Министерство науки и высшего образования Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

Институт машиностроения, материалов и транспорта

Высшая школа автоматизации и робототехники

Работа допущена к защите

Директор ВШАиР

\_\_\_\_\_\_\_\_\_\_\_ А.Н. Волков

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**РАЗРАБОТКА СИСТЕМЫ ДЛЯ ОБНАРУЖЕНИЯ И ФИКСАЦИИ ОПТИКО-ЭЛЕКТРОННЫХ ПРИБОРОВ**

по направлению 15.03.06 «Мехатроника и робототехника»

по профилю подготовки 15.03.06\_04 «Автономные роботы»

Выполнил

студент гр.3331506/60401 <*подпись*> А.В. Стрекозов

Руководитель

Старший преподаватель

ВШАиР ИММиТ <*подпись*> А.С. Габриель

Консультант

по нормоконтролю <*подпись*> С.Г. Чупров

Научный консультант <*подпись*> В.А. Буняков

Санкт-Петербург

2020

**САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО**

**Институт машиностроения, материалов и транспорта**

**Высшая школа автоматизации и робототехники**

УТВЕРЖДАЮ

Руководитель ОП

О.Н. Мацко

«19 »декабря 2019 г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы**

студенту Стрекозову Артуру Владимировичу, гр 3331506/60401

фамилия, имя, отчество (при наличии), номер группы

1. Тема работы: Разработка системы для обнаружения и фиксации оптико-электронных приборов

2. Срок сдачи студентом законченной работы: «26» июня 2020 г.

3. Исходные данные по работе: Требования к разрабатываемому алгоритму.

4. Содержание работы (перечень подлежащих разработке вопросов): Обзор миниатюрных камер; обзор устройств обнаружения скрытых видеокамер; создание и сравнение алгоритмов автоматического обнаружения и фиксации скрытых камер видеонаблюдения.

5. Перечень графического материала (с указанием обязательных чертежей):

6. Консультанты по работе: Буняков Владимир Александрович, начальник лаборатории телевизионных информационно-измерительных систем ЦНИИ РТК

7. Дата выдачи задания «11» декабря 2019 г.

Руководитель ВКР А.С. Габриель

(подпись) инициалы, фамилия

Задание принял к исполнению «11» декабря 2019 г.

(дата)

Студент А.В. Стрекозов

(подпись) инициалы, фамилия

РЕФЕРАТ

Отчёт 56 с., 15 рис., 5 табл., 20 источн., 6 прил.

КЛЮЧЕВЫЕ СЛОВА: ОБНАРУЖЕНИЕ СКРЫТОЙ КАМЕРЫ, СВЕТОВОЗВРАЩЕНИЕ, СТЕНОП, ПИНХОЛ, АКТИВНЫЙ КАДР, ПАССИВНЫЙ КАДР, РАЗНОСТНОЕ ИЗОБРАЖЕНИЕ, ДЕТЕКТОР КЕННИ, ПОРОГОВОЕ ИЗОБРАЖЕНИЕ, МОДЕЛЬ HSV, ПАРАЛЛЕЛЬНОЕ ПОТОКОВОЕ ВИДЕО, ВИДЕОАНАЛИТИКА.

Цель работы – создание системы по автоматическому обнаружению и фиксации скрытых камер видеонаблюдения.

В результате исследования представлен краткий обзор существующих миниатюрных камер, включая «стеноп» («pinhole») камеры, приведены их модели и характеристики. Также проведён обзор разработок в области обнаружения скрытых видеокамер, приведены типы таких устройств и их характеристики. Реализован алгоритм, позволяющий выявлять скрытые видеокамеры.

**ABSTRACT**

56 pages, 15 figures, 5 tables, 20 sourses, 6 appendices

KEY WORDS: HIDDEN CAMERA DETECTION, retroreflection, stenop, pinhole, active frame, passive frame, differential image, Kenny detector, thresholded image, HSV model, concurrent video streaming, video content analysis.

The purpose of the work is to create a system for the automatic detection and fixation of hidden surveillance cameras.

The study provides a brief overview of existing miniature cameras, including a «pinhole» camera, their models and characteristics. A review of developments in the field of detection of hidden video cameras was also conducted, the types of such devices and their characteristics are given. An algorithm has been implemented to detect hidden video cameras.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc43989666)

[1 Аналитический обзор современного состояния в области скрытого видеонаблюдения. 8](#_Toc43989667)

[2 Аналитический обзор современного состояния в области средств обнаружения камер скрытого видеонаблюдения 12](#_Toc43989668)

[2.1 Электромагнитные детекторы скрытых камер 12](#_Toc43989669)

[2.2 Оптические детекторы скрытых камер 14](#_Toc43989670)

[2.3 Выводы по второй главе 17](#_Toc43989671)

[3 Разработка алгоритма по автоматическому определению и фиксации скрытых камер видеонаблюдения 18](#_Toc43989672)

[3.1 Программное обеспечение и использованные материалы 18](#_Toc43989673)

[3.2 Алгоритм программы. Предварительная подготовка кадров 18](#_Toc43989674)

[3.2.1 Предварительная подготовка кадров. Сегментация 19](#_Toc43989675)

[3.2.2 Предварительная подготовка кадров. Вычисление разностного кадра 20](#_Toc43989676)

[3.3 Алгоритм программы. Вариативность 21](#_Toc43989677)

[3.3.1 Вариативность. Canny. 22](#_Toc43989678)

[3.3.2 Вариативность. Пороговое преобразование. 26](#_Toc43989679)

[3.3.3 Вариативность. Модель HSV 29](#_Toc43989680)

[3.4 Сравнение алгоритмов 30](#_Toc43989681)

[3.5 Алгоритм программы. Видеопоток 31](#_Toc43989682)

[3.6 Выводы по третьей главе 33](#_Toc43989683)

[ЗАКЛЮЧЕНИЕ 35](#_Toc43989684)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 37](#_Toc43989685)

[ПРИЛОЖЕНИЕ 39](#_Toc43989686)

[Приложение А. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Вариант с использованием детектора Canny. 39](#_Toc43989687)

[Приложение Б. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Вариант с использованием пороговой фильтрации. 42](#_Toc43989688)

[Приложение В. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Вариант с использованием модели HSV. 46](#_Toc43989689)

[Приложение Г. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Видеопоток. Заголовочный файл. 48](#_Toc43989690)

[Приложение Д. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Видеопоток. Калибровочный файл. 49](#_Toc43989691)

[Приложение Е. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Видеопоток. Основной файл. 51](#_Toc43989692)

# ВВЕДЕНИЕ

Цель работы – создание алгоритма по автоматическому обнаружению и фиксации скрытых камер видеонаблюдения.

В современном мире развитие электроники, оптики и интегральных микросхем развилось до уровня, когда стало возможно создавать видеокамеры на основе ПЗС (приборы с зарядовой связью) или КМОП (комплементарная структура металл-оксид-полупроводник) матриц. При этом стали появляться образцы достаточно малых размеров, обладающих маленькими диаметрами входного зрачка объектива объектива – такими, как правило, обладают видеокамеры типа «стеноп» (другое название - «pinhole»). Эти видеокамеры стали использоваться в различных отраслях, но всё чаще их используют для ведения скрытого видеонаблюдения, которое, как правило, несанкционированно. Данная проблема создала потребность в создании приборов для обнаружения скрытых видеокамер. Сейчас существует два типа таких систем: электромагнитные и оптические. В данной статье будет больше уделено действию оптических. Подобные приборы основаны на эффекте световозвращения, или «обратного блика». Все видеокамеры скрытого наблюдения имеют фоточувствительную ПЗС-матрицу, которая расположена в задней фокальной плоскости объектива. Главной проблемой является возникновение большого количества «ложных» целей – бликов, возникающих от других источников. Поэтому возникает необходимость разработки алгоритма, который правильно идентифицирует блики видеокамер, снижая вероятность ошибки.

Научная новизна работы заключается в следующем:

1) Данный алгоритм позволяет автоматически обнаруживать и выделять объекты с помощью рамки, тем самым значительно облегчая обнаружение скрытых оптико-электронных приборов;

2) Данная работа включает практическую реализацию алгоритма на языке C++ с использованием открытой библиотеки OpenCV 4.

# 1 Аналитический обзор современного состояния в области скрытого видеонаблюдения.

С каждым годом область микропроцессорной техники совершенствуется, а вместе с тем возможности записи фото и видео также растёт, увеличивается качество снимаемой картинки, происходит миниатюризация приборов. Фотоприборы и видеоприборы размещают в помещениях, на элементах одежды (таких как, например, часы, ручка, очки), в автомобилях и деталях интерьера (в картинах, в настенных часах). Благодаря этим устройствам злоумышленники имеют шанс вести несанкционированную съёмку с целью получения секретных сведений и дальнейшего шантажа. Однако количество несанкционированных съемок в социальном пространстве сильно увеличивается из-за массового распространения приборов с камерами, а прежде всего из-за смартфонов и планшетов, оснащенных камерами [1]. Любая камера скрытого видеонаблюдения имеет объектив, формирующий изображение наблюдаемого объекта или пространства, а также матричные ПЗС, которые составляют фоточувствительный преобразователь и в зависимости от формата определяют размер ПЗС матрицы. У камер скрытого видеонаблюдения формат ПЗС матрицы должен быть достаточно маленьким из-за миниатюрного размера самого устройства.

Камеры малого формата должны иметь короткофокусный объектив для увеличения угла обзора. Качество формируемого изображения определяется светосилой объектива, зависящего от фокусного расстояния f’ и заднего фокального зрачка Dвх. Многие камеры скрытого видеонаблюдения характеризуются [2, 3]:

* чувствительностью, позволяющей различать объекты в темноте и при ИК-подсветке до 0,01 лк;
* диаметром входного зрачка объектива в диапазоне 1-2 мм;
* большим углом зрения до 140 градусов;
* качеством изображения с разрешением 720 х 576 пикселей;
* временем работы до 30 часов видеонаблюдения при установке источника питания;

способностью передачи изображения по оптоволокну и радиоканалу.

Камеры скрытого видеонаблюдения обычно идут в комплекте с объективами, у которых входной зрачок выносной. Из-за малых размеров зрачка объектива можно по-разному маскировать камеру: в осветительных приборах, дверных проёмах или предметах интерьера. В объективах типа pinhole плоскость входного зрачка совпадает с плоскостью отверстия диафрагмы, которая вынесена перед передней линзой объектива [4]. При расположении диафрагмирующих объектов в его плоскости или уменьшении апертуры объектива снижается светосила, но не его угол обзора. Зрачок выносится на 0,5-10 мм в зависимости от конструкции объектива. Маскировка камеры для скрытого наблюдения обеспечивается за счёт установки на указанном расстоянии перед передней линзой сеток, щелей или маленьких отверстий.

Вид некоторых камер, которые можно использовать потенциально для ведения скрытого видеонаблюдения, приведены на рисунке 1 [5-8].

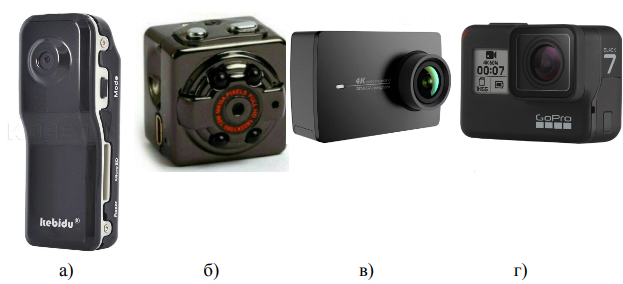


Рисунок 1 - Вид некоторых видеокамер, которые можно использовать потенциально для ведения скрытого видеонаблюдения: а) компания Kebidu, б) компания Ambertek, в) компания Yi, г) компания GoPro

На рисунке 2 приведены замаскированные скрытые видеокамеры[9].



Рисунок 2 - Замаскированные видеокамеры: а) наручные часы с видеокамерой, б) настенные часы с видеокамерой, в) ручка с видеокамерой

Технические характеристики камер, приведённых выше, представлены в таблице 1 [5-8].

Таблица 1 – Технические характеристики камер

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Модель видеокамеры | Разрешение | Минимальная освещенность, лк | Фокусное расстояние объектива, мм | Отношение сигнал/шум |
| Kebidu MD80 | 720 x 480 | 0.3 | 3.7 | 48 Дб |
| Ambertek SQ10 | 1920 x 1080 | 0.1 | 3.7 | 48 Дб |
| Yi 4k | 4000 x 3000 | 0.5 | Сменный объектив | 50 Дб |
| GoPro HERO7 (CHDHX-701-RW) | 4096 x 3072 | 0.5 | Сменный объектив | 45 Дб |

Анализ таблицы 1 показывает, что потенциально пригодные для несакционированной видеозаписи камеры, обладают высокой разрешающей способностью (от 720 x 480 до 4096 x 3072 пикселей), низкой пороговой освещенностью (от 0.1 до 0.5 лк), короткофокусными объективами, которые в зависимости от модели, могут быть на расстоянии от 2.5 до 4 мм.

Короткофокусные объективы, которыми оснащаются камеры скрытого видеонаблюдения, производятся такими компаниями как «HanKwang» (Южная Корея), «Ambertek» (США), «Daiwon» (Южная Koрея), Tokina (Япония) [10-13]. Технические характеристики объективов этих фирм представлены в таблице 2.

Таблица 2 – Характеристики объективов «pinhole»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Наименование | Фокусное расстояние, мм | Относительное отверстие | Диаметр входного зрачка | Задний фокальный отрезок, мм | Поле зрение |
| DW9701 | 3.8 | 1:3.5 | - | 4.12 | 90 |
| DW9702 | 3.4 | 1:3.5 | - | 3.4 | 117 |
| VC-MC380 AHD | 2.8 | 1:2.5 | 1.6 | - | 75 |
| MC600 AHD | 3.0 | 1:2.6 | 1.4 | - | 140 |
| P431380 | 4.3 | 1:3.8 | 1,1 | 4,6 | 78 |
| P321450S | 3.2 | 1:4.5 | 0.72 | 3.4 | 105 |
| TCP0420 | 4.0 | 1:2.0 | 1.8 | 9.19 | 75 |

Анализируя таблицу 2, можно прийти к выводу, что большинство объективов «pinhole», обладают фокусными расстояниями в диапазоне от 2.8 до 9.19 мм, большими угловыми полями зрения до 140 градусов и сравнительно малыми диаметрами входного зрачка до 1.6 мм.

# 2 Аналитический обзор современного состояния в области средств обнаружения камер скрытого видеонаблюдения

Технология, применяемая для обнаружения оптико-электронных приборов активно используется в сфере военной обороны. Наибольшее распространение она получила в технологиях, которые позволяют находить снайперские прицелов и приборов ночного видения. Такие технологии облегчали задачу прицеливания и получения параметров траектории. Также этими приборами пользуются различные спецслужбы для организации безопасности важных персон. Применимость таких устройств выросла благодаря изготовлению ПЗС(прибор с зарядовой связью) и КМОП матриц. В связи с этим появились лазерные диоды, позволяющие получать монохроматические излучения, которые помимо прочего обладали небольшими размерами. Такие системы стали использоваться в том числе для ведения несанкционированной съёмки в кинотеатрах. Детекторы скрытых камер на сегодняшний день делятся на два типа: электромагнитные и оптические.

2.1 Электромагнитные детекторы скрытых камер

Данный тип детекторов сначала обследует помещение на предмет электромагнитных волн, а затем при нахождении каких-либо частот сравнивает их с занесёнными в память образцами. Более детальный анализ происходит во время разбивки полосы спектра. В зависимости от частоты осциллятора видеокамеры нужный спектр находится в некотором промежутке, который ищется путём увеличения чувствительности детектора. Затем детектор в зависимости от частоты принимает решение, является ли частотой процессора видеокамеры текущая частота или это помеха. Чтобы избавиться от помех, приборы могут делать несколько итераций (доходит до четырёх) для проверки и подтверждения, подозрительных участков, а затем результат о принадлежности частоты процессора видеокамере поступает пользователю. На рисунке 3 приведены некоторые модели таких устройств[14, 15].



Рисунок 3 – Электромагнитные детекторы

Технические характеристики данных устройств приведены в таблице 3.

Таблица 3 – Технические характеристики электромагнитных детекторов

|  |  |  |
| --- | --- | --- |
| Модель детектора | Амулет | Аркам (SEL C-200ВТ) |
| Габариты | 235x230x70 мм | 85x140x30 мм |
| Радиус обнаружения | До 10 м | До 15 м |
| Время детектирования | До 40 с | До 13 с |

В зависимости от внесённых в память детектора типов видеокамер время поиска может существенно отличаться. Благодаря различным видам индикации (звуковая, световая, вибрационная) данный тип детекторов позволяет вести поиск видеокамер практически незаметно, однако стоит заметить, что у некоторых моделей есть антенна, чувствительность которой при прилегании к человеческому телу снижается, поэтому проводить обследование рекомендуется, держа прибор в вытянутой руке.

2.2 Оптические детекторы скрытых камер

Данный тип детекторов основан на эффекте «обратного блика». Этот эффект возникает благодаря тому, что в фокальной плоскости любой камеры находится светоотражающий элемент. В камерах скрытого наблюдения эту роль выполняет ПЗС матрица. Таким образом камера выдаёт обратный блик в поле своего зрения при её подсвечивании. Описанный принцип изображён на рисунке 4.

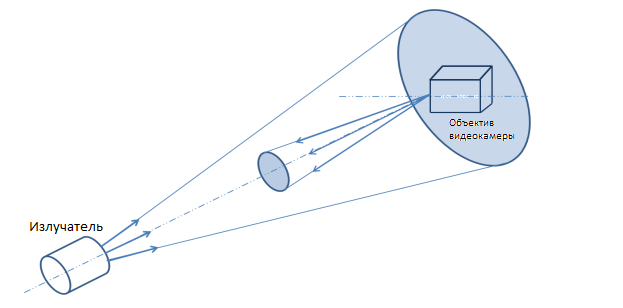


Рисунок 4 – Эффект «обратного блика»

Проблема этого метода заключается в возникновении «ложных» бликов от диффузных отражателей, интенсивность которых бывает иногда даже больше «полезных» бликов. Именно поэтому в процессе разработки алгоритма обнаружения камер скрытого наблюдения необходимо учитывать блики отражённые от диффузно-отражающих поверхностей. Также не обойтись без расчёта оптимальных настроек: алгоритм должен рассчитывать время накопления ПЗС матрицы и длительность импульса лазерной подсветки.

На рисунке 5 [16-18] представлены модели приборов для обнаружения камер скрытого видеонаблюдения, которые можно приобрести в свободной продаже.



Рисунок 5 - модели приборов для обнаружения камер скрытого видеонаблюдения: а) Гранат-2, б) WEGA-M, в) СС308+

В таблице 4 представлены характеристики моделей приборов обнаружения камер скрытого видеонаблюдения, представленных выше.

Таблица 4 – Технические характеристики оптических детекторов

|  |  |  |  |
| --- | --- | --- | --- |
| Модель детектора | Гранат-2 | WEGA-M | СС308+ |
| Габарит | 91х75х50 | 140х34х16 | 93х48х17 |
| Дальность обнаружения | До 25м | до 20м | до 10м |
| Длина волны | 625/525нм | 920нм | 920нм |
| Угол подсветки | 18° | 24° | 30° |
| Угол поля зрения | 10° | 16° | 25° |
| Производитель | Россия | Украина | Китай |
| Режим | Ручной | Ручной | Ручной |

Устройства из таблицы 4 имеют малые габариты, а также сравнительно низкую стоимость. Тем не менее у них присутствуют следующие недостатки:

* отсутствие автоматизированного режима (человеческий фактор может негативно сказаться на обнаружении цели),
* светодиодная подсветка (ограничивает максимальную дистанцию обнаружения),
* узкая направленность излучения.

Представленные устройства позволяют наблюдать обратный блик, используя подсветку с определённой частотой. Во время подсвечивания цели пользователь, смотря в глазок, в котором установлена ИК линза, анализирует получаемую картин и самостоятельно выявляет скрытые видеокамеры. В некоторых моделях присутствует режим вибрации, который должен способствовать нахождению скрытых устройств: во время наведения на потенциальные источники бликов прибор начинает вибрировать. Во время опыта использования данная функция показала себя плохо: прибор реагирует абсолютно на все источники бликов, которые содержаться в окружающем пространстве в большом количестве, и вибрация больше отвлекает от поставленной задачи. Из-за малого контраста изображения блика на контрастном фоне обнаружение бликов часто затрудняется. В свободной продаже отсутствуют приборы с автоматизированным режимом, который базировался бы на методах и алгоритмах цифровой обработки изображения.

2.3 Выводы по второй главе

Во второй главе сделан краткий обзор устройств, которые могут быть пригодны для ведения скрытого видеонаблюдения. Даны сведения основных параметров «pinhole» камер: фокусное расстояние, относительный размер отверстия, диаметр входного зрачка, задний фокальный отрезок, поле зрения. Сделан краткий обзор устройств, позволяющих находить камеры скрытого видеонаблюдения: даны основные характеристики и описан их принцип работы. Всего существует два типа таких устройств: электромагнитные и оптические. Электромагнитные довольно хорошо справляются с задачей выявления скрытых устройств и из недостатков было выявлено только зависимость от внесённых в память частот процессоров видеокамер и возможные внешние помехи. Из анализа представленных на рынке оптических устройств был сделан вывод, что отсутствие автоматизированного поиска скрытых видеокамер делает разработку алгоритма для этой цели актуальной.

# 3 Разработка алгоритма по автоматическому определению и фиксации скрытых камер видеонаблюдения

3.1 Программное обеспечение и использованные материалы

Для реализации было решено использовать открытую библиотеку компьютерного зрения OpenCV версии 4, поскольку в ней на данный момент реализованы множество алгоритмов обработки изображений, которые планируются использоваться в данной работе. Программа будет написана на языке программирования C++, так как данный язык поддерживается библиотекой OpenCV, а сам язык является компилируемым, что поможет улучшить производительность программы. Для написания используется актуальная среда программирования Microsoft Visual Studio 2019.

Специально для проведения экспериментов, была создана установка, с помощью которой были получены кадры, которые используются в работе алгоритма.

3.2 Алгоритм программы. Предварительная подготовка кадров

Данный этап служит подготовительным и одинаково полезен для дальнейшей обработки, поскольку существует вариативность, на основе которой производится оценка алгоритмов друг с другом.

На вход алгоритма поступает два изображения: «активный» (используется подсветка пространства) и «пассивный» кадр (без подсветки). «Активный» кадр содержит искомые объекты, которые соответствуют световозвращённому блику и диффузному отражению излучения. К каждому из них применяется сегментация MeanShift. Конечной целью является получение разностного кадра: вычитание пассивного из активного. Основные этапы подготовки представлены на рисунке 6.

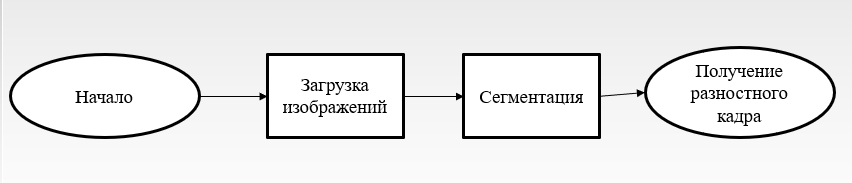


Рисунок 6 – Схема алгоритма подготовки кадров для дальнейшей обработки

### 3.2.1 Предварительная подготовка кадров. Сегментация

В данном случае, порядок использования важен: получить разностный кадр нужно, предварительно использовав сегментацию. Это сильно влияет, если в дальнейшем будет применяться пороговая фильтрация, то при одних и тех же порогах, будет портиться геометрия отражённых объектов. Поскольку MeanShift группирует объекты с близкими признаками, то пиксели со схожими признаками объединяются в один сегмент, на выходе получается изображение с однородными областями и в дальнейшем это помогает при вычислении площади найденных бликов, тем самым фильтруются ложные блики. Изображение после сегментирования MeanShift представлено на рисунке 7.



Рисунок 7 – Применение сегментации MeanShift на «активном» кадре

### 3.2.2 Предварительная подготовка кадров. Вычисление разностного кадра

Данный этап алгоритма позволяет исключить из последующей обработки отражённое фоновое излучение от объектов, находящихся в зоне подсвета лазера.

Последовательно поступающие кадры Xk(m,n) представляют собой цифровые данные о распределении интенсивности по дискретным координатам m и n в видеокадре под номером k. При поступлении каждого второго кадра осуществляется подсветка пространства. Таким образом, кадры делятся на два типа: на «активные» XАk(m, n) = X2k+1(m, n) и «пассивные» XПk(m, n) = X2k(m, n). «Активные» кадры содержат изображения фона, локационное изображение (включающее блики от обнаруженных скрытых камер) и изображения диффузно-отражающих элементов пространства. «Пассивные» кадры несут в себе информацию только о фоне. Соседние кадры различаются в шумовой реализации Rk(m, n), которая прибавляется к изображению Ek(m, n). Матрица разности ΔX(m, n) «активного» и «пассивного» кадров содержит данные о локационном сигнале и шумовом поле.

Тогда можно записать следующие выражения:

Xk(m, n) = Ek(m, n) + Rk(m, n),(1)

ΔX(m, n) = XAk(m, n) – XПk(m, n) (2)

где ΔX *-* разностный кадр,

XA - "активный" кадр,

XП - "пассивный" кадр.

Эффект при использовании разностного кадра можно увидеть на рисунке 8.

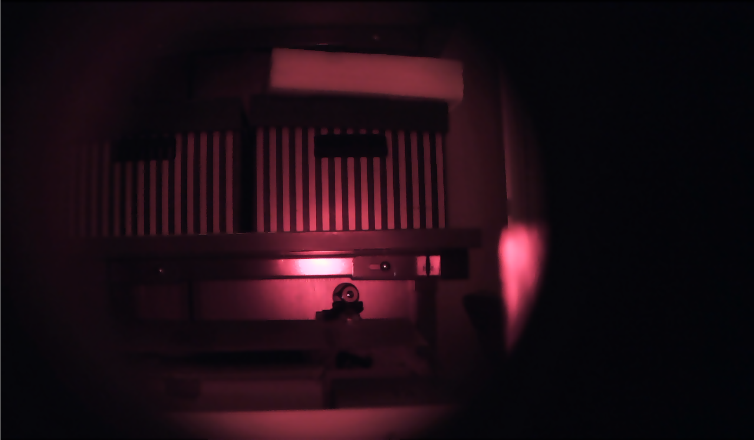


Рисунок 5 – Различие между загруженным изображением (слева) и разностным кадром (справа). Яркость фонового излучения от объектов уменьшена

3.3 Алгоритм программы. Вариативность

Снизив яркость фонового излучения, снижается вероятность обнаружения «ложных» бликов, после чего появляется вариативность методов алгоритма при дальнейшем использовании, которые напрямую влияют на быстродействие и сложность обработки кадра. Началом тут является разностный кадр, а концом – выделение и отрисовка окружностей с наложением на текущий кадр. Схема возможных дальнейших шагов представлена на рисунке 9.



Рисунок 9 – Вариативность алгоритма

Всего здесь представлено три варианта: использование детектора Canny (приложение А), пороговой фильтрации (приложение Б) и HSV модели (приложение В).

### 3.3.1 Вариативность. Canny.

Первым этапом служит цифровая фильтрация входного сигнала, которая преследует две цели:

1) подавление шумовой составляющей *Rk(m, n)* в высокочастотной области спектра за счёт повышения сигнала,

2) повышение помехоустойчивости излучателя (уменьшение вероятности появления «ложных» бликов).

Заметность шума на экране уменьшается повсеместно с подавлением блика светоотражателя при сужении полосы фильтра. Именно поэтому нужно найти наилучшее сочетание типа фильтра с его порядком.

Использование матрицы *h(i, j)* большого порядка (третьего и выше) требуют больших вычислительных мощностей, поэтому стоит применять фильтры 3×3 или 5×5. Для данной задачи приемлемо рассмотреть фильтры двух типов:

1) фильтр Гаусса с дисперсией,

2) усредняющий rect-образный фильтр.

Фильтр Гаусса описывается формулой (3):

где – дисперсия,

i – строка,

j – столбец,

*h(i, j) -* матрица фильтра.

Усредняющий rect-образный фильтр описывается формулой (4):

где L — длина фильтра.

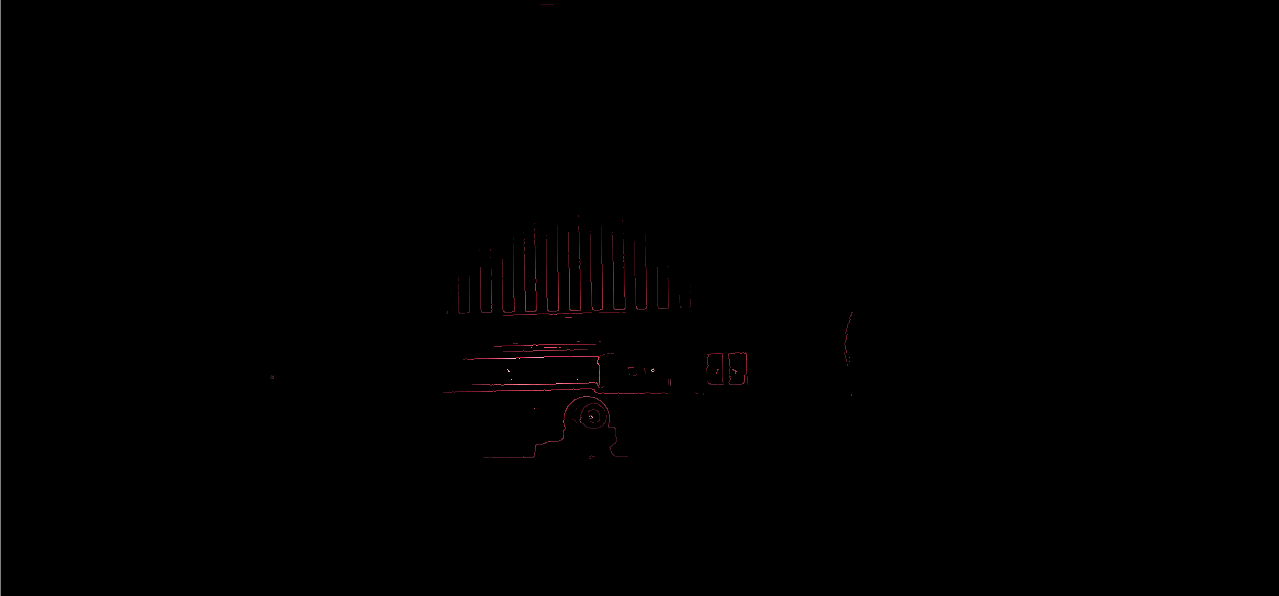
Остальные фильтры могут быть аппроксимированы перечисленными типами. В таблице 4 приведены результаты оценки аппаратно-реализуемого отношения сигнал/шум μ в зависимости от типа и размера фильтра [19].

Таблица 4 - Результаты вычисления отношения сигнал/шум в зависимости от типа и размера цифрового фильтра

|  |  |  |
| --- | --- | --- |
| Тип фильтра | Матрица h(i, j) | μ |
| Усредняющий 3×3 |  | 27.6 |
| Усредняющий 5×5 |  | 25.7 |
| Гауссов, 3×3, *σ =* 0.8 |  | 26.5 |
| Гауссов, 3×3, *σ =* 0.5 |  | 14.6 |

Анализ таблицы 4 показывает, что максимальный эффект отношения сигнал/шум достигается при использовании усредняющего фильтра 3×3. При дальнейшем сужении полосы пропускания фильтра (усредняющий, 5×5), существенных изменений в отношении сигнал/шум нет, при этом его использование требует больше ресурсов, учитывая это, применение более широкополосного фильтра (гауссова, 3×3, σ = 0,5) снижает отношение сигнал/шум из-за менее эффективного подавления шумов в высокочастотной области спектра.

На втором этапе выделяют границы объектов. Для этого используется Canny Edge Detection с двумя ключевыми параметрами - верхним и нижним порогами. Верхний порог используется для обозначения ребер, которые определенно являются ребрами. Нижний порог - нахождение слабых пикселей, которые могут являться частью ребра. Для этого в каждом пикселе проверяется, является ли он локальным максимумом в его окрестности в направлении градиента. Выставленная верхняя градиентная величина означает, что цвета быстро меняются, что подразумевает грань и помечает её как «сильная». Это как раз является приметой бликующих объектов. Все пиксели, которые меньше верхнего порога, но больше нижнего, помечаются как «слабая» грань, которые проверяются дополнительно на наличие в окрестности «сильных» пикселей и в случае отсутствия таковых такая грань удаляется. Оставшиеся пиксели образуют результат, изображённый на рисунке 10.

Рисунок 10 - Выделение границ объекта

На рисунке 11 показано, как с помощью функции polylines происходит отрисовка замеченных граней камер функцией findCounters на наложением на текущий (активный) кадр. Нужные контуры находятся по их площади, параметры которой регулируются специально написанной функцией.

Рисунок 11 - Выделение и отрисовка окружностей

Как можно заметить, тут достаточно много ложных бликов. Одним из недостатков этого метода является сравнительно большой ряд параметров, которые требуют регулировки. Помимо настройки нижнего и верхнего порогов, в некоторых случаях может применяться морфология для улучшения результата, тем самым увеличивая параметры регулировки и используя дополнительные ресурсы.

### 3.3.2 Вариативность. Пороговое преобразование.

Данный метод применяется для повышения обнаружительной характеристики алгоритма. Пороговое преобразование выделяет объекты из окружающего фона, когда яркость пикселей объектов и фона сосредоточены вблизи двух преобладающих значений. Выделение объектов происходит путем определения значения порога, разделяющего области распределения яркостей. Если значение яркости определенного пикселя больше порогового значения, то это точка принадлежит объекту, если меньше — фону. «Полезные» блики от световозвращающих объектов не меняют свою площадь в зависимости от мощности излучения, в отличие от «ложных» [20]. На данном этапе, стояла задача подобрать пороги, при которых зрачок камер не исчезал и был годен для дальнейшей обработки.

Следующим этапом является использование морфологии. Размыкание позволяет сгладить границы объектов, избавившись от мелких выступов и зубцов. Замыкание позволит объединить близкорасположенные на изображении объекты, что увеличит площадь "ложных" бликов от диффузно-отражающих объектов, и позволит их отфильтровать на следующих этапах алгоритма. Морфологическое размыкание и замыкание представлено на рисунке 12.



Рисунок 12 – Морфологическое размыкание (слева) и замыкание (справа)

Этот этап является опциональным, поскольку при размыкании точно также могут пропасть и полезные блики, а при замыкании, как видно на рисунке 12 (справа), зрачок вебкамеры может слиться с остальными деталями. Важным параметром на данном этапе является выбор размера ядра: площадь бликов от световозвращающих объектов мала, поэтому размер ядра должен тоже не должен быть большим. В ходе экспериментов, наиболее оптимальными размерами являются: для размыкания 2х2, а для замыкания 3х3. В дальнейшем используется, как и для варианта с использованием детектора Canny, используется поиск по площади блика. Результат использования варианта с использованием пороговой фильтрации и морфологии представлен на рисунке 16.

Также как и для варианта с использованием детектора Canny, можно использовать усредняющий фильтр для уменьшения шумов, но использование данной операции нецелесообразно, поскольку в этом случае портится геометрия и отследить «полезные» блики становится сложнее, также это дополнительная операция, которая отрицательно влияет на производительность. При разумном использовании морфологии, можно добиться приблизительно равных результатов.

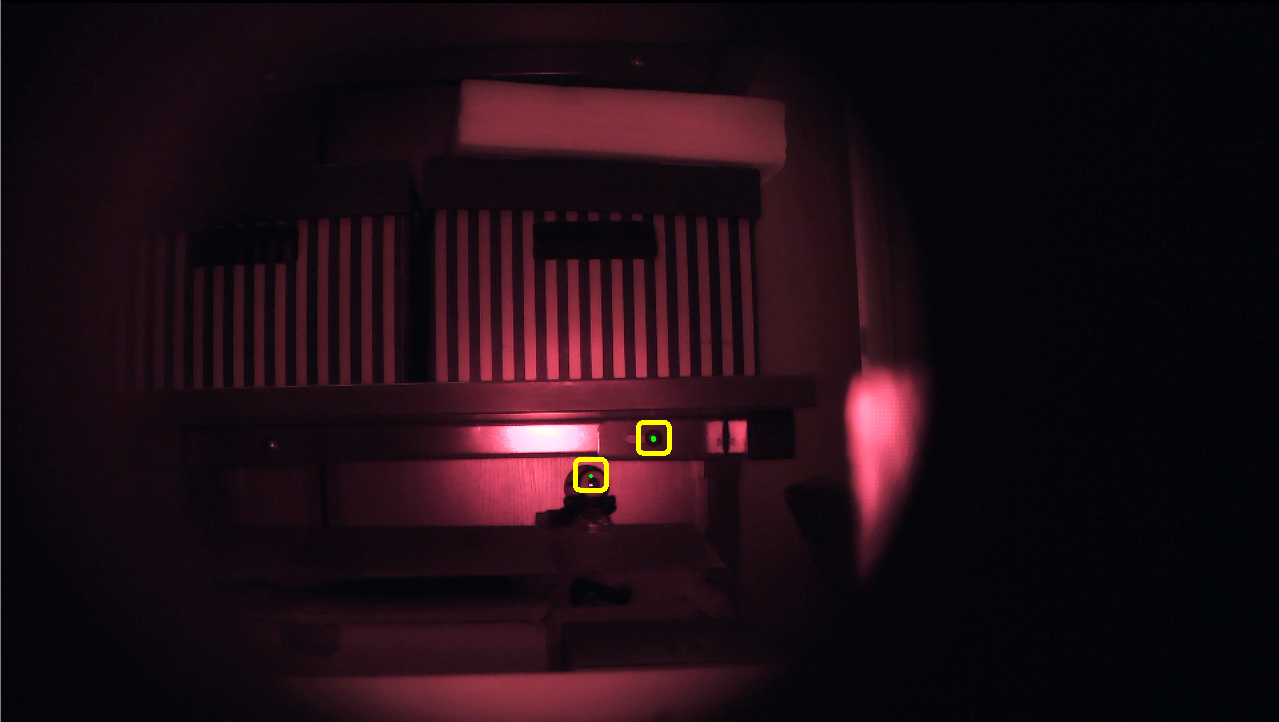


Рисунок 13 – Результат работы алгоритма с использованием порогового преобразования

Использование порогового преобразование в меру эффективно, но у этого способа есть недостатки в виде трудности настройки, но наиболее существенным недостатком является использование одного канала с признаками при определении. Обычное пороговое преобразование не учитывает разницу освещённости. Оттенки серого будут менять свой оттенок в зависимости от освещения, но в данном случае цвета искажены и не зависят линейно от освещения.

### 3.3.3 Вариативность. Модель HSV

HSV — цветовая модель, в которой координатами цвета являются:

* Hue — цветовой тон. Варьируется в пределах 0—360;
* Saturation — насыщенность. Варьируется в пределах 0—100;
* Value (значение цвета) или Brightness — яркость. Также задаётся в пределах 0—100 и 0—1;

Также как и при использовании порогового преобразования, оптимальная настройка параметров в модели HSV даёт хорошие результаты, которые приведены на рисунке 17.

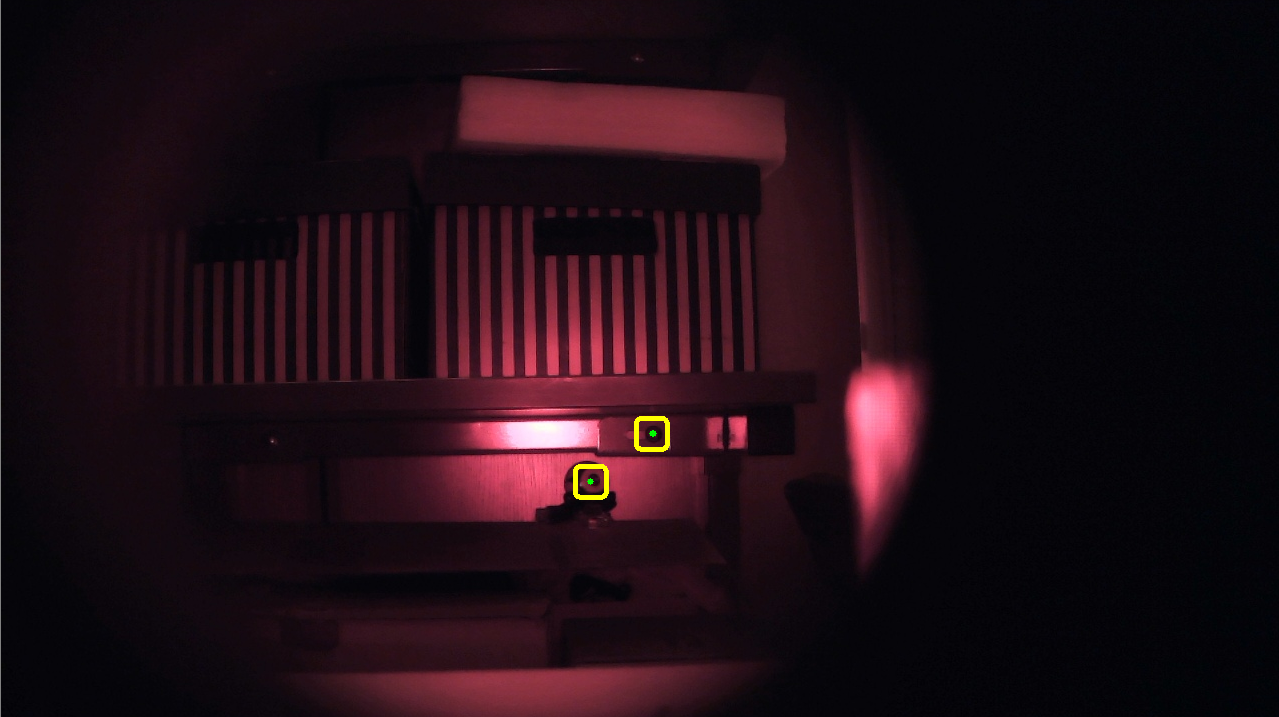


Рисунок 14 – Результат работы алгоритма с модели HSV с наложением на активный кадр

HSV трудно настроить, но предположительно, возможно задать настройки для системы при различных параметрах помещения (например, освещения) и пользоваться ими в зависимости от ситуации. Плюсом данного метода можно назвать использование параметра value, который отвечает за яркость, что является важным признаком при поиске блика от камеры. В данном примере программа справляется даже без применения функции поиска площади, но для надёжности работы алгоритма, она крайне желательна.

3.4 Сравнение алгоритмов

В таблице 5 приведено сравнение алгоритмов по трём характеристикам: количестве операций, скорости.

Таблица 5 – Сравнение алгоритмов

|  |  |  |
| --- | --- | --- |
| Алгоритм | Время исполнения, мс | Количество настраиваемых параметров |
| Canny | 453.215 | 5-6 |
| Пороговое преобразование | 438.272 | 4-6 |
| HSV | 230.141 | 6-8 |

Анализ таблицы 5 показывает, что лучшее быстродействие показало использование модели HSV, но у этого алгоритма наибольшее количество минимальных (обязательных) настраиваемых параметров, при этом сам алгоритм качественно справляется с задачей детектирования. Самое минимальное количество настраиваемых параметров у алгоритма пороговой фильтрации, ввиду вероятного использования морфологии, данный алгоритм нельзя отнести к простым, хотя он также качественно справился с поставленной задачей в эксперименте. Алгоритм с использованием детектора Canny показал худший результат и не справился в полной мере с поставленной задачей: было найдено слишком много «ложных» бликов.

3.5 Алгоритм программы. Видеопоток

Программа, обрабатывающая видеопоток, состоит из трёх частей: заголовочный файл (приложение Г), калибровочный файл (приложение Д), основная часть (приложение Е).

Главная цель данного этапа состоит в том, чтобы оптимизировать обработку кадров видеопотока, посредством создание класса с несколькими потоками, каждый из которых выполняет только одну (или несколько) функций обработки изображений, тем самым получая высокую частоту кадров в сочетании с большим разрешением изображения.  
Считывание кадра с камеры обычно занимает много времени, поэтому важно иметь свой собственный поток, чтобы получить максимально возможную частоту кадров при максимально возможном размере видео. Для этой цели каждый поток имеет свой буфер кадров, который дополнительно помогает отсеивать «ложные» блики.

На рисунке 15 представлена визуализация последовательности обработки изображений через три потока:

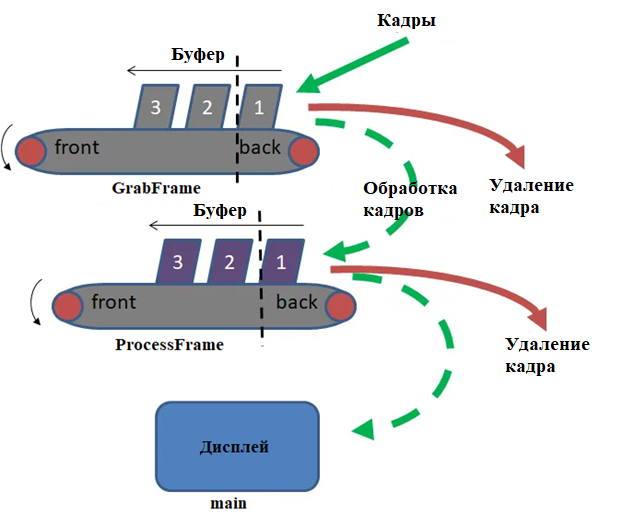


Рисунок 15 – Обработка изображений через три потока

В калибровочном файле объявляется конструктор и деструктор класса Camera, также объявляется несколько переменных, которые содержат последние захваченные кадры. VideoCapture объявляется вне функции класса: это позволяет получить доступ к нему из любой точки класса.

В основной части программы, для получения максимально возможной частоты кадров при использовании некоторых функций OpenCV, применяется несколько потоков.  
Помимо главной части, эта часть кода содержит содержит три функции:

* GrabFrame
* ProcessFrame
* ResultImage

Некоторые важные переменные объявлены глобальными в этом коде:

* cam1 - является экземпляром класса камеры и должен быть доступен из main и GrabFrame.
* frame\_buffer - является указанным максимальным пределом кадров, которые могут быть помещены в стек, прежде чем весь стек будет удален.
* frame\_stack – стек кадров, содержащий кадры, считанные с камеры (до кадров frame\_buffer).
* flare\_stack - стек кадров, содержащий кадры, взятые из стека frame\_stack и обработанные с использованием функций OpenCV.
* stop\_sig - указывает, что при значении 1 все потоки останавливаются и возвращаются в основную процедуру.

3.6 Выводы по третьей главе

Во второй главе описаны алгоритмы обработки изображения для селекции световозвращающих бликов с целью определения скрытых видеокамер. Алгоритм условно разделён на подготовительную и основную части.

В подготовительной части применяется сегментация (делает группирует пиксели со схожими признаками) и вычитание кадров для получения разностного кадра, которое исключает помехи от излучателя.

В основном этапе представлены три варианта алгоритма для дальнейшего анализа, где применялись: Canny Edge Detector, пороговое преобразование и модель HSV.

Сравнение показало, что лучшим быстродействием обладает алгоритм с использованием модели HSV, а самое минимальное (обязательное) количество настраиваемых параметров у алгоритма пороговой фильтрации. Оба этих алгоритма справляются с задачей обнаружения «полезных» бликов и решают проблему «ложных» бликов. Алгоритм с использованием детектора Canny показал худший результат из-за слишком большого количества найденных «ложных» бликов.

Создан класс для обработки видеопотока, состоящих из трёх файлов: заголовочного, калибровочного и основного. Его цель: автоматически подстраиваеться под конфигурацию системы, а также реализует буфер кадров, который выполняет функцию распределения нагрузки при обработке кадров и дополнительно отсеивает ложные блики.

# ЗАКЛЮЧЕНИЕ

В результате данной работы, был проведён обзор миниатюрных камер, которые существуют сейчас на рынке, в том числе и типа «pinhole», чтобы дать общие представления о камерах и их параметрах, на которые делается упор при создании таких приборов. Также был сделан обзор существующих в мире разработок в области обнаружения скрытых видеокамер. Обзор дал общие понятия о существующих приборах и выявил, что в свободной продаже отсутствуют оптические приборы с автоматизированным поиском скрытых камер, а также осветил главную проблему для реализации такого устройства – «ложные» блики, тем самым обосновав актуальность разработки такого алгоритма.

Алгоритм условно разделён на подготовительную и основную части. Рассматривалось три варианта (Canny, пороговое преобразование, модель HSV) основной части создания алгоритма. В приоритет ставилось использование функций, которые содержатся в стандартной библиотеке OpenCV; качество обнаружения «полезных» бликов; скорость работы; количество настраиваемых параметров. При создании использовались знания об элементарных свойств изображений.

Сравнение показало, что лучшим быстродействием обладает алгоритм с использованием модели HSV, а самое минимальное (обязательное) количество настраиваемых параметров у алгоритма пороговой фильтрации. Оба этих алгоритма справляются с задачей обнаружения «полезных» бликов и решают проблему «ложных» бликов.

Программа была оптимизирована для видеопотока: создан класс, который автоматически подстраивается под конфигурацию системы, а также реализует буфер кадров, который выполняет функцию распределения нагрузки при обработке кадров и дополнительно отсеивает ложные блики.

Рекомендации к дальнейшей работе следующие: при фильтрации на основе анализа формы и размеров объектов, допустимый размер блика должен меняться в зависимости от расстояния детектора до блика (используя дальномер), а также автоматически подбирать параметры алгоритма в зависимости от освещённости помещения и учитывать параметры подобранной конфигурации детектора.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. W. Choung. The Rational Regulation of Illegal & Harmful Information in Cyberspace // KIC. 2005. Vol.29, C.9-68.
2. Pin-hole video camera: a.c. 6335759 B1 U.S. / N. Harada.
3. ChumdanSa. Miniature CCD camera functions and instructions // Automation systems. 1987. Vol.7, C.126-133.
4. M. Young. Pin-hole Optics // Opt. Soc. America. 1971. Vol.10, P.2763-2767.
5. ООО – Kebidu – CCTV cameras - Miniature type // http://www.kebidu.com/en/html/product/product\_search.php?searchstr=MD80 (дата обращения 20.11.2019)
6. OOO – Ambertek Inc. – Miniature type camera // http://www.ambertek.ru/produkcija-kompanii-ambertek#sq10 (дата обращения 20.11.2019)
7. OOO – Yi Technology – Action camera // http://www.yitechnology.ru/yi-4k-plus-action-camera-specs (дата обращения 20.11.2019)
8. OOO – GoPro – Action camera // https://gopro.ru/product/kamera-gopro-hero7-black-edition-chdhx-701-rw (дата обращения 20.11.2019)
9. Cкрытые миниатюрные камеры // http://ru.aliexpress.com/popular/lighter- 164 spy-cam.html (дата обращения 20.11.2019)
10. OOO – HanKwang Opto – Pin-hole lenses Р431380, Р321450S, P321450S // www://optical.hkopto.com (дата обращения 20.11.2019)
11. OOO – Daiwon – Pin-hole lenses // http://www.dwopt.com/shopsinye\_eng/shopping\_mall.php?code=000003&lv=1& mode=list (дата обращения 20.11.2019)
12. OOO – Ambertek Inc. – Miniature type camera // http://shop.ambertek.ru/collection/besprovodnye-kamery (дата обращения 20.11.2019)
13. OOO – Tokina – Pin-hole lens // http://www.tokina.co.jp/security/news/-tcp0420-tcp0420dc-1.html (дата обращения 20.11.2019)
14. Прибор обнаружения, камер скрытого видеонаблюдения, Амулет // http://www.spectr-sks.ru/product/20802 (дата обращения 20.11.2019)
15. Прибор обнаружения, камер скрытого видеонаблюдения, SEL C-200BT // https://suritel.ru/catalog/all/apparatura-poiska-kanalov-utechki/hidden-cameras-detectors/hidden-cameras-detector-sel-c-200bt-arkam/ (дата обращения 20.11.2019)
16. Прибор обнаружения, камер скрытого видеонаблюдения, СС308 // http://www.chinaetop.com/sdp/528584/4/pd-2782949/11110284 2607856/CC308\_camera\_lends\_detector.html (дата обращения 20.11.2019)
17. Прибор обнаружения, камер скрытого видеонаблюдения, WEGA-M // https://plux.in.ua/index.php?route=product/product&product\_id=261 (дата обращения 20.11.2019)
18. Прибор обнаружения, камер скрытого видеонаблюдения, ГРАНАТ-2 // http://taskt.ru/catalog/apparatura-nabljudenija-i-kontrolja/optiko-elektronnye-pribory/granat-2/?sphrase\_id=3153 (дата обращения 20.11.2019)
19. Бокшанский В.Б.,Карасик В.Е.,Таранов М.А. Автоматическое обнаружение световозвращателей с помощью лазерных локационных систем // Вестник МГТУ. Сер.Приборостроение. 2011. №2. C.25-35.
20. Construction project management office : Multiplex design -2009 // http://www.hmglobal.com/uploads/cm/%EB%B3%B5%ED%95%A9%EC%98%81%EC%83%81%EC%8B%9C%EC%84%A4COP.pdf (дата обращения 20.11.2019)

# ПРИЛОЖЕНИЕ

### Приложение А. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Вариант с использованием детектора Canny.

#include <iostream>

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

using namespace cv;

using namespace std;

Mat img\_Canny, current\_frame;

void ResultImage(int mini, int maxi, Vec3b color, int n);

int main()

{

Mat img\_active = imread("./example diff/img\_active.jpg"); // "активный" кадр (текущий!)

if (img\_active.empty()) // Проверка на ошибки (загружено изображение или нет)

{

cout << "Could not open or find the image" << endl;

return -1;

}

Mat img\_passive = imread("./example diff/img\_passive.jpg"); // "пассивный" кадр

if (img\_passive.empty()) // Проверка на ошибки (загружено изображение или нет)

{

cout << "Could not open or find the image" << endl;

return -1;

}

/// Получение разностного изображения (картинки должны быть одного разрешения)

Mat diff\_img;

absdiff(img\_active, img\_passive, diff\_img);

imshow("diff\_img", diff\_img); // проверка

/// Получение изображения с оттенками серого

Mat img\_gray;

cvtColor(diff\_img, img\_gray, COLOR\_BGR2GRAY);

/// Фильтрация кадра

Mat img\_blurred;

medianBlur(img\_gray, img\_blurred, 3);

imshow("img\_blurred", img\_blurred);

/// Задание порогов и применение детектора Canny

int lowTh = 100;

int highTh = 100;

Canny(img\_blurred, img\_Canny, lowTh, highTh);

namedWindow("img\_Canny", WINDOW\_NORMAL);

imshow("img\_Canny", img\_Canny); // проверка

/// Придание цвета краям

Mat dst;

dst = Scalar::all(0);

diff\_img.copyTo(dst, img\_Canny);

current\_frame = img\_active.clone();

namedWindow("dst", WINDOW\_NORMAL);

imshow("dst", dst);

ResultImage(3, 11, Vec3b(0, 255, 0), 0); // Задание площади

imshow("draw\_frame", current\_frame);

waitKey(0);

destroyAllWindows(); // Уничтожение всех окон

return(0);

}

void ResultImage(int mini, int maxi, Vec3b color, int n)

{

Moments mnts, imnts, lmnts;

vector<vector<Point>> cnts;

// Поиск контуров объектов

findContours(img\_Canny, cnts, RETR\_EXTERNAL, CHAIN\_APPROX\_NONE);

double d = 0;

double minum = 1000;

/// Проход по всем контурам

for (int i = 0; i < cnts.size(); i++)

{

// Если размер контура не меньше минимума и не превышает максимум, то рисуем его и находим его центр масс

if (cnts[i].size() > mini&& cnts[i].size() < maxi)

{

// Находим момент контура

mnts = moments(cnts[i]);

if (n == 0)

{

d = sqrt((lmnts.m10 / lmnts.m00 - mnts.m10 / mnts.m00) \* (lmnts.m10 / lmnts.m00 - mnts.m10 / mnts.m00) +

(lmnts.m01 / lmnts.m00 - mnts.m01 / mnts.m00) \* (lmnts.m01 / lmnts.m00 - mnts.m01 / mnts.m00));

if (minum > d)

{

// Обновление максимального расстояния

minum = d;

imnts = mnts;

}

}

else if (n == 1)

{

lmnts = mnts; // для запоминания

imnts = mnts; // для рисования

}

polylines(current\_frame, cnts[i], true, color, 2, 8); // Рисование контура

}

}

circle(current\_frame, Point(imnts.m10 / imnts.m00, imnts.m01 / imnts.m00), 5, Vec3b(0, 0, 0), -1);

}

### Приложение Б. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Вариант с использованием пороговой фильтрации.

#include "opencv2/imgproc.hpp"

#include "opencv2/highgui.hpp"

#include <iostream>

using namespace cv;

using namespace std;

Mat close\_img, current\_frame;

void ResultImage(int mini, int maxi, Vec3b color, int n);

int main()

{

/// Загрузка изображений

Mat img\_active = imread("./example diff/img\_active.jpg"); // активный кадр

if (img\_active.empty())

{

cout << "Could not open or find the image!\n" << endl;

return -1;

}

Mat img\_passive = imread("./example diff/img\_passive.jpg"); // пассивный кадр

if (img\_passive.empty())

{

cout << "Could not open or find the image!\n" << endl;

return -1;

}

// Сегментация MeanShift

Mat image\_MeanShift\_active, image\_MeanShift\_passive;

int spatial\_radius = 1; //очень важный параметр (влияет на скорость работы и на выделение участков)

int color\_radius = 25; //25 - уровень, при котором находятся камеры

int pyramid\_levels = 1; // при увеличении выделяет больше областей, появляются мелкие шумы

TermCriteria term\_criteria{};

term\_criteria.maxCount = 5; // как ни меняй - разницы нет

term\_criteria.epsilon = 1; // как ни меняй - разницы нет

term\_criteria.type = TermCriteria::COUNT + TermCriteria::EPS;

pyrMeanShiftFiltering(img\_active, image\_MeanShift\_active, spatial\_radius, color\_radius, pyramid\_levels, term\_criteria); // пассивный кадр

imshow("MeanShift\_1", image\_MeanShift\_active); //проверка

pyrMeanShiftFiltering(img\_passive, image\_MeanShift\_passive, spatial\_radius, color\_radius, pyramid\_levels, term\_criteria); // активный кадр

imshow("MeanShift\_2", image\_MeanShift\_passive); //проверка

/// Получение разностного изображения (картинки должны быть одного разрешения)

Mat diff\_img;

absdiff(image\_MeanShift\_active, image\_MeanShift\_passive, diff\_img);

imshow("diff\_img", diff\_img); // проверка

/// Получение изображения с оттенками серого

Mat img\_gray;

cvtColor(diff\_img, img\_gray, COLOR\_BGR2GRAY);

/// Пороговое преобразование

Mat threshold\_val\_1;

threshold(img\_gray, threshold\_val\_1, 50, 255, THRESH\_BINARY); // более высокий порог, при котором теряется геометрия

imshow("threshold\_val\_1", threshold\_val\_1); //проверка

/// Морфологическое открытие (опционально, меняет геометрию)

Mat open\_img;

Mat kernel\_1 = getStructuringElement(MORPH\_RECT, Size(2, 2));

morphologyEx(threshold\_val\_1, open\_img, MORPH\_OPEN, kernel\_1);

imshow("Open", open\_img); //проверка

/// Морфологическое закрытие (опционально, меняет геометрию)

Mat kernel\_2 = getStructuringElement(MORPH\_RECT, Size(3, 3));

morphologyEx(open\_img, close\_img, MORPH\_CLOSE, kernel\_2);

imshow("Close", close\_img); //проверка

/// Фильтрация кадра (опционально, меняет геометрию)

//Mat img\_blurred;

//medianBlur(img\_gray, img\_blurred, 3);

current\_frame = img\_active.clone();

ResultImage(3, 10, Vec3b(0, 255, 0), 0); // Задание площади. 9-11 на кадрах Варлашина; 1-7 для видео с Ютуба

imshow("draw\_frame", current\_frame);

waitKey(0);

destroyAllWindows(); // Уничтожение всех окон

return 0;

}

void ResultImage(int mini, int maxi, Vec3b color, int n)

{

Moments mnts, imnts, lmnts;

vector<vector<Point>> cnts;

/// Поиск контуров объектов

findContours(close\_img, cnts, RETR\_EXTERNAL, CHAIN\_APPROX\_NONE);

double d = 0;

double minum = 1000;

/// Проход по всем контурам

for (int i = 0; i < cnts.size(); i++)

{

/// Если размер контура не меньше минимума и не превышает максимум, то рисуем его и находим его центр масс

if (cnts[i].size() > mini&& cnts[i].size() < maxi)

{

mnts = moments(cnts[i]);

if (n == 0)

{

d = sqrt((lmnts.m10 / lmnts.m00 - mnts.m10 / mnts.m00) \* (lmnts.m10 / lmnts.m00 - mnts.m10 / mnts.m00) +

(lmnts.m01 / lmnts.m00 - mnts.m01 / mnts.m00) \* (lmnts.m01 / lmnts.m00 - mnts.m01 / mnts.m00));

if (minum > d)

{

minum = d;

imnts = mnts;

}

}

else if (n == 1)

{

lmnts = mnts; // для запоминания

imnts = mnts; // для рисования

}

polylines(current\_frame, cnts[i], true, color, 2, 8); // Рисование контура

}

}

circle(current\_frame, Point(imnts.m10 / imnts.m00, imnts.m01 / imnts.m00), 5, Vec3b(0, 0, 0), -1);

}

### Приложение В. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Вариант с использованием модели HSV.

#include "opencv2/imgproc.hpp"

#include "opencv2/highgui.hpp"

#include <iostream>

using namespace cv;

using namespace std;

Mat current\_frame, mask;

void ResultImage(Vec3b color);

int main()

{

/// Загрузка изображений

Mat img\_active = imread("./example diff/img\_active.jpg"); // "Актиный" кадр

if (img\_active.empty())

{

cout << "Could not open or find the image!\n" << endl;

return -1;

}

Mat img\_passive = imread("./example diff/img\_passive.jpg"); // "Пассивный" кадр

if (img\_passive.empty())

{

cout << "Could not open or find the image!\n" << endl;

return -1;

}

/// Сегментация MeanShift

Mat img\_MeanShift\_active, img\_MeanShift\_passive;

int spatial\_radius = 1; //очень важный параметр (влияет на скорость работы и на выделение участков)

int color\_radius = 25; //25 - уровень, при котором находятся камеры

int pyramid\_levels = 1;

TermCriteria term\_criteria{};

term\_criteria.maxCount = 5;

term\_criteria.epsilon = 1;

term\_criteria.type = TermCriteria::COUNT + TermCriteria::EPS;

pyrMeanShiftFiltering(img\_active, img\_MeanShift\_active, spatial\_radius, color\_radius, pyramid\_levels, term\_criteria);

imshow("img\_MeanShift\_active", img\_MeanShift\_active); //проверка

pyrMeanShiftFiltering(img\_passive, img\_MeanShift\_passive, spatial\_radius, color\_radius, pyramid\_levels, term\_criteria);

imshow("img\_MeanShift\_passive", img\_MeanShift\_passive); //проверка

/// Получение разностного изображения (картинки должны быть одного разрешения)

Mat diff\_img;

absdiff(img\_MeanShift\_active, img\_MeanShift\_passive, diff\_img);

imshow("diff\_img", diff\_img); // проверка

/// Задание границ HSV

inRange(diff\_img, Scalar(68, 145, 0), Scalar(180, 171, 215), mask);

current\_frame = img\_active.clone(); // Текущий кадр, для рисования обнаруженных бликов

ResultImage(Vec3b(0, 255, 0)); // Задание площади (в варианте для HSV не используется, но её возможно внедрить)

imshow("draw\_frame", current\_frame);

waitKey(0);

destroyAllWindows(); // Уничтожение всех окон

return 0;

}

/// Поиск контуров объектов

void ResultImage(Vec3b color)

{

Moments mnts, imnts, lmnts;

vector<vector<Point>> cnts;

findContours(mask, cnts, RETR\_EXTERNAL, CHAIN\_APPROX\_NONE);

drawContours(current\_frame, cnts, -1, Scalar(0, 255, 0), 6);

}

### Приложение Г. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Видеопоток. Заголовочный файл.

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/opencv.hpp"

#include <iostream>

#include <vector>

#include <thread>

using namespace cv;

using namespace std;

class Camera

{

public:

Camera(void);

~Camera(void);

Mat captureVideo(void);

private:

Mat frame;

double width;

double height;

double fps;

};

### Приложение Д. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Видеопоток. Калибровочный файл.

#include "camLib.h"

using namespace std;

VideoCapture cap(0); // Получение видеопотка

/// Конструктор

Camera::Camera(void)

{

//Проверка работоспособности открытия камеры

cout << " Camera warming up..." << endl;

int isrunning = 0;

if (!cap.isOpened()) // Если не удалось, выйдите из программы

{

cout << "Cannot open the video cam" << endl;

}

else

{

isrunning = 1;

}

if (isrunning == 0)

{

cout << "Camara did not start up - Exiting..." << endl;

this->~Camera();

}

/// Автоматическое определение размера камеры

cap.set(CAP\_PROP\_FRAME\_WIDTH, 1920);

cap.set(CAP\_PROP\_FRAME\_HEIGHT, 1080);

cap.set(CAP\_PROP\_FPS, 30);

width = cap.get(CAP\_PROP\_FRAME\_WIDTH); // Получение ширины кадров видео

height = cap.get(CAP\_PROP\_FRAME\_HEIGHT); // Получение высоты кадров видео

fps = cap.get(CAP\_PROP\_FPS); // Получение количество кадров в секунду с устройства

cout << "Frame size : " << width << " x " << height << " --- fps: " << fps << endl;

}

/// Деструктор

Camera::~Camera(void)

{

cout << "Shutting down camera and closing files..." << endl;

cap.release();

}

/// Функция доступа к камере

Mat Camera::captureVideo(void)

{

cap >> frame;

return frame;

}

### Приложение Е. Алгоритм по автоматическому обнаружению и фиксации камер видеонаблюдения. Видеопоток. Основной файл.

#include "camLib.h"

using namespace cv;

using namespace std;

/// Выделение памяти для хранения кадров и запуска камеры

Camera cam1;

const int frame\_buffer = 50; // Максимальный предел кадров, которые могут быть помещены в стек, прежде чем весь стек будет удален

vector<Mat> frame\_stack = \*new vector<Mat>[frame\_buffer \* sizeof(cam1.captureVideo())]; // Cтек кадров, содержащий кадры, считанные с камеры (до кадров frame\_buffer)

vector<Mat> flare\_stack = \*new vector<Mat>[frame\_buffer \* sizeof(cam1.captureVideo())]; // Cтек кадров, взятых из стека frame\_stack и обработанные с использованием функций OpenCV

int stop\_sig = 0; // Глобальный сигнал остановки (при значении 1, все потоки останавливаются и возвращаются в основную процедуру)

Mat current\_frame, mask;

void ProcessFrame(void);

void GrabFrame(void);

void ResultImage(Vec3b color);

int main()

{

Mat frame; // Захваченные отдельные кадры

Mat contour; // Видеопоток, показывающий счетчики объектов

// Запуск бесконечного цикла для захвата кадров

// Этот бесконечный цикл останавливается нажатием клавиши ESC

frame\_stack.clear();

flare\_stack.clear();

thread t1(GrabFrame);

thread t2(ProcessFrame);

while (1)

{

if (flare\_stack.size() >= 2)

{

contour = flare\_stack.back();

imshow("Contour Video", contour);

}

if (waitKey(1) == 30) // Ожидание нажатия клавиши «esc» в течение 30 мс. Если нажата клавиша «esc» - цикл обрывается

{

cout << "Main: esc key is pressed by user" << endl;

stop\_sig = 1; // Сигнал потокам, чтобы закончить их запуск

frame\_stack.clear();

flare\_stack.clear();

break;

}

}

t1.join();

t2.join();

return 0;

}

void ProcessFrame(void)

{

Mat frame;

Mat contour;

Mat frame\_prev;

char charCheckForEscKey = 0;

while (charCheckForEscKey != 27)

{

/// Проверка - есть ли данные в буфере кадров. Если стек кадров не пустой, захватывается новый кадр, не удаляя его для дальнейшей обработки

if (!frame\_stack.empty())

{

frame = frame\_stack.front(); // Взятие первого кадра из исходного стека без его удаления

frame\_prev = Mat::zeros(frame.size(), frame.type()); // Нулевой кадр (чёрный кадр)

/// Получение разностного кадра

Mat frameAbsDiff;

absdiff(frame, frame\_prev, frameAbsDiff); // без знака

//imshow("frameAbsDiff", frameAbsDiff);

if (waitKey(90) == 27)

break;

frame.copyTo(frame\_prev);

/// HSV

Mat HSV\_frame;

cvtColor(frameAbsDiff, HSV\_frame, COLOR\_BGR2HSV); // Преобразование из BGR в HSV

//imshow("HSV\_frame", HSV\_frame);

/// Задание границ HSV

inRange(HSV\_frame, Scalar(132, 107, 170), Scalar(150, 110, 200), mask); // test2 video

current\_frame = frame.clone(); // Текущий кадр, для рисования обнаруженных бликов

ResultImage(Vec3b(0, 255, 0)); // Поиск по площади

imshow("draw\_frame", current\_frame);

charCheckForEscKey = waitKey(1); // delay and get key press

}

/// Если в стеке бликов более 2 кадров, последний кадр удаляется (в конце стека)

if (flare\_stack.size() > 2)

{

flare\_stack.pop\_back();

}

/// Если доступен новый обработанный кадр и стек еще не заполнен, новый обработанный кадр помещается вперёд стека:

if (!contour.empty() && flare\_stack.size() < frame\_buffer)

{

flare\_stack.push\_back(contour);

}

/// Очистка стека, только если он заполнен

else if (flare\_stack.size() >= frame\_buffer)

{

flare\_stack.clear();

}

}

cout << "process\_frame: esc key is pressed by user" << endl;

return;

}

void GrabFrame(void)

{

Mat frame;

frame\_stack.clear();

while (!stop\_sig)

{

frame = cam1.captureVideo(); // Захват кадра с камеры

/// Если в стеке более 2 кадров - удалить один кадр сзади

if (frame\_stack.size() > 2)

{

frame\_stack.pop\_back(); // Эта строка удаляет последний кадр из стека

}

/// Если стек не заполнен, добавить кадр в начало стека

if (frame\_stack.size() < frame\_buffer)

{

frame\_stack.push\_back(frame); // Эта строка помещает кадр за кадром в конец стека

}

/// Очистка стека, когда он заполнен

else

{

frame\_stack.clear();

}

}

cout << "grabFrame: esc key is pressed by user" << endl;

return;

}

/// Поиск контуров объектов, нахождение площади

void ResultImage(Vec3b color)

{

vector<vector<Point>> contours;

findContours(mask, contours, RETR\_LIST, CHAIN\_APPROX\_SIMPLE);

vector<vector<Point>> contours\_poly(contours.size());

for (unsigned int i = 0; i < contours\_poly.size(); i++)

{

approxPolyDP(contours[i], contours\_poly[i], 1, true);

cout << "# of contour points: " << contours\_poly[i].size() << endl; // Количество найденных бликов

for (unsigned int j = 0; j < contours\_poly[i].size(); j++)

{

cout << "Point(x,y)=" << contours\_poly[i][j] << endl; // Вывод координат обнаруженных бликов

}

if (contourArea(contours\_poly[i]) < 1) // Задание размера площади, меньше которой будет рисоваться в кадре

{

cout << " Area: " << contourArea(contours\_poly[i]) << endl;

if ((contours\_poly[i].size()) < 2)

{

drawContours(current\_frame, contours, -1, Scalar(0, 255, 0), 6); // Нарисовать найденные блики

/// Создание рамки

Rect rect = boundingRect(contours[i]);

Point pt1, pt2;

pt1.x = rect.x - 20;

pt1.y = rect.y - 20;

pt2.x = rect.x + 20;

pt2.y = rect.y + 20;

rectangle(current\_frame, pt1, pt2, Scalar(0, 255, 255));

}

}

}

}