

Кроме того, на результат кластерного анализа могут повлиять используемые единицы измерения. Для того, чтобы решить данную проблему необходимо произвести нормализацию всех переменных, тем самым мы придадим всем переменным одинаковый вес. Однако, если вес каждой переменной зависит от степени важности самой переменной, то в этом случае расстояние между некоторыми элементами может быть определено следующим путем:

$$d(z,x) = \left(\omega_1|z_1 - x_1|^g + \omega_2|z_2 - x_2|^g + \cdots + \omega_p\left|z_p - x_p\right|^g\right)^{1/g}, \quad (2.2)$$

где $\omega_i \in [0, \infty)$.

2.1.2. Классификация алгоритмов

Существует большое множество самых различных методов кластеризации. Основной причиной этого является то, что само понятие «кластер» определено не точно. По этой причине разработано большое количество методов кластеризации, в каждом из которых применяется свой принцип индукции. Все множество существующих методов кластеризации можно классифицировать по двум особенностям:

1. Иерархические и так называемые плоские методы. Иерархические методы кластерного анализа создают группы посредством рекурсивного деления элементов сверху вниз (здесь изначально каждый элемент является некоторым кластером. Впоследствии эти кластеры последовательно объединяются друг с другом до тех пор, пока не будет получена желаемая структура кластеров) и снизу вверх (здесь, наоборот, изначально все имеющиеся элементы входят в одну общую группу. Впоследствии эта группа делится на более мелкие группы, которые в свою очередь также последовательно делятся на еще более мелкие группы. Так продолжается, пока не будет получена желаемая структура групп). Таким образом на выходе мы имеем некую дендрограмму, представляющую из себя вложенные кластеры элементов и степени схожести, по которым отличаются эти кластеры. В отличие от иерархических методов так называемые плоские методы строят всего лишь одно единственное разбиение элементов на подмножества.
2. Метод мягкой и жесткой кластеризации. Жесткий метод кластерного анализа определяет каждый элемент исходных данных в какое-либо одно определенное подмножество, т.е. получается, что каждый отдельный экземпляр принадлежит одному и только одному кластеру. Мягкий метод кластерного

анализа в свою очередь каждому элементу исходных данных ставит в соответствие некоторый набор вещественных значений, показывающих степень вхождения конкретного элемента в различные группы. Таким образом получаем, что каждый экземпляр определен в каждое подмножество с различной вероятностью.

2.2. EM–алгоритм

Алгоритм ожидание-максимизация (expectation-maximization, EM) - это итерационный метод, позволяющий определить оценку максимального правдоподобия параметра θ некоторого параметрического распределения вероятностей [2, 3, 8, 9, 19, 23, 26].

Для лучшего понимания принципа работы данного алгоритма рассмотрим пример. Допустим, мы будем измерять температуру за окном каждый час в течение суток, и представим полученные данные следующим образом: $x \in R^{24}$. Отметим, что измеренная нами температура зависит от времени года (сезона): $\theta \in \{\text{лето}, \text{осень}, \text{зима}, \text{весна}\}$. Предположим, что мы знаем распределение температуры в зависимости сезона (времени года): $p(x \mid \theta)$. Но что, если у нас будут данные лишь о средней температуре за какой-то определенный день: $y = \bar{x}$, и мы захотим определить (оценить), какое сейчас время года θ (например, сейчас лето или зима?). В таком случае, мы можем попытаться отыскать оценку максимального правдоподобия θ, то есть значение $\hat{\theta}$, которое бы максимизировала условную вероятность $p(y \mid \theta)$. В том случае, если проблема отыскания максимального правдоподобия не является тривиальной, мы можем обратиться к EM-алгоритму. Данный алгоритм итеративно чередуется между предположением о полных данных x (мы можем не обладать этими данными) и поиском θ, максимизирующим $p(x \mid \theta)$ при условии θ. Таким образом EM-алгоритм пытается найти оценку максимального правдоподобия θ для y.

Для того, чтобы применять EM-алгоритм нам необходимо обладать некоторыми данными y, которые мы наблюдаем, параметрическую плотность распределения вероятности $p(y \mid \theta)$, описание полных данных x и параметрическую плотность распределения вероятности $p(x \mid \theta)$. Предполагается, что полные данные могут быть смоделированы в виде непрерывного случайного вектора X с плотностью распределения вероятности $p(x \mid \theta)$, где $\theta \in \Omega$ для некоторого множества Ω. Отметим, что мы не наблюдаем непосредственно вектор X, вместо этого мы наблюдаем реализацию некоторого случайного вектора Y, который зависит от X. Так, например, X может быть случайным вектором, а Y - средним его компонент и тому подобное.

Учитывая то, что мы обладаем только y, наша цель - найти оценку максимального правдоподобия для θ:

$$\hat{\theta} = \arg\max_{\theta \in \Omega} p(y \mid \theta) \tag{2.1}$$

Часто на практике легче вычислить θ, которое максимизирует логарифмическую вероятность y:

$$\hat{\theta} = \arg\max_{\theta \in \Omega} \log p(y \mid \theta) \tag{2.2}$$

Так как функция $\log$ является монотонно возрастающей, то решение (2.1) будет абсолютно таким же, как и решение (2.2). В случае, если решение (2.1) и (2.2) затруднено, применяется EM-алгоритм: сперва делается предположение о данных X и определяется θ, которое бы максимизировало (ожидаемую) логарифмическую вероятность X. После того, как мы получим оценку для θ, мы можем сделать новое предположение по поводу данных X и повторить этот шаг еще раз.

EM - алгоритм традиционно описывается в двух шагах (E-шаг и M-шаг), но для лучшего понимания разобьем его на пять шагов:

- Шаг 1. Пусть m = 0 и сделаем начальную оценку $\theta^{(m)}$ для θ.

- Шаг 2. Учитывая наблюдаемые данные y и предполагая, что наша текущая догадка $\theta^{(m)}$ верна, сформулируем условное распределение вероятности $p(x \mid y, \theta^{(m)})$ для полных данных x.
- Шаг 3. Используя условное распределение вероятностей $p(x \mid y, \theta^{(m)})$, рассчитанное на шаге 2, сформируем условное ожидаемое логарифмическое правдоподобие, которое называется Q-функцией [26]:

$$Q(\theta \mid \theta^{(m)}) = \int_{X(y)} log\, p(x \mid \theta)\, p(x \mid y, \theta^{(m)}) dx \qquad (2.3)$$
$$= E_{X|y,\theta^{(m)}}[\log p(X \mid \theta)],$$

где интеграл по множеству X является замыканием множества $\{x \mid p(x \mid y, \theta) > 0\}$, также мы предполагаем, что $X(y)$ не зависит от θ.

Обратим внимание, что θ является свободной переменной в формуле (2.3), поэтому Q-функция является функцией от θ, помимо этого она также зависит от нашего текущего предположения $\theta^{(m)}$ через $p(x \mid y, \theta^{(m)})$, вычисленное в шаге 2.

- Шаг 4. Найдем такое θ, которое бы максимизировало Q-функцию (1.3). В результате мы получим новую оценку $\theta^{(m+1)}$.
- Шаг. 5. Задаем m = m + 1 и возвращаемся к шагу 2 (EM-алгоритм не определяет критерий остановки; стандартными критериями для алгоритма являются итерации до тех пор, пока оценка не перестанет меняться, т.е. $\|\theta^{(m+1)} - \theta^{(m)}\| < \varepsilon$ для $\varepsilon > 0$, либо пока не перестанет изменяться логарифмическая вероятность $l(\theta) = \log p(y \mid \theta)$: $\|l(\theta^{(m+1)}) - l(\theta^{(m)})\| < \varepsilon$ для $\varepsilon > 0$).

Гарантируется, что оценка EM-алгоритма никогда не ухудшится. Рассматриваемый алгоритм находит пик в вероятности $p(y \mid \theta)$, но если функция правдоподобия $p(y \mid \theta)$ имеет несколько пиков, то в этом случае нет никаких гарантий, что EM-алгоритм найдет глобальный максимум

вероятности. Для реализации EM–алгоритма на практике часто начинают с нескольких случайных начальных догадок и в качестве окончательного предположения для θ выбирают тот, который имеет наибольшую вероятность.

Как упоминалось ранее, традиционное описание EM - алгоритма состоит лишь из двух шагов. Первый шаг, называемый E-шагом, объединяет шаги 2 и 3, рассмотренные выше. Второй шаг, называемый M-шагом, содержит в себе 4-й шаг:

- E – шаг. Вычислим условное ожидание $Q\left(\theta\middle|\theta^{(m)}\right)$, приведенное в (2.3), учитывая оценку предыдущей итерации $\theta^{(m)}$.
- M – шаг. Следующее предположение (m+1) для θ:

$$\theta^{(m+1)} = \arg\max_{\theta\in\Omega} Q\left(\theta\middle|\theta^{(m)}\right) \quad (2.4)$$

Так как E-шаг предназначен лишь для вычисления Q-функции, которая впоследствии используется в M-шаге, то EM-алгоритм можно определить как итеративное решение M-шага, заданного формулой (2.4). Лучший способ, чтобы рассматривать EM–алгоритм, это применять его к конкретной проблеме, так как в таком случае нет необходимости тратить время на вычисление частей Q-функции, независящих от θ.

ГЛАВА 3. ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ

По мере внедрения в нашу жизнь все большего количества различных цифровых систем связи неуклонно растет актуальность решения задачи восстановления и улучшения качества изображений при помощи различных методов фильтрации [27], так как изображения в процессе различных преобразований, а также передачи подвергаются воздействию помех, что часто приводит к потере некоторых участков изображений, а также к ухудшению их качества.

Основной задачей фильтрации является улучшение качества изображения. Одними из наиболее распространенными на сегодня видами шумов являются импульсный и Гауссовский шумы. Источниками шумов могут выступить:

- оборудование (например, шумы, вносимые камерой);
- условия съемки (дождь, ночь);
- различные помехи, возникающие при передаче по аналоговым каналам;
- также различные алгоритмы, обрабатывающие изображения (например, алгоритмы бинаризации изображения могут вносить шумы, проявляющие себя в виде как локального скопления точек, так и их скопления по всему изображению, либо же в виде отдельных точек на изображении)

В алгоритмах обнаружения движущихся объектов шумы на изображении могут быть причиной того, что, например, будут проигнорированы движущиеся объекты в кадре либо, наоборот, статические объекты будут приняты в качестве движущихся. С целью предотвращения и сведения к минимуму подобных ложных обнаружений необходимо применять фильтрацию. Рассмотрим некоторые существующие методы фильтрации изображений.

3.1. Морфологическая фильтрация

К морфологическим фильтрам относятся операции эрозии и дилатации:

- *Дилатацией* называется процесс свертки исходного изображения с некоторым ядром, имеющим произвольный размер и форму. При применении данного метода ядро проходит по всему изображению, применяя к значениям всех пикселей изображения, накрываемых шаблоном, некоторый оператор поиска локального максимума. Результатом работы этого алгоритма является увеличение на изображении белых областей. Дилатацию бинарного изображения (I) и ядра (Y) можно задать формулой:

$$I \oplus Y = \bigcup_{b \in Y} I_b \tag{3.1}$$

Рассмотрим пример работы данного метода:

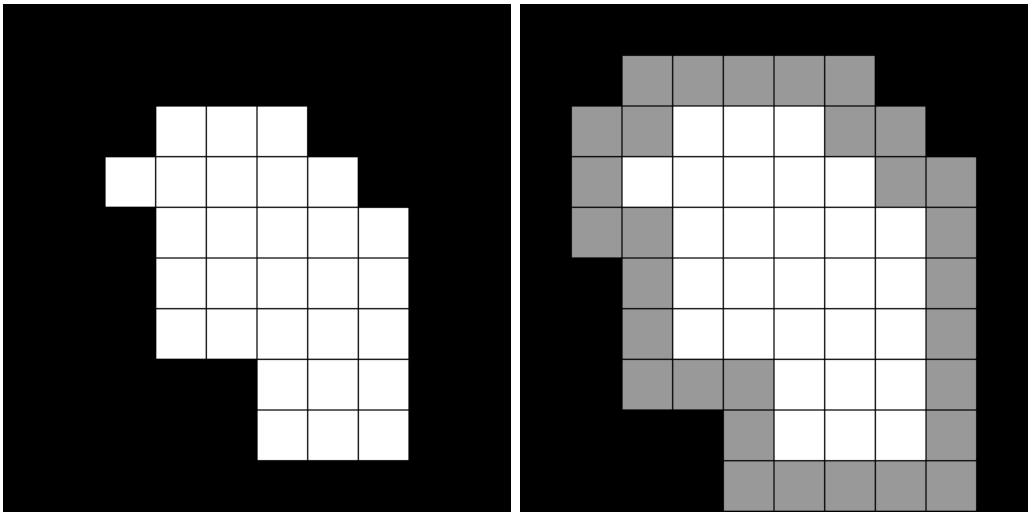

Рис. 3.1. Результат работы дилатации. Серым цветом отмечены все изменения на изображении после применения данного метода

- *Эрозия* отличается от дилатации лишь тем, что при прохождении ядром по всему изображению к значениям пикселей изображения, накрываемых шаблоном, применяется некоторый оператор поиска локального минимума. Результатом работы этого алгоритма является уменьшение на изображении белых

областей. Эрозию бинарного изображения (I) и ядра (Y) можно задать формулой:

$$I \ominus Y = \{ z \in I \mid Y_z \subseteq I \} \quad (3.2)$$

Рассмотрим пример работы данного метода:

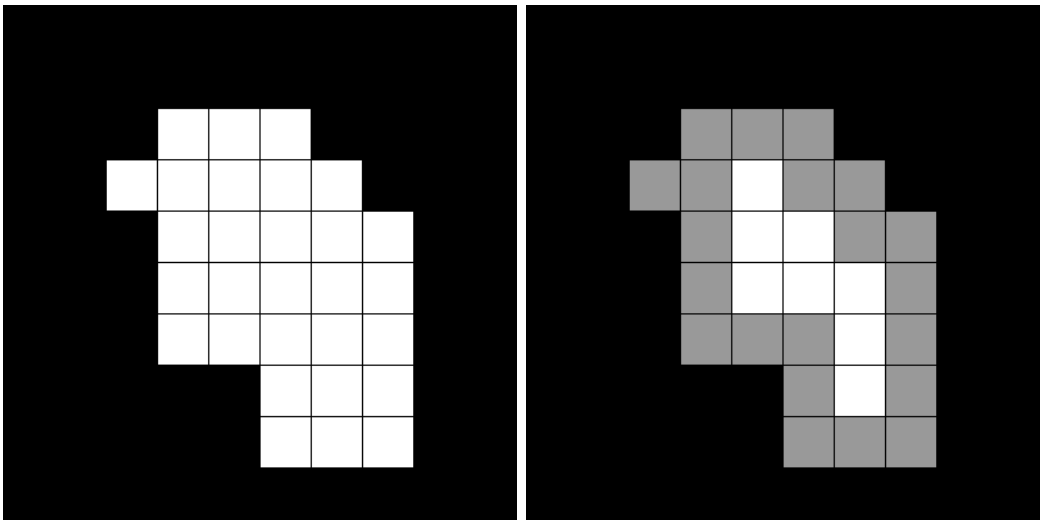

Рис. 3.2. Результат работы эрозии. Серым цветом отмечены все изменения на изображении после применения данного метода

Комбинирование рассмотренных базовых морфологических операций порождает производные морфологические операции:

- *Размыканием* называется процесс применения дилатации к эрозии исходного изображения. При применении эрозии на первом этапе удаляются мелкие шумы на изображении и уменьшаются размеры основных объектов. Последующее применение дилатации восстанавливает размеры объектов. Подобный метод применяется для удаления шумов и мелких деталей и задается формулой:

$$open(I, elem) = dilate(erode(I, elem) \quad (3.3)$$

- *Замыканием* называется процесс применения эрозии к дилатации исходного изображения. При применении дилатации на первом этапе удаляются разрывы внутри объектов на изображении и увеличиваются их размеры. Последующее применение эрозии

восстанавливает размеры объектов. Подобный метод применяется для устранения черных участков внутри объектов и задается формулой:

$$close(I, elem) = erode(dilate(I, elem)) \quad (3.4)$$

- *Морфологическим градиентом* называется разница между дилатацией и эрозией исходного изображения, размер ядра которых одинаков. Применение данного метода позволяет эффективно обнаруживать контуры объектов. Морфологический градиент задается формулой:

$$grad(I, elem) = dilate(I, elem) - erode(I, elem) \quad (3.5)$$

- *Верхом шляпы* называется разница между исходным изображением и его размыканием. Применение данного метода позволяет обнаруживать наиболее яркие участки в кадре. Задается формулой:

$$tophat(I, elem) = I - open(I, elem) \quad (3.6)$$

- *Черной шляпой* называется разница между замыканием исходного изображения и самим исходным изображением. Применение данного метода позволяет обнаруживать темные участки в кадре. Задается формулой:

$$blackhat(I, elem) = close(I, elem) - I \quad (3.7)$$

3.2. Фильтр Гаусса

Данный метод фильтрации для вычисления элементов матрицы свертки использует нормальное распределение:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (3.8)$$

где x и y – это расстояния до центрального элемента в матрице. Пример работы данного метода фильтрации:

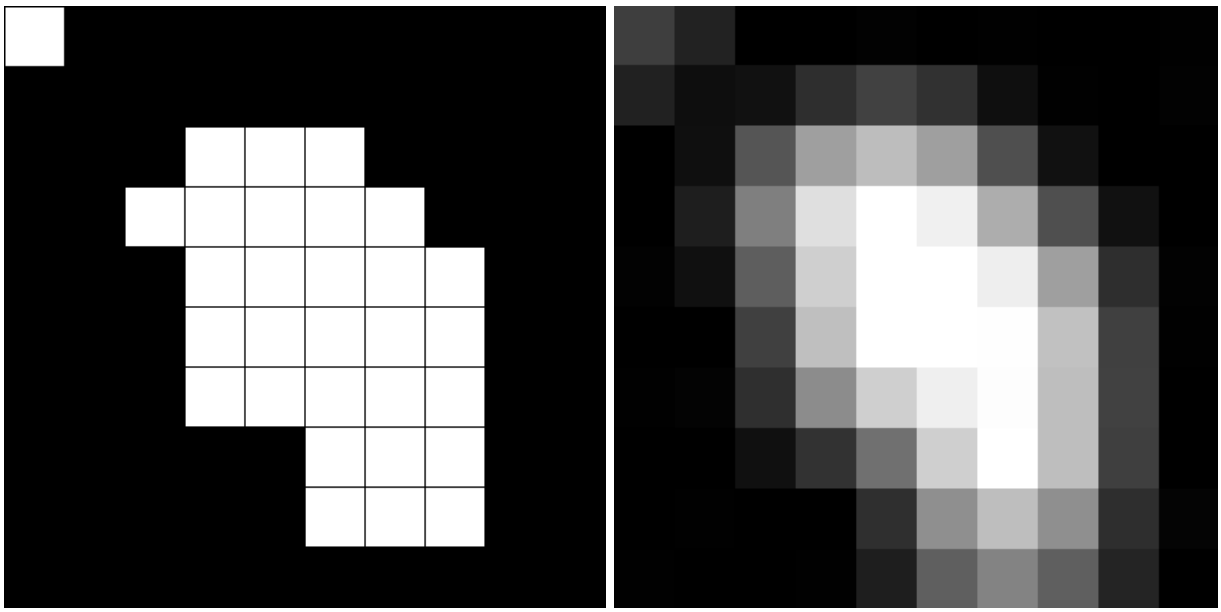

Рис. 3.3. Результат работы Гауссовской фильтрации

3.3. Медианный фильтр

Суть данного метода фильтрации изображения состоит в том, чтобы начальное значение пикселя заменить на медиану значений в некоторой окрестности рассматриваемого пикселя. Рассматриваемый алгоритм фильтрации достаточно хорошо справляется с удалением различного рода мелких рассредоточенных вкраплений, но пропускает немного более крупные области помех. Медианный фильтр представляет из себя скользящие по полю кадра окно определенного размера (например, 5x5 или 3x3). Этот метод сортирует входящие в окно пиксели в возрастающем порядке и выдает на выходе медиану (средний результат) значений этих пикселей для центрального пикселя. Пример работы медианного фильтра:

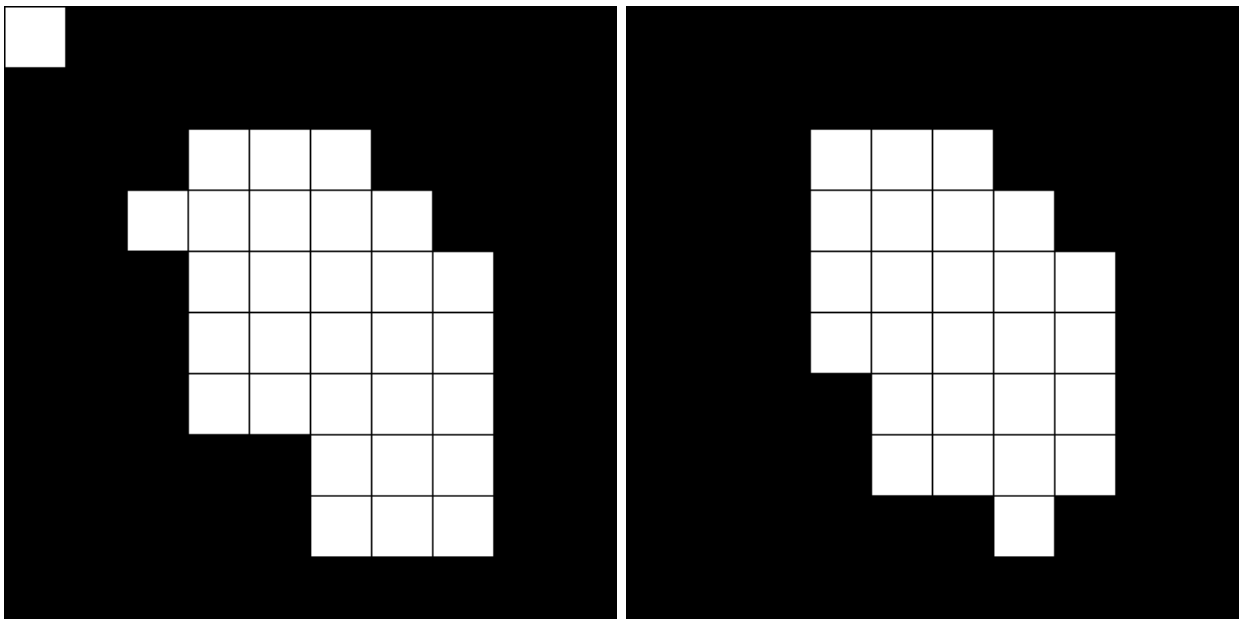

Рис. 3.4. Результат работы медианной фильтрации

3.4. Оконная фильтрация

Данный алгоритм фильтрации изображения основан на геометрии распределения точек шума на исходном изображении. Суть работы оконной фильтрации состоит в том, что окно определенного размера пробегает по всему исходному изображению и в случае, если процент содержания белых точек (для бинарного изображения) в этом окне меньше определенного заданного порога, алгоритм удаляет их с изображения. Блок–схема работы оконной фильтрации представлена ниже:

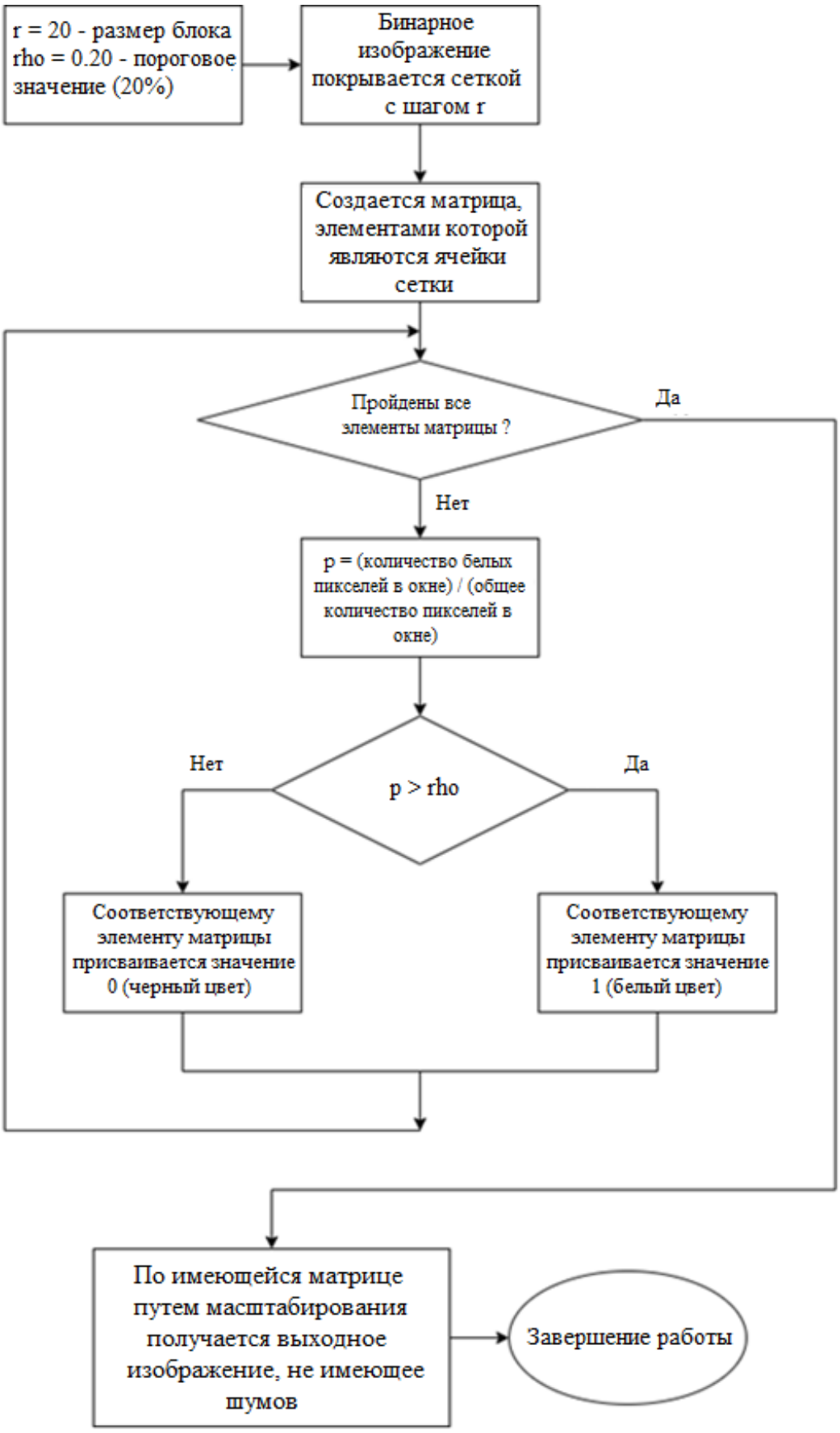

Рис. 3.5. Блок-схема работы оконной фильтрации

ГЛАВА 4. РЕАЛИЗАЦИЯ АЛГОРИТМА MOG V2 (MIXTURE OF GAUSSIAN)

4.1 Описание алгоритма

С целью решения задачи определения движущихся объектов с данных камер беспилотного автомобиля в данной работе мной был выбран алгоритм MOG v2 (Mixture of Gaussian). Этот алгоритм основан на методе вычитания фона и использует метод смеси гауссовых распределений [1, 4, 15, 18, 21, 22]. Однако отметим, что в выбранном алгоритме MOG v2 адаптация или же обновление параметров вероятностной модели происходит при помощи итерационного EM - алгоритма.

В MOG v2 цвета каждого отдельного пикселя модулируются посредством k гауссовых распределения, причем, необходимо отметить, что количество распределений k для каждого пикселя динамически адаптируется и может быть любым числом не большим чем k. Предполагается, что каждое из имеющихся гауссовых распределения представляет различный цвет фона и переднего плана изображения. Так, например, одно распределение может модулировать голубой цвет неба, другое серый цвет дороги или красный цвет припаркованного автомобиля.

При загрузке видеоряда в первую очередь для всех пикселей в кадре определяются компоненты гауссовой смеси наблюдаемых значений цвета, то есть, по другому говоря, определяется среднее значение или математическое ожидание и отклонение для каждого из k распределений. При появлении новых значений цвета они будут определяться одной из k основных компонент гауссовой смеси. После первые B распределения используются в качестве модели фона сцены при выполнении условия (1.16). В процессе анализа следующего видеокадра принадлежность пикселя к определенному распределению или компоненту гауссовой смеси, определяется по правилу (1.17), согласно которому принадлежность пикселя фону определяется в случае, если его значение находится в пределах 2,5 стандартного отклонения

распределения, модулирующего фон в кадре. В противном случае пиксель считается принадлежащим движущемуся объекту. Если выполняется условие наличия соответствующего распределения Гаусса для текущего цвета пикселя, то параметры распределений обновляются при помощи EM–алгоритма, в противном же случае крайняя (согласно введенному порядку (1.15) компонента гауссовой смеси заменяется новым распределением с новыми параметрами согласно (1.22-1.24).

Описанный принцип работы алгоритма продемонстрирован с помощью блок-схемы на рис. 4.1:

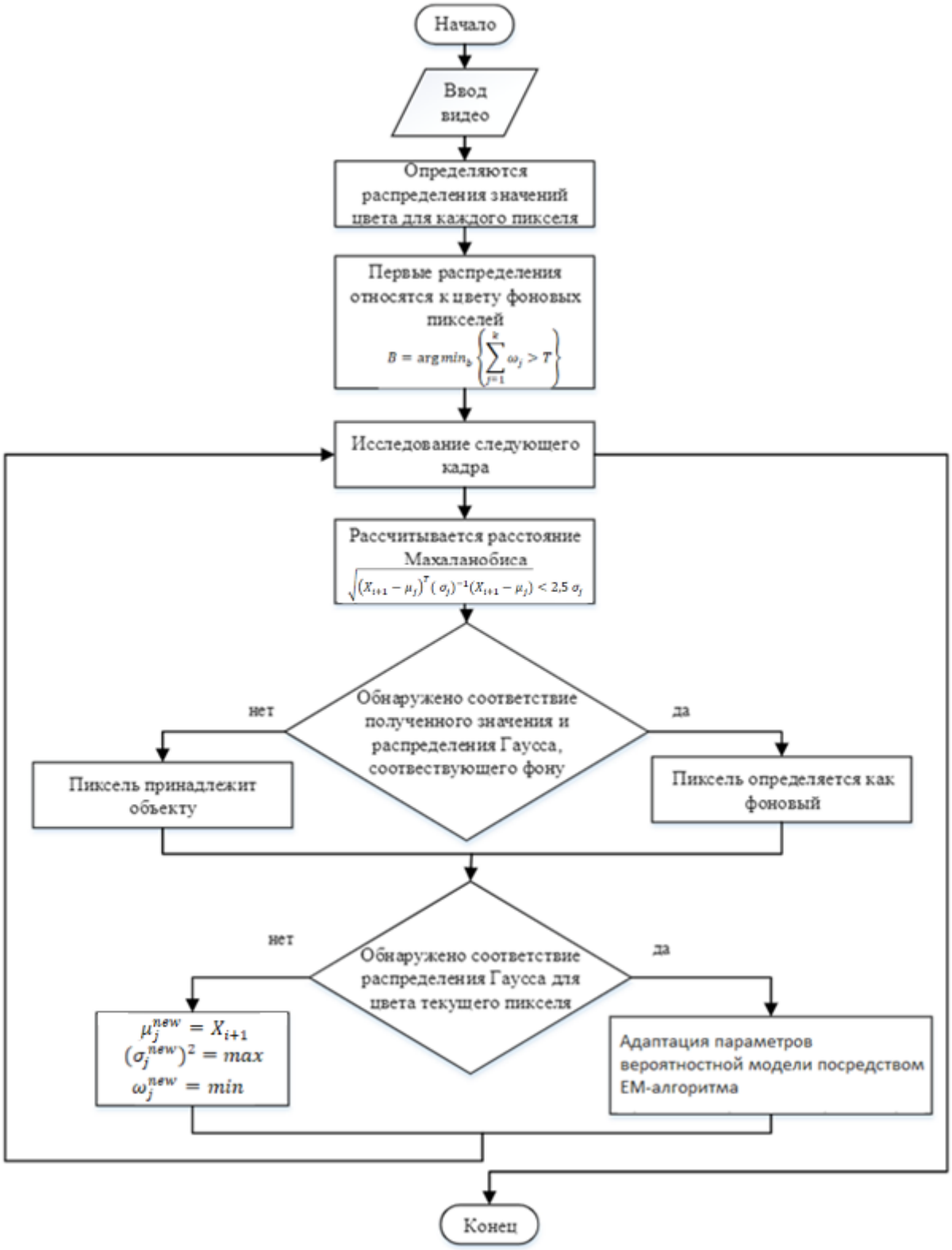

Рис. 4.1. Блок-схема работы алгоритма MOG v2 (Mixture of Gaussian), использующего смесь Гауссовых распределений и основанного на методе вычитания фона

4.1.1 Адаптация параметров с помощью EM-алгоритма

С целью установления параметров модели алгоритма MOG v2 применяется EM-алгоритм, в качестве средства кластерного анализа. Применение EM-алгоритма обусловлено тем, что в рассматриваемой модели принадлежность пикселя движущемуся объекту либо фону, т.е. вероятность принадлежности пикселя k-ому распределению является случайным, как и само наблюдение X_i. Также стоит отметить, что задача установления параметров модели не относится к классу байесовских, так как, несмотря на то, что наблюдение X_i нам известно, неизвестным для нас остается условная вероятность наблюдения X_i при условии, что пиксель принадлежит k-ому распределению $P(X_i \mid k)$, X_i и k являются случайными величинами, а априорная вероятность $P(k)$ хоть и существует, но также не известна. Таким образом получается, что задача нахождения параметров модели алгоритма MOG v2 принадлежит к классу задач кластеризации.

Как говорилось ранее, нам известно $\{X_1, X_2, \dots, X_D\} = \{I_i(x_0, y_0), i = \overline{1, D}\}$, определим вектор $S = \{x_1, x_2, \dots, x_N\}$, где x_i – это D – размерный вектор наблюдений $\{X_1, X_2, \dots, X_D\}$ каждого пикселя. Уточним формулу (1.12) для $p(x)$ в виде:

$$P(x \mid \Theta) = \sum_{j=1}^{k} \omega_j \, p_j(x \mid z_j, \theta_j), \tag{4.1}$$

где:

- $p_j(x \mid z_j, \theta_j)$ – это функция плотности нормального распределения, определяемая формулой (1.13), с параметрами $\theta_j = \{\mu_j, \sum_j\}$, $1 \le j \le k$.
- $z = (z_1, \dots, z_k)$ – это k-мерный бинарный вектор индикаторных переменных, где $z_i \in \{0,1\}$. z – это k-мерная случайная величина, представляющая идентичность компонента смеси, который сгенерировал x.

- $\omega_j = p(z_j)$ – это вес j-ого распределения, представляющее вероятность того, что случайно выбранный x был сгенерирован компонентом k, где $\sum_{j=1}^{k} \omega_j = 1$.

Полный набор параметров для модели смеси с k компонентами:

$$\Theta = \{\omega_1, \dots, \omega_k, \theta_1, \dots, \theta_k\}$$

Мы можем вычислить вес, вносимый каждой x_i в распределение j, определенной параметрами Θ:

$$a_{ij} = p\big(z_{ij} = 1 \mid x_i, \Theta\big) = \frac{p_j(x_i \mid z_j, \theta_j) * \omega_j}{\sum_{m=1}^{k} p_m(x_i \mid z_m, \theta_m) * \omega_m}, \quad (4.2)$$

$$1 \le j \le k, 1 \le i \le N$$

Благодаря EM-алгоритму обеспечивается итеративное нахождение оценок максимального правдоподобия параметров нашей вероятностной модели, начиная с некоторой начальной оценки Θ (возможно случайной), заканчивая итеративным обновлением Θ до тех пор, пока не будет обнаружена сходимость. Каждая итерация рассматриваемого алгоритма состоит из двух основных шагов:

1) На E-шаге вычисляется оценка ожидания полной условной вероятности данных с учетом наблюдаемых значений и текущих параметров. Также для всех точек данных x_i и всех распределений k в формуле (4.2) вычисляется a_{ij}, $1 \le i \le N$, $1 \le j \le k$. Отметим, что для каждой x_i справедливо $\sum_{j=1}^{k} a_{ij} = 1$. В результате мы получаем матрицу весов N x k, где сумма всех компонентов в строке равна 1.

2) На M-шаге определяются параметры, которые максимизируют ожидаемую вероятность на E-шаге, т.е. решается задача максимизации правдоподобия и находится следующее приближение векторов параметров вероятностной модели. Вычисленные ранее веса используются для нахождения новых значений параметров. Определим $N_j = \sum_{i=1}^{N} a_{ij}$ (сумма весов всех x_i для j-ого распределения). В этом случае имеем:

$$\omega_j^{new} = \frac{N_j}{N}, 1 \le j \le k \tag{4.3}$$

$$\mu_j^{new} = \left(\frac{1}{N_j}\right)\sum_{i=1}^{N} a_{ij} * x_i, 1 \le j \le k \tag{4.4}$$

$$\delta_j^{new} = \left(\frac{1}{N_j}\right)\sum_{i=1}^{N} a_{ij} * \left(x_i - \mu_j^{new}\right)\left(x_i - \mu_j^{new}\right)^T, 1 \le j \le k \tag{4.5}$$

4.2 Исходные данные

Рассматриваемый нами алгоритм принимает на вход кадры исходного видеофайла в формате RGB, отделяет движущийся объект от фона и на выходе возвращает бинарные изображения переднего плана и заднего фона кадра, в которых пиксели, относящиеся к движущимся объектам, отмечаются белым цветом, черным же цветом отмечаются оставшиеся пиксели, принадлежащие фону.

4.3 Примеры работы алгоритма

Тестирование работы алгоритма проводилось на видеозаписях с разрешением 1920x1080, снятых автором, а также на видеозаписях из набора данных kitty с разрешением 1224x370. Примеры работы алгоритма обнаружения движущихся объектов представлены на рисунках 4.2-4.5.

Работа алгоритма обеспечивается множеством параметров одними из которых являются: максимально возможное число гауссовых распределений, которое в рамках данного тестирования принимает значение равное 3, а также пороговое значение, благодаря которому определяется принадлежность каждого пикселя либо фону, либо множеству движущихся объектов, которое в рамках данного тестирования принимает значение равное 0,95. Как можно увидеть из примеров в результате применения алгоритма MOG v2 белым цветом выделяются все движущиеся объекты в

видеокадре, а черным цветом выделяется фон. Также можно заметить наличие ложных срабатываний, т.е. шумов, обусловленных движением веток и листьев деревьев, на итоговом изображении, а также наличие небольших разрывов объектов. Одной из причин наличия разрывов является то, что некоторые участки (области) движущихся объектов могут обладать цветом, схожим с цветом фона.

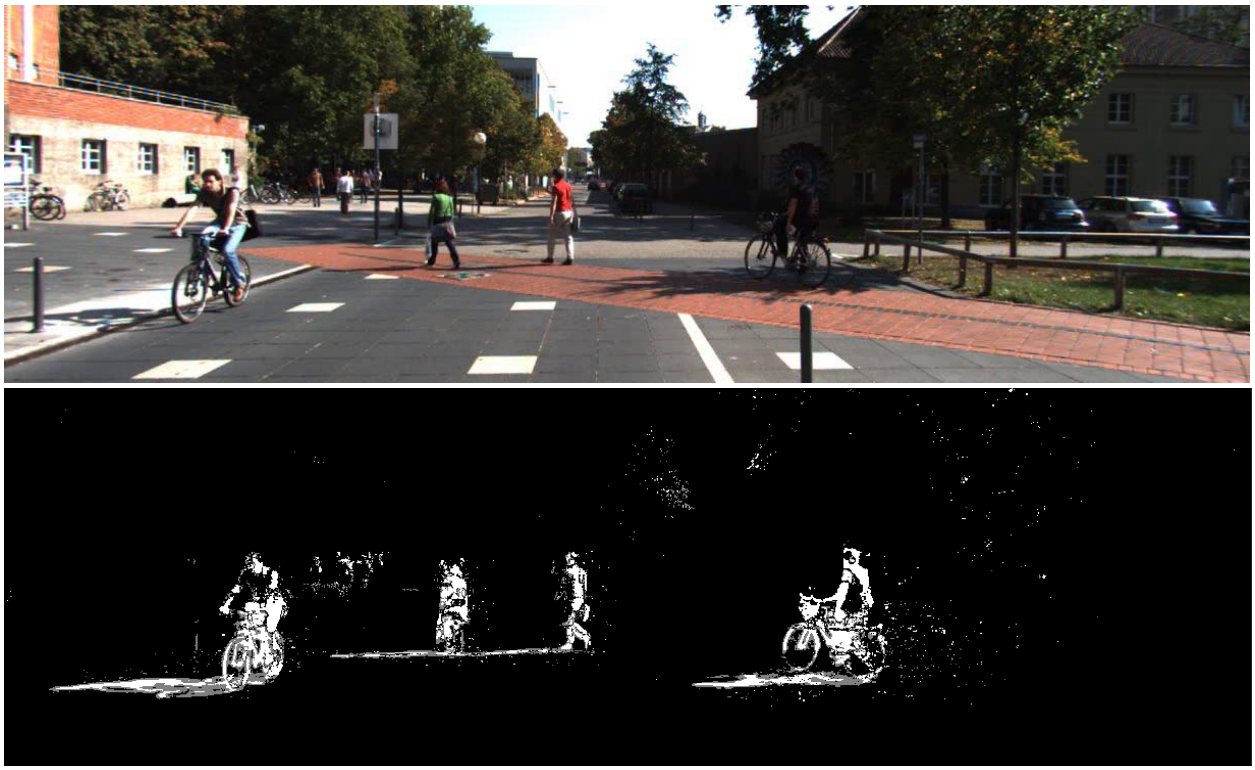

Рис. 4.2. Пример обнаружения движущихся объектов алгоритмом MOG v2 на видеозаписях из набора данных kitty. Здесь мы можем увидеть, что алгоритм довольно хорошо справился с поставленной задачей и обнаружил всех движущихся людей, несмотря на наличие небольших шумов, обусловленных в том числе движением листьев на деревьях, от которых можно избавиться при использовании одного из методов фильтрации изображения

Рис. 4.3. Пример обнаружения движущихся объектов алгоритмом MOG v2 на видеозаписях из набора данных kitty. Здесь как в предыдущем, так и в последующих примерах мы можем заметить, что рассматриваемый нами алгоритм обнаруживает тени от движущихся объектов. Подобная проблема решается путем модификации существующего алгоритма

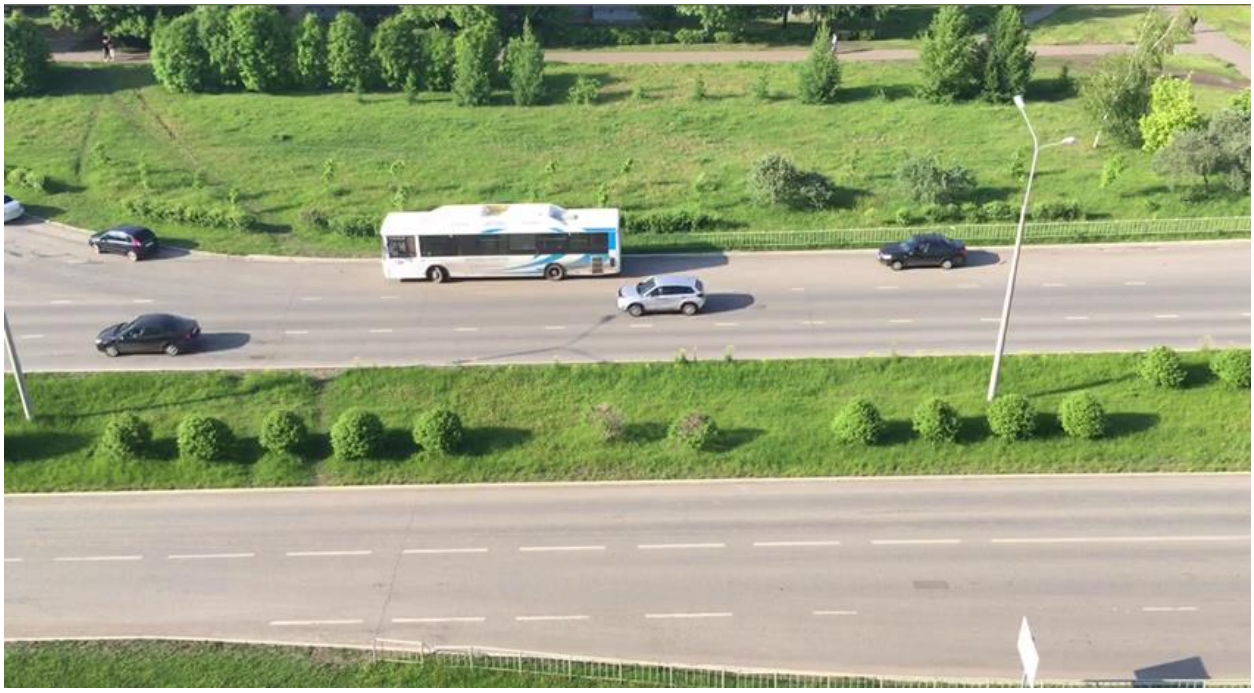

Рис. 4.4. Пример обнаружения движущихся объектов алгоритмом MOG v2. Здесь можно увидеть, что в условиях хорошей освещенности алгоритм без каких-либо затруднений обнаружил два движущихся автомобиля и двух движущихся в дали людей

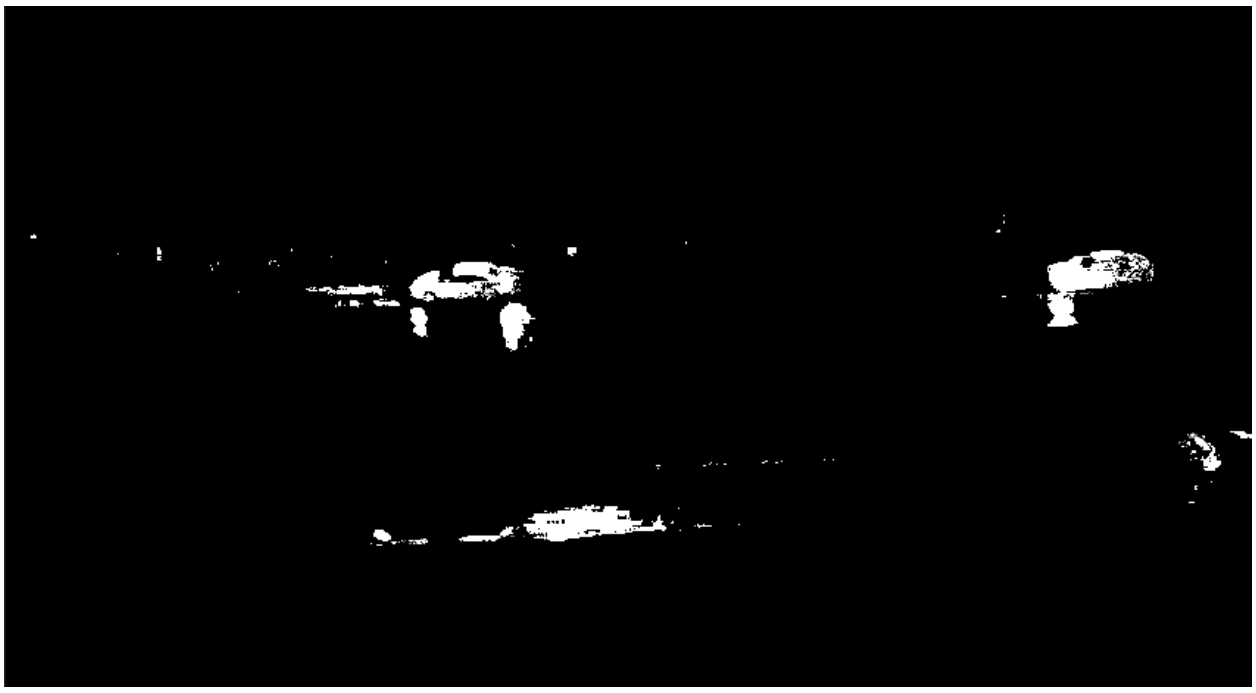

Рис. 4.5. Пример работы алгоритма MOG v2. В данном случае видно, что в условиях недостаточной освещенности алгоритм не обнаружил движущийся автомобиль, имеющий темную окраску, так как он сливается с дорогой, но обнаружил движущееся световое пятно на дороге от его фар. Отметим, что человек в такой ситуации также определяет факт наличия движения у темного автомобиля по движению световых пятен и светящихся фар

4.4 Применение фильтрации

Как было сказано ранее результаты работы алгоритма содержат большое количество шумов и нуждаются в дополнительной фильтрации. Ниже представлены примеры содержания большого количества шумов:

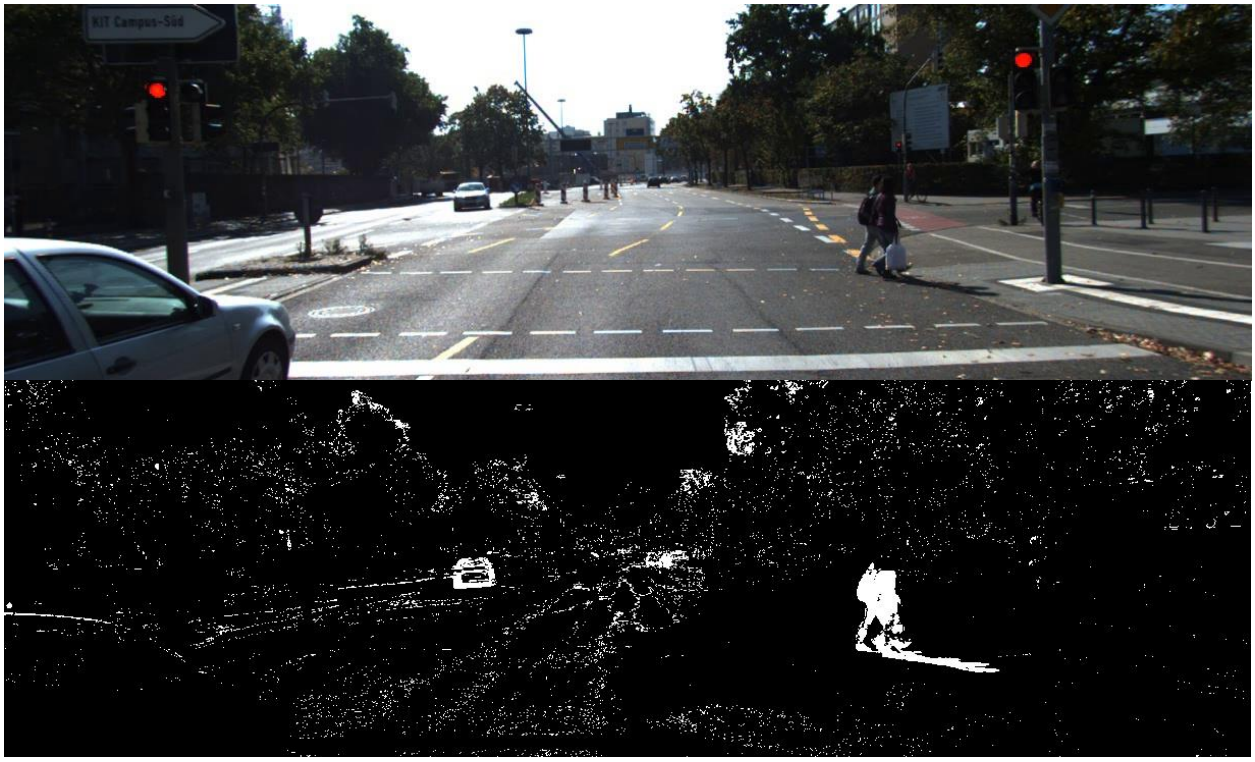

Рис. 4.6. Пример наличия шумов на изображениях в результате работы алгоритма MOG v2 на видеозаписи из набора данных kitty

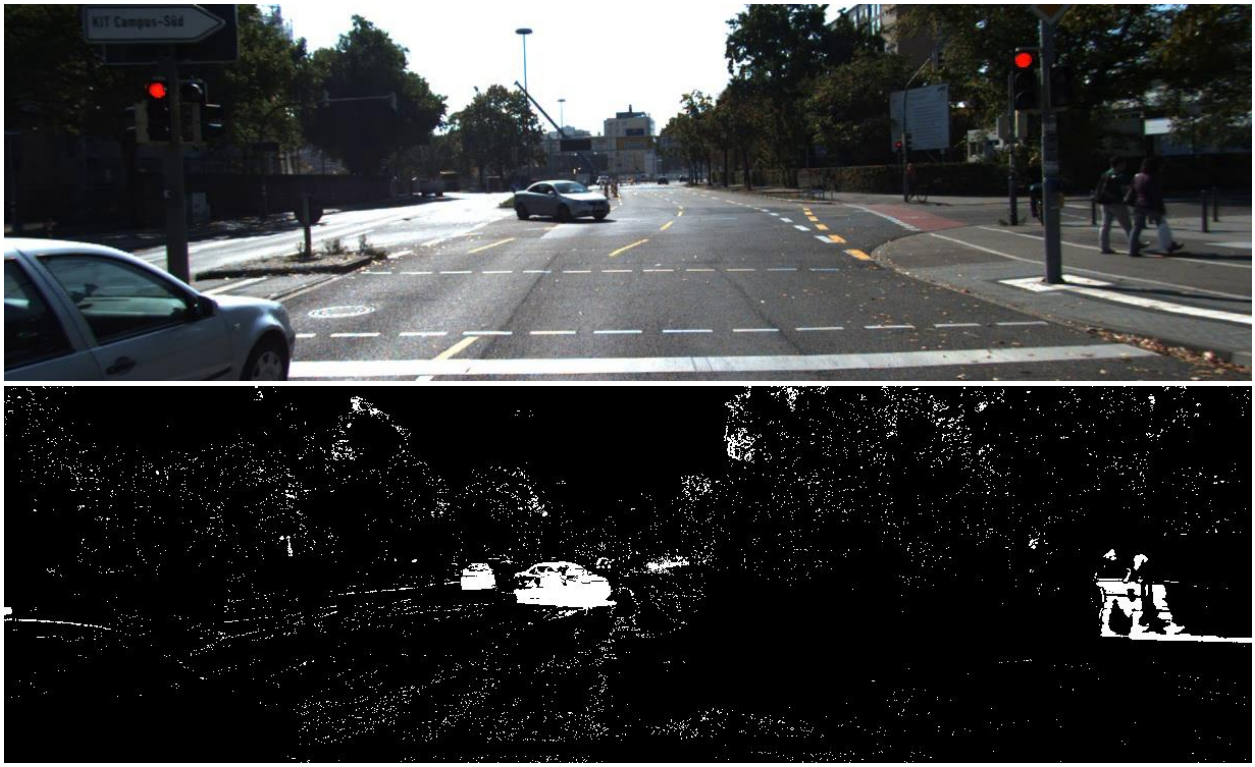

Рис. 4.7. Пример наличия шумов на изображениях в результате работы алгоритма MOG v2 на видеозаписи из набора данных kitty.

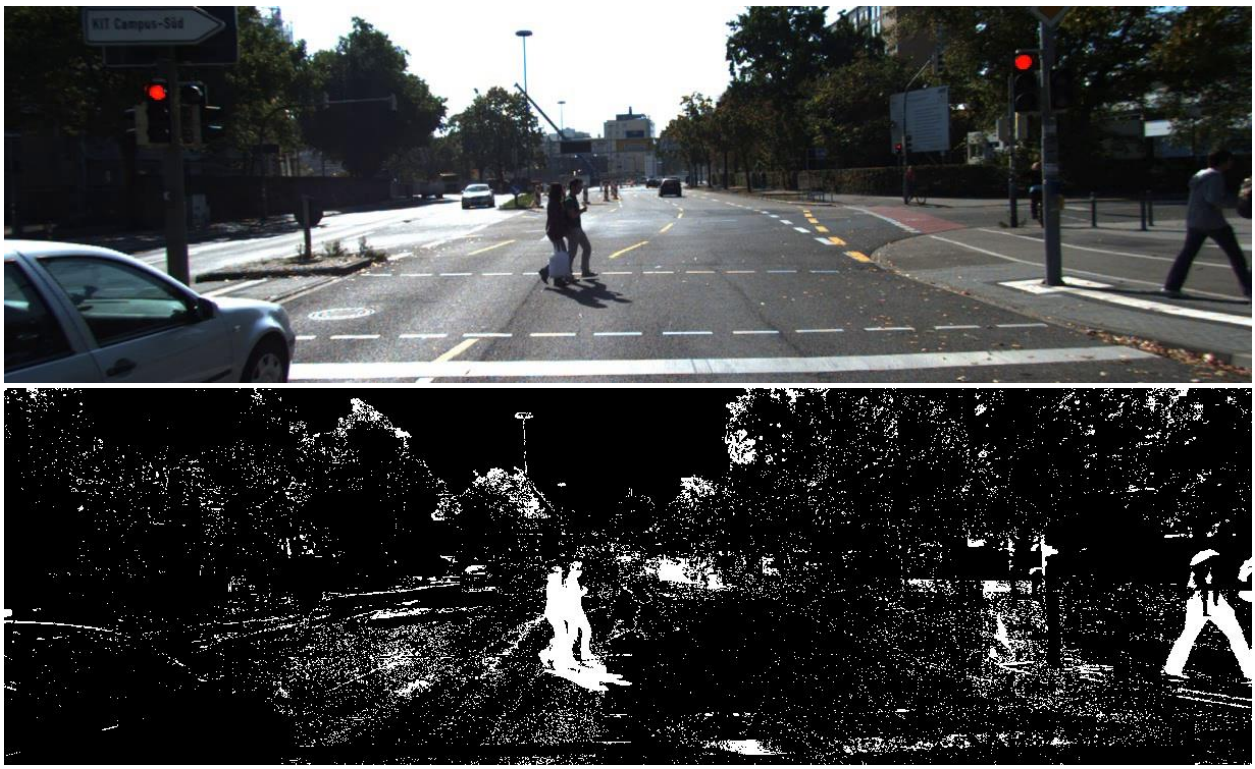

Рис. 4.8. Пример наличия шумов на изображениях в результате работы алгоритма MOG v2 на видеозаписи из набора данных kitty

Для решения этой проблемы был выбран метод оконной фильтрации. Выбор данного метода обусловлен простотой реализации, высокой скоростью обработки одного видеокадра и достаточной эффективностью удаления шумов и выделения основных объектов. Оконная фильтрация в зависимости от параметра плотности окна помимо того, что уменьшает количество шумов на изображении, также достаточно неплохо справляется с задачей устранения разрывов объектов, т.е. с устранением черных областей (пикселей) внутри движущегося объекта.

Недостатком данного метода фильтрации изображения является то, что в зависимости от размеров окна возможно сильное искажение формы обнаруженных движущихся объектов.

На рис. 4.9-4.12 демонстрируется применение метода оконной фильтрации:

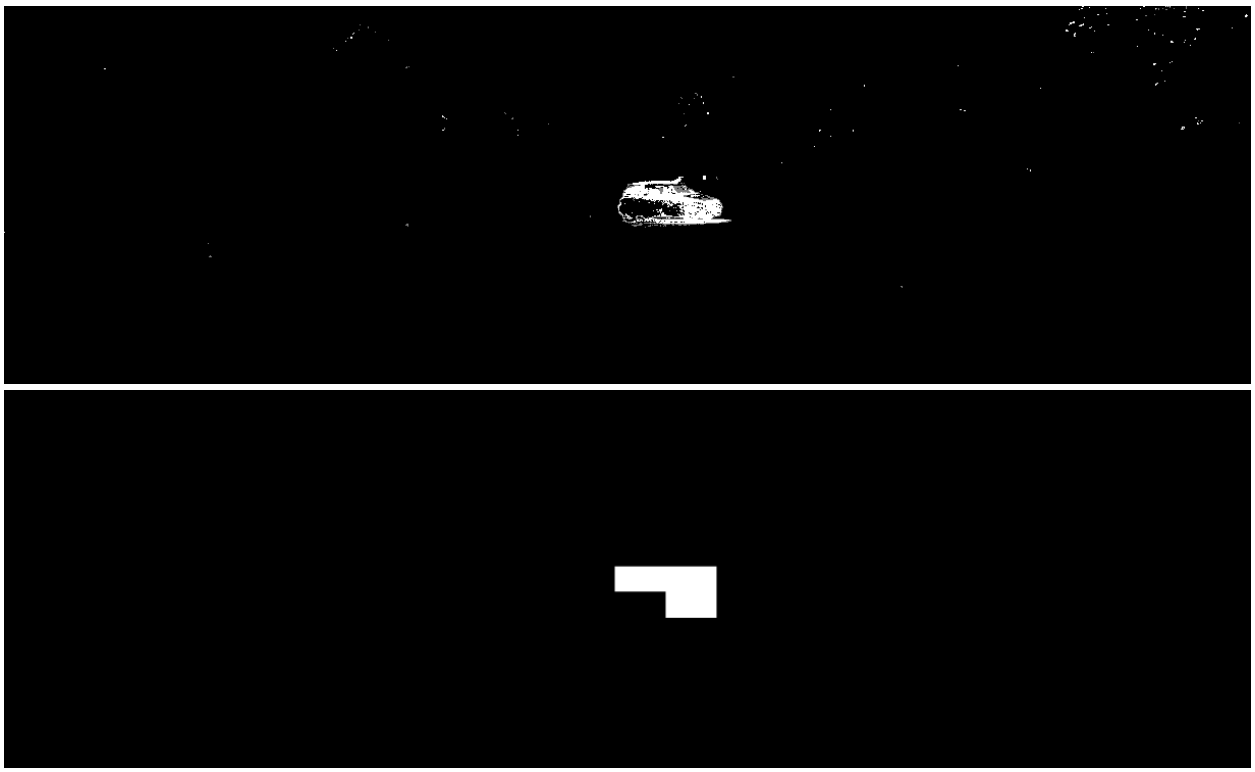

Рис. 4.9. Пример работы оконной фильтрации изображений (видеозаписи из набора данных kitty)

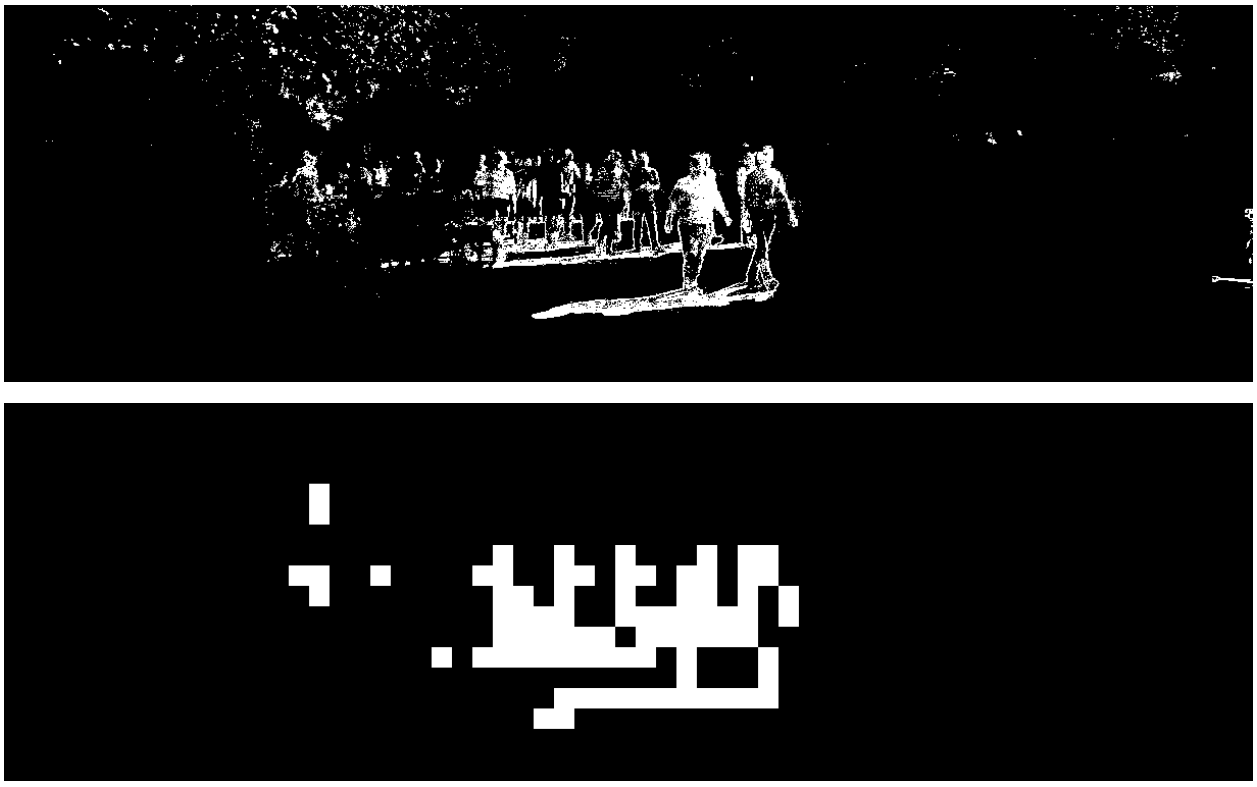

Рис. 4.10. Пример работы оконной фильтрации изображений (видеозаписи из набора данных kitty). Видно, что данный метод фильтрации выделяет те области видеокадра, в которых происходит наиболее интенсивное

движение, тем самым отмечаются объекты, в обнаружении движения которых мы заинтересованы

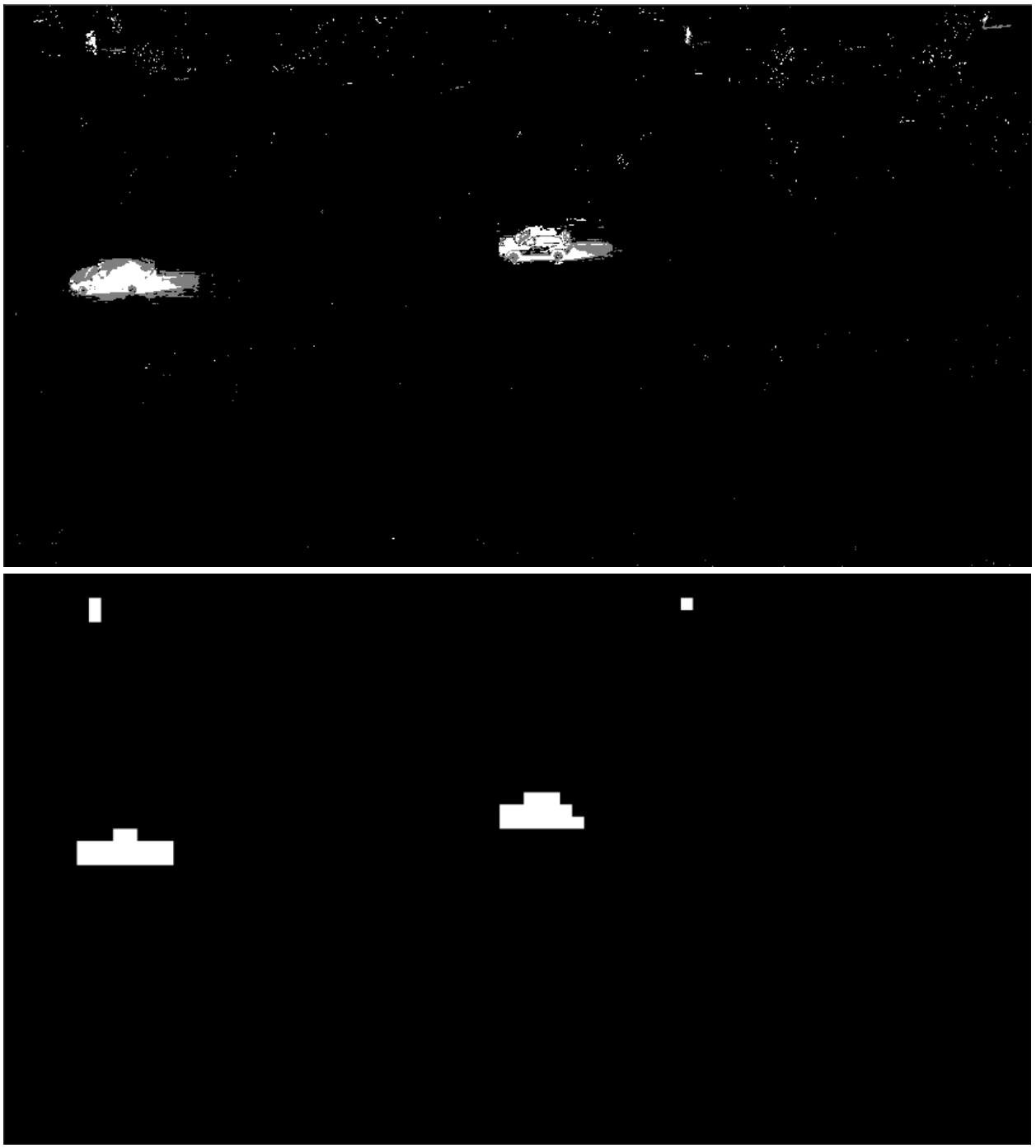

Рис. 4.11. Пример работы оконной фильтрации на изображении, представленного на рис.4.4

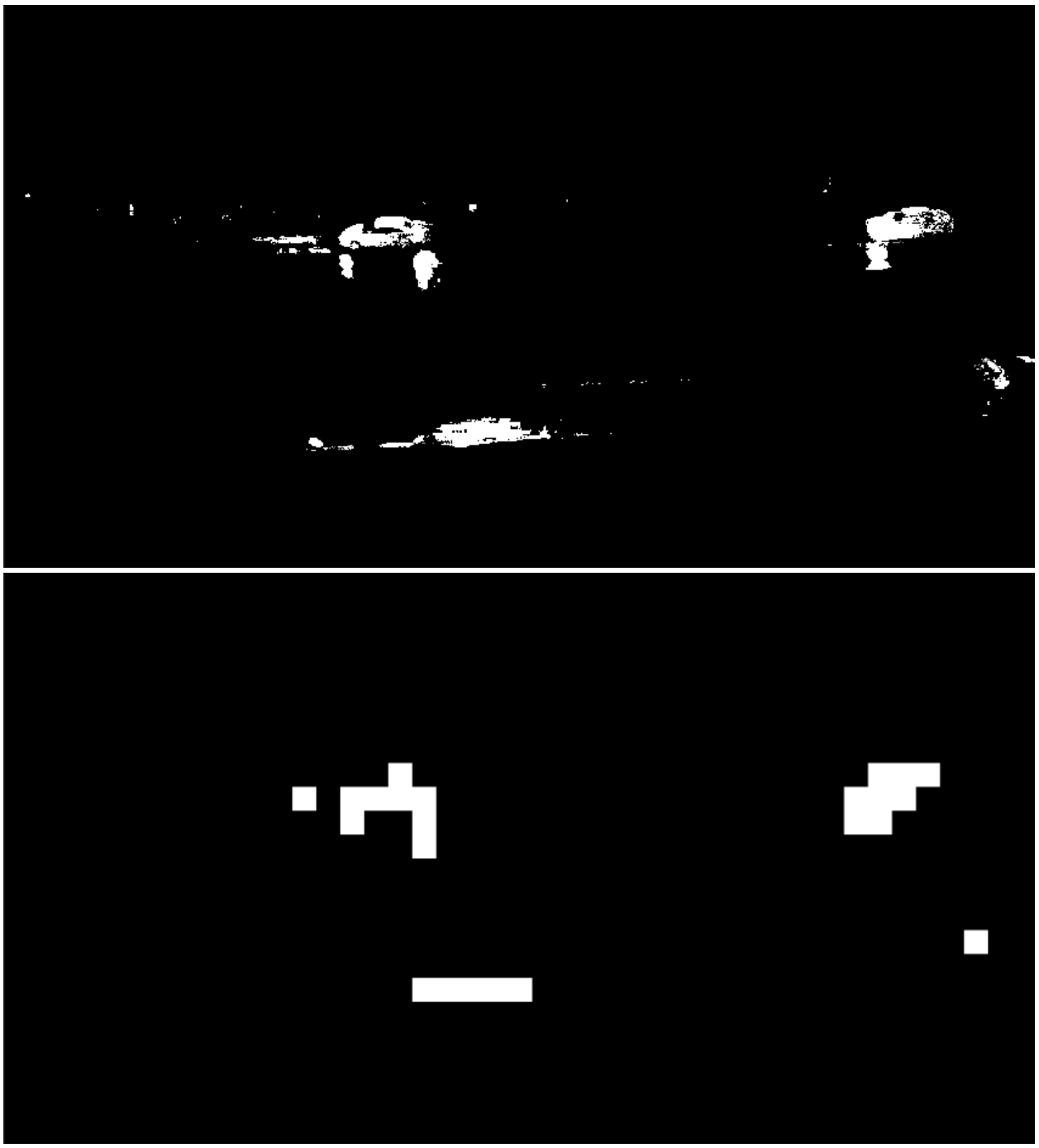

Рис. 4.12. Пример работы оконной фильтрации на изображении, представленного на рис.4.5

4.5 Оценка работы метода оконной фильтрации изображения

Основным способом оценки качества результатов работы фильтрации является визуальная оценка. Так, можно сделать вывод, что применение оконной фильтрации в зависимости от параметра плотности окна достаточно хорошо уменьшает количество шумов на изображении, улучшает качество

объектов, т.е. связывает объект в единое целое, убирая разрывы, и имеет при этом довольно высокую скорость обработки 1 кадра равную около 0,6 мс (скорость обработки зависит от количества движущихся объектов в видеокадре, тестирование проводилось на видеозаписях из набора данных kitty с разрешением 1224x370 пикселей на рабочей станции с характеристиками, указанными в таблице 4.1).

Таблица 4.1. Основные характеристики первой рабочей станции

Наименование составляющих	Описание
Центральный процессор	Intel Core i7-7700HQ
Графический процессор	NVIDIA GeForce 1050Ti
Объем оперативной памяти	8

4.6 Оценка работы алгоритма

Из рассмотренных ранее примеров можно увидеть, что алгоритм MOG v2, несмотря на искажение формы объектов, довольно хорошо справляется со своей основной задачей, а именно с задачей обнаружения движущихся объектов. Проблема разрывов в объектах (связность объекта в единое целое, вследствие этого отличимость его от окружающих шумов) и появления большого количества шумов на изображении в результате применения алгоритма MOG v2 эффективно решается путем применения метода оконной фильтрации изображения. Качество детектирования движущихся объектов довольно сильно зависит от скорости движения различных объектов в кадре и от степени цветового различия самого объекта и расположенного за ним фона (на это, среди прочего, влияют и условия съемки такие, как погодные условия, время, когда проводилась съемка (ночь, день) и т.п.). Так, лучшее качество обнаружения достигается при достаточной освещенности сцены, при хорошем контрасте между объектом и фоном и более быстром движении объекта в видеокадре.

Отметим также, что рассматриваемый алгоритм обладает довольно высокой скоростью обработки 1 кадра, которая составляет около 15 мс, что очень важно для систем, работающих в режиме реального времени (скорость обработки зависит от количества движущихся объектов в видеокадре, тестирование проводилось на изображениях (видеозаписи из набора данных kitty) с разрешением 1224x370 пикселей на рабочей станции с характеристиками, указанными в таблице 4.1).

Из недостатков алгоритма можно отметить, во-первых, невысокое качество обнаружения движущихся объектов, если объект и фон в кадре обладают схожей окраской, т.е. сливаются друг с другом, или при медленном движении крупного объекта в кадре, который обладает однородной либо однотонной текстурой (в таком случае происходит выделение не всего объекта, а лишь его границ), во-вторых, большое количество параметров, благодаря которым обеспечивается работа алгоритма MOG v2 и для которых необходимо подбирать оптимальные значения. Параметры можно менять в зависимости от различных условий съемки (например, предполагаемое среднее расстояние до движущихся объектов), наших потребностей и имеющихся вычислительных ресурсов.

ЗАКЛЮЧЕНИЕ

В рамках данной работы мною были:

1. Рассмотрены существующие на сегодняшний день методы обнаружения движущихся объектов с данных камер, основанные на вычислении оптического потока и вычитании фона.
2. Подробно изучен и выбран для тестирования алгоритм MOG v2 - метод обнаружения движущихся объектов с данных камер, основанный на смеси гауссовых распределений и использующий метод вычитания фона.
3. Изучен итерационный EM – алгоритм кластерного анализа, необходимый для работы выбранного алгоритма, а именно для обновления параметров вероятностной модели.
4. Рассмотрены различные методы фильтрации изображения.
5. Изучен и выбран для последующего тестирования метод оконной фильтрации.
6. Использована реализация на языке программирования python алгоритма MOG v2 для тестирования его работы на тестовых видео (включая видеофайлы из набора данных kitty), различной степени сложности: условия достаточной и недостаточной освещенности, наличие большого и малого количества движущихся объектов в сцене, условия сильного и слабого цветового различия между движущимся объектом и фоном позади него, а также различной скоростью движения объектов в кадре.
7. По итогам тестирования проведена оценка работы алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. Xun Wang, Jie Sun, Hao-Yu Peng. Foreground Object Detecting Algorithm based on Mixture of Gaussian and Kalman Filter in Video Surveillance [электронный ресурс] // Сайт ResearchGate. URL: https://www.researchgate.net/publication/272799045_Foreground_Object_Detecting_Algorithm_based_on_Mixture_of_Gaussian_and_Kalman_Filter_in_Video_Surveillance
2. Nima Sammaknejad, Yujia Zhao, Biao Huang. A review of the Expectation Maximization algorithm in data-driven process identification [электронный ресурс] // Сайт ResearchGate. URL: https://www.researchgate.net/publication/330069649_A_review_of_the_Expectation_Maximization_algorithm_in_data-driven_process_identification
3. Maya R. Gupta and Yihua Chen. Theory and Use of the EM Algorithm [электронный ресурс] // Сайт Mayagupta. URL: http://mayagupta.org/publications/EMbookGuptaChen2010.pdf
4. Chris Stauffer, W. Eric L. Grimson. Adaptive background mixture models for real - time tracking [электронный ресурс] // Сайт Csail. URL:http://www.ai.mit.edu/projects/vsam/Publications/stauffer_cvpr98_track.pdf
5. Lee P.H., Chiu T.H., Lin Y.L., Hung Y.P. Real-time pedestrian and vehicle detection in video using 3D cues [электронный ресурс] // Сайт ResearchGate. URL: https://www.researchgate.net/publication/221265866_Real-time_pedestrian_and_vehicle_detection_in_video_using_3D_cues
6. O. Barnich, M. Van Droogenbroeck. ViBe: Auniversal background subtraction algorithm for video sequences [электронный ресурс] // Сайт ResearchGate. URL:

https://www.researchgate.net/publication/224206851_ViBe_A_Universal_Background_Subtraction_Algorithm_for_Video_Sequences

7. Hui Yin, Yuanhao Gong, Guoping Qiu. Side Window Filtering [электронный ресурс] // Сайт CVF. URL: http://openaccess.thecvf.com/content_CVPR_2019/papers/Yin_Side_Window_Filtering_CVPR_2019_paper.pdf
8. Martin Haugh. Machine Learning for OR&FE [электронный ресурс] // Сайт Columbia 2015. URL: http://www.columbia.edu/~mh2078/MachineLearningORFE/EM_Algorithm.pdf
9. Роман Вячеславович Шамин. Лекции по искусственному интеллекту и машинному обучению [электронный ресурс] // Сайт Ai.lector. URL: http://ai.lector.ru/?go=lections
10. David J. Fleet, Yair Weiss. Optical Flow Estimation [электронный ресурс] // Сайт ResearchGate. URL: https://www.researchgate.net/publication/299690967_Optical_Flow_Estimation
11. Coderik. Вычисление оптического потока методом Лукаса-Канаде. Теория [электронный ресурс] // Сайт Habr. 18 февраля 2013. URL: https://habr.com/ru/post/169055/
12. ScayTrase. Трекинг точек. Lucas-Kanade [электронный ресурс] // Сайт Blog.scaytrase. URL:http://blog.scaytrase.ru/computer_vision/369/
13. Русанов А.Д., Некрасов Д.К. Обзор принципов работы и алгоритмов распознавания объектов окружающей среды в беспилотных автомобилях [электронный ресурс] // Сайт КиберЛенинка. URL: https://cyberleninka.ru/article/n/obzor-printsipov-raboty-i-algoritmov-raspoznavaniya-obektov-okruzhayuschey-sredy-v-bespilotnyh-avtomobilyah/viewer
14. Szeliski R. Computer Vision: Algorithms and Applications [электронный ресурс] // Сайт Szeliski. 2010. 979 стр. URL:

http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf

15. Sonka M., Hlavac V., Boyle R. Image processing, analysis and machine vision [электронный ресурс] // Сайт ResearchGate. URL: https://www.researchgate.net/publication/220695728_Image_processing_analysis_and_and_machine_vision_3_ed
16. Horn B., Schunk B. Determing Optical Flow [электронный ресурс] // Сайт di.ku. URL: http://image.diku.dk/imagecanon/material/HornSchunckOptical_Flow.pdf
17. Форсайт Д., Понс Ж. Компьютерное зрение. Современный подход. М.: Изд. д. Вильямс, 2004. 465с.
18. T. Bouwmans, F. El Baf, B. Vachon Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey [электронный ресурс] // Сайт ResearchGate. URL: https://www.researchgate.net/publication/215737671_Background_Modeling_Using_Mixture_of_Gaussians_for_Foreground_Detection-A_Survey
19. Leibe B., Leonardis A., Schiele B. Robust. Object Detection with Interleaved Object Categoization and Segmentation [электронный ресурс] // Сайт CiteSeerX. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.464&rep=rep1&type=pdf
20. A. P. Dempster, N. M. Laird, D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm [электронный ресурс] // Сайт Auburn. URL: https://www.eng.auburn.edu/~troppel/courses/7970%202015A%20AdvMobRob%20sp15/literature/paper%20W%20refs/dempster%20EM%201977.pdf
21. Helly M Desai, Vaibhav Gandhi. A Survey: Background Subtraction Techniques [электронный ресурс] // Сайт Ijser. URL:

https://www.ijser.org/researchpaper/A-Survey-Background-Subtraction-Techniques.pdf

22. Zoran Zivkovic. Improved Adaptive Gaussian Mixture Model for Background Subtraction. [электронный ресурс] // Сайт ResearchGate URL:https://www.researchgate.net/publication/4090386_Improved_Adaptive_Gaussian_Mixture_Model_for_Background_Subtraction
23. Padhraic Smyth. The EM Algorithm for Gaussian Mixtures Probabilistic Learning: Theory and Algorithms [электронный ресурс] // Сайт UCI. URL: https://www.ics.uci.edu/~smyth/courses/cs274/notes/EMnotes.pdf
24. Махаланобиса расстояние - Mahalanobis distance [электронный ресурс] // URL: https://ru.qwe.wiki/wiki/Mahalanobis_distance
25. Andreycha. Обзор алгоритмов кластеризации данных [электронный ресурс] // Сайт Hubr. 11 августа 2010. URL: https://habr.com/ru/post/101338/
26. Lior Rokach, Oded Maimon. Clustering Methods [электронный ресурс] // Сайт ResearchGate. January 2005. DOI: 10.1007/0-387-25465-X_15. URL:https://www.researchgate.net/publication/226748490_Clustering_Methods
27. Alexei Lufkin. Tips& Tricks: Fast Image Filtering Algorithms [электронный ресурс] // Сайт ResearchGate. January 2007. URL: https://www.researchgate.net/publication/228758794_Tips_tricks_Fast_image_filtering_algorithms
28. Narjis Mezaal Shati, Sundos Abdulameer Alazawi, Huda Abdulaali Abdulbaqi. Background subtraction (bs) using instant pixel histogram [электронный ресурс] // Journal of southwest jiaotong university. ISSN: 0258-2724. DOI：10.35741/issn.0258-2724.54.5.14. URL: http://jsju.org/index.php/journal/article/view/362/359