Московский государственный университет имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра интеллектуальных информационных технологий

Зеленцов Алексей Викторович

Разработка метода оценки качества исправления чересстрочного видео

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:

к.ф.-м.н., младший научный сотрудник

Михаил Викторович Ерофеев

Москва, 2022

Разработка метода оценки качества исправления чересстрочного видео.

Зеленцов Алексей

Чересстрочная развертка — это способ уменьшить количество информации видео в 2 раза, не теряя при этом частоту кадров. Однако при применении этой техники на видео возникают артефакты — в частности эффект ”гребенки” и ”призрачный” эффект. Для устранения этих артефактов и восстановления утерянной половины информации используются алгоритмы исправления чересстрочности. В данной работе разрабатывается набор данных и система для сравнения качества этих алгоритмов. Кроме того, в данной работе предлагается метод оценки качества исправления чересстрочного видео.

Deinterlacing quality assessment.

Zelentsov Aleksei

Interlacing is a way to reduce the amount of video information by a factor of 2 without losing frame rate. However, when applying this technique, artifacts appear on the video — in particular the ”comb” effect and the ”ghost” effect. To eliminate these artifacts and restore the lost half of the information, deinterlacing algorithms are used. In this paper, we develop a dataset and a system for comparing the quality of these algorithms. Besides, this paper proposes a method for assessing the quality of interlaced video correction.

Содержание

1. Введение

С каждым годом неуклонно растет количество видеотрафика, передаваемого по сети. Исследования компании Invideo [1] показывают, что в 2021 году 83.8% всех пользователей интернета просматривали видеоконтент. На Рис. 1 показана доля пользователей сети интернет, которые скачивали видео.

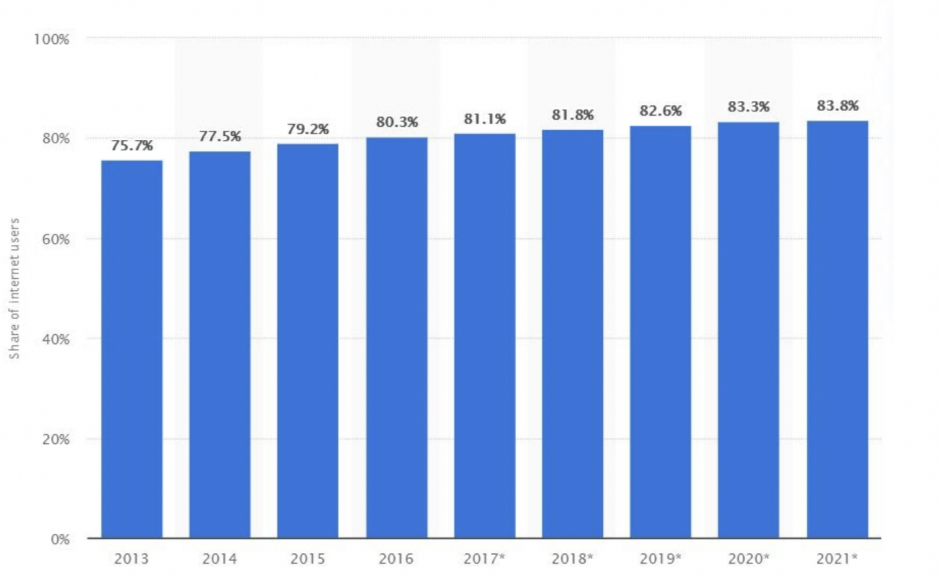


Рис. 1: Процент пользователей, просматривавших видео, среди всех пользователей сети интернет за год

Вместе с общим увеличением количества видео в интернете растет потребность в улучшении качества технологий передачи видео по сети.

Одним из таких способов является интерлейсинг или чересстрочная развертка. В дальнейшем оба этих термина будут использованы эквивалентно. Интерлейсинг позволяет при передаче видео снизить количество передаваемой информации в 2 раза без уменьшения частоты кадров в секунду (FPS), либо соответсвенно в 2 раза увеличить частоту кадров в секунду без изменения количества передаваемой информации.

Опишем метод интерлейсинга. Для начала введем понятие кадра и видео. Каждый кадр видео представим в виде трехмерного тензора, имеющего размерности (c, h, w), где c — количество компонент цветового пространства видео, h и w — высота и ширина соответсвенно. То есть каждый кадр имеет h строчек и w столбцов, а нумерация начинается с 0. Видео, в свою очередь, возможно представить в виде четырехмерного тензора — (n, c, h, w), где добавлена одна размерность n — количество кадров в видео.

Далее введем понятие полукадра. Каждый отдельный кадр можно разделить на два полукадра. Верхний полукадр состоит из строчек кадра с четными номерами (четный индекс размерности h), а нижний, соответственно, — из строчек с нечетными номерами. Более наглядно разделение кадра на полукадры показано на Рис. 2.

Рис. 2: Чересстрочная развертка наглядно

Суть метода интерлейсинга заключается в том, чтобы сформировать новый кадр, состоящий из полукадров соседних кадров, а затем по очереди менять полукадры. Таким образом, можно добиться увеличения частоты кадров в 2 раза, меняя полукадры вместо целых кадров, либо уменьшения количества передаваемой информации в 2 раза, сформировав новые кадры из полукадров старых, а лишние полукадры просто удалив. Наглядно процесс формирования нового кадра из полукадров показан на Рис. 3.

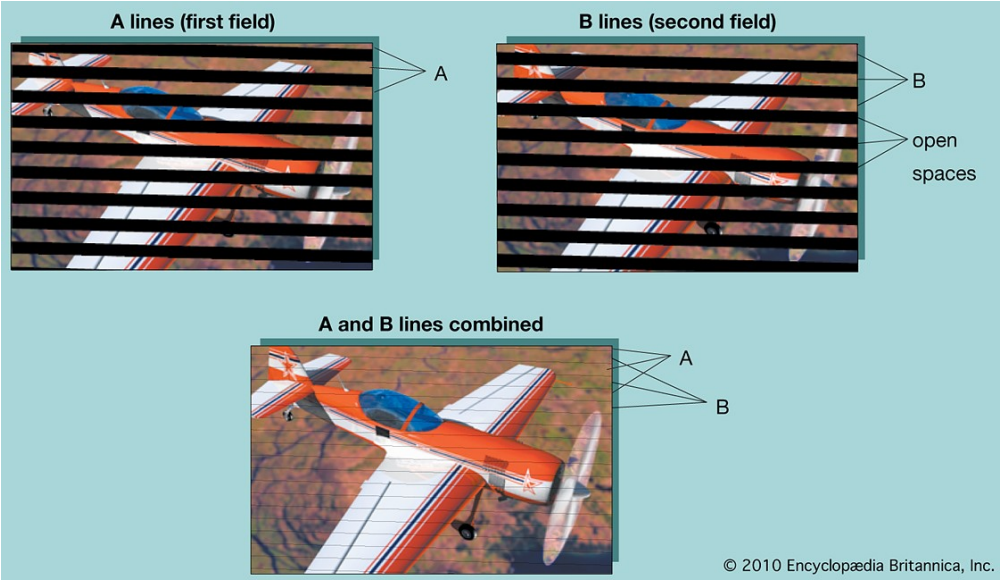


Рис. 3: Процесс формирования нового кадра из полукадров

Описанный подход прост и эффективен, однако имеет существенный недостаток. Побочным эффектом совмещения полукадров соседних кадров исходного видео является характерный для чересстрочных видео эффект “гребенки” на границах объектов, примеры которого можно увидеть на Рис. 4, 5, а также ”призрачный” эффект — артефакт видео, при котором за быстро движущимся обьектом движется его полупрозрачная копия. Примеры ”призрачного” эффекта можно увидеть на Рис. 6, 7.

Описанные эффекты ухудшают впечатления от просмотра видео. Чтобы избавиться от негативных эффектов интерлейсинга, был придуман деинтерлейсинг — процесс исправления чересстрочности, то есть получения исходного видео из чересстрочного видео.

Как описано ранее, интерлейсинг может быть использован с двумя целями:

Рис. 4: Пример эффекта ”гребенки”

Рис. 5: Пример эффекта ”гребенки”

1. Повысить частоту кадров в 2 раза, не меняя количество передаваемой информации;
2. Уменьшить количество передаваемой информации в 2 раза, не меняя частоту кадров.

В первом случае деинтерлейсинг сводится к тривиальной задаче совмещения соответсвующих полукадров в кадры исходного видео, поэтому первый случай мало интересен с научной точки зрения.

Во втором случае задача деинтерлейсинга — восстановить удаленные полукадры, пользуясь информацией, содержащейся в присутствующих полукадрах. Этот случай и будет рассмотрен в данной работе.

Алгоритм, восстанавливающий утерянную информацию во втором случае, называется деинтерлейсером (от англ. *deinterlacer*). За все время использования интерлейсинга было изобретено и опубликовано огромное множество различных алгоритмов деинтерлейсинга — от простого усреднения известных соседних строчек до сложных нейросетевых архитектур, которые требуют значительное время для обучения.

Рис. 6: Пример ”эффекта призрака”

Однако для оценивания алгоритмов деинтерлейсинга до сих пор, даже в самых современных работах по деинтерлейсингу, используются классические методы Peak Signal-to-Noise Ratio — PSNR [3] и Structural Similarity Index Measure — SSIM [4]. В ряде исследований, в том числе в проведенном в рамках этой работы, эти методы показали недостаточную корреляцию с субьективными оценками респондентов.

Первой целью данной работы является разработка системы оценки алгоритмов деинтерлейсинга, публикация ее на сайте [5] и предоставление к ней доступа всем исследователям и разработчикам деинтерлейсеров в мире. Задача состоит в том, чтобы сравнить современные наработки с классическими подходами к задаче деинтерлейсинга между собой с помощью существующих обьективных методов оценки качества видео и субьективных оценок респондентов и ответить на вопросы: “Насколько глубоко научное сообщество уже исследовало задачу деинтерлейсинга?” и ”Насколько ценны современные наработки?”

Второй целью данной работы является разработка метода, позволяющего сравнить качество различных алгоритмов таким образом, чтобы это наиболее соответствовало субьективным оценкам людей.

Рис. 7: Пример ”эффекта призрака”

2. Постановка задачи

В данной работе рассматриваются две задачи, поэтому постановок тоже две.

2.1. Задача №1

Из открытых источников и других крупных наборов видео, а также из открытых данных с видеохостингов, таких как *Youtube*, *Rutube*, *Vimeo* и подобных, нужно собрать N видео длины L, таких, чтобы:

- Набор был бы разнообразен, то есть в набор были бы включены видео из всех наиболее частых категорий;
- Набор был не избыточен, то есть в нем не было бы видео, схожих по результатам работы деинтерлейсеров на них;
- Набор соответствовал бы заданным ограничениям на количество, разрешение и длину видео, связанным с количеством доступных вычислительных мощностей.

Построить систему и метод оценки алгоритма деинтерлейсинга с помощью существующих обьективных методов оценки качества.

- Алгоритм оценки должен принимать на вход алгоритм деинтерлейсинга в виде исходного кода и инструкций запуска, либо уже результат работы алгоритма, исходное видео, а также набор методов оценки качества, результаты которых нужно посчитать, в виде

функций;

- Алгоритм оценки должен запускать деинтерлейсер, проверять его результаты на предмет возможных отклонений (например, восстановлены не те полукадры, кадры поменялись местами, произошел цветовой сдвиг). Затем, либо предупреждать об ошибке проверки, либо считать результаты заданных методов оценки качества;
- Система должна отображать текущие результаты сравнения на сайте [6];
- Система должна давать возможность добавить результаты алгоритма и результаты методов оценки качества;
- Система не должна давать разработчикам методов возможности “подогнать” свой алгоритм под набор данных, то есть сделать алгоритм, показывающий хороший результат именно на данном наборе, но плохо работающий на других.

2.2. Задача №2

Разработать алгоритм, позволяющий оценить качество деинтерлейсинга.

Для начала определим необходимые термины:

- Пиксель p — это тройка (r, g, b) или (y, u, v) цветовых компонент, где каждая компонента представляет какую-то размерность цветового пространства. Вместе данные компоненты кодируют цвет. $r, g, b \in H$, где H — некоторое множество допустимых для кодировки значений. $H \subset \mathbb{R}$. Обычно H это $[0...1]$ или $[0...255]$;
- Кадр $I \in P^{(h \times w)}$, где $P^{(h \times w)}$ — множество, состоящее из матриц, элементами которых являются пиксели: $P = (p_{i,j})$, где $p_{i,j}$ — пиксель, $i = 1, ..., h; j = 1, ..., w$. Иначе говоря, кадр I — это матрица, состоящая из пикселей;
- Видеопоследовательность V — это упорядоченный набор кадров одного размера: $V = \{I_i\}_{i=1}^{N}, I_i \in P^{(h \times w)}$, где N — количество кадров в видео V, $N \in \mathbb{N}$;
- Верхний полукадр кадра $TF(I)$ — это матрица $\in P^{(\frac{h}{2} \times w)}$, состоящая из только четных строчек матрицы пикселей I. Строчки считаются сверху вниз, начиная с 0;
- Нижний полукадр кадра $BF(I)$ — это матрица $\in P^{(\frac{h}{2} \times w)}$, состоящая из только нечетных строчек матрицы пикселей I. Строчки считаются сверху вниз, начиная с 0;
- Чересстрочная видеопоследовательность от исходной видеопоследовательности

  $interlaced(V, TFF/BFF)$ — упорядоченный набор полукадров исходной видеопоследовательности. Положение полукадра чередуется, то есть

  либо $\{TF(V[0]), BF(V[1]), TF(V[2]), BF(V[3]), ...\}$,

  либо $\{BF(V[0]), TF(V[1]), BF(V[2]), TF(V[3]), ...\}$ в зависимости от TFF/BFF.

Далее в тех случаях, когда неважно BFF интерлейсинг или TFF, будем опускать параметр TFF/BFF и писать просто $interlaced(V)$, имея ввиду какой угодно тип интерлейсинга;

- TFF или BFF интерлейсинг (от англ. *Top Field First* и *Bottom Field First* соответственно) — типы интерлейсинга. При TFF интерлесинге чересстрочная видеопоследовательность выглядит как $\{TF(V[0]), BF(V[1]), TF(V[2]), BF(V[3]), \ldots\}$, при BFF интерлейсинге — $\{BF(V[0]), TF(V[1]), BF(V[2]), TF(V[3]), \ldots\}$;
- $deinterlacer(interlaced(V), TFF/BFF)$ — алгоритм, принимающий чересстрочную видеопоследовательность и тип интерлейсинга и возвращающий $\hat{V}$ — видеопоследовательность, восстановленную из $interlaced(V)$;
- Метод оценки качества $A(V, \hat{V})$ — алгоритм, принимающий восстановленное видео и в некоторых случаях исходное видео и возвращающий число $m \in \mathbb{R}$, которое должно отражать восприятие визуального качества видео человеком. Важно отметить, что число m, вообще говоря, не отражает меру близости между исходным видео и восстановленным. Число m — это попытка перевести на язык математики восприятие визуального качества видео человеком.

Задача — разработать метод оценки качества $A(V, \hat{V})$, превосходящий по качеству существующие методы. Качество метода будет измеряться как ранговая корреляция Спирмена [2] между субьективными оценками респондентов и результатами метода. Детали расчета ранговой корреляции Спирмена будут описаны в Главе 6.

3. Обзор области

Классификация методов деинтерлейсинга

Существуют различные алгоритмы деинтерлейсинга. Прежде всего, их можно разделить на нейросетевые и не использующие нейросети. На Рис. 8 можно увидеть примерную классификацию алгоритмов деинтерлейсинга, не использующих нейросети или, как их еще можно назвать, эвристических алгоритмов.

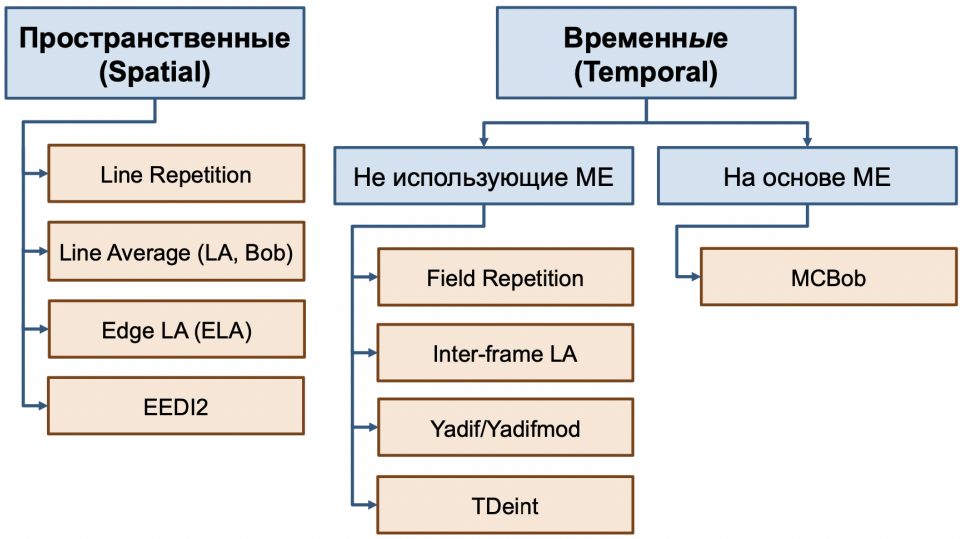


Рис. 8: Классификация алгоритмов деинтерлейсинга

Самым базовым считается Bob (или Line Average). Алгоритм вычисляет отсутствующие полукадры путем усреднения соседних строчек известных полукадров. Этот алгоритм работает очень быстро в силу своей простоты, и даже показывает неплохие результаты по объективным методам оценки качества, таким как PSNR [3] и SSIM [4].

Эвристические алгоритмы деинтерлейсинга можно разбить на те, которые используют информацию только с текущего кадра (Пространственные) и те, которые используют информацию с соседних кадров (Временные). Естественно, у двух этих множеств есть пересечение — Пространственно-Временные алгоритмы, использующие информацию как с соседних, так и с текущего кадра. Кроме того, отдельного внимания заслуживают алгоритмы, использующие Motion Estimation (ME) — технику, позволяющую оценить направление и скорость движения для каждого пикселя. Формально, задачу Motion Estimation (ME) в данном случае можно поставить так:

- Вход: Чересстрочная видеопоследовательность $I(V)$
- Выход: Карта движения $M \in \mathbb{M}^{height \times width}$, где $height$ и $width$ — высота и ширина соответственно, а $m \in \mathbb{M}$ является тройкой (s, dx, dy), где s является скоростью предсказанного движения, а dx, dy задают направление предсказанного движения. $s, dx, dy \in \mathbb{R}$

Эта техника нашла применение в сфере деинтерлейсинга потому, что характерные артефакты

интерлейсинга обычно возникают на границах движущихся объектов, а ME позволяет “предсказать” движение для отсутствующих пикселей и, пользуясь этим знанием, интерполировать их. Яркими примерами данной категории алгоритмов являются Motion Compensation Bob (MCBob) и [7].

Однако большинство современных деинтерлейсеров — нейросетевые. Они работают медленнее, но зачастую существенно превосдят по качеству эвристические. Их затруднительно более глубоко классифицировать, потому как каждая нейросетевая архитектура по-своему уникальна. Примерами таких деинтерлейсеров будут MFDIN [8] и Real-time Deep Video Deinterlacer [9].

Также все деинтерлейсеры можно разделить на восстанавливающие все кадры и восстанавливающие только четные/нечетные кадры.

Методы оценки качества деинтерлейсинга

В области оценки качества восстановления чересстрочного видео существует два устоявшихся классических метода — PSNR [3] и SSIM [4]. Авторы современных и передовых исследований в этой области до сих пор используют только эти методы для сравнения качества деинтерлейсеров, что подтверждают работы, приведенные в пример в предыдущем абзаце. Данная работа показывает, что существуют другие методы, имеющие большую корреляцию с человеческим восприятием.

Сравнения алгоритмов деинтерлейсинга

В области деинтерлейсинга до недавних пор существовало только одно открытое сравнение (бенчмарк) деинтерлейсеров - Vegas Deinterlacing Challenge 2019 [10] . Набор данных состоит из одного синтетического видео длиной 300 кадров с частотой кадров в секунду 30, на котором вращается трехмерная кривая Лиссажу. Из-за большого количества движения на этом видео, многие деинтерлейсеры показывают плохие результаты. Бенчмарк состоит из сравнения 2 деинтерлейсеров по методам оценки качества - Vegas Deinterlace Blend и Vegas Deinterlace Interpolate, а так же сравнения 5 других деинтерлейсеров визуально.

Однако, в рамках этой работы был разработан и опубликован MSU Deinterlacer Benchmark [6]. Набор данных для бенчмарка состоит из 28 видео длиной по 60 кадров. Частота кадров в секунду варьируется от 24 до 60. В бенчмарке сравниваются между собой 30 деинтерлейсеров, в том числе современные разработки, пока не доступные для широкого использования в индустрии.

4. Алгоритм создания тестового набора данных

Одной из задач, поставленных перед началом данной работы, является создание набора видеопоследовательностей для оценки качества существующих алгоритмов деинтерлейсинга.

Для создания подобного набора был в начале собран большой, разнообразный и избыточный набор случайных видео.

Изначальные видео были собраны с *Vimeo*. Всего видео 10312. Средняя длина видео примерно 71,34 секунды. Количество видео и информация об источнике позволяют полагать, что среди этих видео содержится разнообразный контент. Просмотр случайной выборки из 100 видео показал, что в наборе содержатся видеопоследовательности следующих тематик: спорт, анимация, панорама, новости, пейзажи, сцены из кино, телешоу, документальные фильмы, интервью, реклама, а также несколько видео, которые сложно отнести к какой-либо категории.

В силу ограниченности вычислительных ресурсов далее были заданы ограничения на размер набора видеопоследовательностей. Были выбраны следующие ограничения — не более 40 видео в наборе, длиной по 60 кадров. 40 видео, на первый взгляд, может показаться маленьким числом, но при умножении на количество алгоритмов-участников сравнения и при условии, что данные нужно хранить в сыром формате, так как любые преобразования могут повлечь искажения, сравнимые по силе с искажениями от деинтерлейсинга, мы получаем около 300 гигабайт сырых данных, что довольно много. Число 60 выбрано, потому что большинство видео имеют частоту 60 кадров в секунду.

Таким образом, для сокращения количества видео был разработан и применен специальный алгоритм.

4.1. Описание алгоритма

Вход алгоритма — около 10312 видеопоследовательностей разрешения 3840 $\times$ 2160 и произвольной длины, заданная длина L кадров, заданное максимальное количество видеопоследовательностей M.

Выход — N <= M — это итоговое число последовательностей, а также сами эти N видеопоследовательностей длины L кадров.

1. Первым шагом алгоритма является отображение большого количества видео в двумерное пространство. Для того, чтобы отобразить видеопоследовательность в точку в двумерном пространстве, использовался алгоритм Google Si/Ti, описанный, например, в статье [11]. С кратким описанием этого алгоритма можно ознакомиться в подсекции ”Описание сопутствующих алгоритмов”;

2. Второй шаг алгоритма — кластеризация видео с помощью классического алгоритма кластеризациии K-Means[13] на $B = round(0.75 * M)$ кластеров;

3. Третий шаг — вычисление геометрических центров кластеров и отбор B видео, наиболее близких к центру каждого из кластеров;
4. Четвертый — разбиение отобранных видео на отрезки по L кадров и повторение шагов 1-3, но уже для отрезков видео. Разбиение снова на B кластеров;
5. Далее следует ручной отсмотр содержимого кластеров и выбор “мусорных” кластеров, если таковые имеются — например, все отрезки, на которых видно только черный экран, образовали отдельный кластер. Очевидно, что черные кадры неинформативны и в силу своей примитивности являются слишком простыми входными данными для алгоритма деинтерлейсинга;
6. Шестой шаг — выбор 5 видео, ближайших к центру кластера из каждого не “мусорного” кластера. Итого $5 * B$ последовательностей;
7. Седьмой — Ручной отбор из $5 * B$ видеопоследовательностей M, проходящих далее;
8. Восьмой шаг — запуск всех доступных алгоритмов деинтерлейсинга на данных M видеопоследовательностях;
9. Девятый — подсчет PSNR [3] между результатами всех деинтерлейсеров и построение cross-PSNR таблицы алгоритмов деинтерлейсинга из шага 8 для каждой видеопоследовательности;
10. Десятый шаг — подсчет корреляции cross-PSNR таблиц и получение таблицы корреляции видеопоследовательностей. По этой таблице можно смотреть, на каких последовательностях деинтерлейсеры выдают в одинаковой степени похожие друг на друга результаты. Чем выше корреляция в этой таблице, тем больше видеопоследовательности близки друг к другу с точки зрения алгоритмов деинтерлейсинга. Визуализацию такой матрицы можно увидеть на Рис. 9;
11. Наконец, последний шаг — выбор ровно одной видеопоследовательности из всех кластеров, близких между собой.

Таким образом, набор видео, собранный данным алгоритмом:

- Разнообразен, так как является поднабором достаточно большого множетсва видео и отобран с помощью статистик, позволяющих кластеризовать видео;
- Неизбыточен, так как на шагах 10-11 выбрасываются похожие с точки зрения результатов работы деинтерлейсеров последовательности;
- Вписывается в заданные ограничения по построению.

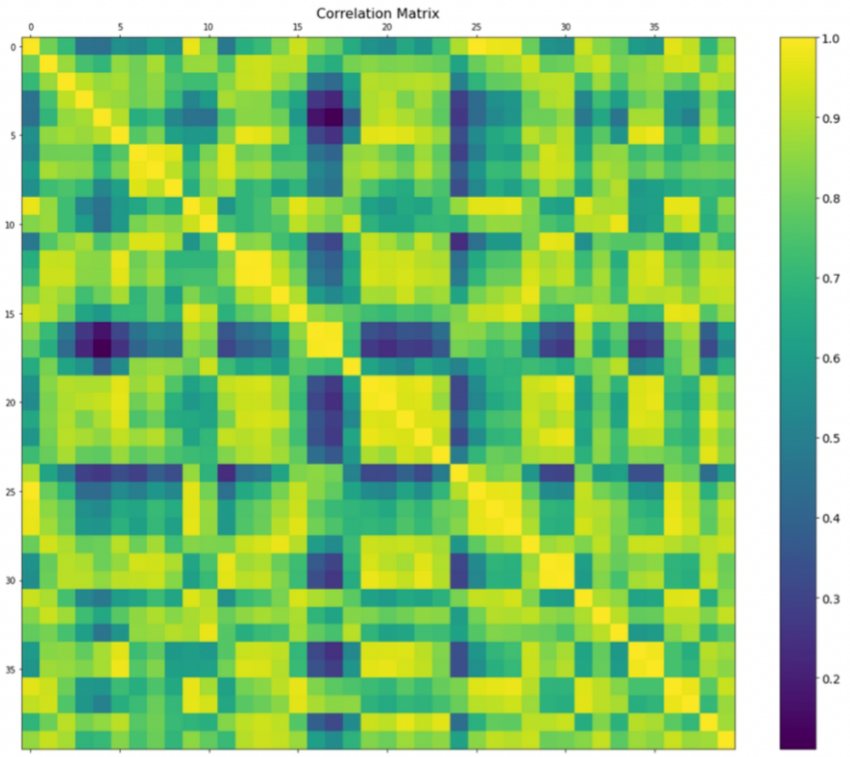


Рис. 9: Матрица корреляций cross-PSNR матриц

4.2. Описание сопутствующих алгоритмов

4.2.1. Описание Google Si/Ti

1. Используется кодек libx264. С помощью фреймворка ffmpeg [12] видео кодируется командой **ffmpeg -i video.mkv -c:v libx264 -qp 28 -b_qfactor 1 -i_qfactor 1 encoded_video.mkv**;
2. Все кадры размечаются на группы: I, P, B;
   - I - независимо сжатые кадры;
   - P - кадры, при сжатии которых используются предыдущие;
   - B - кадры, при сжатии которых используются предыдущие и следующие.
3. $$Si = \frac{meansize(I)}{(channel * width * height)}$$
   где $meansize(I)$ — это средний размер сжатых I кадров в байтах;
4. $$Ti = \frac{meansize(P)}{meansize(I)}$$
   где $meansize(I)$ — это средний размер сжатых I кадров в байтах, а $meansize(P)$ — это средний размер сжатых P кадров в байтах.

Физический смысл Si - сложность структурных деталей, границ обьектов в видео. Физический

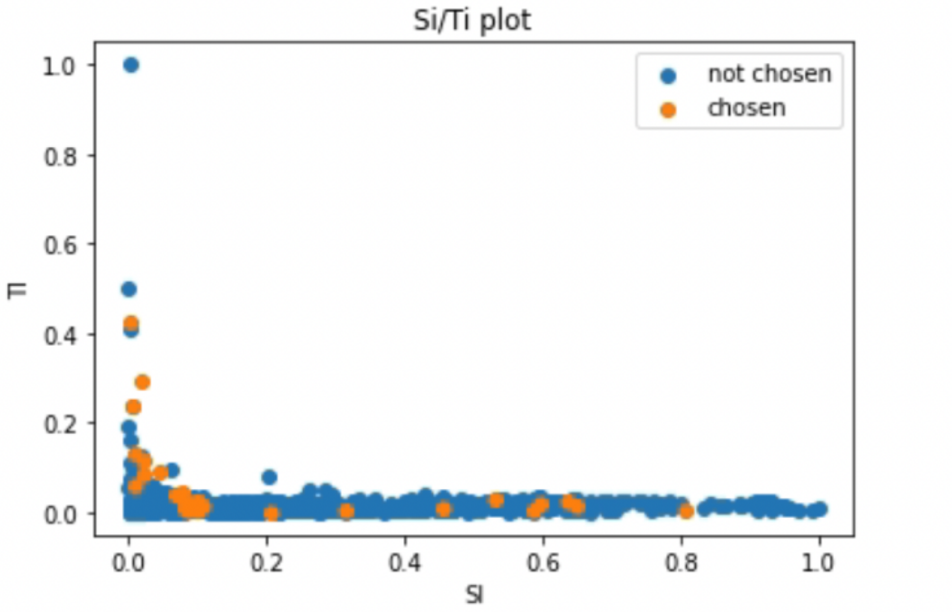


Рис. 10: Результат работы алгоритма сбора набора данных на исходных видео

смысл Ti - количество и сложность движения в видео.

5. Система оценки алгоритмов деинтерлейсинга

5.1. Используемые методы оценки

Для сравнения алгоритмов деинтерлейсинга были использованы 3 метода. Из них 2 были классическими для данной задачи - PSNR [3] и SSIM [4], а так же один современный и общепризнанный метод оценки качества видео - Video Multi-Method Assessment Fusion (VMAF) [14].

Однако, методы для оценки качества видео возможно применять по разному, так как само по себе видео может быть в различных цветовых пространствах и по разному кодироваться.

В случае деинтерлейсеров, самое популярное цветовое пространство - это YUV, где Y представляет общую яркость изображения, а U и V - цветоразностные компоненты. То есть, каждый кадр видео задается тремя матрицами - $Y, U, V \in \mathbb{H}$, где H — множество допустимых значений. Матрица Y по сути является черно-белым кадром.

Однако, у YUV существует множество модификаций. Например, YUV422 в 2 раза уменьшает матрицы компонентов U и V. Таким образом, в кодировке YUV422 кадр представлен тремя матрицами компонентов $Y \in \mathbb{H}^{h \times w}$ и $U, V \in \mathbb{H}^{h/2 \times w/2}$, где h, w — высота и ширина, а H — множество допустимых значений, а модификация YUV444 оставляет все матрицы исходного размера. Но для всех модификаций верно одно утверждение — Y-компонента остается неизменной.

Так как деинтерлейсеры иногда в процессе работы переводят видео в одно из цветовых пространств, было решено применять методы только по Y-компоненте.

К тому же, помимо этой практической причины, применение метода по Y-компоненте позволяет измерить уровень шума не хуже, чем измерение по всем трем компонентам в цветовом пространстве RGB. Это так, потому что Y-компоненту можно приблизительно посчитать по формуле

$$Y \approx 0.299 * R + 0.587 * G + 0.114 * B$$

На Рис. 11 показаны результаты экспериментального подтверждения — Peak Signal-to-Noise Ratio [3] по Y-компоненте и по RGB-компонентам имеют высокую корреляцию.

Кроме того, некоторые деинтерлейсеры имеют такое понятие, как “разгон”. Качество восстановления на первых T ”разгоночных” кадрах может быть существенно ниже, чем на последующих. Как правило T < 5. Обычно “разгоночные” кадры составляют крайне малую часть видео, поэтому их не учитывают, однако в нашем случае само по себе видео очень короткое, поэтому “разгоночные” кадры могут существенно влиять на результат. Если исходная видеопоследовательность имеет длину 60 кадров, то чересстрочное видео будет иметь длину 30 кадров. Поэтому результат на первых 5 кадрах просто не учитывается. Результаты методов оценки на самом деле считаются только на 50 кадрах.

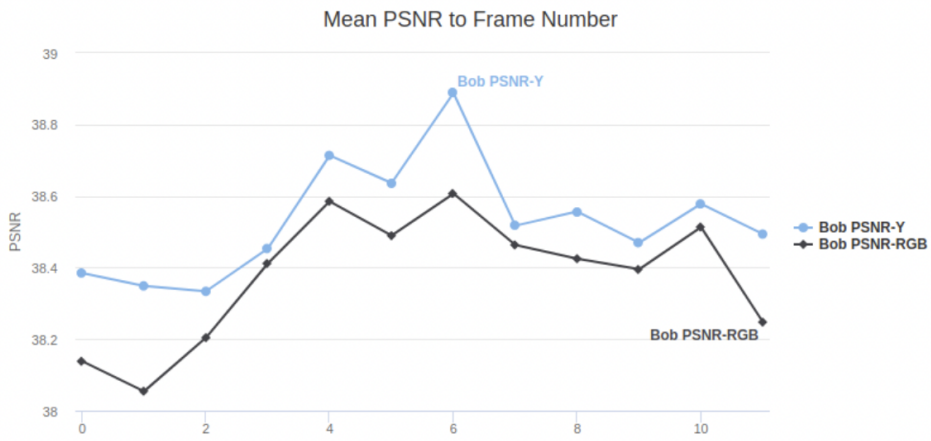


Рис. 11: Графики покадровых PSNR-Y и PSNR-RGB на алгоритме Bob

5.2. Методология субъективного сравнения

Кроме сравнения алгоритмов деинтерлейсинга по обьективным методам, было также проведено исследование субъективного восприятия с помощью сбора оценок респондентов. Для проведения субъективного сравнения была использована система subjectify [15], разработанная и поддерживаемая лабораторией компьютерной графики и мультимедиа ВМК МГУ.

Для подсчета субъективных оценок был использован алгоритм сравнения ”каждый против каждого”. Респонденту показывалось одновременно две восстановленных версии одного и того же исходного видео. Задачей респондента было выбрать одно из них, которое кажется ему наиболее качественным.

Немаловажной проблемой во время проведения подобных исследований являются недобросовестные респонденты — то есть респонденты, отвечающие на вопросы случайно, нанося таким образом вред обьективности результатов исследования. Для борьбы с такими респондентами были использованы проверочные вопросы. Для проверочных вопросов были отобраны пары видео $(V, interlaced(V))$, где V — видео, такие, где на глаз были сразу заметны артефакты $interlaced(V)$.

Каждый участник получал 30 вопросов, из которых 3 были проверочными. При непрохождении хотя бы одного из проверочных вопросов, результаты участника были аннулированы и не учитывались в исследовании.

Количество участников выбиралось таким образом, чтобы каждая пара алгоритмов на каждой паре видео получила 10 оценок.

Необходимое количество участников примерно рассчитывалось по формуле:

$$P \approx \frac{\frac{M*(M-1)}{2} * N * 10}{30 - 3}$$

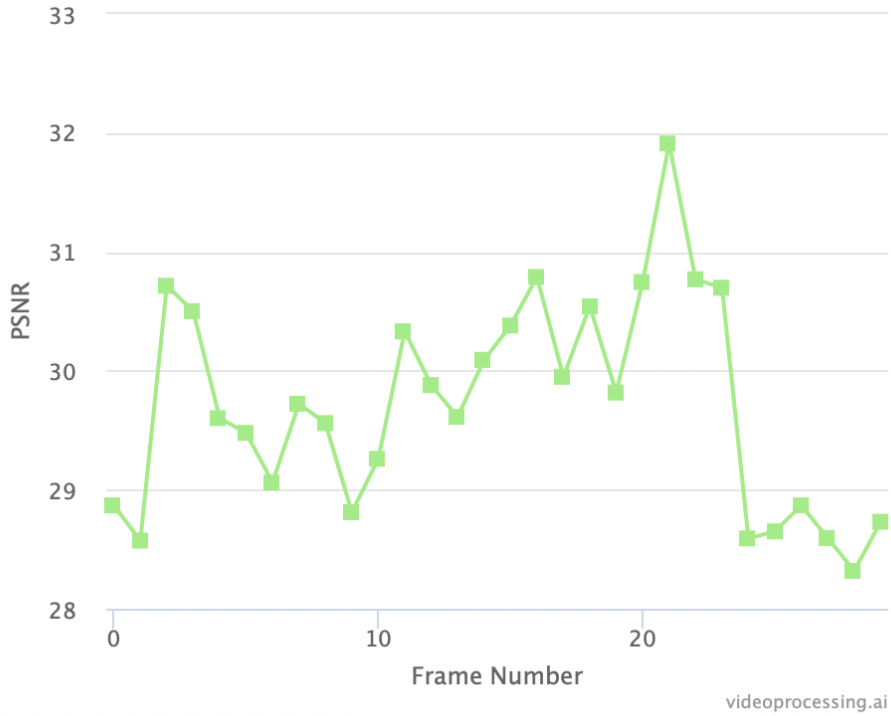


Рис. 12: Пример “разгона” алгоритма PAL Interpolation

где M — количество алгоритмов, N — количество видео.

Количество участников можно рассчитать только примерно, потому что не все участники будут добросовестными, и не все участники ответят на все 30 вопросов.

Далее, для преобразования матрицы оценок ”каждый против каждого” используется модель Брэдли–Терри [16].

Кратко алгоритм преобразования можно описать так:

- На входе матрица $W \in \mathbb{Z}^{N \times N}$, где N — число алгоритмов. $W_{i,j}$ - количество побед i-того участника над j-ым;
- Нужно найти $p_i \in \mathbb{R}$, $i \in 1..N$ — оценки максимального математического правдоподобия. Оценка вероятности $P(i \text{ лучше } j)$ рассчитывается по формуле $\frac{p_i}{p_i + p_j}$.

1. Случайно инициализируется p_i, задаем количество итераций E;
2. Пусть W_i - суммарное количество побед i-того участника, то есть строковая сумма матрицы W. p'_i пересчитывается по формуле:

$$p'_i = \frac{W_i}{\sum_{j \neq i} \frac{W_{i,j} + W_{j,i}}{p_i + p_j}}$$

3. p_i нормализуется:

$$p_i = \frac{p'_i}{\sum p'_j}$$

4. Повторяются шаги 2-3 E раз или до шага n, когда $p_{i,n-1} = p_{i,n}\ \forall i \in 1..N$.

Результат работы данной модели — субьективные оценки — далее будем называть их MOS (от англ. *Mean Opinion Score*).

На Рис. 13 показан пример результатов работы данной модели.

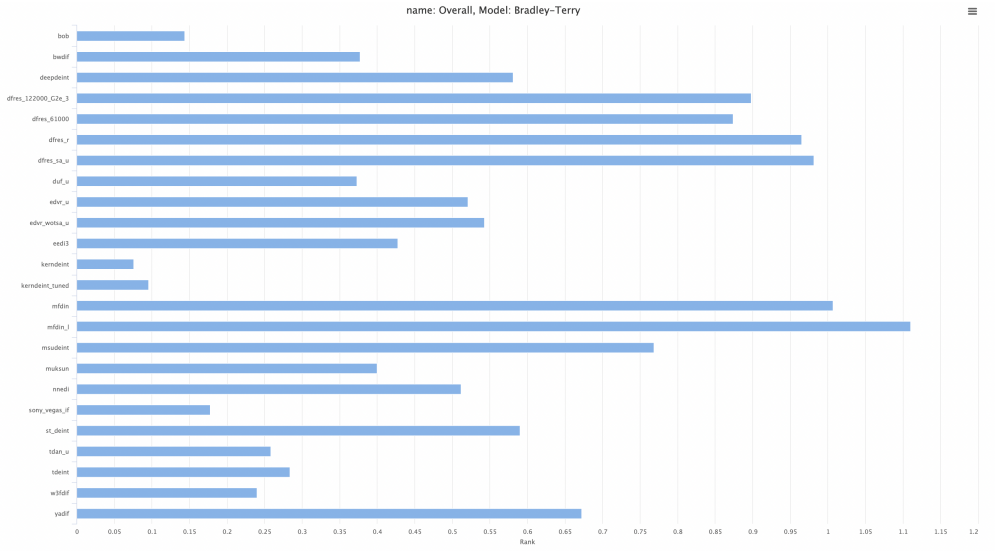


Рис. 13: Пример результатов работы модели Брэдли-Терри

5.3. Алгоритм приема и проверки данных

У разработчика алгоритма деинтерлейсинга есть несколько вариантов участия в сравнении:

1. Прислать исходный код своего алгоритма;
2. Прислать исполняемый файл своего алгоритма;
3. Скачать набор данных, самостоятельно применить свой алгоритм к нему и прислать результат.

Способ	Количество присланных методов
Код	0
Исполняемый файл	4
Результат работы	27

Таблица 1: Количество присланных методов

Самым популярным, очевидно, является третий способ, потому что немногие хотят делиться исполняемой версией своего алгоритма и тем более его исходным кодом.

Однако в этом случае есть вероятность, что присланные данные будут каким-либо образом испорчены и их будет некорректно использовать в сравнении с остальными алгоритмами. Основных причины может быть две:

1. При использовании деинтерлейсера был неправильно указан тип интерлейсинга (BFF вместо TFF);
2. Деинтерлейсер использовал конвертации цветовых пространств, которые влияют на качество выходных данных.

В случае деинтерлейсеров есть способ однозначно определить, были ли применены какие-либо преобразования.

Рассмотрим случай TFF-интерлейсинга.

Алгоритму деинтерлейсинга на входе известна последовательность.

$$\{TF(V[0]), BF(V[1]), TF(V[2]), BF(V[3]), \ldots\}$$

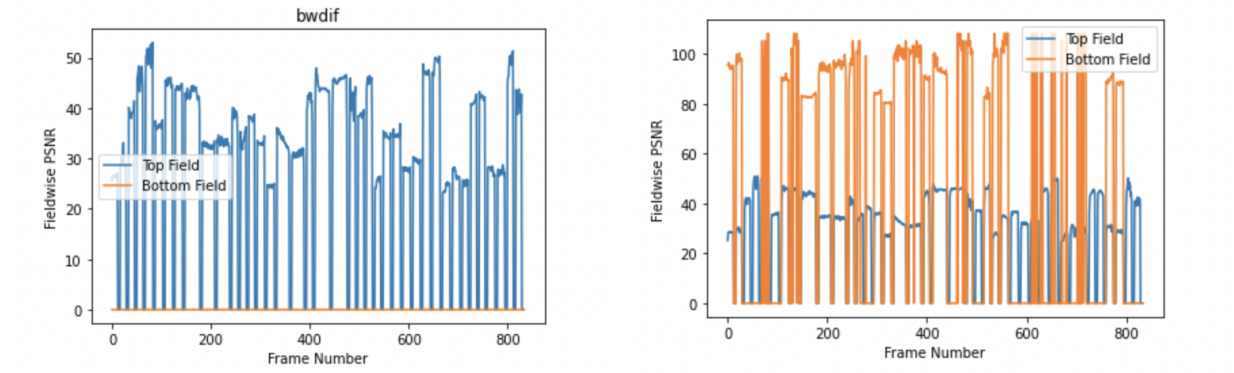
Задачей алгоритма деинтерлейсинга в этом случае является восстановить утерянные поля $\{BF(V[0]), TF(V[1]), BF(V[2]), TF(V[3]), \ldots\}$ и добавить их к тому, что ему уже известно из $interlaced(V)$.

То есть выходит, что в случае TFF наборы $\{TF(V[0]), BF(V[1]), TF(V[2]), BF(V[3]), \ldots\}$ и $\{TF(\hat{V}[0]), BF(\hat{V}[1]), TF(\hat{V}[2]), BF(\hat{V}[3]), \ldots\}$ должны быть поэлементно равны при корректной работе деинтерлейсера.

Эти поля могут быть не равны лишь в двух описанных выше случаях.

Поэтому для проверки результатов работы алгоритма деинтерлейсинга строится график PSNR [3] отдельно для верхних полей, отдельно для нижних. Так как в случае корректной работы алгоритма PSNR [3] формально должен быть равен бесконечности, слишком большие значения на графике заменяются нулями.

На Рис. 14 показаны примеры графиков при корректной и некорректной работе алгоритма соотвественно.



(a) Пример графика валидации для алгоритма Bob-Weave, прошедшего валидацию

(b) Пример графика валидации для алгоритма MSU Deinterlacer, не прошедшего валидацию

Рис. 14: Примеры графиков валидации

Следующим этапом валидации является построение карт PSNR для каждого кадра с помощью фреймворка VQMT [17]. Результаты метода PSNR возможно измерить для каждого пикселя отдельно, что позволяет построить матрицу $M \in \mathbb{R}^{h \times w}$, где h и w - высота и ширина соотвественно, и $M_{i,j} = PSNR(V[n][i][j], \hat{V}[n][i][j])$.

Так как соотвествующие поля референсного видео V и $\hat{V}$ должны совпадать, то карта PSNR для каждого кадра должна быть ”полосатой”.

На Рис. 15, 16 можно увидеть примеры ”правильной” карты PSNR, а на Рис. 17 — неправильной.

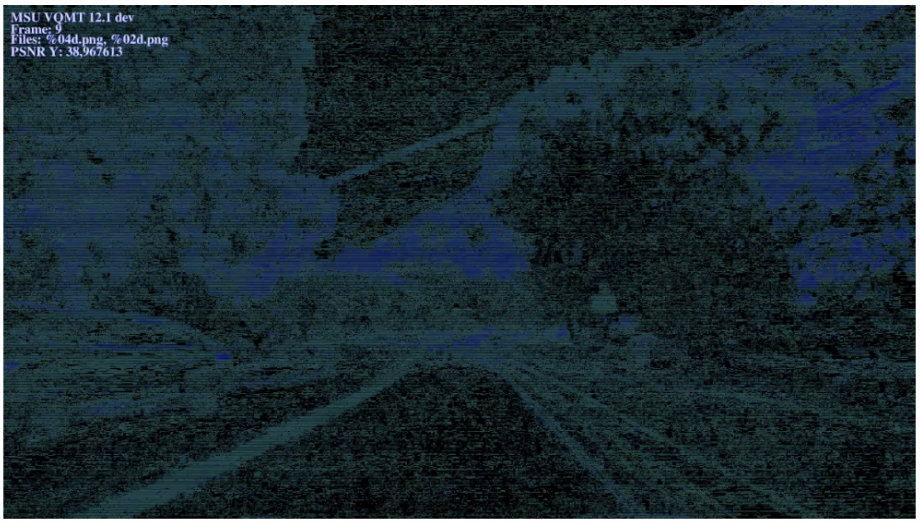


Рис. 15: Пример ”правильной” карты PSNR

Если результат деинтерлейсера не проходит один из шагов проверки, то предпринимается попытка получить видео V' из исходного видео V.

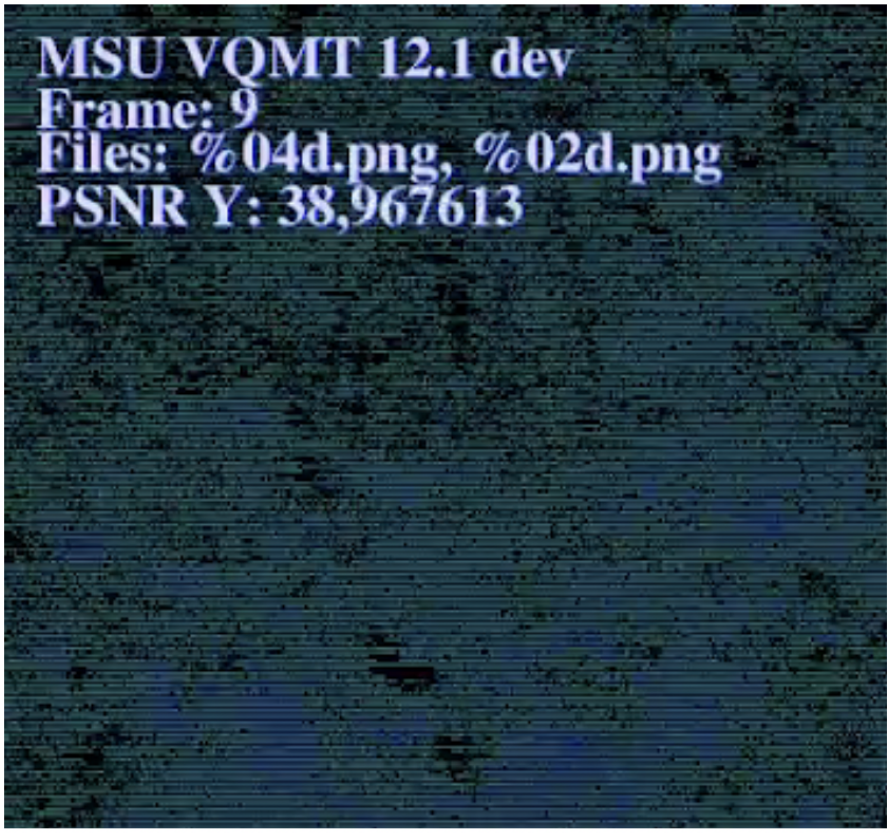


Рис. 16: Пример ”правильной” карты PSNR вблизи

Видео V' — это наиболее подходящее для сравнения ”исходное видео” для деинтерлейсера.

1. Если известно цветовое пространство деинтерлейсера, то видео V' получается из исходного видео V с помощью конвертации его в это цветовое пространство;

2. Если не известно и не может быть угадано цветовое пространство деинтерлейсера, то программно формируется таблица поиска. Таблица поиска T - это функция, которая переводит тройку (y, u, v) — пиксель исходного видео V в тройку $(y', u', v)'$ — пиксель видео V'. Таблица поиска формируется с помощью итерации по пикселям тех полей V и $\hat{V}$, которые должны совпадать. Если таблицу поиска удалось сформировать, она применяется к V и таким образом получается видео V'. Таблицу поиска может не получиться сформировать, если найдется два равных пикселя V, соотвествующих неравным пикселям $\hat{V}$;

3. Если таблицу поиска на втором шаге не удалось сформировать, то выбирается видео V' из предыдущих шагов с наибольшим PSNR с видео $\hat{V}$.

Кроме того, так как набор данных собран из видео в открытом доступе с Vimeo и алгоритм сбора этого набора детально расписан на странице [6], участник может поступить недобросовестно и найти исходные материалы для набора данных, получив таким образом несправедливое преимущество.

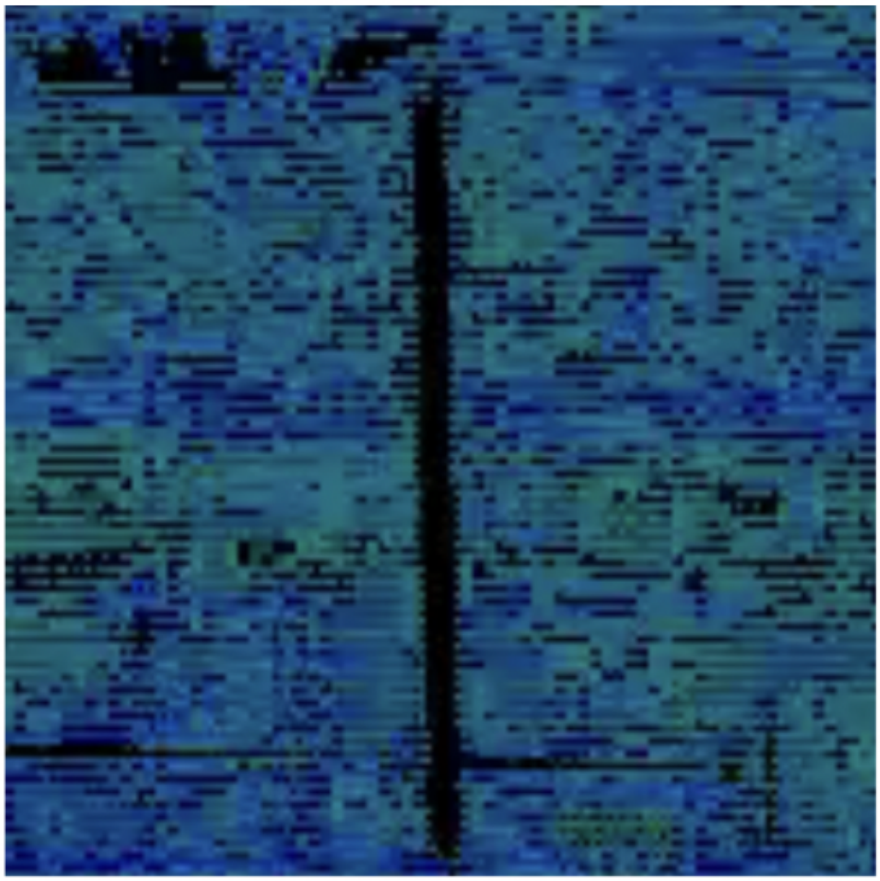


Рис. 17: Пример ”неправильной” карты PSNR вблизи

Для защиты от этого делается субпиксельный сдвиг на этапе создания набора видео. Мы двигаем видео в разрешении 3840×2160 на 1 пиксель в случайном направлении (влево, вправо, вверх или вниз), а затем уменьшаем разрешение до 1980×1080. Таким образом, мы получаем сдвиг на $\frac{1}{2}$ пикселя. Если недобросовестный участник найдет и использует исходное видео в своем алгоритме, то он наоборот ухудшит результаты метода оценки из-за этого сдвига.

6. Метод оценки качества алгоритмов деинтерлейсинга

6.1. Методология сравнения методов

В качестве способов оценки методов были выбраны:

- Коэффициент ранговой корреляции Спирмена — далее SROCC
- Коэффициент ранговой корреляции Кендалла — далее KROCC
- Линейный коэффициент корреляции Пирсона — далее PLCC

Формулы, по которым рассчитываются данные коэффициенты, приведены ниже

$$SROCC = 1 - \frac{6}{n(n+1)(n-1)} \sum_{i=1}^{n} (R_i - S_i)^2$$

где n — количество деинтерлейсеров, R_i — позиция деинтерлейсера, если их все отсортировать по результатам метода оценки качетсва, S_i — позиция деинтерлейсера, если их все отсортировать по результатам субьективного сравнения. $SROCC \in [-1, 1]$. $SROCC = -1$, когда сортировки полностью противоположны, $SROCC = 1$, когда сортировки полностью совпадают

$$KROCC = 1 - \frac{4}{n(n-1)} V$$

где n — количество деинтерлейсеров, V — количество неправильно упорядоченных методом оценки качества пар деинтерлейсеров относительно субьективного сравнения. $KROCC \in [-1, 1]$. $KROCC = -1$, когда сортировки полностью противоположны, $KROCC = 1$, когда сортировки полностью совпадают

$$PLCC = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2 \sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

где n — количество деинтерлейсеров, x_i — оценка i-того деинтерлейсера методом оценки качества, y_i — субьективная оценка i-того деинтерлейсера, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$. $PLCC \in [-1, 1]$. $PLCC$ помимо ранжирования учитывает также численную разницу результатов методов относительно численной разницы субьективного сравнения

6.2. Сравнение существующих методов

Для того, чтобы создать собственный метод, наиболее соотвествующий результатам субьективного сравнения, прежде всего было решено проверить применимость к оценке качества алгоритмов деинтерлейсинга существующих методов. В том числе классические методы, используемые для задачи деинтерлейсинга, несколько классических методов, используемых

в других задачах обработки видео — как референсные, то есть использующие информацию с исходного видео V и получившегося видео $\hat{V}$, так и нереференсные, то есть использующие информацию только с получившегося видео $\hat{V}$. В отдельности стоит выделить методы, разработанные и используемые для другой задачи — повышения разрешения видео (Video Super-Resolution)[18]. Сходство задач повышения разрешения видео и деинтерлейсинга в том, что в обеих нужно дополнить видео отсутсвующей информацией, пользуясь только самим видео. В задаче деинтерлейсинга это отсутсвующие строки, в задаче повышения разрешения — квадраты пикселей.

Таким образом, итоговый список рассматриваемых методов:

- PSNR [3]
- SSIM [4]
- NIQE [21]
- SRE [20]
- MS-SSIM [19]
- VMAF [14]
- SAM [22]
- ERQA [23]
- LPIPS [24] в комбинации с VGG [26]
- LPIPS [24] в комбинации с AlexNet [27]
- NeuralSBS [25]

Для исследования применимости данных методов к задаче деинтерлейсинга были использованы набор данных и субъективное сравнение, сделанные в рамках данной работы.

С результатами данного сравнения можно ознакомиться на Рис. 18 и в Таблице 2. Как мы можем видеть, наилучший результат показывают методы, изначально разработанные для оценки качества повышения разрешения видео.

Гипотеза о применимости методов оценки качества повышения разрешения видео к задаче оценки качества деинтерлейсинга кажется логичной и интуитивно понятной из-за описанной выше схожести задач. **Результат, полученный в данном сравнении, экспериментально доказывает эту гипотезу**.

6.3. Предложенный метод

Одним из важнейших результатов сравнения является тот факт, что методы, разработанные для оценки качества повышения разрешения видео показывают себя лучше, чем классически используемые для данной задачи. По этой причине было принято решение выстраивать

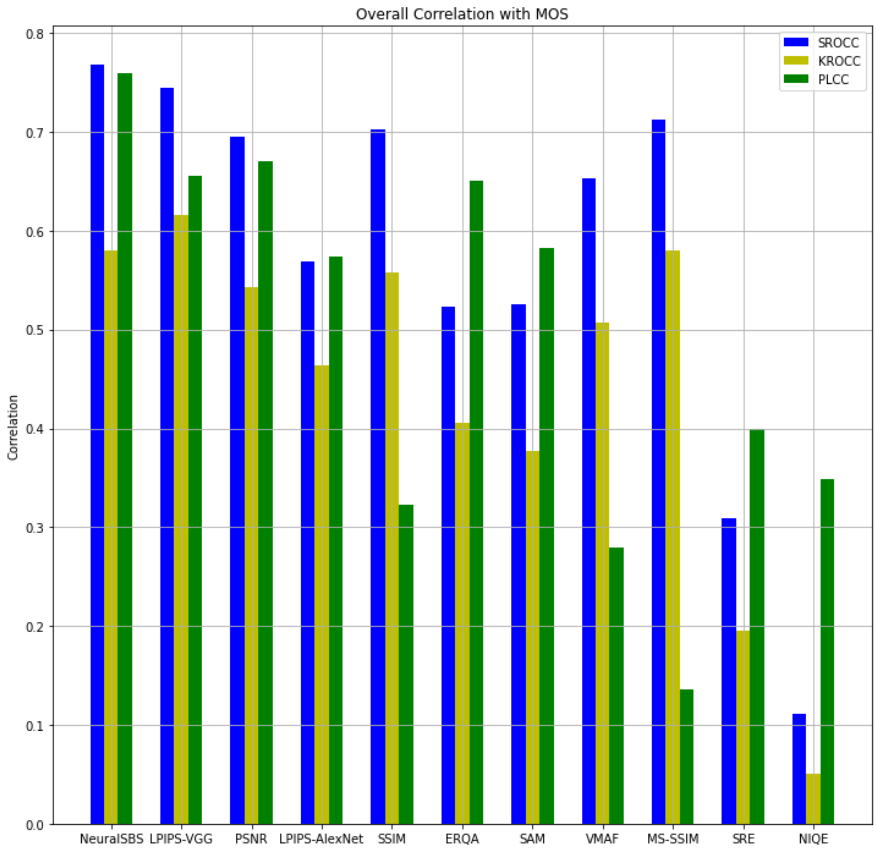


Рис. 18: Результаты сравнения существующих методов

	SROCC	KROCC	PLCC
PSNR [3]	0.696	0.543	0.671
SSIM [4]	0.702	0.558	0.323
NIQE [21]	0.111	0.051	0.349
SRE [20]	0.309	0.196	0.400
MS-SSIM [19]	0.712	0.580	0.136
VMAF [14]	0.653	0.507	0.279
SAM [22]	0.525	0.377	0.583
ERQA [23]	0.523	0.406	0.650
LPIPS-VGG [24]	0.744	**0.616**	0.656
LPIPS-AlexNet [24]	0.569	0.464	0.574
NeuralSBS [25]	**0.769**	0.580	**0.759**

Таблица 2: Результаты сравнения существующих методов на собранном наборе данных. В каждом столбце лучшее значение выделено жирным шрифтом.

свой метод, используя готовые результаты методов, превосходящих классические на данной задаче, — [24] и [25]. Однако, у данных методов есть один существенный недостаток в терминах работы с видео, а именно — они работают отдельно по каждому кадру, а затем усредняют результат. Но, как показывает практика, усреднение результатов метода по всем кадрам видео не является хорошей практикой и часто плохо коррелирует с человеческим понимаем ”качества” видео. Это происходит потому что человек воспринимает видео с отдельными кадрами, имеющими очень много артефактов, гораздо хуже чем видео, на всех

кадрах которого присутствует небольшое количество артефактов. Иными словами, если на двух видео получились одинаковые средние результаты методов, то лучше из них то, где эти результаты равномернее распределены по кадрам. Существует множество интересных идей и подходов к агрегации покадровых результатов метода по всему видео, например описанный в статье DeepVQA [28]. Однако, явным недостатком в данной области является отсутсвие крупных наборов данных, на которых можно было бы обучать агрегирующие нейросети, в частности подходы вроде Attention [29] или одноразмерные свертки, как в [28]. По этой причине способы агрегации в данной задаче ограничены лишь классическими математическими способами: среднее, мода, медиана, квантили.

Проведя сравнение данных способов был сделан вывод, что лучшим способом является подбор квантилей под данные, а затем усреднение данных квантилей для LPIPS-VGG[24] и NeuralSBS[25]

	SROCC	KROCC	PLCC
Среднее	0.758	0.595	0.701
Мода	0.764	0.613	0.737
Медиана	0.812	0.648	0.797
Квантили	0.843	0.667	0.813

Таблица 3: Результаты комбинации LPIPS-VGG+NeuralSBS разными способами агрегации

Таким образом, финальный результат метода рассчитывается по формуле

$$Proposed = \frac{1}{2} * Q(\{..., LPIPSVGG_i, ...\}, A) + \frac{1}{2} * Q(\{..., NeuralSBS_i, ...\}, B)$$

где $LPIPSVGG_i$, $NeuralSBS_i$ — результаты методов на i-ом кадре, Q — квантиль, A, B — настраиваемые параметры. Эксперименты показали, что лучшие значения для A и B — $A = 0.15$, $B = 0.85$. Иными словами, качество худших кадров видео оценивается с помощью $LPIPS$, а качество лучших — с помощью $NeuralSBS$. Итого, финальный метод принимает на вход два видео — V и $\hat{V}$ и возвращает число $m \in [0...1]$. Чем больше число m, тем лучше метод оценивает качество восстановления чересстрочности.

6.4. Эксперименты с предложенным методом

Прежде всего, сравним предложенный метод с конкурентами, в том числе с теми методами, на которых он основан, на наборе данных, собранном в рамках данной ВКР. Результаты сравнения можно увидеть на Рис. 19 и Таблице 4.

Однако, так как во время разработки и контроля качества метода использовался набор данных MSU Deinterlacer Benchmark [6], сравнение на нем может искажать реальное качество метода, так как существует риск неявно ”подогнать” метод под этот набор — то есть сделать метод, хорошо работающую только на данном наборе. По этой причине:

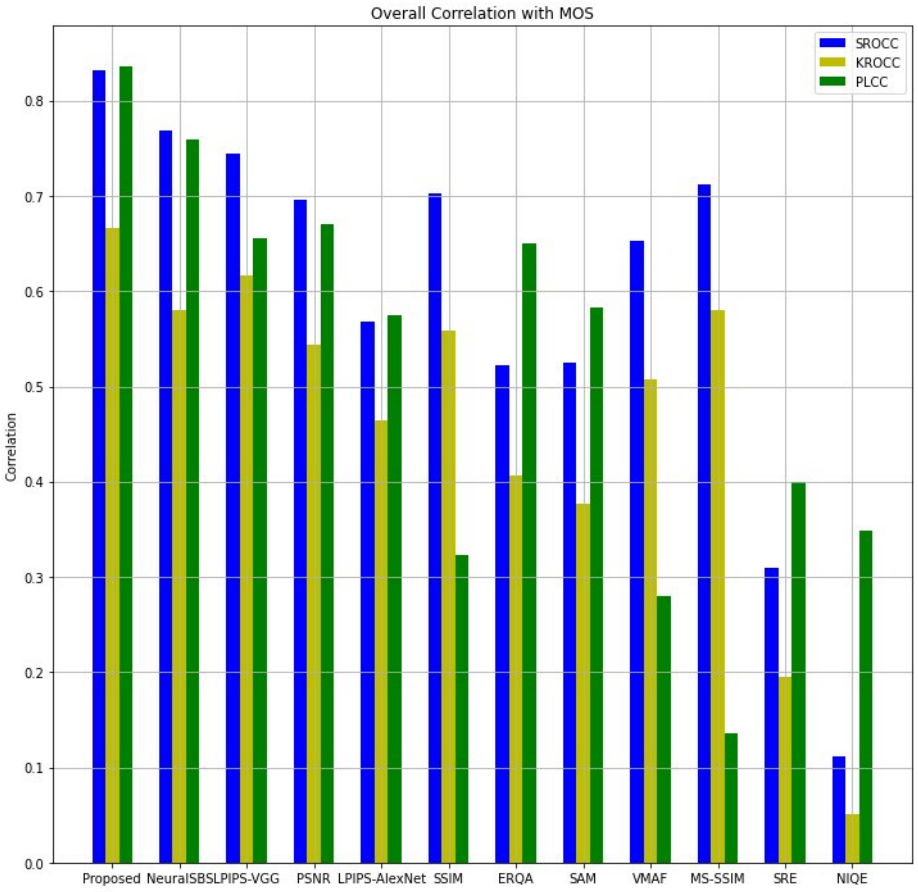


Рис. 19: Результаты сравнения на наборе данных MSU Deinterlacer Benchmark [6]

	SROCC	KROCC	PLCC
PSNR [3]	0.696	0.543	0.671
SSIM [4]	0.702	0.558	0.323
NIQE [21]	0.111	0.051	0.349
SRE [20]	0.309	0.196	0.400
MS-SSIM [19]	0.712	0.580	0.136
VMAF [14]	0.653	0.507	0.279
SAM [22]	0.525	0.377	0.583
ERQA [23]	0.523	0.406	0.650
LPIPS-VGG [24]	0.744	0.616	0.656
LPIPS-AlexNet [24]	0.569	0.464	0.574
NeuralSBS [25]	0.769	0.580	0.759
Proposed	**0.843**	**0.667**	**0.813**

Таблица 4: Результаты сравнения методов на наборе данных MSU Deinterlacer Benchmark [6]. В каждом столбце лучшее значение выделено жирным шрифтом.

Чтобы протестировать метод на явный ”подгон” параметров под данный набор видео, метод был представлен в виде модели машинного обучения с двумя параметрами и была проведена кросс-валидация. Результаты кросс-валидации можно увидеть в Таблице 5. Видно, что на валидации результат немного падает, однако в графиках в приложении к данной работе показано, что это лучшие результаты на данных разбиениях среди прочих методов.

Разбиение	1	2	3	4	5	6	7	Средний
Результат	0.858	0.879	0.869	0.840	0.875	0.869	0.752	0.849
Валидация	0.745	0.620	0.670	0.889	0.629	0.751	0.937	0.749

Таблица 5: SROCC для кросс-валидации на 7 разбиениях на наборе данных MSU Deinterlacer Benchmark [6]

Кроме того, чтобы убедиться, что найденные параметры устойчивы, были посчитаны оптимальные параметры для каждого разбиения отдельно. Результаты в Таблице 6 показывают, что данные параметры действительно несущественно отличаются для каждого разбиения.

Разбиение	1	2	3	4	5	6	7
A	0.15	0.20	0.15	0.20	0.15	0.20	0.15
B	0.80	0.85	0.85	0.85	0.75	0.80	0.85

Таблица 6: Оптимальные параметры для каждого разбиения

Чтобы протестировать метод на неявный ”подгон” (на уровне идеи) был собран отдельный набор данных с субьективными оценками, сформированный на базе набора данных YT-UGC [30] и предложенный метод протестирован на нем без какой-либо оптимизации под новый набор. Результаты сравнения на наборе данных YT-UGC [30] можно увидеть на Рис. 20. Как можно увидеть, метод показывает себя хорошо в том числе и на наборе данных с субьективными оценками, собранном из YT-UGC [30].

Результаты проведенных экспериментов показывают, что метод имеет устойчивые параметры, несущественно ”подогнан” под набор данных [6], но тем не менее хорошо себя показывает на совершенно другом наборе данных, не связанном с [6].

Это, в свою очередь, позволяет предположить, что метод будет хорошо себя показывать, то есть иметь высокую корреляцию с субьективным восприятием людей на произвольных данных.

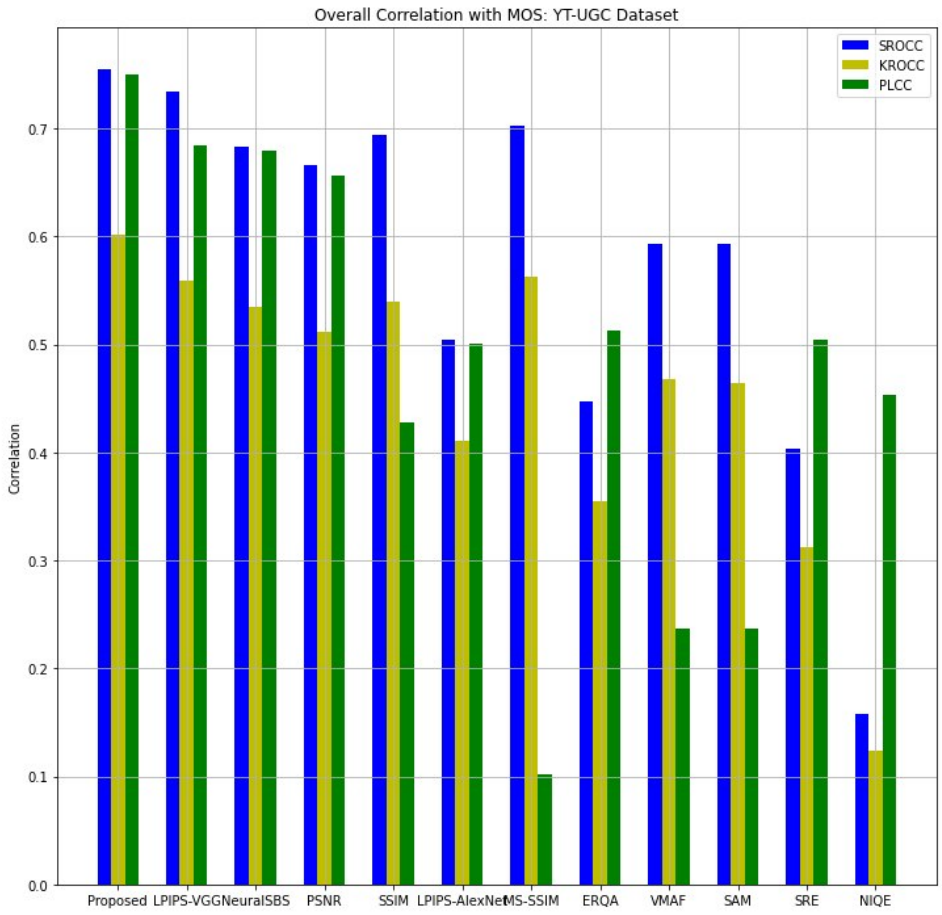


Рис. 20: Результаты сравнения на наборе данных YT-UGC [30]

7. Программная реализация

Система для оценки качества алгоритмов деинтерлейсинга была реализована в виде программы, состоящей из нескольких модулей

- Генерация набора видео
- Валидация присланного результата
- Расчет результатов методов оценки качества
- Построение графиков и таблицы лидеров
- Построение визуализаций
- Перевод данных в формат, нужный для субьективного сравнения

Метод оценки качетсва был написан в виде монолитной программы, которая принимает на вход видео, а возвращает число — свою оценку.

Для решения поставленных задач был выбран язык программирования Python 3 из-за его простоты и наличия огромного количества библиотек. Для работы с изображениями были использованы библиотеки OpenCV и Scikit-Image. Для построения графиков и визуализаций

использовались библиотеки Matplotlib и Seaborn. Для работы с таблицами использовалась библиотека Pandas. Для ускорения векторных вычислений использовалась библиотека NumPy. Для работы с нейронными сетями использовалась библиотека PyTorch. В качестве среды разработки были выбраны Jupyter Notebook на этапе исследований из-за удобства визуализаций и PyCharm на этапе разработки финального решения из-за удобного автозаполнения и подсказок программисту.

Суммарное количество строк кода на Python 3 — 2856. Для обработки и хранения видео использовался инструмент под названием ffmpeg, а для расчета результатов методов оценки качетсва видео использовался инструмент под названием vqmt. Для проведения субьективных сравнений был использован интернет-ресурс под названием subjectify.us.

8. Заключение

В рамках данной выпускной квалификационной работы было проведено исследование существующих методов исправления чересстрочного видео и существующих методов оценки качества исправления чересстрочного видео. Исследование показало, что лучшие по качеству на текущий момент методы исправления чересстрочного видео основываются на глубоких нейросетях.

Был разработан разнообразный и неизбыточный набор данных, позволяющий оценить качество алгоритма восстановления чересстрочности. Была разработана система для автоматической оценки качетства алгоритмов деинтерлейсинга. Было проведено субъективное исследование и разработан первый в своем роде набор данных результатов работы алгоритмов деинтерлейсинга с субъективными оценками качества. Этот набор данных может быть использован исследователями для решения подобных проблем в дальнейшем.

На основе этих результатов был разработан метод оценки качества исправления чересстрочного видео. Были проведены исследования и тестирование метода. В процессе экспериментальной оценки было показано, что метод устойчив к переобучению и превосходит сравниваемые подходы на двух различных наборах данных.

9. Литература

[1] https://invideo.io/blog/video-marketing-statistics/

[2] https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

[3] https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

[4] https://en.wikipedia.org/wiki/Structural_similarity

[5] https://videoprocessing.ai

[6] https://videoprocessing.ai/benchmarks/deinterlacer.html

[7] Kwon O., Sohn K., Lee C. Deinterlacing using directional interpolation and motion compensation //IEEE Transactions on Consumer Electronics. – 2003. – Т. 49. – №. 1. – С. 198-203.

[8] Zhao Y. et al. Multi-frame Joint Enhancement for Early Interlaced Videos //arXiv preprint arXiv:2109.14151. – 2021.

[9] Zhu H. et al. Real-time deep video deinterlacing //arXiv preprint arXiv:1708.00187. – 2017.

[10] https://www.vegascreativesoftware.info/us/forum/deinterlacing-challenge-2019–114485/

[11] Wang Y., Inguva S., Adsumilli B. YouTube UGC dataset for video compression research //2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP). – IEEE, 2019. – С. 1-5.

[12] https://www.ffmpeg.org

[13] https://en.wikipedia.org/wiki/K-means_clustering

[14] Li Z. et al. VMAF: The journey continues //Netflix Technology Blog. – 2018. – Т. 25.

[15] https://www.subjectify.us

[16] https://en.wikipedia.org/wiki/Bradley–Terry_model

[17] https://www.compression.ru/video/quality_measure/vqmt_download.html

[18] https://en.wikipedia.org/wiki/Video_super-resolution

[19] Wang Z., Simoncelli E. P., Bovik A. C. Multiscale structural similarity for image quality assessment //The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. – Ieee, 2003. – Т. 2. – С. 1398-1402.

[20] Chang H. W. et al. Sparse feature fidelity for perceptual image quality assessment //IEEE Transactions on Image Processing. – 2013. – Т. 22. – №. 10. – С. 4007-4018.

[21] Mittal A., Soundararajan R., Bovik A. C. Making a “completely blind” image quality analyzer //IEEE Signal processing letters. – 2012. – Т. 20. – №. 3. – С. 209-212.

[22] Alparone L. et al. Multispectral and panchromatic data fusion assessment without reference //Photogrammetric Engineering & Remote Sensing. – 2008. – Т. 74. – №. 2. – С. 193-200.

[23] Kirillova A. et al. ERQA: Edge-Restoration Quality Assessment for Video Super-Resolution //arXiv preprint arXiv:2110.09992. – 2021.

[24] Zhang R. et al. The unreasonable effectiveness of deep features as a perceptual metric //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2018. – С. 586-595.

[25] Khrulkov V., Babenko A. Neural Side-By-Side: Predicting Human Preferences for No-Reference Super-Resolution Evaluation //Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2021. – С. 4988-4997.

[26] Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition //arXiv preprint arXiv:1409.1556. – 2014.

[27] Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks //Advances in neural information processing systems. – 2012. – Т. 25.

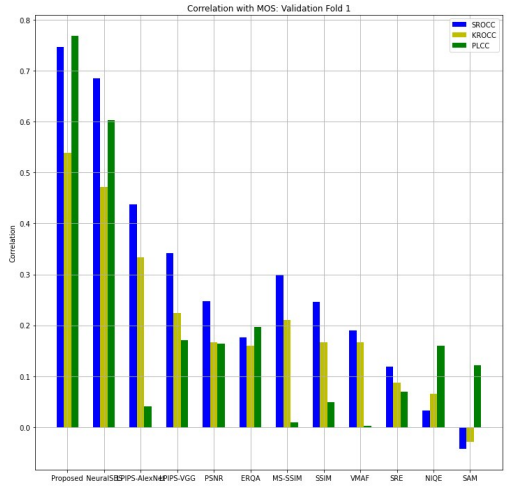
[28] Kim W. et al. Deep video quality assessor: From spatio-temporal visual sensitivity to a convolutional neural aggregation network //Proceedings of the European Conference on Computer Vision (ECCV). – 2018. – С. 219-234.

[29] Vaswani A. et al. Attention is all you need //Advances in neural information processing systems. – 2017. – Т. 30.

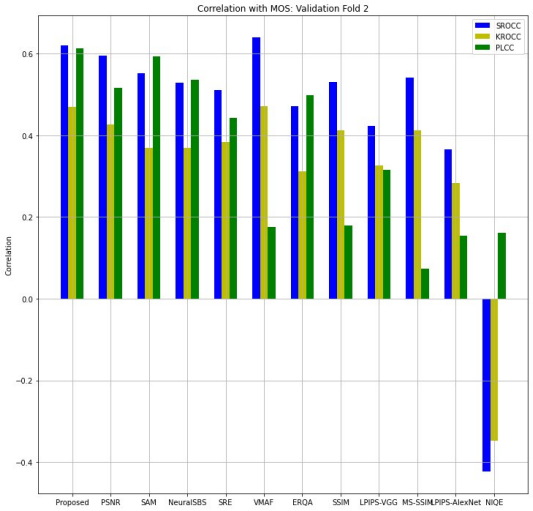
[30] Wang Y., Inguva S., Adsumilli B. YouTube UGC dataset for video compression research //2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP). – IEEE, 2019. – С. 1-5.

10. Приложение

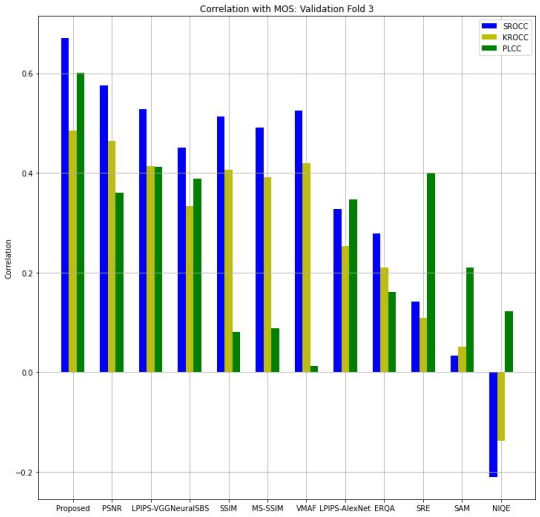
10.1. Сравнения метрики с другими метриками на различных разбиениях датасета MSU Deinterlacer Benchmark [6]



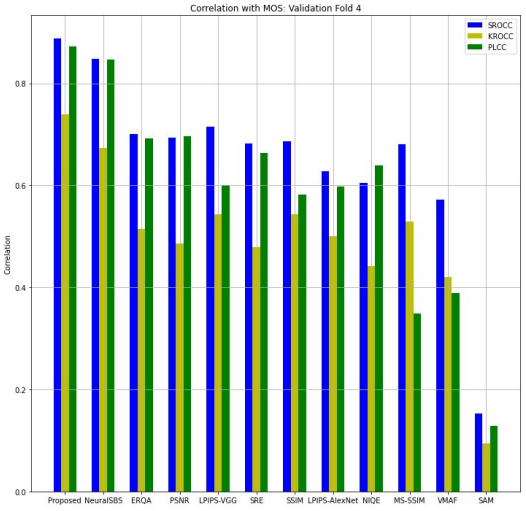
(a) Результаты сравнения на разбиении 1 набора данных [6]



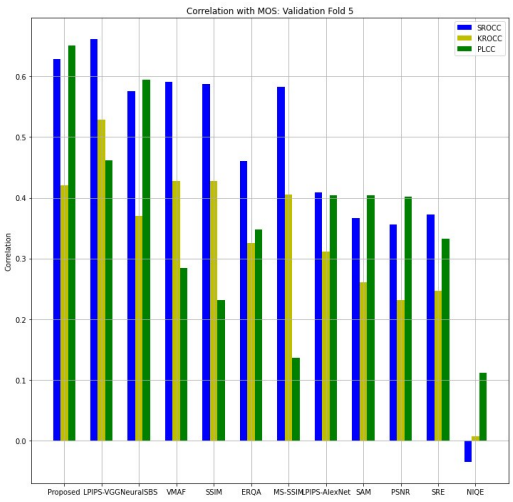
(b) Результаты сравнения на разбиении 2 набора данных [6]



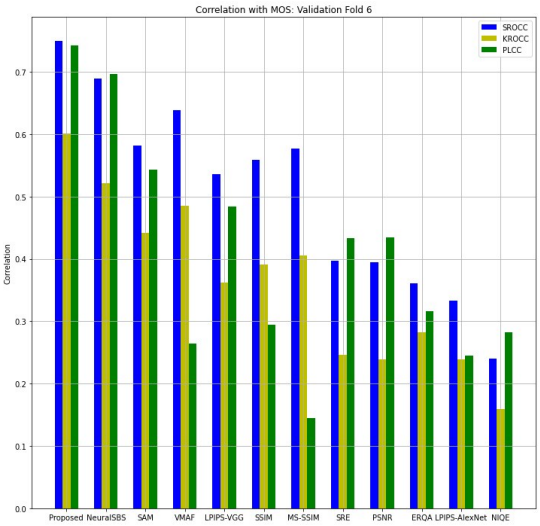
(a) Результаты сравнения на разбиении 3 набора данных [6]



(b) Результаты сравнения на разбиении 4 набора данных [6]



(a) Результаты сравнения на разбиении 5 набора данных [6]



(b) Результаты сравнения на разбиении 6 набора данных [6]

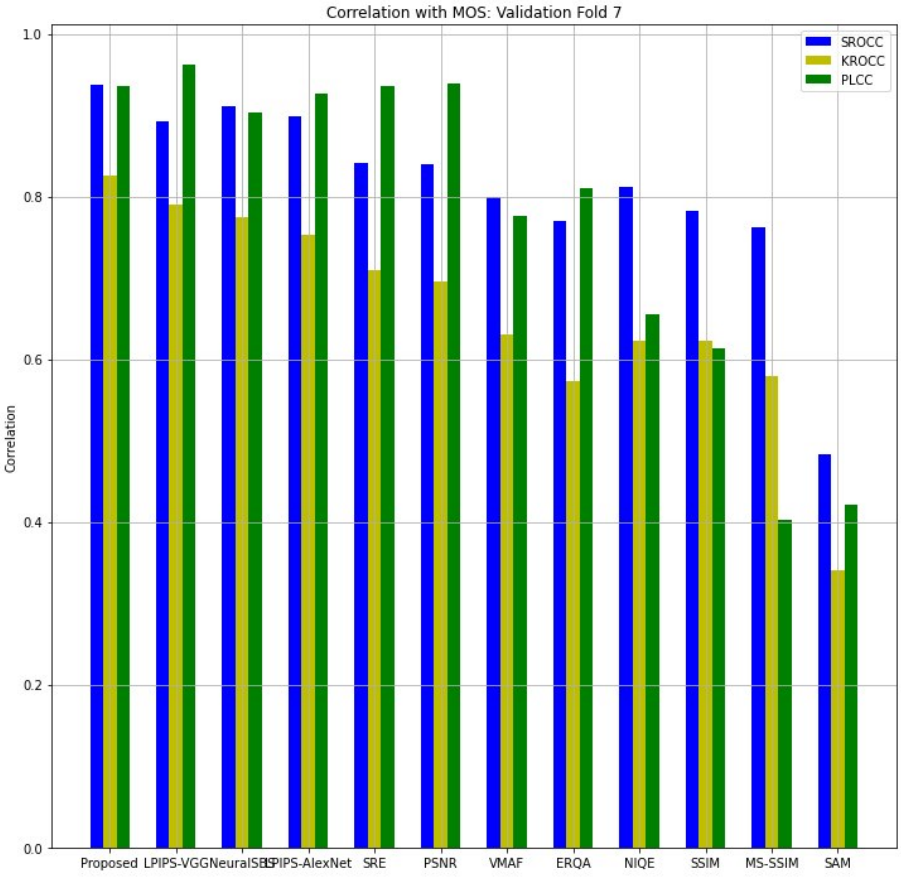


Рис. 24: Результаты сравнения на разбиении 7 датасета [6]