Lecture 12:

Misc Video Topics

Visual Computing Systems
Stanford CS348K, Spring 2020

Massively parallel video encoding

Challenge

- Video encoding is inherently sequential (current frame represented in terms of data from previous frames)

- Coarse-grained parallelism is possible, at the cost of reduced compression (additional keyframes)
  - 4K video: keyframe ~ 11MB, interframe ~ 10s of KB

- Question: Is it possible to parallelize video encode onto thousands of cores?

Expressing a video encoder in a continuation passing style

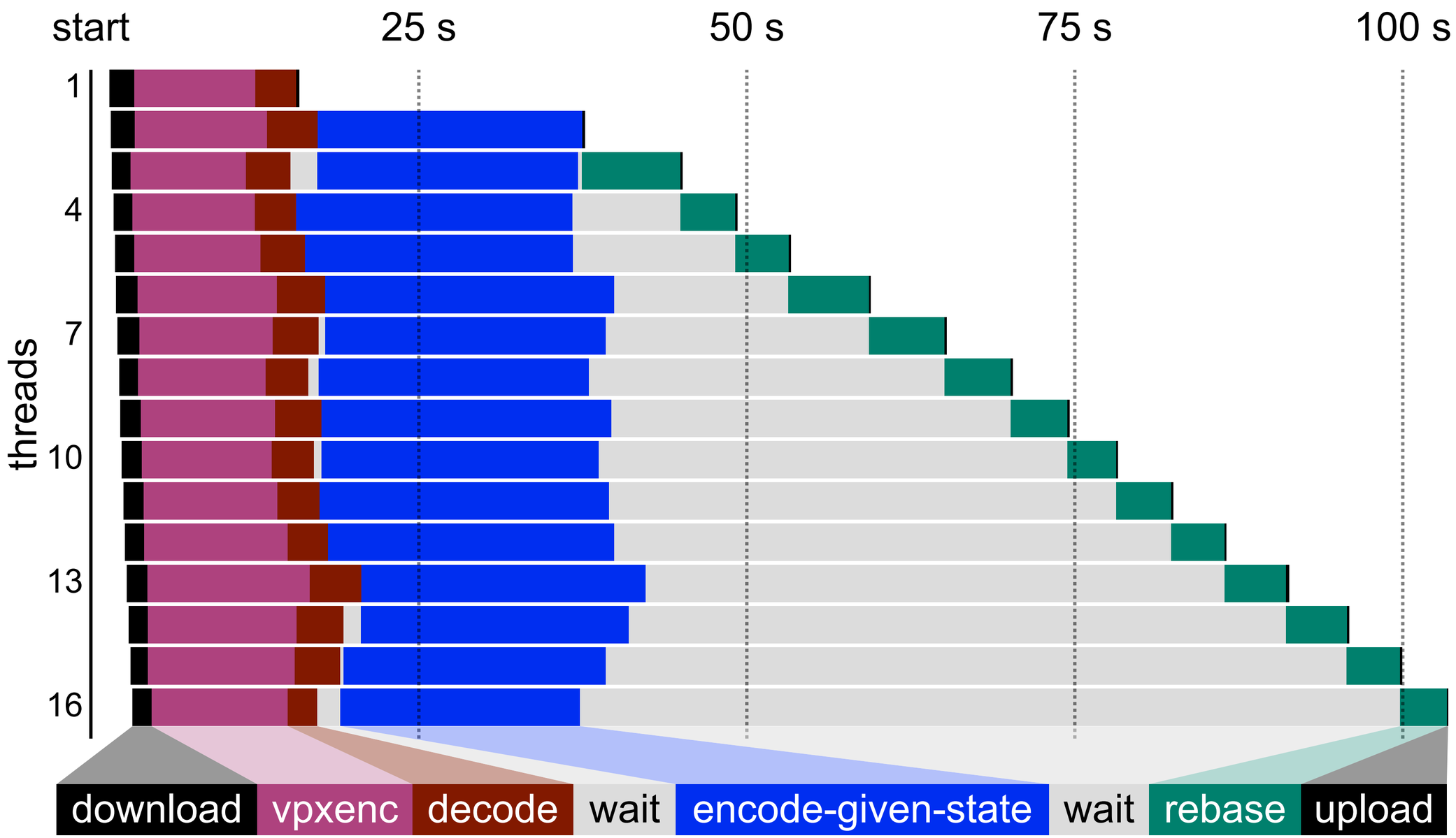
```
// prob model: tables representing encoding of values in video stream
// reference_images contains three prior images
state:= (prob_model, reference_images[3]);

// just a full image
keyframe := image pixels for entire frame

// prediction_modes and motion vectors define how to predict current
// frame given decoder state
// residue is correction to this prediction
interframe := (prediction_modes, motion_vectors, residue)

// decoding a frame generates one image of pixels, and
// an updated decoder state
decode(state, compressed_frame) -> (new_state, image)

// generate an interframe approximating image given the current
// decoder state.  This operation requires expensive motion estimation.
encode-given-state(state, image, quality_param) -> interframe

// use prediction info from interframe to estimate prediction for
// image given new state, store new residue in new_interface
// Note: rebase is cheap because it does not perform motion estimation
rebase(new_state, image, interframe) -> new_interframe
```

Massively parallel video encoding via “rebasing”

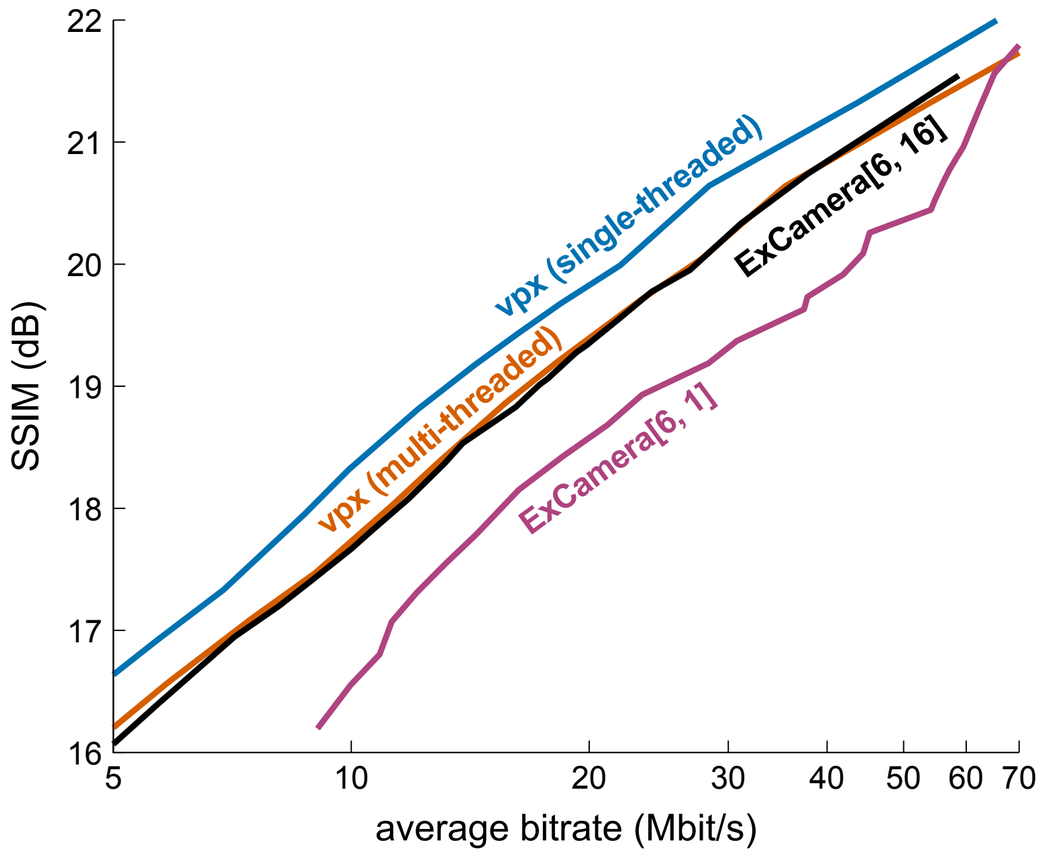
```
// N = frames per thread
// x = threads to use
ParallelEncode[N,x]

In parallel, each thread encodes N consecutive frames of video
(generates 1 keyframe + N–1 interframes)


In parallel, each thread decodes its N frames of video
Thread t passes its final decoder state to thread t+1.

In parallel, each thread executes encode-given-state() on its first frame
using the decoder state it receives, replacing what was a keyframe with an
interframe that is dependent on decoding the prior thread’s output.
(requires the original image)

In sequence, each thread rebases its remaining frames based on state after
decoding first frame.  When complete, thread t sends final decoder state
to thread t+1
```

Computation schedule for ParallelEncode[16,6]

Using 16 threads to encode 16 x 6 = 96 frames of video



Algorithm leverages wide parallelism and preserves reasonable quality/bitrate by avoiding insertion of unnecessary keyframes



ExCamera results are on 3600 cores

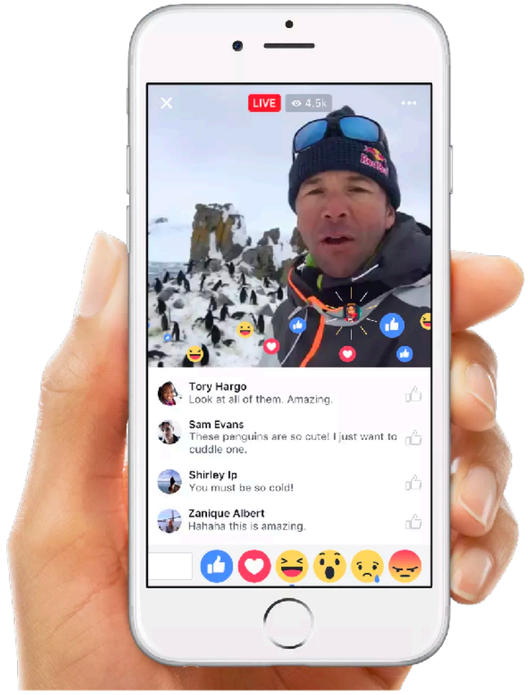
System	Bitrate at 20 dB SSIM (lower is better)	Encode time (lower is better)
ExCamera[6,16][1]	27.4 Mbps	2.6 minutes
ExCamera[6,1][2]	43.1 Mbps	0.5 minutes
`vpxenc` multi-threaded	27.2 Mbps	149 minutes
`vpxenc` single-threaded	22.0 Mbps	453 minutes

Processing uploaded videos at Facebook

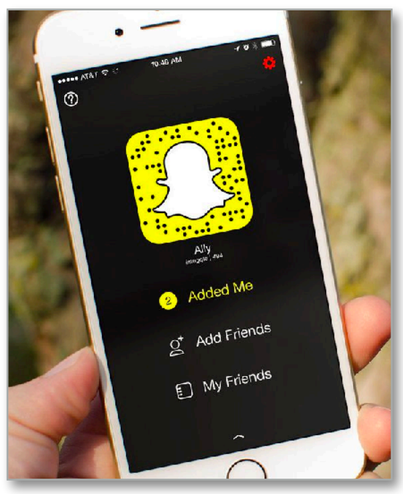
Big video data

Facebook 2016:
100 million hours of video watched per day

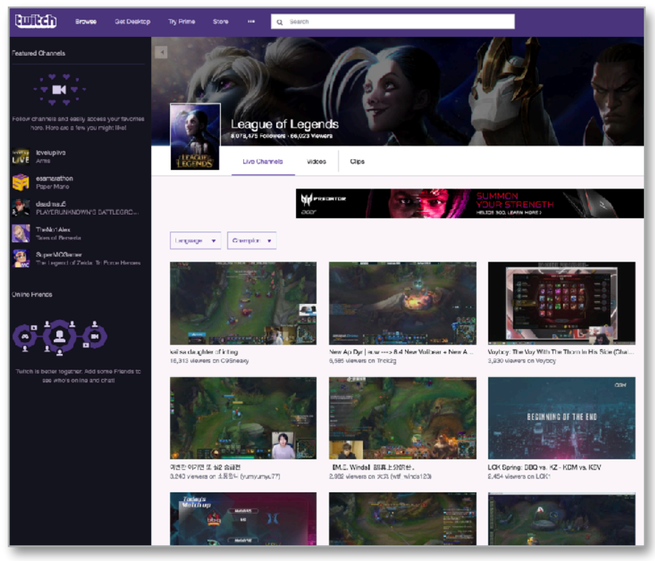
Youtube 2015: 300 hours uploaded per minute [Youtube]



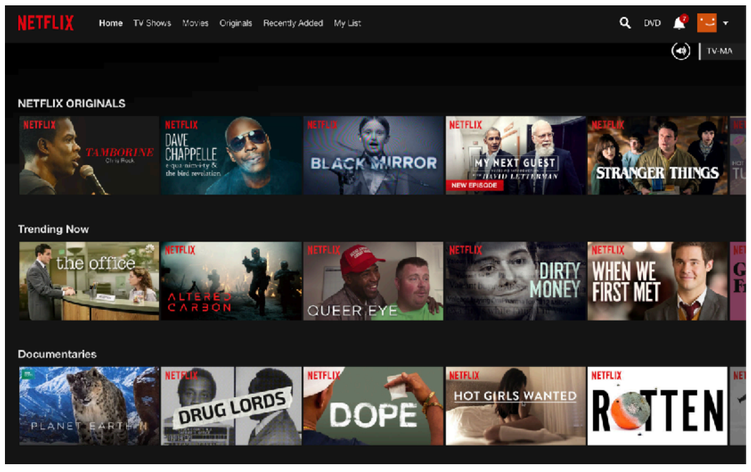
FB Live Video



Snapchat Video



Twitch



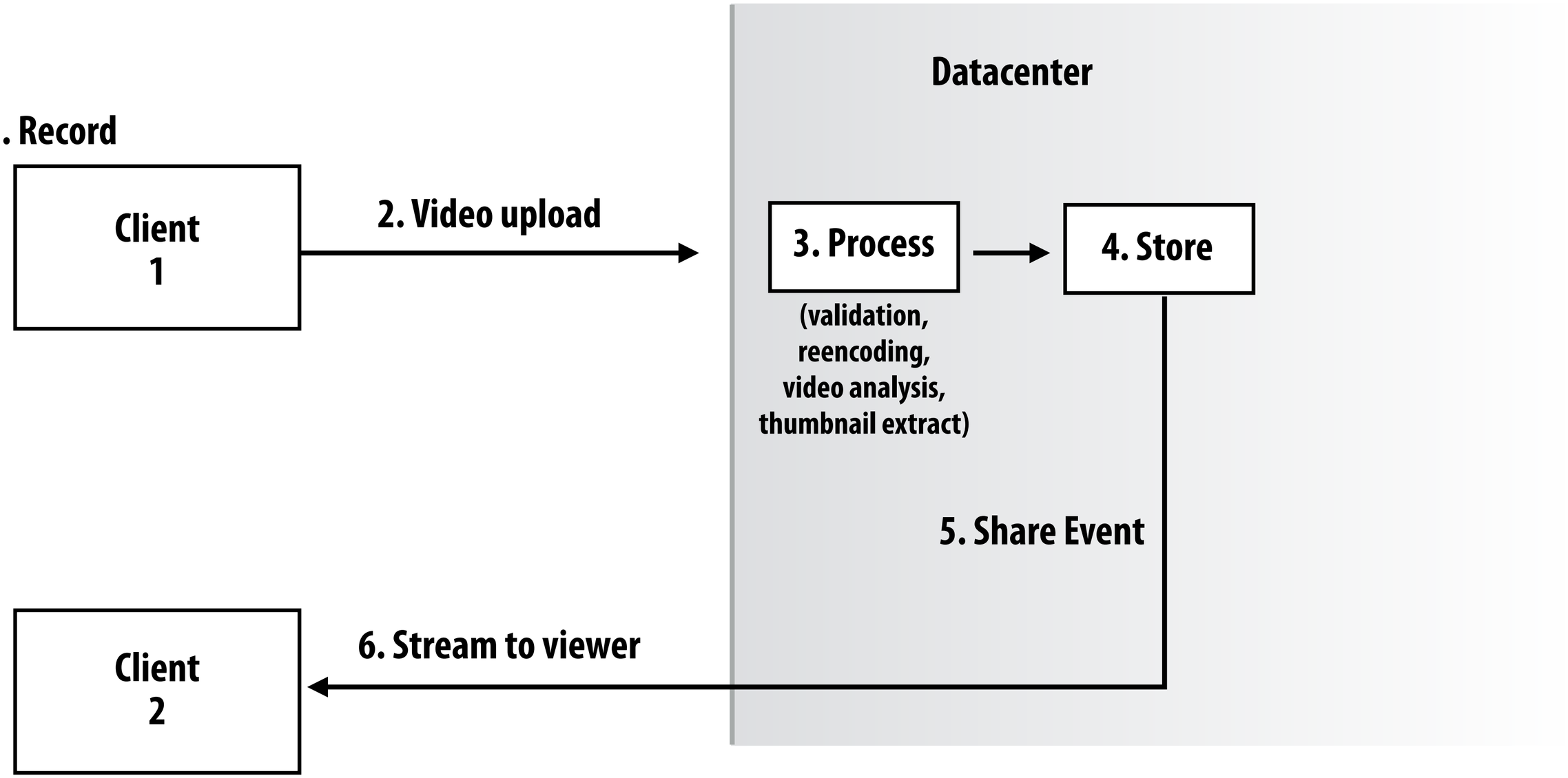
Netflix

Facebook Streaming Video Engine (SVE)

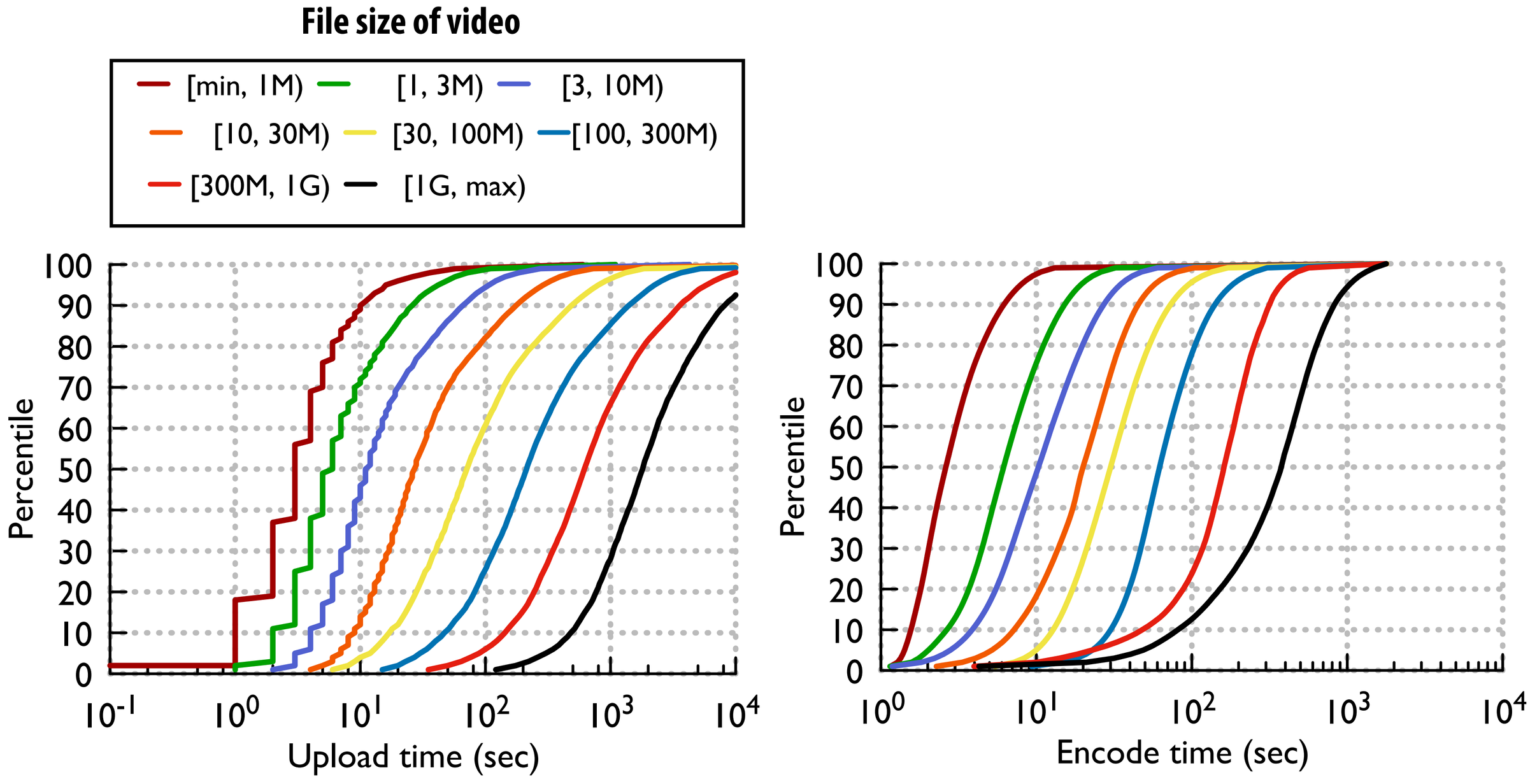
- Designed for non-streaming video upload applications (not Facebook Live)
  - Facebook video posts
  - FB Messenger video shares
  - Instagram Stories
  - 360 videos

- Goals/requirements:
  - Low latency: *minimize latency* of start of upload to sharable state
    - Particularly important for FB Messenger uploads
  - Flexible (support variety of applications such as those listed above, with different processing pipelines after upload)
  - Robust to faults and overload

Basic video sharing pipeline



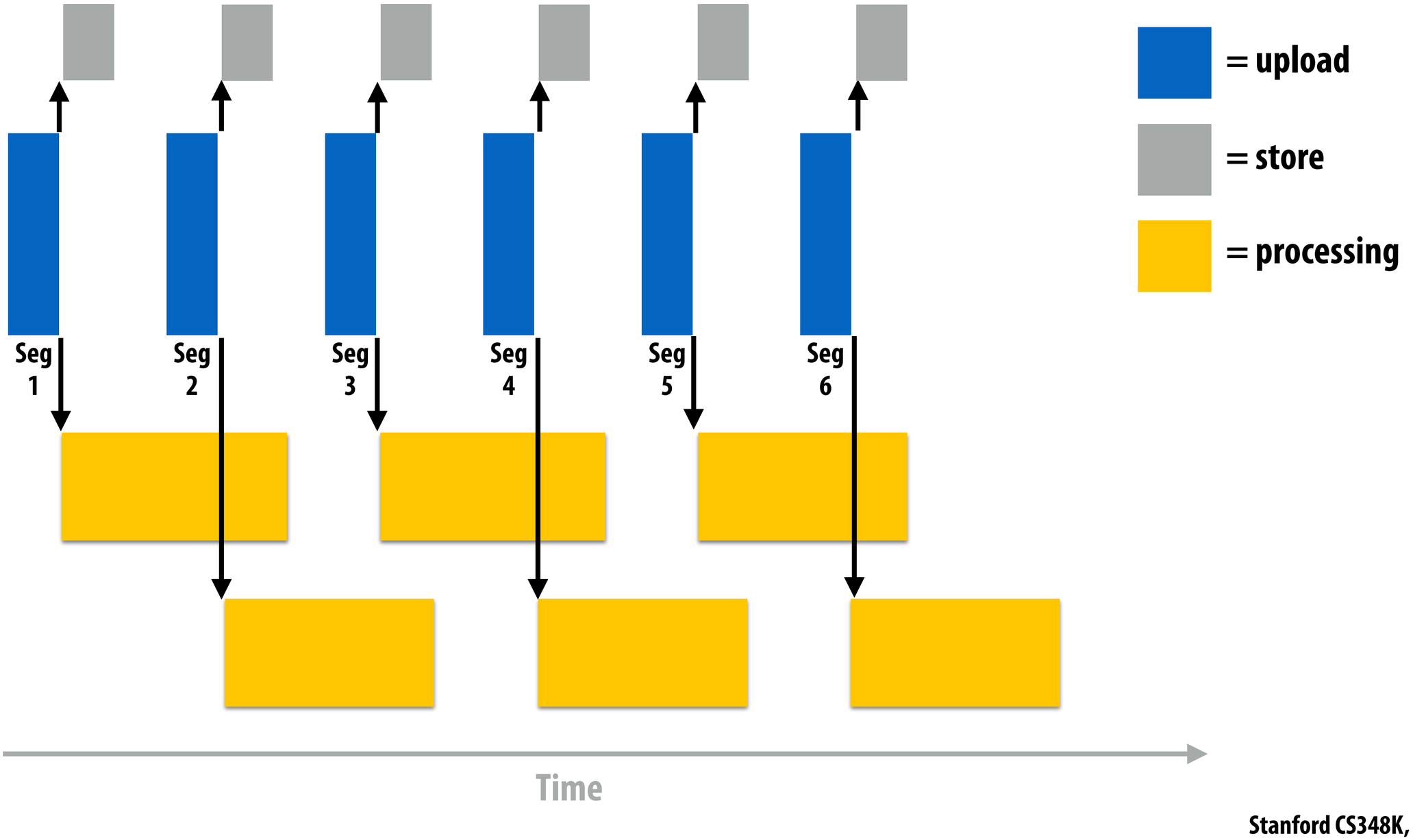
Video upload and processing times *



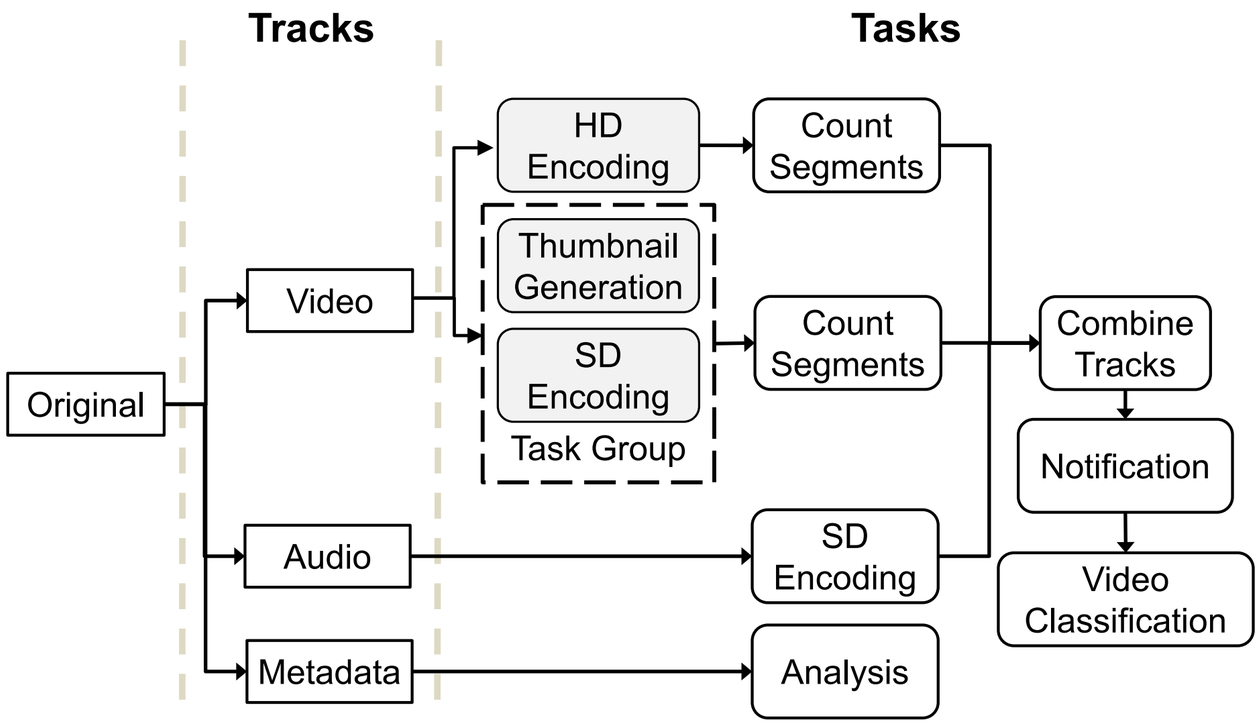
*** Serialized times (SVE system will parallelize encoding across segments as discussed in a few slides)**

Pipelining upload and processing

- Client application partitions video into segments prior to upload
- Client application optionally downsamples video (skipped if video recorded at low enough resolution, internet connection is fast, or device does not support HW accelerated encode)
- Upload and processing of video is pipelined (upload and processing is mostly parallelized)
- Processing itself can be parallelized across segments



DAG representation of processing



Simple DAG:
Encodes HD and SD version of uploaded video

DAG node = "task"
Each task is executed serially on one video segment
Overall DAG execution can be parallelized (across tracks and segments)

Facebook Video Posts: ~153 tasks
Messenger shares: 18 tasks
Instagram stories: 22 tasks

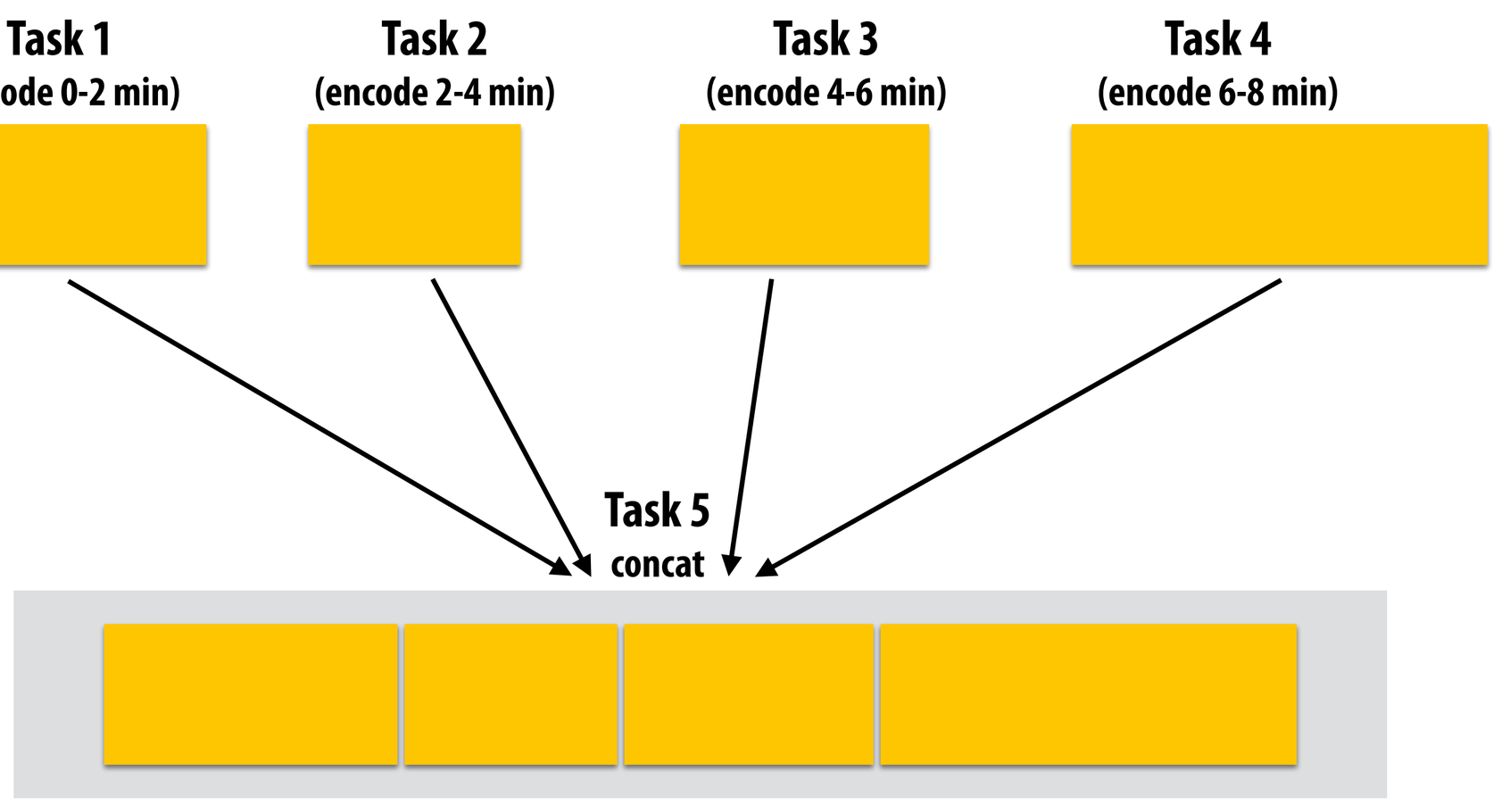
DAG Specification in Python:

Nodes defined on audio, video, metadata tracks:

```
pipeline = create_pipeline(video)

video_track = pipeline.create_video_track()
if video.should_encode_hd
  hd_video = video_track.add(hd_encoding)
   .add(count_segments)
sd_video = video_track.add(
  {sd_encoding, thumbnail_generation},
 ).add(count_segments)

audio_track = pipeline.create_audio_track()
sd_audio = audio_track.add(sd_encoding)

meta_track =
  pipeline.create_metadata_track()
  .add(analysis)

pipeline.sync_point(
  {hd_video, sd_video, sd_audio},
  combine_tracks,
 ).add(notify, 'latency_sensitive')
  .add(video_classification)
```

Coarse-grained parallel video encoding

- Parallelized across segments (I-frame inserted at start of segment)
- Concatenate independently encoded bitstreams



Smaller segments = more potential parallelism, worse video compression
Latency-sensitive applications: 10 second segments
Non-latency sensitive, long videos: 2 minute segments (maximize compression)

Overload control

- When FB cannot keep up with the world's video upload rate…

- Delay latency-insensitive tasks
- Rebalance load: redirect uploads to new datacenter region
- Delay processing of new uploads

Discussion:

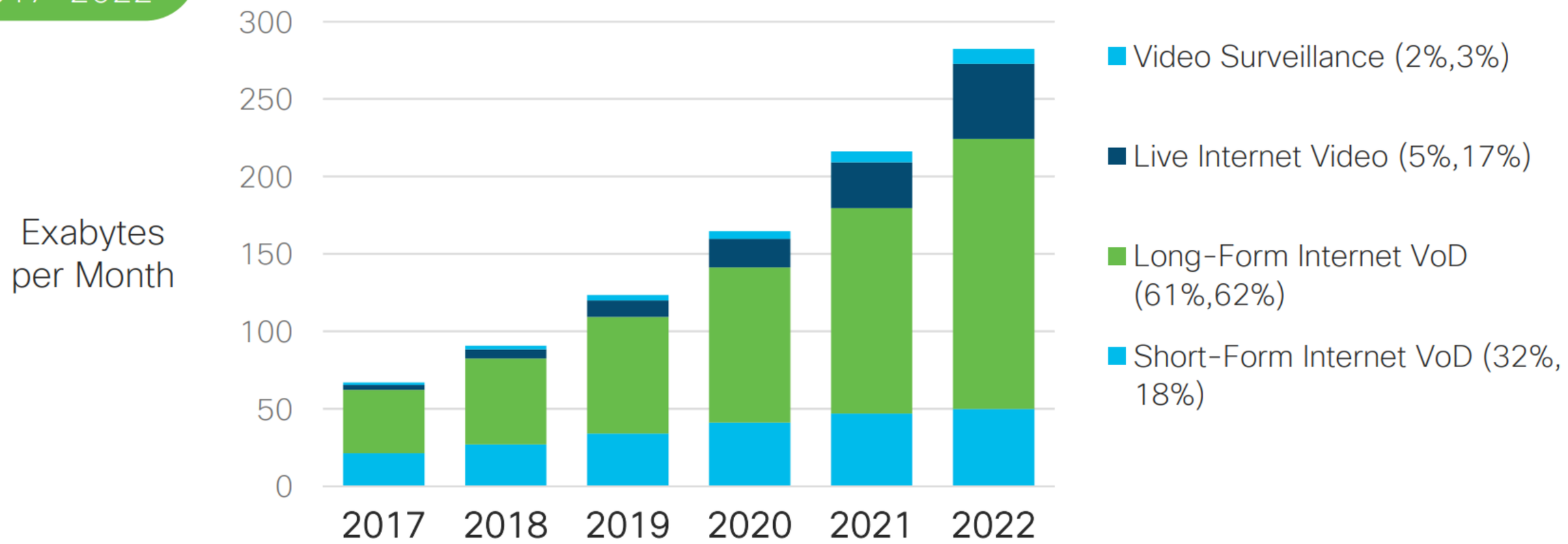
When should the cameras always be on?

Video internet usage (from last time)

Global Internet Video Traffic by Type

By 2022, live video will increase 15-fold and reach 17% of Internet video traffic

33% CAGR
2017-2022

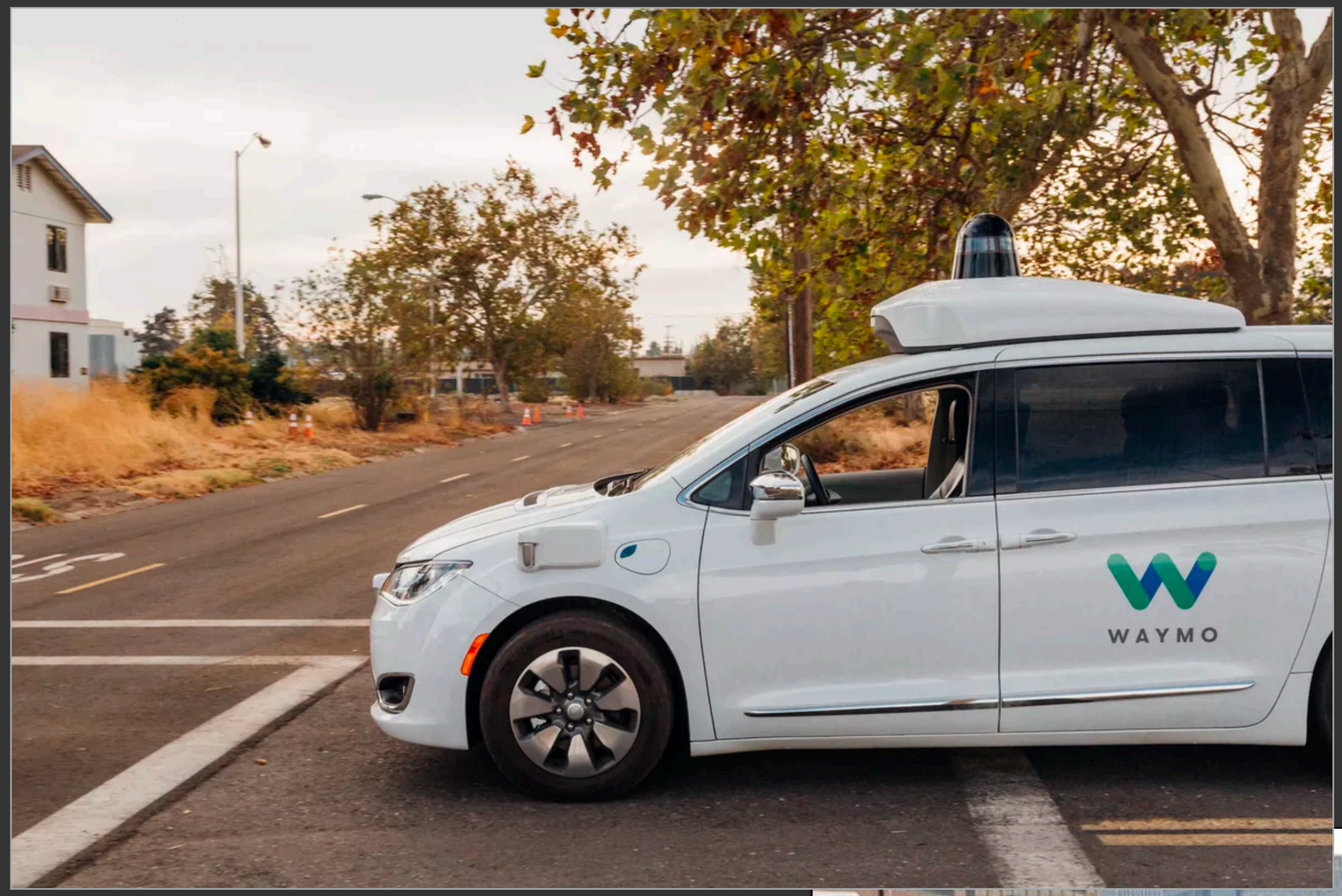


* Figures (n) refer to 2017, 2022 traffic share

Source: Cisco VNI Global IP Traffic Forecast, 2017-2022

Keep in mind, at the moment this is a small fraction of the video we transmit.

Analyzing images for robot navigation



Analyzing images for urban efficiency

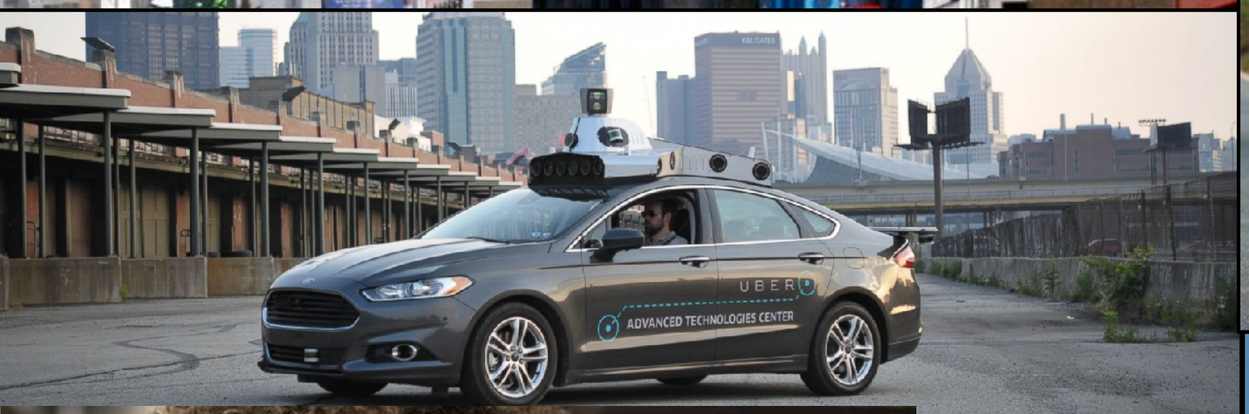
“Managing urban areas has become one of the most important development challenges of the 21st century. Our success or failure in building sustainable cities will be a major factor in the success of the post-2015 UN development agenda.”

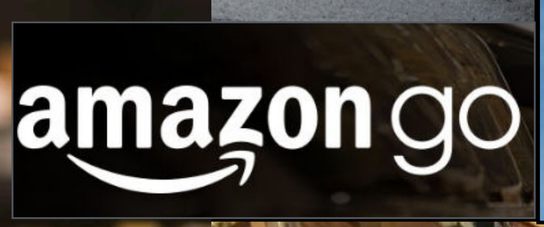
- UN Dept. of Economic and Social Affairs

Analyzing egocentric images to augment humans

Gwangjiang Market (Seoul)


DISCOVER CARD
TDK
Twizzlers


UBER
ADVANCED TECHNOLOGIES CENTER


amazon go


ovement

Some recent examples

- Comprehensive capture of athlete performance

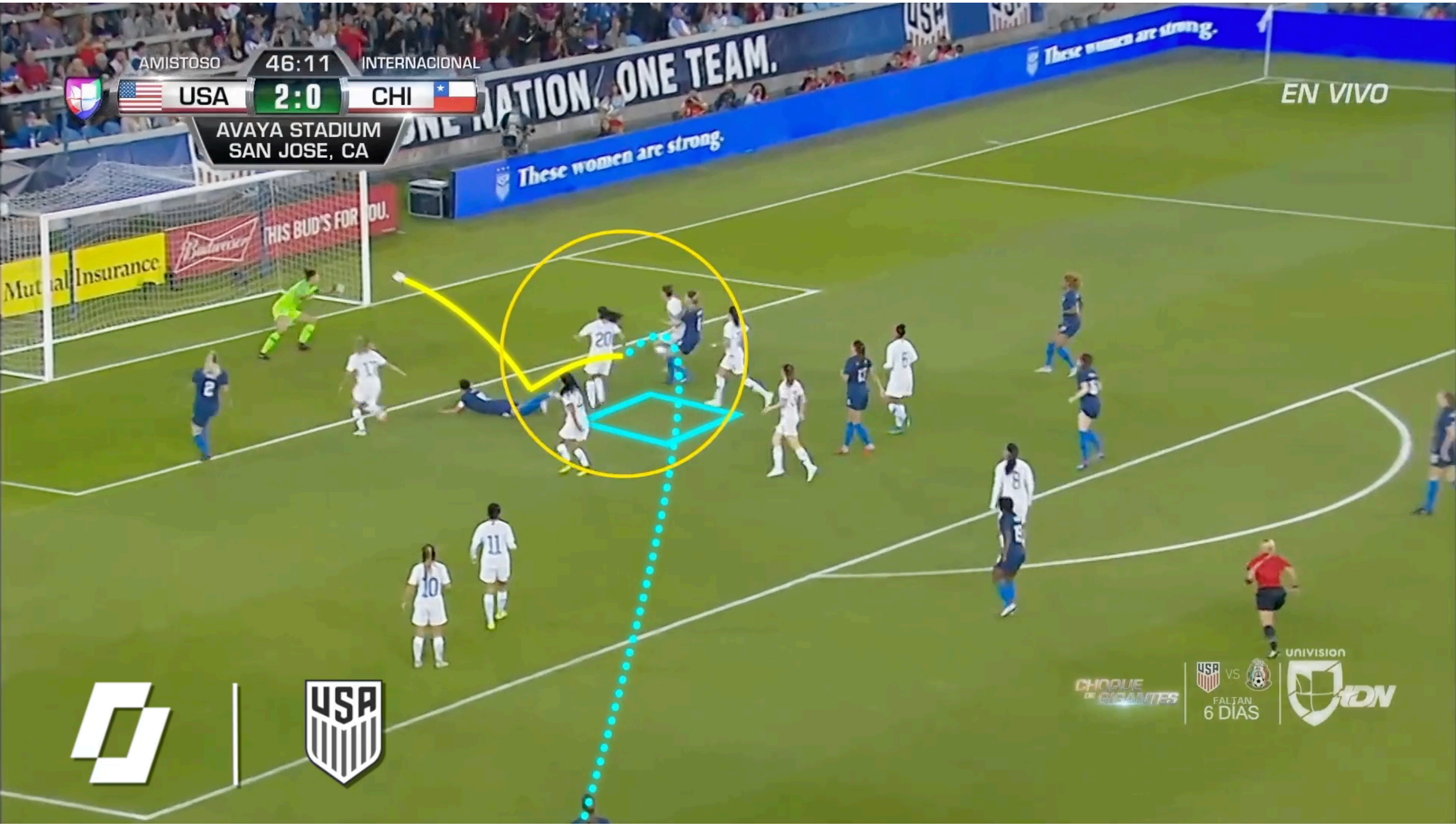


Image credit: Second Spectrum

Some recent examples

- Comprehensive capture of athlete performance

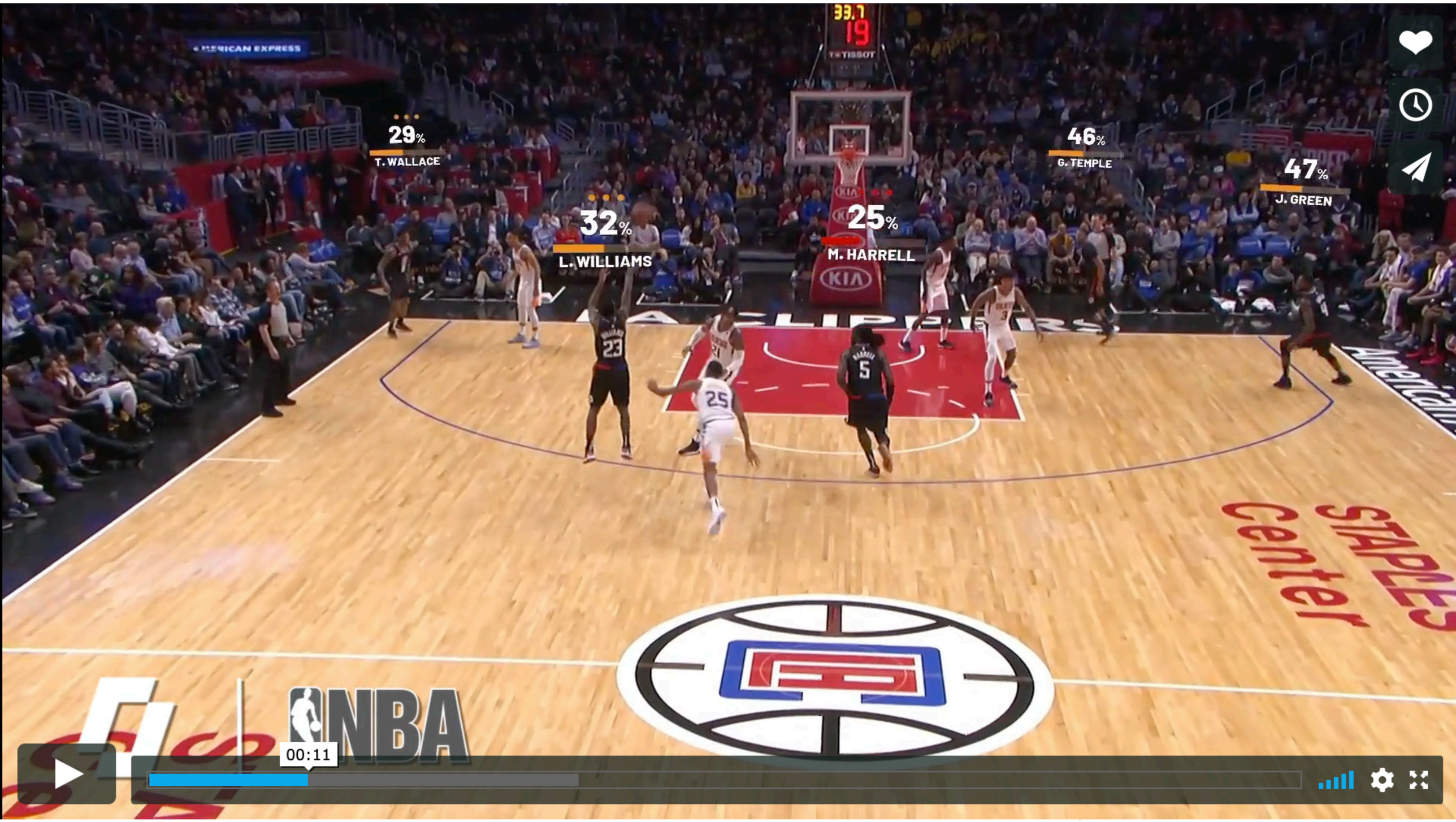


Image credit: Second Spectrum

Comprehensive capture of worker performance?

If Workers Slack Off, the Wristband Will Know. (And Amazon Has a Patent for It.)

By Ceylan Yeginsu

Feb. 1, 2018

118

Leer en español

LONDON — What if your employer made you wear a wristband that tracked your every move, and that even nudged you via vibrations when it judged that you were doing something wrong?

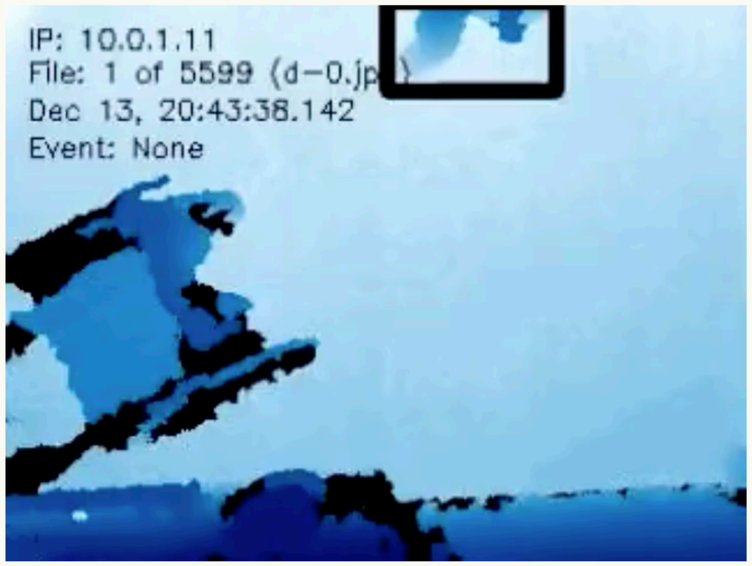
What if your supervisor could identify every time you paused to scratch or fidget, and for how long you took a bathroom break?

Image credit: Second Spectrum

Some recent examples

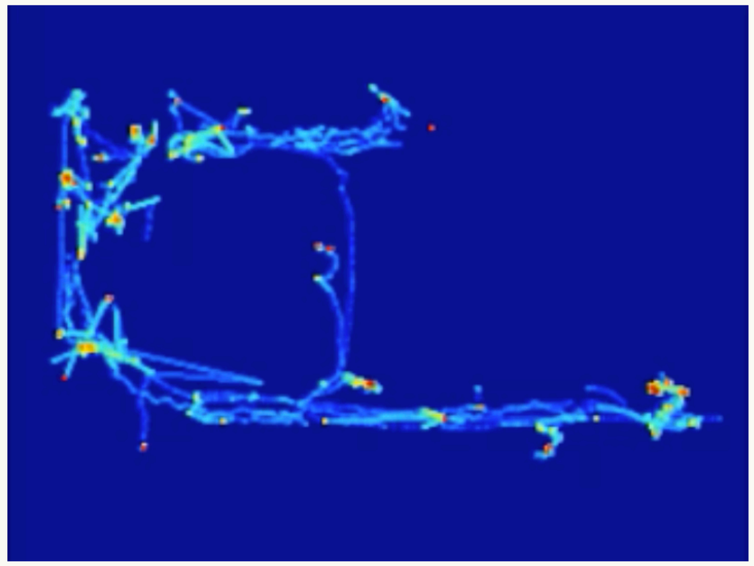
- **Surveillance of hospital workers (hand washing)**

Dispenser Usage Detection



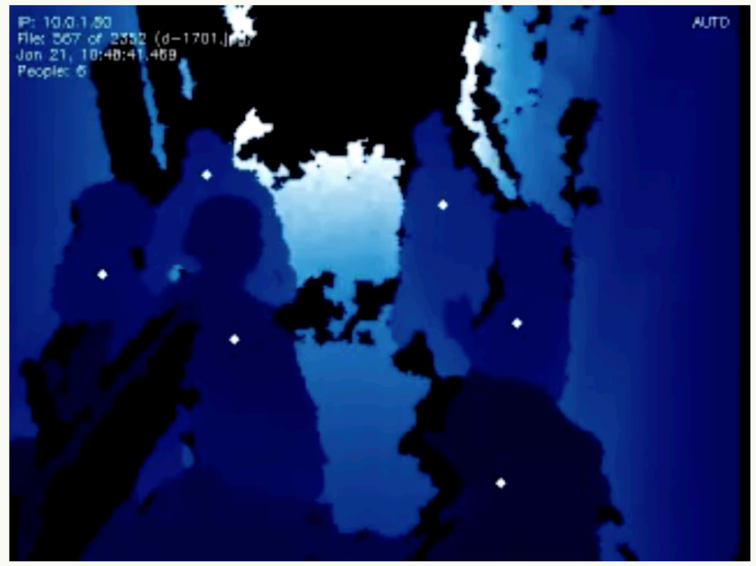
With the help of artificial neural networks, our method uses deep learning to automatically detect usage of an alcohol-based sanitizer dispenser from challenging ceiling-mounted top views.

Physical Space Analytics



Intuitive, qualitative results analyze human movement patterns and conduct spatial analytics which convey our method's interpretability. Red regions denote high traffic areas while blue denotes low traffic regions.

Privacy Safe Assessment



To comply with privacy regulations, we use de-identified depth images instead of color photos to track and analyze hand hygiene compliance. Our method can track multiple clinicians throughout a hospital ward.

Towards Vision-Based Smart Hospitals: A System for Tracking and Monitoring Hand Hygiene Compliance
Haque et al. 2017

Surveillance for contact tracing

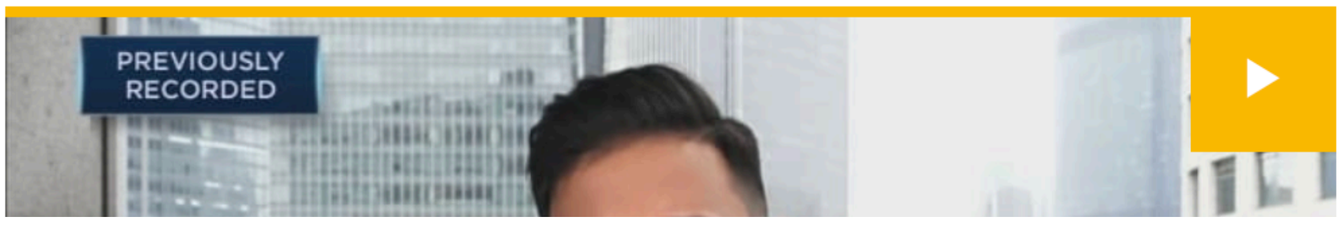
Use of surveillance to fight coronavirus raises concerns about government power after pandemic ends

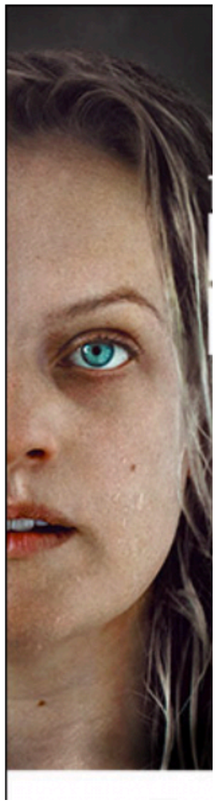
PUBLISHED THU, MAR 26 2020•7:58 PM EDT | UPDATED MON, MAR 30 2020•12:17 PM EDT

Arjun Kharpal

KEY POINTS

- China mobilized its mass surveillance tools, from drones to CCTV cameras, to monitor quarantined people and track the spread of the coronavirus.
- Other nations like Israel, Singapore and South Korea are also using a combination of location data, video camera footage and credit card information, to track COVID-19 in their countries.
- But privacy experts raised concerns about how governments were using the data, how it was being stored and the potential for authorities to maintain heightened levels of surveillance — even after the coronavirus pandemic is over.





What are your standards for when observational technology is reasonable to be deployed?

What safeguards (both technical and non-technical) should be put in place to protect privacy?