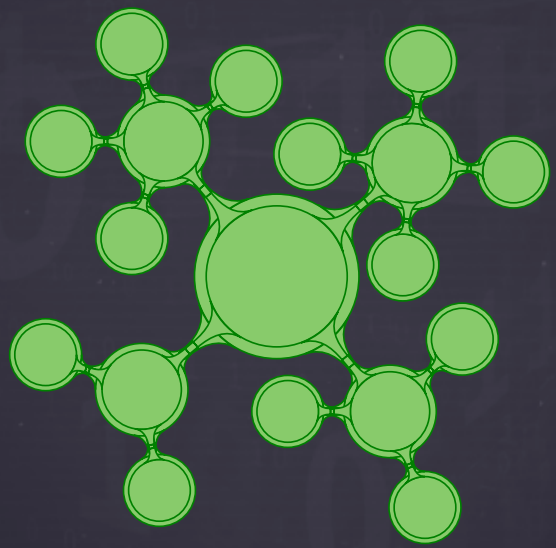
МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)

Поиск событий в видео



Никитин Илья Константинович,
асп. каф. 806 МАИ
twitter: @w_495
почта: w@w-495.ru
nikitin.i@tvzavr.ru
w495@yandex-team.ru

Кто я

аспирант, преподаватель

разработчик

разработчик

Яндекс

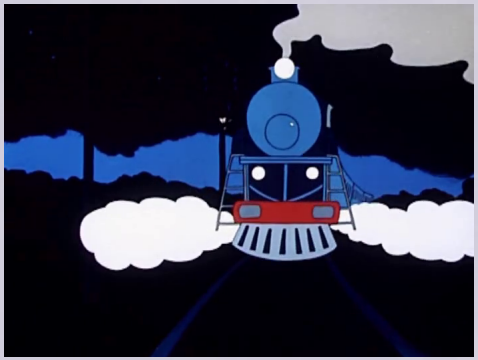
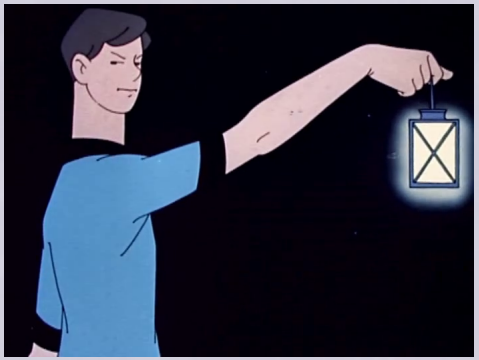
Содержание

Что такое видео? — Последовательность ...

Видео — последовательность фактов или событий

- События развиваются во времени.
- Свойства событий — *пространственная* характеристика видео,
- продолжительность и порядок фактов — *временная*.

Пример:


;

;

.

(Кадры событий из начала мультфильма «Дядя Стёпа — милиционер»)

Зачем искать события: видео-аналитика

Полуавтоматическое наблюдение

⇐ Пульт видео-наблюдения:

- много камер;
  - → система видео-наблюдения банка ≈ 200 камер.
- просматривает человек;
- не всегда понятно на какой из камер «тревога»;

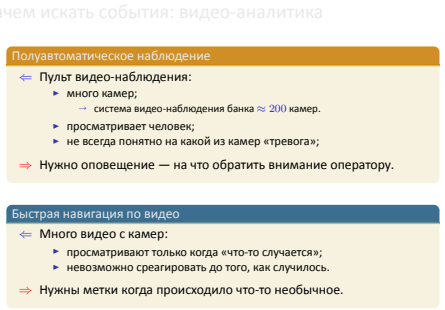
⇒ Нужно оповещение — на что обратить внимание оператору.

Быстрая навигация по видео

⇐ Много видео с камер:

- просматривают только когда «что-то случается»;
- невозможно среагировать до того, как случилось.

⇒ Нужны метки когда происходило что-то необычное.



За 20-30 лет человечество накопило большое количество видеоинформации. Речь идет не только о видео в интернете, но в первую очередь о камерах наблюдения. Этот массив информации невозможно просмотреть целиком. А анализировать видео, как это делают с текстом еще не научились. Нужно иметь возможность я в каком-то приближении индексировать видео и искать по нему. Причем делать это не обязательно в каких-то логических терминах, или терминах текстового поиска.

Зачем искать события: видео-поиск

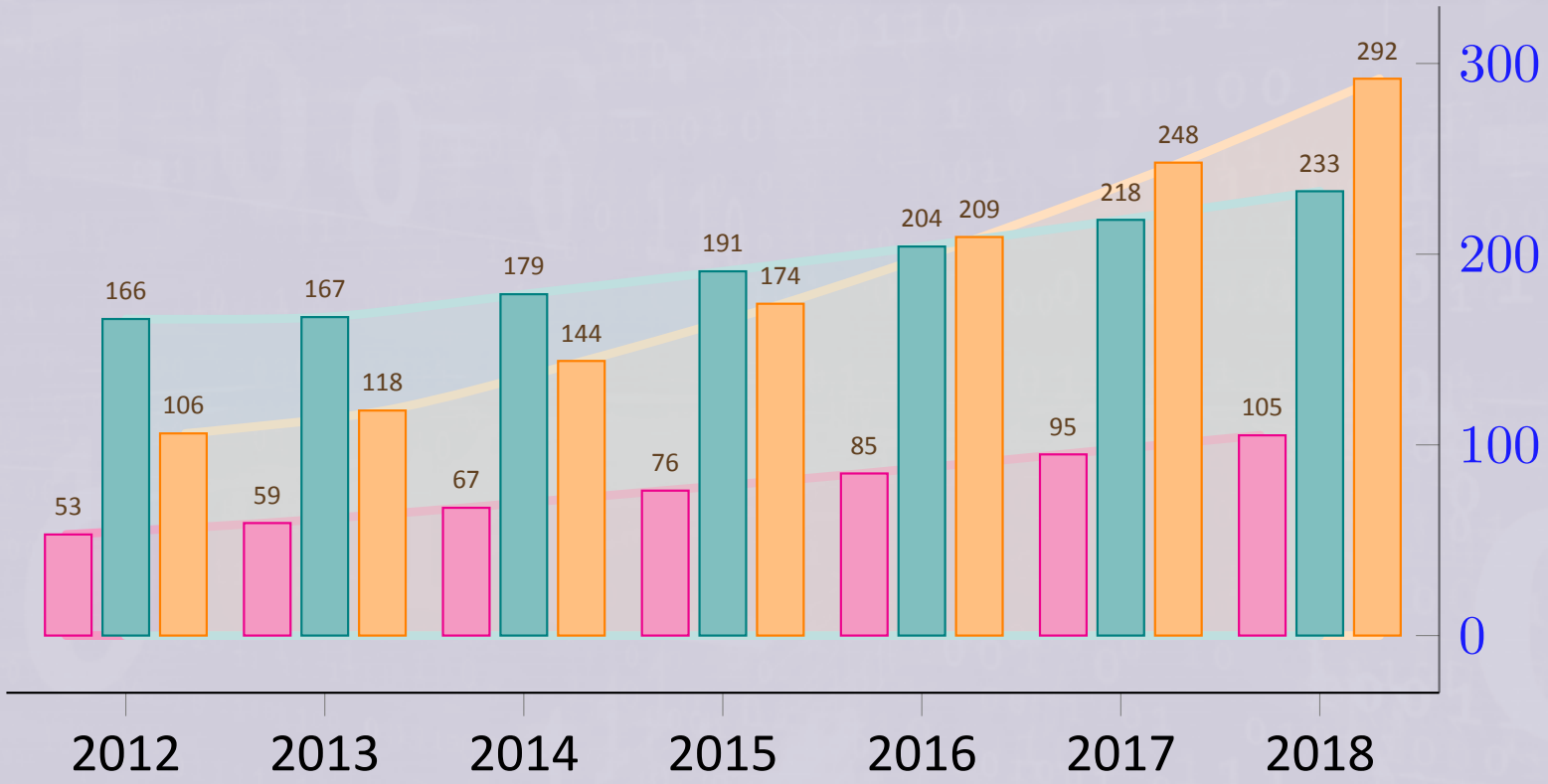
Поиск внутри видео

- поиск известного фрагмента;
- поиск неизвестного фрагмента по шаблону;
- поиск видео с заданными характеристиками;

Поиск похожих видео (поиск нечетких дубликатов)

- системы рекомендаций;
- ранжирование в поисковых системах;
- группировка новостей и других материалов;

Российский рынок онлайн-видео в 2012—2018 годах



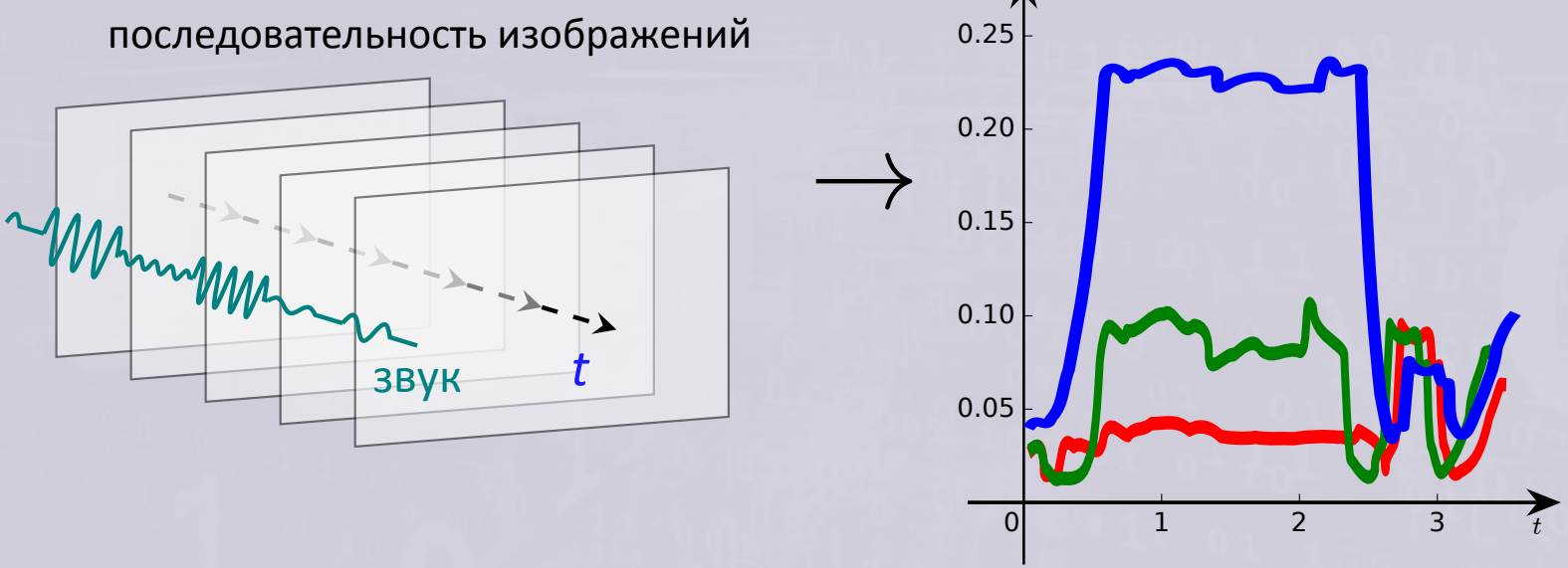
Поиск событий: временные ряды

Изначально имеем

- последовательность изображений;
- и последовательность аудио-отчетов.

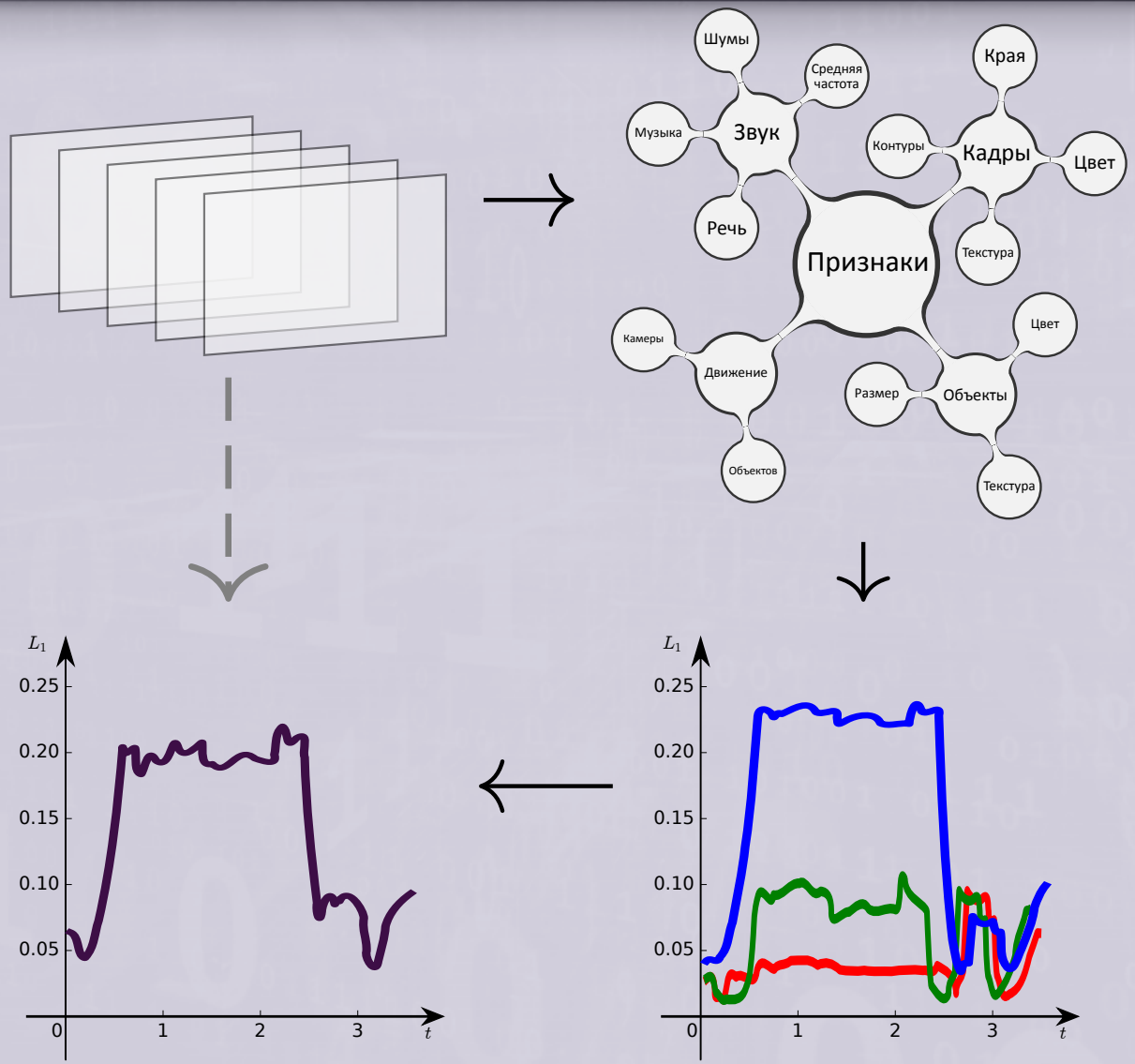
Хотим получить

- временной ряд;
- описание ситуации в видео.




Признаки
Звук
Шумы
Средняя частота
Музыка
Речь
Кадры
Края
Контуры
Цвет
Текстура
Движение
Камеры
Объектов
Объекты
Цвет
Размер
Текстура

Можно свести к одномерному случаю



Сведение к одномерному случаю

Зачем сводить к одномерному случаю

Сможем в явном виде применить:

- методы экономического анализа;
- методы обработки временных рядов из радиотехники;
- методы поиска разладок.

Почему сведение корректно

- можем свести из n-мерного:
  - нужны только весовые коэффициенты;
  - пример: $Y_{Y'P_rP_b} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$;
- сможем обобщить до n-мерного:
  - для визуальной информации n может быть > 3;
  - очень актуально для трехмерных камер.

Как свести к одномерному случаю

Дана последовательность кадров

$$F(t) = F_{rgb,t}$$

$$F_{rgb,t} = \begin{pmatrix} x_{1,1}, & \cdots & x_{1,m} \\ \cdots & \cdots & \cdots \\ x_{n,1}, & \cdots & x_{n,m} \end{pmatrix}_t$$

$$x_{i,j} = \{r_{i,j}; g_{i,j}; b_{i,j}\} \in \mathbb{R}^3_{[0,1]};$$

Работаем только с визуальным рядом

- может быть видео без звука;
- звука без картинки в видео не бывает.

Для каждого кадра

Значение по *выбранной* норме, например:

$$\|F_t\|_{L_1} = \frac{1}{n \cdot m} \sum_{i=0}^{n} \sum_{j=0}^{m} S(x_{i,j}), \text{ где}$$

$$S : \mathbb{R}^3_{[0,1]} \longmapsto \mathbb{R}_{[0,1]}$$

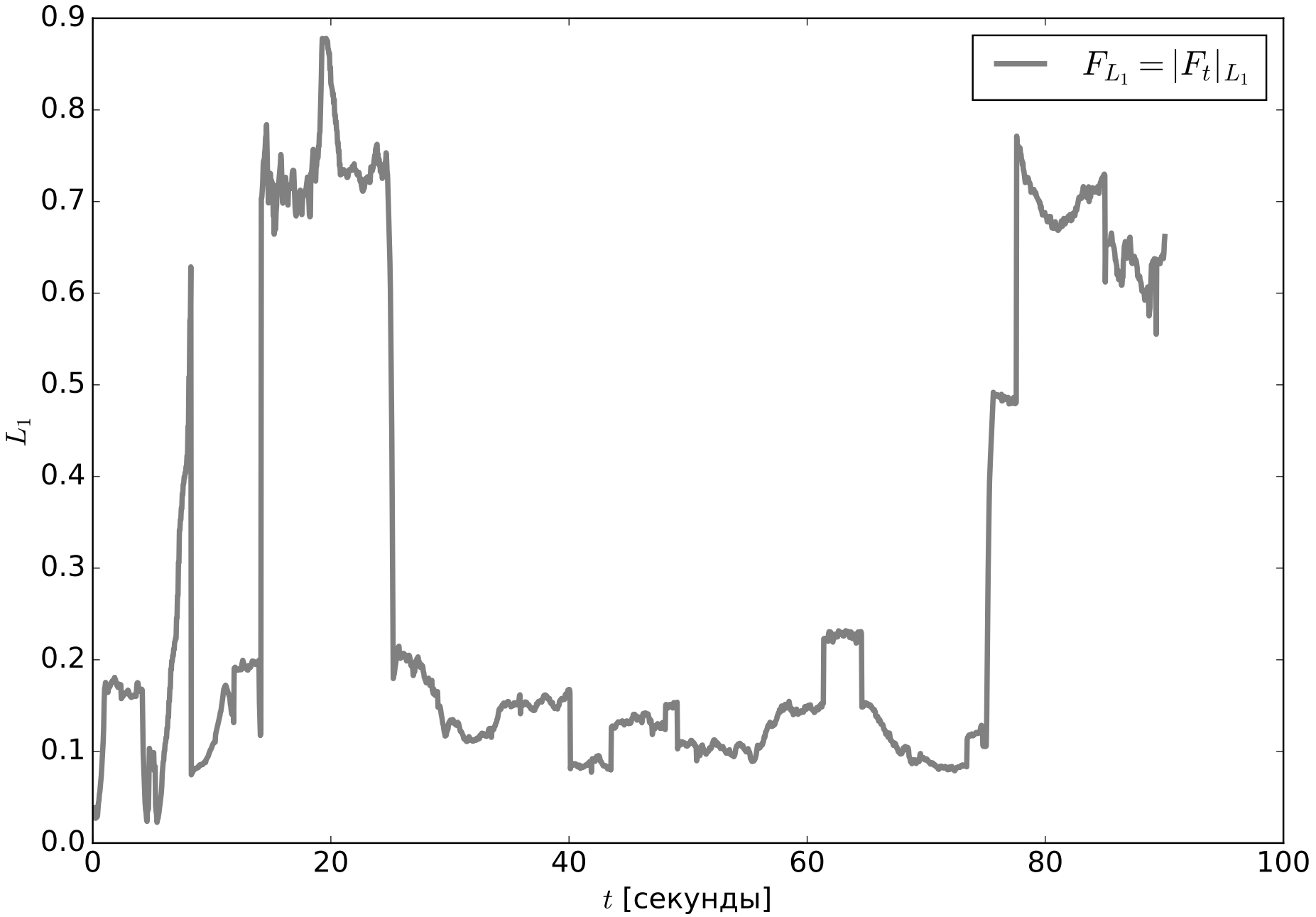
Например, $S(x) = |Y_{Y'P_rP_b}|$

Далее, строим ряд

$$F_{L_1}(t) = \|F_t\|_{L_1}$$

и пытаемся найти в нем «аномалии».

«Дядя Стёпа — милиционер», внешний вид сигнала F_{L_1}



Что считаем аномалией

Аномалия

- резкое изменение свойств наблюдаемого ряда
- в заранее неизвестный момент времени.

Иногда явление называют «разладкой».

Как связано с событиями

- Моменты аномалий — моменты событий.
- В художественном видео:
  - монтажные склейки или «сцены»;
  - собственные события сюжета.

Пороговые методы со статическим порогом

Как устроены

- разница соседних величин (по некоторой норме);
- заранее задается некоторый порог;
- превышения порога — «аномалия».

Плюсы

- просты в реализации, втч аппаратной;
- не требовательны к ресурсам.

Минусы

- требуется заранее знать порог;
- не применимо для разных типов видео;
- чувствительны к случайным всплескам;
- ловят только краткосрочные события.

Наивный пороговый метод

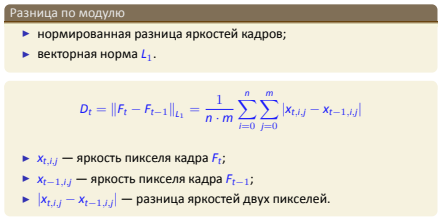
Разница по модулю

- нормированная разница яркостей кадров;
- векторная норма L_1.

$$D_t = \|F_t - F_{t-1}\|_{L_1} = \frac{1}{n \cdot m} \sum_{i=0}^{n} \sum_{j=0}^{m} |x_{t,i,j} - x_{t-1,i,j}|$$

- $x_{t,i,j}$ — яркость пикселя кадра F_t;
- $x_{t-1,i,j}$ — яркость пикселя кадра F_{t-1};
- $|x_{t,i,j} - x_{t-1,i,j}|$ — разница яркостей двух пикселей.

Наивный пороговый метод



Это самый простой и очевидный из вех возможных методов.

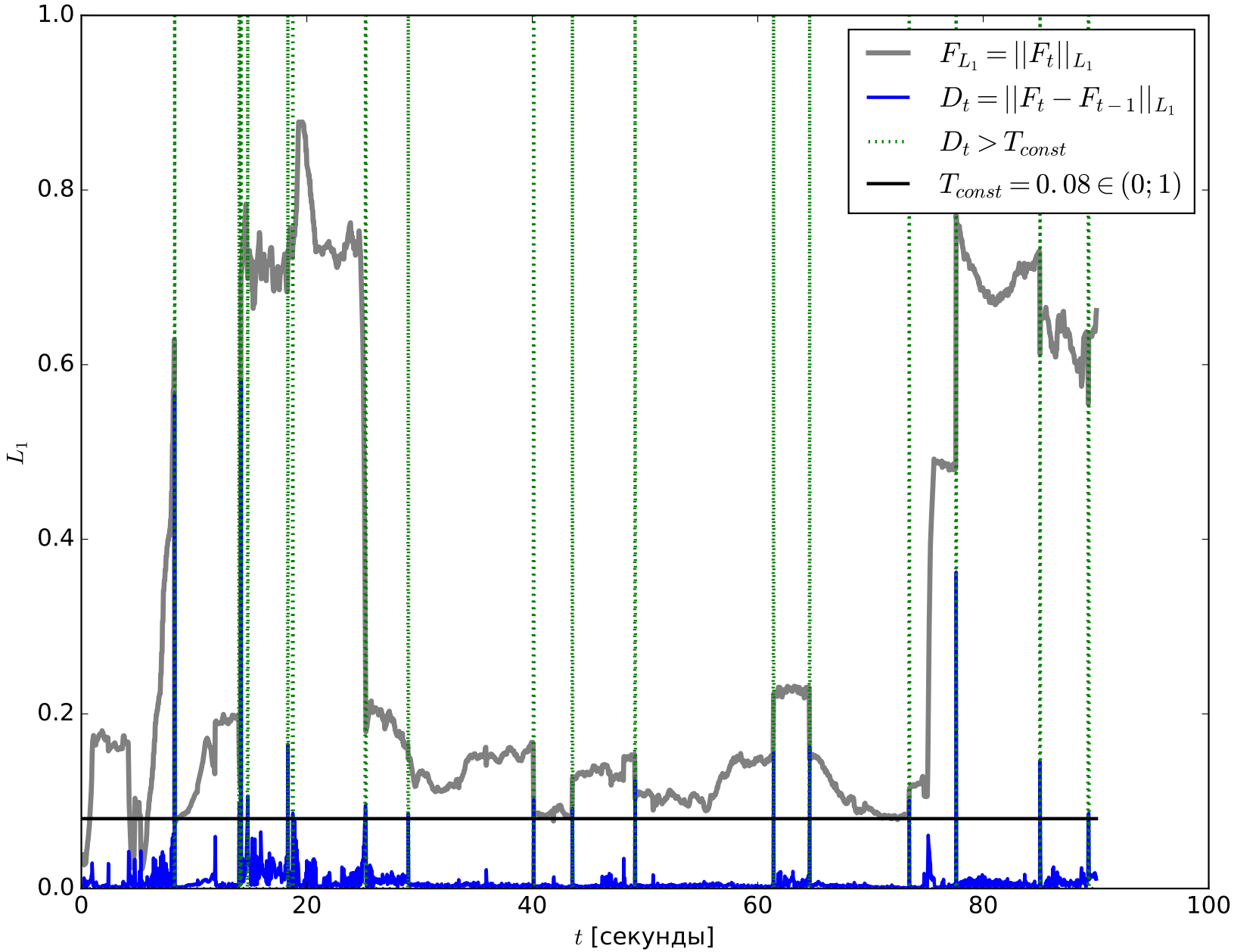
В качестве исследуемой величины тут используется значение нормированной попиксельной абсолютной разницы яркостей двух соседних кадров. В качестве нормы используется векторная норма L_1.

- $x_{t,i,j}$ — пиксель кадра F_t;
- $x_{t-1,i,j}$ — пиксель кадра F_{t-1};

Физический смысл величины $|x_{t,i,j} - x_{t-1,i,j}|$ заключается в том, насколько один пиксель отличается от другого.

Если средняя разница пикселей превысит заранее заданное число — порог, то мы считаем, что мы нашли «разладку».

Наивный пороговый метод



Мета-язык потоковой обработки

```
1   delay = DelayFilter()          # Фильтр линейной задержки.
2   orig  = delay(0)               # Входящий сигнал без изменения.
3   shift = ShiftSWFilter()        # Сдвиг сигнала на один кадр.
4   diff = orig - shift            # Разница соседних кадров.
5   norm = NormFilter()            # Норма сигнала.
6   T_CONST = 0.08                 # Значение порога. Константа.
7   threshold = orig > T_CONST     # Пороговый фильтр.
8
9   #  Фильтр нормированной попиксельной абсолютной разницы.
10  d_filter = diff | abs | norm(l=1)
11
12  #  Результирующий фильтр: точки разладок.
13  result_filter = d_filter | threshold
14
```

- Оператор «|» означает «конвейер».
- Фильтры собирается отложено до непосредственного применения.

Мета-язык потоковой обработки

```
delay = DelayFilter()       # Фильтр линейной задержки.
orig  = delay(0)            # Входящий сигнал без изменения.
shift = ShiftSWFilter()     # Сдвиг сигнала на один кадр.
diff = orig - shift         # Разница соседних кадров.
norm = NormFilter()         # Норма сигнала.
T_CONST = 0.08              # Значение порога. Константа.
threshold = orig > T_CONST # Пороговый фильтр.

# Фильтр нормированной попиксельной абсолютной разницы.
d_filter = diff | abs | norm(l=1)

# Результирующий фильтр: точки разладок.
result_filter = d_filter | threshold
```

- Оператор «|» означает «конвейер».
- Фильтры собирается отложено до непосредственного применения.

Для решения задачи мы разработали мета-язык на базе языка Python. Свойства мета-языка:

1. Основной сущностью является фильтр:
   - по-факту это просто абстрактный тип на языке Python с перегруженными операторами,
   - фильтры описывают набор действий над последовательностью кадров видео;
2. Любые прочие объекты приводятся к фильтру и действия с ними осуществляются как с фильтром;

Мета-язык потоковой обработки

```
delay = DelayFilter()       # Фильтр линейной задержки.
orig  = delay(0)            # Входящий сигнал без изменения.
shift = ShiftSWFilter()     # Сдвиг сигнала на один кадр.
diff = orig - shift         # Разница соседних кадров.
norm = NormFilter()         # Норма сигнала.
T_CONST = 0.08              # Значение порога. Константа.
threshold = orig > T_CONST  # Пороговый фильтр.

# Фильтр нормированной попиксельной абсолютной разницы.
d_filter = diff | abs | norm(l=1)

# Результирующий фильтр: точки разладок.
result_filter = d_filter | threshold
```

- Оператор «|» означает «конвейер».
- Фильтры собирается отложено до непосредственного применения.

Жизненный цикл фильтра состоит из двух этапов:

- объявление фильтра — при создании фильтра никаких вычислений над последовательностью кадров не происходит,
- выполнении фильтра — (вызов метода класса *filter_objects*) внутри фильтра происходит обработка последовательности объектов: кадров или результатов работы других фильтров;

Мета-язык потоковой обработки

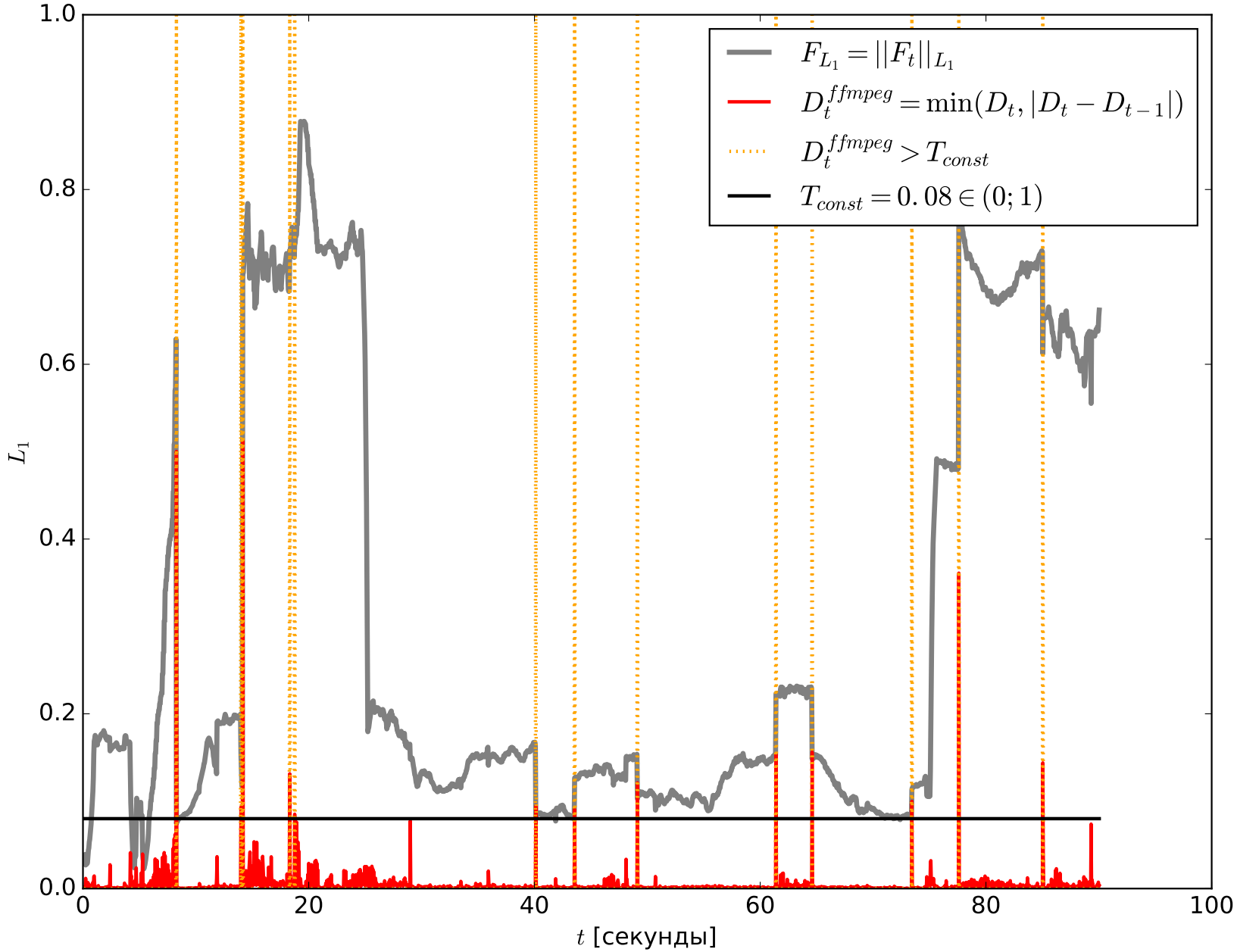
```
delay = DelayFilter()       # Фильтр линейной задержки.
orig  = delay(0)            # Входящий сигнал без изменения.
shift = ShiftSWFilter()     # Сдвиг сигнала на один кадр.
diff = orig - shift         # Разница соседних кадров.
norm = NormFilter()         # Норма сигнала.
T_CONST = 0.08              # Значение порога. Константа.
threshold = orig > T_CONST  # Пороговый фильтр.

# Фильтр нормированной попиксельной абсолютной разницы.
d_filter = diff | abs | norm(l=1)

# Результирующий фильтр: точки разладок.
result_filter = d_filter | threshold
```

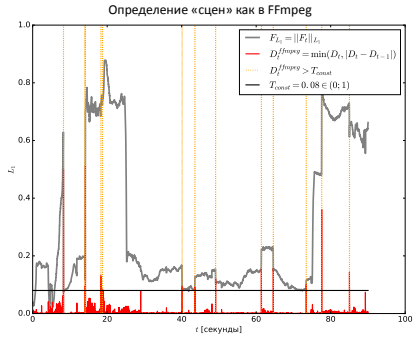
- Оператор «|» означает «конвейер».
- Фильтры собирается отложено до непосредственного применения.

Операции над фильтрами:

1. Любая операция над фильтрами кроме выполнения, приводит к созданию нового фильтра:
   - на этапе выполнения фильтра будут вычислены аргументы этой операции и выполнена сама операция;
   - порядок вычисления аргументов операций не определен, и на многоядерных архитектурах аргументы операций могут быть вычислены параллельно;
2. Основная операция над фильтрами — последовательное их применение — конвейер. Она тоже приводит к созданию нового фильтра.
3. Прочие операции над фильтрами приводят к изменению последовательности объектов или их свойств в процессе выполнения итогового фильтра.

Определение «сцен» как в FFmpeg





FFmpeg — набор свободных библиотек, которые позволяют записывать, обрабатывать и передавать цифровое видео в различных форматах. В том числе, FFmpeg содержит в себе возможность выделения «сцен».

В качестве исследуемой величины тут используется значение наименьшей величины из двух:

- нормированной разницы яркостей двух соседних кадров;
- и абсолютной разницы двух последовательных разниц.

Такой минимум используется для того, чтобы избежать всплесков на случай если график сигнала монотонно убывает или возрастает. Разница двух последовательных разниц имеет смысл скорости изменения яркости. Малая скорость изменения при значительном изменении говорит о том, что яркость равномерно меняется в некоторой окрестности данного кадра. При быстром, но плавном изменении яркостей кадров с точки зрения авторов порога из FFmpeg — разладки нет.

Описание FFmpeg-порога на мета-языке

```
1  delay = DelayFilter()          # Фильтр линейной задержки.
2  orig  = delay(0)               # Входящий сигнал без изменения.
3  shift = ShiftSWFilter()        # Сдвиг сигнала на один кадр.
4  diff = orig - shift            # Разница соседних кадров.
5  norm = NormFilter()            # Норма сигнала.
6  T_CONST = 0.08                 # Значение порога. Константа.
7  threshold = orig > T_CONST     # Пороговый фильтр.
8
9  # Фильтр нормированной попиксельной абсолютной разницы.
10 d_filter = diff | abs | norm(l=1)
11 # Абсолютная разница двух последовательных разниц.
12 d_diff_filter = d_filter | diff | abs
13
14 # Минимум между разницей и разницей разниц.
15 ffmpeg_like = Filter.union(d_filter, d_diff_filter) | min
16
17 # Результирующий фильтр: точки разладок.
18 result_filter = ffmpeg_like | threshold
19
```

Описание FFmpeg-порога на мета-языке

```
delay = DelayFilter()       # Фильтр линейной задержки.
orig  = delay(0)            # Входящий сигнал без изменения.
shift = ShiftSWFilter()     # Сдвиг сигнала на один кадр.
diff = orig - shift         # Разница соседних кадров.
norm = NormFilter()         # Норма сигнала.
T_CONST = 0.08              # Значение порога. Константа.
threshold = orig > T_CONST # Пороговый фильтр.

# Фильтр нормированной попиксельной абсолютной разницы.
d_filter = diff | abs | norm(l=1)
# Абсолютная разница двух последовательных разниц.
d_diff_filter = d_filter | diff | abs

# Минимум между разницей и разницей разниц.
ffmpeg_like = Filter.union(d_filter, d_diff_filter) | min

# Результирующий фильтр: точки разладок.
result_filter = ffmpeg_like | threshold
```

В приведенном листинге применяется функция *Filter.union*. В качестве аргументов передаются фильтры, и выходом функции тоже будет фильтр. *Filter.union* объединяет выходные последовательности (потоки) своих аргументов, так, что единица результирующей последовательности функции *Filter.union* содержит в себе соответствующие единицы последовательности аргументов. Единицей последовательности в данном случае выступает значение вычисляемой функции для текущего кадра. Последовательности аргументов должны быть синхронизированы по времени.

К результату функции *Filter.union* применяется оператор min. В контексте работы с фильтрами это означает, что оператор min применяется к каждому элементу выходной последовательности. А это в свою очередь означает что будет вычислен минимум двух величин.

$$D_t^{ffmpeg} = \min(D_t, |D_t - D_{t-1}|)$$

FFmpeg: Примечание (1)

ffmpeg_like не совсем ffmpeg:

формула и результат совпадают с результатом FFmpeg, с точностью до коэффициента *FFMPEG_CORRECTION*.

```
1   # Минимум между разницей и разницей разниц.
2   ffmpeg_like = Filter.union(d_filter, d_diff_filter) | min
3
4   # Коррекция возникает из-за того что в ffmpeg
5   # цвета представляют целыми числами, без нормировки.
6   COLOUR_CORRECTION = 3 * 256.0
7
8   # Нормировка взята из исходных кодов ffmpeg.
9   FFMPEG_NORM = 100.0
10
11  FFMPEG_CORRECTION = COLOUR_CORRECTION / FFMPEG_NORM
12
13  # Как в настоящем FFmpeg.
14  true_ffmpeg = ffmpeg_like | orig * FFMPEG_CORRECTION
15
```

FFmpeg: Примечание (2)

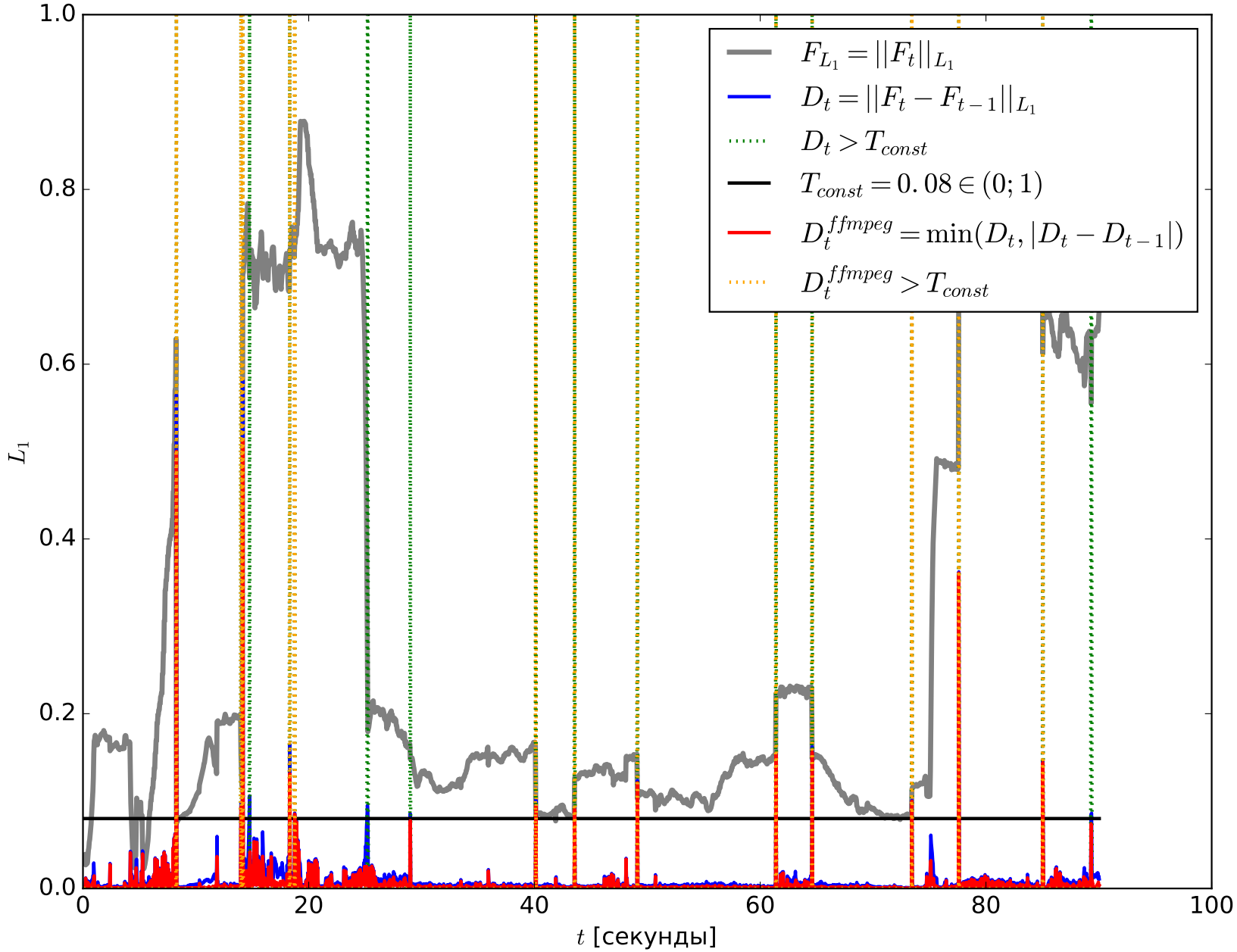
Точки линейных склеек «по FFmpeg» можно получить с помощью консольной команды:

```
ffmpeg -i 'file.mp4' -filter:v "yadif=1:-1:0,select='gt(scene,0.4)',
    showinfo" -f 'null' -y 'qq' 2> >( grep 'pts_time' | sed -uE 's/.*n:\s
    *?([0-9]+).*pts:\s*?([0-9]+).*pts_time:\s*?([0-9\.]+).*pos:\s*?([0-9]
    +).*.*type:\s*?([IPB?]+).*.*mean:\s*?\[(.+)\].*.*stdev:\s*?\[(.+)\].*
    /n:\1\tpts_time:\3\t\tframe_type:\5\tmean:\6␣std:\7/gi' | tee ./out/
    file-null.log 1>&2);
```

Вывод команды:

```
n: 0  pts_time: 8.279  frame_type:P  mean: 20  std:47.0
n: 1  pts_time:13.999  frame_type:B  mean: 40  std:71.3
n: 2  pts_time:14.159  frame_type:P  mean:180  std:75.7
n: 3  pts_time:14.719  frame_type:B  mean:183  std:69.7
n: 4  pts_time:14.759  frame_type:B  mean:179  std:70.6
n: 5  pts_time:18.319  frame_type:P  mean:187  std:73.2
n: 6  pts_time:28.999  frame_type:P  mean: 38  std:53.6
n: 7  pts_time:49.119  frame_type:P  mean: 27  std:46.0
n: 8  pts_time:61.399  frame_type:P  mean: 57  std:71.3
n: 9  pts_time:64.599  frame_type:P  mean: 38  std:58.4
```

Сравнение порогов «Дядя Стёпа — милиционер»



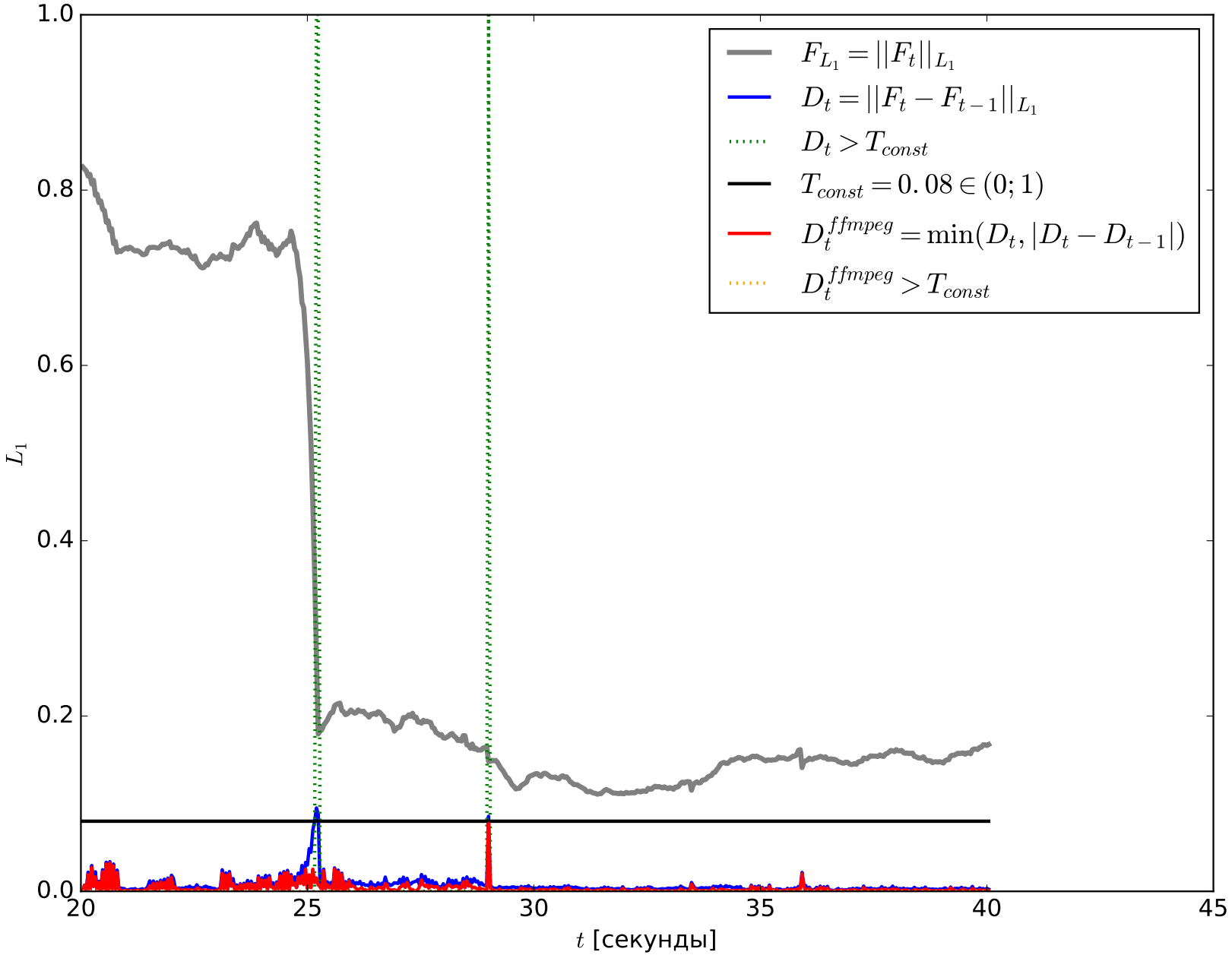
Сравнивать изложенные методы удобно, если их вывести на одном графике. В большинстве случаев для данного временного ряда наивный пороговый метод и порог по FFmpeg совпадают.

Исключение составляет участок с двадцатой по тридцатую секунды. В этом месте, наивный метод выявил значимую разницу. Порог по FFmpeg эту разницу не обнаружил:

- в первом случае, скорость изменения разницы оказалась недостаточной: при приближении на графике отчетливо видно пологий спуск;
- во втором — не хватило величины порога.

С формальной точки зрения, алгоритм предложенный в FFmpeg, лучше наивного метода. Условие минимума разницы и скорости изменения разница отсекает ложные срабатывания. Но в данном случае, много зависит от того что считать «разладкой».

Сравнение порогов, секунды с 20 по 40 «Дядя Стёпа — милиционер»



Чем плох статический порог

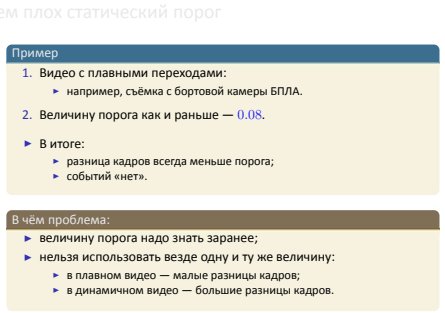
Пример

1. Видео с плавными переходами:
   - например, съёмка с бортовой камеры БПЛА.
2. Величину порога как и раньше — 0.08.

- В итоге:
  - разница кадров всегда меньше порога;
  - событий «нет».

В чём проблема:

- величину порога надо знать заранее;
- нельзя использовать везде одну и ту же величину:
  - в плавном видео — малые разницы кадров;
  - в динамичном видео — большие разницы кадров.

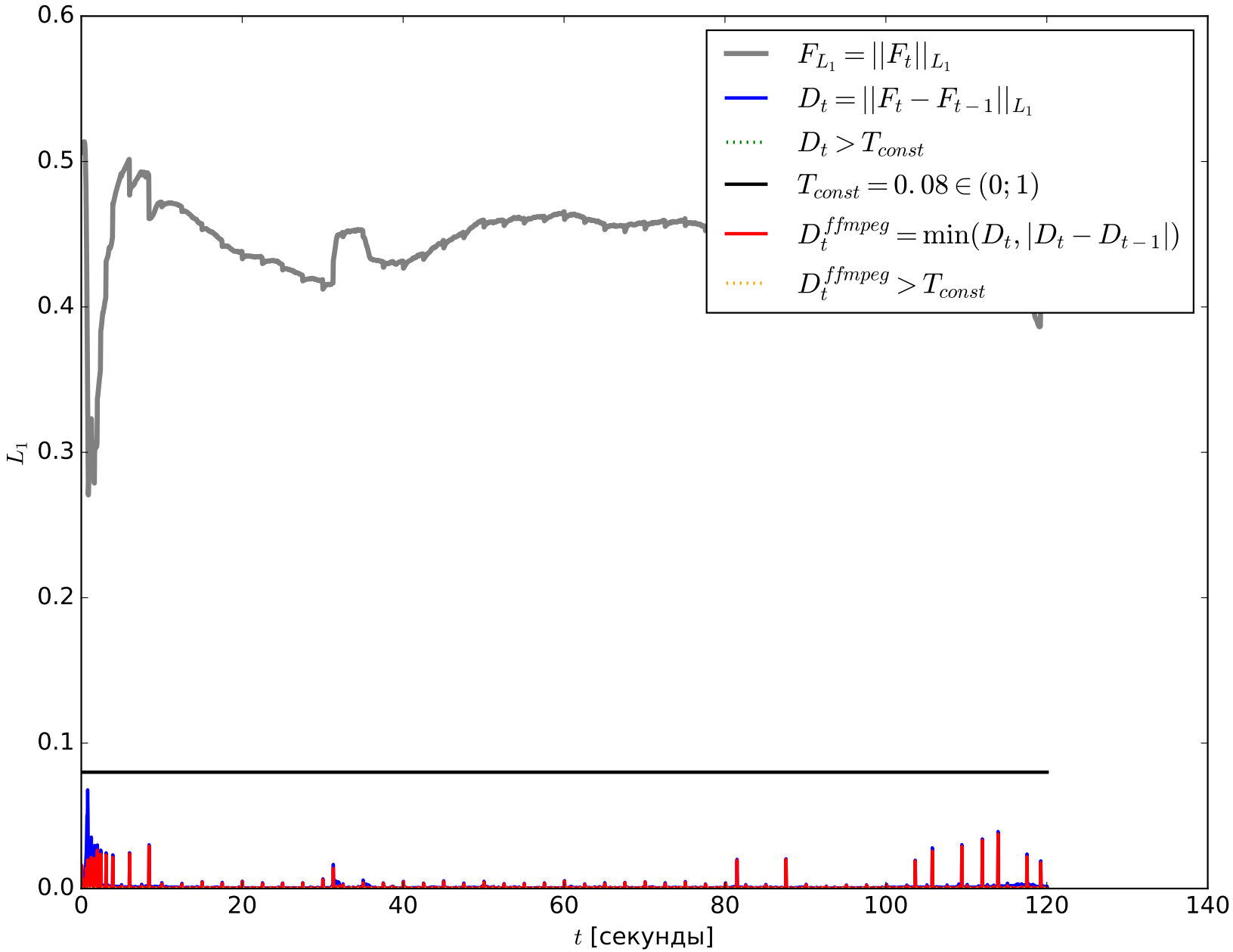


В качестве плавного видео рассмотрим съёмку с БПЛА над побережьем Тулума (Мексика).

Видео снимается с камеры, которая установлена на радиоуправляемый вертолет (квадракоптер). БПЛА взлетает с лодки, и медленно поднимается над морем. На видео видны сначала море и пустыня с пирамидами майя, потом только пустыня, пирамиды и край неба. В процессе подъема камера несколько раз поворачивается из стороны в сторону.

Эти самые повороты камеры, а так тот факт, что в какой-то момент с кадра пропало море, могут быть классифицированы как разладка. При пороге 0.08 ни наивный пороговый метод, ни метод из FFmpeg не зафиксировали никаких событий в этом видео.

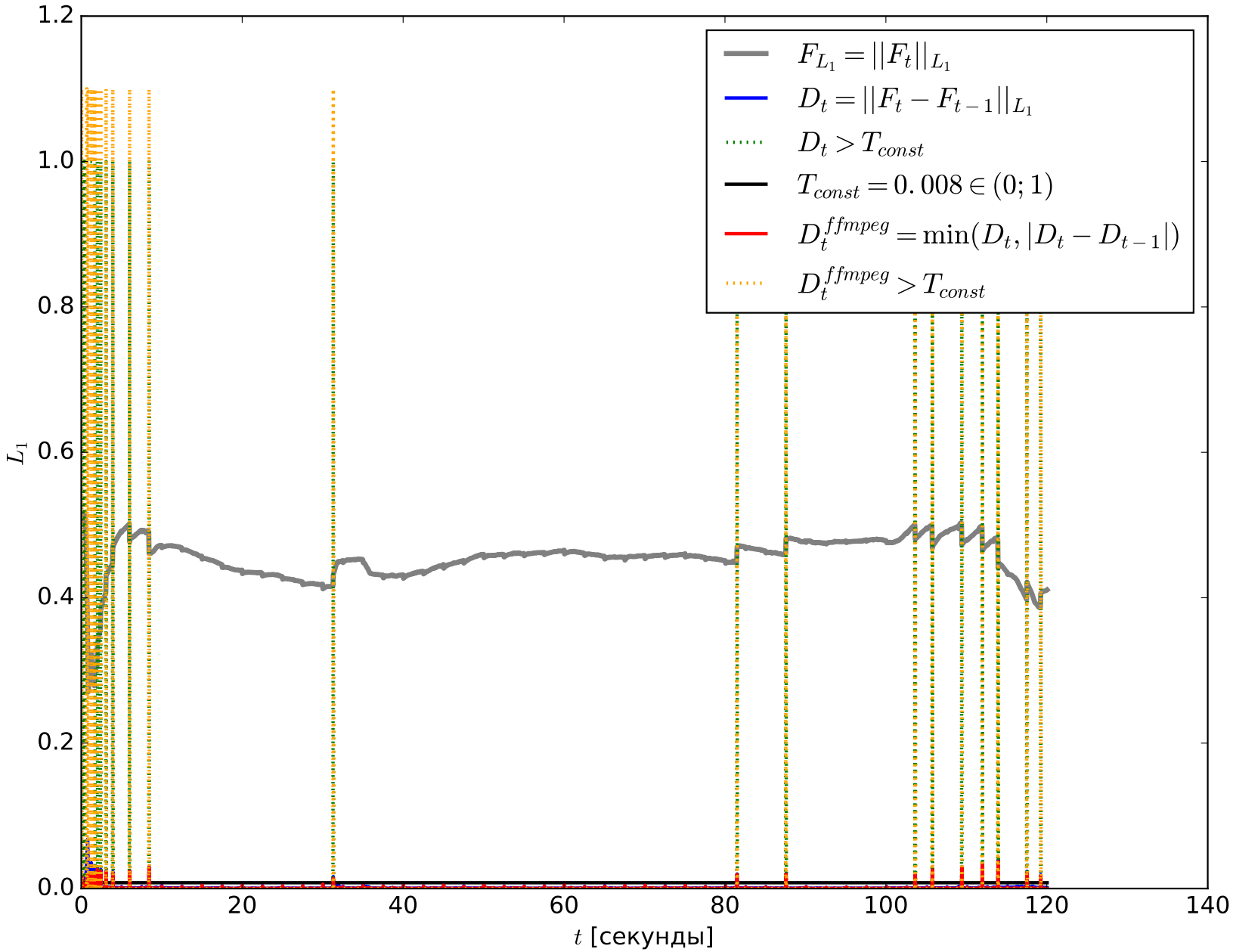
Съёмка с БПЛА над побережьем Тулума ($T_{const} = 0.08$)



На графике средней яркости кадра, и на графиках разностей отчетливо видны зазубрины. Это автоматическая калибровка баланса белого в камере. Калибровки происходят через заранее известные промежутки времени, и считать их событиями нельзя.

Если уменьшить величину порога до 0.008, то оба метода будут детектировать события. Но при этом, некоторые из них останутся без внимания, например, спад перед сороковой секундой. Однако, при дальнейшем понижении порога в качестве событий будут определять паразитные шумы калибровки.

Съёмка с БПЛА над побережьем Тулума ($T_{const} = 0.008$)



Улучшения статического порога

Проблема:

- большой порог для плавного видео:
  - события найдены не будут;
- маленький порог для динамичного видео:
  - много ложных срабатываний;
- если видео имеет плавные и динамичные участки:
  - порог подобрать невозможно.

Решение:

- Учитывать «динамичность» видео:
  1. Масштабировать разницы кадров:
     → нормировать по окрестности.
  2. Учитывать среднюю величину и дисперсию.

Нормировка по размаху

Нормировка разницы кадров

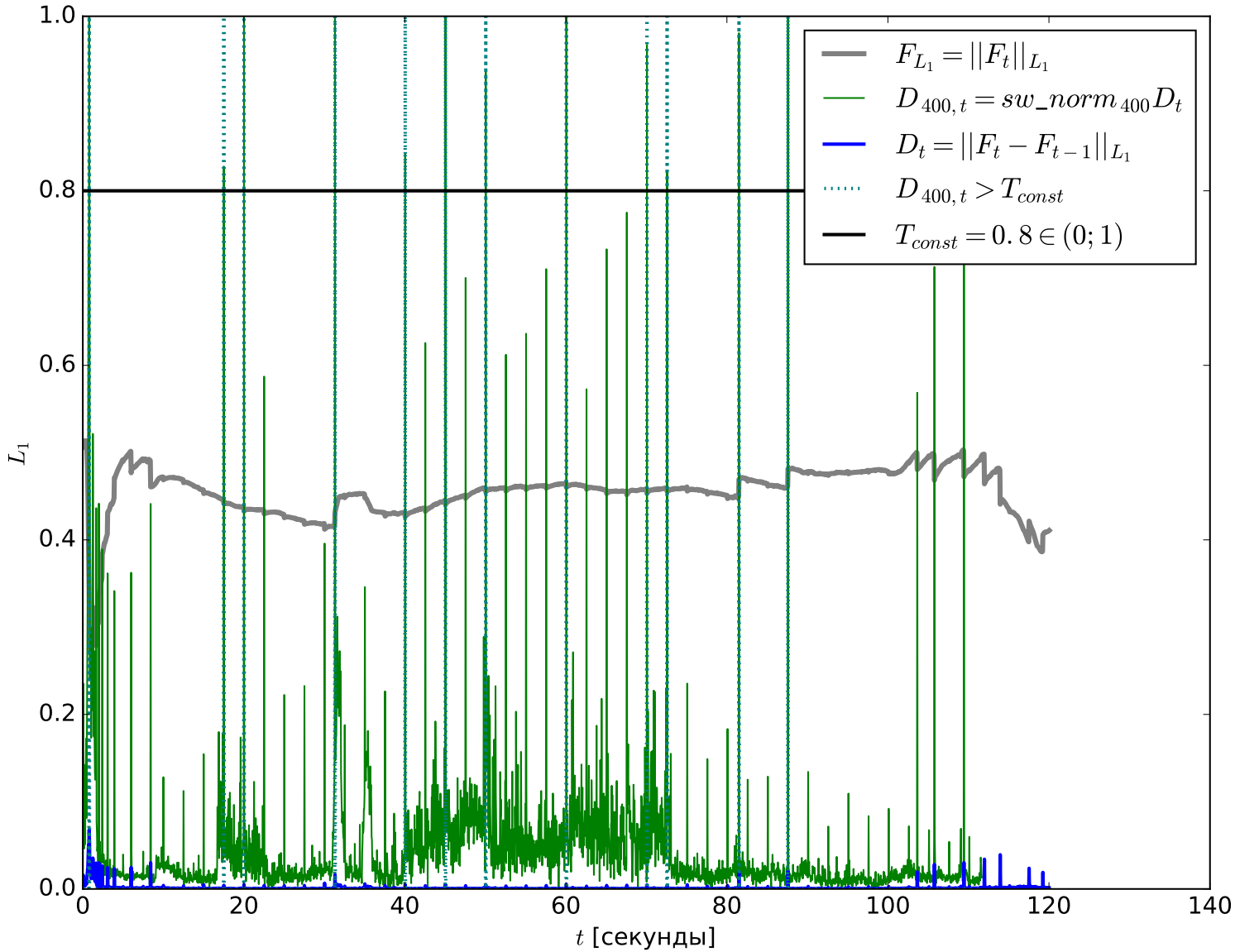
1. Выберем размер скользящего окна (вектора задержек) — s.
2. Для каждого кадра из последовательности:
   - 2.1 Вычислим максимальное и минимальное значение разницы кадров на этом окне.
   - 2.2 Вычислим размах разниц для данного скользящего окна.
   - 2.3 От текущего значения разницы соседних кадров отнимем минимальное значение разницы на скользящем окне, и поделим на размах
3. Таким образом мы получим нормированное по размаху значение разницы кадров для данного скользящего окна.

$$D_{s,t} = \frac{D_t - \min\left(D_{t-s} \ldots D_t\right)}{\max\left(D_{t-s} \ldots D_t\right) - \min\left(D_{t-s} \ldots D_t\right)}$$

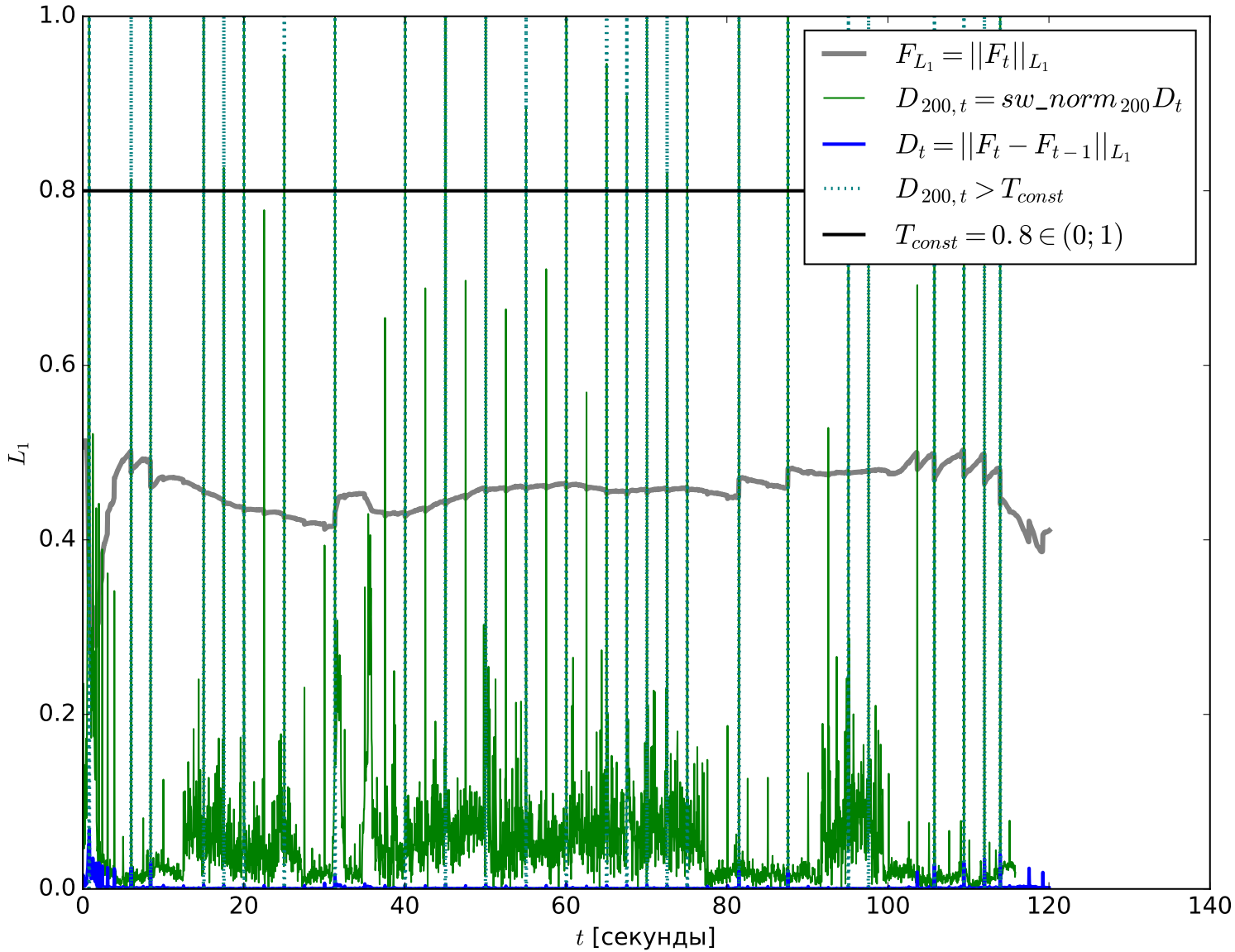
Нормировка по размаху: описание на мета-языке

```
delay = DelayFilter()        # Фильтр линейной задержки.
orig  = delay(0)             # Входящий сигнал без изменения.
shift = ShiftSWFilter()      # Сдвиг сигнала на один кадр.
diff  = orig - shift         # Разница соседних кадров.
norm  = NormFilter()         # Норма сигнала.

# Возвращает скользящее окно размером 400 для каждого кадра
sw = BaseSWFilter(s=400, min_size=2)

sw_max = sw | max  # Максимум по скользящему окну.
sw_min = sw | min  # Минимум по скользящему окну.

sw_norm = (orig - sw_min) / (sw_max - sw_min) # Масштабирование.

# Масштабированная разница по скользящему окну
d_sw_filter = diff | abs | norm(l=1) | sw_norm

# Результирующий фильтр: точки разладок.
result_filter = d_sw_filter | threshold
```

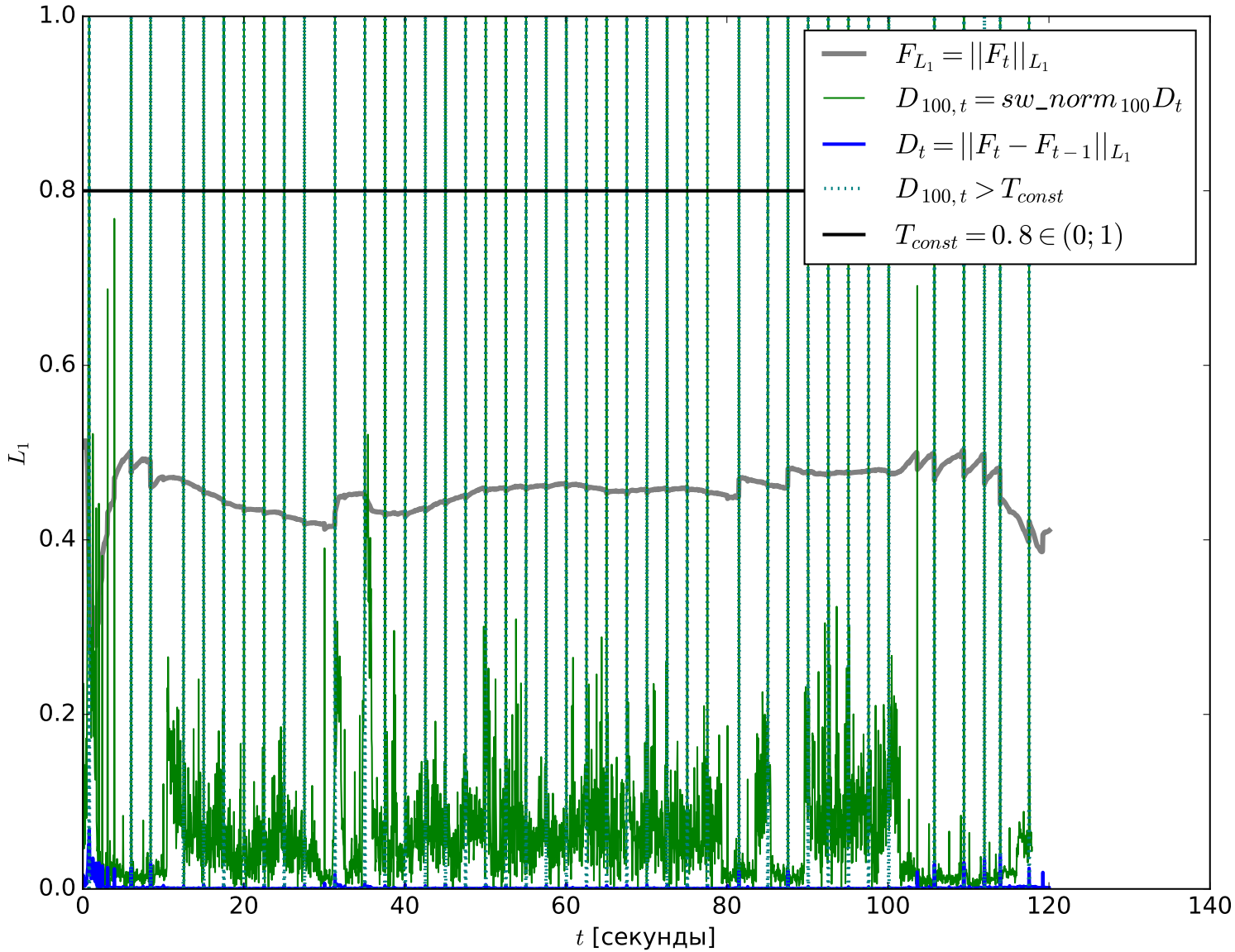
нормировка по размаху (s = 400)



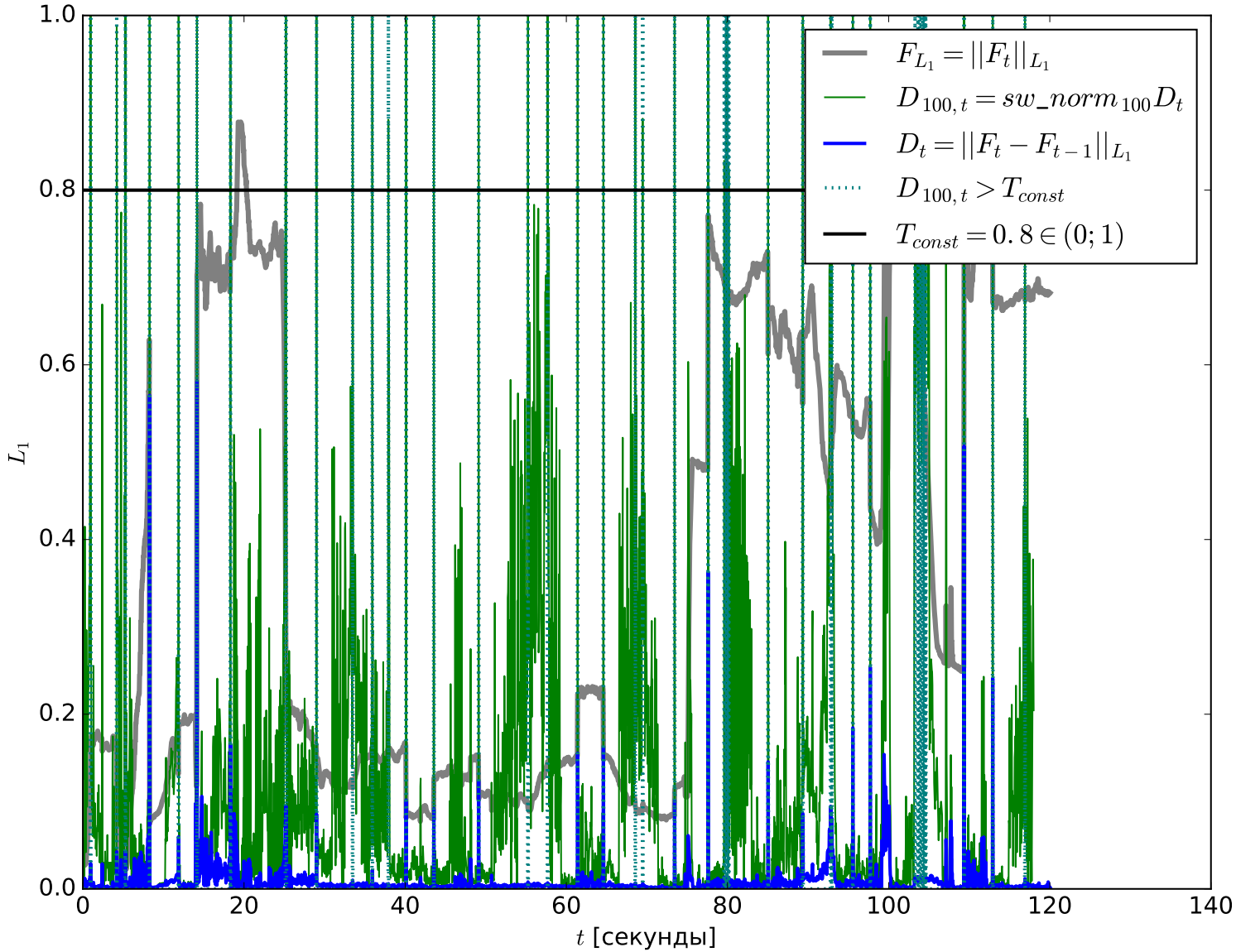
Нормировка по размаху (s = 200)



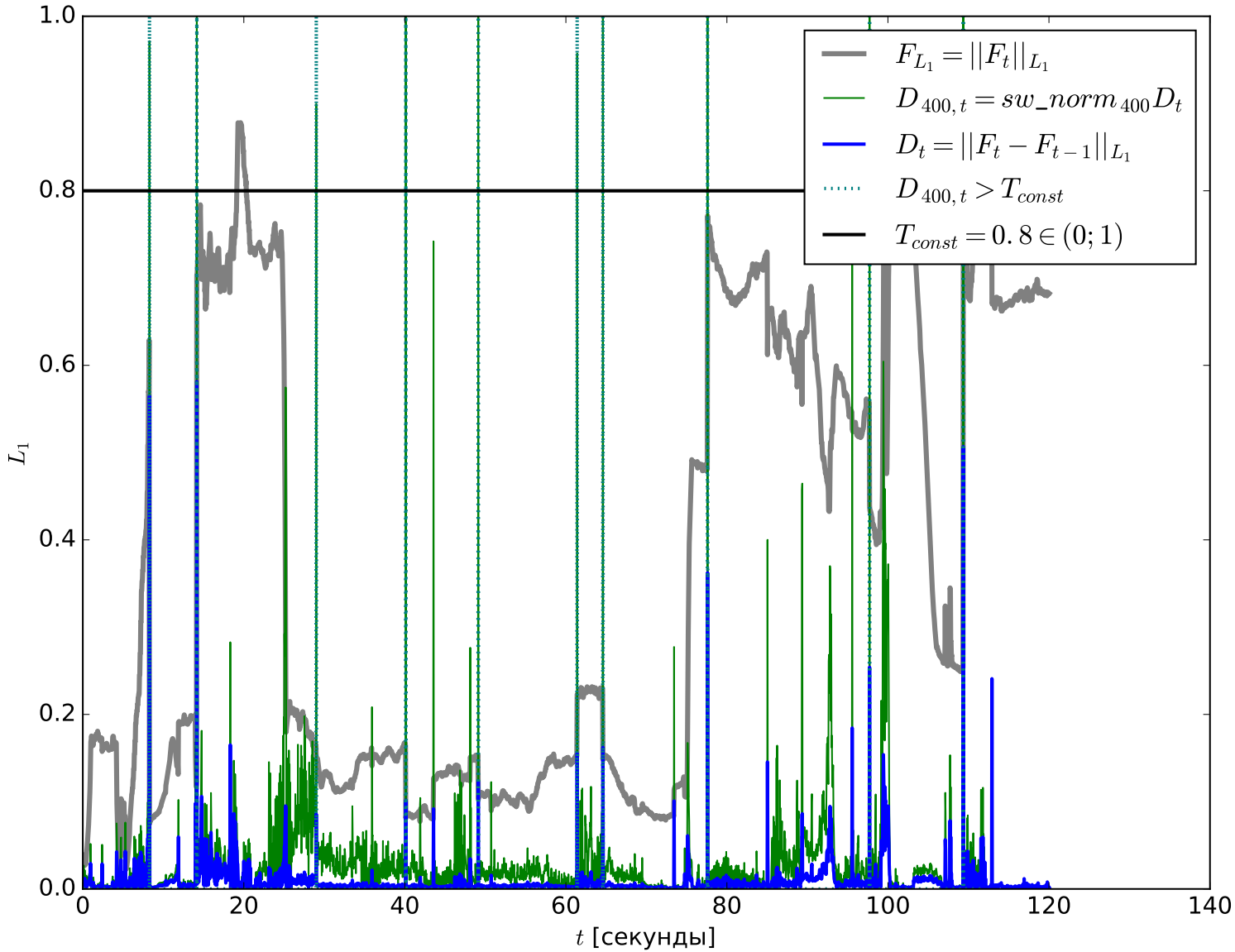
Нормировка по размаху (s = 100)



Нормировка по размаху (s = 100), «Дядя Стёпа — милиционер»



Нормировка по размаху (s = 400), «Дядя Стёпа — милиционер»



Свойства нормировки по размаху

Достоинства:

- относительная величина порога:
  - не нужно подбирать для каждого случая;
  - можно использовать везде одну и ту же величину.

Недостатки:

- нужно подбирать размер окна:
  - при малом размере окна — нормируем для паразитных шумов
  - при большом размере — пропускаем разладки.

Метод голосований

Достоинства:

- относительная величина порога:
  - не нужно подбирать для каждого случая;
  - можно использовать везде одну и ту же величину.

Недостатки:

- нужно подбирать размер окна:
  - при малом размере окна — нормируем для паразитных шумов
  - при большом размере — пропускаем разладки.

Пороговые методы адаптивным порогом

Как устроены

- разница соседних величин (по некоторой норме);
- превышения порога — «аномалия».
- порог вычисляется динамически (критерий Смирного-Граббса).

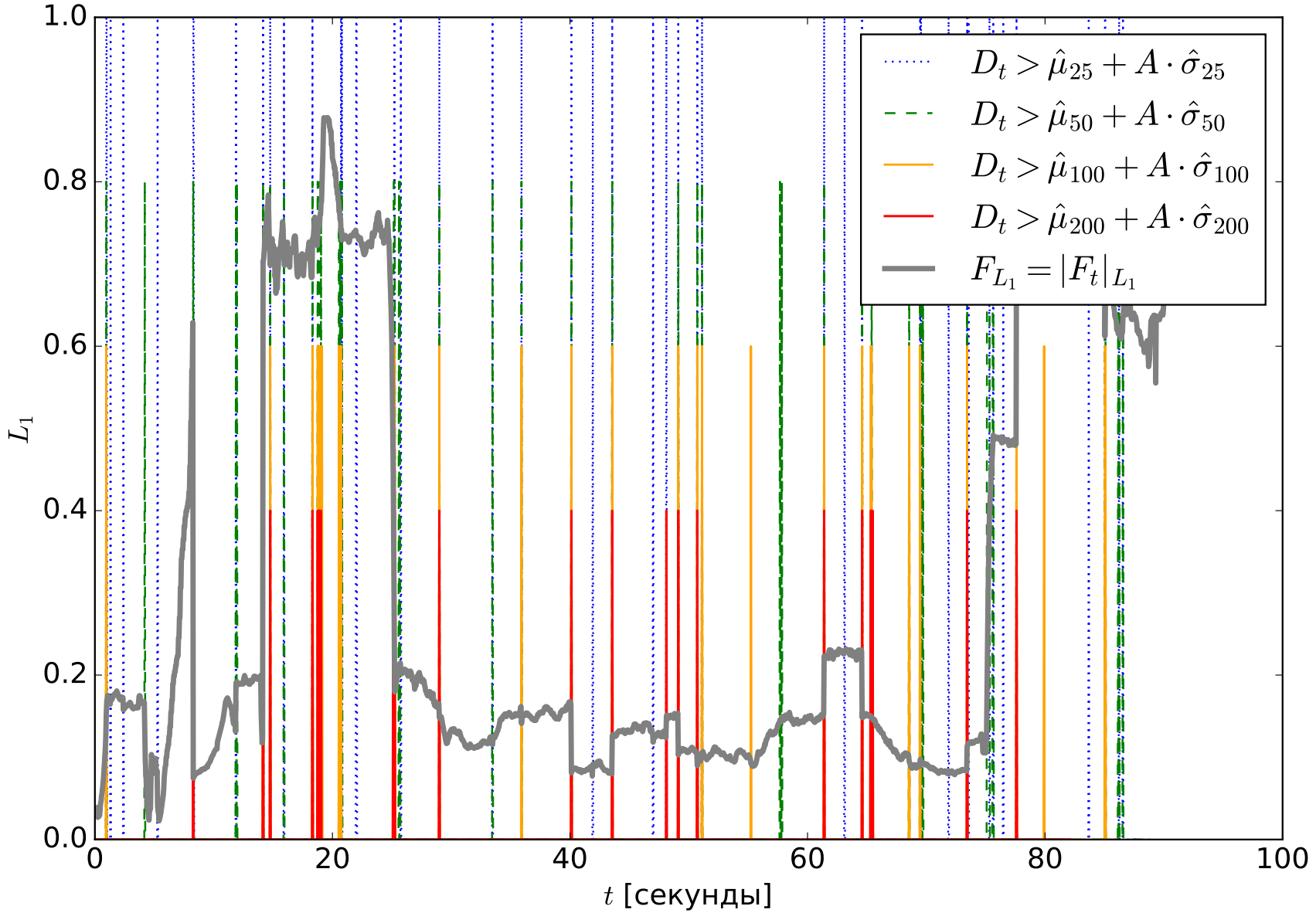
Плюсы

- не требовательны к ресурсам.

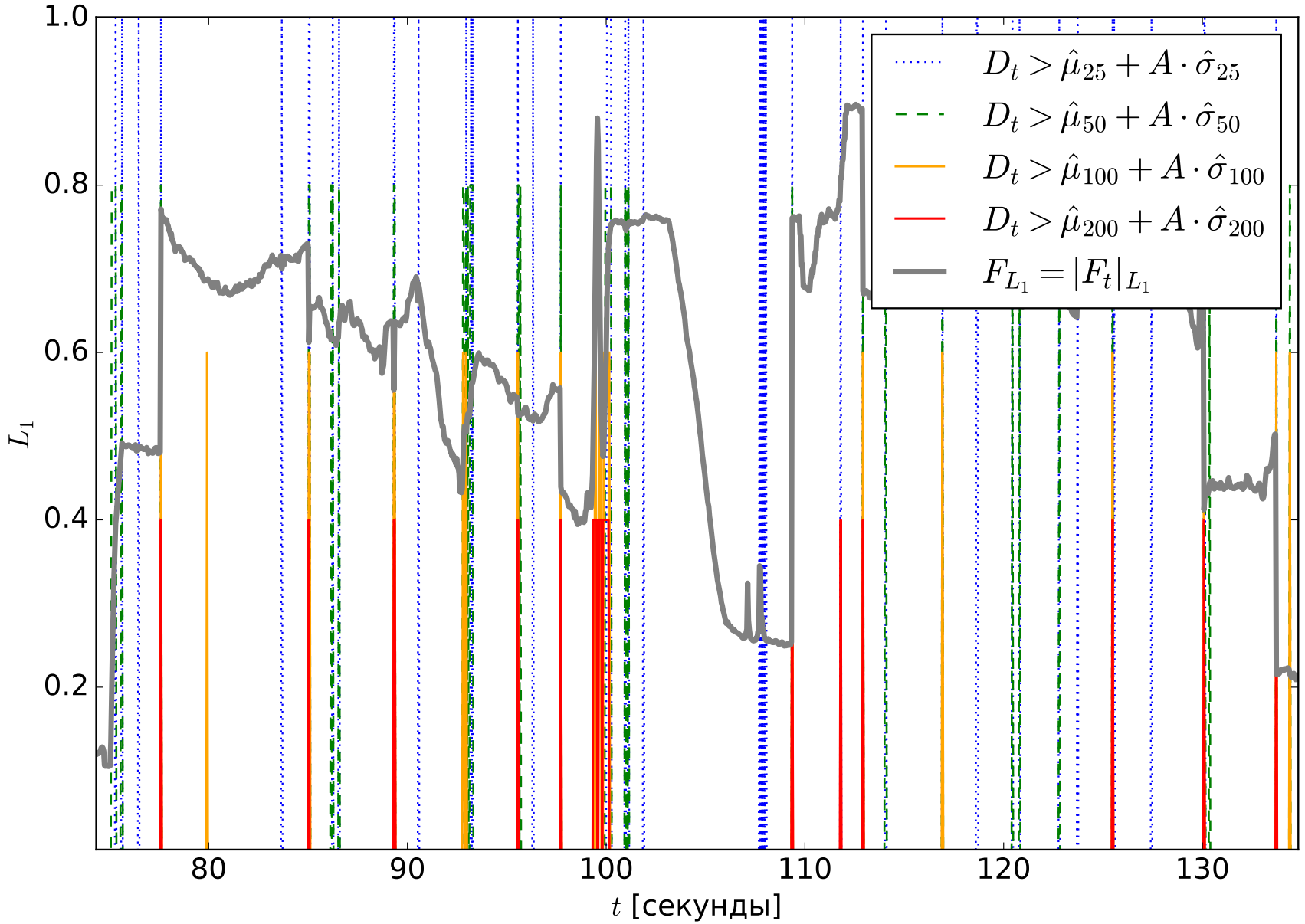
Минусы

- требуется заранее подобрать размер скользящего окна;
- не применимо для разных типов видео;
- чувствительны к случайным всплескам;
- ловят только краткосрочные события;

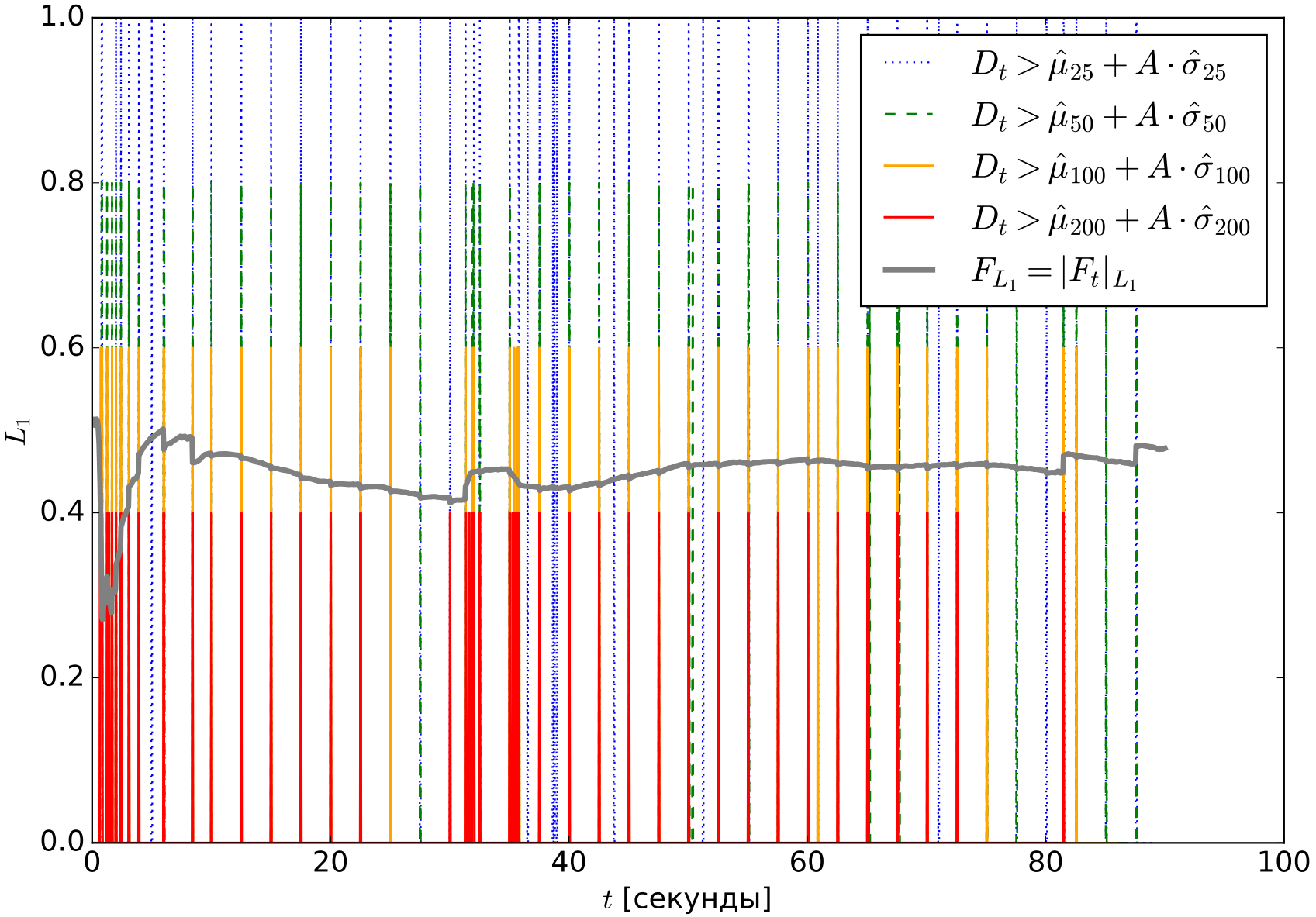
«Дядя Стёпа — милиционер», адаптивный порог $D_t > \hat{\mu}_k + A \cdot \hat{\sigma}_k$



«Дядя Стёпа — милиционер», адаптивный порог $D_t > \hat{\mu}_k + A \cdot \hat{\sigma}_k$ (не нашли плавный переход)



Съёмка с БПЛА над побережьем Тулума, адаптивный порог $D_t > \hat{\mu}_k + A \cdot \hat{\sigma}_k$



Сравнение средних

Как устроены

- разница двух оценок сигнала;
- сигнал оцениваем через средние.

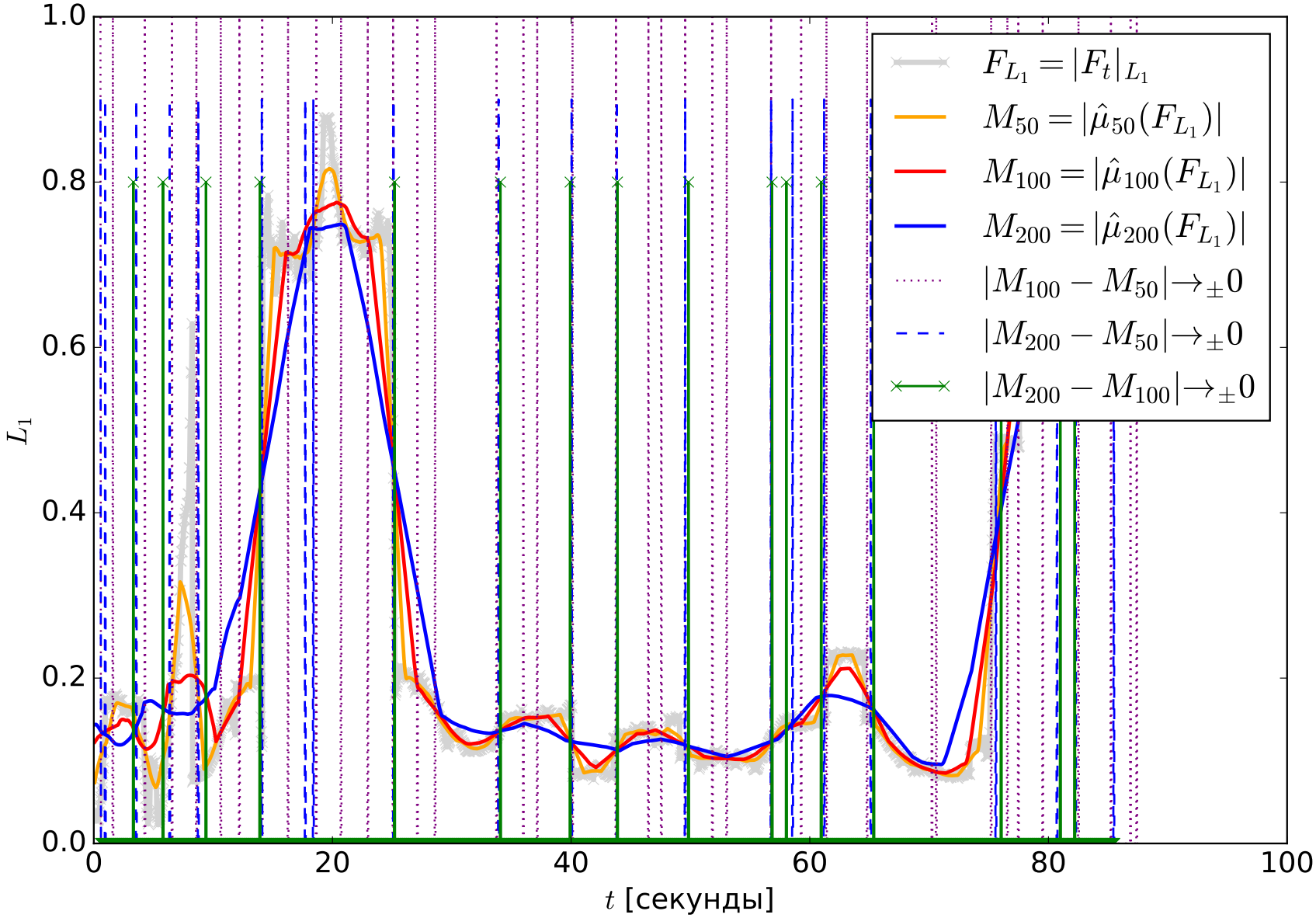
Плюсы

- не очень требовательны к ресурсам.
- могут ловить и резкие и плавные переходы

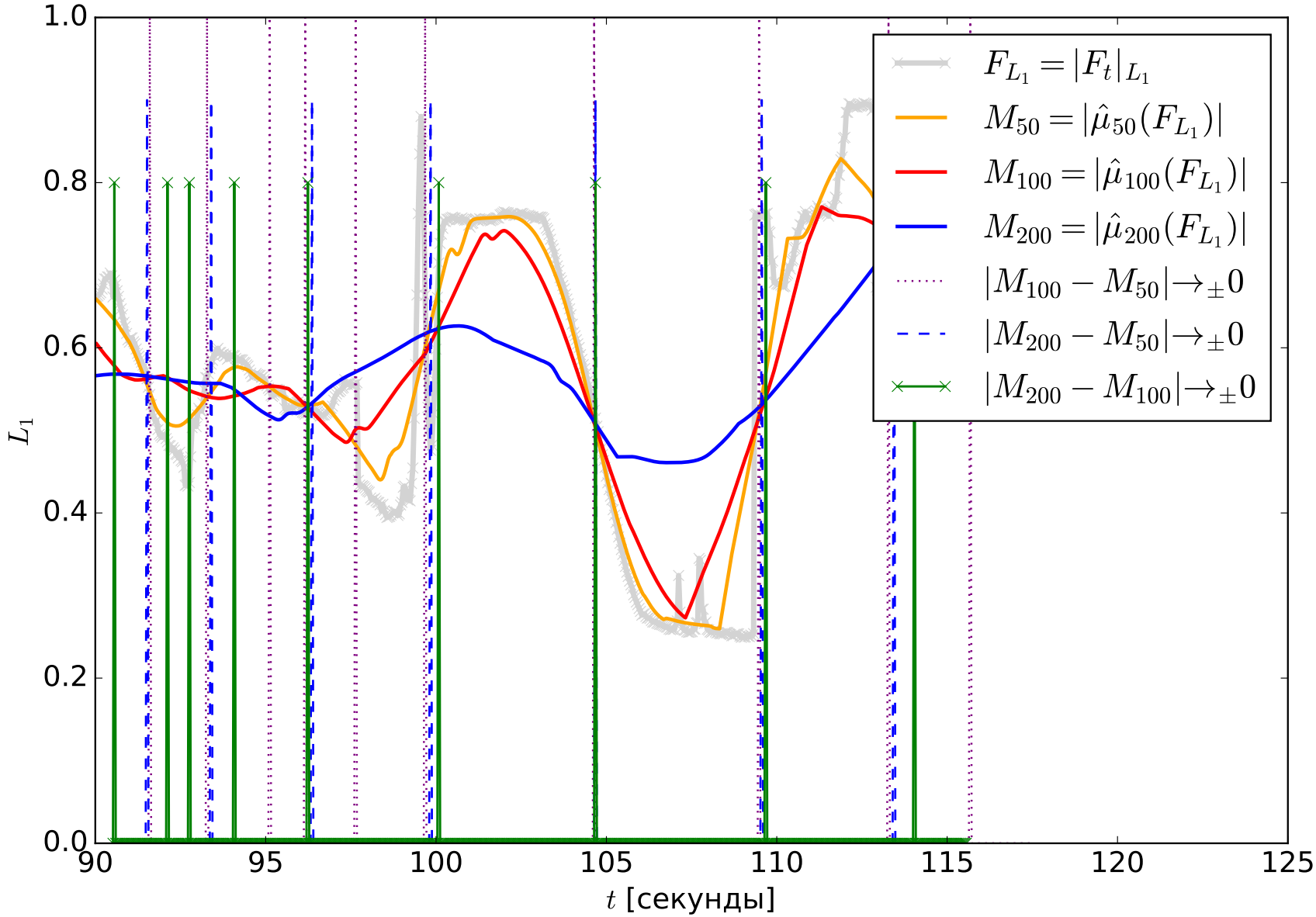
Минусы

- требуется заранее подобрать размеры скользящих окон;
  - при малом размере ловят шум;
  - при большом — могут пропустить событие;
- требуется хранить скользящее окно.

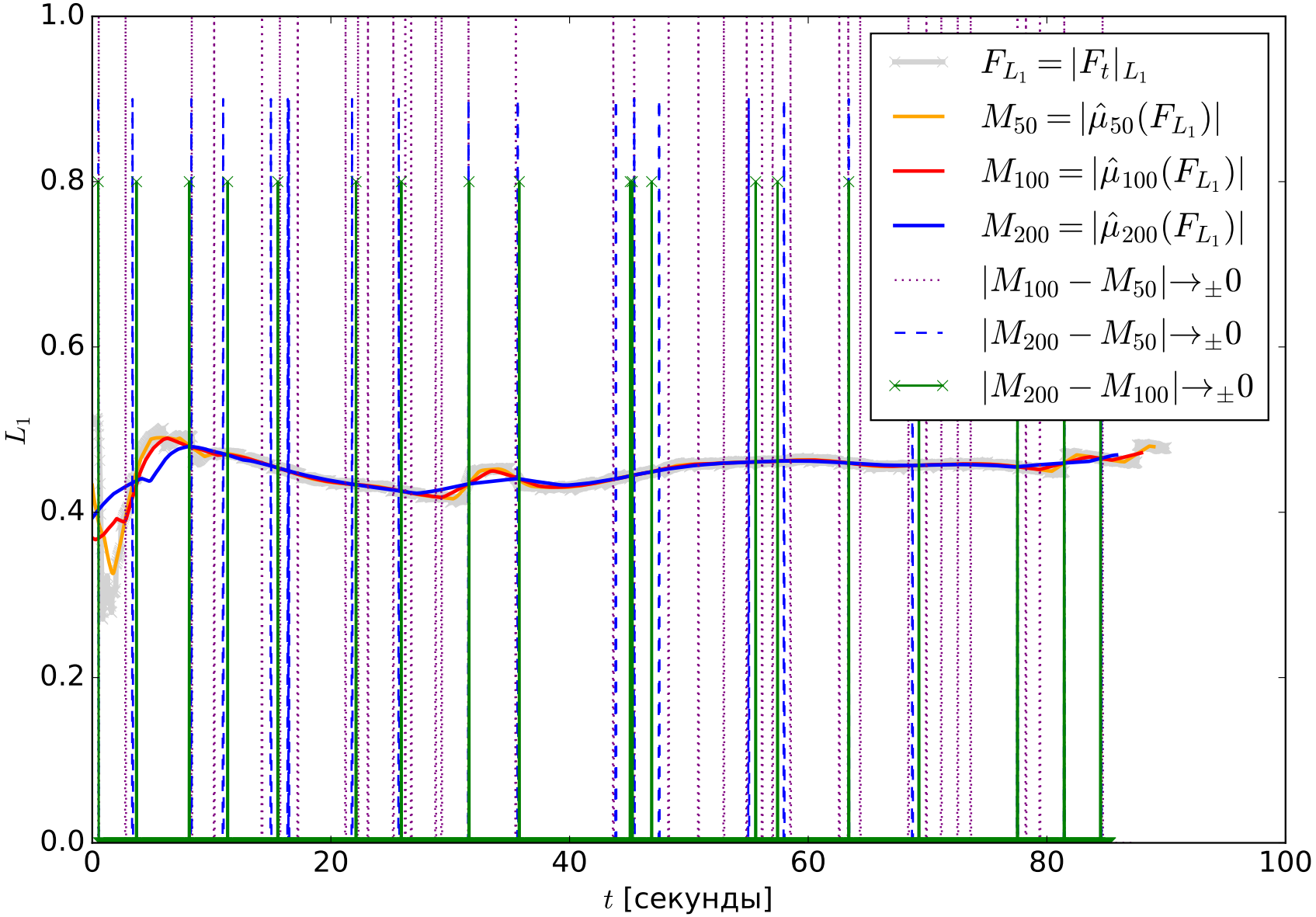
«Дядя Стёпа — милиционер», разницы средних



«Дядя Стёпа — милиционер», разницы средних (нашли центральную точку перехода)



Съёмка с БПЛА над побережьем Тулума, разницы средних



Что предлагаем

Штрафы по производной средней оценки

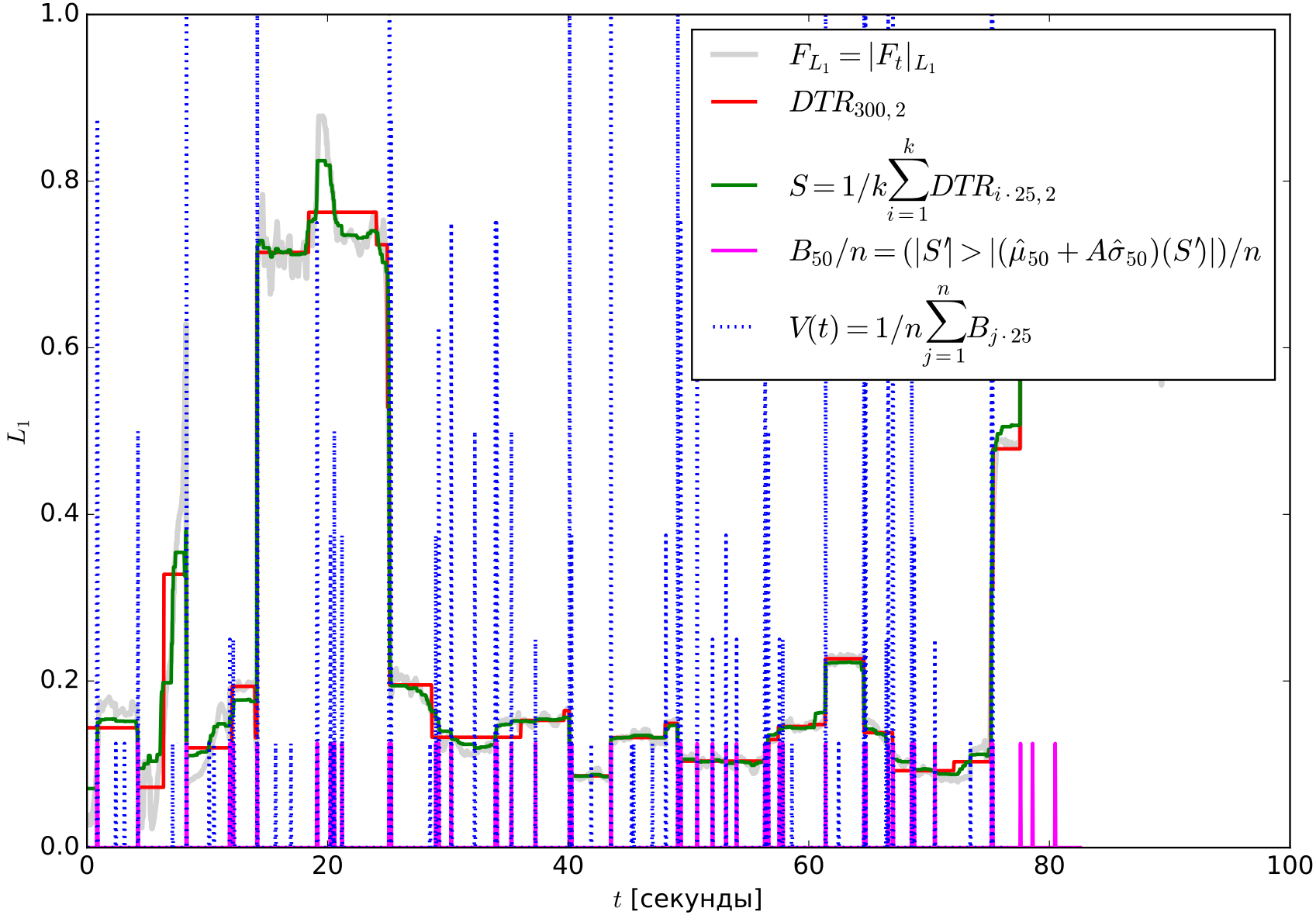
- оценим сигнал кусочно-постоянной функцией $DTR_{w,d}(t)$:
  - используем решающее дерево регрессии глубины d;
  - по скользящему окну размера w;
- размножение оценок:
  - усредненная оценка по k регрессиям $S_d(t) = \frac{1}{k}\sum_{i=1}^{k+1} DTR_{i\cdot y,d}(t)$;
  - окна разного размера $w = i \cdot g$, g — шаг размера окна;
- разница соседних точек:
  - $S'(t) = \frac{dS(t)}{dt} = S(t) - S(t-1)$;
- штраф за отклонение:
  - $B_w(t) = (|S'(t)| > |\hat{\mu}_w(S'(t)) + A \cdot \hat{\sigma}_w(S'(t))|) \in \{0, 1\}$;
- еще раз размножение оценок (голосование):
  - $V(t) = \frac{1}{n}\sum_{j=1}^{n+1} B_{j\cdot z}(h)$; h — шаг размера окна;

DSL

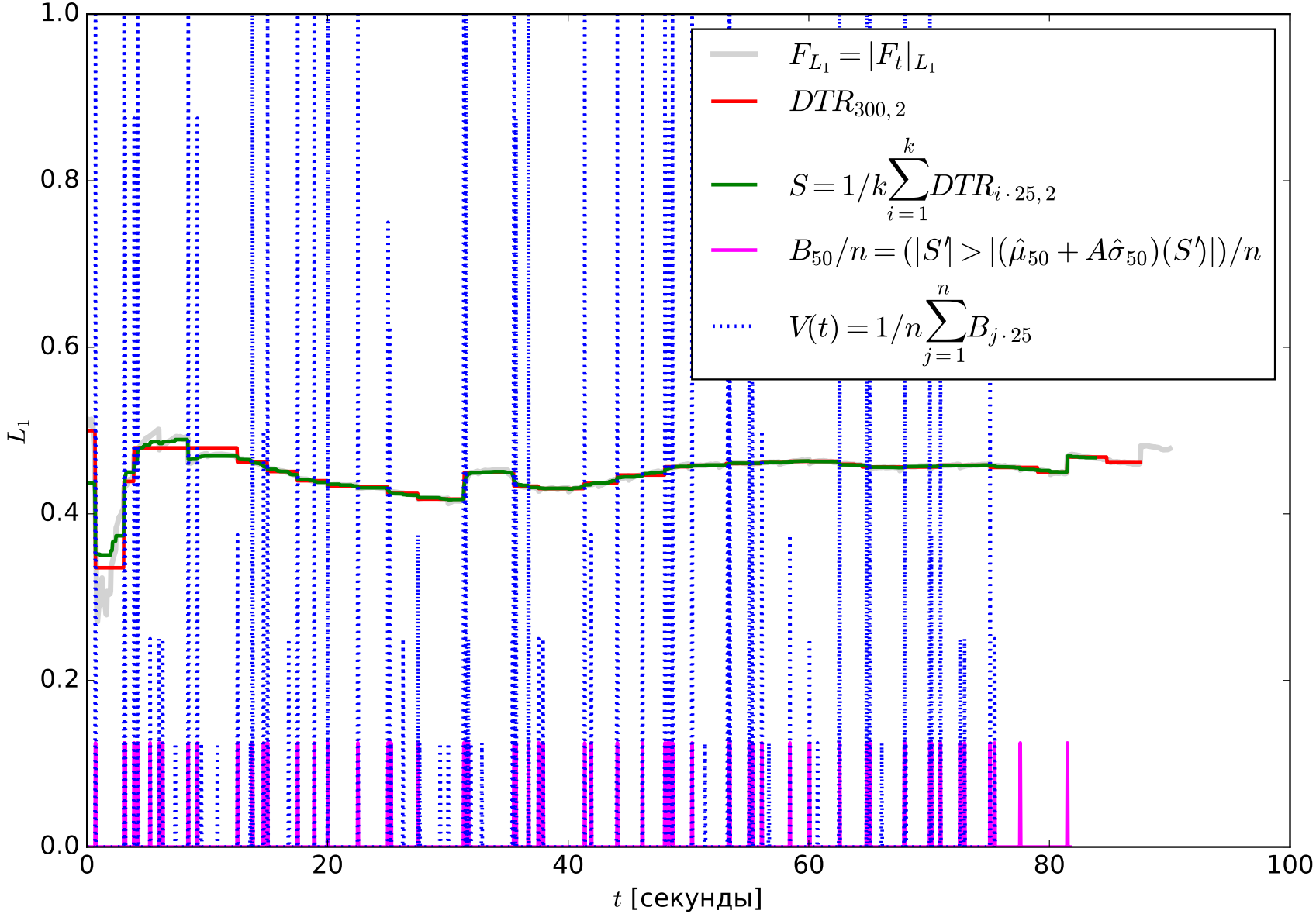
Для решения задачи мы разработали мета-язык на базе языка Python.

```
1  delay = DelayFilter()        # Фильтр линейной задержки.
2  original = delay(0)          # Входящий сигнал без изменения.
3  shift = ShiftSWFilter()      # Сдвиг сигнала на один кадр.
4  diff = original - shift      # Разница соседних кадров.
5  norm = NormFilter()          # Норма сигнала.
6  modulus = ModulusFilter()    # Модуль сигнала.
7  mean = MeanSWFilter()        # Среднее по скользящему окну.
8  std = StdSWFilter()          # Стандартное отклонение по окну.
9  dtr = DecisionTreeRegressorSWFilter()
10
11 n=8, k=8, h=25, g=25, d=2, a=3.0 # параметры алгоритма
12
13 S = sum(dtr(w=s*i+1,d=d) for i in range(1,k + 1)) / k
14 B = lambda w: delay(0) > (mean(w=w) + a * std(w=w))
15 V = sum(B(w=z*j) for j in range(1,n + 1)) / n
16
17 # Фильтр: собирается отложено до непосредсвенного применения.
18 # Оператор «|» означает «конвейер».
19 vfilter = norm(l=1) | S | diff | modulus | V
```

«Дядя Стёпа — милиционер», штрафы по дискретной производной усредненной кусочно-постоянной аппроксимации



Съёмка с БПЛА над побережьем Тулума, штрафы по дискретной производной усредненной кусочно-постоянной аппроксимации



Что предлагаем

Плюсы

- интуитивный поиск аномалий;
- ловит и резкие и плавные переходы;
- относительный порог аномалии можно выбрать после вычислений (в пределах).

Минусы

- требования вычислительным к ресурсам;
- пока плохо изучен, нужно исследовать свойства.

Итог

Чего достигли

- предложен и реализован алгоритм поиска аномалий видео;
- по аномалия — определяем, что в видео произошло событие
- проведены эксперименты;
- результаты совпадают с тем, как разметил человек.

Куда двигаться дальше

- получить размеченную выборку видео-событий;
- сравнить различные методы по этой выборке;
- получить коэффициенты доверия для конкретных методов;
- получить комбинированный метод.