**Remote Heart Rate Detector**

**Дистанционный детектор пульса**

**Документ составил студент НИТУ МИСиС**

**Кущ Артем**

**БПМ-18-1**

Оглавление

[Теоретическая часть 3](#_Toc59979801)

[Обработка видео 3](#_Toc59979802)

[Поиск лиц 3](#_Toc59979803)

[Поиск лиц на CPU 3](#_Toc59979804)

[Поиск лиц на GPU 3](#_Toc59979805)

[Выбор области (ROI) 3](#_Toc59979806)

[Понижение шумов видеоизображения 3](#_Toc59979807)

[Пирамида и фильтр Гаусса 3](#_Toc59979808)

[Преобразование Фурье для 3-ех мерного тензора 5](#_Toc59979809)

[Amplify filter 5](#_Toc59979810)

[Понижение шумов сигнала 5](#_Toc59979811)

[Детрендинг сигнала 5](#_Toc59979812)

[Линейная интерполяция 6](#_Toc59979813)

[Окно Хэмминга 7](#_Toc59979814)

[Алгоритм 7](#_Toc59979815)

[Улучшение алгоритма с применением машинного обучения 8](#_Toc59979816)

[*FastICA* 8](#_Toc59979817)

[Практическая часть 9](#_Toc59979818)

[Запуск программы 9](#_Toc59979819)

[Источники 11](#_Toc59979820)

# Теоретическая часть

## Обработка видео

### Поиск лиц

Ранее в курсовой работе я подробно рассматривал различные подходы для поиска лица на фото: Elastic Graph Matching, CMM/HMM, PCA, AMM/ASM, CNN-архитектуры.

### Поиск лиц на CPU

За основу была выбрана модель типа Elastic Graph Matching – фильтры Хаара. Модель обладает хорошей точностью и производительностью. Основная проблема заключается в непостоянстве размера рамки поиска для каждого кадра одного видео.

Для решения этой проблемы необходимо обработать моделью все кадры, найти центр для каждого, усреднить размер рамки и с помощью медианного фильтра избавиться от выбросов координат центра (ошибочное определения лица).

### Поиск лиц на GPU

За основу был выбран ансамбль CNN-архитектур – FastMTCNN-pytorch. Эффективная, облегченная и улучшенная архитектура MTCNN. На CPU работает в несколько раз медленнее каскадов Хаара. Некоторые сравнения из статьи автора по скорости обработки изображений на GPU - P100:

|  |  |  |  |
| --- | --- | --- | --- |
| Package | FPS (1080x1920) | FPS (720x1280) | FPS (540x960) |
| facenet-pytorch | 12.97 | 20.32 | 25.50 |
| facenet-pytorch (non-batched) | 9.75 | 14.81 | 19.68 |
| dlib | 3.80 | 8.39 | 14.53 |
| mtcnn | 3.04 | 5.70 | 8.23 |

## Выбор области (ROI)

Было выбрано 4 основных области:

* исходное вырезанное лицо;
* лоб;
* левая щека;
* правая щека.

## Понижение шумов видеоизображения

### Пирамида и фильтр Гаусса

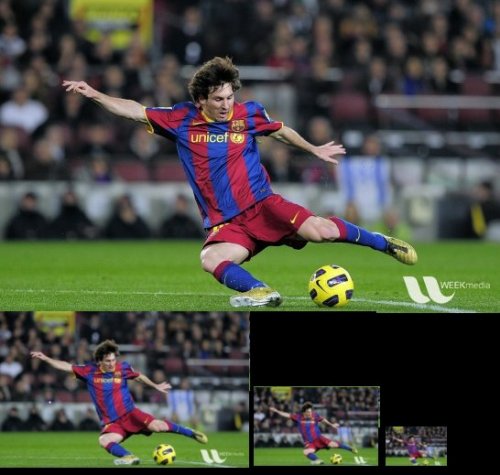
При покадровом извлечении видео все элементы одинакового размера. Но после применения алгоритмов и нейросетей для выявления лиц на кадрах зачастую приходится работать с изображениями разного разрешения.

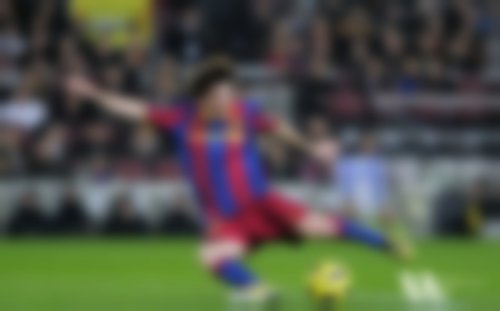
В этом случае нам нужно будет создать набор изображений с различным разрешением и искать объект во всех элементах. Эти наборы изображений с разным разрешением называются пирамидами изображений (потому что, когда они хранятся в стопке с самым большим изображением внизу и самым маленьким изображением вверху, они выглядят как пирамида).

Существует два вида пирамид изображений:

1. пирамида Гаусса
2. пирамида Лапласа

Более высокий уровень (низкое разрешение) в Гауссовой пирамиде формируется путем удаления последовательных строк и столбцов в изображении более низкого уровня (более высокое разрешение). Затем каждый пиксель на более высоком уровне формируется вкладом от 5 пикселей на нижележащем уровне с гауссовыми весами. Таким образом, изображение . становится изображением . Таким образом, площадь уменьшается до одной четвертой от первоначальной площади. Это называется Octave. Аналогично, когда мы поднимаемся в пирамиде (т. е. разрешение уменьшается).

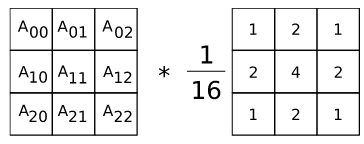
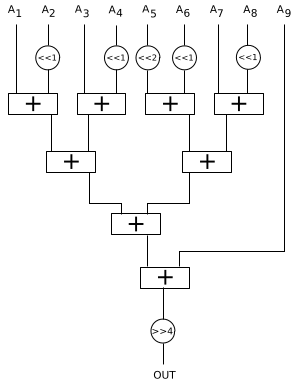




Фильтр Гаусса — электронный фильтр, чьей импульсной переходной функцией является функция Гаусса.

Фильтр Гаусса, как и медианный фильтр, используется для устранения шума в кадре, однако у него есть и побочный эффект — размытие изображения.

Пример для матрицы 3x3:



### Преобразование Фурье для 3-ех мерного тензора

Преобразование Фурье (символ ℱ) — операция, сопоставляющая одной функции вещественной переменной другую функцию вещественной переменной. Эта новая функция описывает коэффициенты («амплитуды») при разложении исходной функции на элементарные составляющие — гармонические колебания с разными частотами.

Преобразование Фурье функции вещественной переменной является интегральным и задаётся следующей формулой:

Применяя данную формулу для 4-ех мерного тензора (набор кадров) по нулевой оси, можем получить частоты необходимых гармонических колебаний. Все сигналы, с частотой меньше 50 и больше 175 необходимо занулить. Данный диапазон соответствует пульсу человека в покое и после силовых упражнений. Обратное преобразование Фурье даст необходимый результат. Если буфер слишком мал, зачастую почти все пиксели кадра будут 0.

### Amplify filter

Усиление результата обработки фильтром Фурье. По умолчанию коэффициент домножения равен 30, но рекомендуется изменять данный параметр вручную в зависимости от освещенности и прочих факторов, таких как частые повороты головой и автобаланс белого.

## Понижение шумов сигнала

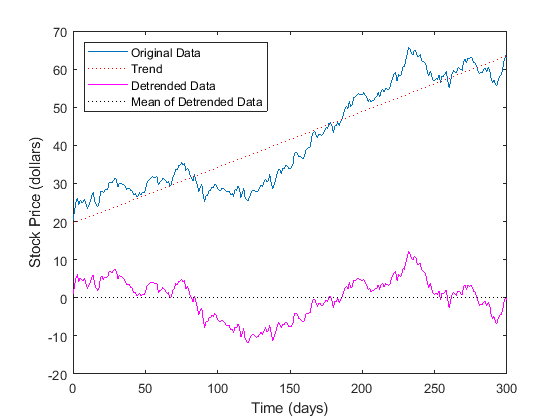
### Детрендинг сигнала

Пусть дан ограниченный временной ряд длины N, где , интегрирую или суммирую переведем его в неограниченный процесс :

– кумулятивная сумма.

Далее, ряд разбивается на временные окна длиной , а в каждом окне высчитывается локальный тренд с помощью метода наименьших квадратов. Пусть показывает результирующую кусочную последовательность прямолинейных подгонок. Затем вычисляется среднеквадратичное отклонение от тренда, флуктуация:

Пример



### Линейная интерполяция

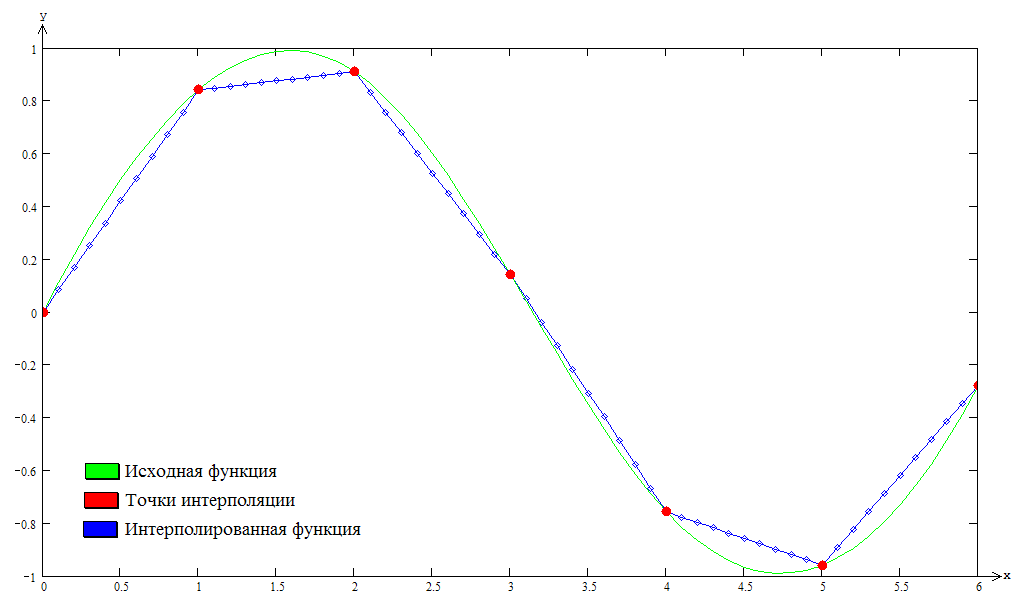
интерполяция алгебраическим двучленом  функции f, заданной в двух точках x0 и x1 отрезка [a, b]. В случае, если заданы значения в нескольких точках, функция заменяется кусочно-линейной функцией.

Геометрически это означает замену графика функции  прямой, проходящей через точки  и .

Уравнение прямой имеет вид:

Это и есть формула линейной интерполяции.

Пример:



### Окно Хэмминга

Окно Хэмминга определяется следующим образом:

Фильтр “Хэмминг” был назван в честь Р. У. Хэмминга, соратника Дж. У. Тьюки, и описан в книге "Blackman and Tukey". Он был рекомендован для сглаживания усеченной автоковариантной функции во временной области. Большинство ссылок на окно Хэмминга взяты из литературы по обработке сигналов, где оно используется как одна из многих оконных функций для сглаживания значений. Она также известна как аподизация (что означает “удаление границ”, то есть сглаживание разрывов в начале и конце дискретизированного сигнала) или функция сужения.

## Алгоритм

В первую очередь необходимо разметить положение лиц на каждом кадре с помощью модели MTCNN и пересобрать новое “обрезанное” видео только с кадрами лица.

Дале необходимо вырезать интересующие ROI-области: щёки (левая и правая), лоб. Был проведен ряд тестов с другими ROT-областями, а конкретно нос и подбородок, но результат получился не удовлетворительный, а ошибки были слишком велики.

После получение нескольких ROI-областей, необходимо для каждой из них выбрать размер буфера, включающий в себя X последних кадров. Размер буфера может варьироваться от 30 кадров до максимального размера видеоряда. Чем длиннее видеоизображение, тем точнее будет предсказание алгоритма.

Стоит отметить, что необходимо обрабатывать только зелёный канал каждого кадра, т.к. влияние красного и синего канал вносит дополнительные шумы, но никак не влияет на предсказание алгоритма. Следовательно, нет смысла использовать каналы G и B.

Далее, для выбранного буфера применяется пирамида Гаусса и фильтр Гаусса для устранения шумов в кадре. Побочный эффект в виде размытия никак не влияет на результат. После обработки фильтром Гаусса необходимо применить преобразование Фурье для устранения высокочастотных и низкочастотных шумов (50 уд. /м. и 175 уд. /м.), применить небольшое усиление, а после чего совместить с начальным видео.

Для улучшения периодичности сигнала применим известные и популярные функции: детрендинг сигнала – удаление линейного тренда вдоль оси X, одномерная линейная интерполяция и окно Хэмминга.

В конце необходимо применить преобразование Фурье и вычислить наиболее повторяющуюся частоту сигнала. Стоит отметить, что частоты <50 и >180 учитывать нет смысла, т.к. они не соответствуют пульсу человека.

## Улучшение алгоритма с применением машинного обучения

Проведя ряд тестов, конечный результат не был удовлетворительным и содержал много ошибок. Было принято решение обрабатывать сигнал небольшими буферами без последнего преобразования Фурье и сохранять промежуточный сигнал в стек. По завершению работы алгоритма обработки к данному стеку с фрагментами сигналов применялся алгоритм машинного обучения, , а после применялось преобразования Фурье. Таким образом мы смогли добиться ещё большей периодичности сигнала и повышения точности. Рассмотрим данный алгоритм машинного обучения более подробно.

### *FastICA*

Анализ независимых компонент (ICA) пытается разложить множественный сигнал на независимые негауссовые сигналы. Например, звук обычно является сигналом, который состоит из сложения в каждый момент одиночных t-сигналов, идущих из нескольких источников. Вопрос заключается в том, возможно ли разделить эти источники, выделяя их из общего сигнала. Если допущение статистической независимости верно, слепое разделение независимых компонент смешанного сигнала даст очень хорошие результаты. Метод также применяется для анализа сигналов, которые могут быть и не смешанными.

Важно также учитывать, что если представлено  источников, нужно по меньшей мере  наблюдений, чтобы обнаружить исходные сигналы. В этом случае матрица квадратна (). Иначе получаем и исследуем недоопределённый () или переопределённый () случай.

Метод FastICA базируется на двух допущениях и трёх эффектах источников смешанного сигнала, что даёт очень хорошие результаты.

Двумя допущениями являются:

1. Источники сигналов независимы друг от друга.
2. Значения каждого источника сигнала имеют негауссово распределение.

Тремя эффектами источника смешанного сигнала являются:

1. Независимость: как в и допущении 1, источники сигналов независимы, однако их смесь не является независимой от источников, потому что смесь сигналов имеет одни и те же источники.
2. Нормальность: согласно центральной предельной теореме, распределение суммы независимых случайных переменных с конечной дисперсией стремится к гауссовому распределению. Попросту говоря, сумма двух независимых случайных переменных обычно имеет распределение более близкое к гауссовому, чем любое из двух исходных случайных переменных. Здесь мы рассматриваем каждый сигнал как случайную переменную.
3. Сложность: временна́я сложность любой смеси сигналов больше, чем сложность одного сигнала, более простого по его составляющим.

Эти принципы составляют базовые основы FastICA. Если сигналы, которые нам удалось извлечь из смеси, независимы, подобно исходным сигналам, и имеют негауссовые гистограммы или имеют малую сложность, подобную сигналу источников, они должны быть сигналами источников.

# Практическая часть

Программа была реализована на языке Python3. Использовались следующие вспомогательные пакеты:

* *numpy*
* *cv2*
* *matplotlib.pyplot*
* *PIL*
* *matplotlib.backends.backend\_agg*
* *types*
* *tqdm*
* *from scipy.signal import medfilt, spline\_filter*
* *scipy.fftpack*
* *facenet\_pytorch*
* *torch*
* *warnings*
* *sklearn.decomposition*

## Запуск программы

Для корректной работы программы необходимо установить на ПК python3 версии не ниже 3.6 и необходимые библиотеки с помощью команды:

***>>pip install <название пакета>***

Запуск осуществляется с помощью команды в терминале:

***>> python3.X HeartRateDetector.py <название файла>***

Настройка параметров производится в файле конфигурации – **settings.cfg**:

*[default]*

*visualize = False*

*FastICA = True*

*[video]*

*idx\_start = 1*

*idx\_end = None*

*step = 2*

*[color magnification]*

*levels = 3*

*[areas]*

*areas = ['LeftCheek', 'RightCheek', 'FullFace', 'Forehead']*

*areas\_for\_ICA = ['FullFace', 'Forehead', 'LeftCheek', 'RightCheek']*

*[signal]*

*buff\_size = 50*

*channel = 1*

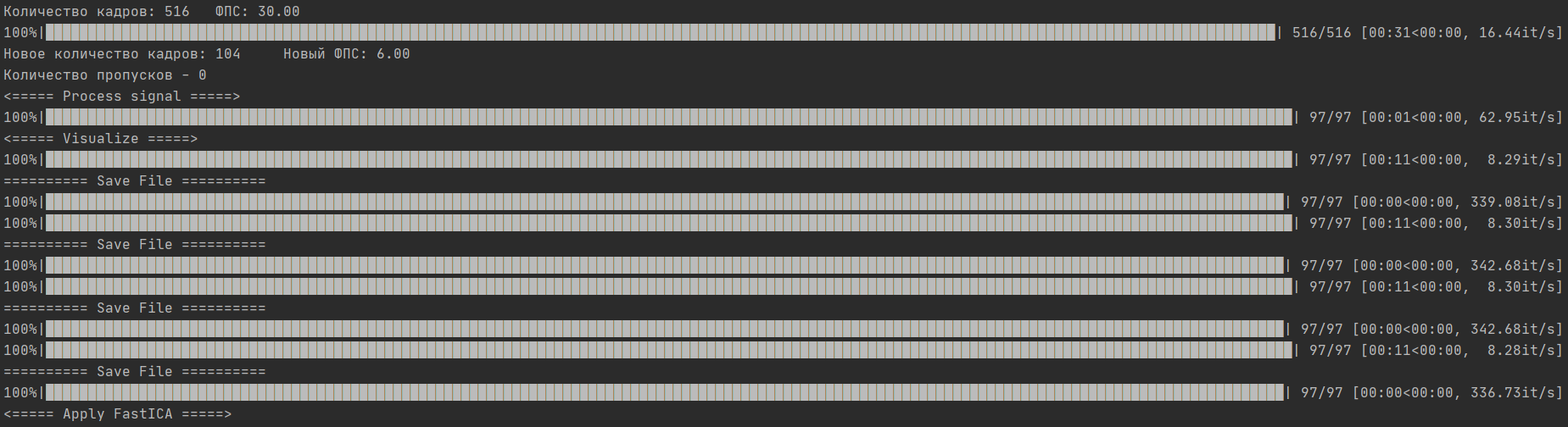
*[save]*

*speedx = 0.25*

*filename\_video = 'output/OUTPUT'*

*filename\_ICA = 'output/ICA'*

В терминале осуществляется вывод полезной информации:



# Источники

* <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_pyramids/py_pyramids.html>
* <https://ru.wikipedia.org/wiki/Фильтр_Гаусса>
* <https://ru.wikipedia.org/wiki/Признаки_Хаара>
* <https://ru.wikipedia.org/wiki/Преобразование_Фурье>
* <https://ru.wikipedia.org/wiki/Линейная_интерполяция>
* <https://ru.wikipedia.org/wiki/Кусочно-линейная_функция>
* <https://www.statisticshowto.com/detrend-data/>
* <https://www.investopedia.com/terms/d/detrend.asp>
* <https://en.wikipedia.org/wiki/Detrended_fluctuation_analysis>
* <https://en.wikipedia.org/wiki/Window_function#Hann_and_Hamming_windows>
* <https://cran.r-project.org/web/packages/fastICA/fastICA.pdf>
* <https://en.wikipedia.org/wiki/FastICA>
* <https://ru.wikipedia.org/wiki/Анализ_независимых_компонент>
* <https://scikit-learn.org/stable/auto_examples/decomposition/plot_ica_blind_source_separation.html>
* <http://tetraquark.ru/archives/311>
* <https://research.ics.aalto.fi/ica/fastica/>
* <http://mdp-toolkit.sourceforge.net/>