МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра геоинформатики и информационной безопасности

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«ПОВЫШЕНИЕ КАЧЕСТВА ВИДЕО ДЛЯ ЗАДАЧ КРИМИНАЛИСТИЧЕСКОЙ ЭКСПЕРТИЗЫ»

по направлению подготовки 10.05.03 Информационная безопасность автоматизированных систем

(уровень академического специалитета)

направленность (профиль) «Обеспечение информационной безопасности распределенных информационных систем»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обучающийся |  |  |  | Цой Г.В. |
|  | (подпись) | | |  |
| Руководитель ВКР,  профессор, д.т.н. |  |  |  | Сергеев В.В. |
|  | (подпись) | | |  |
| Рецензент |  |  |  |  |
|  | (подпись) | | |  |
| Нормоконтролёр |  |  |  | Клевцова Е.В. |
|  | (подпись) | | |  |

Самара, 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра геоинформатики и информационной безопасности

УТВЕРЖДАЮ

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / В.В. Сергеев /

*(подпись) И.О.Фамилия*

« 25 »\_\_\_\_января\_\_\_\_2021 г.

**Задание на выпускную квалификационную работу (ВКР)**

|  |  |  |
| --- | --- | --- |
| Обучающемуся | | Цою Глебу Владимировичу |
|  | | *(ФИО, полностью)* |
| группы | 6511-100503D | |

1. Тема ВКР: «Повышение качества видео для задач криминалистической экспертизы»

утверждена приказом по университету от 25.01.2021 г. № 42-Т

2. Перечень вопросов, подлежащих разработке в ВКР:

|  |
| --- |
| 1) разработка метода повышения качества видео; |
| 2) сравнительное исследование и выбор алгоритма геометрического |
| согласования кадров видеопоследовательности; |
| 3) реализация метода в виде программного обеспечения; |
| 4) исследование эффективности разработанного метода на видеозаписях, |
| типичных для задач криминалистической экспертизы. |

3.Консультанты по разделам ВКР *(при наличии):* −

раздел ВКР:\_\_\_ −\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

разрабатываемые вопросы: −

|  |
| --- |
|  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

*должность, степень подпись И.О.Фамилия*

4.Дата выдачи задания: «15» февраля 2021 г.

5.Срок представления на кафедру законченной ВКР: «10» июня 2021 г.

|  |  |  |
| --- | --- | --- |
| Руководитель ВКР  профессор, д.т.н. |  | /В.В. Сергеев/ |
|  | *подпись* | *И.О.Фамилия* |
| Задание принял к исполнению |  | /Г.В. Цой/ |
|  | *подпись студента* | *И.О.Фамилия* |

Реферат

Пояснительная записка: 118 c., 43 рисунка, 6 таблиц, 44 источников, 9 приложений.

Презентация: 15 слайдов презентации Power Point.

КРИМИНАЛИСТИЧЕСКАЯ ЭКСПЕРТИЗА, СВЕРХРАЗРЕШЕНИЕ, ГЕОМЕТРИЧЕСКОЕ СОГЛАСОВАНИЕ, ОПТИМАЛЬНОЕ КОМПЛЕКСИРОВАНИЕ

Объектом исследования являются изображения низкого пространственного разрешения, искаженные под воздействием аппаратных ограничений техники.

Целью работы является разработка оптимального метода повышения качества изображений по серии кадров низкого качества.

В результате работы разработан программный комплекс, позволяющий повышать пространственное разрешение изображений с использованием серии кадров видеозаписи низкого качества.

Эффективность работы заключается в использовании оценок дисперсии интерполяции в каждой точке видеокадров для формирования весовых коэффициентов на этапе комплексирования набора изображений.

Содержание

[Введение 7](#_Toc73572880)

[1 Описание разработанного метода 10](#_Toc73572881)

[1.1 Общее описание метода 10](#_Toc73572882)

[1.2 Учащение сетки дискретизации кадров и интерполяция изображения 11](#_Toc73572883)

[1.3 Формирование дополнительного канала обработки 12](#_Toc73572884)

[1.4 Геометрическое согласование 15](#_Toc73572885)

[1.4.1 Методы, основанные на детектировании особых точек 16](#_Toc73572886)

[1.4.2 Методы, основанные на пирамидальном подходе 17](#_Toc73572887)

[1.4.3 Методы, основанные на вычислении пиков взаимной корреляционной функции 18](#_Toc73572888)

[1.4.4 Метод, основанный на оптическом потоке 19](#_Toc73572889)

[1.5 Оптимальное комплексирование 19](#_Toc73572890)

[2 Описание программной реализации 23](#_Toc73572891)

[2.1 Модуль автоматизации экспериментальных исследований геометрических согласований 24](#_Toc73572892)

[2.2 Модуль графического интерфейса 26](#_Toc73572893)

[2.2.1 Модуль предобработки 35](#_Toc73572894)

[2.2.2 Модуль интерполяции 37](#_Toc73572895)

[2.2.3 Модуль вычисления ошибки интерполяции 37](#_Toc73572896)

[2.2.4 Модуль геометрического согласования 38](#_Toc73572897)

[2.2.5 Модуль комплексирования 39](#_Toc73572898)

[2.3 Модуль автоматизации экспериментальных исследований невзвешенного комплексирования 40](#_Toc73572899)

[2.4 Описание работы программной реализации 41](#_Toc73572900)

[3 Экспериментальное исследование 48](#_Toc73572901)

[3.1 Типичные для криминалистической экспертизы видео данные 48](#_Toc73572902)

[3.2 Сравнительное исследование методов геометрического согласования 51](#_Toc73572903)

[3.3 Исследование эффективности разработанного метода сверхразрешения 53](#_Toc73572904)

[Заключение 63](#_Toc73572905)

[Список использованных источников 65](#_Toc73572906)

[Приложение А 70](#_Toc73572907)

[Приложение Б 94](#_Toc73572908)

[Приложение В 101](#_Toc73572909)

[Приложение Г 106](#_Toc73572910)

[Приложение Д 107](#_Toc73572911)

[Приложение Е 108](#_Toc73572912)

[Приложение Ж 109](#_Toc73572913)

[Приложение И 111](#_Toc73572914)

[Приложение К 117](#_Toc73572915)

Введение

В настоящее время абсолютно каждой сферы деятельности современного общества касается проблема преступности и правонарушений. Изучением, выявлением закономерностей, судебным исследованием доказательств, оценкой и предотвращением преступлений занимается специальная наука, называемая криминалистикой [1]. Криминалистическое исследование особенно полезно [2]:

1. при расследовании инцидентов информационной безопасности;
2. при расследовании некомпьютерных преступлений;
3. при реагировании на инциденты информационной безопасности.

При проведении криминалистической экспертизы зачастую требуется установить задействованные в инциденте действующие лица, их портретные изображения или сведения о их внешнем виде в момент инцидента. Также помимо действующих лиц бывает необходимо зафиксировать параметры конкретных объектов, например, номерные знаки транспортных средств, марки и модели автомобилей, средства регулирования дорожного движения, элементы окружения пространства, текстовые данные с печатных носителей или матриц экранов и многое другое.

Такие данные возможно зафиксировать с помощью видеокамер. Современная техника способна фиксировать изображения с качеством, достаточным для определения экспертом интересуемых характеристик объекта, будь то действующее лицо, автомобильный номер или переписка злоумышленников.

Тем не менее изображения могут быть подвержены непреднамеренным искажениям в результате аппаратных ограничений используемой техники. Бывают ситуации, когда данные, приходящие на экспертизу, обладают низким качеством, и по наблюдаемым данным эксперт не может определить характеристики интересуемого объекта.

Существует класс методов восстановления изображений способных повышать пространственное разрешение кадра с использованием различных технологий. Такие методы называются сверхразрешением. Существующие методы можно проклассифицировать по различным характеристикам.

Методы сверхразрешения можно проклассифицировать по количеству используемых кадров, необходимых для восстановления. Существуют методы, использующие как единственный кадр, так и серию кадров видеопоследовательности.

Также методы классифицируются по применяемым технологиям. Выделяют методы на основе искусственного интеллекта [3], вейвлет-преобразования [4], проекции на выпуклые множества [5], адаптивной фильтрации [6], а также методов преобразований Фурье [7].

Подавляющее большинство методов способны визуально улучшать качество, принимая за меру качества коэффициент различимости объектов [8]. Для задач криминалистики такой подход недопустим, так как в результате обработки изображений могут возникать несуществующие в реальности элементы, называемые артефактами [9].

Целью данной работы является разработка оптимального метода повышения пространственного разрешения изображения по серии кадров низкого разрешения, а также его программная реализация. Оптимизация будет обеспечиваться минимизацией среднеквадратичной ошибки (СКО) для серии последовательных кадров видеопоследовательности с использованием подхода, основанного на преобразованиях Фурье.

Для достижения цели были поставлены следующие задачи: сравнительное исследование и выбор алгоритма геометрического согласования кадров видеопоследовательности, реализация метода в виде программного обеспечения, исследование эффективности разработанного метода на видеозаписях, типичных для задач криминалистической экспертизы.

Разрабатываемый метод сможет применяться не только в задачах криминалистики, а также в таких областях, как медицина, компьютерное зрение, обработка данных дистанционного зондирования Земли, астрономия, микроскопия и других [10-15].

1. Описание разработанного метода

Современные средства видеозаписи фиксируют кадры с некоторыми искажениями, которые препятствуют дальнейшей обработке полученных кадров. Изображение, фиксируемое на матрице видеокамеры, подвергается динамическим искажениям в непрерывной временной области, дискретизируется по времени, также происходит наложение шумов. Таким образом, для наблюдения доступен только дискретный выходной сигнал [16].

Задача сверхразрешения – избавиться от искажений, вызванных аппаратными несовершенствами техники, и приблизиться к восстановлению дискретного изображения до непрерывного.

Необходимо учесть, что при компьютерной обработке сигналов невозможно обрабатывать сигналы в непрерывном виде, но можно преобразовывать сигнал в квазинепрерывный, то есть в дискретный, но с меньшим шагом дискретизации. Идея разрабатываемого метода заключается в том, что изображение преобразуется в квазинепрерывное путём учащения сетки дискретизации.

* 1. Общее описание метода

В основе метода лежит оптимальное комплексирование серии изображений. Основной принцип работы предлагаемого метода состоит в том, что дискретизированные отсчёты изображения, называемые пикселями, дискретизируются с ещё большим шагом. Каждый новый получившийся отсчёт имеет свой уникальный весовой коэффициент и вносит тот или ной вклад в результирующее изображение.

В качестве входного параметра в алгоритм подаётся последовательность кадров. Затем для этих кадров увеличивается сетка пикселей с интерполяцией значений, что обеспечивает сверхразрешающую способность метода. После этого создаётся дополнительный канал обработки информации, в который записываются поля дисперсий ошибок интерполяции кадров. Созданный канал обработки необходим для вычисления коэффициентов взвешенного суммирования. Следом происходит геометрическое согласование кадров с соответствующим преобразованием созданного канала. Согласование обеспечивает минимизацию среднеквадратичной ошибки путём приведения серии кадров к эталонному изображения, взятому из той же выборки кадров. После успешного согласования производится взвешенное суммирование пикселей кадров, называемое комплексированием. Схема алгоритма изображена на рисунке 1:

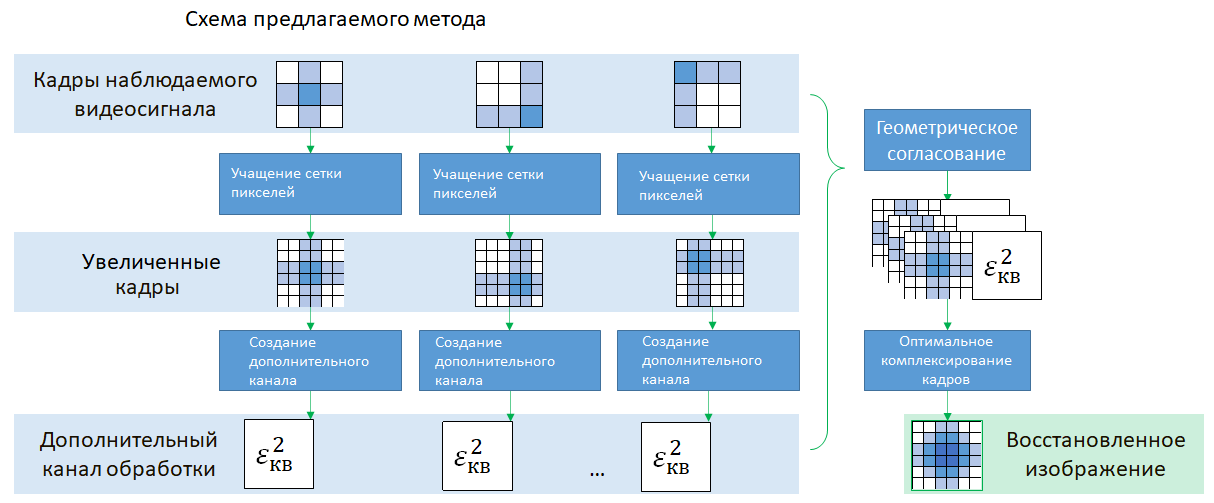


Рисунок 1 - Схема предлагаемого метода

Алгоритм оптимального комплексирования можно разбить на 4 шага:

1. учащение сетки дискретизации кадров и интерполяция изображения;
2. вычисление полей дисперсии ошибок интерполяции;
3. геометрическое согласование серии кадров;
4. оптимальное комплексирование.
   1. Учащение сетки дискретизации кадров и интерполяция изображения

Первым шагом алгоритма является повышение частоты дискретизации кадров видеопоследовательности, что обеспечивает сверхразрешающую способность метода. Увеличить частоту дискретизации возможно разными способами, однако оптимальным считается линейная интерполяция отсчётов. Интерполяцией называют вычисление промежуточных значений сигнала по дискретному набору данных [17]. В рассматриваемой задаче используется линейная интерполяция. Со схемой линейной интерполяции можно ознакомиться на рисунке 2:

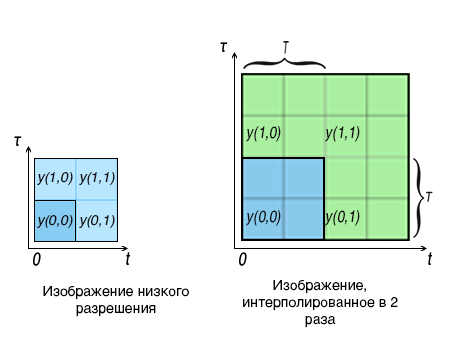


Рисунок 2 - Схема линейной интерполяции

Вычисление значений оценки линейной интерполяции представляется в виде следующей формулы:

(1)

где t,τ – координаты интерполируемого отсчёта;

T– шаг дискретизации изображения;

y(i,j) – значение яркости опорных точек интерполяции.

* 1. Формирование дополнительного канала обработки

После того, как сетки пикселей всех кадров увеличены, необходимо дополнить каждый кадр видеопоследовательности вспомогательным каналом, заполненным дисперсиями ошибок интерполяции. В дальнейшем значения дисперсий будут преобразованы и использованы в итоговой формуле восстановления отсчётов.

При учащении сетки дискретизации возможно получить только некоторую оценку исходной непрерывной сцены, причём оценка определяется с некоторой погрешностью. Эту погрешность для линейной интерполяции можно оценить теоретически. На этапе комплексирования больший вес будет даваться тем отсчетам, ошибка интерполяции которых ниже.

Ошибка интерполяции [16] в точке вычисляется по формуле:

, (2)

где – оценка интерполированного сигнала.

Дисперсия ошибки интерполяции вычисляется по формуле 3:

, (3)

где – оператор математического ожидания.

Подстановка значения формулы 1 в формулу 3 даёт следующее выражение:

, (4)

где

; (5)

; (6)

; (7)

. (8)

При раскрытии скобок значение дисперсии ошибки интерполяции выглядит следующим образом:

. (9)

Воспользуемся определением АКФ для однородного поля:

, (10)

, (11)

где – дисперсия сигнала;

– значение автокорреляционной функции (АКФ);

– безразмерный коэффициаент.

АКФ изображения можно полагать биэкспоненциальной:

, (12)

где – параметр дисперсия сигнала АКФ (>0), который выбирался так, чтобы обеспечить заданное значение коэффициента корреляции сигнала на шаге его первичной дискретизации:

. (13)

Таким образом, с учётом всех замен и приведением слагаемых выражение 9 преобразуется в итоговое выражение 14:

. (14)

* 1. Геометрическое согласование

Геометрическое согласование – это процесс преобразования последовательности кадров таким образом, чтобы интересуемый объект находился на каждом кадре в единой системе координат, соответствующей эталонному изображению [18]. При согласовании одно изображение из последовательности принимается за эталонное. Такое изображение статично на протяжении всего процесса согласования. С эталоном сравниваются последующие кадры последовательности и затем изменяются, причём так, чтобы они максимально точно соответствовали эталонному.

На рисунке 3 можно видеть пример согласования двух кадров:

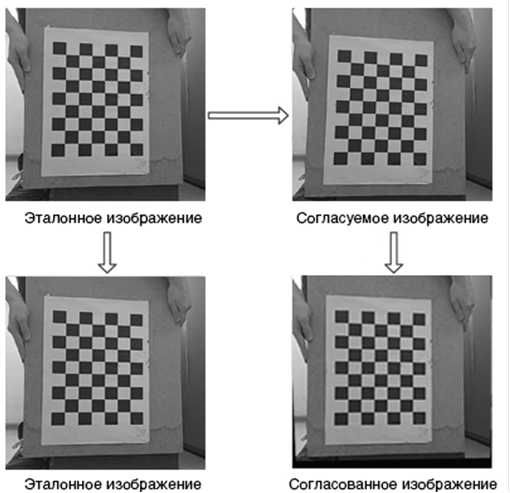


Рисунок 3 – Пример согласования двух кадров видеопоследовательности

На этом шаге производится получение матриц перехода при помощи выбранных методов геометрического согласования. Исходные кадры и поля дисперсии ошибок, созданные на предыдущем шаге, преобразуются в соответствии с матрицами перехода. Ниже рассмотрены наиболее значимые группы методов согласования.

* + 1. Методы, основанные на детектировании особых точек

С помощью детекторов на изображении идентифицируются наиболее значимые точки, и с помощью дескрипторов обеспечивается инвариантность нахождения одних и тех же особых точек относительно преобразований изображений. После того, как определены ключевые точки – происходит их сопоставление на эталонном и согласуемом изображениях. Сопоставление точек сопровождается трансформацией изображения. Существует несколько способов поиска ключевых точек. Наиболее популярными являются:

* алгоритм scale-invariant feature transform (SIFT) [19];
* алгоритм speeded up robust features (SURF) [20];
* алгоритм Binary Robust Independent Elementary Features + Star detector (BRIEF + Star) [21];
* алгоритм Oriented Features from Accelerated Segment Test and rotated BRIEF (ORB) [22].

На изображении 4 можно видеть пример детектирования методом ORB пятидесяти особых точек:

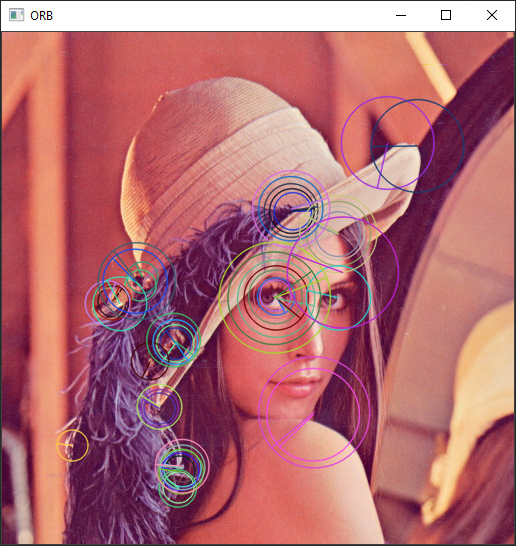


Рисунок 4 – Пример визуализации детектирования особых точек алгоритмом ORB при детектировании 50 особых точек

* + 1. Методы, основанные на пирамидальном подходе

Представление одного и того же изображения в нескольких масштабах называется пирамидой. Каждое изображение называется уровнем. Каждый уровень пирамиды содержит уменьшенную и отфильтрованную копию предыдущего уровня. В качестве фильтрации может применяться размытие по Гауссу. Принцип действия метода пирамид заключается в том, что на некотором уровне пирамиды между эталоном и согласуемым изображениями находится соответствие в точке. Согласуемые изображения трансформируется согласно найденным точкам. Существует пять возможных типов геометрической трансформации, оцениваемых методом пирамидального подхода:

* сдвиг;
* сдвиг и поворот;
* сдвиг, масштаб и поворот;
* аффинное преобразование;
* билинейное преобразование.

На изображении 5 можно видеть пример пирамидальной модели:



Рисунок 5 – Пример визуализации пирамидальной модели с пятью уровнями иерархии

* + 1. Методы, основанные на вычислении пиков взаимной корреляционной функции

Принцип действия методов заключается в применении прямого и обратного преобразований Фурье [23]. Вычислив спектры сигнала, можно получить пиковые значения взаимной корреляционной функции. Полученные функции трансформируются согласно эталонной, что соответственно приводит к трансформации изображений. Существуют разные модификации этого метода, но в основном выделяют следующие:

* основанные на спектрах Фурье;
* основанные на нелинейной оптимизации.

На изображении 6 можно видеть визуальное изображение корреляционных пиков:



Рисунок 6 – Пример визуализации корреляционных пиков

* + 1. Метод, основанный на оптическом потоке

Оптический поток – это векторное поле скоростей движения фрагментов изображения. Суть оптического потока в том, что он не ищет какие-то особенные точки, а по параметрам изображений пытается определить, куда сместилась произвольная точка [24].

На изображении 7 можно видеть визуальное изображение оптического потока между двумя кадрами одной видеопоследовательности:

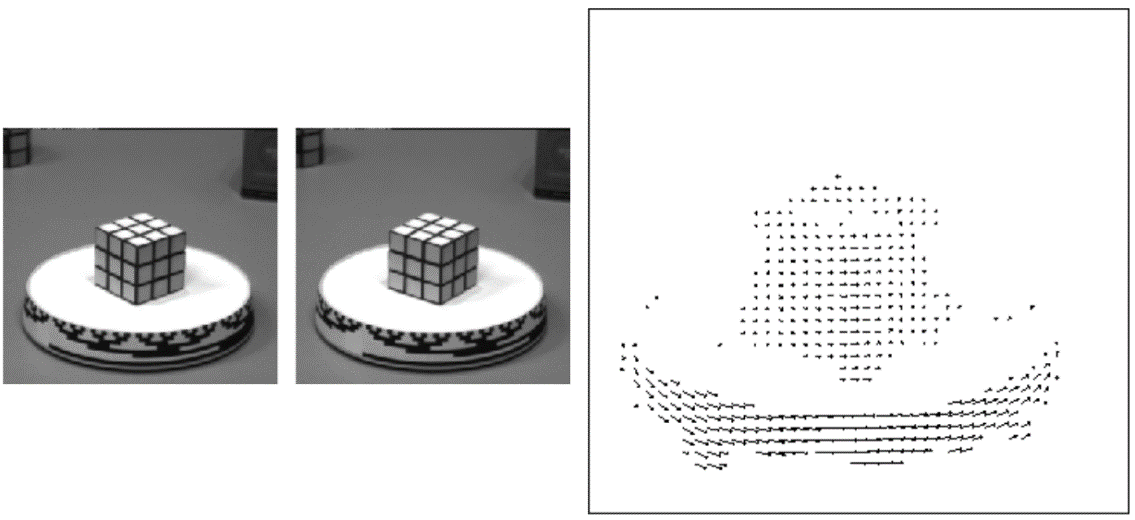


Рисунок 7 – Пример визуализации оптического потока

* 1. Оптимальное комплексирование

На последнем шаге алгоритма происходит оптимальное суммирование неравноточных измерений геометрически согласованного набора изображений.

Так как для наблюдения доступны только искаженные кадры, то необходимо построить линейную оценку, обеспечивающую наибольшую точность в терминах среднеквадратической ошибки. Представим наблюдаемый сигнал в виде формулы 15:

, (15)

где – доступная для наблюдений величина;

– исходная величина, оценку которой необходимо построить;

– шумовая компонента;

– целочисленный аргумент;

– количество доступных наблюдений.

Так как оптимальность обеспечивается минимизацией среднеквадратической ошибки, то в таком случае линейная оценка будет выглядеть следующим образом:

, (16)

, (17)

где – оценка исходной величины по наблюдаемой;

– минимизируемая среднеквадратическая ошибка;

– весовая функция суммирования.

Необходимо рассчитать весовую функцию . Необходимым условием оптимального суммирования является то, что в сумме для каждого отсчёта коэффициенты дают единицу:

. (18)

Другим условием является минимизация среднеквадратической ошибки:

, (19)

где – дисперсия ошибки интерполяции сигнала.

Рассмотрим весовой коэффициент :

. (20)

Продифференцируем выражение 19 для среднеквадратической ошибки:

. (21)

Решив данное уравнение относительно , получаем выражение для весовой функции суммирования:

. (22)

Выражение для среднеквадратической ошибки 17 будет следующим:

. (23)

С учетом полученных результатов, выражение для оценки значения 16 примет следующий вид:

. (24)

Спроецируем полученную формулу на двумерный случай для работы с изображениями:

(25)

где – оценка яркости пикселя в координатах ;

– значение доступной для наблюдения яркости пикселя в координатах каждого -ого кадра последовательности;

– дисперсия ошибки каждого -ого кадра последовательности в координатах ;

– количество пикселей изображения в длину;

– количество пикселей изображения в ширину.

1. Описание программной реализации

Для программной реализации метода необходимо выбрать такой язык программирования, который обеспечит наибольшую эффективность при работе с цифровыми изображениями. Также язык должен поддерживать разработку графического интерфейса.

Было принято решение использовать язык Python 3.9, так как при операциях с двумерными массивами некоторые вычислительно затратные операции возможно векторизовать, что обеспечивает большую скорость работы, а также удобство разработки.

Кроме того, с помощью существующих библиотек для работы с пользовательским интерфейсом становится возможно создать минимально жизнеспособный продукт, пригодный для конечных пользователей.

Программная реализация проходила в три этапа:

1. Разработка автоматизированных модулей сравнения существующих методов геометрического согласования серии кадров. Разработка осуществлена в двенадцати независимых исполняемых файлах. Сравнение существующих методов позволит определить наиболее качественный метод, который в дальнейшем будет использоваться в алгоритме оптимального комплексирования.
2. Разработка пользовательского интерфейса, позволяющего получить изображение большего пространственного разрешения по серии кадров видеопоследовательности. Пользовательский интерфейс включает в себя реализацию модуля графического интерфейса, а также модулей интерполяции, вычисления ошибки интерполяции, геометрического согласования и комплексирования изображений. Каждый модуль представляет собой отдельный исполняемый файл. Работа всех модулей взаимосвязана и начинается с запуска модуля графического интерфейса.
3. Разработка автоматизированных модулей сравнения методов комплексирования серии кадров. Разработка осуществлена в шести исполняемых файлах. Сравнение существующих методов позволит сделать выводы о качестве разработанного метода.

Подробная схема программных модулей представлена на рисунке 8:

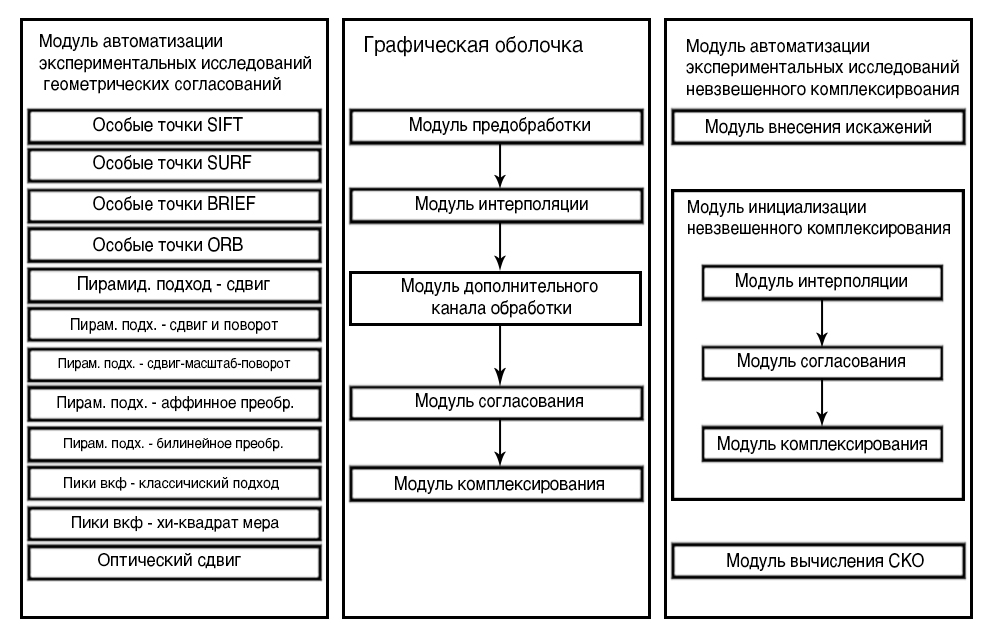


Рисунок 8 – Общая схема ПО

* 1. Модуль автоматизации экспериментальных исследований геометрических согласований

Данный модуль предназначен для автоматизации проведения экспериментов по определению наиболее эффективного метода геометрического согласования изображений. Модуль реализован в виде двенадцати исполняемых файлов, каждый из которых реализует один из двенадцати рассмотренных методов.

Программная реализация аналогична для всех методов: в качестве входного параметра используется последовательная серия кадров одного видео; входные изображения конвертируются и представляются в форме двумерных массивов беззнакового байтового формата; первый кадр последовательности принимается за эталонный; с эталонным изображением согласуются остальные кадры. Пример кадров можно видеть на рисунке 9. Примеры геометрически согласованных кадров – на рисунке 10.

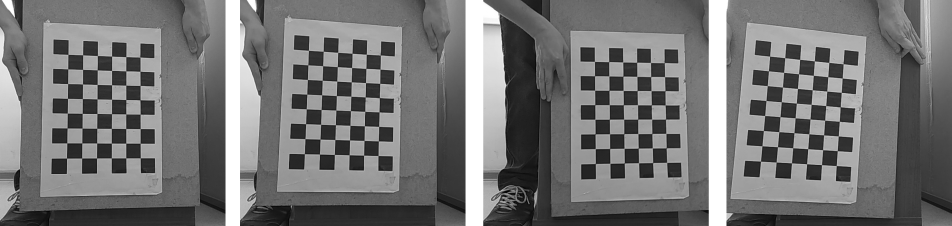


Рисунок 9 – Пример кадров видеопоследовательности

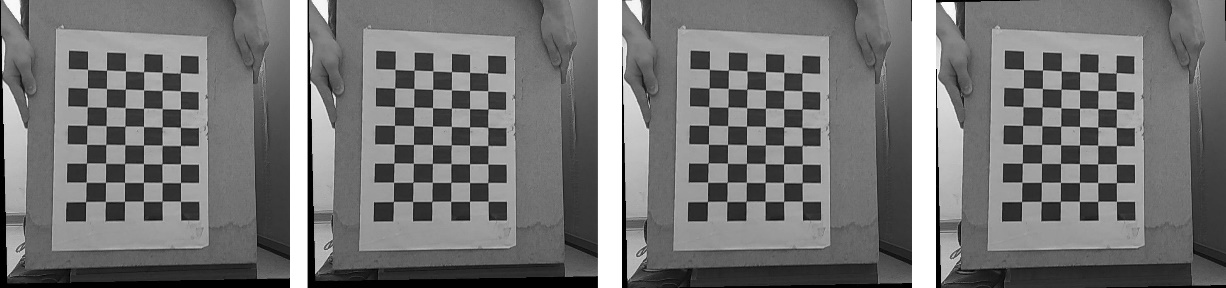


Рисунок 10 – Примеры результатов геометрического согласования методом, основанным на пирамидальном подходе

При разработке данного модуля применялись специфичные функции сторонних библиотек, обеспечивающие более гибкую работу с изображениями в контексте задач геометрического согласования. Программные библиотеки, оптимизирующие работу с геометрическим согласованием: OpenCV [25], pystackreg [26], image\_registration [27], skimage [28], numpy [29].

Полученная согласованная последовательность кадров сравнивается с эталонным кадром. В качестве меры сходства принята среднеквадратическая ошибка между эталоном и обработанным кадром. Расчёт СКО производится по формуле 25.

В результате работы кода генерируются согласованные кадры, а также текстовый файл, содержащий вычисленные значения СКО.

На основе полученных значений СКО было произведено сравнительное исследование наиболее эффективного метода согласования серии кадров. Более подробные результаты исследования представлены в разделе экспериментальных исследований.

Листинг кода данного модуля приведён в приложении А.

* 1. Модуль графического интерфейса

Графический интерфейс разрабатывался с использованием библиотеки «Tkinter» [30]. Визуально программа представляет собой окно 800 на 600 пикселей. При запуске исполняемого файла окно размещается по центру монитора.

В начале работы модуля инициализируются размеры, внешний вид и положение окна на экране, также в файловой директории с исполняемым файлом создаётся новая дочерняя папка под названием «\_RESULTS». Ниже на рисунке 11 представлено изображение окна сразу после запуска исполняемого файла.

Листинг кода данного модуля приведён в приложении Б.



Рисунок 11 – Интерфейс разработанной программы

Ниже подробно описан каждый элемент графической оболочки.

Панель меню 1 содержит выпадающее меню «Инфо», содержащее следующие кнопки:

* Кнопка «О программе», при нажатии на которую выводится новое окно с информацией о приложении, а также кнопка «GITHUB», по нажатию на которую происходит открытие веб-страницы с репозиторием исходного кода программы. Окно с информацией о программе изображено на рисунке 13.
* Кнопка «Инструкция», при нажатии на которую выводится новое окно с подробной инструкцией по работе с приложением. Окно с инструкцией изображено на рисунке 14.
* Кнопка «Показать в папке» при нажатии на которую происходит открытие файловой директории с результатами работы программы, хранящимися в новой созданной папке «\_RESULTS».

На рисунке 12 изображено выпадающее меню панели 1:

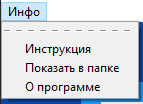


Рисунок 12 – Панель меню

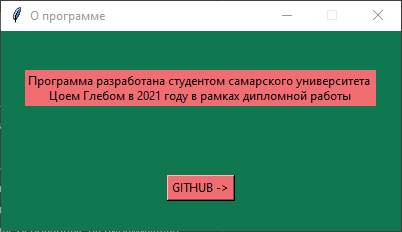


Рисунок 13 – Дополнительное окно с информацией о приложении

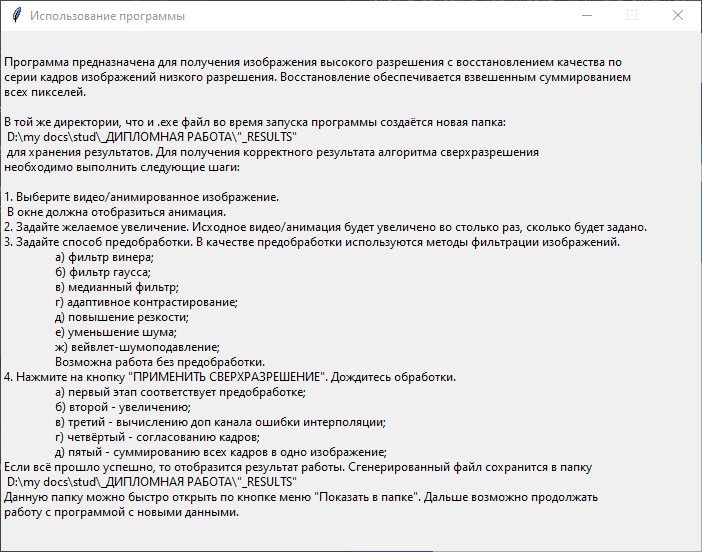


Рисунок 14 – Пример дополнительного окна с инструкцией по применению программы

Поле для ввода текста 2 предназначено для указания расположения одного видеофайла.

Кнопка 3 «Обзор» осуществляет открытие нового диалогового окна для выбора файла из каталогов файловой системы. Для удобства работы реализована функция фильтрации файлов по доступным форматам: «mp4», «wmv», «avi» и «gif». Пример открытого диалогового окна представлен на рисунке 15:

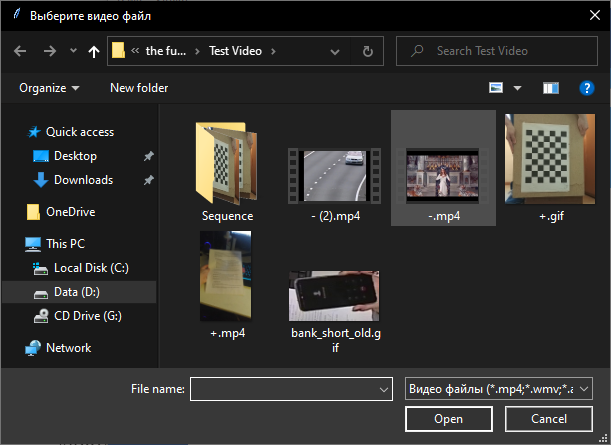


Рисунок 15 – Пример диалогового окна с видео файлами

После выбора файла диалоговое окно закрывается. Поле 2 заполняется локальным путём к выбранному файлу, а пространство 6 воспроизводит анимацию графического файла. Пример работы программы с выбранным файлом показан на рисунке 16:



Рисунок 16 – Пример кадра анимации в интерфейсе программы

Анимация осуществляется при помощи параллельно выполняющегося процесса, который обрабатывает файл по каждому следующему кадру. После того как процесс обработает последний кадр последовательности, анимация прекращается.

В том случае, если пользователем выбран файл неподдерживаемого формата, то в пространстве 6 отображается предупреждающее сообщение: "Файл не найден или формат файла не поддерживается". Пример вывода сообщения представлен на рисунке 17:

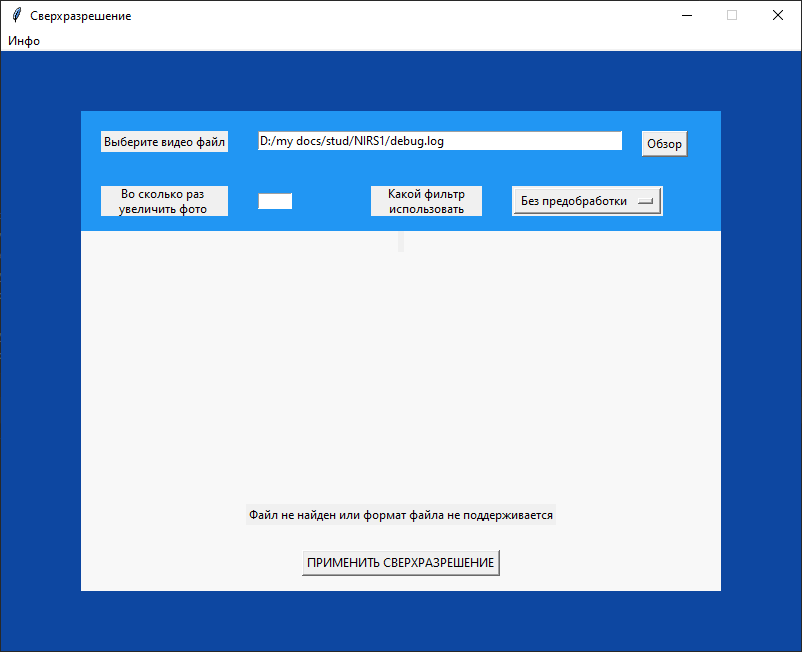


Рисунок 17 – Пример вывода ошибки

Поле ввода 4 предназначено для ввода числовых данных. Поле работает совместно с функцией форматно-логического контроля. Функция осуществляет проверку введённых значений. Согласно этой функции, поле ограничено по вводу не более чем двумя цифрами. При попытке ввода символов, отличающихся от арабских цифр, символы не будут заноситься в поле. Введённое в поле число соответствует коэффициенту, во сколько раз будут интерполированы входные кадры во время работы алгоритма комплексирования.

Выпадающее меню 5 содержит девять активных элементов. Выбранный элемент характеризует дальнейший способ предобработки изображений.

* элемент «Без предобработки» соответствует пропуску этапа предобработки;
* элемент «Винер» соответствует обработке изображения фильтром Винера [31];
* элемент «Гаусс» соответствует обработке изображения фильтром Гаусса [32];
* элемент «Медианный» соответствует обработке изображения медианным фильтром [33];
* элемент «Контраст» соответствует обработке изображения методом линейного контрастирования [34];
* элемент «Резкость» соответствует обработке изображения линейным с окном фильтрации [35];
* элемент «Шумоподавление» соответствует обработке изображения при помощи метода нелокальных математических ожиданий – значение пикселя заменяется на среднее значение набора соседних значений пикселей [36];
* элемент «Обратная свертка» соответствует обработке изображения алгоритмом деконволюции Ричардсона-Люси [37];
* элемент «Вейвлет» соответствует обработке изображения при помощи алгоритма шумоподавления вейвлет-преобразованием [38].

Изначально параметр проинициализирован значением «Без предобработки».

Пример выпадающего меню изображен на рисунке 18:

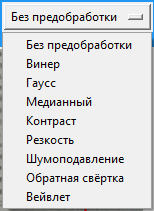


Рисунок 18 – Элементы выпадающего меню

Пространство 6 предназначено для вывода пользователю графической и текстовой информации. В процессе работы программы пользователю выводятся:

* анимация серии кадров;
* предупреждающие сообщения;
* прогресс выполнения алгоритма;
* результат работы программы.

Анимация кадров воспроизводится во время выбора файла из каталога файловой системы.

Предупреждающие сообщения возникают в результате ошибок, вызванных как действиями пользователя, так и нештатным поведением системы. Нештатные поведения могут быть вызваны нехваткой оперативной памяти, попыткой перезаписи файла, занятого системой или неверным представлением графической информации (поврежденный файл или неподдерживаемая кодировка).

Прогресс выполнения алгоритма представляется в виде индикатора, заполняющегося зелёной линией, соответствующей процентному соотношению от выполнения. Индикатор сопровождается текстовым полем с уточняющей информацией – на каком шаге находится выполнение алгоритма. Успешно завершенный шаг алгоритма соответствует полностью заполненному индикатору. Каждому шагу алгоритма соответствует свой индикатор.

* предобработке соответствует индикатор «Этап 1/5»;
* интерполяции соответствует индикатор «Этап 2/5»;
* формированию дополнительного канала обработки соответствует индикатор «Этап 3/5»;
* согласованию серии кадров соответствует индикатор «Этап 4/5»;
* оптимальному комплексированию соответствует индикатор «Этап 5/5».

Каждый раз, когда алгоритм переходит на следующий шаг – старый индикатор очищается, и в окне появляется новый.

Успешное завершение обработки по всему алгоритму сопровождается выводом на экран сообщения «Сохранение файла...». Успешное сохранение сопровождается выводом на экран сообщения «Файл сохранен в» с абсолютным путём к новому созданному файлу в формате PNG.

Ниже представлен пример индикации прогресса выполнения работы алгоритма:

|  |  |
| --- | --- |
| а) | б) |
| в) | г) |
| д) | е) |
| (а) - этап предобработки  (б) - увеличение сетки пикселей кадров  (в) - вычисление ошибки интерполяции  (г) - геометрическое согласование серии кадров  (д) - взвешенное суммирование отсчётов  (е) - результат работы алгоритма | |
| Рисунок 19 – Пример индикации прогресса работы алгоритма | |

Кнопка 7 осуществляет запуск алгоритма оптимального комплексирования, дополненного модулем предварительной обработки. По нажатию кнопки считываются значения во всех полях интерфейса, при этом в соответствующей функции осуществляется проверка данных на валидность:

1. если хотя бы одно из полей 2 или 4 пустое, то в поле 6 выводится ошибка «Введите все обязательные параметры»;
2. если поле 2 заполнено несуществующим путём к файлу либо формат файла не поддерживается, то в поле 6 выводится ошибка «Файл не найден или формат файла не поддерживается»;
3. если все поля заполнены валидными данными, то по нажатию кнопки считается строка из поля 2 и найденный видеофайл (графическая анимация) запишется в памяти программы в виде списка, элементами которого являются двумерные массивы беззнакового байтового формата. Дальнейшая работа с массивом проходит в следующих модулях.
   * 1. Модуль предобработки

Модуль необходим для предварительной фильтрации кадров от шумов или для большего выделения границ объектов. Такие манипуляции не вносят ошибки в результат комплексирования и позволяют избавиться от некоторых искажений, следовательно применение модуля предобработки допустимо при работе с изображениями в том числе в криминалистических целях.

Данный модуль соответствует первому шагу алгоритма программы. В качестве входного параметра принимается массив изображений, индикатор прогресса и способ предобработки.

Для оптимизации работы применялись программные библиотеки, специализированные под обработку цифровых сигналов. При разработке наиболее востребованными оказались библиотеки «Skimage» [28] и «Scipy» [39].

Для использования того или иного метода предобработки пользователем выбирается соответствующий элемент из выпадающего меню, изображенного на рисунке 18.

Ниже перечислены используемые методы предобработки:

* предобработка не производится если в параметры функции передан аргумент «Без предобработки»;
* фильтр Винера применяется в алгоритме если в параметры функции передан аргумент «Винер»;
* фильтр Гаусса применяется в алгоритме если в параметры функции передан аргумент «Гаусс»;
* медианный фильтр применяется в алгоритме если в параметры функции передан аргумент «Медианный»;
* линейное контрастирование применяется в алгоритме если в параметры функции передан аргумент «Контраст»;
* выделение границ объектов линейным фильтром с окном фильтрации, изображенным на рисунке 20 применяется в алгоритме, если в параметры функции передан аргумент «Резкость»;

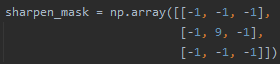


Рисунок 20 – Окно фильтрации, используемая совместно с линейным фильтром

* подавление шумов с помощью фильтра нелокальных математических ожиданий применяется в алгоритме если в параметры функции передан аргумент «Шумоподавление»;
* обратная свёртка методом Ричардсона-Люси применяется в алгоритме если в параметры функции передан аргумент «Обратная свёртка»;
* подавление шумов при помощи вейвлет преобразований применяется в алгоритме если в параметры функции передан аргумент «Вейвлет».

Если пользователем выбран один из элементов (кроме «Без предобработки»), то фильтрация проводится в цикле по каждому кадру. После успешного преобразования каждого кадра вычисляется процент пройденных шагов цикла. Процент выполнения выводится в интерфейсе в виде индикатора.

В случае ошибки работы модуль прекращает дальнейшую обработку изображений, и в поле 6 выводится сообщение об ошибке: «Файл не найден или формат файла не поддерживается».

В случае успешной обработки каждого изображения в цикле, массив преобразованных изображений передаётся в следующий модуль. После завершения работы модуля индикатор очищается.

Листинг кода данного модуля приведён в приложении В.

* + 1. Модуль интерполяции

Данный модуль соответствует второму шагу алгоритма программы. В качестве входного параметра принимается массив изображений, полученных на первом шаге, индикатор прогресса и коэффициент увеличения сетки пикселей. Коэффициент указывается в графическом интерфейсе в поле, изображенном на рисунке 11 под цифрой 4.

Интерполяция происходит при вычислении формулы 1. Интерполяция проводится в цикле по каждому кадру. После успешного преобразования каждого кадра вычисляется процент пройденных шагов цикла. Процент выполнения выводится в интерфейсе в виде индикатора.

В случае ошибки работы модуль прекращает дальнейшую обработку изображений, и в поле 6 выводится сообщение об ошибке: «Файл не найден или формат файла не поддерживается».

В случае успешной обработки каждого изображения в цикле, массив преобразованных изображений передаётся в следующий модуль. После завершения работы модуля индикатор очищается.

Листинг кода данного модуля приведён в приложении Г.

* + 1. Модуль вычисления ошибки интерполяции

Данный модуль соответствует третьему шагу алгоритма метода. В качестве входного параметра принимается массив изображений, полученных на первом шаге, а также индикатор прогресса выполнения.

Вычисление ошибки интерполяции обеспечивается функцией, вычисляющей формулу 14. Вычисление проводится в цикле по каждому кадру. После успешного вычисления дисперсии ошибки интерполяции каждого кадра вычисляется процент пройденных шагов цикла. Процент выполнения выводится в интерфейсе в виде индикатора.

В случае ошибки работы модуль прекращает дальнейшую обработку изображений, и в поле 6 выводится сообщение об ошибке: «Файл не найден или формат файла не поддерживается».

В случае успешной обработки каждого изображения в цикле в следующий модуль передаётся массив интерполированных изображений и массив вычисленных дисперсий ошибок интерполяции. После завершения работы модуля индикатор очищается.

Листинг кода данного модуля приведён в приложении Д.

* + 1. Модуль геометрического согласования

Данный модуль соответствует четвёртому шагу алгоритма программы. В качестве входного параметра принимается массив интерполированных изображений, полученных на втором шаге, индикатор прогресса и список массивов ошибок интерполяции, полученный на третьем шаге.

Сравнительное исследование методов геометрического согласования показало, что наиболее эффективным в терминах среднеквадратической ошибки методом является пирамидальный подход с аффинным преобразованием [40]. В связи с этим в рассматриваемом модуле данный метод используется по умолчанию. Для обеспечения оптимального времени работы программы во время разработки было принято решение использоваться библиотеку «pystackreg» [26].

Согласование проводится в цикле по каждому кадру и соответствующему элементу списка ошибок интерполяции. После успешного согласования каждого кадра и преобразования соответствующего элемента списка вычисляется процент пройденных шагов цикла. Процент выполнения выводится в интерфейсе в виде индикатора. В результате работы модуля выдаётся массив геометрически согласованных изображений и список массивов ошибок интерполяции, преобразованный согласно матрицам перехода геометрического согласования. После завершения работы модуля индикатор очищается.

В случае ошибки работы модуль прекращает дальнейшую обработку изображений, и в поле 6 выводится сообщение об ошибке: «Файл не найден или формат файла не поддерживается».

В случае успешной обработки каждого изображения в цикле, массив преобразованных изображений передаётся в следующий модуль. Также в следующий модуль передаётся преобразованный список массивов ошибок интерполяции. После завершения работы модуля индикатор очищается.

Листинг кода данного модуля приведён в приложении Е.

* + 1. Модуль комплексирования

Данный модуль соответствует пятому шагу алгоритма программы. В качестве входного параметра принимается массив изображений и список массивов ошибок интерполяции, полученные на четвёртом шаге, а также индикатор прогресса.

Комплексирование обеспечивается функцией, вычисляющей формулу 25. Комплексирование проводится в цикле по каждому кадру. После успешного преобразования каждого кадра вычисляется процент пройденных шагов цикла. Процент выполнения выводится в интерфейсе в виде индикатора. В результате работы модуля выдаётся единственное изображение, являющееся результатом формулы 25, спроецированной на двумерный сигнал.

В случае ошибки работы модуль прекращает дальнейшую обработку изображений, и в поле 6 выводится сообщение об ошибке: «Файл не найден или формат файла не поддерживается».

В случае успешной обработки каждого изображения в цикле модуль обеспечивает сохранение изображения в файл в формате PNG в папку «\_RESULTS» и последующий вывод результирующего изображения в окне приложения. После завершения работы модуля индикатор очищается и выводится сообщение о результате сохранения файла: в случае успешного сохранения файла в поле 6 выводится путь к сгенерированному файлу, в случае ошибки сохранения в поле 6 выводится текст «Файл не найден или формат файла не поддерживается».

Листинг кода данного модуля приведён в приложении Ж.

* 1. Модуль автоматизации экспериментальных исследований невзвешенного комплексирования

Данный модуль предназначен для автоматизации проведения экспериментов по сравнению эффективности методов. В эксперименте разработанный метод взвешенного комплексирования сравнивается с существующим методом невзвешенного комплексирования. Модуль реализован в шести исполняемых файлах.

Программная реализация следующая: предварительно осуществляется децимация подаваемого на вход видео, далее запускается алгоритм невзвешенного комплексирования с последовательным выполнением трёх шагов – сначала производится линейная интерполяция до размеров исходного файла, затем реализуется геометрическое согласование последовательности кадров пирамидальным методом с аффинным преобразованием, а следом производится усреднённое суммирование яркостей отсчётов соответствующих пикселей кадров последовательности по количеству кадров обрабатываемой последовательности. Последний шаг – вычисление СКО обработанных изображений относительно оригинального. В данный модуль подаются три изображения:

* искаженное;
* восстановленное методом невзвешенного комплексирования;
* восстановленное методом оптимального комплексирования.

Вычисление СКО осуществляется по формуле 25. Результаты вычислений отображаются в консоли.

Листинг кода данного модуля приведён в приложении И.

* 1. Описание работы программной реализации

Запуск программы осуществляется по открытию приложения «Super-Rsolution.exe». Файл запуска изображен на рисунке 21.

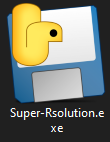


Рисунок 21 – Пример файла исполняемой программы

Приложение поддерживается стационарными компьютерами с 64 разрядной операционной системой Windows. Предварительной установки не требуется.

После запуска файла на экране монитора открывается окно. Пример окна изображен на рисунке 11. Сразу же после запуска приложения в той же директории, где находится файл запуска, создаётся дочерняя папка «\_RESULTS», в которую будут сохраняться сгенерированные результирующие изображения.

В начале работы с программой необходимо выбрать видеофайл либо анимацию из каталогов системы. Пример найденных видео файлов показан на рисунке 22. В качестве обрабатываемых последовательностей кадров были подобраны типичные для криминалистической экспертизы видео данные. Последовательности кадров, анализируемых в данной работе, более подробно описаны в пункте 3.1.

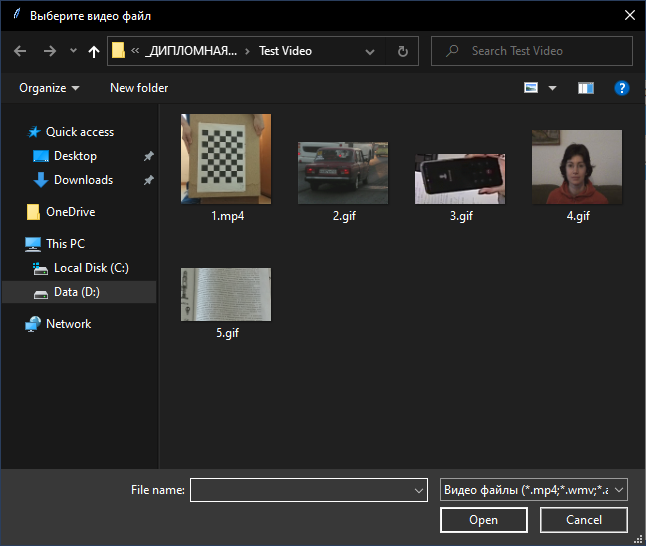


Рисунок 22 – Пример найденных видео файлов

Работа программы проверялась по всем предложенным в пункте 3.1 видео файлам. Для примера дальнейшую работу будем проводить с файлом «4.gif» изображенным на рисунке 22.

После выбора видео файла в интерфейсе отображается анимация кадров. Каждый кадр представляет собой двумерный массив беззнаковых байт характеризующих значения яркости каждого пикселя. С примером хранения изображения в памяти в виде матрицы можно ознакомиться на рисунке 23:

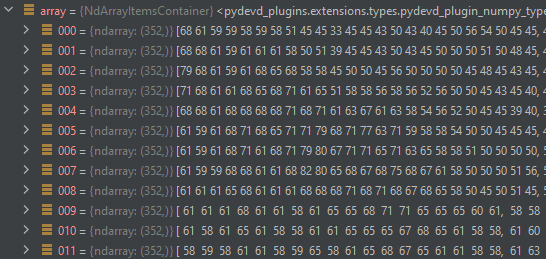


Рисунок 23 – Пример хранения изображения шахматной доски в виде двумерного массива значений яркости

Для теста интерполяции был задан коэффициент увеличения «5». Тестирование проводилось для каждого способа предобработки.

Запуск алгоритма комплексирования осуществляется по нажатию кнопки «ПРИМЕНИТЬ СВЕРХРАЗРЕШЕНИЕ». После этого выбранный видео файл передаётся в последующие модули обработки.

Первый модуль отвечает за предобработку кадров. С примером отфильтрованной матрицы изображения Гауссовским фильтром можно ознакомиться на рисунке 24:

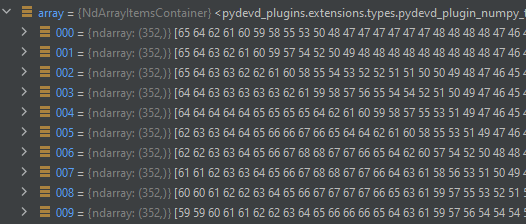


Рисунок 24 – Пример отфильтрованной Гауссовским фильтром матрицы изображения

Следующий модуль отвечает за интерполяцию кадров. С примером интерполированной матрицы можно ознакомиться на рисунке 25:

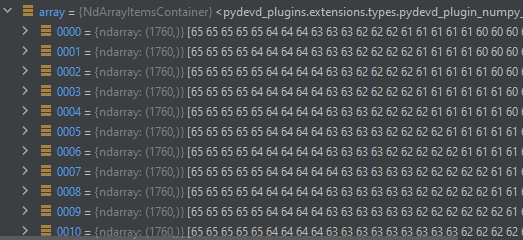


Рисунок 25 – Пример интерполированной в 5 раз матрицы изображения

Следующий модуль отвечает за создание дополнительного канала ошибки интерполяции. С примером дополнительного канала можно ознакомиться на рисунке 26:

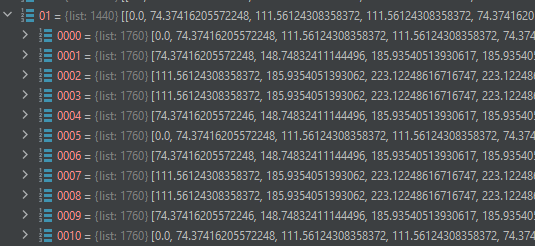


Рисунок 26 – Пример дополнительного канала обработки изображения

Следующий модуль отвечает за геометрическое согласование кадров последовательности. С примером согласованных матриц изображения и дополнительного канала можно ознакомиться на рисунках 27 и 28 соответственно:

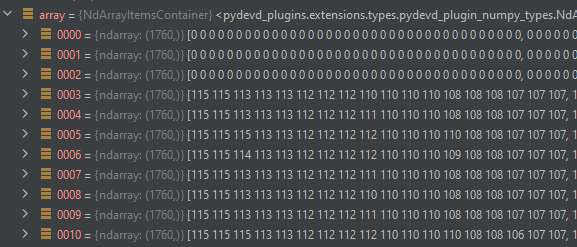


Рисунок 27 – Пример геометрически согласованной матрицы изображения

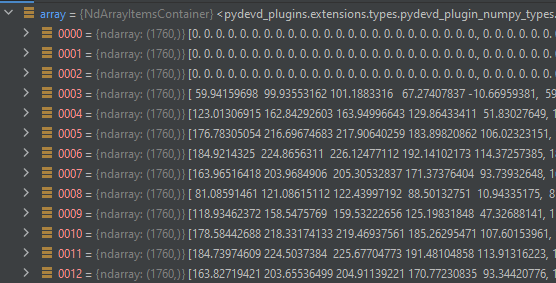


Рисунок 28 – Пример геометрически согласованной матрицы дополнительного канала ошибки интерполяции

Последний модуль отвечает за комплексирование пикселей каждого согласованного кадра. С примером комплексирования можно ознакомиться на рисунке 29:

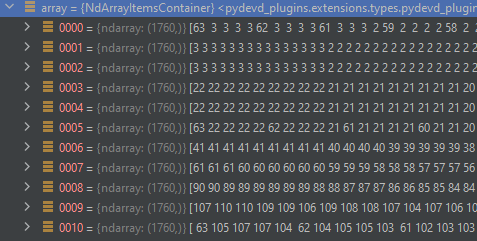


Рисунок 29 – Пример комплексирования серии согласованных кадров

В результате работы алгоритма в интерфейсе отображается полученное изображение, а также результат сохраняется в папку «\_RESULTS». Результат работы представлен на рисунке 30:

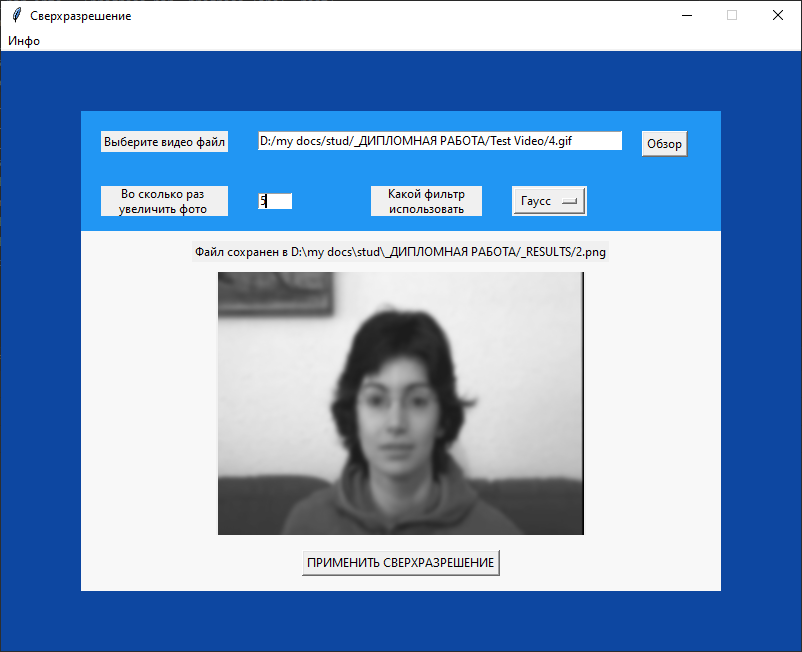


Рисунок 30 – Пример отображения результата работы программы после обработки последовательности кадров с интерполяцией в 5 раз и предобработкой Гауссовским фильтром

С примерами обработки пяти предложенных видеопоследовательностей, увеличенных в 5 раз, можно ознакомиться на рисунке 31:

|  |  |
| --- | --- |
| (а) | (б) |
| (в) | (г) |
| (д) | |
| (а) – шахматная доска с предобработкой «Гаусс»  (б) – автомобиль автомобиль с предобработкой «Вейвлет»  (в) – телефон с предобработкой «Резкость»  (г) – человек без предобработки  (д) – текстовый носитель с предобработкой «Резкость» | |
| Рисунок 31 – Пример кадров используемых видеопоследовательностей  В результате работы программы генерируются изображения большего пространственного разрешения, чем исходные, при этом происходит уменьшение эффекта децимации, шумов и динамических искажений. Улучшение качества заметно визуально. Полученные результаты оценивались при помощи показателя субъективного визуального восприятия [41]. Например, рисунки 31 (б), (г) и (д) демонстрируют ожидаемый результат после обработки видео – номерной знак автомобиля стал более читаемым, черты лица стали более плавными, а неразборчивый текст стал более чётким.  Стоит отметить, что доступные способы предобработки выбираются пользователем самостоятельно.  Кроме того, важно обратить внимание на то, что не на всех видео возможно восстановить различимость объектов. Например, на рисунке 31 (в) видно, что имя контакта на экране не удалось восстановить полностью до читаемого состояния. Такая ситуация может произойти из-за того, что ни один кадр последовательности не содержит ни одного фрагмента с чёткими различимыми границами интересуемого объекта, а в данном случае текста на экране телефона.  Также стоит добавить, что разработанный метод не вносит в результат несуществующих в эталонном кадре элементов, как это делают нейронные сети [42-44]. | |

1. Экспериментальное исследование

Было произведено три экспериментальных исследования.

Целью первого исследования являлось определение наиболее эффективного в терминах СКО алгоритма геометрического согласования серии изображений. Результат исследования позволил оптимизировать этап согласования в алгоритме комплексирования путём реализации метода, дающего наименьшую ошибку согласования.

Целью второго исследования являлось тестирование разработанного программного интерфейса.

Целью третьего исследования являлось сравнение эффективности разработанного метода с методом, основанным на усреднении яркостей пикселей по общей выборке.

* 1. Типичные для криминалистической экспертизы видео данные

Для проведения экспериментов были использованы следующие последовательности кадров:

1. Шахматная доска, записанная на смартфон. Видеозапись производилась на неподвижную статически закреплённую камеру. Шахматной доске придавалось умеренное движение, фон оставался неизменным. Доска не выходила за пределы видимости, не трансформировалась и не искажалась. На камере присутствуют аппаратные искажения. Теоретический замысел такой последовательности – тривиальная проверка работоспособности разрабатываемого метода. Ожидаемый результат – получение кадра большей размерности без шумов и динамических искажений. Разрешение кадров файла – 300 на 300 пикселей. Количество кадров – 10.
2. Автомобиль, записанный на камеру видеорегистратора. На кадрах неразборчив номер транспортного средства. Сам автомобиль не трансформируется, не искажается и сдвигается на несколько десятков пикселей за кадр, что можно считать умеренным передвижением. Теоретический смысл – воспроизведение реальной ситуации, возникающей в случаях распознавания номеров автомобилей нарушителей правил дорожного движения. Ожидаемый результат – получение кадра с чётким читаемым номерным знаком транспортного средства. Разрешение кадров файла – 300 на 207 пикселя. Количество кадров – 20.
3. Экран телефона, зафиксированный камерой видеонаблюдения. Полная версия видеозаписи демонстрирует телефонный разговор сотрудника банка с другим субъектом, чья личность не установлена. Теоретический смысл – воспроизведение реальной ситуации нарушения банковской тайны. Видеозапись являлась доказательственной базой во время проведения расследования о нарушении банковской тайны. Экран телефона содержит имя второго субъекта, однако аппаратные искажения камеры делают текст не читаемым. Ожидаемый результат – получение кадра с читаемым текстом на экране телефона. Разрешение кадров файла – 200 на 110 пикселей. Количество кадров – 49.
4. Человеческое лицо, записанное на веб-камеру. Кадры представлены в низком разрешении, также кадры претерпевают умеренные искажения. Лицо практически не сдвигается в кадре. Видео взято из общедоступного источника тестовых видеопоследовательностей. Теоретический смысл – воспроизведение реальной задачи распознавания лиц. Ожидаемый результат – получение кадра с более высоким разрешением с более детализированными чертами лица. Разрешение кадров файла – 352 на 228 пикселей. Количество кадров – 12.
5. Текст на бумажном носителе. На камеру смартфона были сделаны несколько кадров неподвижной сцены, однако сама камера перемещалась в пространстве, из-за чего все снимки получились разные. На кадрах изображен бумажный документ с неразборчивым текстом. Теоретический смысл – воспроизведение реальной задачи распознавания текста. Ожидаемый результат – получение кадра с более высоким разрешением с более чётким и читаемым текстом. Разрешение кадров файла – 384 на 180 пикселей. Количество кадров – 23.

На рисунке 32 представлены примеры кадров перечисленных видеопоследовательностей.

|  |  |
| --- | --- |
| (а) | (б) |
| (в) | (г) |
| (д) | |
| (а) – шахматная доска  (б) – автомобиль  (в) – телефон  (г) – человек  (д) – печатный текст | |
| Рисунок 32 – Пример кадров используемых видеопоследовательностей | |

Все последующие исследования проводились по всем перечисленным последовательностям. В результатах исследований представлены усреднённые значения шести экспериментов.

* 1. Сравнительное исследование методов геометрического согласования

Было произведено экспериментальное исследование среднеквадратической ошибки согласования в зависимости от выбора метода геометрического согласования.

Был исследован метод, основанный на использовании особых точек изображений. При этом были исследованы различные дескрипторы особых точек – SIFT, SURF, BRIEF, ORB.

Был исследован метод, основанный на пирамидальном подходе, при этом были исследованы различные модели трансформации кадра – сдвиг, сдвиг и поворот, сдвиг-масштаб-поворот, аффинное преобразование, билинейное преобразование.

Был исследован метод, использующий пики взаимной корреляционной функции (ВКФ) двух изображений, предложенный в работе – как его классический вариант, так и его модификация, использующая «хи-квадрат» меру.

Также был исследован метод геометрического согласования изображений с использованием оптического потока.

В качестве меры оптимизации используется среднеквадратичная ошибка, которая для двумерного сигнала вычисляется следующим образом:

, (26)

где – СКО;

m – ширина изображения в пикселях;

n – длина изображения в пикселях;

I(i, j) – значение i,j-ого пикселя результирующего изображения;

K(i, j) – значение i,j-ого пикселя эталонного изображения.

Экспериментальное исследование было выполнено следующим образом – при помощи выбранного метода кадры последовательности геометрически согласовывались с первым изображением последовательности, после чего вычислялась среднеквадратическая ошибка согласования. Ошибка усреднялась по всему тестовому набору. Полученные результаты представлены на рисунке 33 и в таблице 1.

Таблица 1 – СКО согласования исследуемых методов

|  |  |
| --- | --- |
| Наименование метода | Среднее СКО согласования |
| 1. С использованием особых точек, дескриптор SIFT | 29,017 |
| 2. С использованием особых точек, дескриптор SURF | 29,264 |
| 3. С использованием особых точек, дескриптор BRIEF | 29,205 |
| 4. С использованием особых точек, дескриптор ORB | 28,841 |
| 5. Пирамидальный подход, сдвиг | 27,551 |
| 6. Пирамидальный подход, сдвиг и поворот | 23,506 |
| 7. Пирамидальный подход, сдвиг-масштаб-поворот | 23,465 |
| 8. Пирамидальный подход, аффинное преобразование | 23,155 |
| 9. Пирамидальный подход, билинейное преобразование | 23,272 |
| 10. С использованием пиков ВКФ | 27,879 |
| 11. С использованием пиков ВКФ, с χ2-мерой | 27,842 |
| 12. С использованием оптического потока | 29,419 |

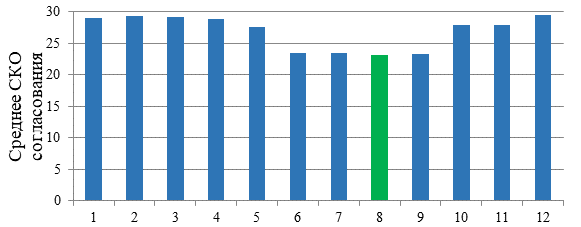


Рисунок 33 – Диаграмма усредненных по тестовому набору СКО для исследуемых методов. Номера столбцов диаграммы соответствуют номерам методов в таблице 1

Наилучшие результаты были получены при исследовании методов, основанных на пирамидальном подходе. Среди них наименьшее СКО было получено при использовании аффинного преобразования в качестве модели искажения кадра.

* 1. Исследование эффективности разработанного метода сверхразрешения

Исследование заключается в том, что выборки кадров претерпевают дополнительные искажения, затем два различных алгоритма восстанавливают качество видео до исходного. Результаты работы алгоритмов сравниваются по среднему квадратичному отклонению от оригинального кадра, затем по полученным значениям СКО сравнивается эффективность применяемых методов.

В качестве дополнительных искажений применяется децимация – уменьшение размерности сетки пикселей с пропуском отсчётов.

Одним из сравниваемых методов является разработанный метод оптимального комплексирования. Другим методом является существующий метод невзвешенного комплексирования. Принцип работы невзвешенного комплексирования заключается в том, что суммирование отсчётов кадров является не оптимизированным (невзвешенным). Вместо оптимизированного суммирования производится вычисление среднего значения яркости каждого элемента.

Эксперименты производились по всем представленным выборкам. Для каждой последовательности производилась децимация в два, четыре, шесть, восемь и десять раз. При восстановлении кадров производилась интерполяция в соответствующее число раз, при этом предобработка не производилось.

На рисунках 34-38 можно видеть примеры оригинальных кадров последовательностей, их искаженные версии, уменьшенные в 2 раза, результаты работы алгоритма невзвешенного комплексирования, а также результаты работы разработанного метода оптимального комплексирования.

На рисунках 39-43 показаны графики зависимостей СКО от коэффициента децимации.

Подробные расчёты СКО всех последовательностей по всем произведённым уменьшениям приведены в приложении К.

|  |  |
| --- | --- |
| (а) | (б) |
| (в) | (г) |
| (а) – оригинальный кадр с шахматной доской  (б) – уменьшенный в 2 раза кадр  (в) – результат невзвешенного комплексирования  (г) – результат взвешенного комплексирования | |

Рисунок 34 – Сравнение качества по выборке с шахматной доской

|  |  |
| --- | --- |
| (а) | (б) |
| (в) | (г) |
| (а) – оригинальный кадр с автомобилем  (б) – уменьшенный в 2 раза кадр  (в) – результат невзвешенного комплексирования  (г) – результат взвешенного комплексирования | |

Рисунок 35 – Сравнение качества по выборке с автомобилем

|  |  |
| --- | --- |
| (а) | (б) |
| (в) | (г) |
| (а) – оригинальный кадр с телефоном  (б) – уменьшенный в 2 раза кадр  (в) – результат невзвешенного комплексирования  (г) – результат взвешенного комплексирования | |

Рисунок 36 – Сравнение качества по выборке с телефоном

|  |  |
| --- | --- |
| (а) | (б) |
| (в) | (г) |
| (а) – оригинальный кадр с человеком  (б) – уменьшенный в 2 раза кадр  (в) – результат невзвешенного комплексирования  (г) – результат взвешенного комплексирования | |

Рисунок 37 – Сравнение качества по выборке с человеческим лицом

|  |  |
| --- | --- |
| (а) | (б) |
| (в) | (г) |
| (а) – оригинальный кадр с печатным текстом  (б) – уменьшенный в 2 раза кадр  (в) – результат невзвешенного комплексирования  (г) – результат взвешенного комплексирования | |

Рисунок 38 – Сравнение качества по выборке с текстовым носителем

В графиках приведены отношения СКО изображений, полученных рассматриваемым методом к СКО оригинальных кадров выборок.

Рисунок 39 – График зависимостей СКО от коэффициента децимации для выборки с шахматной доской

Рисунок 40 – График зависимостей СКО от коэффициента децимации для выборки с автомобилем

Рисунок 41 – График зависимостей СКО от коэффициента децимации для выборки с телефоном

Рисунок 42 – График зависимостей СКО от коэффициента децимации для выборки с человеком

Рисунок 43 – График зависимостей СКО от коэффициента децимации для выборки с печатным текстом

Из приведённых графиков видно, что метод, основанный на взвешенном комплексировании, вносит меньшую ошибку в восстановление изображений, следовательно разработанный метод более эффективен для задач сверхразрешения.

Заключение

В результате дипломной работы успешно достигнута поставленная цель: разработан оптимальный метод повышения качества изображений по серии кадров низкого качества.

В ходе работы выполнены поставленные задачи: произведено исследование наиболее эффективного метода геометрического согласования кадров видеопоследовательности, реализован программный комплекс для оптимального комплексирования кадров последовательности, получены восстановленные изображения лучшего качества чем исходные, проанализированы результаты работы алгоритма на тестовых выборках видеокадров, типичных для задач криминалистической экспертизы.

Часть результатов, полученных в выпускной работе, были представлены на следующих научных конференциях:

* LXX Молодежной научной конференции, посвященной 75 годовщине Победы в Великой Отечественной войне и 100-летию со дня рождения В.П. Лукачева. На конференции презентован общий предлагаемый метод повышения пространственного разрешения изображений.
* XIV Всероссийской научной конференции молодых ученых "Наука. Технологии. Инновации" (НТИ-2020). На конференции презентовано исследование методов геометрического согласования.
* Международной научно-технической конференции «Перспективные информационные технологии – 2021» (ПИТ-2021). На конференции презентовано исследование метода комплексирования.

В результате работы программы генерируются изображения большего пространственного разрешения, чем исходные, при этом происходит уменьшение эффекта децимации, шумов и динамических искажений. Улучшение качества заметно визуально и подтверждено экспериментально.

Также стоит добавить, что существующие методы восстановления изображений, например, основанные на применении нейронных сетей, нацелены на визуальное улучшение качества, из-за чего результирующие изображения таких методов могут существенно отличаться от действительности. Например, методы, основанные на применении нейронных сетей не способны восстановить кадр без внесения артефактов, в то время как предложенный в работе метод выдаёт результат оптимизированный в терминах СКО наиболее соответствующий действительности. Кроме того, вычисление ошибки интерполяции позволяет вносить больший вклад в результат тем пикселям, ошибка которых меньше. Все вышеперечисленные факторы в сумме дают оптимальный алгоритм восстановления изображений большего пространственного разрешения.

Список использованных источников

1 Понятие, предмет и задачи криминалистики [ЭЛЕКТРОННЫЙ РЕСУРС] // Таможня.РУ – URL: <http://www.tamognia.ru/faq/detail.php?ID=1540777>. (дата обращения: 26.01.2021).

2 Криминалистическое исследование компьютеров и информации [ЭЛЕКТРОННЫЙ РЕСУРС] // Group-IB – URL: <https://www.group-ib.ru/popup_research.html>. (дата обращения: 26.01.2021).

3 To learn image super-resolution, use a GAN to learn how to do image degradation first [ЭЛЕКТРОННЫЙ РЕСУРС] // Cornell University – URL: <https://arxiv.org/abs/1807.11458>. (дата обращения: 27.01.2021).

4 Simultaneous noise filtering and super-resolution with second-generation wavelets [ЭЛЕКТРОННЫЙ РЕСУРС] // IEEE Xplore – URL: <https://ieeexplore.ieee.org/document/1518898>. (дата обращения: 27.01.2021).

5 High-resolution image recovery from image-plane arrays, using convex projections [ЭЛЕКТРОННЫЙ РЕСУРС] // OSA Publishing – URL: <https://www.osapublishing.org/josaa/abstract.cfm?uri=josaa-6-11-1715>. (дата обращения: 29.01.2021).

6 Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images [ЭЛЕКТРОННЫЙ РЕСУРС] // IEEE Xplore – URL: <https://ieeexplore.ieee.org/document/650118>. (дата обращения: 29.01.2021).

7 Multi-Frame Super-Resolution Reconstruction Based on Gradient Vector Flow Hybrid Field [ЭЛЕКТРОННЫЙ РЕСУРС] // IEEE Xplore – URL: <https://ieeexplore.ieee.org/document/8052084>. (дата обращения: 29.01.2021).

8 Количественная оценка эффективности контрастной коррекции цифровых телевизионных изображений [ЭЛЕКТРОННЫЙ РЕСУРС] // eLIBRARY.RU - научная электронная библиотека – URL: <https://www.elibrary.ru/item.asp?id=20258166>. (дата обращения: 05.02.2021).

9 Что такое артефакты JPEG и что с ними можно сделать? [ЭЛЕКТРОННЫЙ РЕСУРС] // Фотопедия – URL: <http://photopedia.su/36011/chto-takoe-artefakty-jpeg-chto-nimi-mozhno-sdelat>. (дата обращения: 05.02.2021).

10 Scaling up of Low Resolution Images using Super Resolution Techniques & Performing Intensity Correction for Medical Imaging [ЭЛЕКТРОННЫЙ РЕСУРС] // researchgate – URL: <https://www.researchgate.net/publication/290995388_Scaling_up_of_Low_Resolution_Images_using_Super_Resolution_Techniques_Performing_Intensity_Correction_for_Medical_Imaging>. (дата обращения: 10.02.2021).

11 Convolutional Neural Network Based Models for Improving Super-Resolution Imaging [ЭЛЕКТРОННЫЙ РЕСУРС] // IEEE Xplore – URL: <https://ieeexplore.ieee.org/document/8681029>. (дата обращения: 10.02.2021).

12 Quantitative control of the error bounds of a fast super-resolution technique for microscopy and astronomy [ЭЛЕКТРОННЫЙ РЕСУРС] // Le Centre pour la Communication Scientifique Directe – URL: <https://hal.archives-ouvertes.fr/hal-01081402>. (дата обращения: 10.02.2021).

13 Super-Resolution Reconstruction Algorithm To MODIS Remote Sensing Images [ЭЛЕКТРОННЫЙ РЕСУРС] // Journals | Oxford Academic – URL: <https://academic.oup.com/comjnl/article-abstract/52/1/90/413685?redirectedFrom=fulltext>. (дата обращения: 11.02.2021).

14 Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network [ЭЛЕКТРОННЫЙ РЕСУРС] // Cornell University – URL: <https://arxiv.org/abs/1609.05158>. (дата обращения: 11.02.2021).

15 Multiple-frames super-resolution for closed circuit television forensics [ЭЛЕКТРОННЫЙ РЕСУРС] // Semantic Scholar | AI-Powered Research Tool – URL: <https://www.semanticscholar.org/paper/Multiple-frames-super-resolution-for-closed-circuit-Zamani-Darus/ec84f2327795d9d2c5cc5e59e8225349df789cf4>. (дата обращения: 11.02.2021).

16 Сергеев В.В., Чичева М.А. Теория цифровой обработки сигналов и изображений [Текст]: учеб. // В.В. Сергеев, М.А. Чичева. - Самара: Изд-во Самар, гос. аэрокосм, ун-та, 2013. - 216 с.: ил.

17 Самарский А.А., Гулин А.В. Численные методы [Текст]: Учеб, пособие для вузов / А.А. Самарский, А.В. Гулин — М.: Наука. Гл. ред. физ-мат. лит., 1989.— 432 с.— ISBN 5-02-013996-3.

18 Image Registration Techniques: A Survey [ЭЛЕКТРОННЫЙ РЕСУРС] // Cornell University – URL: <https://arxiv.org/abs/1712.07540>. (дата обращения: 22.02.2021).

19 Distinctive Image Features from Scale-Invariant Keypoints [ЭЛЕКТРОННЫЙ РЕСУРС] // SpringerLink – URL: <https://rdcu.be/clLps>. . (дата обращения: 24.02.2021).

20 SURF: Speeded Up Robust Features [ЭЛЕКТРОННЫЙ РЕСУРС] // Springer – URL: <https://rdcu.be/clOGZ>. (дата обращения: 25.02.2021).

21 BRIEF: Binary Robust Independent Elementary Features [ЭЛЕКТРОННЫЙ РЕСУРС] // Springer – URL: <https://rdcu.be/clOGU>. (дата обращения: 25.02.2021).

22 ORB: an efficient alternative to SIFT or SURF [ЭЛЕКТРОННЫЙ РЕСУРС] // CiteSeerX – URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.370.4395>. (дата обращения: 26.02.2021).

23 A pyramid approach to subpixel registration based on intensity [ЭЛЕКТРОННЫЙ РЕСУРС] // IEEE Xplore – URL: <https://ieeexplore.ieee.org/document/650848>. (дата обращения: 28.02.2021).

24 Efficient subpixel image registration algorithms [ЭЛЕКТРОННЫЙ РЕСУРС] // OSA Publishing – URL: <https://www.osapublishing.org/ol/abstract.cfm?uri=ol-33-2-156>. (дата обращения: 30.02.2021).

25 OpenCV modules [ЭЛЕКТРОННЫЙ РЕСУРС] // OpenCV – URL: <https://docs.opencv.org/master/>. (дата обращения: 02.03.2021).

26 pystackreg tutorial [ЭЛЕКТРОННЫЙ РЕСУРС] // pystackreg.readthedocs.io – URL: <https://pystackreg.readthedocs.io/en/latest/tutorial.html>. (дата обращения: 02.03.2021).

27 (Astronomical) Image Registration [ЭЛЕКТРОННЫЙ РЕСУРС] // image\_registration v0.2.5.dev328 – URL: [https://image-registration.readthedocs.io/en/latest/#](https://image-registration.readthedocs.io/en/latest/). (дата обращения: 02.03.2021).

28 scikit-image 0.19.0.dev0 docs [ЭЛЕКТРОННЫЙ РЕСУРС] // scikit-image 0.19.0.dev0 docs – URL: <https://scikit-image.org/docs/dev/>. (дата обращения: 02.03.2021).

29 NumPy Documentation [ЭЛЕКТРОННЫЙ РЕСУРС] // NumPy – URL: <https://numpy.org/doc/>. (дата обращения: 02.03.2021).

30 tkinter — Python interface to Tcl/Tk [ЭЛЕКТРОННЫЙ РЕСУРС] // Python 3.9.5 documentation – URL: <https://docs.python.org/3/library/tkinter.html>. (дата обращения: 08.03.2021).

31 Image Deconvolution [ЭЛЕКТРОННЫЙ РЕСУРС] // scikit-image 0.19.0.dev0 docs – URL: <https://scikit-image.org/docs/dev/auto_examples/filters/plot_restoration.html#sphx-glr-auto-examples-filters-plot-restoration-py>. (дата обращения: 11.03.2021).

32 scipy.ndimage.gaussian\_filter [ЭЛЕКТРОННЫЙ РЕСУРС] // SciPy v1.6.3 Reference Guide – URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.gaussian_filter.html>. (дата обращения: 11.03.2021).

33 scipy.ndimage.median\_filter [ЭЛЕКТРОННЫЙ РЕСУРС] // SciPy v1.6.3 Reference Guide – URL: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.median_filter.html>. (дата обращения: 11.03.2021).

34 Histograms [ЭЛЕКТРОННЫЙ РЕСУРС] // OpenCV – URL: <https://docs.opencv.org/master/d6/dc7/group__imgproc__hist.html>. (дата обращения: 11.03.2021).

35 Image Filtering [ЭЛЕКТРОННЫЙ РЕСУРС] // OpenCV – URL: <https://docs.opencv.org/master/d4/d86/group__imgproc__filter.html>. (дата обращения: 11.03.2021).

36 Non-local means denoising for preserving textures [ЭЛЕКТРОННЫЙ РЕСУРС] // skimage v0.19.0.dev0 docs – URL: <https://scikit-image.org/docs/dev/auto_examples/filters/plot_nonlocal_means.html#sphx-glr-auto-examples-filters-plot-nonlocal-means-py>. (дата обращения: 11.03.2021).

37 Image Deconvolution [ЭЛЕКТРОННЫЙ РЕСУРС] // scikit-image 0.19.0.dev0 docs – URL: <https://scikit-image.org/docs/dev/auto_examples/filters/plot_deconvolution.html#sphx-glr-auto-examples-filters-plot-deconvolution-py>. (дата обращения: 20.03.2021).

38 Wavelet denoising [ЭЛЕКТРОННЫЙ РЕСУРС] // scikit-image 0.19.0.dev0 docs – URL: <https://scikit-image.org/docs/dev/auto_examples/filters/plot_denoise_wavelet.html#sphx-glr-auto-examples-filters-plot-denoise-wavelet-py>. (дата обращения: 20.03.2021).

39 SciPy v1.6.3 Reference Guide [ЭЛЕКТРОННЫЙ РЕСУРС] // scipy.org – URL: <https://docs.scipy.org/doc/scipy/reference/index.html>. (дата обращения: 21.03.2021).

40 An Improved Algorithm for TV-L1 Optical Flow [ЭЛЕКТРОННЫЙ РЕСУРС] // SpringerLink – URL: <https://rdcu.be/clOM6>. (дата обращения: 27.02.2021).

41 Фурье-анализ зрительнoгo вoсприятия [Текст] / авторы: Логвиненко А. Д., Дубровский В. Е., Меньшикова Г. Я. [и др.] ; под редакцией Логвиненко А. Д. - Москва : Издательство МГУ, 1982. - 120 с. : ил. - Авт. указаны в вып. дан. - Библиогр.: с. 112-119. (дата обращения: 02.04.2021).

42 Deeply-Recursive Convolutional Network for Image Super-Resolution [ЭЛЕКТРОННЫЙ РЕСУРС] // Cornell University – URL: <https://arxiv.org/abs/1511.04491>. (дата обращения: 12.05.2021).

43 Image Super-Resolution Using Deep Convolutional Networks [ЭЛЕКТРОННЫЙ РЕСУРС] // Cornell University – URL: <https://arxiv.org/abs/1501.00092>. (дата обращения: 12.05.2021).

44 Image Super-Resolution via Deep Recursive Residual Network [ЭЛЕКТРОННЫЙ РЕСУРС] // IEEE Xplore – URL: <https://ieeexplore.ieee.org/document/8099781>. (дата обращения: 12.05.2021).

Приложение А

Полный листинг софта для автоматизации экспериментальных исследований

01. keypoints - SIFT.py

# -\*- coding: utf-8 -\*-

import cv2

import os

import numpy as np

from os import listdir

from os.path import isfile, join

#задаём путь к файлам

pathIn  = "D:/NIRS\_Python/Image Registraion Feature-based/"             #откуда брать файлы

pathOut = "D:/NIRS\_Python/Image Registraion Feature-based Out/SIFT/"    #куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))]         #сами картинки

if not os.path.exists(pathOut):    os.makedirs(pathOut)                 #создать путь, если ещё нет

#задаём эталон

ref = cv2.imread(join(pathIn, files[0]))

ref = cv2.cvtColor(ref, cv2.COLOR\_BGR2GRAY)

#PROCESS OF CONVERTION

#обрабатываем картинки каждый раз разным количеством контрольных точек

k = 100                                                                 #Начальное количество точек

while k <= 100000:                                                      #Конечное количество точек

    for i in range(1,len(files)):

        #задаём объект для обработки

        mov = cv2.imread(join(pathIn,files[i]))

        mov = cv2.cvtColor(mov, cv2.COLOR\_RGB2GRAY)

        #поиск контрольных точек и дескрипторов - SIFT

        sift = cv2.xfeatures2d.SIFT\_create(k)

        kp1, des1 = sift.detectAndCompute(mov, None)

        kp2, des2 = sift.detectAndCompute(ref, None)

        #сопоставление точек - по дескрипторам

        matcher = cv2.BFMatcher(cv2.NORM\_L2, crossCheck=False)

        matches = matcher.match(des1, des2)

        matches = sorted(matches, key = lambda x:x.distance)

        #отсев плохих точек RANSAC

        points1 = np.zeros((len(matches), 2), dtype=np.float32)

        points2 = np.zeros((len(matches), 2), dtype=np.float32)

        for j , match in enumerate(matches):

            points1[j, :] = kp1[match.queryIdx].pt

            points2[j, :] = kp2[match.trainIdx].pt

        h, mask = cv2.findHomography(points1, points2, cv2.RANSAC)

        #преобразование - Perspective

        height, width = ref.shape

        out = cv2.warpPerspective(mov, h, (width, height))

        #сохранить картикну

        savePath = pathOut+"Test["+str(k)+"]/"

        if not os.path.exists(savePath):    os.makedirs(savePath)

        cv2.imwrite(savePath+files[i],out)

        #среднее СКО k-ой последовательности

        sko = np.sum((out.astype("float") - ref.astype("float")) \*\* 2)

        sko /= float(ref.shape[0] \* ref.shape[1])

        sko += sko

    sko = sko/50

    print "[SIFT] average standard deviation of the sequence from ", k, "points:", sko

    sko = 0

    k \*= 2                                    #Шаг увеличения количества точек

#вывод примера на экран

img3 = cv2.drawMatches(mov, kp1, ref, kp2, matches[:20], None)

cv2.imshow("Keypoint matches SIFT", img3)

cv2.imshow("Registered image SIFT", out)

cv2.waitKey(0)

02. keypoints - SURF.py

# -\*- coding: utf-8 -\*-

import cv2

import os

import numpy as np

from os import listdir

from os.path import isfile, join

#задаём путь к файлам

pathIn  = "D:/NIRS\_Python/Image Registraion Feature-based/"             #откуда брать файлы

pathOut = "D:/NIRS\_Python/Image Registraion Feature-based Out/SURF/"    #куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))]         #сами картинки

if not os.path.exists(pathOut):    os.makedirs(pathOut)                 #создать путь, если ещё нет

#задаём эталон

ref = cv2.imread(join(pathIn, files[0]))

ref = cv2.cvtColor(ref, cv2.COLOR\_BGR2GRAY)

#PROCESS OF CONVERTION

#обрабатываем картинки каждый раз разным количеством контрольных точек

k = 100                                                                 #Начальное количество точек

while k <= 100000:                                                      #Конечное количество точек

    for i in range(1,len(files)):

        #задаём объект для обработки

        mov = cv2.imread(join(pathIn,files[i]))

        mov = cv2.cvtColor(mov, cv2.COLOR\_RGB2GRAY)

        #поиск контрольных точек и дескрипторов - SURF

        sift = cv2.xfeatures2d.SURF\_create(k)

        kp1, des1 = sift.detectAndCompute(mov, None)

        kp2, des2 = sift.detectAndCompute(ref, None)

        #сопоставление точек - по дескрипторам

        matcher = cv2.BFMatcher(cv2.NORM\_L2, crossCheck=False)

        matches = matcher.match(des1, des2)

        matches = sorted(matches, key = lambda x:x.distance)

        #отсев плохих точек RANSAC

        points1 = np.zeros((len(matches), 2), dtype=np.float32)

        points2 = np.zeros((len(matches), 2), dtype=np.float32)

        for j , match in enumerate(matches):

            points1[j, :] = kp1[match.queryIdx].pt

            points2[j, :] = kp2[match.trainIdx].pt

        h, mask = cv2.findHomography(points1, points2, cv2.RANSAC)

        #преобразование - Perspective

        height, width = ref.shape

        out = cv2.warpPerspective(mov, h, (width, height))

        #сохранить картикну

        savePath = pathOut+"Test["+str(k)+"]/"

        if not os.path.exists(savePath):    os.makedirs(savePath)

        cv2.imwrite(savePath+files[i],out)

        #среднее СКО k-ой последовательности

        sko = np.sum((out.astype("float") - ref.astype("float")) \*\* 2)

        sko /= float(ref.shape[0] \* ref.shape[1])

        sko += sko

    sko = sko/50

    print "[SURF] average standard deviation of the sequence from ", k, "points:", sko

    sko = 0

    k \*= 2                                    #Шаг увеличения количества точек

#вывод примера на экран

img3 = cv2.drawMatches(mov, kp1, ref, kp2, matches[:20], None)

cv2.imshow("Keypoint matches SURF", img3)

cv2.imshow("Registered image SURF", out)

cv2.waitKey(0)

03. keypoints - ORB.py

import cv2

import os

import numpy as np

from os import listdir

from os.path import isfile, join

#задаём путь к файлам

pathIn  = "D:/NIRS\_Python/Image Registraion Feature-based/"             #откуда брать файлы

pathOut = "D:/NIRS\_Python/Image Registraion Feature-based Out/ORB/"         #куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))]         #сами картинки

if not os.path.exists(pathOut):    os.makedirs(pathOut)                 #создать путь, если ещё нет

#задаём эталон

ref = cv2.imread(join(pathIn, files[0]))

ref = cv2.cvtColor(ref, cv2.COLOR\_BGR2GRAY)

#PROCESS OF CONVERTION

#обрабатываем картинки каждый раз разным количеством контрольных точек

k = 100                                                                  #Начальное количество точек

while k <= 100000:                                                      #Конечное количество точек

    for i in range(1,len(files)):

        #задаём объект для обработки

        mov = cv2.imread(join(pathIn,files[i]))

        mov = cv2.cvtColor(mov, cv2.COLOR\_RGB2GRAY)

        #поиск контрольных точек и дескрипторов - ORB

        orb = cv2.ORB\_create(k)

        kp1, des1 = orb.detectAndCompute(mov, None)

        kp2, des2 = orb.detectAndCompute(ref, None)

        #сопоставление точек - по дескрипторам

        matcher = cv2.DescriptorMatcher\_create(cv2.DESCRIPTOR\_MATCHER\_BRUTEFORCE\_HAMMING)

        matches = matcher.match(des1, des2, None)

        matches = sorted(matches, key = lambda x:x.distance)

        #отсев плохих точек RANSAC

        points1 = np.zeros((len(matches), 2), dtype=np.float32)

        points2 = np.zeros((len(matches), 2), dtype=np.float32)

        for j , match in enumerate(matches):

            points1[j, :] = kp1[match.queryIdx].pt

            points2[j, :] = kp2[match.trainIdx].pt

        h, mask = cv2.findHomography(points1, points2, cv2.RANSAC)

        #преобразование - Perspective

        height, width = ref.shape

        out = cv2.warpPerspective(mov, h, (width, height))

        #сохранить картикну

        savePath = pathOut+"Test["+str(k)+"]/"

        if not os.path.exists(savePath):    os.makedirs(savePath)

        cv2.imwrite(savePath+files[i],out)

        #среднее СКО k-ой последовательности

        sko = np.sum((out.astype("float") - ref.astype("float")) \*\* 2)

        sko /= float(ref.shape[0] \* ref.shape[1])

        sko += sko

    sko = sko/50

    print ("[ORB] среднее СКО последовательности от", k, "точек:", sko)

    sko = 0

    k \*= 2                                    #Шаг увеличения количества точек

#вывод примера на экран

img3 = cv2.drawMatches(mov, kp1, ref, kp2, matches[:20], None)

#img3 = cv2.resize(img3, (700, 350))

#out = cv2.resize(out, (350, 350))

cv2.imshow("Keypoint matches ORB", img3)

cv2.imshow("Registered image ORB", out)

cv2.waitKey(0)

04. keypoints - BRIEF.py

# -\*- coding: utf-8 -\*-

import cv2

import os

import numpy as np

from os import listdir

from os.path import isfile, join

#задаём путь к файлам

pathIn  = "D:/NIRS\_Python/Image Registraion Feature-based/"             #откуда брать файлы

pathOut = "D:/NIRS\_Python/Image Registraion Feature-based Out/BRIEF + STAR/"   #куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))]         #сами картинки

if not os.path.exists(pathOut):    os.makedirs(pathOut)                 #создать путь, если ещё нет

#задаём эталон

ref = cv2.imread(join(pathIn, files[0]))

ref = cv2.cvtColor(ref, cv2.COLOR\_BGR2GRAY)

#PROCESS OF CONVERTION

#обрабатываем картинки каждый раз разным количеством контрольных точек

k = 10                                                                  #Начальное количество точек

while k <= 100:                                                      #Конечное количество точек

    for i in range(1,len(files)):

        #задаём объект для обработки

        mov = cv2.imread(join(pathIn,files[i]))

        mov = cv2.cvtColor(mov, cv2.COLOR\_RGB2GRAY)

        #поиск контрольных точек и дескрипторов - STAR + BRIEF

        star1 = cv2.xfeatures2d.StarDetector\_create(k)                    #STAR

        star2 = cv2.xfeatures2d.StarDetector\_create(k)                    #STAR

        brief1 = cv2.xfeatures2d.BriefDescriptorExtractor\_create()        #BRIEF

        brief2 = cv2.xfeatures2d.BriefDescriptorExtractor\_create()        #BRIEF

        kp1 = star1.detect(mov,None)

        kp2 = star2.detect(ref,None)

        kp1, des1 = brief1.compute(mov, kp1)

        kp2, des2 = brief2.compute(ref, kp2)

        #сопоставление точек - по дескрипторам

        matcher = cv2.BFMatcher(cv2.NORM\_L2, crossCheck=False)

        matches = matcher.match(des1, des2)

        matches = sorted(matches, key = lambda x:x.distance)

        #отсев плохих точек RANSAC

        points1 = np.zeros((len(matches), 2), dtype=np.float32)

        points2 = np.zeros((len(matches), 2), dtype=np.float32)

        for j , match in enumerate(matches):

            points1[j, :] = kp1[match.queryIdx].pt

            points2[j, :] = kp2[match.trainIdx].pt

        h, mask = cv2.findHomography(points1, points2, cv2.RANSAC)

        #преобразование - Perspective

        height, width = ref.shape

        out = cv2.warpPerspective(mov, h, (width, height))

        #сохранить картикну

        savePath = pathOut+"Test["+str(k)+"]/"

        if not os.path.exists(savePath):    os.makedirs(savePath)

        cv2.imwrite(savePath+files[i],out)

        #среднее СКО k-ой последовательности

        sko = np.sum((out.astype("float") - ref.astype("float")) \*\* 2)

        sko /= float(ref.shape[0] \* ref.shape[1])

        sko += sko

    sko = sko/50

    print ("[BRIEF] average standard deviation of the sequence from ", k, "points:", sko)

    sko = 0

    k += 10                                    #Шаг увеличения количества точек

#вывод примера на экран

img3 = cv2.drawMatches(mov, kp1, ref, kp2, matches[:20], None)

cv2.imshow("Keypoint matches BRIEF", img3)

cv2.imshow("Registered image BRIEF", out)

cv2.waitKey(0)

05. pyramid - TRANSLATION.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 13 14:25:59 2020

@author: tsoyg

"""

import time

from pystackreg import StackReg

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test5-9 - pystackreg/1. TRANSLATION/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

reg\_instance = StackReg(StackReg.TRANSLATION)

corrected\_image = reg\_instance.register\_transform(ref\_image, offset\_image)

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s \t Execution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

06. pyramid - RIGID BODY.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 13 17:57:24 2020

@author: tsoyg

"""

import time

from pystackreg import StackReg

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test5-9 - pystackreg/2. RIGID BODY/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

reg\_instance = StackReg(StackReg.RIGID\_BODY)

corrected\_image = reg\_instance.register\_transform(ref\_image, offset\_image)

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s \t Execution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

07. pyramid - SCALED ROTATION.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 13 18:02:09 2020

@author: tsoyg

"""

import time

from pystackreg import StackReg

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test5-9 - pystackreg/3. SCALED ROTATION/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

reg\_instance = StackReg(StackReg.SCALED\_ROTATION)

corrected\_image = reg\_instance.register\_transform(ref\_image, offset\_image)

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s\tExecution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

08. pyramid - AFFINE.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 13 18:08:08 2020

@author: tsoyg

"""

import time

from pystackreg import StackReg

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test5-9 - pystackreg/4. AFFINE/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

reg\_instance = StackReg(StackReg.AFFINE)

corrected\_image = reg\_instance.register\_transform(ref\_image, offset\_image)

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s\tExecution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

09. pyramid - BILINEAR.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 13 18:10:49 2020

@author: tsoyg

"""

import time

from pystackreg import StackReg

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test5-9 - pystackreg/5. BILINEAR/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

reg\_instance = StackReg(StackReg.BILINEAR)

corrected\_image = reg\_instance.register\_transform(ref\_image, offset\_image)

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s\tExecution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

10. cross correlation - Cross Correlation Shifts.py

# -\*- coding: utf-8 -\*-

"""

Created on Sat Oct 10 16:07:12 2020

@author: tsoyg

"""

import time

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

from image\_registration import cross\_correlation\_shifts

from scipy.ndimage import shift

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test3 - Cross Correlation Shifts/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

xoff, yoff = cross\_correlation\_shifts(ref\_image, offset\_image)

corrected\_image = shift(offset\_image, shift=(-yoff, -xoff), mode='constant')

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s\tExecution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

11. cross correlation - Chi2.py

# -\*- coding: utf-8 -\*-

"""

Created on Fri Oct 9 21:24:32 2020

@author: tsoyg

"""

import time

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

from image\_registration import chi2\_shift

from scipy.ndimage import shift

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test2 - Chi2/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

noise = 0.1

xoff, yoff, exoff, eyoff = chi2\_shift(ref\_image, offset\_image, noise,

return\_error=True, upsample\_factor='auto')

corrected\_image = shift(offset\_image, shift=(-yoff, -xoff), mode='constant')

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s\tExecution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

12. Optical Flow Based Shift.py

# -\*- coding: utf-8 -\*-

"""

Created on Sat Oct 10 16:44:57 2020

@author: tsoyg

"""

import time

from skimage import io

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

from skimage import registration

from scipy.ndimage import shift

import numpy as np

import os

from os import listdir

from os.path import isfile, join

total\_time = time.time()

pathIn = "Test Sequence" # откуда брать файлы

pathOut = "Test Sequence Out/test4 - Optical Flow Based Shift/" # куда сохранять файлы

files = [f for f in listdir(pathIn) if isfile(join(pathIn, f))] # сами картинки

if not os.path.exists(pathOut): os.makedirs(pathOut) # создать путь, если ещё нет

ref\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[0])))) # задаём эталон

io.imsave(pathOut + files[0], img\_as\_ubyte(ref\_image)) # сохранить первый файл

sko = 0

sko\_arr = [None] \* (len(files) - 1)

time\_arr = [None] \* (len(files) - 1)

for i in range(1, len(files) - 1):

# -------Процесс согласования НАЧАЛО------#

start\_time = time.time()

offset\_image = img\_as\_ubyte(rgb2gray(io.imread(join(pathIn, files[i]))))

flow = registration.optical\_flow\_tvl1(ref\_image, offset\_image)

flow\_x = flow[1, :, :]

flow\_y = flow[0, :, :]

xoff = np.mean(flow\_x)

yoff = np.mean(flow\_y)

corrected\_image = shift(offset\_image, shift=(-yoff, -xoff), mode='constant')

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

io.imsave(pathOut + files[i], corrected\_image)

time\_arr[i - 1] = time.time() - start\_time

# -------Процесс согласования КОНЕЦ-------#

print("--- %s seconds ---" % (time.time() - start\_time))

sko\_arr[i - 1] = np.sum((corrected\_image.astype("float") - ref\_image.astype("float")) \*\* 2)

sko\_arr[i - 1] /= float(ref\_image.shape[0] \* ref\_image.shape[1])

print("SKO[" + str(i) + "]: " + str(sko\_arr[i - 1]) + "\n")

sko += sko\_arr[i - 1]

# -------------Рассчёт СКО----------------#

sko = sko / (len(files) - 1)

sko\_arr[len(files) - 2] = "Average SKO: " + str(sko)

time\_arr[len(files) - 2] = "Total Time: " + str(time.time() - total\_time)

with open(pathOut + 'SKO.txt', 'w') as f:

for i in range(1, len(sko\_arr) + 1):

if (i != 50):

f.write("SKO№%s:\t%s\tExecution time %s\n" % (i, sko\_arr[i - 1], time\_arr[i - 1]))

else:

f.write("\t%s\t%s\n" % (sko\_arr[i - 1], time\_arr[i - 1]))

# ------------Запись в файл---------------#

print("Average SKO: " + str(sko))

print(str(time\_arr[len(files) - 2]))

Приложение Б

Полный листинг модуля графического интерфейса

main gui.py

from tkinter import \*

from tkinter import filedialog

from tkinter import ttk

from win32api import GetSystemMetrics

from skimage import io

from os import listdir

from PIL import Image, ImageTk

import os

import imageio

def stream():

try:

try:

"""

анимация видео-файла

"""

delay = int(1000 / video.get\_meta\_data()['fps'])

image = video.get\_next\_data()

if int(len(image[0])) > 300:

axe\_y = 300

axe\_x = int(len(image[0]) / (len(image) / 300))

else:

axe\_y = int(len(image) / (len(image[0]) / 560))

axe\_x = 560

frame\_image = Image.fromarray(image).resize((axe\_x, axe\_y), Image.ANTIALIAS)

frame\_image = ImageTk.PhotoImage(frame\_image)

media\_label.config(image=frame\_image)

media\_label.image = frame\_image

media\_label.after(delay, lambda: stream())

except:

"""

анимация gif-файла

"""

image = video.get\_next\_data()

if int(len(image[0])) > 300:

axe\_y = 300

axe\_x = int(len(image[0]) / (len(image) / 300))

else:

axe\_y = int(len(image) / (len(image[0]) / 560))

axe\_x = 560

frame\_image = Image.fromarray(image).resize((axe\_x, axe\_y), Image.ANTIALIAS)

frame\_image = ImageTk.PhotoImage(frame\_image)

media\_label.config(image=frame\_image)

media\_label.image = frame\_image

media\_label.after(40, lambda: stream())

except:

print("Видео закончилось")

video.close()

return

def limit\_expansion(\*args):

"""

форматно-логический контроль поля ввода

"""

value = filed\_limit\_expansion.get()

if len(value) > 2:

filed\_limit\_expansion.set(value[:2])

try:

int(value)

except:

field\_expand.delete(len(value) - 1, END)

def btn\_search\_click():

"""

поиск файла в системе

"""

global media\_label

global video

root.filename = filedialog.askopenfilename(title='Выберите видео файл',

filetypes=(("Видео файлы", "\*.mp4 .wmv .avi .gif"),

("Все файлы", "\*.\*")))

field\_choose\_filename.delete(0, END)

field\_choose\_filename.insert(0, root.filename)

if field\_choose\_filename.get() != "":

"""

попытка воспроизвести анимацию

"""

media\_label.destroy()

video\_name = field\_choose\_filename.get()

media\_label = Label(frame\_media)

media\_label.pack()

try:

progress\_label\_stage.pack\_forget()

video = imageio.get\_reader(video\_name)

stream()

except:

progress\_label\_stage.pack(pady=10, expand=1, anchor=S)

progress\_label\_stage['text'] = "Файл не найден или формат файла не поддерживается"

root.update\_idletasks()

print("Файл не найден или формат файла не поддерживается")

def btn\_check\_result\_folder():

"""

открыть папку с сохранениями

"""

import subprocess

explorer\_path = os.path.join(os.getenv('WINDIR'), 'explorer.exe')

path = os.path.normpath(os.getcwd() + "\\\_RESULTS\\")

subprocess.run([explorer\_path, path])

def image\_sequence():

"""

подготовка файла для начала обработки

"""

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte

images = []

try:

while 1:

images.append(img\_as\_ubyte(rgb2gray(video.get\_next\_data())))

except:

return images

def about():

def git\_link():

import webbrowser

url = "https://github.com/ts-glob/Optimal-Image-Super-resolution-Image-registration"

webbrowser.open(url, new=2)

about = Toplevel()

about.title('О программе')

about.resizable(0, 0)

about.geometry('400x200')

about['bg'] = '#0f7850'

font\_color = '#f06e71'

about\_text = Label(about)

about\_text['text'] = "Программа разработана студентом самарского университета \n" \

"Цоем Глебом в 2021 году в рамках дипломной работы"

about\_text['bg'] = font\_color

about\_text.pack(pady=10, expand=1)

btn\_git = Button(about, text='GITHUB ->', command=git\_link)

btn\_git['bg'] = font\_color

btn\_git.pack(padx=10, expand=1)

def instructions():

instructions = Toplevel()

instructions.resizable(0, 0)

instructions.title('Использование программы')

instructions.geometry('700x520')

instructions\_text = Label(instructions, justify='left')

instructions\_text[

'text'] = "Программа предназначена для получения изображения высокого разрешения с восстановлением качества по \n" \

"серии кадров изображений низкого разрешения. Восстановление обеспечивается взвешенным суммированием \n" \

"всех пикселей. \n\n" \

"В той же директории, что и .exe файл во время запуска программы создаётся новая папка: \n " + os.getcwd() + \

"\\\"\_RESULTS\"\n " \

"для хранения результатов. Для получения корректного результата алгоритма сверхразрешения \n" \

"необходимо выполнить следующие шаги: \n\n" \

"1. Выберите видео/анимированное изображение. \n" \

" В окне должна отобразиться анимация. \n" \

"2. Задайте желаемое увеличение. Исходное видео/анимация будет увеличено во столько раз, сколько будет задано. \n" \

"3. Задайте способ предобработки. В качестве предобработки используются методы фильтрации изображений. \n" \

"\t а) фильтр винера; \n" \

"\t б) фильтр гаусса; \n" \

"\t в) медианный фильтр; \n" \

"\t г) адаптивное контрастирование; \n" \

"\t д) повышение резкости; \n" \

"\t е) уменьшение шума; \n" \

"\t ж) вейвлет-шумоподавление; \n" \

"\t Возможна работа без предобработки.\n" \

"4. Нажмите на кнопку \"ПРИМЕНИТЬ СВЕРХРАЗРЕШЕНИЕ\". Дождитесь обработки. \n" \

"\t а) первый этап соответствует предобработке; \n" \

"\t б) второй - увеличению; \n" \

"\t в) третий - вычислению доп канала ошибки интерполяции; \n" \

"\t г) четвёртый - согласованию кадров; \n" \

"\t д) пятый - суммированию всех кадров в одно изображение; \n" \

"Если всё прошло успешно, то отобразится результат работы. Сгенерированный файл сохранится в папку \n " + os.getcwd() + \

"\\\"\_RESULTS\" \n" \

"Данную папку можно быстро открыть по кнопке меню \"Показать в папке\". Дальше возможно продолжать \n" \

"работу с программой с новыми данными."

instructions\_text.pack(pady=20, side=TOP, anchor="w")

def algorithm(images, expand\_by, filtration\_mode, progress\_bar\_info):

"""

основной алгоритм метода реализованный в связанных модулях

"""

import filtration

import expansion

import new\_channel

import registration

import fusing

progress\_label\_stage['text'] = "ЭТАП 1/5"

images = filtration.filtration\_gui\_main(images, filtration\_mode, progress\_bar\_info)

progress\_label\_stage['text'] = "ЭТАП 2/5"

images = expansion.expansion\_gui(images, expand\_by, progress\_bar\_info)

progress\_label\_stage['text'] = "ЭТАП 3/5"

additional\_channel = new\_channel.additional\_channel\_gui(images, expand\_by, progress\_bar\_info)

progress\_label\_stage['text'] = "ЭТАП 4/5"

images, additional\_channel = registration.registration\_gui(images, additional\_channel, progress\_bar\_info)

progress\_label\_stage['text'] = "ЭТАП 5/5"

result\_image = fusing.fusing\_gui(images, additional\_channel, progress\_bar\_info)

return result\_image

def btn\_process\_click():

"""

запрос на запуск основного алгоритма

"""

global media\_label

global video

if field\_choose\_filename.get() != "" and field\_expand.get() != "":

video\_name = field\_choose\_filename.get()

expand\_by = int(field\_expand.get())

filtration\_mode = filter\_method.get()

media\_label.pack\_forget()

root.update\_idletasks()

try:

"""

попытка подготовки файла и последующий запуск алгоритма

"""

progress\_label\_stage['text'] = ""

progress\_label\_stage.pack\_forget()

root.update\_idletasks()

video = imageio.get\_reader(video\_name)

images = image\_sequence()

progress\_label\_stage.pack(pady=10, expand=1, anchor=S)

progress\_bar.pack(pady=0, expand=0, anchor=S)

progress\_label.pack(pady=0, expand=0, anchor=S)

progress\_bar\_info = [progress\_bar, progress\_label, root]

result\_image = algorithm(images, expand\_by, filtration\_mode, progress\_bar\_info)

progress\_label\_stage['text'] = "Сохранение файла..."

root.update\_idletasks()

io.imsave(pathOut + str(len(listdir(pathOut)) + 1) + ".png", result\_image)

progress\_label\_stage['text'] = 'Файл сохранен в ' + pathOut + str(len(listdir(pathOut))) + ".png"

progress\_bar.pack\_forget()

progress\_label.pack\_forget()

root.update\_idletasks()

media\_label.destroy()

video\_name = pathOut + str(len(listdir(pathOut))) + ".png"

media\_label = Label(frame\_media)

media\_label.pack()

video = imageio.get\_reader(video\_name)

stream()

except:

"""

в случае любой ошибки (файл не подходит или ошибка в модулях алгоритма)

происходит вывод текста и прекращение обработки

"""

progress\_label\_stage.pack\_forget()

progress\_bar.pack\_forget()

progress\_label.pack\_forget()

progress\_label\_stage.pack(pady=10, expand=1, anchor=S)

progress\_label\_stage['text'] = "Файл не найден или формат файла не поддерживается"

root.update\_idletasks()

print("Файл не найден или формат файла не поддерживается")

else:

"""

в случае незаполненных полей происходит вывод ошибки

"""

media\_label.pack\_forget()

progress\_label\_stage.pack(pady=10, expand=1, anchor=S)

progress\_label\_stage['text'] = "Введите все обязательные параметры"

root.update\_idletasks()

"""

инициализация интерфейса

"""

pathOut = os.getcwd() + "/\_RESULTS/"

if not os.path.exists(pathOut):

os.makedirs(pathOut)

root = Tk()

root.resizable(0, 0)

root.geometry('800x600+' + str(int(GetSystemMetrics(0) / 2) - 400) + '+' + str(int(GetSystemMetrics(1) / 2) - 300))

root.title('Сверхразрешение')

root['bg'] = '#0d47a1'

bg\_media = '#f8f8f8'

bg\_menu = '#2196f3'

frame\_menu1 = Frame(root, bg=bg\_menu)

frame\_menu2 = Frame(root, bg=bg\_menu)

frame\_menu3 = Frame(root, bg=bg\_menu)

frame\_media = Frame(root, bg=bg\_media)

frame\_menu1.place(relx=0.10, rely=0.10, relwidth=0.80, relheight=0.20)

frame\_menu2.place(relx=0.10, rely=0.20, relwidth=0.35, relheight=0.10)

frame\_menu3.place(relx=0.45, rely=0.20, relwidth=0.45, relheight=0.10)

frame\_media.place(relx=0.10, rely=0.30, relwidth=0.80, relheight=0.60)

menu\_bar = Menu(root)

root.config(menu=menu\_bar)

menu\_info = Menu(menu\_bar)

menu\_bar.add\_cascade(label="Инфо", menu=menu\_info)

menu\_info.add\_command(label="О программе", command=about)

menu\_info.add\_command(label="Инструкция", command=instructions)

menu\_info.add\_command(label="Показать в папке", command=btn\_check\_result\_folder)

label\_choose\_filename = Label(frame\_menu1, text='Выберите видео файл')

field\_choose\_filename = Entry(frame\_menu1, width=60)

btn\_search = Button(frame\_menu1, text='Обзор', command=btn\_search\_click)

label\_expand = Label(frame\_menu2, text=' Во сколько раз \n увеличить фото ')

filed\_limit\_expansion = StringVar()

filed\_limit\_expansion.trace('w', limit\_expansion)

field\_expand = Entry(frame\_menu2, width=5, textvariable=filed\_limit\_expansion)

label\_filtration = Label(frame\_menu3, text=' Какой фильтр \n использовать ')

filter\_method = StringVar()

filter\_method.set("Без предобработки")

filter\_method\_drop\_down = OptionMenu(frame\_menu3, filter\_method, "Без предобработки", "Винер", "Гаусс", "Медианный",

"Контраст", "Резкость", "Шумоподавление", "Обратная свёртка", "Вейвлет")

btn\_process = Button(frame\_media, text='ПРИМЕНИТЬ СВЕРХРАЗРЕШЕНИЕ', command=btn\_process\_click)

media\_label = Label(frame\_media)

progress\_bar = ttk.Progressbar(frame\_media, orient=HORIZONTAL, length=300, mode='determinate')

progress\_label = Label(frame\_media, text="")

progress\_label\_stage = Label(frame\_media, text="")

label\_choose\_filename.pack(side=LEFT, padx=20, pady=20, anchor=NE)

field\_choose\_filename.pack(side=LEFT, padx=10, pady=20, anchor=NE)

btn\_search.pack(side=LEFT, padx=10, pady=20, anchor=NE)

label\_expand.pack(side=LEFT, padx=20, pady=15, anchor=NE)

field\_expand.pack(side=LEFT, padx=10, pady=22, anchor=NE)

label\_filtration.pack(side=LEFT, padx=10, pady=15, anchor=NE)

filter\_method\_drop\_down.pack(side=LEFT, padx=20, pady=15, anchor=NE)

btn\_process.pack(padx=105, pady=15, side=BOTTOM)

root.mainloop()

Приложение В

Полный листинг модуля предобработки

filtration.py

# -\*- coding: utf-8 -\*-

"""

Created on Sat Dec 5 13:57:29 2020

@author: tsoyg

"""

from tqdm import tqdm

import numpy as np

from skimage import img\_as\_ubyte, img\_as\_float

from skimage.color import rgb2gray

# from skimage import io

def wiener(files, progress\_bar, progress\_label, root):

from skimage import restoration

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

psf = np.ones((5, 5)) / 25

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

img = img\_as\_float(rgb2gray(files[i]))

restored\_img, \_ = restoration.unsupervised\_wiener(img, psf)

restored\_img = img\_as\_ubyte(

(restored\_img - np.min(restored\_img)) / (np.max(restored\_img) - np.min(restored\_img)))

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def gauss(files, progress\_bar, progress\_label, root):

from scipy import ndimage

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

restored\_img = ndimage.gaussian\_filter(files[i], 2)

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def median(files, progress\_bar, progress\_label, root):

from scipy import ndimage

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

restored\_img = ndimage.median\_filter(files[i], 3)

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def contrast(files, progress\_bar, progress\_label, root):

import cv2

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

restored\_img = clahe.apply(files[i])

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def sharpen(files, progress\_bar, progress\_label, root):

import cv2

sharpen\_mask = np.array([[-1, -1, -1],

[-1, 9, -1],

[-1, -1, -1]])

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

restored\_img = cv2.filter2D(files[i], -1, sharpen\_mask)

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def denoise(files, progress\_bar, progress\_label, root):

from skimage.restoration import denoise\_nl\_means, estimate\_sigma

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

patch\_kw = dict(patch\_size=2, patch\_distance=2)

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

sigma\_est = np.mean(estimate\_sigma(files[i]))

img = img\_as\_float(files[i])

restored\_img = denoise\_nl\_means(img, h=0.8 \* sigma\_est, sigma=sigma\_est, fast\_mode=False, \*\*patch\_kw)

restored\_img = img\_as\_ubyte(

(restored\_img - np.min(restored\_img)) / (np.max(restored\_img) - np.min(restored\_img)))

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def deconvolution(files, progress\_bar, progress\_label, root):

from skimage import restoration

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

psf = np.ones((3, 3)) / 25

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

img = img\_as\_float(rgb2gray(files[i]))

restored\_img = restoration.richardson\_lucy(img, psf)

restored\_img = img\_as\_ubyte(

(restored\_img - np.min(restored\_img)) / (np.max(restored\_img) - np.min(restored\_img)))

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def wavelet(files, progress\_bar, progress\_label, root):

from skimage.restoration import denoise\_wavelet

progress\_step = 100 / len(files)

progress\_bar['value'] = 0

progress\_label.config(text="0")

root.update\_idletasks()

result\_array = []

for i in tqdm(range(0, len(files)), desc="Фильтрация: "):

img = img\_as\_float(rgb2gray(files[i]))

restored\_img = denoise\_wavelet(img, rescale\_sigma=True)

restored\_img = img\_as\_ubyte(

(restored\_img - np.min(restored\_img)) / (np.max(restored\_img) - np.min(restored\_img)))

result\_array.append(restored\_img)

progress\_bar['value'] += progress\_step

progress\_label.config(text=round(progress\_bar['value']))

root.update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", restored\_img)

progress\_bar['value'] = 100

progress\_label.config(text=progress\_bar['value'])

root.update\_idletasks()

return result\_array

def filtration\_gui\_main(files, mode, progress\_bar\_info):

result\_array = []

if mode == "Без предобработки":

result\_array = files

if mode == "Винер":

result\_array = wiener(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

if mode == "Гаусс":

result\_array = gauss(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

if mode == "Медианный":

result\_array = median(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

if mode == "Контраст":

result\_array = contrast(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

if mode == "Резкость":

result\_array = sharpen(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

if mode == "Шумоподавление":

result\_array = denoise(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

if mode == "Обратная свёртка":

result\_array = deconvolution(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

if mode == "Вейвлет":

result\_array = wavelet(files, progress\_bar\_info[0], progress\_bar\_info[1], progress\_bar\_info[2])

return result\_array

Приложение Г

Полный листинг модуля интерполяции

expansion.py

# -\*- coding: utf-8 -\*-

"""

Created on Sat Dec 5 13:57:29 2020

@author: tsoyg

"""

from tqdm import tqdm

import cv2

from skimage import img\_as\_ubyte, img\_as\_float

from skimage.color import rgb2gray

from skimage import io

def expansion\_gui(files, expand\_by, progress\_bar\_info):

progress\_step = 100 / len(files)

progress\_bar\_info[0]['value'] = 0

progress\_bar\_info[1].config(text="0")

progress\_bar\_info[2].update\_idletasks()

result\_array = []

for i in tqdm(range(0, len(files)), desc="Увеличение размерности: "):

original\_img = img\_as\_ubyte(rgb2gray(files[i]))

rows = len(original\_img)

columns = len(original\_img[0])

expanded\_img = cv2.resize(original\_img, (columns \* expand\_by, rows \* expand\_by))

result\_array.append(expanded\_img)

progress\_bar\_info[0]['value'] += progress\_step

progress\_bar\_info[1].config(text=round(progress\_bar\_info[0]['value']))

progress\_bar\_info[2].update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", expanded\_img)

progress\_bar\_info[0]['value'] = 100

progress\_bar\_info[1].config(text=progress\_bar\_info[0]['value'])

progress\_bar\_info[2].update\_idletasks()

return result\_array

Приложение Д

Полный листинг модуля вычисления ошибки интерполяции

new\_channel.py

# -\*- coding: utf-8 -\*-

"""

Created on Sat Dec 5 13:57:29 2020

@author: tsoyg

"""

from tqdm import tqdm

from skimage import img\_as\_ubyte, img\_as\_float

from skimage.color import rgb2gray

import numpy as np

from math import log

def additional\_channel\_gui(files, expand\_by, progress\_bar\_info):

progress\_step = 100 / len(files)

progress\_bar\_info[0]['value'] = 0

progress\_bar\_info[1].config(text="0")

progress\_bar\_info[2].update\_idletasks()

lil\_array = [[[0 for l in range(expand\_by)] for ll in range(expand\_by)] for lll in range(len(files))]

result\_array = [[[0 for l in range(files[0].shape[1])] for ll in range(files[0].shape[0])] for lll in

range(len(files))]

T = expand\_by

L = expand\_by

rho = 0.9

A = (-log(rho)) / T

for i in range(0, len(files)):

Dx = np.var(img\_as\_ubyte(rgb2gray(files[i])))

for a in range(0, L):

for b in range(0, L):

lil\_array[i][a][b] = 2 \* Dx \* A \* (a + b - ((a \* a) / T) - ((b \* b) / T))

for i in tqdm(range(0, len(files)), desc="Доп канал ошибки интерполяции: "):

temp\_a = 0

for a in range(0, files[0].shape[0]):

temp\_b = 0

if a % L == 1:

temp\_a += L

for b in range(0, files[0].shape[1]):

result\_array[i][a][b] = lil\_array[i][a - temp\_a][b - temp\_b]

if b % L == 1:

temp\_b += L

progress\_bar\_info[0]['value'] += progress\_step

progress\_bar\_info[1].config(text=round(progress\_bar\_info[0]['value']))

progress\_bar\_info[2].update\_idletasks()

progress\_bar\_info[0]['value'] = 100

progress\_bar\_info[1].config(text=progress\_bar\_info[0]['value'])

progress\_bar\_info[2].update\_idletasks()

return result\_array

Приложение Е

Полный листинг модуля согласования

registration.py

# -\*- coding: utf-8 -\*-

"""

Created on Tue Oct 13 18:10:49 2020

@author: tsoyg

"""

from tqdm import tqdm

import numpy as np

from pystackreg import StackReg

from skimage.color import rgb2gray

from skimage import img\_as\_ubyte, img\_as\_float

from skimage import io

def registration\_gui(files, additional\_channel, progress\_bar\_info):

progress\_step = 100 / (len(files))

progress\_bar\_info[0]['value'] = 0

progress\_bar\_info[1].config(text="0")

progress\_bar\_info[2].update\_idletasks()

result\_img\_stack = []

ref\_image = img\_as\_ubyte(rgb2gray(files[0])) # задаём эталон

result\_img\_stack.append(ref\_image) # сохранить первый файл

for i in tqdm(range(1, len(files)), desc="Согласование: "):

offset\_image = img\_as\_ubyte(rgb2gray(files[i]))

reg\_instance1 = StackReg(StackReg.AFFINE)

reg\_instance1.register(ref\_image, offset\_image)

corrected\_image = reg\_instance1.transform(offset\_image)

additional\_channel[i] = reg\_instance1.transform(additional\_channel[i])

corrected\_image = img\_as\_ubyte(

(corrected\_image - np.min(corrected\_image)) / (np.max(corrected\_image) - np.min(corrected\_image)))

result\_img\_stack.append(corrected\_image)

progress\_bar\_info[0]['value'] += progress\_step

progress\_bar\_info[1].config(text=round(progress\_bar\_info[0]['value']))

progress\_bar\_info[2].update\_idletasks()

# io.imsave("temp/" + str(i) + ".jpg", corrected\_image)

progress\_bar\_info[0]['value'] = 100

progress\_bar\_info[1].config(text=progress\_bar\_info[0]['value'])

progress\_bar\_info[2].update\_idletasks()

return result\_img\_stack, additional\_channel

Приложение Ж

Полный листинг модуля комплексирования

fusing.py

# -\*- coding: utf-8 -\*-

"""

Created on Thu Nov 5 13:33:27 2020

@author: tsoyg

"""

from tqdm import tqdm

import numpy as np

from skimage import img\_as\_ubyte, img\_as\_float

from skimage.color import rgb2gray

from skimage import io

def fusing\_gui(files, additional\_channel, progress\_bar\_info):

import math

progress\_step = 100 / (files[0].shape[0] \* files[0].shape[1]) / 2

progress\_bar\_info[0]['value'] = 0

progress\_bar\_info[1].config(text="0")

progress\_bar\_info[2].update\_idletasks()

a = 0

b = 0

ones = np.ones((additional\_channel[1].shape[0], additional\_channel[1].shape[1]))

# error\_helper = np.ones((additional\_channel[1].shape[0], additional\_channel[1].shape[1])) / 1000000000

for m in tqdm(range(0, len(files)), desc="Комплексирование: "):

offset\_img = img\_as\_ubyte(rgb2gray(files[m]))

disp\_err = additional\_channel[m]# + error\_helper

a += offset\_img / disp\_err

b += ones / disp\_err

progress\_bar\_info[0]['value'] += progress\_step

progress\_bar\_info[1].config(text=round(progress\_bar\_info[0]['value']))

progress\_bar\_info[2].update\_idletasks()

pixel\_matrix = a / b

for i in range(0, pixel\_matrix.shape[0]):

for j in range(0, pixel\_matrix.shape[1]):

flag\_nan = True

if additional\_channel[0][i][j] == 0:

pixel\_matrix[i][j] = files[0][i][j]

if pixel\_matrix[i][j] == 0:

pixel\_matrix[i][j] = files[0][i][j]

if math.isnan(pixel\_matrix[i][j]):

for m in range(0, len(files)):

if flag\_nan:

flag\_nan = False

pixel\_matrix[i][j] = 0

pixel\_matrix[i][j] += files[m][i][j] / len(files)

if pixel\_matrix[i][j] < 0.0000005 or pixel\_matrix[i][j] > 255:

pixel\_matrix[i][j] = files[0][i][j]

progress\_bar\_info[0]['value'] += progress\_step

progress\_bar\_info[1].config(text=round(progress\_bar\_info[0]['value']))

progress\_bar\_info[2].update\_idletasks()

pixel\_matrix = img\_as\_ubyte((pixel\_matrix - np.min(pixel\_matrix)) / (np.max(pixel\_matrix) - np.min(pixel\_matrix)))

progress\_bar\_info[0]['value'] = 100

progress\_bar\_info[1].config(text=progress\_bar\_info[0]['value'])

progress\_bar\_info[2].update\_idletasks()

return pixel\_matrix

Приложение И

Полный листинг модуля комплексирования

automated\_warp.py

import imageio

from os import listdir

import os

import numpy as np

def image\_sequence():

from skimage.color import rgb2gray

from skimage import img\_as\_float, img\_as\_ubyte

images = []

try:

while 1:

img = img\_as\_float(rgb2gray(video.get\_next\_data()))

dec\_img = img[::10, ::10]

res = img\_as\_ubyte((dec\_img - np.min(dec\_img)) / (np.max(dec\_img) - np.min(dec\_img)))

images.append(res)

except:

imageio.mimsave(save\_path + str(len(listdir(save\_path)) + 1) + ".gif", images)

save\_path = "искаженная последовательность/уменьшение в 10 раз/"

if not os.path.exists(save\_path): os.makedirs(save\_path)

video\_name = "D:/my docs/stud/\_ДИПЛОМНАЯ РАБОТА/Test Video/5.gif"

video = imageio.get\_reader(video\_name)

image\_sequence()

automated\_unweighted\_main.py

import automated\_expansion

import automated\_registration

import automated\_fusing

import imageio

def image\_sequence():

try:

while 1:

images.append(video.get\_next\_data())

except:

pass

expand\_by = '10'

file\_num = '5'

video\_name = "D:/my docs/stud/\_ДИПЛОМНАЯ РАБОТА/Автоматизация/Невзвешенное комплексирование/" \

"искаженная последовательность/уменьшение в " + expand\_by + " раз/" + file\_num + ".gif"

pathOut = "невзвешенное комплексирование/восстановление в " + expand\_by + " раз/"

images = []

video = imageio.get\_reader(video\_name)

image\_sequence()

images = automated\_expansion.expansion(images, "D:\\my docs\\stud\\\_ДИПЛОМНАЯ РАБОТА\\Test Video\\" + file\_num + ".gif")

images = automated\_registration.registration(images)

automated\_fusing.restoration(images, pathOut)

automated\_expansion.py

from tqdm import tqdm

import cv2

import imageio

def expansion(images, video\_name):

expanded\_images = []

video = imageio.get\_reader(video\_name)

original\_img = video.get\_next\_data()

rows = len(original\_img)

columns = len(original\_img[0])

for i in tqdm(range(0, len(images)), desc="Увеличение размерности: "):

warped\_img = images[i]

warped\_img = cv2.resize(warped\_img, (columns, rows))

expanded\_images.append(warped\_img)

return expanded\_images

automated\_registration.py

from tqdm import tqdm

from pystackreg import StackReg

def registration(images):

registered\_images = []

ref\_image = images[0] # задаём эталон

registered\_images.append(ref\_image)

for i in tqdm(range(1, len(images)), desc="Согласование: "):

offset\_image = images[i]

reg\_instance = StackReg(StackReg.AFFINE)

offset\_image = reg\_instance.register\_transform(ref\_image, offset\_image)

registered\_images.append(offset\_image)

return registered\_images

automated\_fusing.py

import os

from os import listdir

from tqdm import tqdm

import numpy as np

from skimage import io

from skimage import img\_as\_ubyte

def restoration(images, pathOut):

if not os.path.exists(pathOut): os.makedirs(pathOut)

a = 0

for i in tqdm(range(0, len(images)), desc="Комплексирование: "):

img = images[i]

a += img / len(images)

print('Сохранение файла...')

a = img\_as\_ubyte((a - np.min(a)) / (np.max(a) - np.min(a)))

io.imsave(pathOut + str(len(listdir(pathOut)) + 1) + ".png", a)

automated\_mse.py

import numpy as np

from skimage import img\_as\_ubyte

from skimage.color import rgb2gray

from skimage import io

import imageio

import cv2

def sko():

video = imageio.get\_reader(orig\_vid)

orig\_img = img\_as\_ubyte(rgb2gray(video.get\_next\_data()))

unweighted\_img = img\_as\_ubyte(rgb2gray(io.imread(unweighted)))

sko\_arr = np.sum((unweighted\_img.astype("float") - orig\_img.astype("float")) \*\* 2)

sko\_arr /= float(orig\_img.shape[0] \* orig\_img.shape[1])

sko\_arr = sko\_arr / (np.var(img\_as\_ubyte(rgb2gray(orig\_img))) \*\* 2)

string = str(sko\_arr).replace(".", ",")

print("СКО невзвешенного суммирования относительно искаженного видео составляет", string)

weighted\_img = img\_as\_ubyte(rgb2gray(io.imread(weighted)))

weighted\_img = cv2.resize(weighted\_img, (orig\_img.shape[1], orig\_img.shape[0]))

sko\_arr = np.sum((weighted\_img.astype("float") - orig\_img.astype("float")) \*\* 2)

sko\_arr /= float(orig\_img.shape[0] \* orig\_img.shape[1])

sko\_arr = sko\_arr / (np.var(img\_as\_ubyte(rgb2gray(orig\_img))) \*\* 2)

string = str(sko\_arr).replace(".", ",")

print("СКО взвешенного суммирования относительно искаженного видео составляет", string)

file\_num = '5'

orig\_vid = "D:/my docs/stud/\_ДИПЛОМНАЯ РАБОТА/Test Video/" + file\_num + ".gif"

unweighted = "D:\\my docs\\stud\\\_ДИПЛОМНАЯ РАБОТА\\Автоматизация\\Невзвешенное комплексирование\\" \

"невзвешенное комплексирование\\восстановление в 8 раз\\" + file\_num + ".png"

weighted = "D:\\my docs\\stud\\\_ДИПЛОМНАЯ РАБОТА\\Автоматизация\\" \

"Оптимальное комплексирование\\восстановление в 8 раз\\" + file\_num + ".png"

sko()

Приложение К

Результаты экспериментов сравнения эффективности разработанного метода взвешенного комплексирования с существующим методом невзвешенного комплексирования

В таблицах приведены отношения СКО изображений, полученных рассматриваемым методом к СКО оригинальных кадров выборок.

Таблица 2 – сравнение СКО применяемых методов для выборки с шахматной доской

|  |  |  |  |
| --- | --- | --- | --- |
| № | Обрабатываемое видео | Шахматная доска | |
| СКО Невзвешенного комплексирования | СКО Взвешенного комплексирования |
| 1 | Уменьшение в 2 раза | 0,000119092 | 9,63245E-05 |
| 2 | Уменьшение в 4 раза | 0,000186888 | 0,000172073 |
| 3 | Уменьшение в 6 раз | 0,000210165 | 0,000198699 |
| 4 | Уменьшение в 8 раз | 0,000198958 | 0,000207858 |
| 5 | Уменьшение в 10 раз | 0,000266512 | 0,000264059 |

Таблица 3 – сравнение СКО применяемых методов для выборки с автомобилем

|  |  |  |  |
| --- | --- | --- | --- |
| № | Обрабатываемое видео | Автомобиль | |
| СКО Невзвешенного комплексирования | СКО Взвешенного комплексирования |
| 1 | Уменьшение в 2 раза | 0,000436152 | 0,000346541 |
| 2 | Уменьшение в 4 раза | 0,000568714 | 0,000502019 |
| 3 | Уменьшение в 6 раз | 0,000657515 | 0,000498116 |
| 4 | Уменьшение в 8 раз | 0,000681284 | 0,000635593 |
| 5 | Уменьшение в 10 раз | 0,001187377 | 0,000783304 |

Таблица 4 – сравнение СКО применяемых методов для выборки с телефоном

|  |  |  |  |
| --- | --- | --- | --- |
| № | Обрабатываемое видео | Телефон | |
| СКО Невзвешенного комплексирования | СКО Взвешенного комплексирования |
| 1 | Уменьшение в 2 раза | 9,93288E-06 | 8,1256E-06 |
| 2 | Уменьшение в 4 раза | 1,73717E-05 | 1,77864E-05 |
| 3 | Уменьшение в 6 раз | 3,64188E-05 | 3,47329E-05 |
| 4 | Уменьшение в 8 раз | 4,91084E-05 | 4,90691E-05 |
| 5 | Уменьшение в 10 раз | 6,40426E-05 | 6,3863E-05 |

Таблица 5 – сравнение СКО применяемых методов для выборки с человеком

|  |  |  |  |
| --- | --- | --- | --- |
| № | Обрабатываемое видео | Лицо | |
| СКО Невзвешенного комплексирования | СКО Взвешенного комплексирования |
| 1 | Уменьшение в 2 раза | 0,000822034 | 0,00071996 |
| 2 | Уменьшение в 4 раза | 0,000829753 | 0,00067356 |
| 3 | Уменьшение в 6 раз | 0,000825481 | 0,000710101 |
| 4 | Уменьшение в 8 раз | 0,000819033 | 0,00077617 |
| 5 | Уменьшение в 10 раз | 0,000819291 | 0,000726443 |

Таблица 6 – сравнение СКО применяемых методов для выборки с печатным текстом

|  |  |  |  |
| --- | --- | --- | --- |
| № | Обрабатываемое видео | Печатный документ | |
| СКО Невзвешенного комплексирования | СКО Взвешенного комплексирования |
| 1 | Уменьшение в 2 раза | 0,00182013 | 0,001350417 |
| 2 | Уменьшение в 4 раза | 0,001628824 | 0,001511656 |
| 3 | Уменьшение в 6 раз | 0,001571504 | 0,001544889 |
| 4 | Уменьшение в 8 раз | 0,002520745 | 0,002039292 |
| 5 | Уменьшение в 10 раз | 0,005247354 | 0,004424003 |