Video Compression

EE274

Kedar Tatwawadi
(Video Engineering, Apple Inc)

Video Is Growing and Innovating

82%
of the internet will be video by 2021

300%
annual increase of YouTube home page hits

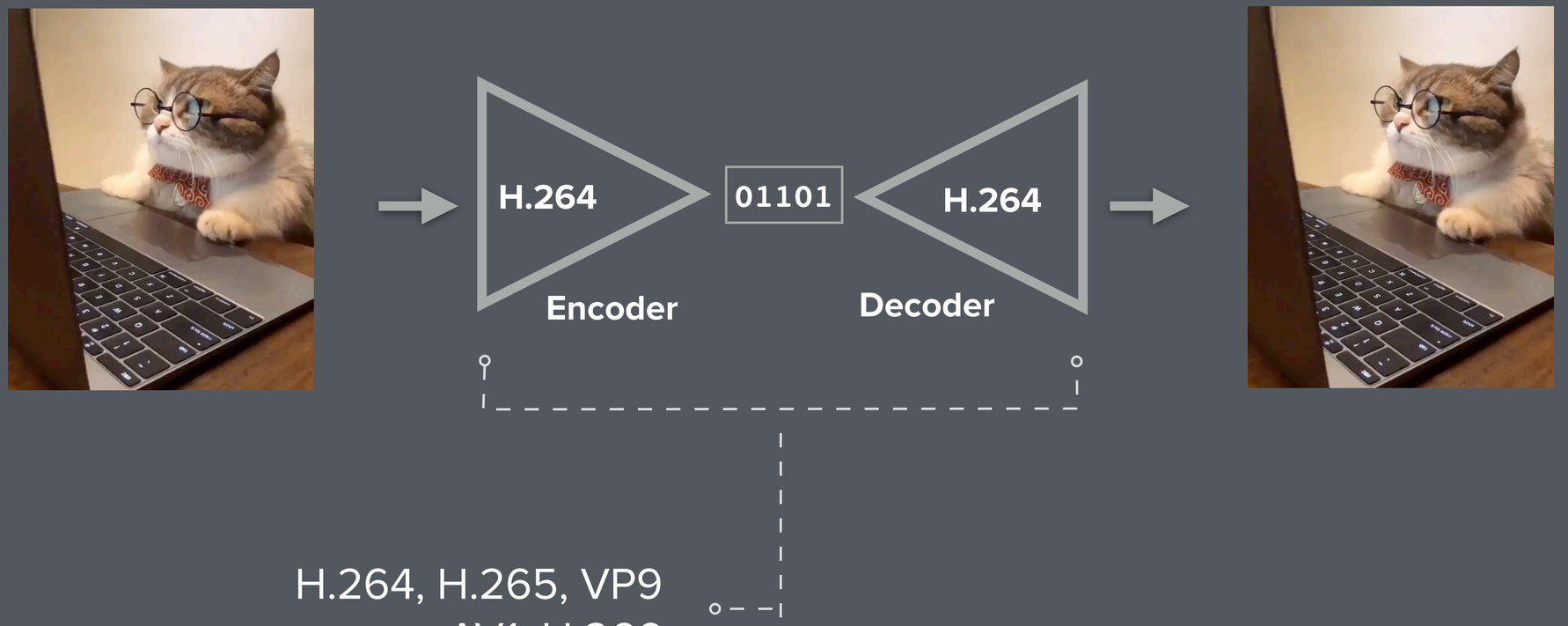
23%
video analytics CAGR over next 6 years

14B /day
videos on Snap

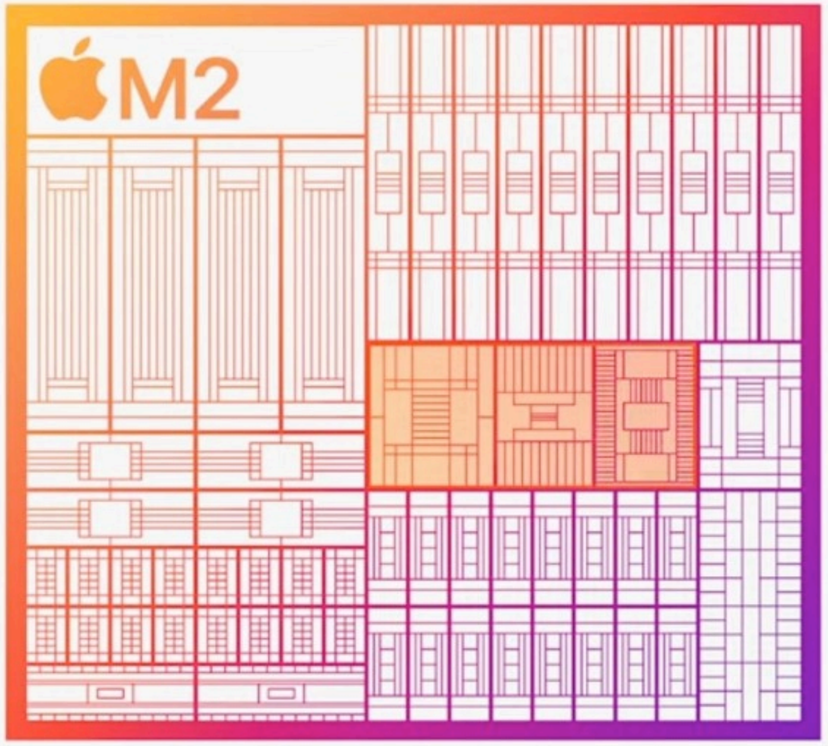
45 billion
cameras in the world by 2022

References: Cisco Visual Networking Index, YouTube, Snap

Video Compression



Video codecs have dedicated silicon!



Media engine

8K H.264, HEVC, ProRes

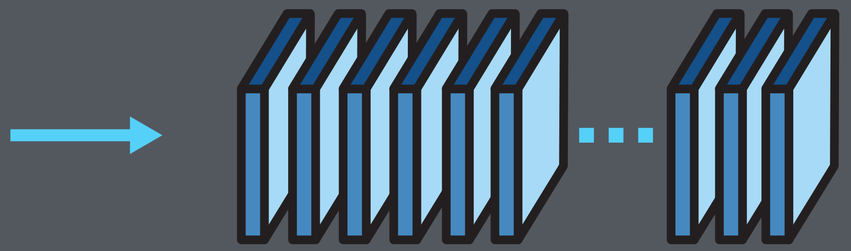
Video decode engine

Video encode engine

ProRes encode/decode engine

Video Compression

Target Video



Frames

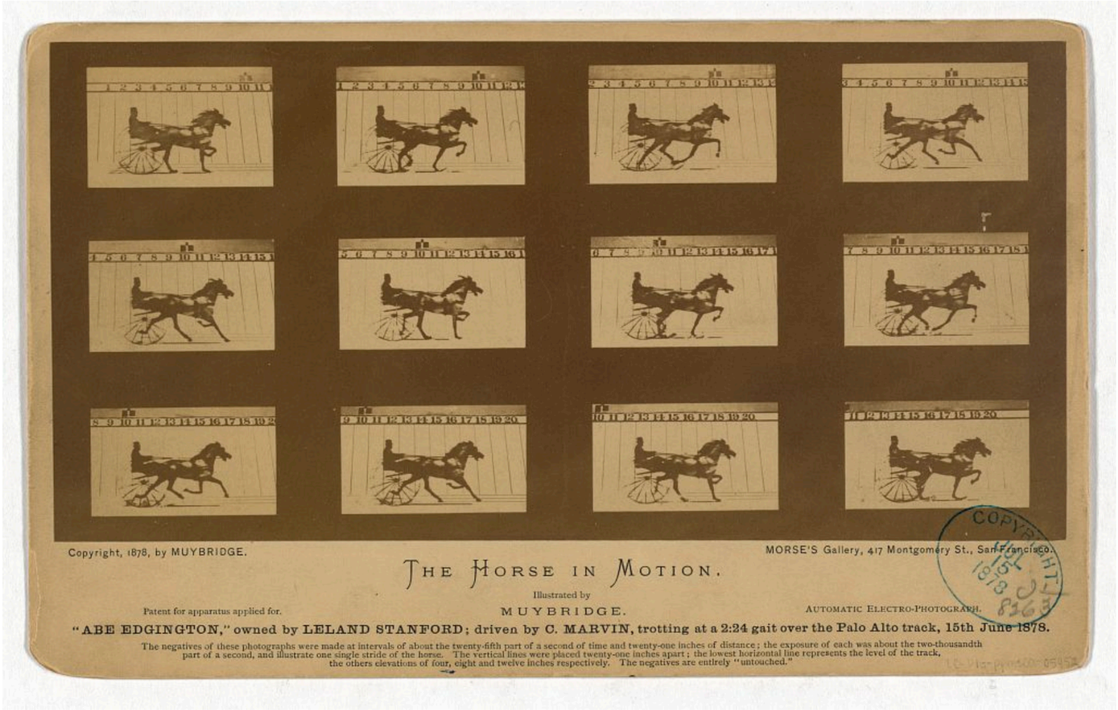
- Video = *“Motion Pictures”*

First “video” ever captured

File:The Horse in motion. "Abe Edgington," owned by Leland Stanford; driven by C. Marvin, trotting at a 2-24 gait over the Palo Alto track, 15th June 1878 LOC 13624627695.jpg

From Wikipedia, the free encyclopedia

File File history File usage Global file usage



Size of this preview: 800 × 509 pixels. Other resolutions: 320 × 204 pixels | 640 × 408 pixels | 1,024 × 652 pixels.

Jockey 720p

- FPS= frames/sec -> 30
- X,Y -> 720x1280

Jockey 720p


```
[→  jockey_videos mediainfo jockey_720p.y4m
General
Complete name                            : jockey_720p.y4m
Format                                   : YUV4MPEG2
File size                                : 169 MiB
Duration                                 : 4 s 267 ms
Overall bit rate                         : 332 Mb/s

Video
Format                                   : YUV
Duration                                 : 4 s 267 ms
Bit rate                                 : 332 Mb/s
Width                                    : 1 280 pixels
Height                                   : 720 pixels
Display aspect ratio                     : 16:9
Frame rate                               : 30.000 FPS
Color space                              : YUV
Chroma subsampling                       : 4:2:0
Scan type                                : Progressive
Compression mode                         : Lossless
Bits/(Pixel*Frame)                       : 12.000
Stream size                              : 169 MiB
```

Jockey 720p -> H264 CRF20

- RAW -> 332 Mb/s
- CRF20 -> 6.2 Mb/s

  (PSNR -> 43)

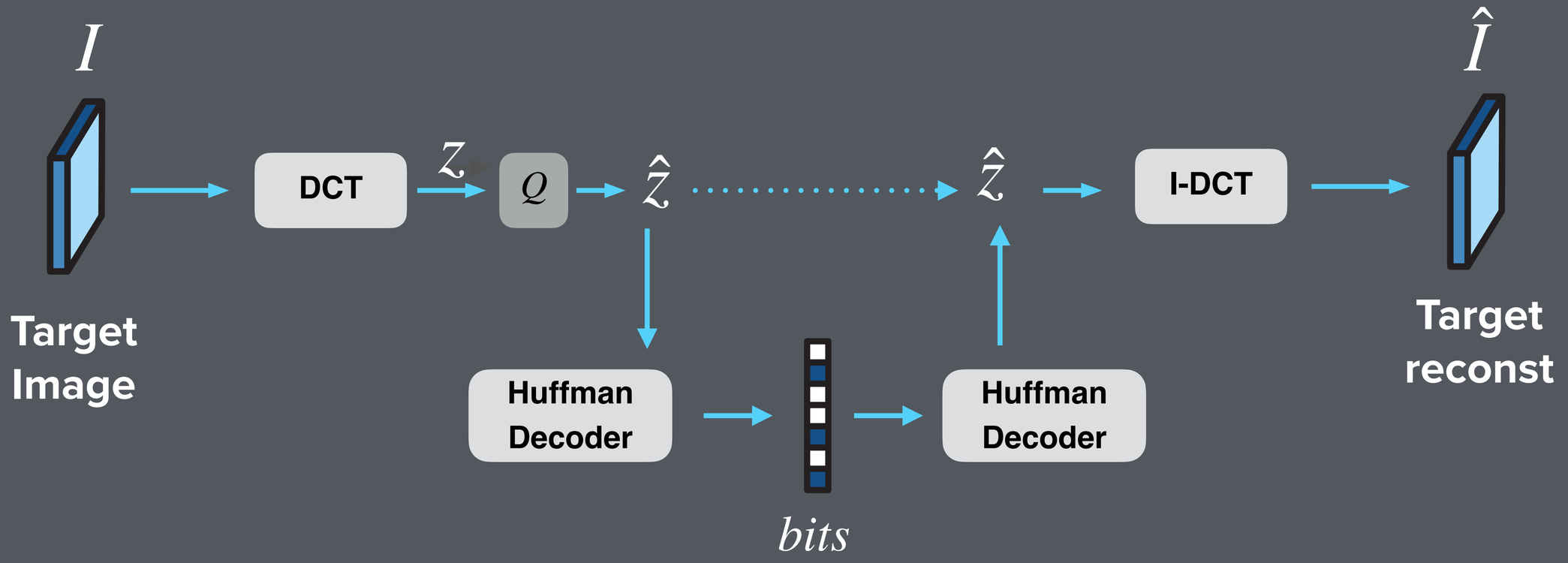
```
~ ffmpeg -y -i jockey_720p.y4m -codec:v libx264 -crf 20 -x264-params keyint=8:bframes=0 jockey_crf20.mp4
```

Jockey 720p -> H264 CRF40

- RAW -> 332 Mb/s
- CRF20 -> 6.2 Mb/s
  (PSNR -> 43)
- CRF40 -> 0.8 Mb/s
  (PSNR -> 33)

```
~ ffmpeg -y -i jockey_720p.y4m -codec:v libx264 -crf 20 -x264-params keyint=8:bframes=0 jockey_crf20.mp4
```

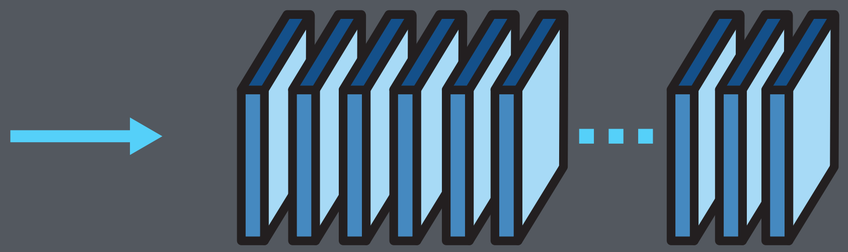
JPEG -> Recap



Goal: $\min_{L(bits) \leq B} d(I, \hat{I})$

Compressing Video as I-frames

Target Video



Compress each frame like a Image (I-frame)

Jockey 720p -> Iframe compression

- RAW -> 332 Mb/s
- CRF20, I-frame -> 9 Mb/s

  (PSNR -> 44)

Frame 0

Frame 1

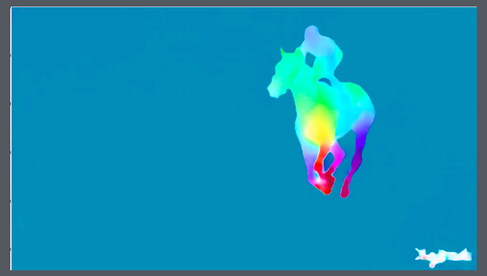
Compressing the second frame

Frame-0

Frame-1

Compressing the second frame

Frame-0



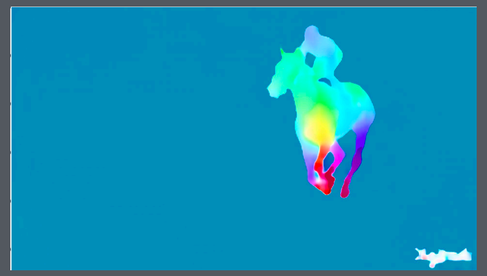
Find,Encode Motion

Motion Compensation

Frame-1

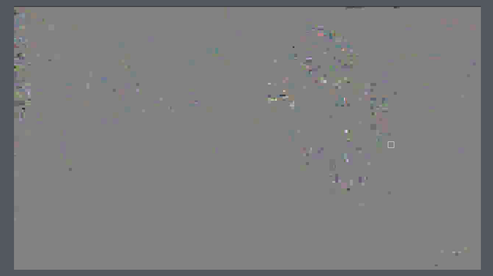
Compressing the second frame

Frame-0



Find,Encode Motion

Motion Compensation

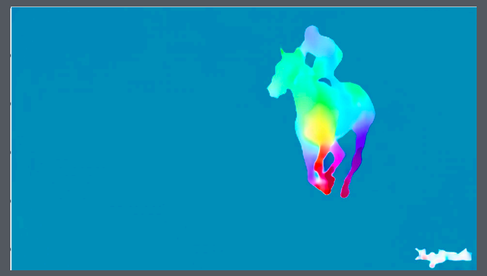


Residue

Frame-1

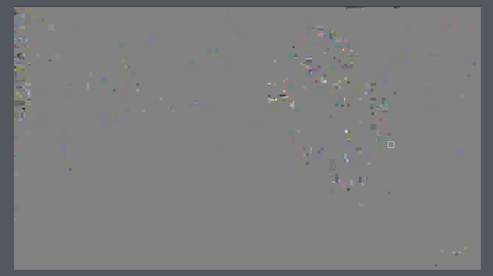
Compressing the second frame

Frame-0

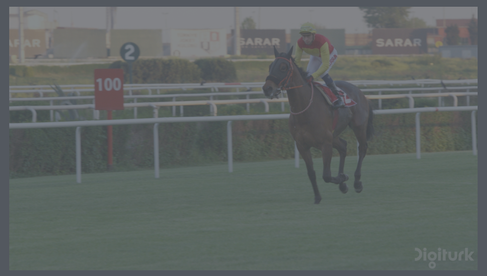


Find,Encode Motion

Motion Compensation



Residue

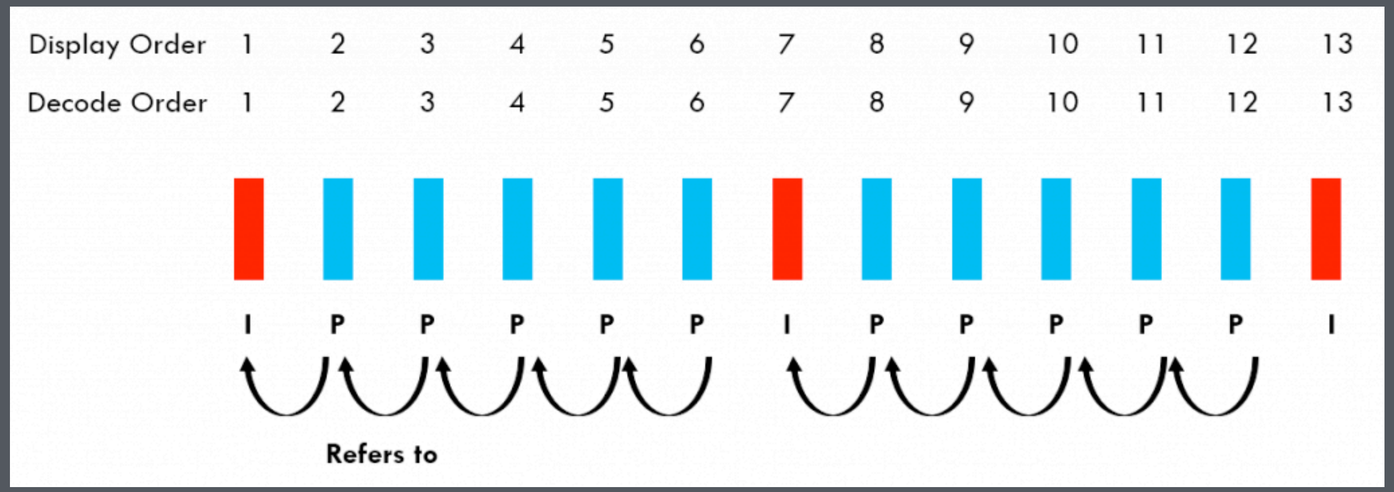


Frame-1

Encoded as part of the bit-stream

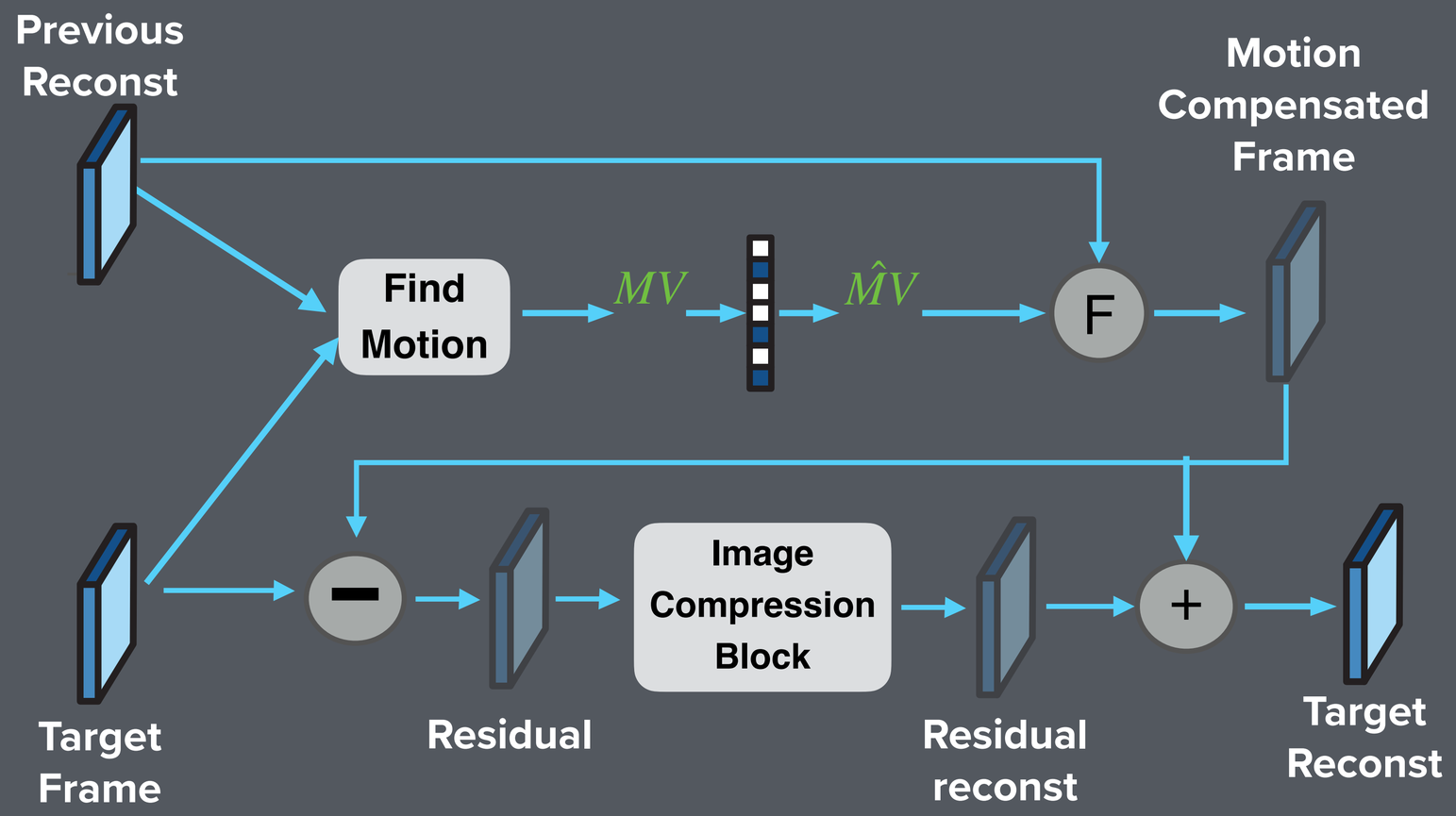
I-P frame coding

Target Video



- P-frame -> “prediction frame”
- Predict based on the previous frame
- **Keyint -> 6** (every 6th frame an I-frame)

IP-coding



Motion-compensated Target Residual

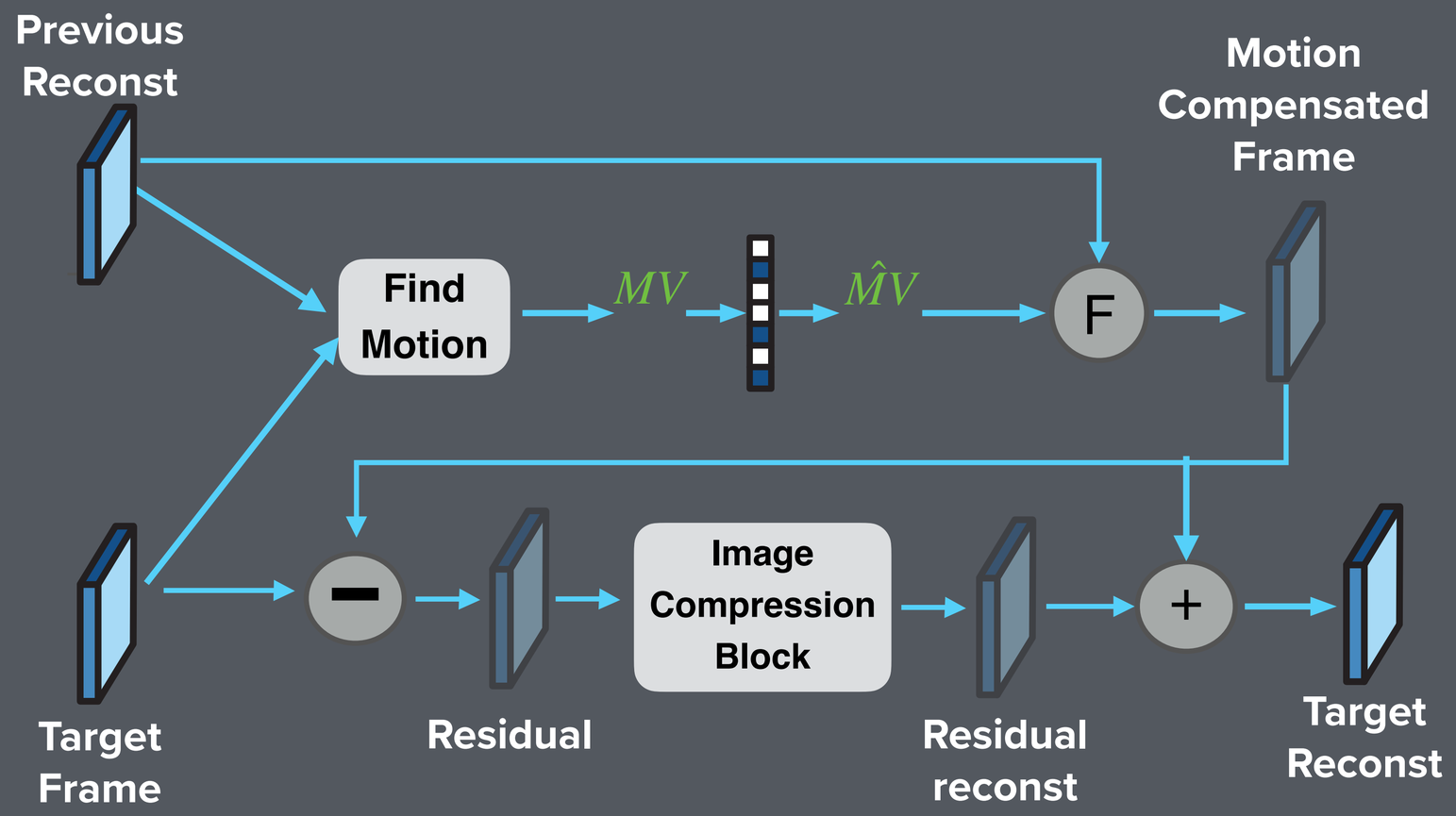
Jockey 720p -> H264 CRF20

- RAW -> 332 Mb/s
- CRF20 -> 6.2 Mb/s

  (PSNR -> 43)

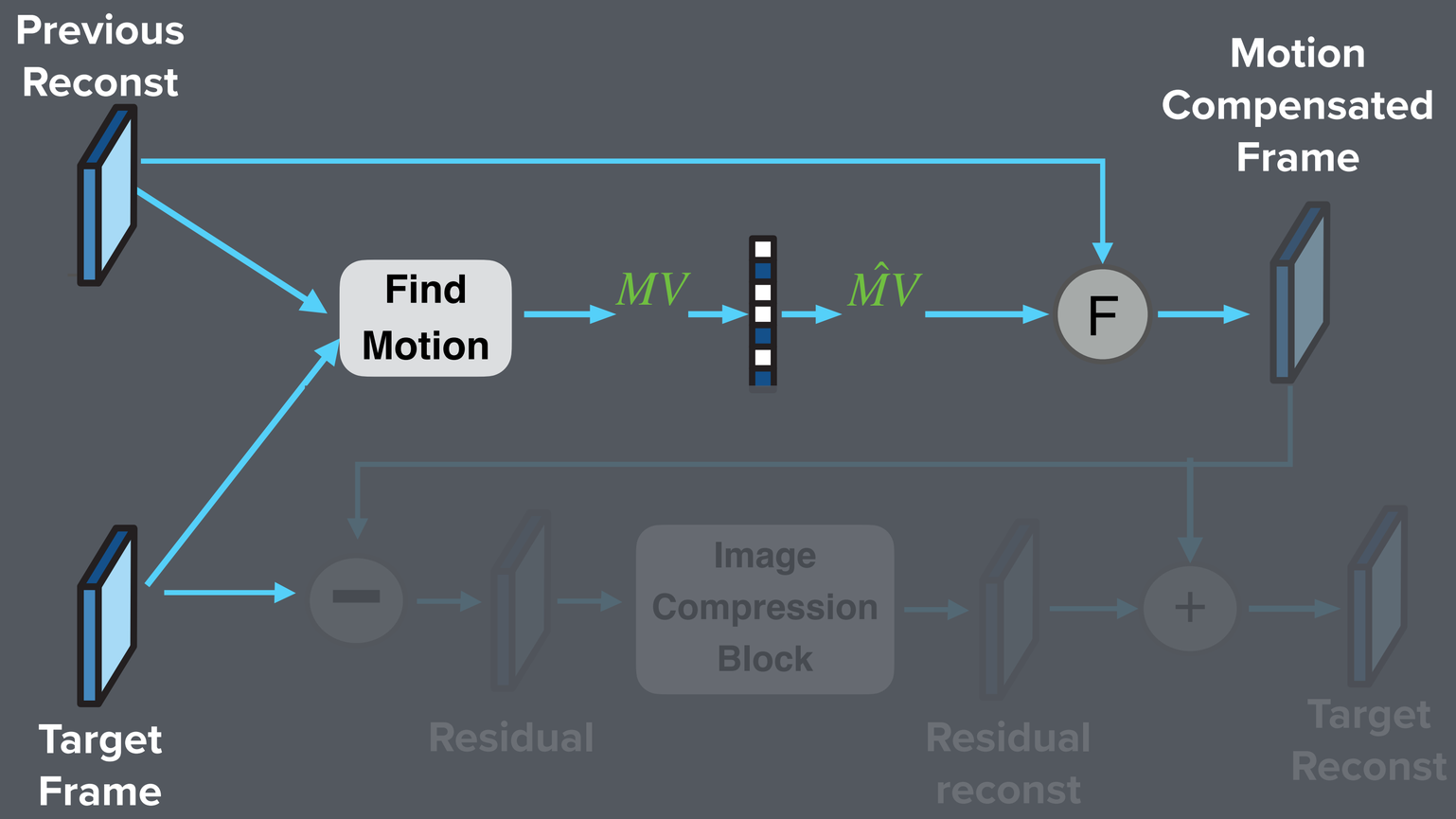
```
~ ffmpeg -y -i jockey_720p.y4m -codec:v libx264 -crf 20 -x264-params keyint=8:bframes=0 jockey_crf20.mp4
```

IP-coding

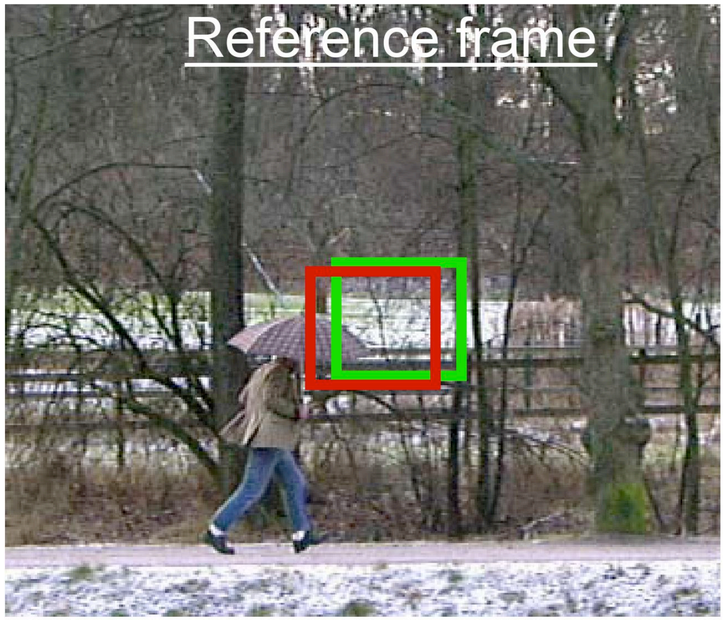


Motion-compensated Target Residual

IP-coding


Previous Reconst
Motion Compensated Frame
Find Motion
MV
$\hat{MV}$
F
Target Frame
Image Compression Block
Residual
Residual reconst
Target Reconst

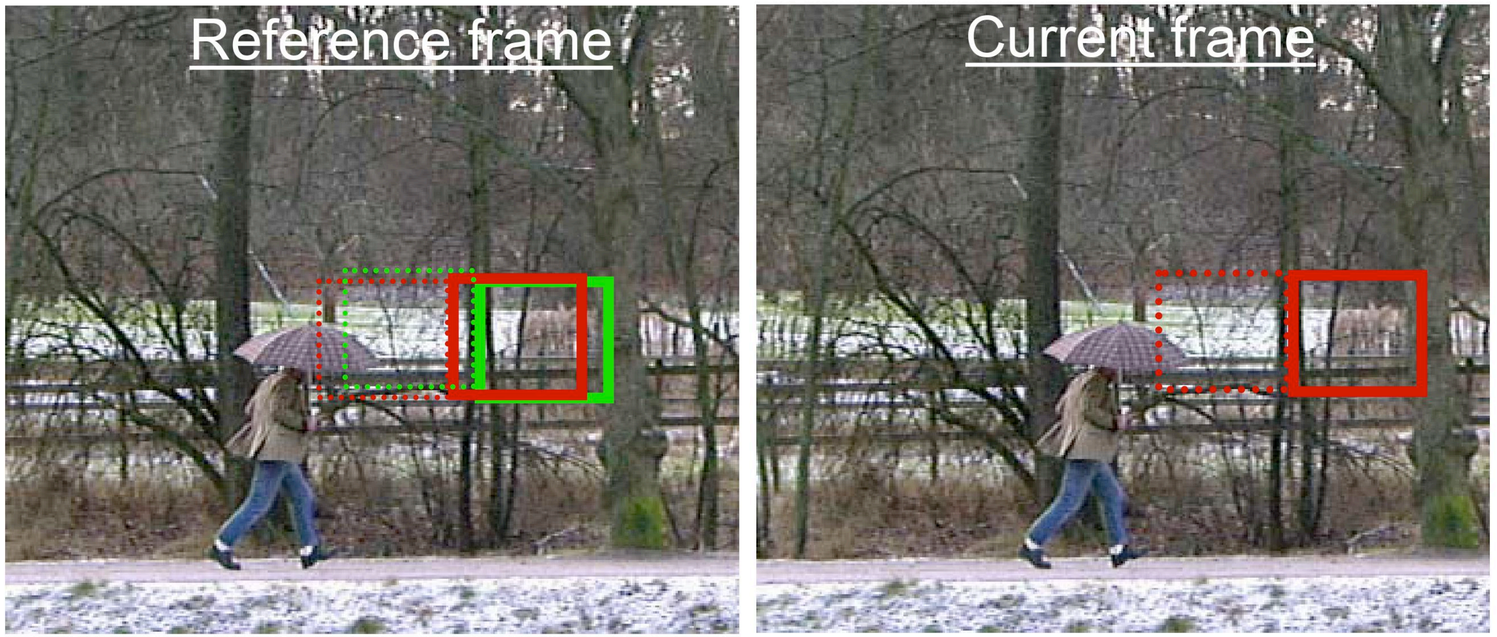
Block-matching algorithm



Block is compared with a shifted array of pixels in the reference frame to determine the best match

Block of pixels is considered

Block-matching algorithm

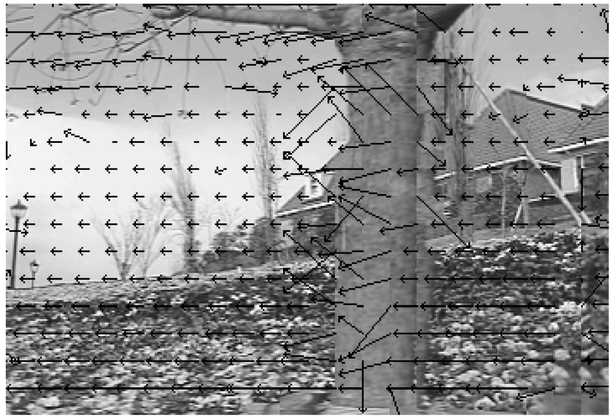


. . . process repeated for the next block

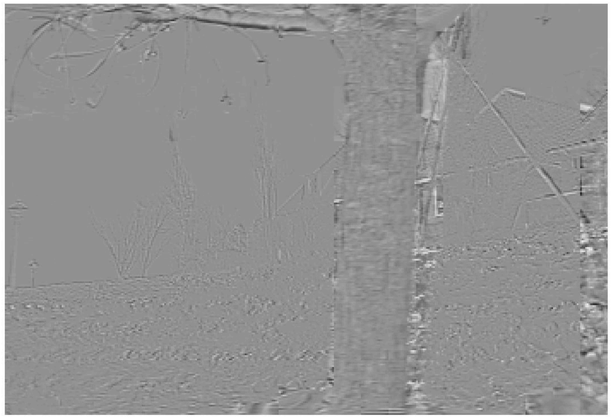
Motion-compensated prediction: example

Previous frame

Current frame



Current frame with displacement vectors

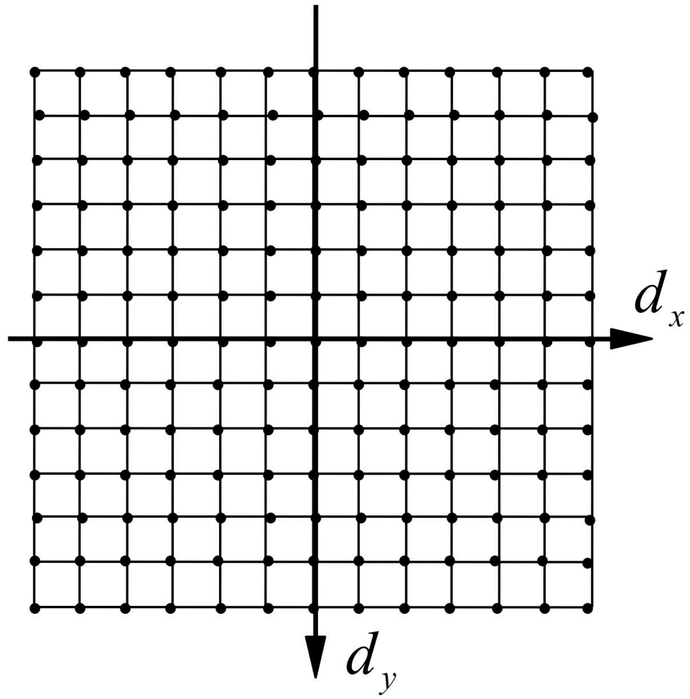


Motion-compensated Prediction error

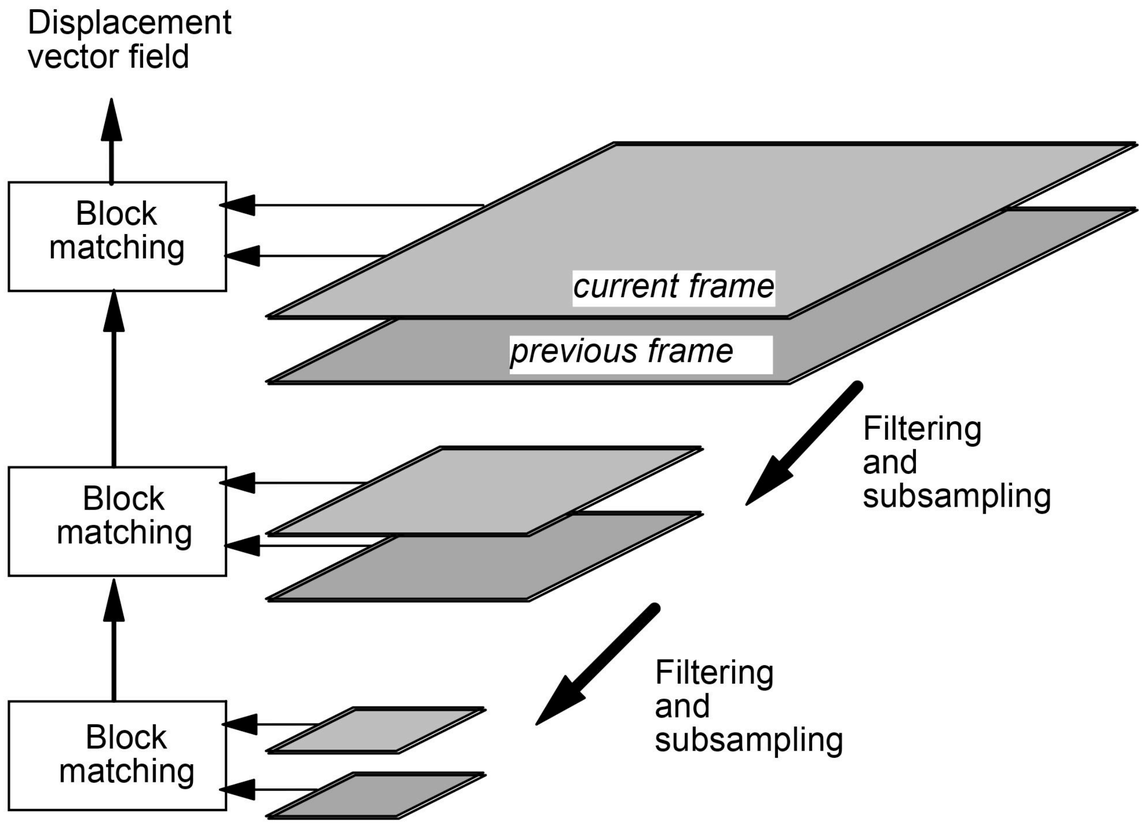
Blockmatching: search strategies I

Full search

- All possible displacements within the search range are compared.
- Computationally expensive
- Highly regular, parallelizable

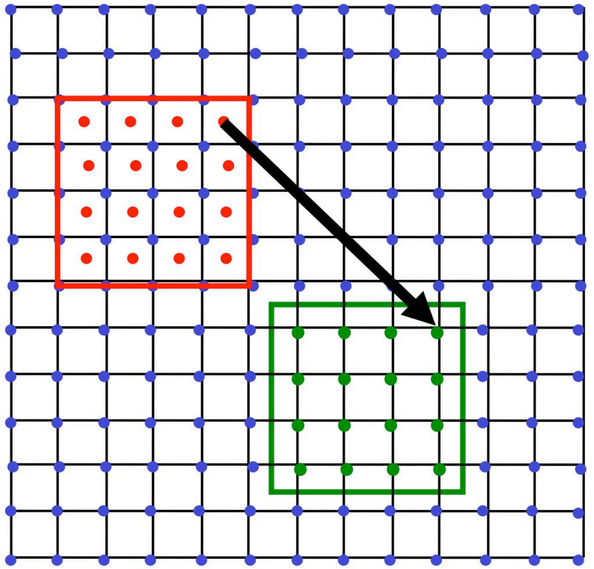


Hierarchical blockmatching



Sub-pel accuracy

- Interpolate pixel raster of the reference frame to desired fractional pel accuracy (e.g., by bi-linear interpolation)
- Straightforward extension of displacement vector search to fractional accuracy
- Example: half-pel accurate displacements



$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} 4.5 \\ 4.5 \end{pmatrix}$$

Case Study -> Foreman Video

- Size: 352x288
- CRF20, H264
- Keyint = 8

  (I frame at 0,8,16,…

  P-frame otherwise)

Case Study -> Jockey CRF20

- RAW -> 332 Mb/s
- CRF20 -> 6.2 Mb/s

  (PSNR -> 43)

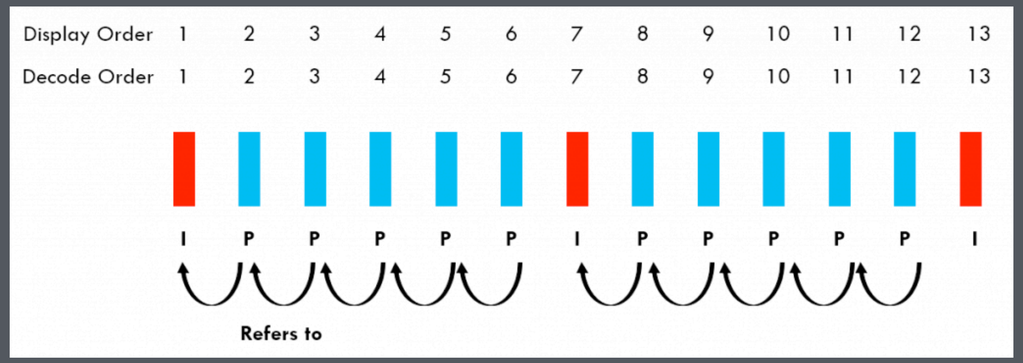
```
~ ffmpeg -y -i jockey_720p.y4m -codec:v libx264 -crf 20 -x264-params keyint=8:bframes=0 jockey_crf20.mp4
```

Case Study -> Jockey CRF20

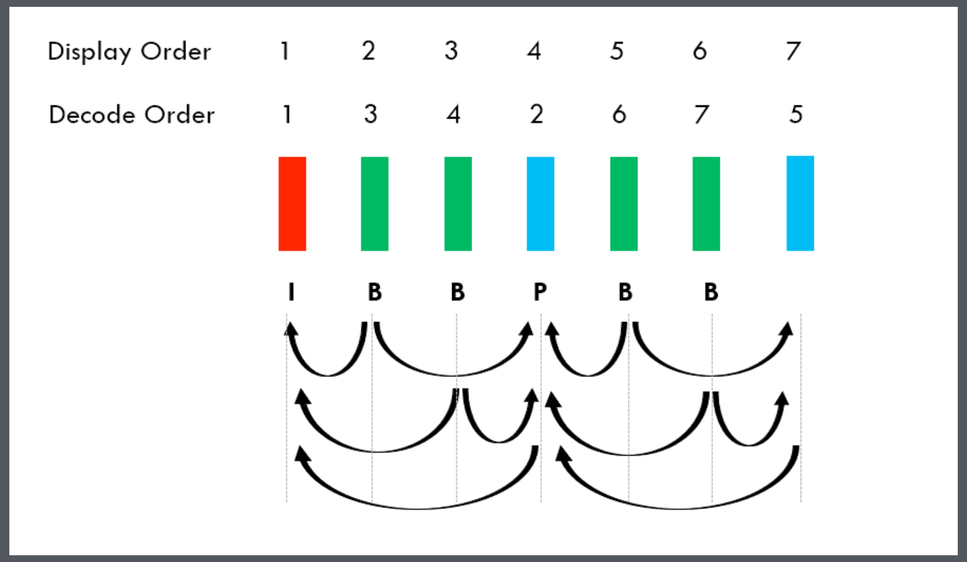

```
(base) (wovenv) ➜  jockey_videos mediainfo jockey_crf20.mp4
General
Complete name                            : jockey_crf20.mp4
Format                                   : MPEG-4
Format profile                           : Base Media
Codec ID                                 : isom (isom/iso2/avc1/mp41)
File size                                : 3.16 MiB
Duration                                 : 4 s 267 ms
Overall bit rate                         : 6 219 kb/s
Writing application                      : Lavf58.29.100

Video
ID                                       : 1
Format                                   : AVC
Format/Info                              : Advanced Video Codec
Format profile                           : High@L3.1
Format settings                          : CABAC / 3 Ref Frames
Format settings, CABAC                   : Yes
Format settings, Reference frames        : 3 frames
Format settings, GOP                     : M=1, N=8
Codec ID                                 : avc1
Codec ID/Info                            : Advanced Video Coding
Duration                                 : 4 s 267 ms
Bit rate                                 : 6 217 kb/s
Width                                    : 1 280 pixels
Height                                   : 720 pixels
Display aspect ratio                     : 16:9
Frame rate mode                          : Constant
Frame rate                               : 30.000 FPS
Color space                              : YUV
Chroma subsampling                       : 4:2:0
Bit depth                                : 8 bits
Scan type                                : Progressive
Bits/(Pixel*Frame)                       : 0.225
Stream size                              : 3.16 MiB (100%)
Writing library                          : x264 core 155 r2917 0a84d98
Encoding settings                        : cabac=1 / ref=3 / deblock=1:0:0 / ana
lyse=0x3:0x113 / me=hex / subme=7 / psy=1 / psy_rd=1.00:0.00 / mixed_ref=1 / me_
range=16 / chroma_me=1 / trellis=1 / 8x8dct=1 / cqm=0 / deadzone=21,11 / fast_ps
kip=1 / chroma_qp_offset=-2 / threads=22 / lookahead_threads=3 / sliced_threads=
0 / nr=0 / decimate=1 / interlaced=0 / bluray_compat=0 / constrained_intra=0 / b
frames=0 / weightp=2 / keyint=8 / keyint_min=1 / scenecut=40 / intra_refresh=0 /
 rc_lookahead=8 / rc=crf / mbtree=1 / crf=20.0 / qcomp=0.60 / qpmin=0 / qpmax=69
 / qpstep=4 / ip_ratio=1.40 / aq=1:1.00
Codec configuration box                  : avcC
```

I,P,B frame coding



IP-frame coding



IPB-frame coding

- **P-frame** -> “prediction frame” (only references past frame)
- **B-frame** -> references past and future frames.
- Interpolation vs Extrapolation

I,P,B frame types

- **I-Frames Only:**

  Simple, used in video editing softwares

- **I-Frames** + **P-Frames:**

  Better compression than I-frame only.

  Also called “low-latency/low-delay” mode. Used for video conferencing

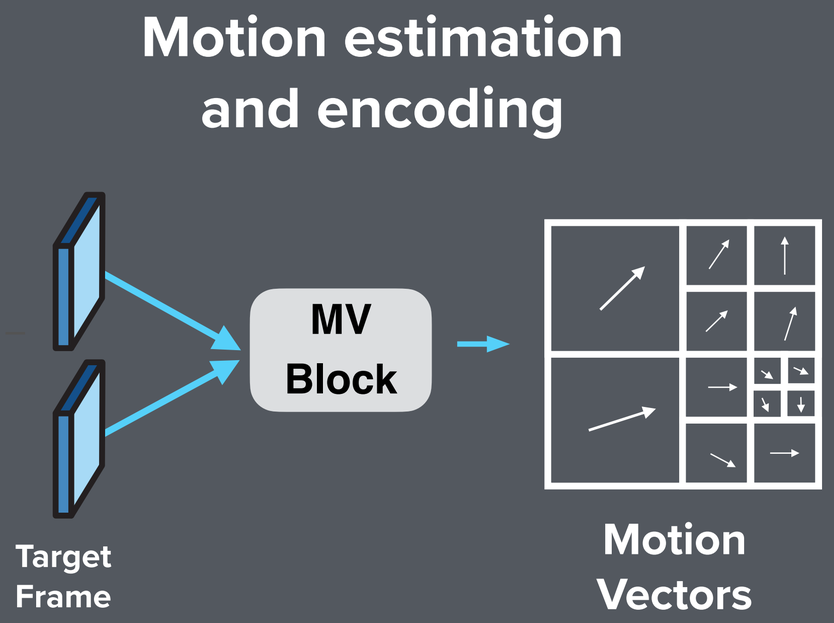
- **I-Frames** + **P-Frames** + **B-Frames:**

  Typically gives the best compression (also called “Random Access Mode”)

  Ideal for Video Streaming (Youtube, Netflix…)

Motion-compensated Target Residual

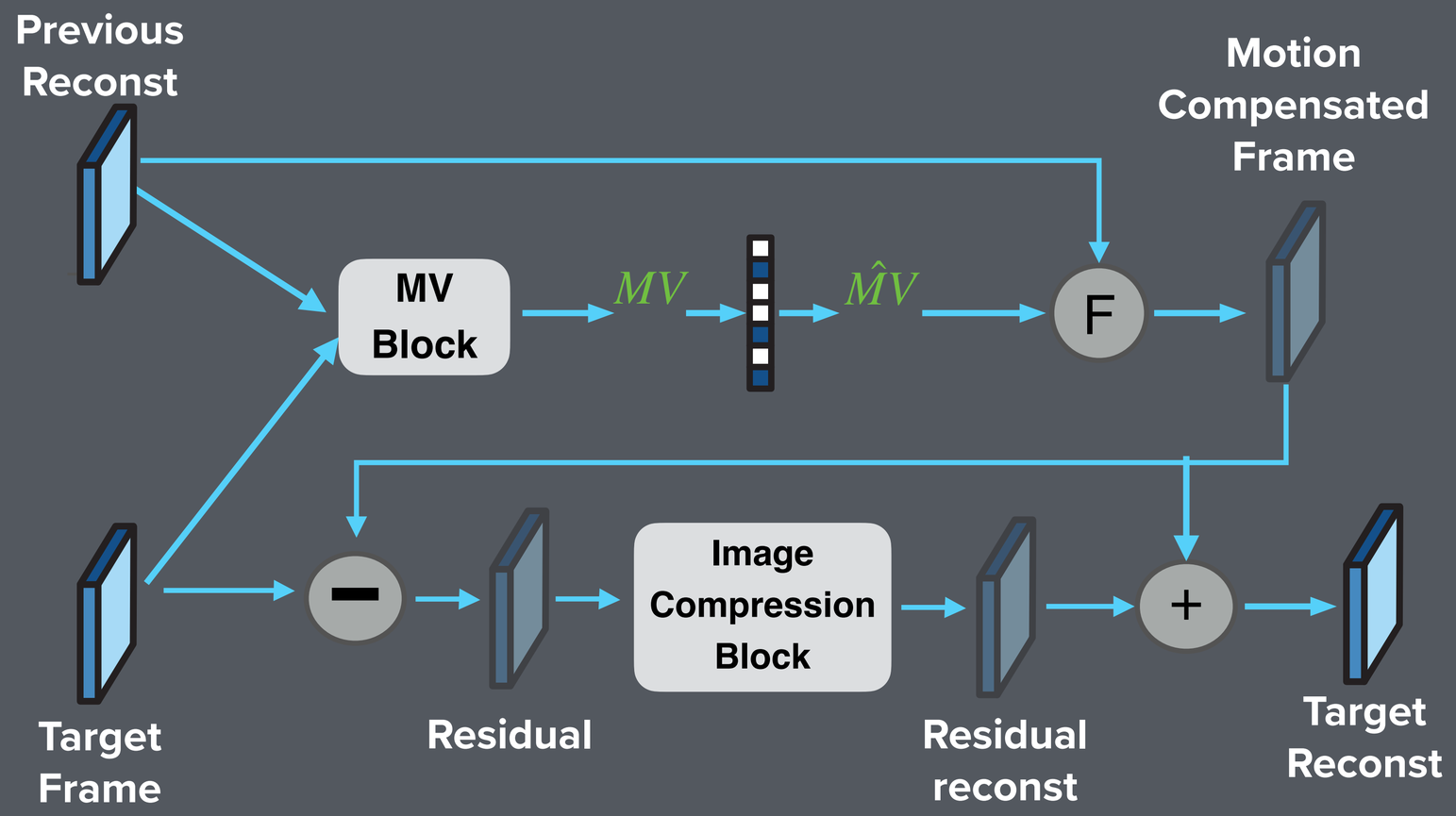
Iterative Block-search based Motion



- Axis-aligned blocks, discretized motion directions and magnitudes
- Extremely efficient (with some algorithmic optimizations)
- Leads to significant blocky artifacts, needing some “de-blocking filtering” at the end

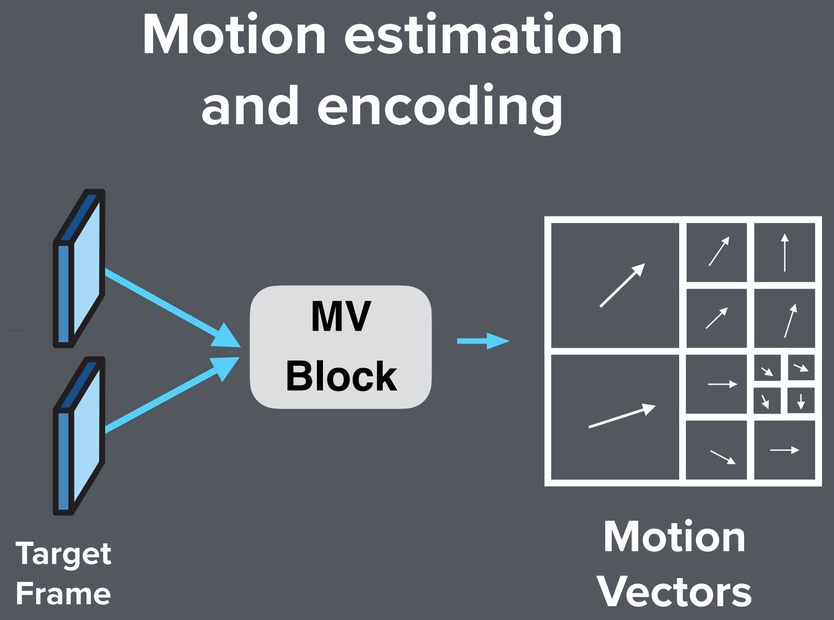
Motion-compensated Target Residual

Traditional IP coding



Motion-compensated Target Residual

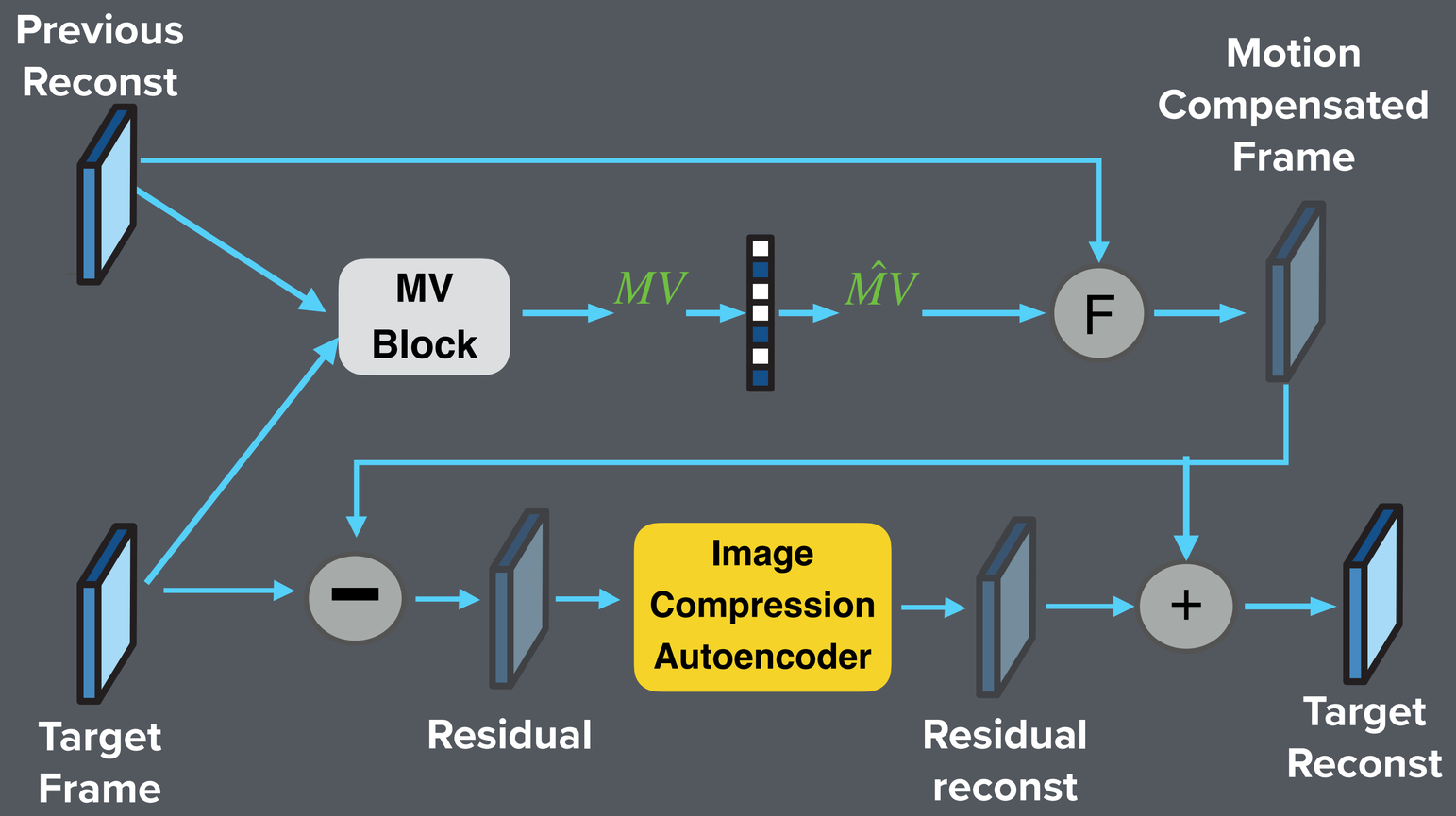
Iterative Block-search based Motion



- Axis-aligned blocks, discretized motion directions and magnitudes
- Extremely efficient (with some algorithmic optimizations)
- Leads to significant blocky artifacts, needing some “de-blocking filtering” at the end

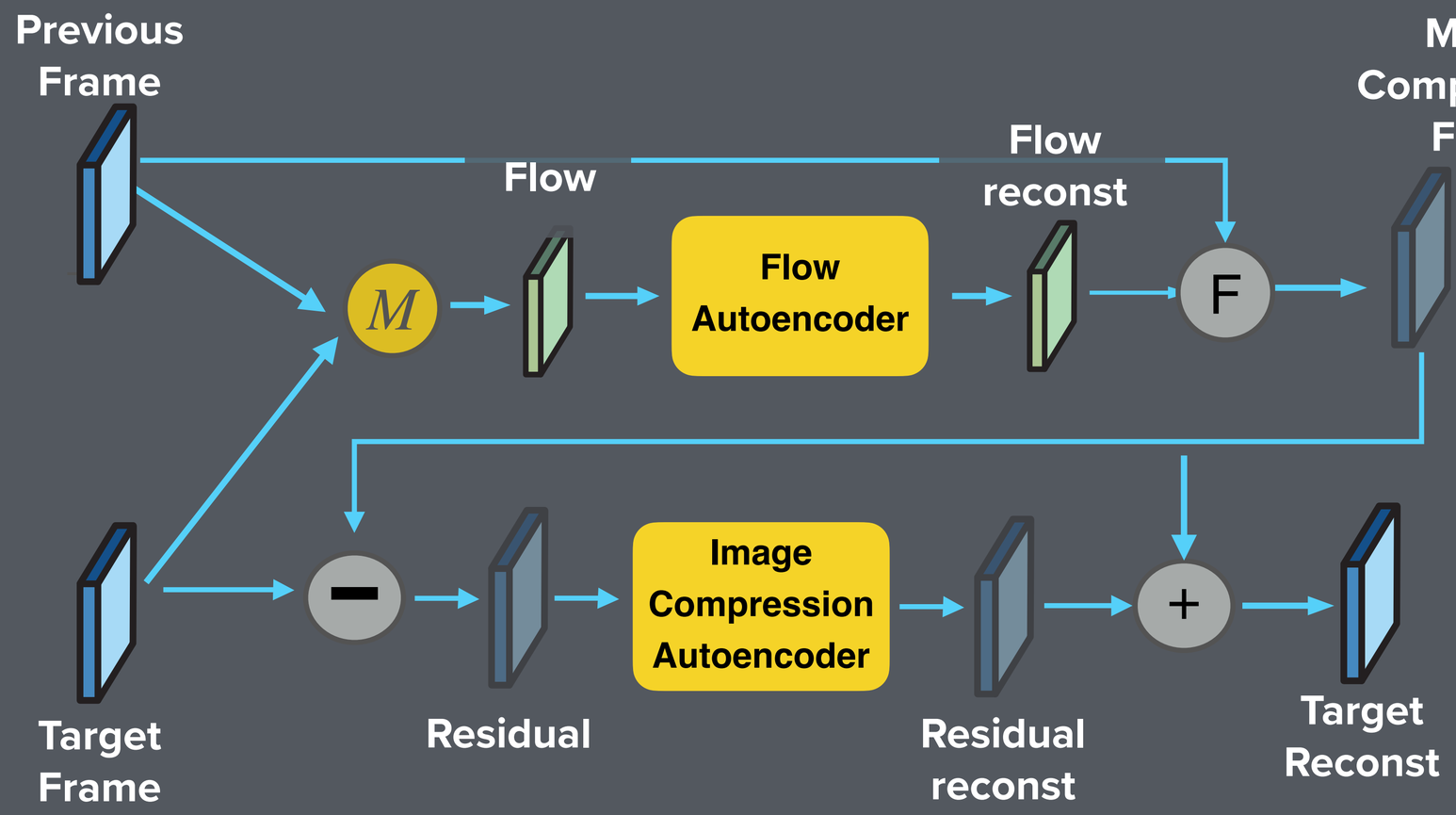
Motion-compensated Target Residual

IP coding -> ML-based



Motion-compensated Target Residual

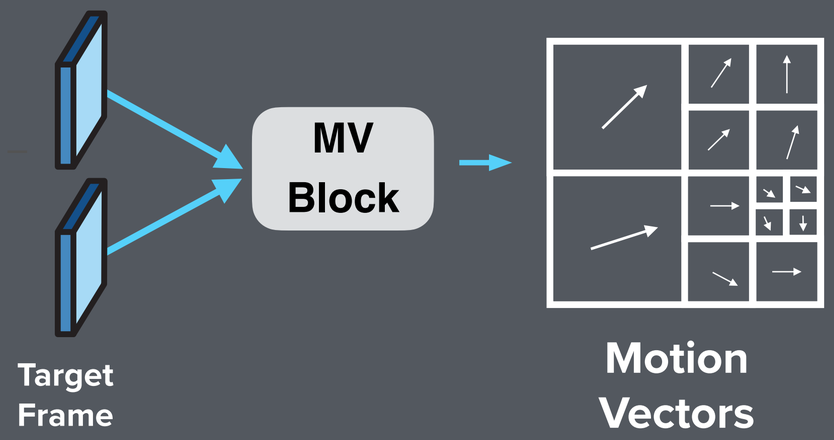
End-to-End Learned Video Codec



Motion-compensated Target Residual

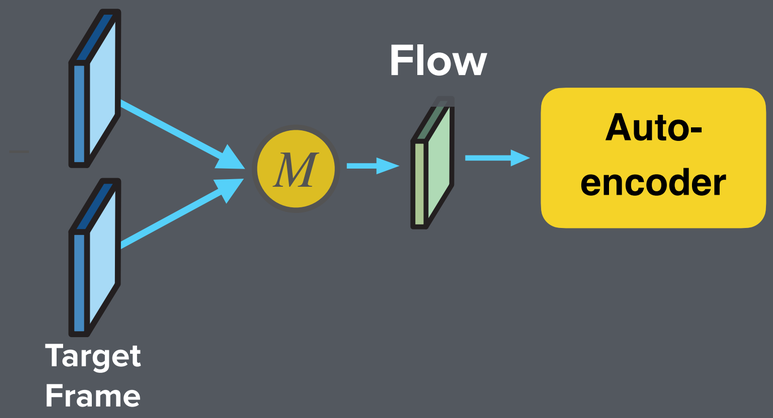
Better understanding of motion

Traditional



- Axis-aligned blocks
- Discretized motion directions and magnitudes

Learned Motion



- Motion is pixel-wise
- Network decides the tradeoff in accuracy vs bits of Flow compression

Motion-compensated Target Residual

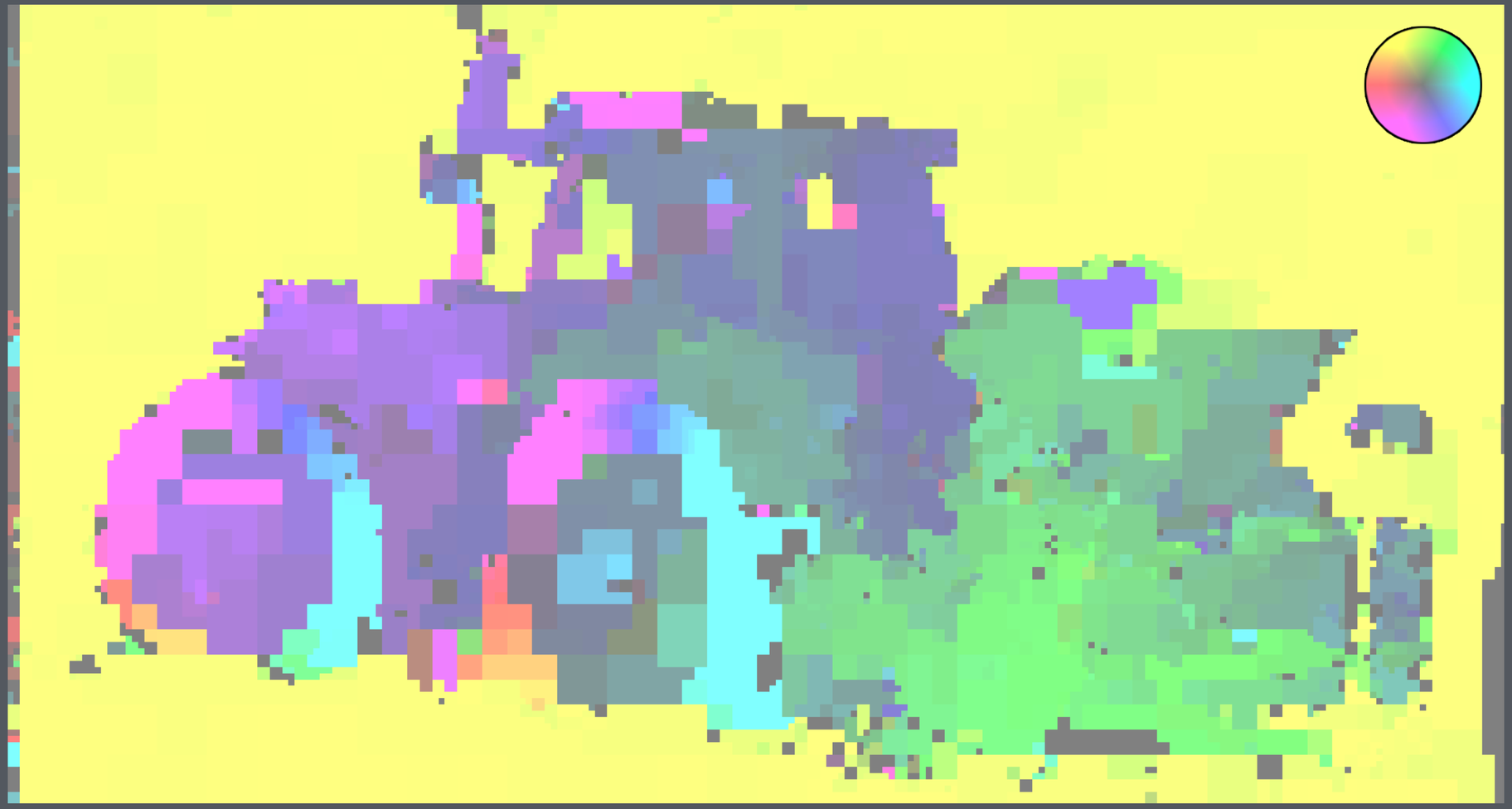
Example: Tractor Video

Motion-compensated

Target

Residual

Example: Tractor Video



Motion-compensated

Target

Residual

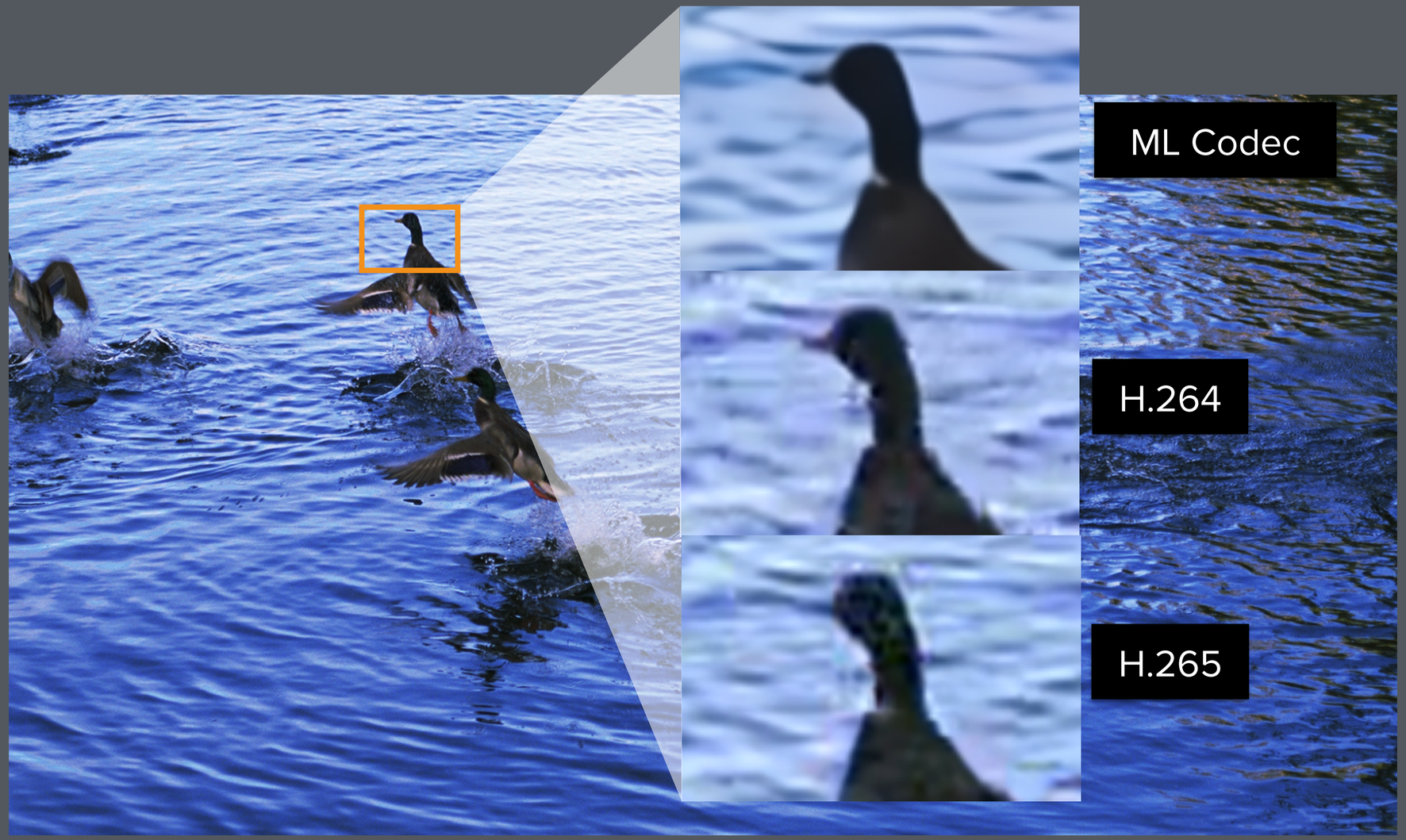
Example: Tractor Video

Motion-compensated

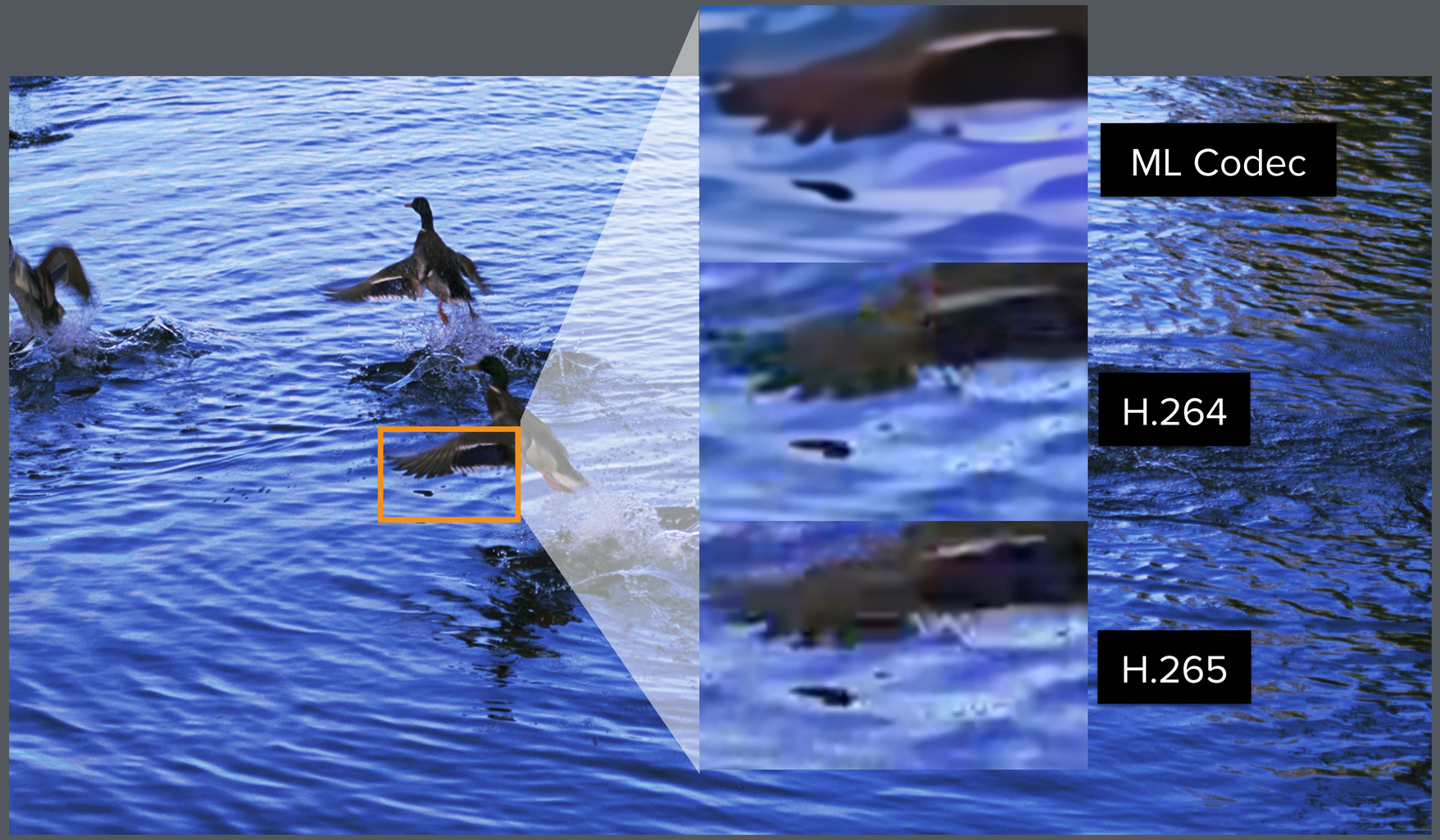
Target

Residual

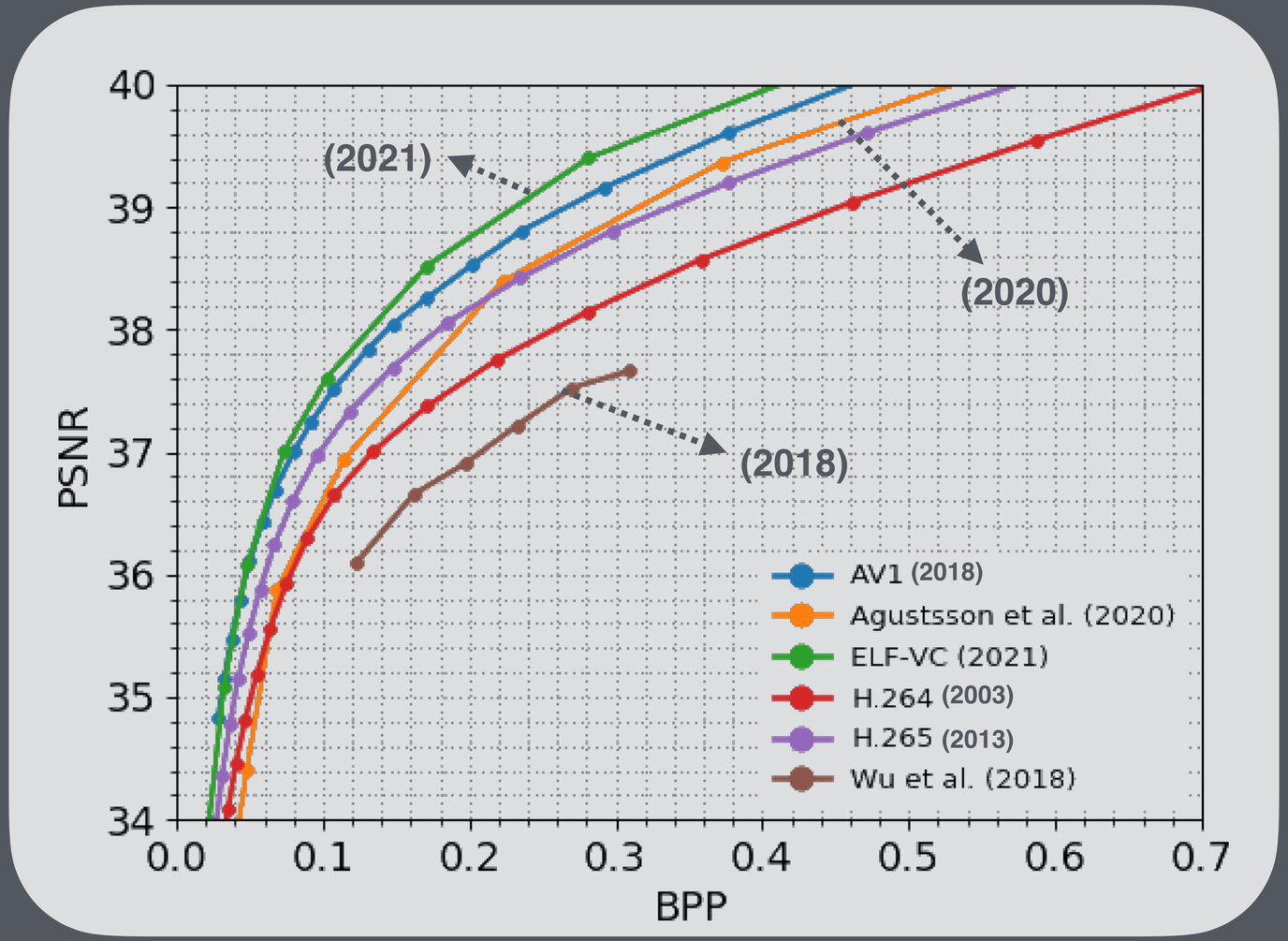
Example: Ducks Take Off



Example: Ducks Take Off

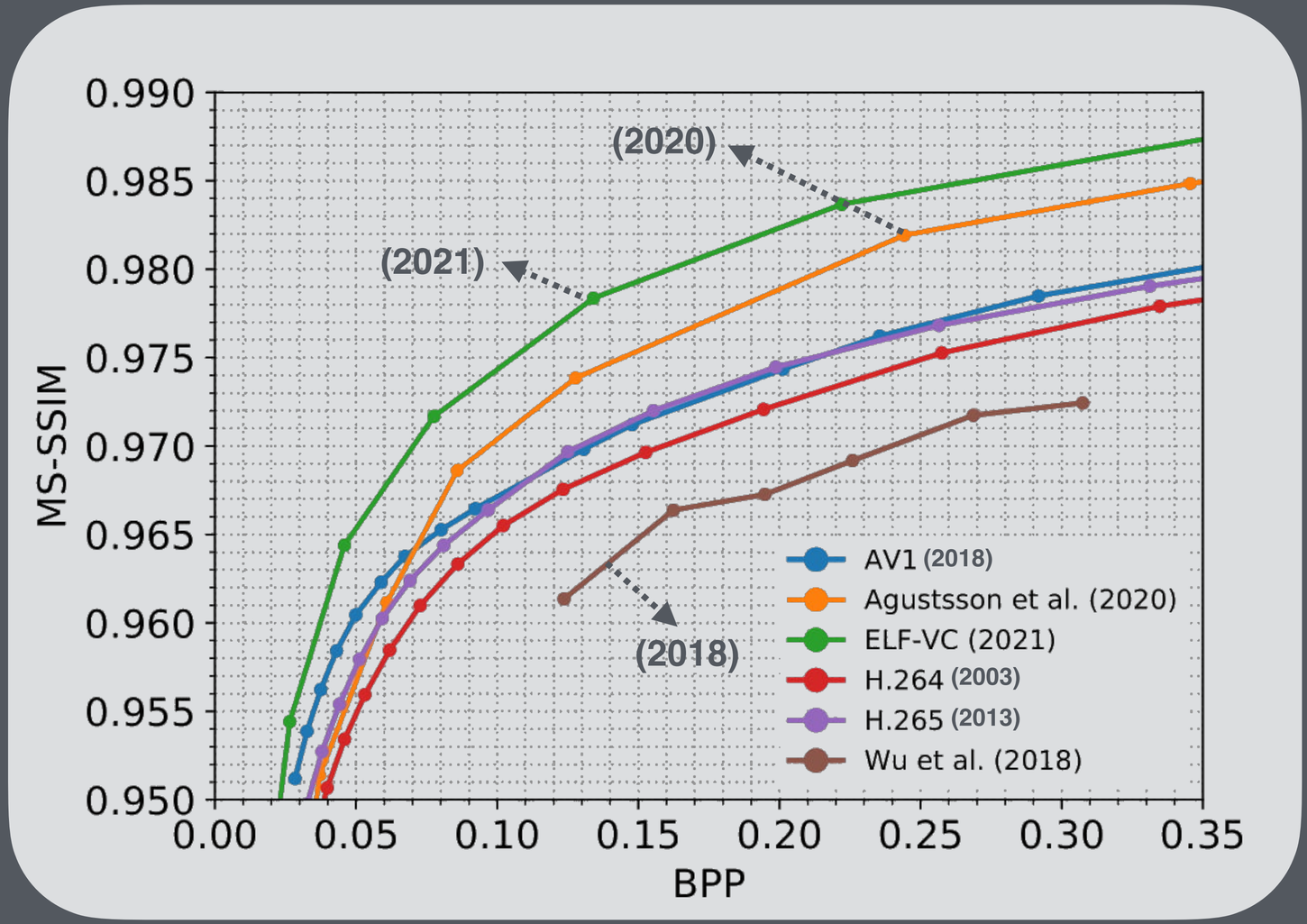


Learned Video Codecs: PSNR



Results on UVG dataset, low-latency setting, PSNR, keyint=16

Learned Video Codecs: MS-SSIM



Results on UVG dataset, low-latency setting, MS-SSIM, keyint

Video Compression -> Conclusion

- **Conceptually Simple -> Motion + Residual coding:**

  Uses 2-step approach -> find and encode motion, encode the residual. The complexity comes in how to implement these blocks.

- **Lots of parameters:**

  keyint=?,

  How many I,P,B?

  How many bits to give to each frame? (“Rate control”)

- **ML-based codecs:**

  Significant improvements in the past 2-3 years, but lot more to come!

Motion-compensated Target Residual

Thank You!

Motion-compensated Target Residual

My Team at Apple is Hiring!

- Hiring to work on ML-based Image/Video Compression/Processing
- **Email: kedar.tatwawadi@apple.com** for More Details

Motion-compensated Target Residual