**Министерство цифрового развития, связи массовых коммуникаций Российской федерации**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**Разрешаю**

**допустить к защите**

**Зав. кафедрой**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_\_г.**

**БАКАЛАВРСКАЯ РАБОТА**

**НА ТЕМУ**

«РАЗРАБОТКА СИСТЕМЫ МОНИТОРИНГА СОСТОЯНИЯ КАМЕРЫ ВИДЕОНАБЛЮДЕНИЯ»

|  |  |  |  |
| --- | --- | --- | --- |
| **Студент** | Чевакин Ростислав Иванович | | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Руководитель** | | Мосева Марина Сергеевна | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**Москва 2022 г.**

**Аннотация**

Выпускная квалификационная работа на тему «Разработка системы мониторинга состояния камеры видеонаблюдения» состоит из 3 глав и 3 приложений, изложенных на 71 страницах, содержит 2 таблицы, 19 рисунков, список использованных источников содержит 24 наименований.

В первой главе рассматривается предметная область, проводится анализ существующих программно-аппаратных решений, пригодных для реализации систем мониторинга состояния камеры. Теоретическое исследование проводилось путем анализа литературы и электронных источников по заданной тематике.

Вторая глава содержит описание различных алгоритмов для решения данной проблемы. Также происходит выбор алгоритмов, которые будут использованы при проектировании системы.

Третья глава подробно описывает реализацию компонентов системы: разработку элементов модуля анализа изображений, разработку графического интерфейса пользователя, тестирование компонентов реализованной системы.

Ключевые слова: Computer vision, компьютерное зрение, анализ изображений, sabotage detector, определение размытия изображения, blur detection, детектор саботажа, системы видеонаблюдения.

**Содержание**

[**Введение** 7](#_Toc106692902)

[**Глава 1. Анализ предметной области** 9](#_Toc106692903)

[**1.1 Анализ исследований по теме** 9](#_Toc106692904)

[**1.2.1** **Видеокамеры** 13](#_Toc106692905)

[**1.2.2 Объективы** 16](#_Toc106692906)

[**1.2.3 Устройства записи изображения** 17](#_Toc106692907)

[**1.2.4 ПО для интеллектуального анализа** 18](#_Toc106692908)

[**1.3 Область применения систем видеонаблюдения** 20](#_Toc106692909)

[**1.4 Обзор существующих решений** 21](#_Toc106692910)

[**Выводы** 23](#_Toc106692911)

[**Глава 2. Анализ существующих алгоритмов** 25](#_Toc106692912)

[**2.1 Свёрточные нейронные сети** 25](#_Toc106692913)

[**2.2 Оператор Лапласа** 26](#_Toc106692914)

[**2.3. Преобразование Фурье** 28](#_Toc106692915)

[**2.4 Алгоритм сопоставления яркости** 30](#_Toc106692916)

[**Выводы** 31](#_Toc106692917)

[**Глава 3. Разработка приложения** 32](#_Toc106692918)

[**3.1 Используемые инструменты** 32](#_Toc106692919)

[**3.1.1 Язык программирования C#** 32](#_Toc106692920)

[**3.1.2 Библиотека OpenCV** 32](#_Toc106692921)

[**3.1.3 Accord.NET Framework** 33](#_Toc106692922)

[**3.1.4 Windows Forms .NET** 33](#_Toc106692923)

[**3.2 Описание структуры приложения** 34](#_Toc106692924)

[**3.3 Жизненный цикл программы** 36](#_Toc106692925)

[**3.4 Описание интерфейса приложения** 39](#_Toc106692926)

[**3.5 Тестирование приложения** 39](#_Toc106692927)

[**Заключение** 43](#_Toc106692928)

[**Список использованных источников** 44](#_Toc106692929)

[**Приложение А. Исходный код класса Form1** 47](#_Toc106692930)

[**Приложение Б. Исходный код класса Camera** 55](#_Toc106692931)

[**Приложение В. Исходный код класса Form2** 70](#_Toc106692932)

# **Введение**

В современных реалиях из-за сложившейся сложной геополитической ситуации все больше внимания уделяется системам обеспечения безопасности людей, одним из главных элементом которых являются системы видеонаблюдения. В связи с этим за 2021 год Россия вышла на второе место по темпу роста видеонаблюдения за гражданами. Общее число камер видеонаблюдения в общественных местах за год увеличилось на 10%. В настоящий момент в России установлено более 15 миллионов видеокамер.

За настоящее время всё больше внимания уделяется развитию интеллектуального уровня систем видеонаблюдения. На рынке существует огромное количество компаний, которые предлагают продукты, позволяющие выполнять обработку, хранение, отображения и анализ видеоданных, а также объединить множество рабочих мест и серверов в единую систему. Эти решения могут включать в себя специальные интеллектуальные модули, позволяющие решать задачи обнаружения и распознавания лиц, перехвата объектов, распознавания номеров, интерактивного поиска и т.д.

В то же время сама система видеонаблюдения может подвергаться атакам злоумышленников, что приводит к падению эффективности интеллектуального анализа и невозможности выполнения базовых функций по обеспечению безопасности. Для решения этой проблемы были разработаны различные алгоритмы обнаружение саботажа камеры видеонаблюдения.

Данная работа подготовлена с целью разработки системы, которая бы могла достаточно эффективно отслеживать различные саботажные действия с видеокамерой, не нагружая при этом сильно CPU и GPU. Саботажные действия, которые будет отслеживать система: отворот камеры, засветки, заслоны, расфокусировка камеры.

В данной работе объектом исследования является система видеонаблюдения, а предметом исследования – процесс разработки системы мониторинга состояния видеокамеры.

Задачи исследовательской работы:

1) Проанализировать предметную область и существующие решения по обнаружению отклонений параметров камеры.

2) Изучить и описать теоретические основы технологий, применяемых в системах мониторинга видеокамер для обнаружения корректной работы камеры.

3) Реализовать систему мониторинга состояния камер видеонаблюдения

4) Провести оценку предлагаемого решения.

В первой главе работы проведён анализ предметной области, более детальное описание поставленной проблемы и её последствий для камер видеонаблюдения. Также были описаны представленные на рынке решения.

Во второй главе был произведён анализ существующих исследований и на их основе были выделены некоторые из них, которые использовались при разработке системы мониторинга состояния камеры видеонаблюдения.

В третьей главе описывается разработка и тестирование системы на основе выбранных алгоритмов. Проводится анализ получившегося решения.

# **Глава 1. Анализ предметной области**

## **1.1 Анализ исследований по теме**

Определение отклонения параметров видеокамеры является актуальной и очень важной проблемой при построении любых программно-аппаратных систем видеонаблюдения или систем видения в автономном транспорте и производстве. Эта проблема является важной причиной снижения производительности интеллектуального анализа изображений. Исследователи выделяют несколько типов потери видимости камеры видеонаблюдения и используют различные подходы для их обнаружения.

Например, в работе [1] рассматриваются следующие категории:

1) Невольный разворот камеры проявляется в изменении ориентации камеры видеонаблюдения в следствии повреждения опоры видеокамеры либо, в случае, когда нарушитель целенаправленно её отклонил. Для того чтобы определять отворот камеры, авторы сохраняют до 5 кадров в оперативной памяти и устанавливают, преодолён ли порог допустимого изменения границ при развороте. Способ основывается на сопоставлении прошлого и текущего кадров с помощью компоненты яркости.

2) Дефокусировка совершается в следствии наложения объектива или светового фильтра на объект, запотевания стекла видеокамеры, появлении воды на объективе. С целью контролирования этих ситуаций авторы [1] используют следующий алгоритм: картинка разделяется на квадраты величиной 8x8 пикселей, а также внутри каждого подобного квадрата подсчитывается число контрастных пикселей. В случае если, их количество близко или равно нулю, то в таком случае считается, что квадрат является размытым. Уже после данной операции рассчитывается процент размытия и, если данный результат больше некоторого порогового значения, то принимается решение причислить данной камере видеонаблюдения статуса расфокусированной.

3) Засветка проявляется в сильном осветлении картинки, как реакция на ослепление видеокамеры внешним источником света, либо при сбое системы авто-экспозиции. Алгоритм базируется на управлении гистограммой изображения. Опираясь на полученные данные, вычисляются значения медианы, среднего и стандартного отклонения. В случае если какая-либо из характеристик резко поменяется, отображается сообщение об отвороте видеокамеры.

4) Перекрывание выражается в сильном затемнении линзы, когда она закрыта, а также в случае выхода из строя осветителей или сбоя системы автоматической экспозиции камеры. Алгоритм, используемый авторами, такой же, как и для проверки неожиданного поворота камеры.

Авторы исследования [2] описывают обнаружение трех категорий саботажа видеокамеры: перемещение(отворот), расфокусировка и закрытие. Они используют активность сигнала для определения количества информации в картинке, по этой информации с помощью модели Гауссовских микстур (GMM) определяется на сколько изображение отличается от нормального. Для обучения GMM использовались синтетические данные. С целью уменьшения влияния шума при моделировании изменений активности сигнала в изображении использовался фильтр Калмана.

Публикация [3] посвящена обнаружению некоторых дефектов объектива видеокамеры – грязи, капель дождя, и царапин. Основная идея данной работы заключается в том, что различные дефекты объектива будут отражаться на изображении появлением статичных областей. Однако данный подход работает только в случае, если камера находится в движении. Для поиска статичных частей изображения авторы использовали алгоритм, основанный на нормализованной кросс-корреляции для двух последовательных изображений.

В исследовательской работе [4] авторы описывают решение проблемы определения размытия изображения с помощью оператора Лапласа. Изображения для тестирования подвергались искажению с помощью аддитивного фильтра с нулевым средним гауссовым шумом. В результате оператор показал высокую эффективность для определения размытия.

Публикация [5] посвящена сравнению различных методов машинного обучения для определения частичной или полной потери видимости у видеокамеры. Для классификации использовались CatBoost, SVM и упрощённые архитектуры сетей VGG, ResNet, Inception V3. Тестирование проводилось на компьютере с Intel Core i5-2500 CPU, 3.30GHz×4, 8.00 GB RAM, NVIDIA GeForce GTX 1050 Ti (RAM: 4 GB) GPU, 64-bit операционной системой Windows 10. Самой эффективным способом определения оказалось нейросеть с архитектурой ResNet, самым эффективным алгоритмом на CPU оказался CatBoost.

В работе [6] авторы предлагают для определения размытия использовать преобразование Фурье совместно с методом опорных векторов. Для анализа изображение разбивалось на 9 частей и применялось преобразование к каждой части и целому изображению. Цель такого подхода заключается в том, чтобы профессиональные фотографии, которые могут быть хорошо сфокусированы на конкретном объекте, однако частично размыты не в фокусе, не определялись как размытые. По получившемся спектрам изображения SVM определял размытие картинки.

В данной работе [7] авторы производят оценку нескольких алгоритмов для определения размытия изображения, включая преобразование Фурье и оператор Лапласа. В итоге, по результатам тестирования, было выявлено что оператор Лапласа вычисляется в 6 раз быстрее преобразования Фурье и имеет 98% true positive срабатываний против 87% соответственно. Однако преобразование Фурье в целом является более точным алгоритмом чем оператор Лапласа (93% против 85%). Из чего можно сделать вывод о большом количестве ложноположительных срабатываний у оператора Лапласа.

Публикация [8] посвящена разработке детектора саботажа, который отслеживает отвороты, заслоны и расфокусировки видеокамеры. Для решения этой проблемы авторы предлагают алгоритм, который основан на отслеживании изменений фоновой модели изображения. Для создания фоновой модели используют неподвижные пиксели из 20-30 изображений видеоряда. После чего находят средние значения пикселей в конкретном местоположении. Чтобы получить хорошую модель рекомендуется использовать изображения с разной освещённостью. Для определения объектов переднего плана используется фильтрация всех пикселей картинки. Если значение пикселя отличается от пикселя из фоновой модели на некоторую пороговую величину, то он считается пикселем переднего объекта. Для избавления от шума отбрасываются все объекты, площадь которых меньше 8. Обнаружение закрытия видеокамеры происходит путём сравнения общей площади объектов переднего плана с некоторой пороговой величиной. Для определения размытия изображения используются пограничные пиксели фонового и текущего изображений. Картинка классифицируется как размытая, если у неё меньше пограничных пикселей, чем у фонового изображения, на некоторую пороговую величину. Для определения закрытия камеры видеонаблюдения используется площадь объектов переднего плана, если она больше порогового значения, то видеокамеру заслонили. При отклонении камеры от заданного положения, алгоритм посчитает что появился новый большой объект переднего плана, т.к. объекты получаются из сравнения пикселей текущего кадра и фона, а значит будет обнаружено саботажное действие. При тестировании данный алгоритм получил значение accuracy равное 95.65%, что является очень хорошим результатом.

**1.2 Основные составляющие систем видеонаблюдения**

Система видеонаблюдения — это программно-аппаратный комплекс, предназначенный для создания видеоконтроля как на территориально-распределённых, так и на локальных объектах [9].

### **1.2.1** **Видеокамеры**

Самыми часто используемыми видеокамеры являются камерами на базе ПЗС-матриц. Короткофокусные объективы типа фикс-фокус, не нуждающиеся в фокусировке, получили наиболее распространение. Основные производители матриц — Panasonic, Sharp, Sony, Samsung, Hynix. Применение матриц данных производителей позволило сформировать достаточно качественные продукты широкого применения по приемлемым ценам. В линейке продуктов любого вендора содержаться как дефолтные и недорогие по характеристикам матрицы, так и матрицы увеличенного разрешения и/или повышенной чувствительности.

По конструктивным особенностям камеры можно разделить на следующие типы:

* **Модульная камера видеонаблюдения (бескорпусные)** — представляют из себя плату на которой закреплен объектив камеры. Данный вид камер имеет схему обработки сигнала, объективом и матрицей. Требуется дополнительный уход из-за пыли. Изображение модульной камеры представлено на рисунке 1.1.



Рисунок 1.1 – Модульная камера

* **Минивидеокамера** — отличаются от обычных камер в основном только размером. Благодаря компактности данный вид видеокамер имеет возможность скрытой установки, однако они имеют более низкие характеристики нежели их крупные аналоги. Изображение минивидеокамеры представлено на рисунке 1.2.



Рисунок 1.2 – Минивидеокамера

* **Корпусная камера видеонаблюдения** — является камерой стандартизированного дизайна. Могут не комплектоваться кронштейнами, сетевыми адаптерами или объективом, что даёт определённую гибкость в использовании. Изображение корпусной видеокамеры представлено на рисунке 1.3.



Рисунок 1.3 – Корпусная видеокамера

* **Купольная камера**, кроме того известная также как **Dome camera** — размещенная в корпусе и защищенная сферической прозрачной крышкой видеокамера, которую устанавливают на горизонтальных поверхностях. Изображение купольной видеокамеры представлено на рисунке 1.4.



Рисунок 1.4 – Купольная видеокамера

* **Гиростабилизированные камеры** — камеры, применяемые в мобильных объектах с целью получения ровного изображения.
* **Управляемые** — комбинированный прибор, состоящий из трансфокатора, видеокамеры и поворачивающего устройства. Являются наиболее используемые встроенные камеры видеонаблюдения, произведённые в варианте купола.

По методу доставки информации видеокамеры подразделяются на **беспроводные и проводные**. Первые включают в себя передающее устройство и антенну. К типу беспроводных камер относятся цифровые IP-камеры, отправляющие картинку по сети Wi-Fi.

По виду выходного сигнала камеры видеонаблюдения разделяют на **цифровые и** **аналоговые**. Большая часть цифровых видеокамер отдают сигнал по стандартизированной сети Ethernet.

### **1.2.2 Объективы**

**Объектив** — прибор, специализирующийся на фокусировки светового потока на матрице видеокамеры.

Виды объективов:

1) Монофокальный - это статичный объектив, то есть он не приближает изображение и не расширяет обзор наблюдаемой территории. Работает в одном режиме - куда направили, там и снимает. Именно благодаря своей постоянности передает хорошую картинку при любой освещенности. Отлично подойдет для постоянного наблюдения территории площадью 10 м.кв.: офис, придомовая территория и т.д. Изображение монофокального объектива представлено на рисунке 1.5.



Рисунок 1.5 – Монофокальный объектив

2)Вариофокальный - это объектив, который может увеличивать картинку, причем без ухудшения качества. Но фокусировку и степень приближения к объекту наблюдения необходимо настраивать вручную на камере при установке. В дальнейшем прибегать к этой процедуре придется только при смене угла обзора. Применяют камеры с таким объективом там, где необходима постоянная увеличенная съемка определенной области, например, валютообменная касса в банке. Пример вариофокального объектива представлен на рисунке 1.6.



Рисунок 1.6 – Вариофокальный объектив

### **1.2.3 Устройства записи изображения**

Для записи, хранения и воспроизведения информации с видеокамеры используют специально предназначенное для этого устройство – видеорегистратор.

Принцип работы таков: камеры присоединяются к видеорегистратору и отправляют на него изображения по кабелям. Видеорегистратор получает сигнал от камеры, преобразует его в картинку, уменьшает размер, и, сохраняет на винчестер. Показ видео осуществляется с помощью подключения к видеорегистратору монитора или телевизора, а также возможно использование удаленного подключения.

Видеорегистратор является электронным устройством, по строению напоминает видеосервер или компьютер и включает в себя АЦП, жёсткий диск, процессор и другие компоненты. Для управления видеорегистратором обычно используются специализированные операционные системы. Перед началом записи оцифрованные изображения подвергаются компрессии с целью снижения занимаемого места. Почти все видеорегистраторы могут работать монохромными и с цветными видеоизображениями. Пример видеорегистратора представлен на рисунке 1.7.



**Рисунок 1.7 – Видеорегистратор для видеонаблюдения**

**1.2.4 ПО для интеллектуального анализа**

Интеллектуальные системы видеонаблюдения — представляют из себя аппаратно-программный комплекс, применяемый для автоматического сбора и анализа информации с потокового видео. Для работы этих систем используются различные алгоритмы распознания изображений, систематизации и обработки полученных данных.

Виды интеллектуальных систем видеонаблюдения:

**1)Встроенные интеллектуальные алгоритмы.**

Применяются напрямую в видеокамерах. Видеорегистратор либо сервер получает уже частично или полностью обработанное изображение с результатами анализа. Данный метод очень сильно снижает нагрузку на каналы передачи информации, однако данные камеры имеют небольшой выбор интеллектуальных функций, а стоимость значительно превышает обычные видеокамеры.

**2) Серверный.**

Интеллектуальная обработка данных происходит централизованно на удалённом сервере или компьютере. Для анализа используются CPU и GPU.

Использование программного обеспечения является ключевым достоинством серверной системы интеллектуального видеонаблюдения, т. к. оно позволяет добавлять и изменять модули и алгоритмы обработки видео.

К недостаткам серверной интеллектуальной обработки видеоданных можно отнести необходимость постоянной передачи изображений в высоком разрешении, что приводит к загрузке канала связи.

**3)Распределенная обработка видеоданных.**

Первичный анализ изображения, не требующий много вычислительных ресурсов, может производиться на видеокамерах. А более серьезный интеллектуальный анализ, требующий загрузки CPU или GPU, производится с помощью мощностей сервера.

Наиболее популярными видами алгоритмов для анализа видеоинформации являются:

**1) Ситуационный анализ** — применяется с целью выявления напряжённости с помощью анализа толпы людей. Например, определение нетипично большого количества людей для конкретного времени.

**2) Бизнес анализ** — используется для отслеживания работы персонала, выявления недовольных клиентов, исследования причин их недовольства, оптимизации процесса обслуживания.

Отличительная особенностью является большое количество отчетов с возможностью создания индивидуальных фильтров данных.

**3) Анализ контроля периметра** — применяется в системах, охраняющих участки периметра, которые имеют большую протяженность. Реагируют на скорость движения, местоположение и форму объекта.

**4) Номерной анализ** — применяется для распознавания номеров вагонов и автомобилей номеров и т.д.

**5) Биометрический анализ** — применяются разные способы биологической идентификации объекта.

**6) Анализ по нескольким камерам** — даёт возможность реализовать автоматическое сопровождение объекта с помощью нескольких камер. В результате будет сформирована траектория движения объекта по наблюдаемой зоне.

**7) Анализ технологических процессов** — количественный анализ формы объекта. Обеспечивает качество процесса производства.

**1.3 Область применения систем видеонаблюдения**

**Внедрение интеллектуальных систем видеонаблюдения повышает уровень безопасности, а также оказывает благоприятный экономический и организационный эффект. Эффективность от внедрения хорошо заметна не только в крупных сетях, но и в небольших системах малого бизнеса.**

Системы городской безопасности:

* определение драк и различных противоправных действий;
* определение мест подозрительной активности, нетипичных скоплений людей и т.д.
* распознавание лиц в розыске;

Транспорт:

* подсчет пассажиров;
* определение автомобильных номеров нарушителей или клиентов парковки;
* определение инородного объекта на рельсах или упавшего пассажира.
* определение пассажиров или оставленного предмета в охраняемой зоне;

Объекты закрытого или режимного типа:

* визуальное обнаружение очага возгорания на ранних стадиях;
* контроль периметра;
* контроль работы персонала;

Спортивные сооружения и развлекательные заведения:

* выявление чрезмерных скоплений людей;
* подсчет посетителей;
* оценка внимания посетителей.
* определение противоправных действий;

Организации общественного питания, торговые и банковские учреждения, автомойки, парикмахерские и т.п.:

* анализ длины очереди;
* оценка внимания персонала к клиентам;
* подсчет и классификация клиентов;

## **1.4 Обзор существующих решений**

1) Hikvision - один из главных мировых поставщиков новаторских продуктов и систем безопасности. Составные продуктовые вендора включают в себя: СКУД, системы видеонаблюдения, охранные сигнализации, видеостены, домофонию. Hikvision продвигает необходимые технологии кодирования видео и аудио, обработки видеоизображения и хранения данных, а также такие перспективные технологии, как deep learning, облачные вычисления и big data [10].

Камеры компании Hikvision имеют встроенные интеллектуальные функции такие как: детекция звуковых событий, обнаружение движения, детектора саботажа.

2) TRASSIR - производитель систем видеонаблюдения и разработчик программного обеспечения для видеонаблюдения. Все передовые решения, внедряемые компанией, разрабатываются собственными силами. Это позволяет выбранным элементам системы TRASSIR работать с максимальной производительностью своевременно и экономично [11].

ПО TRASSIR Sabotage Detector детектирует следующие действия с камерой как саботаж:

* Сдвиг - изменение направления съемки (изменение сцены);
* Дефокусировка - изменения размеров области съемки;
* Засветка - резкое увеличение освещенности области съемки;
* Обрыв связи - потеря сигнала с камеры;
* Закрытие - резкое уменьшение освещенности области съемки.

Данный детектор с помощью скрипта Alarm Monitor позволяет настроить реакции на инцидент, например:

* Привлечение внимания оператора звуковым сигналом и/или сообщением;
* Включение сирены;
* Управление сухими контактами (например, СКУД) и др.

3) Macroscop - производитель программных продуктов и аппаратных решений для систем видеонаблюдения. Компания специализируется на создании программного обеспечения, которое находит широкое использование в розничной торговле, интеллектуальных системах безопасности, робототехнике и других сферах жизни человека [12].

Камеры фирмы Macroscop в своём арсенале имеют модуль детектора саботажа. Детектор саботажа Macroscop распознает следующие события:

* Изменение положения видеокамеры (изменение области наблюдения);
* Расфокусировка камеры (существенное уменьшение резкости картинки);
* Засветка видеокамеры (в объектив камеры направляется яркий источник света).
* Закрытие видеокамеры (перед объективом камеры помещается посторонний объект);

В случае выявления факта саботажа программой, эта информация выводится на монитор и записывается в журнал событий. Оператор системы может настроить генерацию тревоги, отправку SMS или e-mail - сообщения при детекции саботажа.

## **Выводы**

Определение отклонения параметров видеокамеры является актуальной и очень важной проблемой в современном мире. Большое количество отраслей использует системы видеонаблюдения, не только для охраны, но и для отслеживания состояний объектов наблюдения. При перекрытии, засветки, заслона или отклонении от заданного положения камеры видеонаблюдения теряется или ухудшается качество интеллектуального анализа изображений. Если вовремя не исправить проблему, то владелец системы видеонаблюдения будет нести убытки, связанные не только с простоем аналитического устройства, но и возможным противоправным действием. Для преодоления этих проблем следует использовать системы мониторинга состояния видеокамеры, чтобы определить момент времени, когда произошло отклонение, засветка, заслон или размытие камеры.

Рассмотренное в данной главе программные обеспечения различных фирм являются встроенными системами мониторинга состояния видеокамеры. Из-за чего данное ПО нельзя использовать на камерах других вендоров. Для решения данной проблемы необходимо создать систему мониторинга, к которой можно подключить любую видеокамеру по IP, и оно бы не зависело от вендора. Также нужно добавить возможность регулирования строгости алгоритмов, т.к. у разных камер разные характеристики и что для одних является достаточно чёткой изображением, для других будет считаться размытым.

# **Глава 2. Анализ существующих алгоритмов**

## **2.1 Свёрточные нейронные сети**

Сверточная нейронная сеть – это вид нейронной сети, которая применяется для распознавания изображений. Она была разработана для работы с пиксельными картинками [13].

CNN – это современный искусственный интеллект для обработки изображений с помощью Deep Learning для выполнения как описательных задач, так и задач генерации изображений, часто с использованием компьютерного зрения, которое включает распознавание объектов на изображении и видео.

Нейронная сеть – это система аппаратного и программного обеспечения, которая работает похожим образом с нейронами в головном мозге. Классические сети плохо подходят для анализа изображений, т.к. им необходимо подавать кусочки изображения с уменьшенным разрешением. У CNN “нейроны” больше похожи на нейроны лобной доли, которая отвечает за обработку визуальных стимулов у животных.

CNN имеет систему, которая имеет сходство с многослойным персептроном, который был разработан для уменьшения требований к обработке изображений. Сверточная нейронная сеть состоит из выходного, входного и скрытого слоя. Скрытый слой может включать в себя несколько сверточных слоев, pooling слоёв, слоев нормализации и полносвязных слоев. Устранение ограничений и повышение эффективности обработки изображений приводит к созданию системы, которая является гораздо более эффективной, простой в использовании и ограниченной для обработки изображения.

Существуют различные доступные архитектуры CNN, которые сыграли ключевую роль в создании алгоритмов, которые обеспечивают и будут обеспечивать ИИ в целом в обозримом будущем. Некоторые из них перечислены ниже:

1.     GoogLeNet

2.    ResNet

3.    VGGNet

4.    LeNet

5.    AlexNet

6.    ZFNet

Преимущества:

1) Высокая точность

2) Универсальность

Недостатки:

1) Сложность реализации

2) Требовательны к ресурсам

3) Нужно много данных

**2.2 Оператор Лапласа**

Оператор Лапласа – это дифференциальный оператор, заданный расходимостью градиента скалярной функции на евклидовом пространстве. В декартовой системе координат лапласиан задается суммой вторых частных производных функции по каждой независимой переменной [14].

– оператор Лапласа. (2.1)

Оператор Лапласа был назван в честь французского математика Пьера-Симона де Лапласа, который впервые применил этот оператор для изучения небесной механики: лапласиан гравитационного потенциала, обусловленного заданным распределением плотности массы, является постоянным кратным этому распределению плотности. Результат применения оператора Лапласа к изображению представлен на рисунке 2.1.



Рисунок 2.1 – Применение оператора Лапласа

Лапласиан используется во многих дифференциальных уравнениях, описывающих физические явления. Уравнение Пуассона описывает электрический и гравитационный потенциалы; волновое уравнение описывает распространение волн, уравнение диффузии описывает поток тепла и жидкости и уравнение Шредингера в квантовой механике. Оператор Лапласа может использоваться для обнаружения капель и краев на изображении.

Существует два типа оператора Лапласа:

1) Положительный оператор Лапласа: Он состоит из стандартной маски, в которой угловые элементы должны быть равны нулю маски, а центральные элементы должны быть отрицательными по отношению к маске. Пример маски положительного оператора Лапласа приведён на рисунке 2.2.

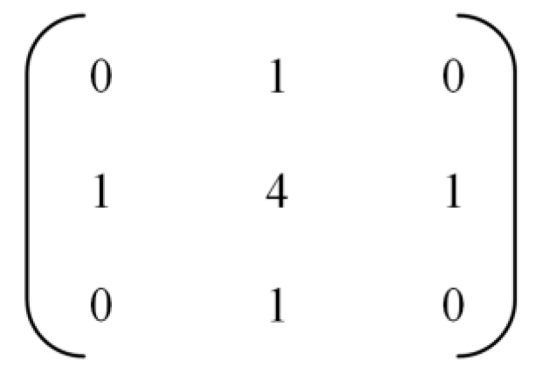


Рисунок 2.2 – Маска положительного оператора Лапласа

2) Отрицательный оператор Лапласа. Он состоит из базовой маски, в которой угловые элементы должны быть равны нулю маски, а центральные элементы должны быть положительными по отношению к маске, а оставшийся элемент должен быть равен -1. Внутренние края изображения удаляются в положительном операторе Лапласа. Пример маски отрицательного оператора Лапласа приведён на рисунке 2.3.

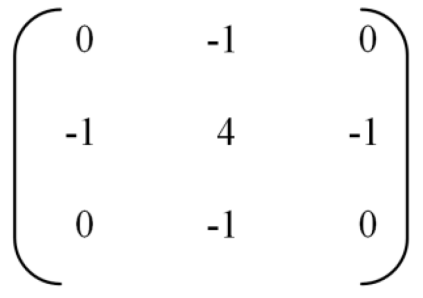


Рисунок 2.3 – Маска отрицательного оператора Лапласа

Преимущества:

1) Прост в реализации

2) Скорость вычисления.

Недостатки:

1) Ложно положительные срабатывания

**2.3. Преобразование Фурье**

**Преобразование Фурье**— это операция, которая сопоставляет две функции вещественных переменных. Эта получившаяся функция описывает коэффициенты при разложении исходной функции на простые составляющие — гармонические колебания с разными частотами [15].

, где k = 0, 1, …, N-1 (2.2)

Преобразование Фурье применяется в анализе частотных характеристик различных фильтров. Для нахождения частотной области у изображений используется 2D дискретное преобразование Фурье (DFT). Для вычисления DFT используется быстрый алгоритм, который называется быстрым преобразованием Фурье (FFT). Результат применения преобразования Фурье к изображению представлен на рисунке 2.4.

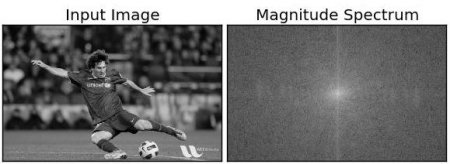


Рисунок 2.4 – Применение преобразования Фурье

При быстром преобразовании Фурье алгоритм вычисляет частоты в изображении в разных точках и на основе уровня частот решает, является ли изображение размытым или нет. Когда присутствует малое количество высокой частоты, то он определяет, что изображение размыто, и решение о том, каким будет пороговые значения для высокой и низкой частоты, зависит от ситуации.

Достоинства:

1) Высокая точность

Недостатки:

2) Достаточно медленный алгоритм

## **2.4 Алгоритм сопоставления яркости**

Алгоритм сопоставления яркости представляет из себя эвристический алгоритм, который определяет на сколько две картинки отличаются друг от друга с помощью анализа яркостной компоненты изображения.

Изображения для анализа берутся через каждые пять кадров. Эти картинки переводятся в HSV цветовое пространство и берется компонент V(яркость) и вычисляются среднее, медиана, среднеквадратичное отклонение и 75 перцентиль. Потом вычисляется расстояние между двумя точками. (где среднее, медиана, RMSD и перцентиль являются координатами в 4-х мерном пространстве). Если расстояние больше порогового значения, то значит обнаружено саботажное действие. Формула 2.3 представляет из себя математическое представление данного алгоритма, где m – среднее, med -медиана, RMSD – среднее квадратичное отклонение [16], p75 – 75 перцентиль.

(2.3)

Преимущества:

1) Простота реализации

2) Скорость вычисления

Недостатки:

1) Ложно положительные срабатывания

## **Выводы**

Несмотря на то, что алгоритмы на основе свёрточных нейронных сетей имеют высокую эффективность в распознавании саботажных действий, они являются слишком ресурсозатратными для системы мониторинга. В разработке системы мониторинга отклонений параметров видеокамеры играет намного более важную роль true positive срабатывания нежели false positive, т.к. выход из строя камеры видеонаблюдения неприемлем. Поэтому основными алгоритмами системы будут оператор Лапласа для определения размытия и алгоритм сопоставления яркости для всего остального.

# **Глава 3. Разработка приложения**

## **3.1 Используемые инструменты**

В данной работе были выбраны язык программирования C#, .NET Framework, WinForms и OpenSource библиотека компьютерного зрения OpenCV, а точнее её обертка для платформы .NET EmguCV [17].

### **3.1.1 Язык программирования C#**

**C#** — современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать разные типы безопасных и надежных приложений, выполняющихся в .NET [18].

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов, анонимные функции с поддержкой замыканий, атрибуты, события, свойства, итераторы, делегаты, исключения[19].

Достоинства:

* Сборщик мусора
* Простота написания кода
* WinForms
* Многопоточность

### **3.1.2 Библиотека OpenCV**

OpenCV - это библиотека программного обеспечения для компьютерного зрения и машинного обучения с открытым исходным кодом. Библиотека проектировалась с целью создания общей инфраструктуры для приложений, использующих компьютерное зрение и ускорения использования машинного зрения в коммерческих продуктах [20].

OpenCV имеет большое количество оптимизированных алгоритмов, которые включают в себя полный набор как классических, так и современных алгоритмов компьютерного зрения и машинного обучения. Эти алгоритмы можно использовать для идентификации объектов, распознавания лиц, классификации действий человека в видео, отслеживания движений камеры, отслеживания движущихся объектов, извлечения 3D-моделей объектов и т.д. Сообщество OpenCV насчитывает более 46 тысяч человек, а количество загрузок может превышает 18 миллионов. Библиотека широко применяется в компаниях, исследователями и государственными органами.

### **3.1.3 Accord.NET Framework**

Accord.NET Framework - это платформа машинного обучения .NET в сочетании с библиотеками обработки аудио и изображений, полностью написанными на C#. Это полноценная платформа для создания приложений компьютерного зрения производственного уровня, компьютерного слуха, обработки сигналов и статистики даже для коммерческого использования. Для создания приложения из этого фреймворка понадобятся функции для статистического анализа [21].

### **3.1.4 Windows Forms .NET**

Windows Forms — это технология для создания пользовательского интерфейса в .NET, представляет из себя набор библиотек, которые упрощают выполнение стандартных задач. С помощью среды разработки Visual Studio можно создавать клиентские приложения Windows Forms, способные запрашивать ввод пользователя, взаимодействовать с удалёнными компьютерами и отображать информацию в форме [22].

В WinForms форма — это окно, на которой могут выводится различные элементы управления для пользователя. Приложение создаётся с помощью добавления элементов управления в формы и создания кода обработки событий на действия пользователя, такие как нажатия клавиш или щелчки мыши. Элемент управления — представляет из себя отдельную часть пользовательского интерфейса, который используется для ввода и вывода информации пользователю.

Когда пользователь выполняет какое-либо действие в форме с одним из её элементов управления, то генерируется событие. Приложение же обрабатывает эти события с помощью кода.

В WinForms присутствует множество элементов управления, которые можно добавлять в формы. Элементы управления могут отображать раскрывающиеся списки, текстовые поля, переключатели, кнопки и даже веб-страницы.

Используя функцию перетаскивания **конструктора Windows Forms** в Visual Studio. Для преодоления трудностей, связанных с выравниванием элементов управления, конструктор предоставляет такие средства, как линии сетки и линии привязки. С помощью элементов управления FlowLayoutPanel, TableLayoutPanel и SplitContainer можно гораздо быстрее создавать сложные макеты форм.

Наконец, если нужно создать свои собственные элементы пользовательского интерфейса, пространство имен System.Drawing содержит широкий набор классов, необходимых для отрисовки линий, кругов и других фигур непосредственно на форме.

**3.2 Описание структуры приложения**

Приложение по мониторингу состояния камеры видеонаблюдения состоит из 3 основных классов: Form1, Camera, Form2.

Класс Form1 представляет из себя основную форму программы с помощью которой происходит управление камерами. Этот класс наследуется от System.Windows.Forms.Form, содержит обработчики событий нажатий различных кнопок и смены значения выпадающего списка камер. Для логирования данный класс использует объект System.Collections.Generic.Queue, который представляет из себя очередь. Логи в процессе работы добавляются в этот объект, после чего раз в 10 секунд из очереди удаляются элементы и записываются в файл. Это сделано для того, чтобы не нагружать систему лишними операциями открытия/закрытия файлового дескриптора. Также класс Form1 содержит в себе объект System.Collections.Generic.Dictionary для хранения экземпляров класса Camera, где ключом является свойство cameraName. С помощью этого словаря можно доставать конкретный экземпляр класса по названию, что может быть полезным при использовании выпадающего списка с названиями камер.

Класс Form2 является дополнительным окном, которое открывается, когда пользователь хочет добавить камеру в приложение. Этот класс, также как и Form1, наследуется от System.Windows.Forms.Form, однако содержит куда меньший набор методов. Создание экземпляра класса Form2 происходит из основной формы. Форма для добавления камеры представляет из себя 4 текстовых поля с двумя кнопка. Данная форма представлена на рисунке 3.1.

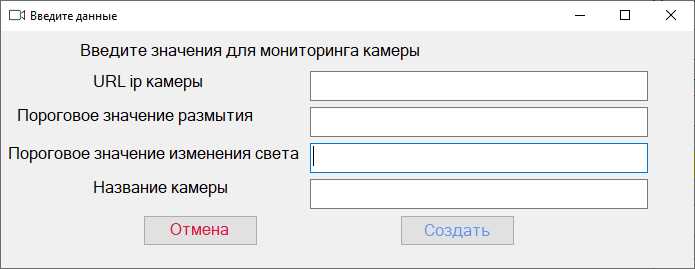


Рисунок 3.1– Форма добавления новой камеры

Класс Camera представляет из себя класс, в котором происходит основная логика анализа изображения. Класс содержит свойства для подключения к IP камере, пороговые значения для алгоритмов, а также название самой камеры. Также данный объект отправляет сообщения для логов о найденных проблемах в основную форму. Данные сообщения отправляются только в случае возникновения какого-либо события. В файл логи событий записываются раз в 10 секунд.

Диаграмма основных классов приложения представлена на рисунке 3.2

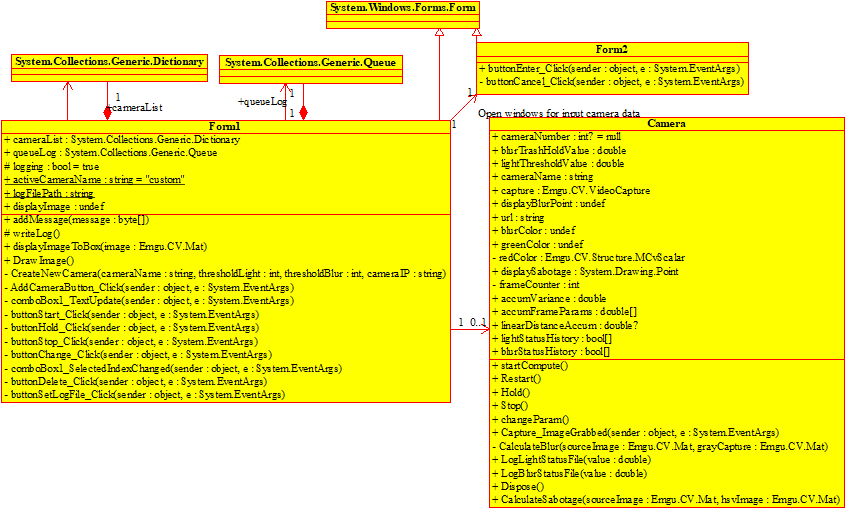


Рисунок 3.2 – Основные классы приложения

## **3.3 Жизненный цикл программы**

Точкой входа программы является статичный internal класс Programm, который в функции Main запускает основную форму. Метод AddCameraButton\_Click создаёт вспомогательную форму Form2 для ввода данных новой видеокамеры и привязывается обработчик события getNewCameraData.

После ввода данных камеры и подтверждения создания, объект Form2 генерирует событие getNewCameraData, которое перехватывает основная форма, сам же объект вспомогательной формы уничтожается. Метод CreateNewCamera создаёт новый экземпляр класса Camera, после чего записывает его в словарь, где ключом служит название камеры. Экземпляр класса запускается в отдельном потоке для анализа. Также происходят подписки на события по логированию и отображению изображения.

Метод Capture\_ImageGrabbed класса Camera считывает кадры с видеокамеры, после чего анализирует их на наличие отклонений. Метод выводит изображение на форму при условии, если выбранная камера пользователем имеет такое же название, как и текущая. Для экономии вычислительных ресурсов анализируется только каждый пятый кадр, а также вручную удаляются объекты, созданные во время выполнения кода.

Метод CalculateBlur класса Camera определяет размытость изображения. Для этого применяется оператор Лапласа к картинке, после чего изображения нормализуется и вычисляется дисперсия. Если она больше порогового значения, установленного в объекте Camera, то к изображение считается размытым. Результат анализа записывается в картинке в виде текста, а также в логи. Все математические операции выполняются с помощью OpenCV.

Метод CalculateSabotage класса Camera представляет из себя программную реализацию алгоритма сопоставления яркости. Для вычисления среднего, медианы, среднеквадратичного отклонения и 75-перцентиль был использован класс Accord.Statistics.Measures который входи в состав Accord.NET Framework. Для вычисления Евклидова расстояния был использован также этот фреймворк. Также метод отображает результат анализа на изображении и записывает логи. Для ускорения вычисления были проведены некоторые оптимизации вычислений параметров для алгоритма, а именно:

1) Исходный массив был отсортирован т.к. методам для вычисления медианы и перцентили в любом случае приходится это делать. Таким образом удаётся сэкономить время на одну сортировку.

2) Для вычисления среднеквадратичного отклонения используется результат вычисления среднего значения.

3) Для вычисления квантили и медианы был использован самый быстрый алгоритм, а именно R3[23]. Самый быстрый алгоритм был выведен эмпирическим путём.

4) Все вычисления, которые можно выполнить параллельно, выполняются в отдельных потоках. В начале выполняется подсчёт среднего значения т.к. этот результат потом подаётся для вычисления среднеквадратичного отклонения.

После анализа изображение с результатами выводится в форму. Перед каждым обновлением картинки предыдущее значение элемента формы PictureBox уничтожается, чтобы избежать лишних вызовов сборщика мусора.

Жизненный цикл программы представлен на рисунке 3.3.

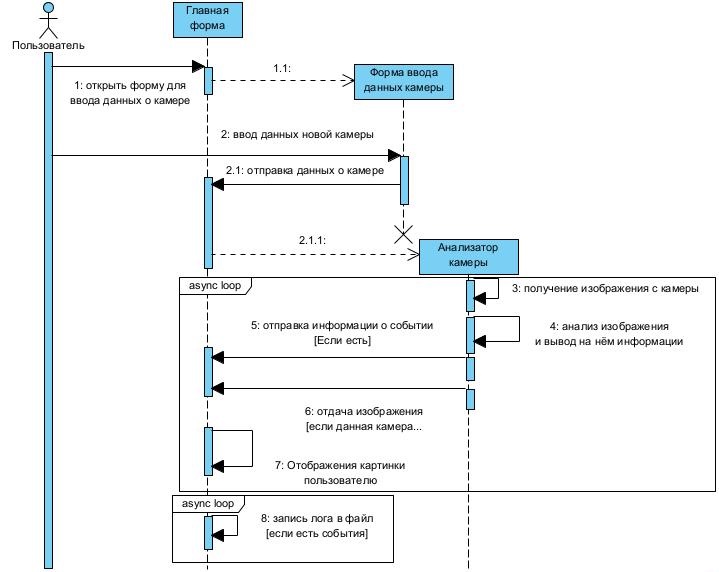


Рисунок 3.3 – Диаграмма последовательности приложения

## **3.4 Описание интерфейса приложения**

В приложении можно подключиться к различным IP камерам для анализа, а также выставит пороговые значения для алгоритмов. Задаётся новая камера при нажатии на плюсик. Все подключённые видеокамеры анализируются в реальном времени и записывают в лог события. Выбрать файл, в который будет записываться лог можно при нажатии на кнопку «Файл лога». Отображаемую камеру можно менять через выпадающий список. Также в реальном времени можно менять пороговые значения у алгоритмов. Камеру при желании можно удалить или отключить анализ. Если будет найдено размытие, то текст в верхней левой части изображения станет красным. Если будет обнаружено какое-либо отклонение параметров видеокамеры, то слева сверху на второй строчке будет написано красным буквами «Light sabotage detected». Интерфейс программы изображён на рисунке 3.4.



Рисунок 3.4 – Интерфейс программы

## **3.5 Тестирование приложения**

Для тестирования системы мониторинга состояния камеры видеонаблюдения была выбрана метрика F2-мера с приоритетом полноты [24], т.к. в данной задаче гораздо важнее выявить ложно отрицательный результат. Формулы 3.1, 3.2, 3.3, используются для расчёта F2-меры, где TP – true positive, FN – false negative, FP – false positive

(3.1)

(3.2)

(3.3)

В тестовый набор данных входило 60 видеозаписей, из которых 20 были контрольными, остальные были распределены равномерно по различным отклонениям параметров видеокамеры. Результат определения размытия представлен на рисунке 3.5.



Рисунок 3.5 – Размытое изображение

Результат определения засветки камеры видеонаблюдения представлен на рисунке 3.6.

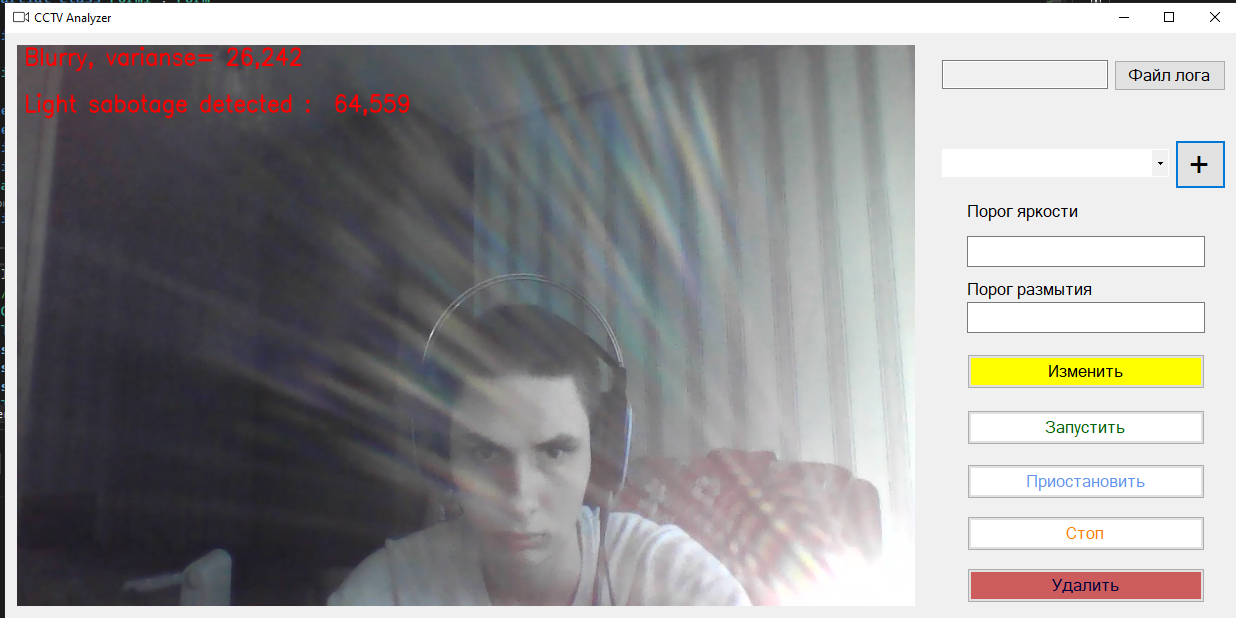


Рисунок 3.6 – Определения засветки программой

Определение заслона видеокамеры представлен на рисунке 3.7.

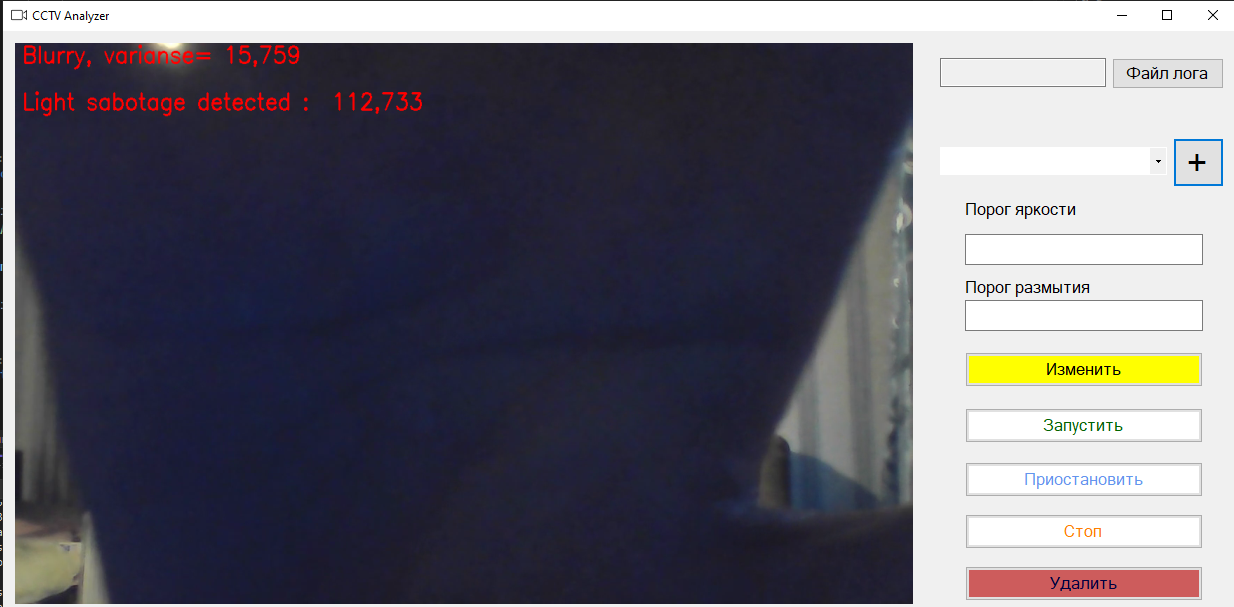


Рисунок 3.7 – Определение заслона программой

Определение отклонения камеры от заданного положения приложением представлено на рисунке 3.8.

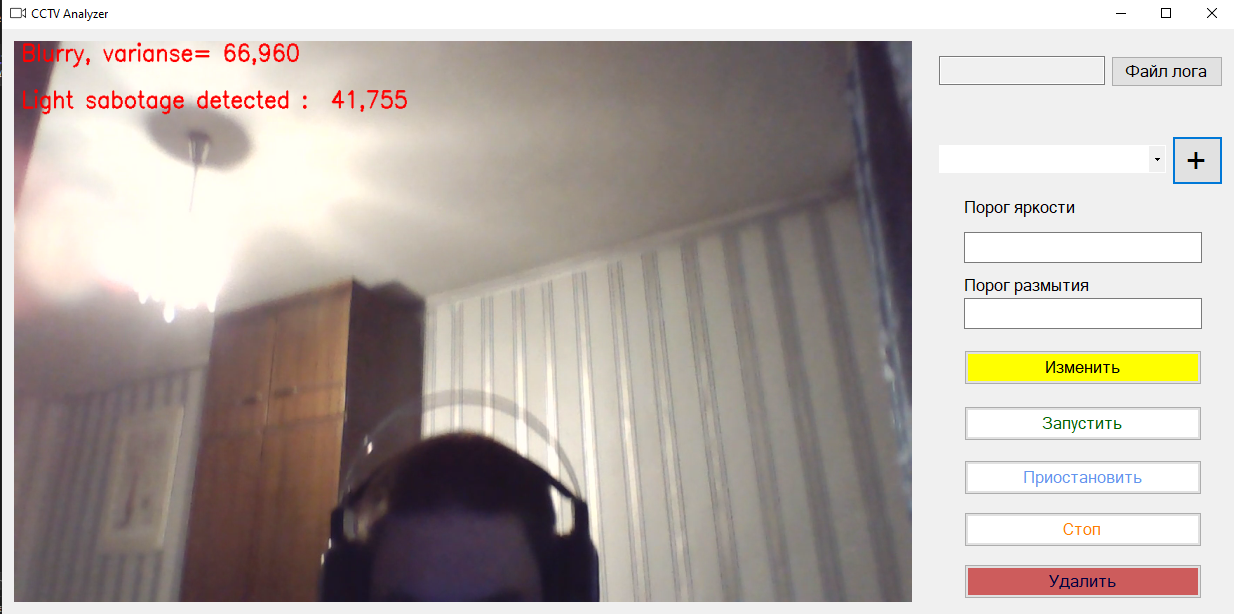


Рисунок 3.8 – Определение отклонения камеры

Результаты тестирования приложения приведены в таблице 3.1.

,,Таблица 3.1 – Результаты тестирования программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вид отклонения | Размытие | Отклонение положения | Засветка | Заслон |
| TP | 9 | 8 | 9 | 9 |
| FP | 5 | 4 | 4 | 4 |
| TN | 15 | 16 | 16 | 16 |
| FN | 1 | 2 | 1 | 1 |

Результат вычисления метрики F2-меры для каждого вида отклонения приведены в таблице 3.2.

Таблица 3.2 – Значение F2-меры для разных алгоритмов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вид отклонения | Размытие | Отклонение положения | Засветка | Заслон |
| F2-мера | 0.83 | 0.77 | 0.85 | 0.85 |

# **Заключение**

В рамках выполнения дипломной работы были исследованы алгоритмы для определения отклонений параметров камеры видеонаблюдения и был пройден ряд этапов, в ходе которых были достигнуты следующие результаты:

1) Проведен анализ предметной области: рассмотрены основные понятия, определения и дальнейшие перспективы развития интеллектуального анализа изображений с камер видеонаблюдения.

2) Был выбран инструментарий, изучены особенности его применения и использование определенных навыков и знаний, полученных в ходе обучения в университете.

3) В ходе исследования алгоритмов была создана эффективная система мониторинга состояния камер видеонаблюдения.

4) Были проведены оптимизации алгоритмов и работы системы.

5) Был проведен анализ результатов тестирования, в результате которого было выявлено, что данные алгоритмы имеют место хорошую точность при определении: засветок, отклонений камеры от заданного положения, размытия и заслонов.

# **Список использованных источников**

1. Tikhova, J. Development sabotage detectors for surveillance systems Macroscop // Perm State National Research University, 2013. URL: https://scienceforum.ru/2013/article/2013007653 (дата обращения 12.06.2022)

2. Mantini, P., Shah, S. K.: A Signal Detection Theory Approach for Camera Tamper Detection // In Proceedings of the 14th IEEE International Conference on Advanced Video and Signal based Surveillance, Lecce, Italy (2017). URL: https://www.researchgate.net/publication/317972940\_A\_Signal\_Detection\_Theory\_Approach\_for\_Camera\_Tamper\_Detection (дата обращения 12.06.2022)

3. Einecke, N., Gandhi, H., Deigmöller, J.: Detection of camera artifacts from camera images // IEEE 17th International Conference on Intelligent Transportation Systems (ITSC), pp. 603–610 (2014).

4. Parvesh Kumar, Gaurav Raj, Tanupriya Choudhury: Blur image detection using Laplacian operator and Open-CV // 2016 International Conference System Modeling & Advancement in Research Trends (SMART). URL: https://www.researchgate.net/publication/288485202\_Detection\_of\_camera\_artifacts\_from\_camera\_images (дата обращения 12.06.2022).

5. Alexey Ivanov, Dmitry Yudin: Visibility Loss Detection for Video Camera Using Deep Convolutional Neural Networks: Volume 1. Proceedings of the Third International Scientific Conference “Intelligent Information Technologies for Industry” (IITI’18) (pp.434-443) (2019).

6. Eftichia Mavridaki, Vasileios Mezaris: No-reference blur assessment in natural images using Fourier transform and spatial pyramids. January 2015. DOI:10.1109/ICIP.2014.7025113.

7. Roxanne A. Pagaduan, Ma. Christina R. Aragon and Ruji P. Medina: iBlurDetect: Image Blur Detection Techniques Assessment and Evaluation Study // Technological Institute of the Philippines, Information Technology Department Quezon City, URL: https://www.scitepress.org/Papers/2020/103077/103077.pdf (дата обращения 12.06.2022).

8. K. Sitara, B. Mehtre : Real-Time Automatic Camera Sabotage Detection for Surveillance Systems. DOI:10.1007/978-3-319-28658-7\_7

9. Herman Kruegle. CCTV Surveillance Video Practices and Technology. 2nd Edition – December 2, 2006. eBook ISBN: 9780080468181. URL: https://books.google.ru/books?id=DaQY8CrmqFcC&pg=PA1&redir\_esc=y#v=onepage&q&f=false.

10. О компании | Hikvision [Электронный ресурс] URL: https://hikvision.ru/about (дата обращения 12.06.2022).

11. About TRASSIR [Электронный ресурс] URL: https://trassir.com/about/ (дата обращения 12.06.2022).

12. Модуль детектора саботажа. | Macroscop [Электронный ресурс] URL: https://macroscop.com/produkty/programma-dlya-ip-kamer/detektor-sabotazha (дата обращения 12.06.2022).

13. Keiron O’Shea, Ryan Nash: An Introduction to Convolutional Neural Networks. URL: https://arxiv.org/pdf/1511.08458.pdf (дата обращения 12.06.2022).

14. Uwe Semmelmann, Gregor Weingart: The Standard Laplace Operator, URL: https://www.researchgate.net/publication/319151317 (дата обращения 12.06.2022).

15. Г. Нуссбаумер. Быстрое преобразование Фурье и алгоритмы вычисления сверток. Издательство «Радио и связь», 1985

16. What does RMSE really mean? [Электронный ресурс] URL: https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e (дата обращения 12.06.2022).

17. Документация по Emgu CV [Электронный ресурс] URL: https://www.emgu.com/wiki/files/4.5.5/document/html/R\_Project\_Emgu\_CV\_Documentation.htm (дата обращения 12.06.2022).

18. Документация по C# | Microsoft Docs [Электронный ресурс] URL: https://docs.microsoft.com/ru-RU/dotnet/csharp/ (дата обращения 12.06.2022).

19. Албахари Бен, Албахари Джозеф. C# 7.0. Справочник. Полное описание языка. Издательство «Вильямс» , 2018, 1024 с.

20. Документация по OpenCV [Электронный ресурс] URL: https://docs.opencv.org/4.x/index.html (дата обращения 12.06.2022).

21. Документация по Accord.NET Framework [Электронный ресурс] URL: http://accord-framework.net/docs/html/R\_Project\_Accord\_NET.htm (дата обращения 12.06.2022).

22. Документация по Windows Forms для .NET | Microsoft Docs [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netframeworkdesktop-4.8 (дата обращения 12.06.2022).

23. Rob J. Hyndman and Yanan Fan: Sample Quantiles in Statistical Packages // The American Statistician Vol. 50, No. 4 (Nov., 1996), pp. 361-365 (5 pages)(дата обращения 12.06.2022).

24. Yutaka Sasaki Research Fellow: The truth of the F-measure // School of Computer Science, University of Manchester MIB, 131 Princess Street, Manchester, M1 URL: https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf (дата обращения 12.06.2022).

# **Приложение А. Исходный код класса Form1**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using Emgu.CV;

using System.Threading;

using System.IO;

namespace CCTVAnalyzer

{

public partial class Form1 : Form

{

public delegate void InvokeDrawImage();

Dictionary<string, Camera> cameraList = new Dictionary<string, Camera>();

protected Queue<byte[]> queueLog = new Queue<byte[]>();

protected bool logging = true;

public static string activeCameraName = "custom";

public static string logFilePath = "";

Bitmap displayImage = new Bitmap(1000, 1000);

public Form1()

{

InitializeComponent();

Task logger = new Task(() => writeLog());

logger.Start();

}

public void addMessage(byte[] message)

{

this.queueLog.Enqueue(message);

}

protected void writeLog()

{

while (logging)

{

Thread.Sleep(10000);

byte[] buffer;

if (logFilePath != "")

{

try

{

FileStream fs = new FileStream(logFilePath, FileMode.Append);

while (queueLog.Count > 0)

{

buffer = queueLog.Dequeue();

fs.Write(buffer, 0, buffer.Length);

}

fs.Close();

} catch

{

}

}

}

}

public void displayImageToBox(Mat image)

{

displayImage = image.ToBitmap();

IAsyncResult res = pictureBox1.BeginInvoke(new InvokeDrawImage(DrawImage));

image.Dispose();

pictureBox1.EndInvoke(res);

}

public void DrawImage()

{

if (pictureBox1.Image != null)

{

pictureBox1.Image.Dispose();

}

pictureBox1.Image = displayImage;

}

private void CreateNewCamera(string cameraName, int thresholdLight, int thresholdBlur, string cameraIP)

{

Camera someCamera = new Camera(cameraIP, thresholdBlur, thresholdLight , cameraName);

cameraList.Add(cameraName, someCamera);

Task someTask = new Task(() => someCamera.startCompute());

comboBox1.Items.Add(cameraName);

someTask.Start();

someCamera.calculateImage += displayImageToBox;

someCamera.Log += addMessage;

}

private void AddCameraButton\_Click(object sender, EventArgs e)

{

Form2 inputBox = new Form2();

inputBox.getNewCameraData += CreateNewCamera;

inputBox.ShowDialog();

}

private void buttonStart\_Click(object sender, EventArgs e)

{

if (cameraList.ContainsKey(comboBox1.Text))

{

Task someTask = new Task(() => cameraList[activeCameraName].startCompute());

someTask.Start();

}

}

private void buttonHold\_Click(object sender, EventArgs e)

{

cameraList[activeCameraName].Hold();

}

private void buttonStop\_Click(object sender, EventArgs e)

{

cameraList[activeCameraName].Stop();

}

private void buttonChange\_Click(object sender, EventArgs e)

{

cameraList[activeCameraName].changeParam(Convert.ToDouble(textBoxBlur.Text), Convert.ToDouble(textBoxLight.Text), "");

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

activeCameraName = comboBox1.Text;

textBoxBlur.Text = cameraList[activeCameraName].blurTrashHoldValue.ToString();

textBoxLight.Text = cameraList[activeCameraName].lightThresholdValue.ToString();

}

private void buttonDelete\_Click(object sender, EventArgs e)

{

cameraList[activeCameraName].Dispose();

cameraList.Remove(activeCameraName);

comboBox1.Items.Remove(activeCameraName);

textBoxBlur.Text = "";

textBoxLight.Text = "";

}

private void buttonSetLogFile\_Click(object sender, EventArgs e)

{

using (OpenFileDialog openFileDialog = new OpenFileDialog())

{

openFileDialog.InitialDirectory = "c:\\";

openFileDialog.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

openFileDialog.FilterIndex = 2;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

logFilePath = openFileDialog.FileName;

textBoxLogPath.Text = openFileDialog.FileName;

}

}

}

}

}

# **Приложение Б. Исходный код класса Camera**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Emgu.CV;

using Emgu.Util;

using Emgu.CV.Structure;

using System.Drawing;

using Accord.Statistics;

using static System.GC;

using System.Threading;

namespace CCTVAnalyzer

{

public class Camera : IDisposable

{

public string url = "";

public int? cameraNumber = null;

public double blurTrashHoldValue = 0;

public double lightThresholdValue = 0;

public string cameraName;

VideoCapture capture;

Point displayBlurPoint = new Point(10, 25);

Point displaySabotage = new Point(10, 85);

MCvScalar blurColor = new MCvScalar(255, 0, 0);

MCvScalar greenColor = new MCvScalar(0, 255, 0);

MCvScalar redColor = new MCvScalar(0, 0, 255);

int frameCounter = 0;

int frameCounterBlur = 0;

double accumVariance = 0;

double[] accumFrameParams = new double[4];

double? linearDistanceAccum = null;

bool[] lightStatusHistory = { false, false };

bool[] blurStatusHistory = { false, false };

public Camera(string url, float blurTrashHoldValue, float lightThresholdValue, string cameraName, int? cameraNumber = null)

{

this.url = url;

this.blurTrashHoldValue = blurTrashHoldValue;

this.lightThresholdValue = lightThresholdValue;

this.cameraName = cameraName;

this.cameraNumber = cameraNumber;

}

public delegate void displayImage(Mat mat);

public event displayImage calculateImage;

public delegate void LogHandler(byte[] message);

public event LogHandler Log;

public void startCompute()

{

if (cameraNumber != null)

{

capture = new VideoCapture((int)cameraNumber);

}

else

{

capture = new VideoCapture(url);

}

capture.ImageGrabbed += Capture\_ImageGrabbed;

capture.Start();

}

public void Restart()

{

capture.ImageGrabbed += Capture\_ImageGrabbed;

capture.Start();

}

public void Hold()

{

capture.Pause();

}

public void Stop()

{

capture.Stop();

frameCounter = 0;

accumVariance = 0;

accumFrameParams = new double[4];

linearDistanceAccum = null;

this.lightStatusHistory[0] = false;

this.lightStatusHistory[1] = false;

this.blurStatusHistory[0] = false;

this.blurStatusHistory[1] = false;

}

public void changeParam(double? blurTrashHoldValue, double? lightThresholdValue, string cameraName)

{

capture.Pause();

if (blurTrashHoldValue != null)

{

this.blurTrashHoldValue = (double)blurTrashHoldValue;

}

if (lightThresholdValue != null)

{

this.lightThresholdValue = (double)lightThresholdValue;

}

if (cameraName != "")

{

this.cameraName = cameraName;

}

capture.Start();

}

private void Capture\_ImageGrabbed(object sender, EventArgs e)

{

Mat sourceImage = new Mat();

Mat grayImage = new Mat();

Mat hsvImage = new Mat();

capture.Read(sourceImage);

if (sourceImage.IsEmpty)

{

return;

}

Thread.Sleep(200);

this.frameCounter++;

this.frameCounterBlur++;

CvInvoke.CvtColor(sourceImage, grayImage, Emgu.CV.CvEnum.ColorConversion.Bgr2Gray);

grayImage.ConvertTo(grayImage, Emgu.CV.CvEnum.DepthType.Cv8U);

sourceImage = CalculateBlur(sourceImage, grayImage);

CvInvoke.CvtColor(sourceImage, hsvImage, Emgu.CV.CvEnum.ColorConversion.Rgb2Hsv);

sourceImage = CalculateSabotage(sourceImage, hsvImage);

if (Form1.activeCameraName == this.cameraName)

{

calculateImage(sourceImage);

}

hsvImage.Dispose();

sourceImage.Dispose();

grayImage.Dispose();

}

private Mat CalculateBlur(Mat sourceImage, Mat grayImage)

{

double variance;

if (frameCounterBlur > 5)

{

Mat outlineImage = new Mat();

CvInvoke.Laplacian(grayImage, outlineImage, Emgu.CV.CvEnum.DepthType.Cv8U, 1);

CvInvoke.Normalize(outlineImage, outlineImage, 255, 0, Emgu.CV.CvEnum.NormType.MinMax, Emgu.CV.CvEnum.DepthType.Cv8U);

var mass = outlineImage.GetData();

byte[] data = new byte[mass.Length];

double[] data2 = new double[mass.Length];

Buffer.BlockCopy(mass, 0, data, 0, mass.Length);

for (int i = 0; i < data.Length; i++)

{

data2[i] = (double)data[i];

}

variance = Measures.Variance(data2);

outlineImage.Dispose();

accumVariance = variance;

frameCounterBlur= 0;

}

else

{

variance = accumVariance;

}

if (variance > blurTrashHoldValue)

{

CvInvoke.PutText(sourceImage, string.Format("Not Blurry, varianse= {0:f3}", variance), displayBlurPoint, Emgu.CV.CvEnum.FontFace.HersheySimplex, 1, blurColor, 2);

blurStatusHistory[0] = blurStatusHistory[1];

blurStatusHistory[1] = false;

}

else

{

CvInvoke.PutText(sourceImage, string.Format("Blurry, varianse= {0:f3}", variance), displayBlurPoint, Emgu.CV.CvEnum.FontFace.HersheySimplex, 1, redColor, 2);

blurStatusHistory[0] = blurStatusHistory[1];

blurStatusHistory[1] = true;

}

grayImage.Dispose();

LogBlurStatusFile(variance);

return sourceImage;

}

public void LogLightStatusFile(double value)

{

if (Form1.logFilePath != "")

{

if (!lightStatusHistory[0] && lightStatusHistory[1])

{

byte[] message = Encoding.UTF8.GetBytes(DateTime.Now.ToString() + " Обнаружено отклонение у камеры:" + cameraName + " световой гаммы:" + value.ToString() + "\n");

Log(message);

}

else if (lightStatusHistory[0] && !lightStatusHistory[1])

{

byte[] message = Encoding.UTF8.GetBytes(DateTime.Now.ToString() + " Световая гамма камеры:" + cameraName +" вернулась в норму:" + value.ToString() + "\n");

Log(message);

}

}

}

public void LogBlurStatusFile(double value)

{

if (Form1.logFilePath != "")

{

if (!blurStatusHistory[0] && blurStatusHistory[1])

{

byte[] message = Encoding.UTF8.GetBytes(DateTime.Now.ToString() + " Обнаружено размытие у камеры " + cameraName + " : " + value.ToString() + "\n");

Log(message);

}

else if (blurStatusHistory[0] && !blurStatusHistory[1])

{

byte[] message = Encoding.UTF8.GetBytes(DateTime.Now.ToString() + " Чёткость изображения камеры:" + cameraName + " вернулось в норму:" + value.ToString() + "\n");

Log(message);

}

}

}

public void Dispose()

{

capture.Stop();

capture.Dispose();

}

private Mat CalculateSabotage(Mat sourceImage, Mat hsvImage)

{

var mass = hsvImage.GetData();

byte[] flatImage = new byte[mass.Length];

double[] componentHData = new double[(mass.Length) / 3];

double linearDistance = 0;

Buffer.BlockCopy(mass, 0, flatImage, 0, mass.Length);

int j = 0;

for (int i = 2; i < mass.Length; i = i+3)

{

componentHData[j++] = (double)flatImage[i];

}

Array.Sort(componentHData);

var mean = Task.Run(() => Measures.Mean(componentHData));

var quantile90 = Task.Run(() => Measures.Quantile(componentHData, 0.75, true, QuantileMethod.Type3));

var median = Task.Run(() => Measures.Median(componentHData, true, QuantileMethod.Type3));

mean.Wait();

var std = Task.Run(() => Measures.StandardDeviation(componentHData, mean.Result));

quantile90.Wait();

std.Wait();

median.Wait();

if (this.frameCounter == 1)

{

this.accumFrameParams[0] = quantile90.Result;

this.accumFrameParams[1] = std.Result;

this.accumFrameParams[2] = mean.Result;

this.accumFrameParams[3] = median.Result;

}

else if (this.frameCounter > 7)

{

double[] currentParams = { quantile90.Result, std.Result, mean.Result, median.Result };

linearDistance = Accord.Math.Distance.Euclidean(accumFrameParams, currentParams);

this.frameCounter = 1;

if (linearDistance > lightThresholdValue)

{

CvInvoke.PutText(sourceImage, string.Format("Light sabotage detected : {0:f3}", linearDistance), displaySabotage, Emgu.CV.CvEnum.FontFace.HersheySimplex, 1, redColor, 2);

lightStatusHistory[0] = lightStatusHistory[1];

lightStatusHistory[1] = true;

}

else

{

this.accumFrameParams[0] = quantile90.Result;

this.accumFrameParams[1] = std.Result;

this.accumFrameParams[2] = mean.Result;

this.accumFrameParams[3] = median.Result;

CvInvoke.PutText(sourceImage, string.Format("Light sabotage no detected : {0:f3}", linearDistance), displaySabotage, Emgu.CV.CvEnum.FontFace.HersheySimplex, 1, greenColor, 2);

lightStatusHistory[0] = lightStatusHistory[1];

lightStatusHistory[1] = false;

}

linearDistanceAccum = linearDistance;

}

else

{

if (linearDistanceAccum > lightThresholdValue)

{

CvInvoke.PutText(sourceImage, string.Format("Light sabotage detected : {0:f3}", linearDistanceAccum), displaySabotage, Emgu.CV.CvEnum.FontFace.HersheySimplex, 1, redColor, 2);

lightStatusHistory[0] = lightStatusHistory[1];

lightStatusHistory[1] = true;

}

else if (linearDistanceAccum >= 0)

{

CvInvoke.PutText(sourceImage, string.Format("Light sabotage no detected : {0:f3}", linearDistanceAccum), displaySabotage, Emgu.CV.CvEnum.FontFace.HersheySimplex, 1, greenColor, 2);

lightStatusHistory[0] = lightStatusHistory[1];

lightStatusHistory[1] = false;

}

}

LogLightStatusFile(Convert.ToDouble(linearDistanceAccum));

quantile90.Dispose();

std.Dispose();

mean.Dispose();

median.Dispose();

return sourceImage;

}

}

}

# **Приложение В. Исходный код класса Form2**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CCTVAnalyzer

{

public partial class Form2 : Form

{

public delegate void inputData(string cameraName, int thresholdLight, int thresholdBlur, string cameraIP);

public event inputData getNewCameraData;

public Form2()

{

InitializeComponent();

}

private void buttonEnter\_Click(object sender, EventArgs e)

{

string cameraName = textBoxName.Text;

int thresholdLight = Convert.ToInt32(textBoxLight.Text);

int thresholdBlur = Convert.ToInt32(textBoxBlur.Text);

getNewCameraData.Invoke(textBoxName.Text, Convert.ToInt32(textBoxLight.Text), Convert.ToInt32(textBoxBlur.Text), TextBoxIP.Text);

this.Close();

}

private void buttonCancel\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}