Видеокодек с межкадровым декодированием и быстрым покадровым кодированием

Беляев Евгений Александрович
eabelyaev@itmo.ru

- Кодер осуществляет выбор параметров кодирования, а декодер лишь их использует.
- Вычислительная сложность кодера значительно выше сложности декодера.
- Сложность кодера: $C(H.264) < C(H.265) < C(H.266)$.

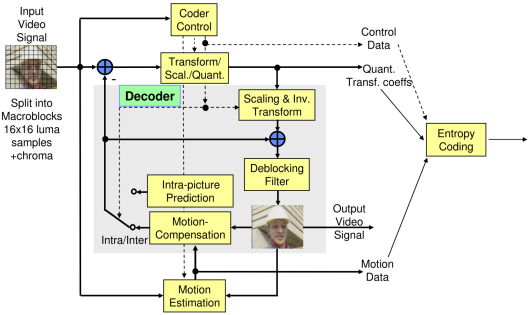


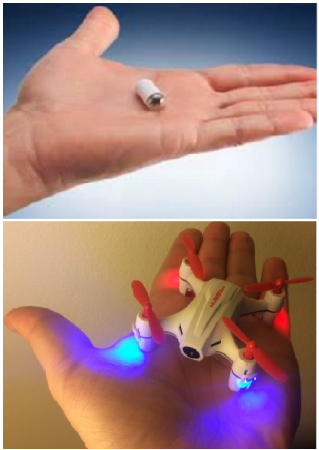
Рис.: Кодер H.264/AVC

- **Что делать, если сложность кодера является критичной?**

- Допустим, что кадр 21 не был доставлен декодеру (потерян).
- Вместо кадра 21 декодер покажет кадр 20.
- Поскольку кодер и декодер ссылаются на разные кадры, возникает распространение ошибки от кадра к кадру.

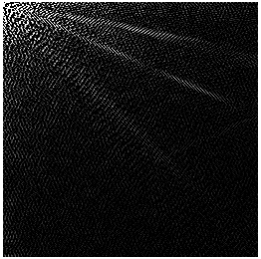
- **Что делать, если требуется передача по ненадёжному каналу связи, и перепосылки невозможны?**

2. Где это может использоваться?

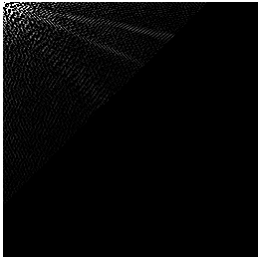


- В микроуйстройствах сжатия и передачи видео (беспроводная эндокапсула и т.д.).
- В качестве резервного кодека, если длительность работы устройства от батареи приоритетнее.
- Если канал передачи очень ненадёжный.

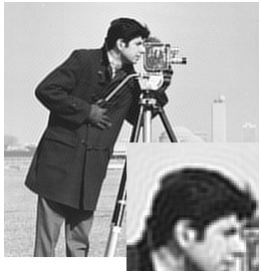
(a) X



(b) $F = dct2(X)$



(c) $\hat{F}$

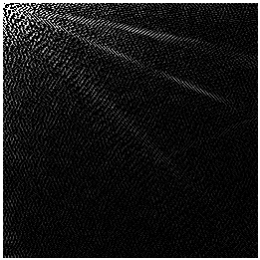


(d) $\hat{X} = idct2(\hat{F})$

- Вычисляем 2-D DCT для изображения X, размером 256×256.
- Берем первые 21000 (33%) коэффициентов по зигзагу (как в JPEG) и вычисляем обратное 2-D DCT.
- Можно ли взять 21000 коэффициентов так, чтобы качество было лучше?[1]

[1] Romberg, J. Imaging via Compressive Sampling, IEEE Signal Processing Magazine, 2008.

(a) X



(b) $F = dct2(X)$

(c) $\hat{F}_0$

(d) $\hat{X}_0 = idct2(\hat{F}_0)$

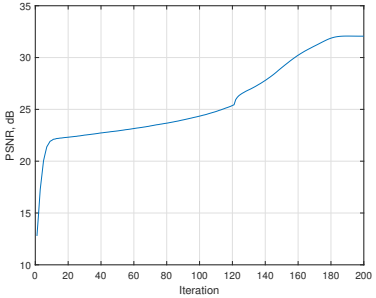
- Вычисляем 2-D DCT для изображения X.
- Берем первые 1000 коэффициентов по зигзагу, остальные 20000 на случайных позициях.

1. Выполняем пороговую функцию для $\hat{X}_0$ (используем фильтр BM3D[2]).
2. Для полученного изображения X_f вычисляем спектр F и в него 'вставляем' ненулевые коэффициенты из изначального спектра $\hat{F}_0$.
3. Вычисляем изображение $\hat{X}_1 = idct2(F)$ и повторяем шаги 1-2.

(a) $\hat{X}_0$ (b) X_f=BM3D($\hat{X}_0$) (c) $\hat{F}_1 = dct2(X_f)$ (d) $\hat{X}_1 = idct2(\hat{F}_1)$

[2]K. Dabov, A. Foi, V. Katkovnik and K. Egiazarian, Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering, *IEEE Transactions on Image Processing*, 2007.

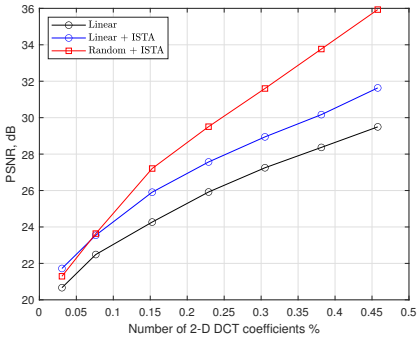
(a) $\hat{X}_0$



(b) PSNR(i)

(c) $\hat{X}_{200}$

Включить деморолик: `ISTA_demo.avi`



- Случайные измерения лучше линейных.
- Для неслучайных измерений тоже можно использовать ISTA:
  - Для superresolution[3].
  - Для мягкого декодирования JPEG и MJPEG[4] и wavelet-кодеков[5].

[3] K. Egiazarian et al., Single Image Super-resolution via BM3D Sparse Coding, EUSIPCO, 2015.

[4] E. Belyaev et al., Motion JPEG Decoding via Iterative Thresholding and Motion-Compensated Deflickering, MMSP, 2020

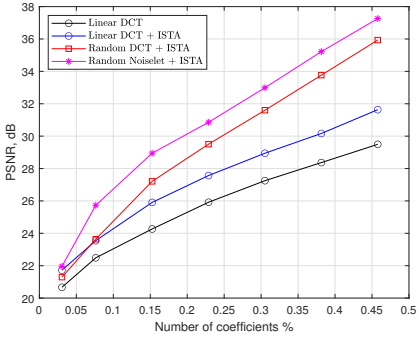
[5] E. Belyaev et al., Error Concealment for 3-D DWT based Video Codec using Iterative Thresholding, IEEE Communications Letters, 2017

Нужно ответить на следующие вопросы[6]:

- Какую модель взятия измерений выбрать?
- Как передавать случайные позиции коэффициентов декодеру?
- Как сжимать измерения?
- Как восстанавливать видео в декодере?

[6] E.Belyaev, An Efficient Compressive Sensed Video Codec with Inter-Frame Decoding and Low-Complexity Intra-Frame Encoding, Sensors, 2023.

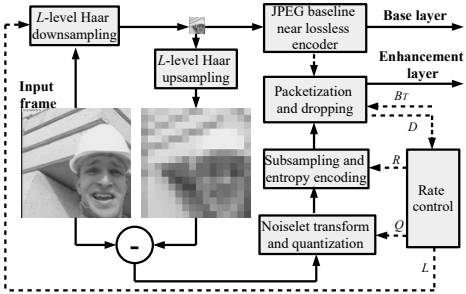
- В общем случае матрица измерений не должна быть хороша для сжатия (DCT или wavelet).
- Можно использовать noiselet.



Noiselet transform

Input: $x_1, x_2, ..., x_N$

$c \leftarrow N^2 - 1$
for $j = 0, ..., \frac{N^2}{2}$ **do**
  $k \leftarrow j \oplus c$
  $y_j \leftarrow x_j + x_k$
  $y_k \leftarrow x_j - x_k$
end for
for $d = \left\lfloor \frac{c}{2} \right\rfloor, \left\lfloor \frac{c}{4} \right\rfloor ..., 1$ **do**
  for $j = 0, ..., \frac{N^2}{2}$ **do**
    $k \leftarrow j \oplus c \oplus d$
    $t \leftarrow y_j$
    $y_j \leftarrow y_j - y_k$
    $y_k \leftarrow t + y_k$
  end for
end for



- Входной кадр уменьшается в L раз и сжимается JPEG на QF=100.
- Уменьшенный кадр увеличивается и вычитается из исходного кадра.
- Выполняется noiselet для разностного кадра.
- Берутся коэффициенты на псевдослучайных позициях.
- Выбранные коэффициенты квантуются (степенью двойки), сжимаются арифметическим кодером и пакетируются.
- Пакеты сохраняются в заголовок JPEG файла.

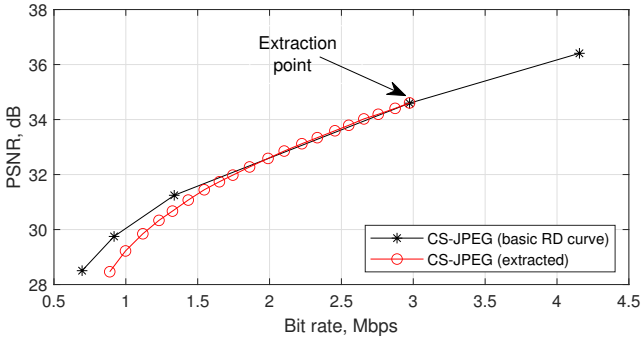


Рис.: Декодирование при удалении части пакетов из битового потока

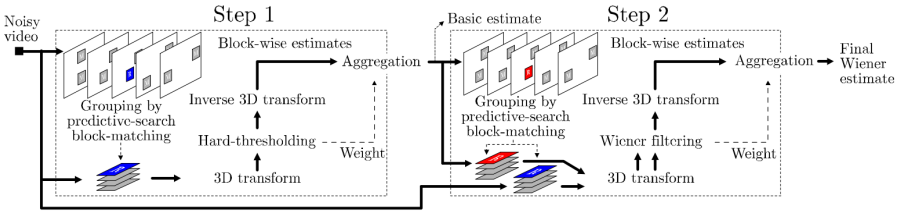
- Так как измерения кодируются независимо, качество восстановления зависит только от числа измерений, которые дошли до декодера.
- Поэтому битовый поток устойчив к потерям пакетов (распространения ошибки не происходит).
- Путём откидывания лишних пакетов можно получить нужный битрейт.

- Используются псеведослучайные векторы, в которых хранится расстояние между коэффициентами.
- Например, для взятия 10% измерений можно использовать вектор

$$\mathbf{v} = \{19, 9, 11, 17, 12, 6, 7, 5, 12, 2\},$$

т.е. передаются коэффциенты $y_1, y_{20}, y_{29}, y_{40}, \ldots$.

- Декодирование осуществляется итеративно для группы из 16 кадров. В качестве пороговой функции можно использовать фильтр VBM3D[7].
  - Step 1. Для блока в текущем кадре выполняется поиск похожих блоков, из найденных блоков формируется 3D блок, для которого выполняется фильтрация в области ДКП.
  - Step 2. В предварительно отфильтрованном видео снова выполняется поиск похожих блоков и т.д.



[7] K. Dabov, A. Foi, and K. Egiazarian. Video denoising by sparse 3D transform-domain collaborative filtering, EUSIPCO, 2007

- Вместо VBM3D была реализован фильтр, который:
  - Использует векторы движения из предыдущей итерации как стартовые векторы.
  - Использует фильтрацию со псевдослучайным выбором размера 3D блока на каждой итерациии.
  - Работает в 20 раз быстрее (0.07 fps на CPU 2.8 GHz на разрешении 352×288).
  - Лучше на 0.76 дБ по PSNR.

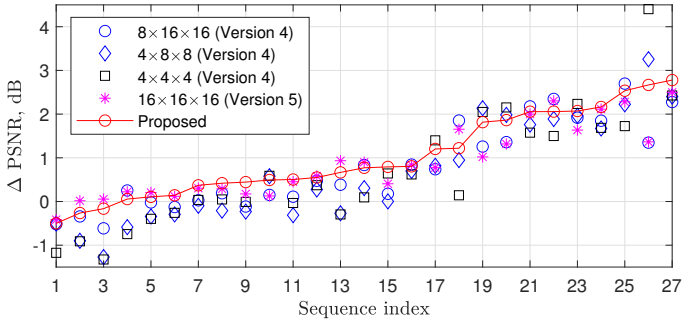


Таблица: Количество кадров, сжимаемых в секунду на видео 'Foreman', разрешение 352×288 на CPU 2.8 GHz

R_T, kbps	*MJPEG*	*MJPEG2000*	*x264-intra*		*x265-intra*	*CS-JPEG*
			ultrafast	*veryslow*	*ultrafast*	
600	223	12	347	24	17	**552**
1000	221	12	271	19	16	**517**
1500	213	12	217	16	15	**444**
2000	199	12	177	14	14	**381**
В среднем	214	12	253	18	16	**474**
%	100	6	118	8	7	**221**

(a) x264-intra (ultrafast) (b) x265-intra (ultrafast) (c) CS-JPEG

Рис.: Кадр 150 видео 'Hall' на 600 kbps

Включить деморолик: `csjpeg_demo.mp4`

- Взятие измерений и восстановление видео можно осуществлять при помощи нейронных сетей.
- Восстановление можно реализовать гораздо быстрее на базе нейронных сетей, аппроксимирующих итеративный алгоритм (ISTA). Например, для разрешения 352×288 достигнуты следующие результаты:
  - ISTA-Net[8] осуществляет декодирование изображений со скоростью 0.47 и 13.8 fps на CPU и GPU, соответственно.
  - CSRN[9] восстанавливает изображения со скоростью 293 fps на GPU (14.32 fps для HD разрешения).

[8] Jian Zhang and Bernard Ghanem, ISTA-Net: Interpretable Optimization-Inspired Deep Network for Image Compressive Sensing, CVPR, 2018.

[9] Y. Zhou et al., A Lightweight Recurrent Learning Network for Sustainable Compressed Sensing, IEEE Transactions on Emerging Topics in Computational Intelligence, 2023

ITMO more than a UNIVERSITY

Спасибо за внимание!