Towards an Exapixel per Second:

Enabling Efficient Visual Data Analysis at Scale

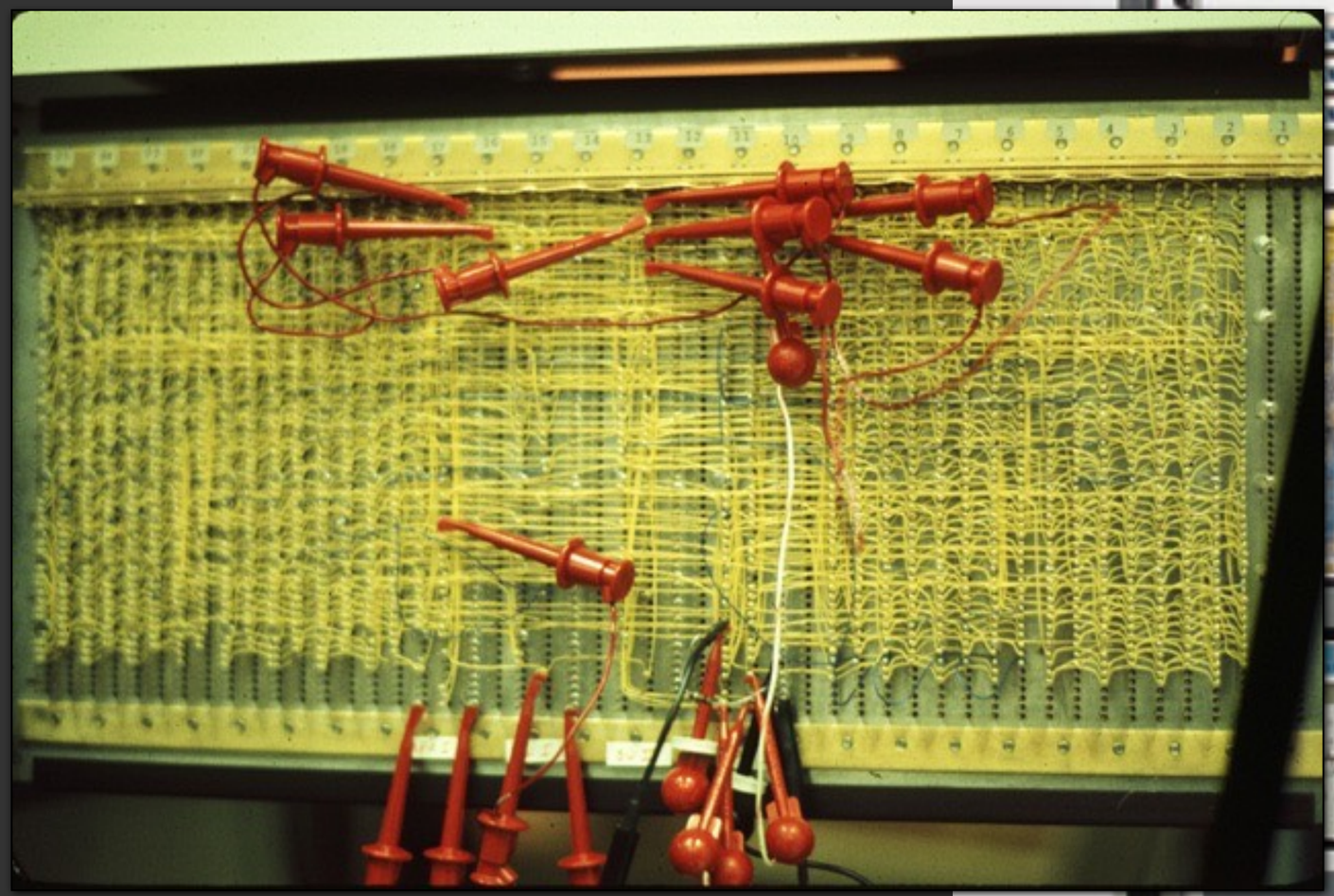
Kayvon Fatahalian
Carnegie Mellon University

(After six years at CMU I will be moving to Stanford in Sept 2017)

Ivan Sutherland's Sketchpad on MIT TX-2 (1962)

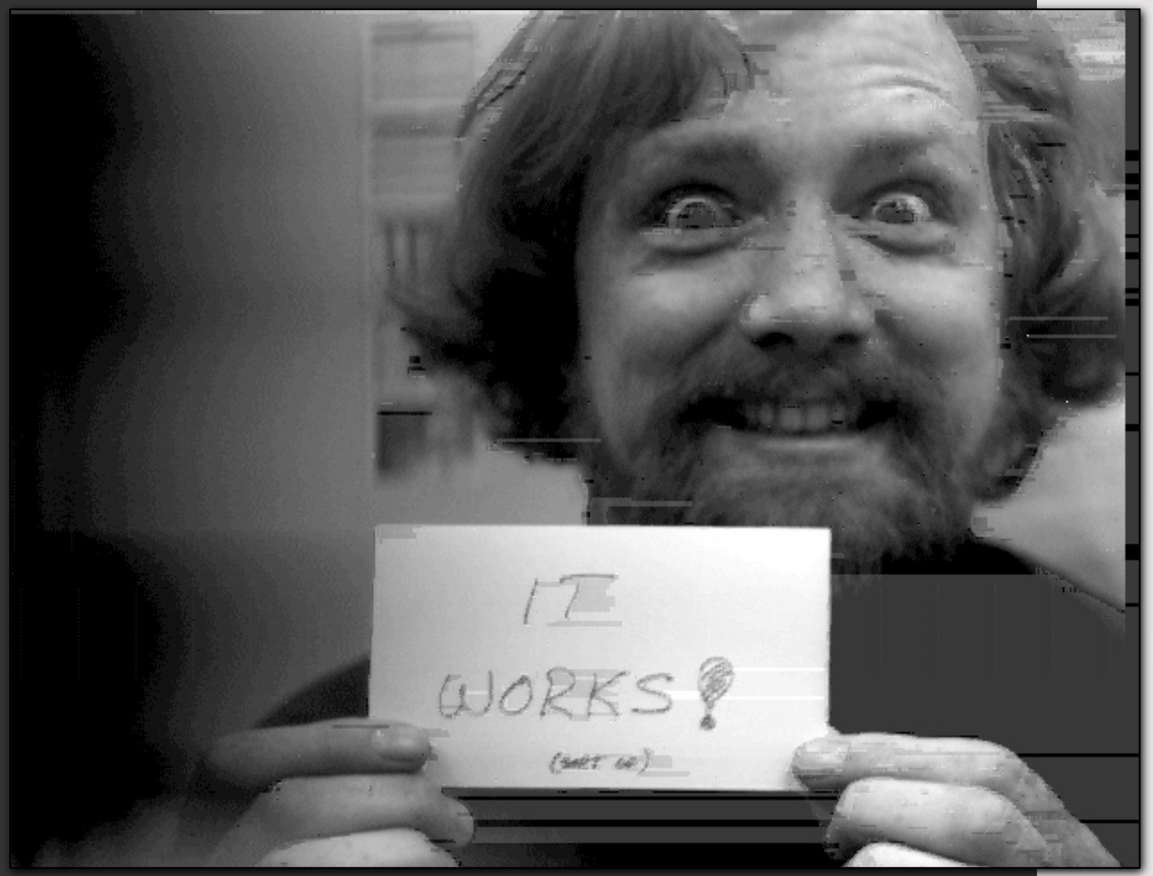
The frame buffer

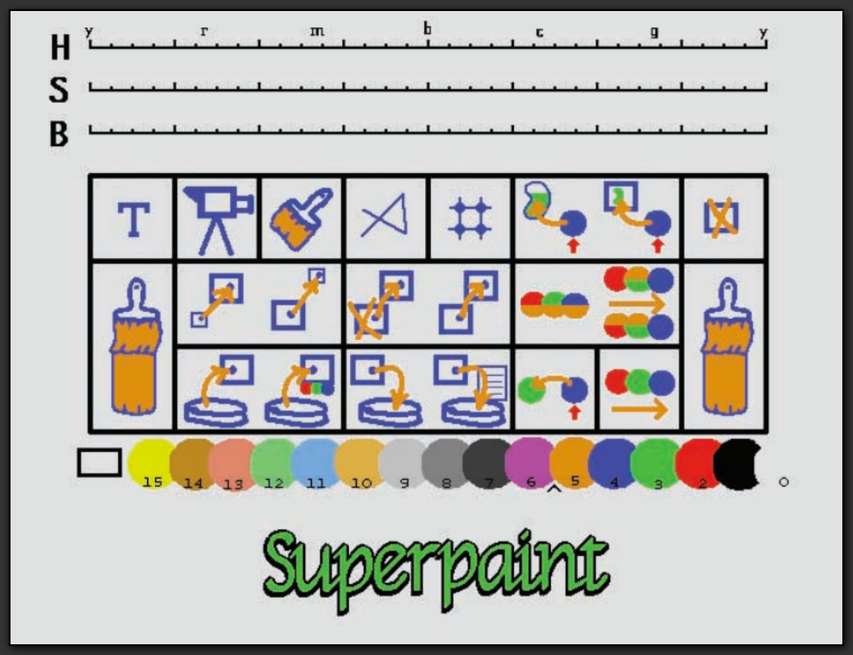
Shoup's SuperPaint (PARC 1972-73)



The frame buffer

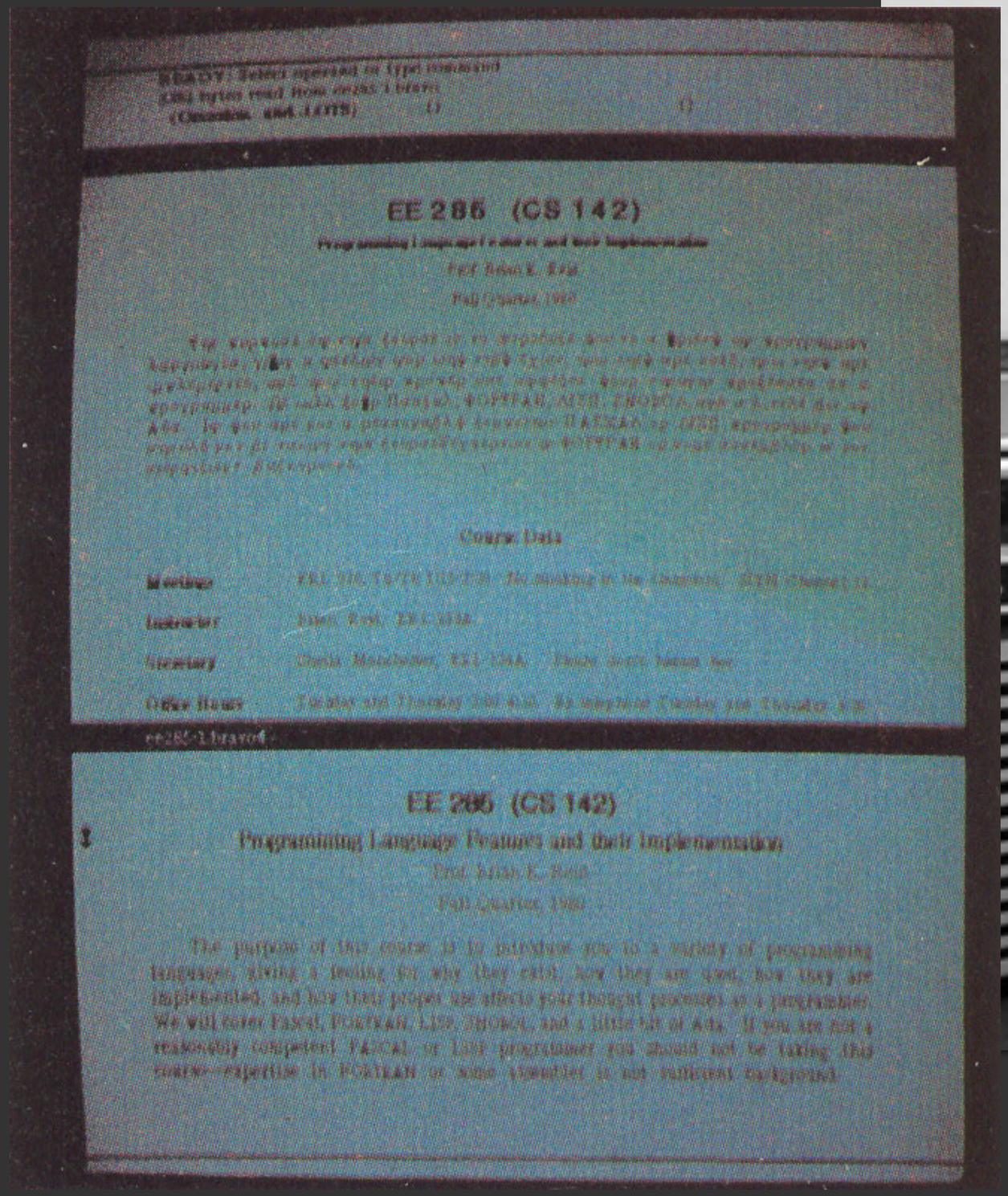
Shoup's SuperPaint (PARC 1972-73)





16 2K shift registers (640 x 486 x 8 bits)

Xerox Alto (1973)



EE 286 (CS 142)

Programming Language Features and their Implementation

Prof. Brian K. Reid

Fall Quarter, 1982

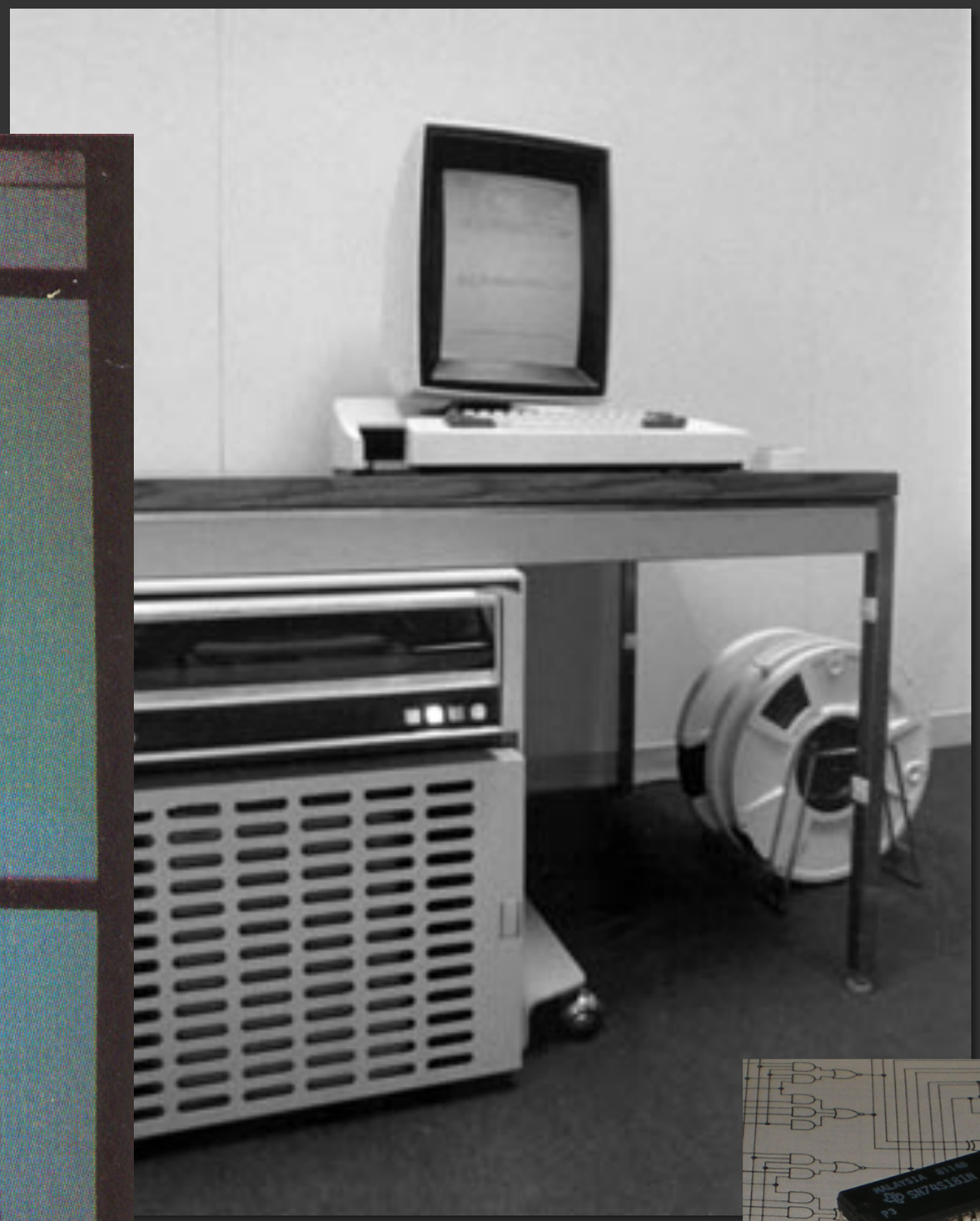
EE 286 (CS 142)

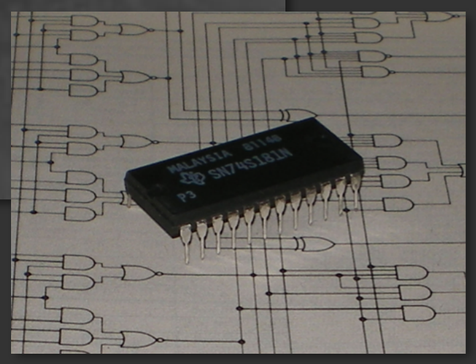
Programming Language Features and their Implementation

Prof. Brian K. Reid

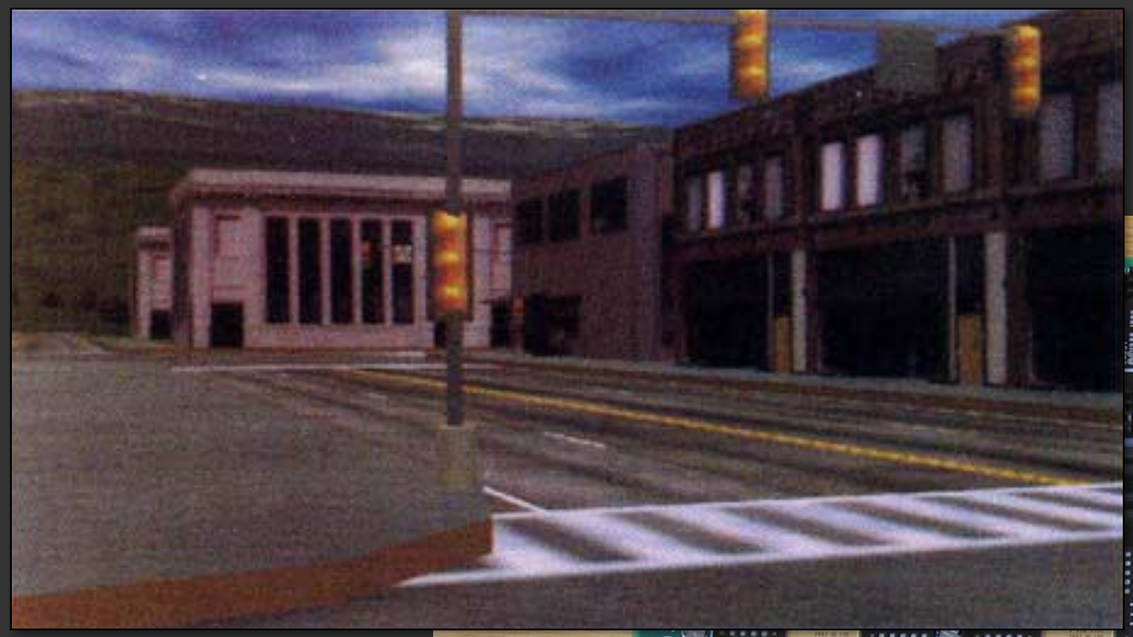
Fall Quarter, 1982

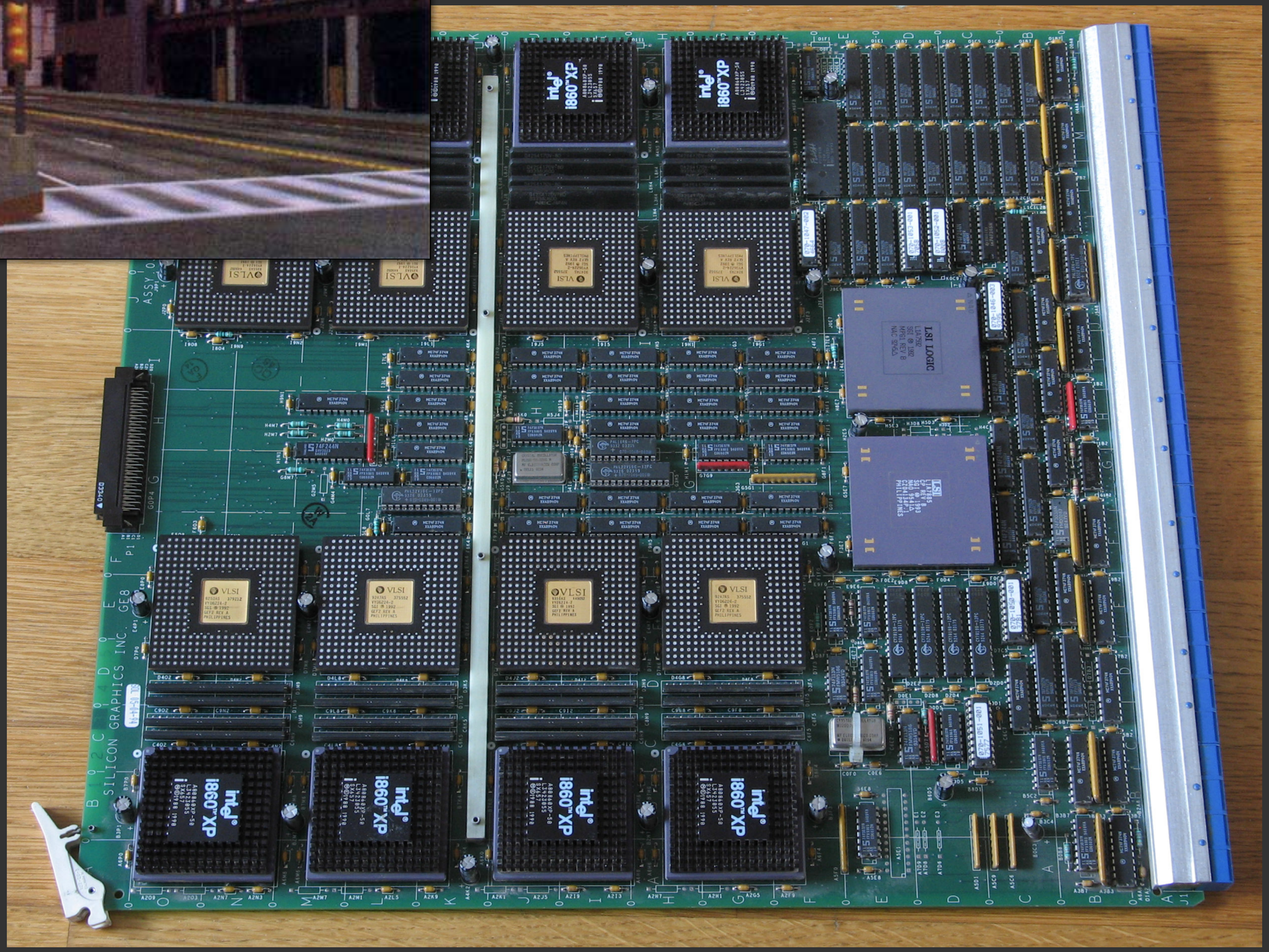
Bravo (WYSIWYG)





TI 74181 ALU





SGI RealityEngine GE8 board (1993)

Real-time (30 fps) on a NVIDIA Titan X

Unreal Engine Kite Demo (Epic Games 2015)

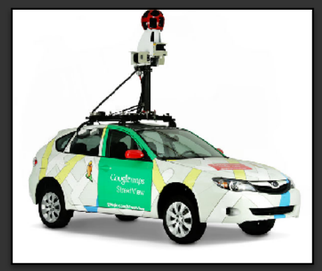
2B shares per day across Facebook sites (includes Instagram+WhatsApp) [FB2015]

Youtube 2015: 300 hours per minute uploaded [Youtube]

80-90% of 2019 internet traffic will be video [Cisco VNI]

Ubiquitous image sensing and analysis

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 images for urban efficiency

Use of image analysis to identify:

Dangerous intersections

Infrastructure needing repair (Pittsburgh potholes!)

Flooding / ice

Air-quality monitoring

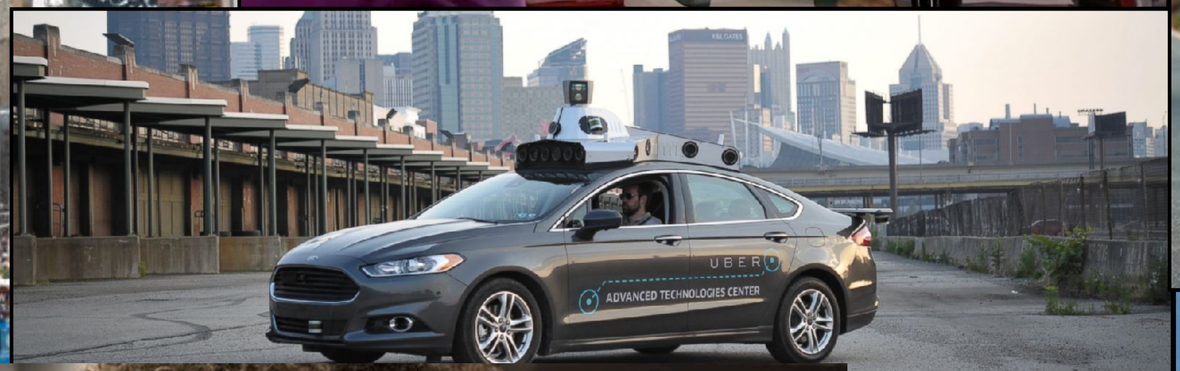
…

Analyzing egocentric images to augment humans

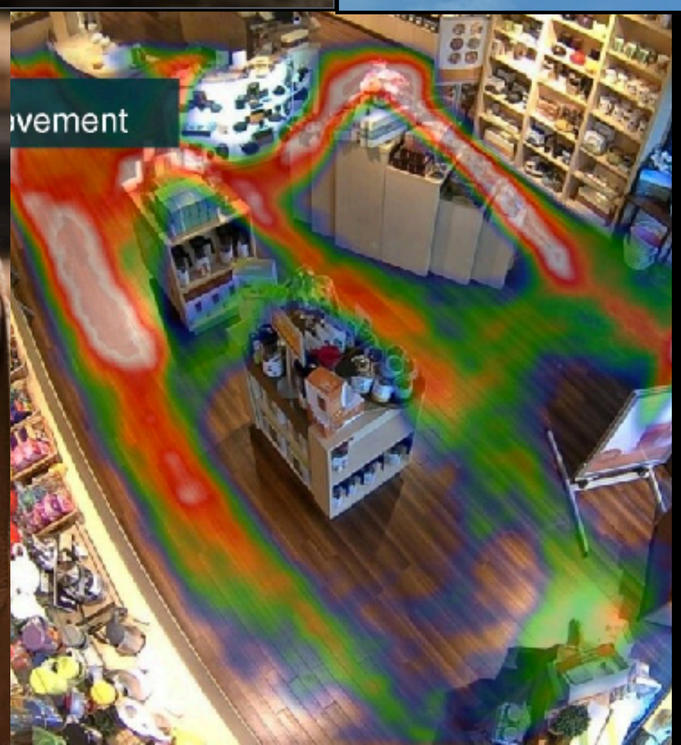
Gwangjiang Market (Seoul)

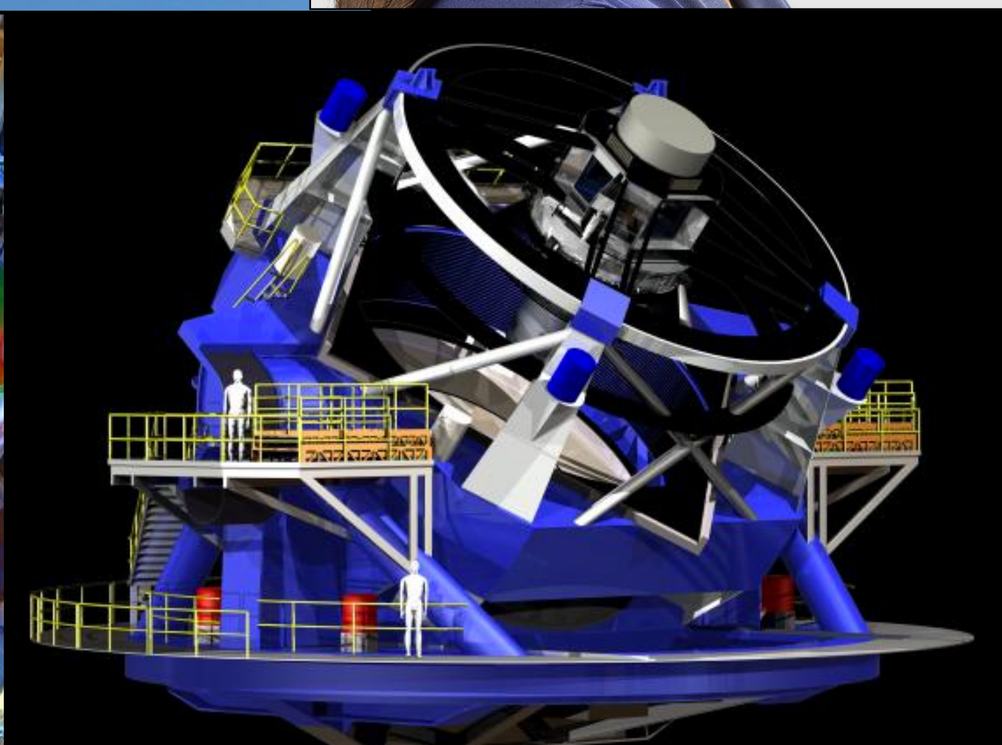

DISCOVER CARD
TDK


MEET THE PRESS
CARSON CONTROVERSY
PAST CLAIMS UNDER THE MICROSCOPE
MEET THE PRESS


UBER
ADVANCED TECHNOLOGIES CENTER


amazon go


ovement



The visual data world in 2030

8.5 billion people (61% urban)
[UN estimate]

70% smartphone penetration
[Statista, 50% in 2020]

25% turned on → **1.5B**

---

2 billion cars
[Sperling 2009]

8 cameras/car

25% load → **4B**
[currently 2%]

1.1B streaming security cameras
Extrapolation from 245M in 2014, 10% annual growth [IHS]

Assume 8K video resolution (33 megapixel)

Total capture capability across the world

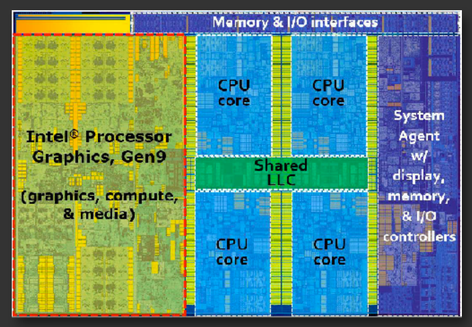
~6.5B video streams = 2.1 x 10^{17} pixels x 30 fps = 6 exapixels/sec

Other considerations: home health care robotics, survey science, infrastructure monitoring…

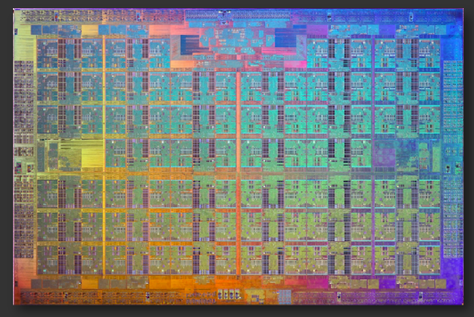
How do we architect efficient (and easy to program) systems for analyzing the worldwide visual signal?

Challenge: compute-intensive, pixel processing algorithms

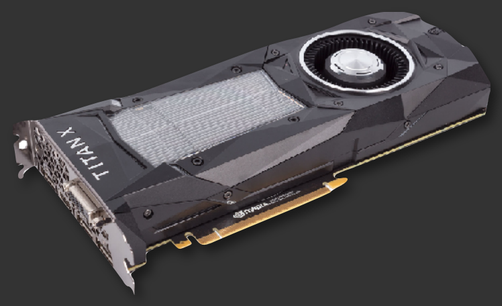
Need: Efficiently map image analysis algorithms to (specialized) accelerated computing platforms (*"use every op you can get"*)



Integrated GPU + media ASIC

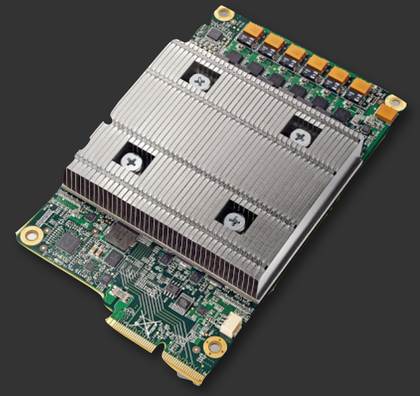


Xeon Phi



GPU

FPGA



TPU (ML ASIC)

Challenge: large scale of visual data to acquire, store, and analyze

Need: distributed computing platforms for productive use of heterogeneous, accelerated computing hardware at datacenter scale

Challenge: brute-force nature of many widely used techniques

Need: performance-centric algorithmic innovation

How can systems automatically approximate programs by eliminating redundancy, using intelligent filtering, inducing sparsity, adaptive techniques, etc.?

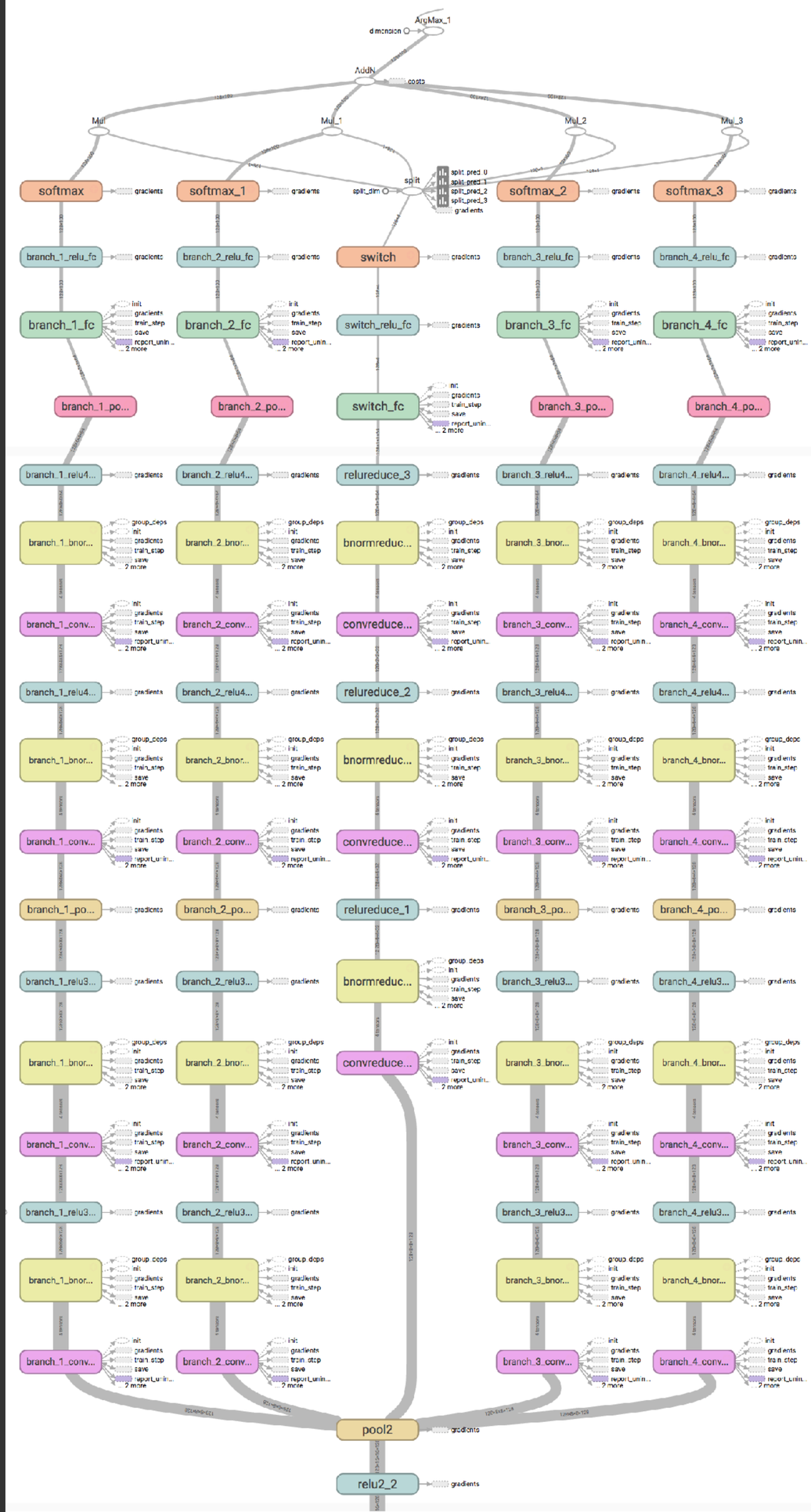
	ImageNet Top-1 Accuracy	Num Params	Cost/image (MADDs)	
VGG-16	71.5%	138M	15B	[2014]
GoogleNet	70%	6.8M	1.5B	[2015]
ResNet-18	73% *	11.7M	1.8B	[2016]
MobileNet-224	70.5%	4.2M	0.6B	[2017]

* 10-crop results (ResNet 1-crop results are similar to other DNNs in this table)

Challenge: authoring complex image analysis applications

How can frameworks encourage desirable program structure: modules, interfaces, etc. in the context of end-to-end optimization

What are primitives for analyzing databases of images, videos, or analyzing scenes? ("SQL for visual computing?")



Big visual computing systems needs

1. **Techniques for efficiently mapping image analysis algorithms to accelerated computing platforms**
   (Efficiently generating kernels for CPUs, GPUs, FPGAs, ASICs)

2. **Distributed computing support for scalable accelerated computing**
   (Connecting efficient processing pipelines to data stores, distribution across many machines)

3. **Performance-centric algorithmic innovation/approximation**
   (New work efficient algorithms and approximations)

4. **Good abstractions for authoring scalable visual data analysis applications**
   (Considering higher-level primitives for authoring future applications e.g., SQL for video DBs?)

Motivating question

If I wanted to grab a few terabytes of video, store it in a database, and perform pixel-level analyses on frames from the collection using a cluster of high-compute-density nodes, what system should I use?

“KrishnaCam” egocentric video dataset

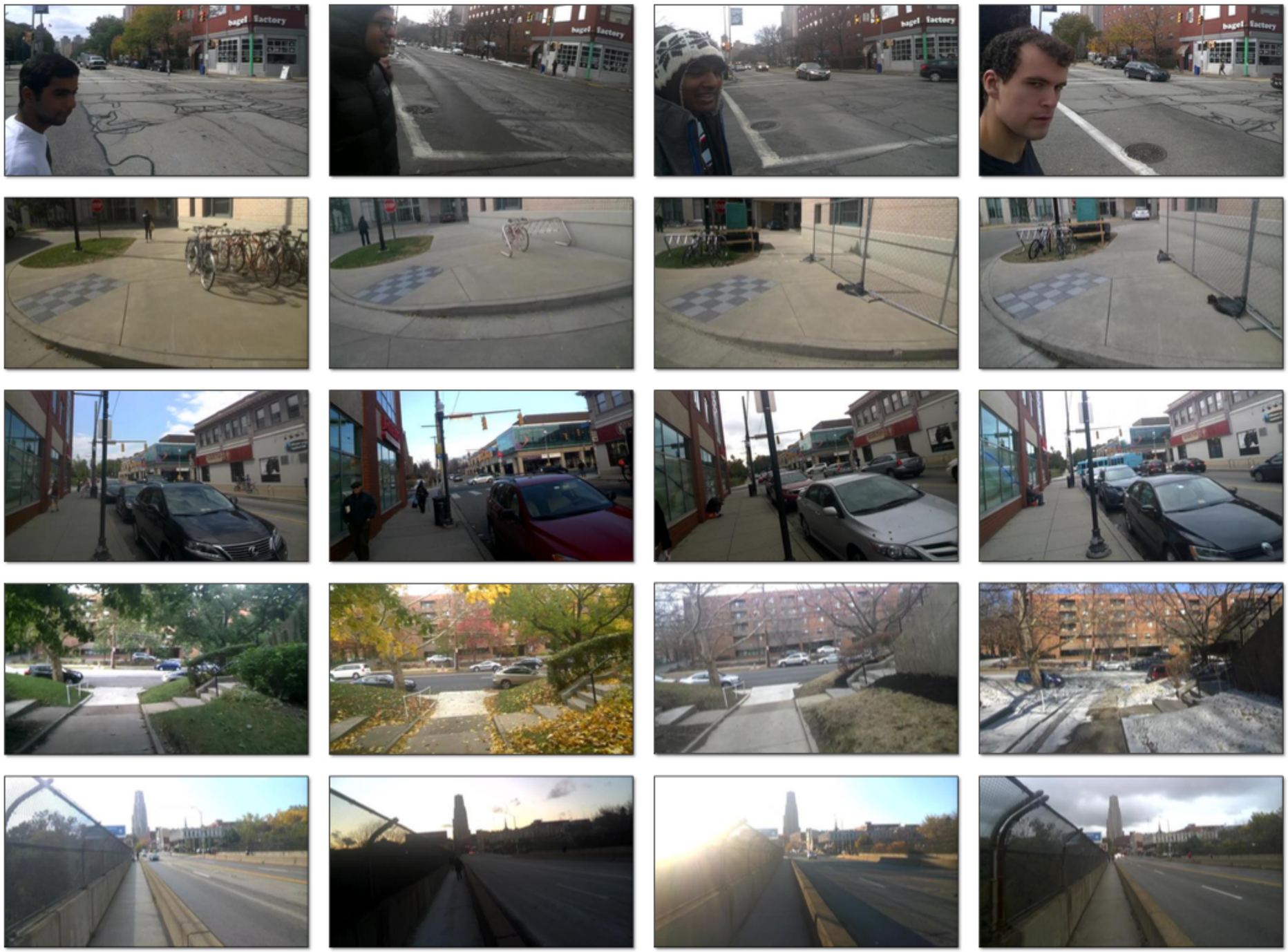
[Singh 2016]

72 hours of recording over nine months: (Sep 2014 – May 2015)

Google Glass

[Singh 2016]

How does the world evolve?



1. Change in companion

2. Change in object location (bike rack moved)

3. Change in object (different parked cars)

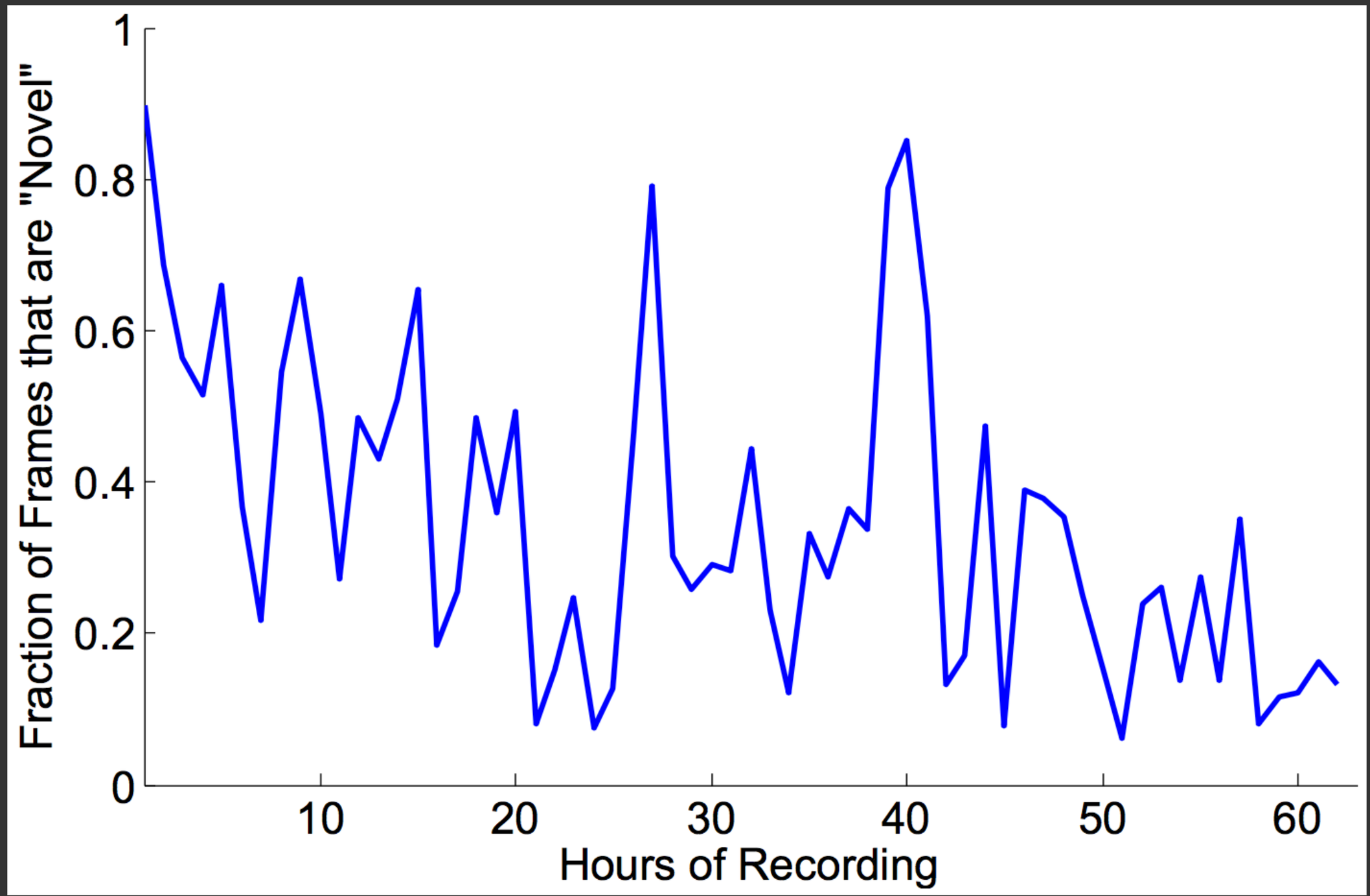
4. Change in season

5. Change in time of day (lighting conditions)

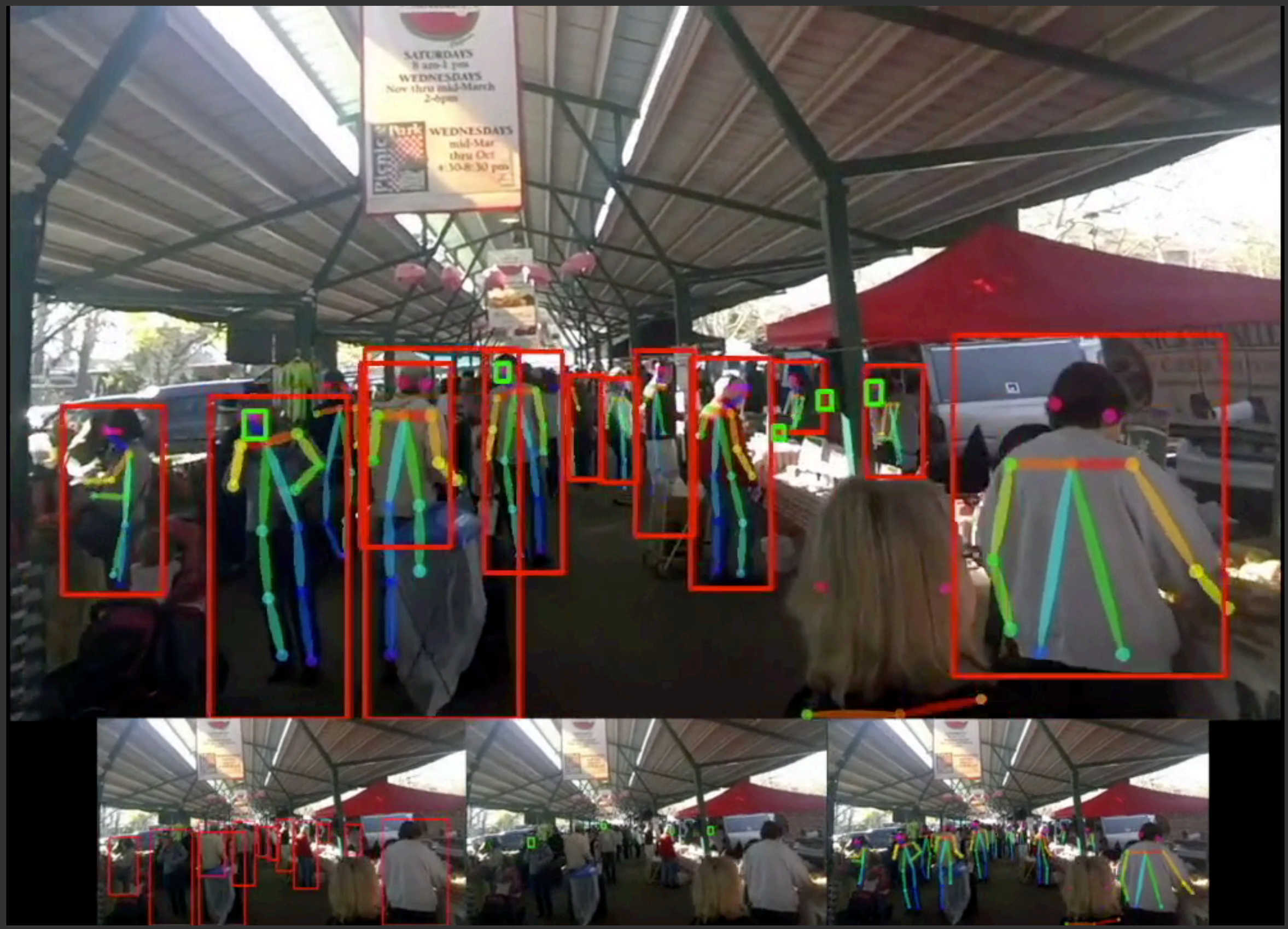
[Singh 2016]

Life-specific data compression: KrishnaCam novel data growth

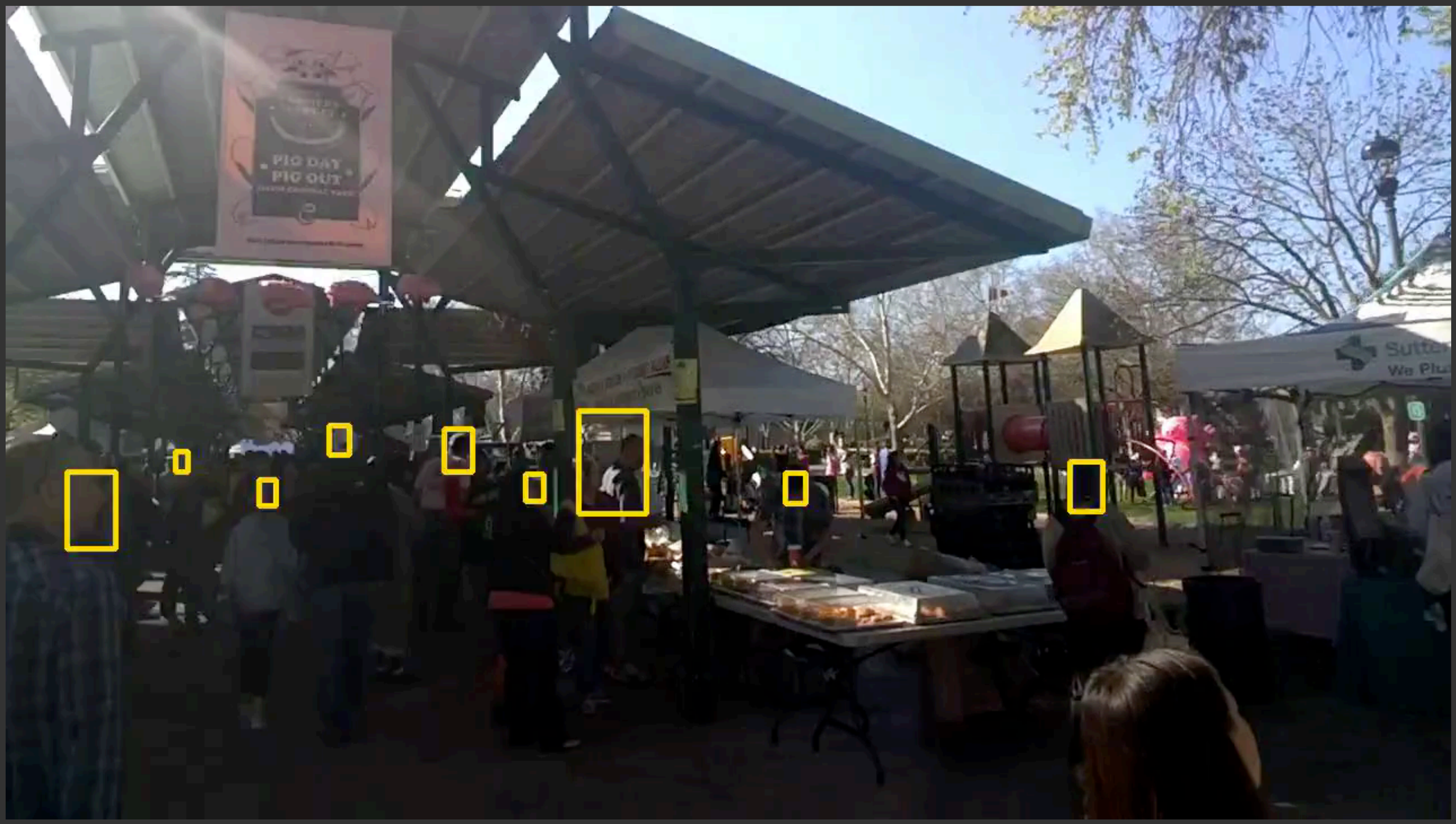
How much new visual data is observed as recording continues?



Ensemble of face detectors for KrishnaCam

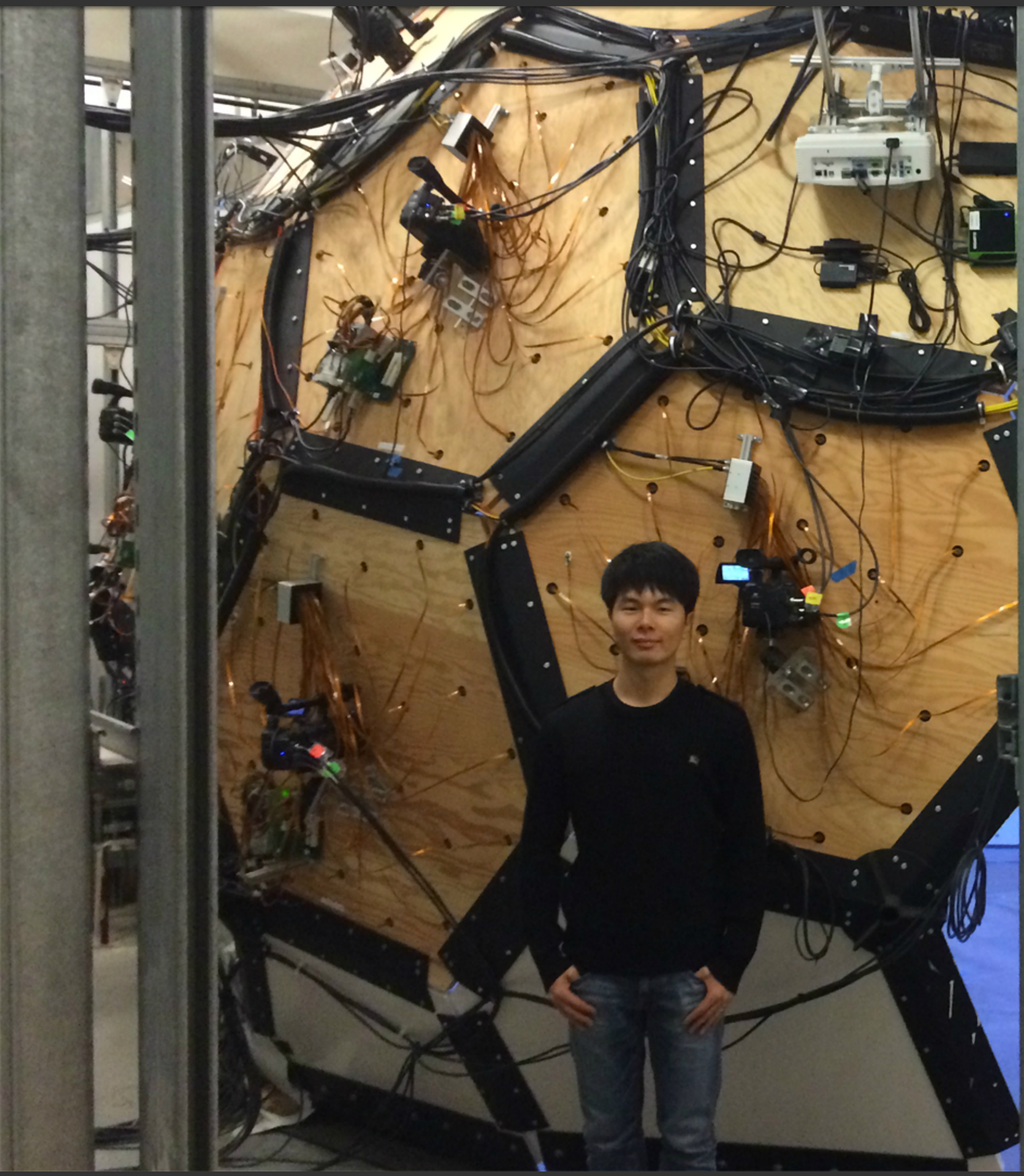


Ensemble of face detectors for KrishnaCam



Sensing human social interactions

[Joo 2015]



CMU Panoptic Studio
480 video cameras (640 x 480 @ 24fps)

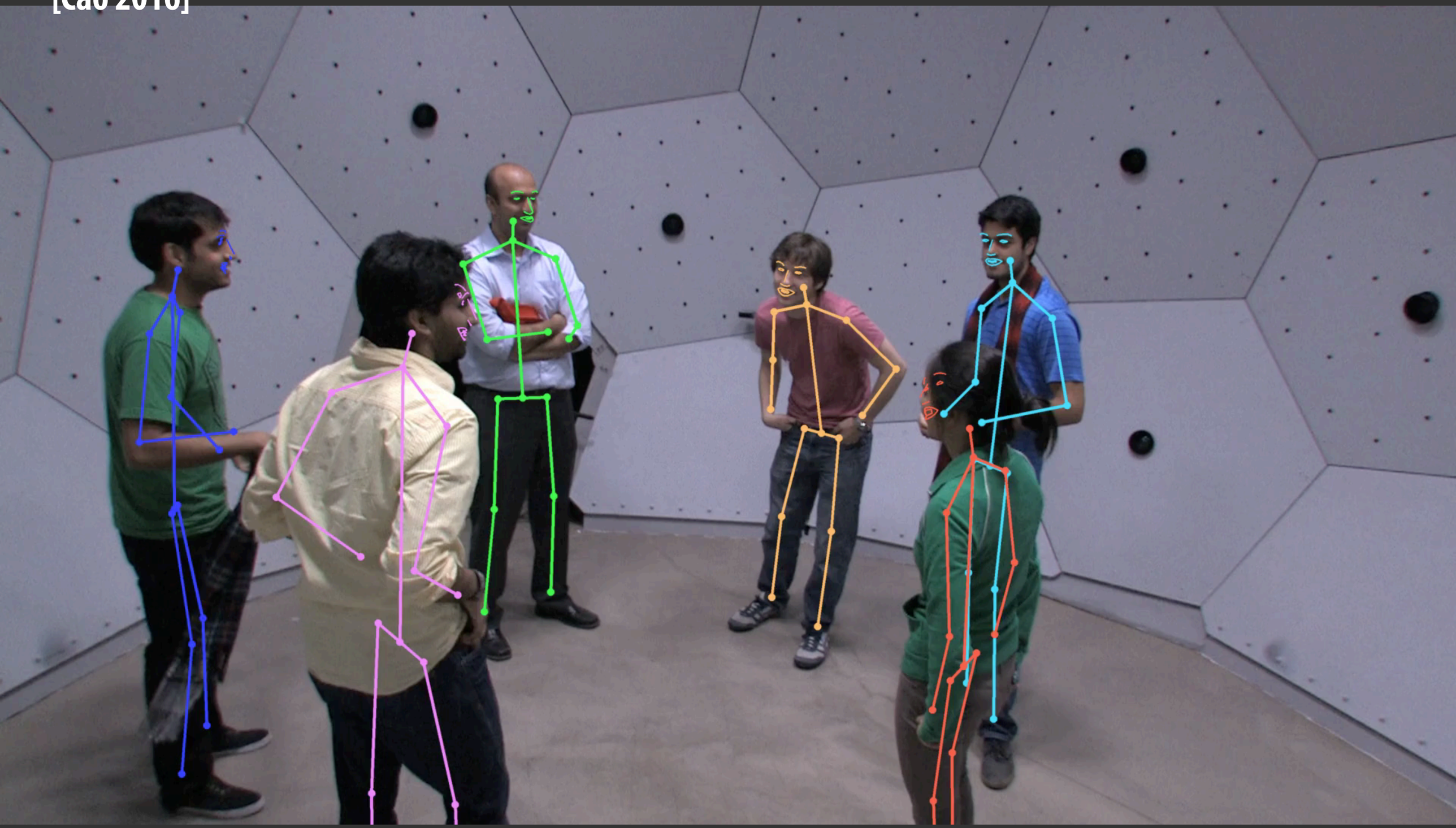
147 MPixel video sensor
(3.5 GPixel/sec)

Data from the Panoptic Studio

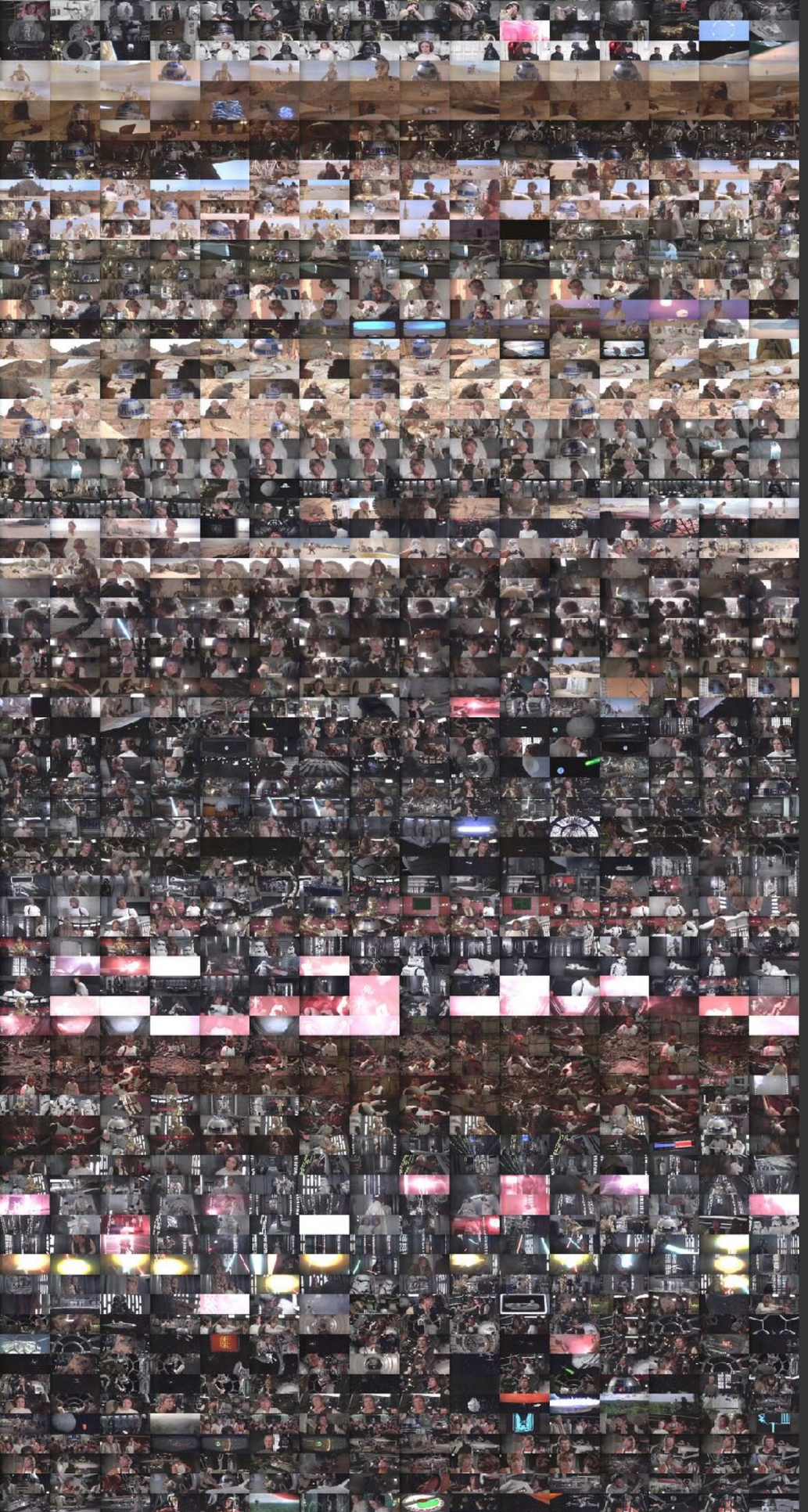
Application: capturing human social interactions

40-second sequence (captured human social interactions)

3D pose reconstruction time: hand-coded solution by grad student — 7 hours on a 4-Titan Xp machine [Cao 2016]



[Courtesy Yaser Sheikh, Tomas Simon, Hanbyul Joo] [Joo et al. 2015]



Cinematography analysis

Collaboration with Alex Hall, Maneesh Agrawala (Stanford)

What is the average length of shot in a movie?

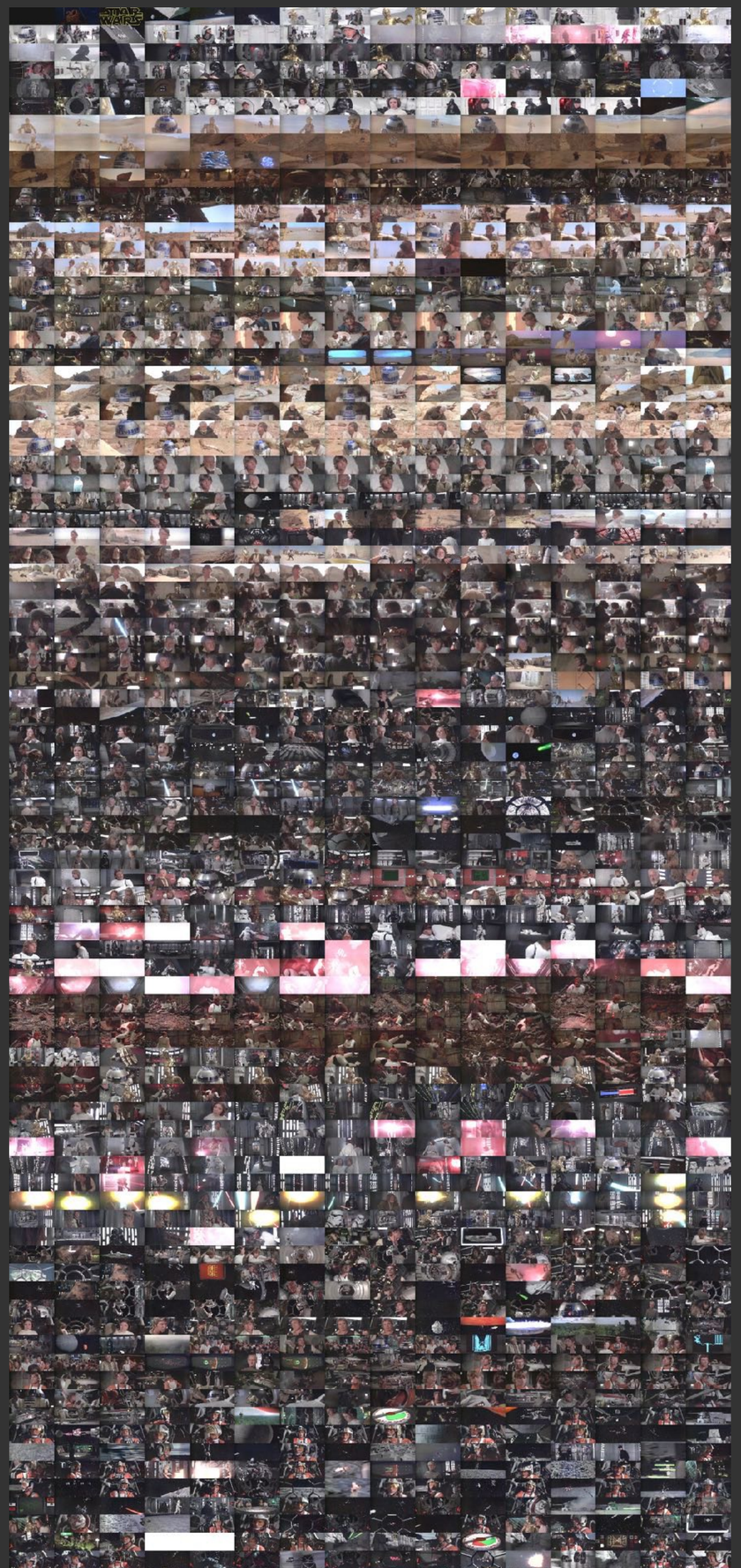
Does the director favor close ups or wide shots? How much camera motion is used?

What are the main color palettes in the film?

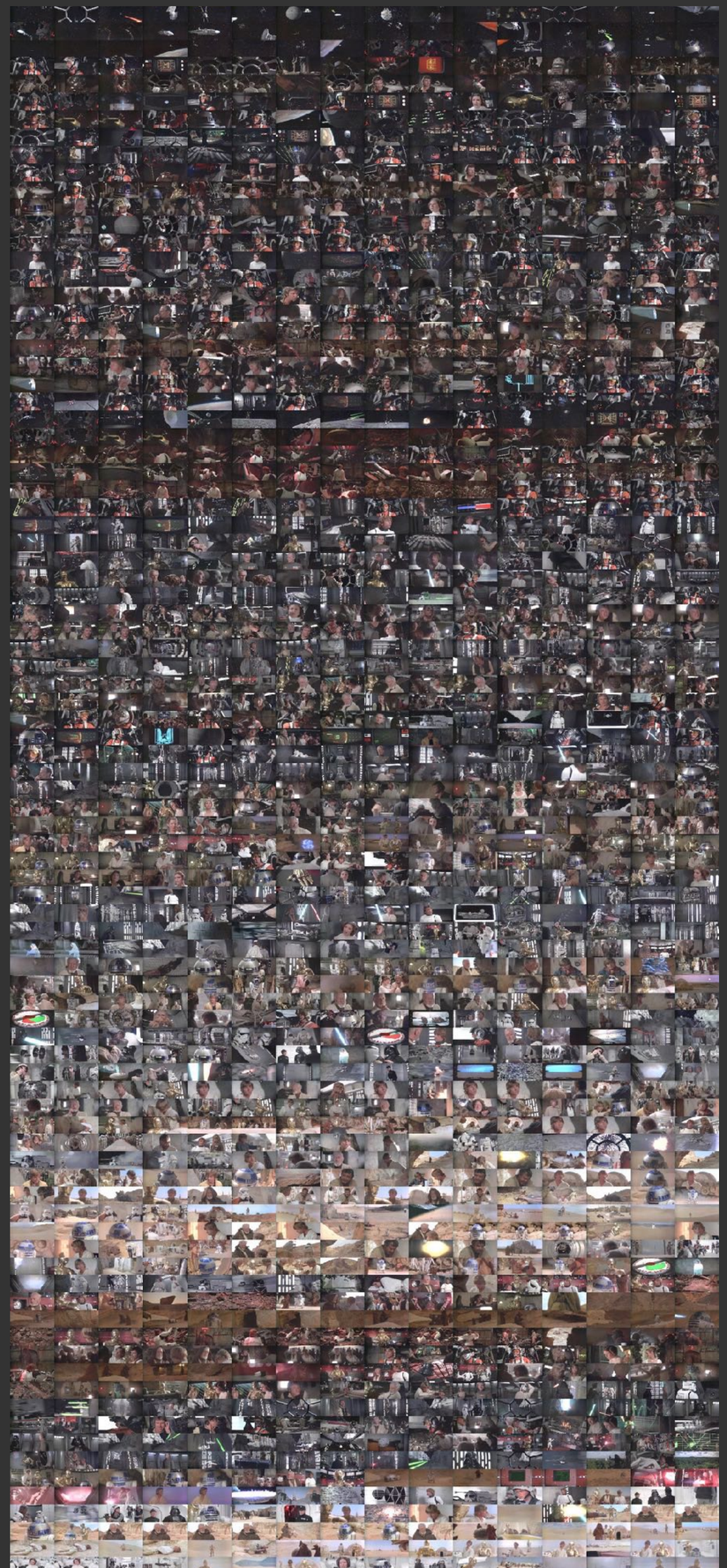
How do these traits vary across films or time?

“Star Wars Episode IV: a New Hope”
Segmented into shot boundaries based on image histograms

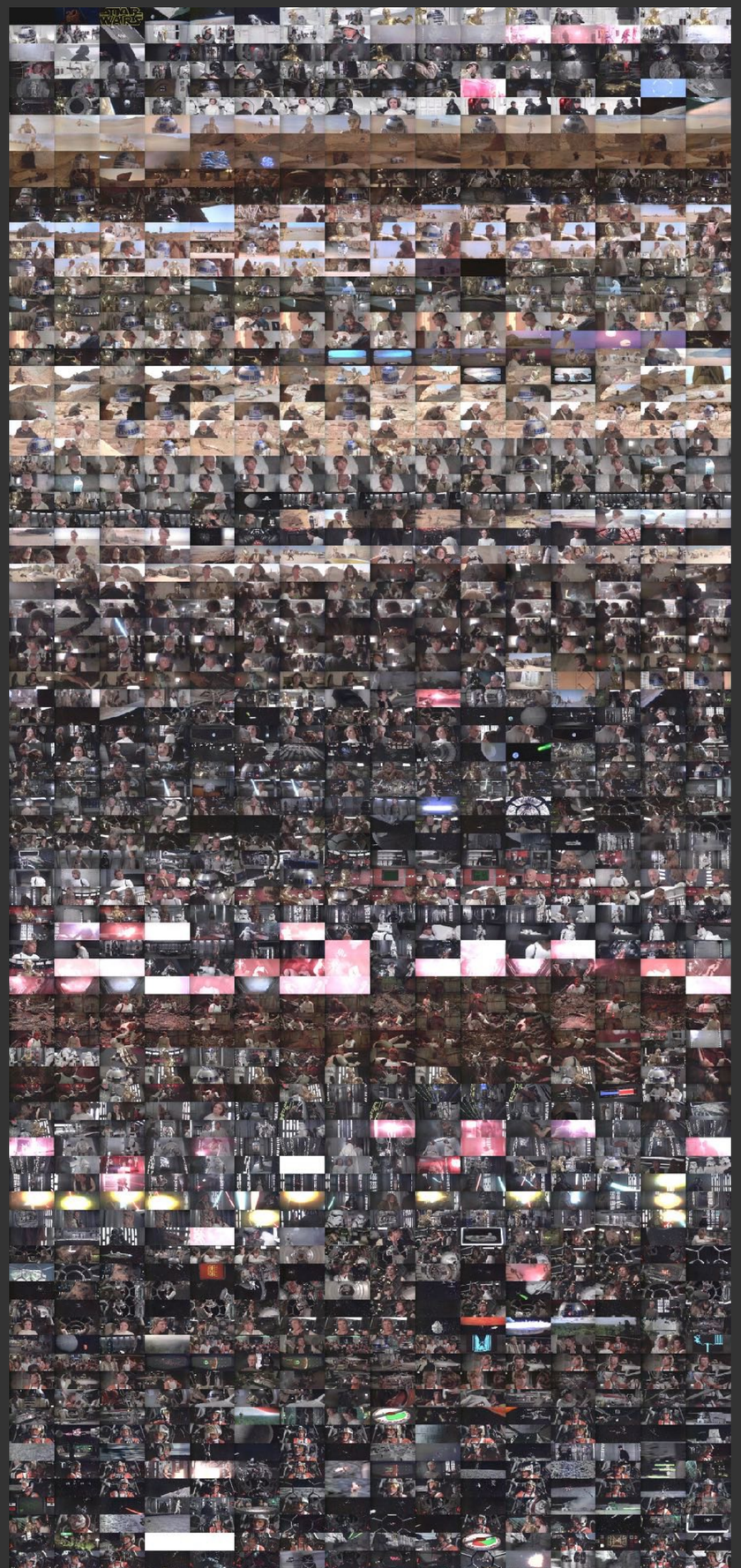
Star Wars Ep IV: Sorted by time



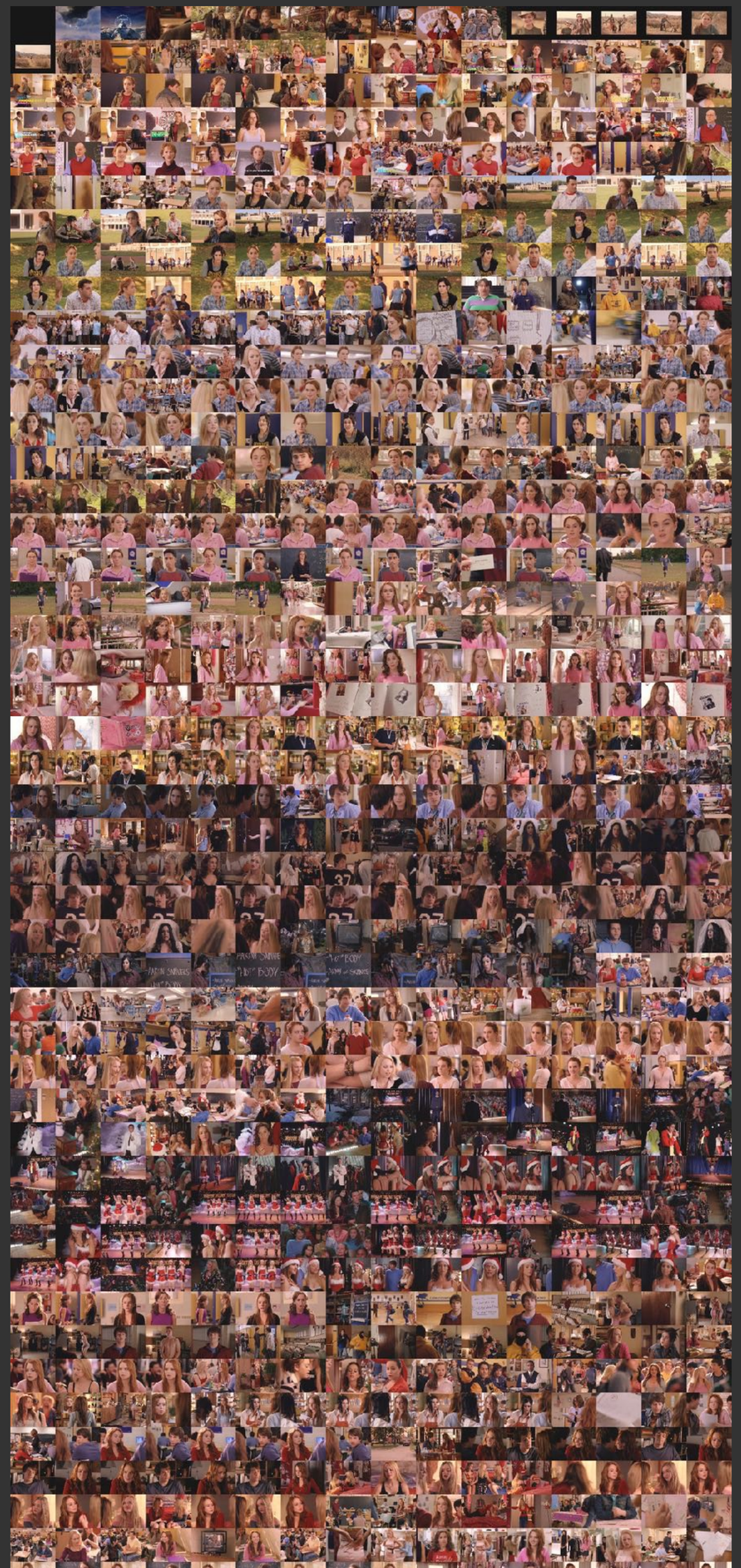
Sorted by color



Star Wars Ep IV



Mean Girls



Workload characteristics

- **Large video collections (100's GBs-to-TBs compressed)**
  - *Decompress and deliver pixels efficiently to compute units*

- **Basic data-parallel operations (map, scan), often performed on sampling of frames**
  - *Analytics-style computations: not tightly coupled, latency sensitive global communication typical of ML model training*

- **Efficient pixel processing pipelines utilize kernels from expert-tuned libraries, generated by DSLs**
  - *e.g., Halide, DNN inference using Caffe, OpenCV, MKL*
  - *complex "UDFs" that are already parallelized, run most efficiently on accelerators*

Alternatives

- Distributed data-analytics frameworks
  - [Hadoop, Spark]
- Array/raster databases
  - [SciDB, RasDaMan, GIS databases]
- Distributed machine learning frameworks
  - [TensorFlow, MxNet, CNTK]
- Emerging closed systems for “vision as a cloud service”
  - Google Cloud Vision API
  - Microsoft Cognitive Services API
  - feature turnkey solutions for object classification, face detection, motion detection, OCR, stabilization, inappropriate content filtering

Efficiently delivering video data to GPU/ASIC accelerated pixel-processing pipelines:

Scanner: efficient video analysis at scale

with Alex Poms (CMU), Will Crichton (Stanford), Pat Hanrahan (Stanford)

Design goals / principles

Design principle 1: keep it simple

- Enable non-expert programmers (vision researchers, visual data analysts) to rapidly develop and deploy video analysis applications at cloud scale

Design principle 2: be efficient

- "Near-HW-peak single-node perf" then scale out
- Utilize heterogenous hardware: ASICs for video encode/decode, run kernels on multi-core CPUs, GPUs, future DNN accelerators

Non goals:

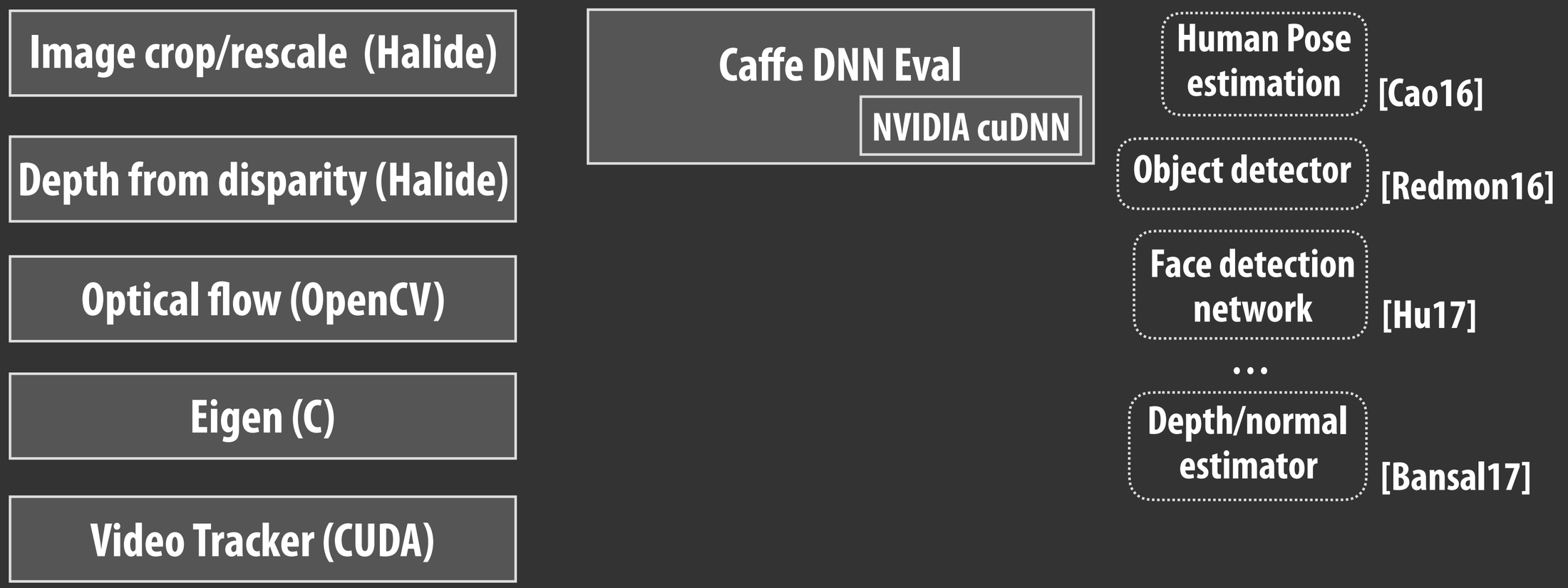
- Do not be a new kernel description language
- Interoperate with state-of-art 3rd-party kernel libraries and kernel code generated from existing DSLs

Setup

I have a list of videos in a directory…

```
myvideos/cam000.mp4
myvideos/cam001.mp4
myvideos/cam002.mp4
…
myvideos/cam479.mp4
```

And I have a library of parallel pixel processing kernels for CPUs and GPUs:

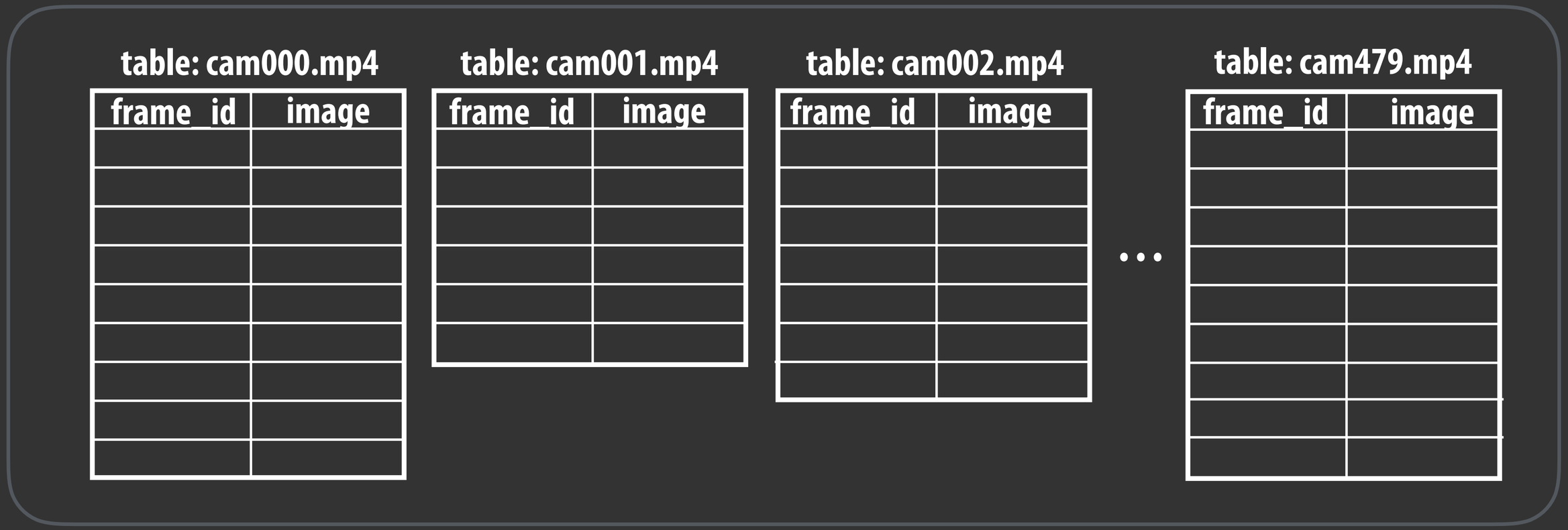


Represent videos as relations (tables)

```
myvideos/cam000.mp4
myvideos/cam001.mp4
myvideos/cam002.mp4
…
myvideos/cam479.mp4
```

Ingest into Scanner…

Scanner dataset: capture_session

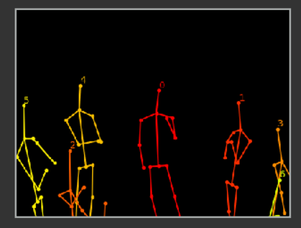


Computations: DAGs of image processing operations

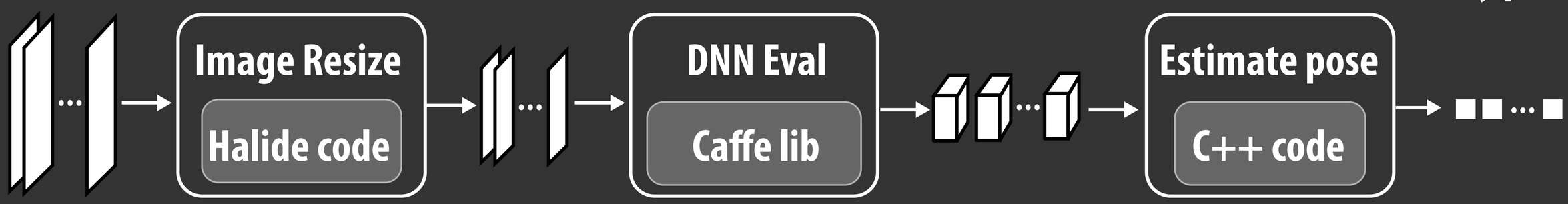
640x480 frames

496x368 frames

62x46x15 activations



(x,y) pos for body parts



Resource: GPU

Resource: GPU

Resource: 4 CPU cores

Pipeline kernels map to heterogeneous resources: CPUs, GPUs, ASICs

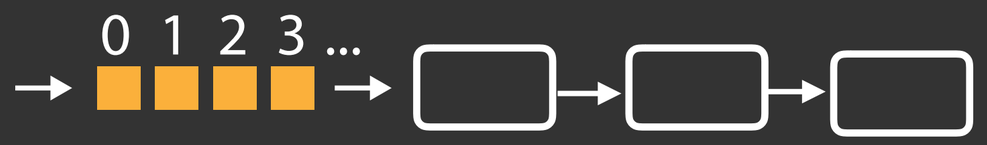
Scanner maps pipelines onto a stream of video frames from tables

Pipeline:

Input:

cam_000

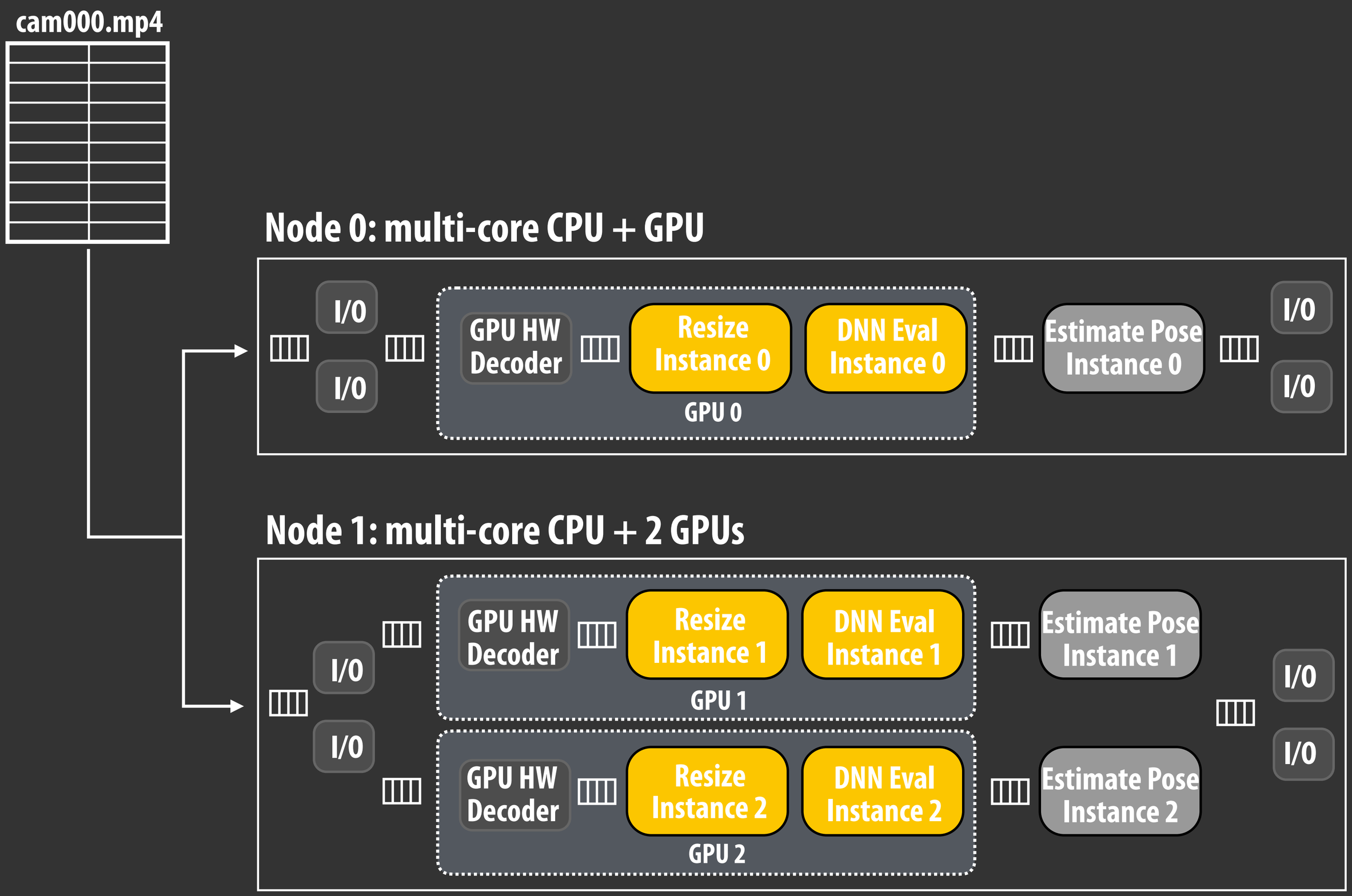
id	frame



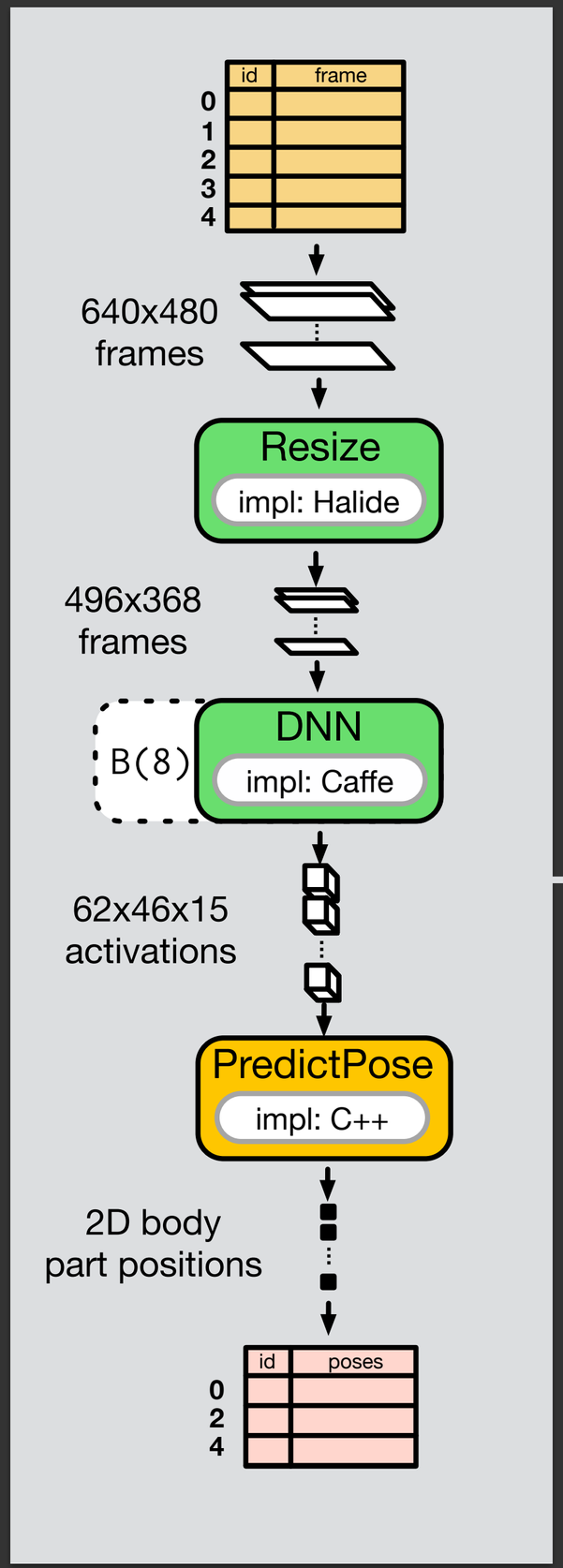
pose_000

id	2dpose

Parallel execution in Scanner



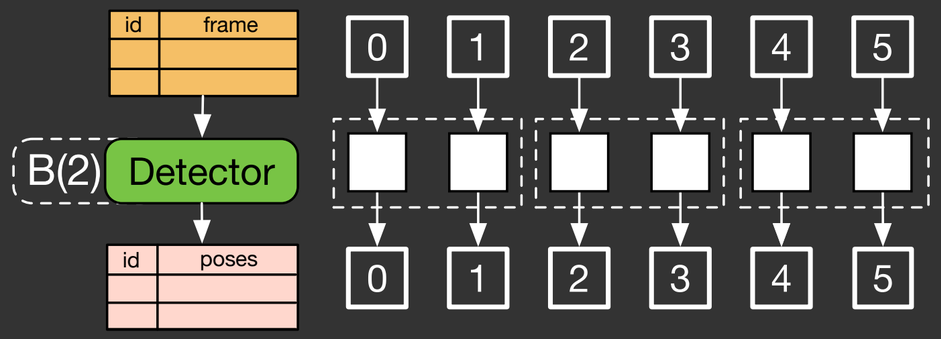
A simple Scanner program




```
db = scanner.Database()
videos = os.listdir('/myvideos')
video_tables = db.ingest_videos(videos)

jobs = []
for table in video_tables:

    resized = db.ops.Resize(
        frame  = table.column('frame').all(),
        width  = 496,
        height = 368,
        device = GPU)

    activations = db.ops.DNN(
        frame  = resized,
        model  ='cpm.prototxt',
        batch  = 8,
        device = GPU)

    poses = db.ops.PredictPose(
        activations = activations,
        device      = CPU)

    jobs.append( Job(columns = [poses],
                     name    = 'poses'
                     output_filter = stride(2) ) )

pose_tables = db.run(jobs)
```

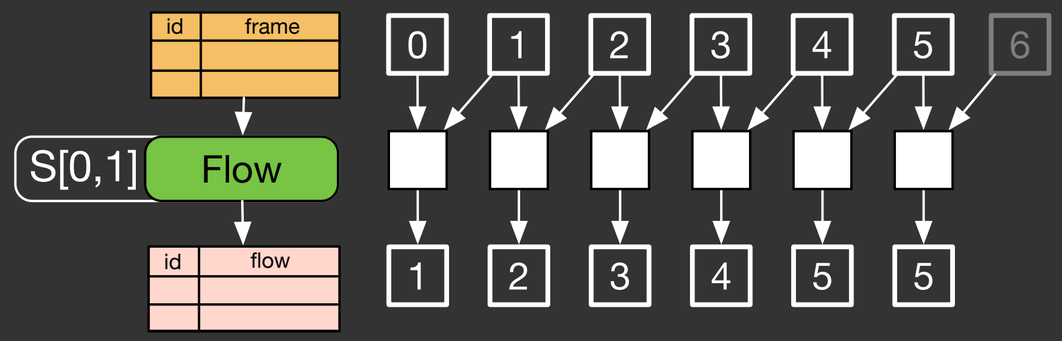
Scanner data-parallel operators

Map (with element batching)



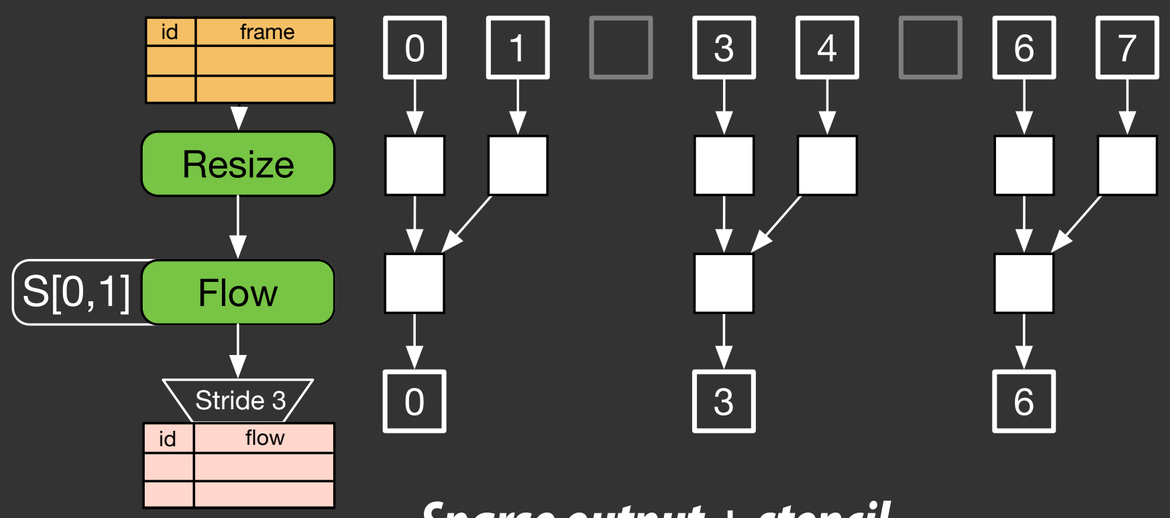
Batching frames together during processing (efficient mini-batch kernels, e.g. DNNs)

Temporal Stencil



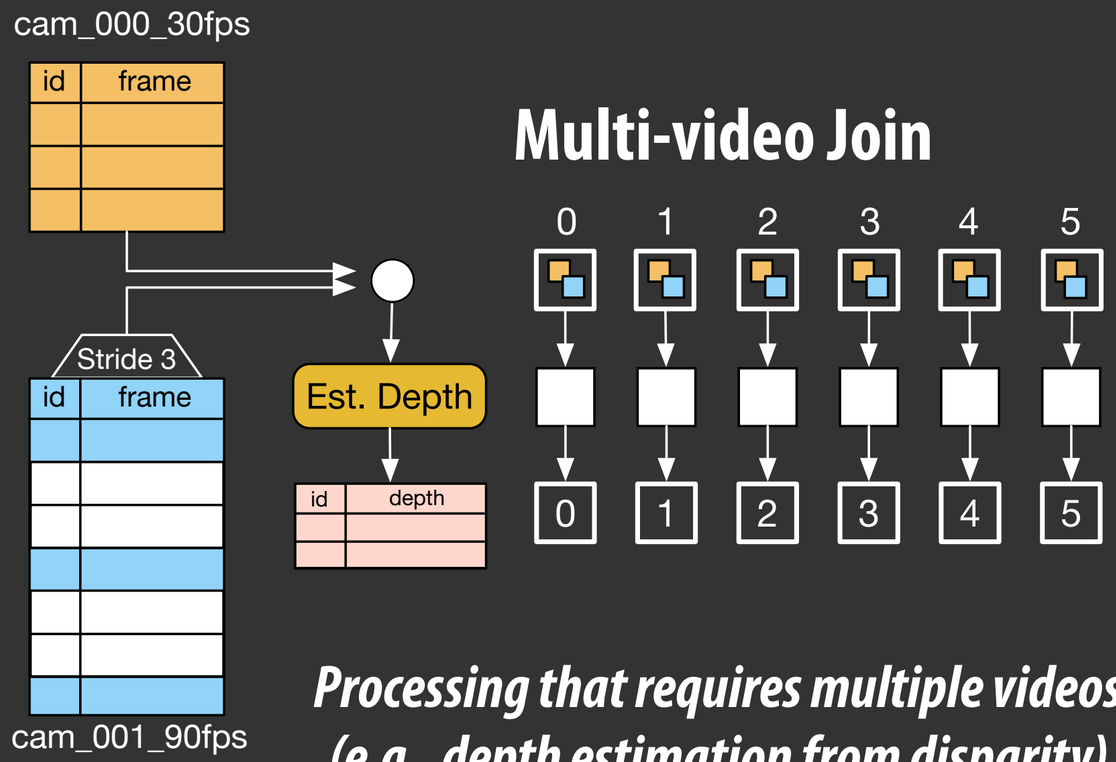
(e.g., computing optical flow, video stabilization/hyperlapse)

Temporal Stencil + Stride



Sparse output + stencil (e.g., optical flow on every 30th frame)

Multi-video Join

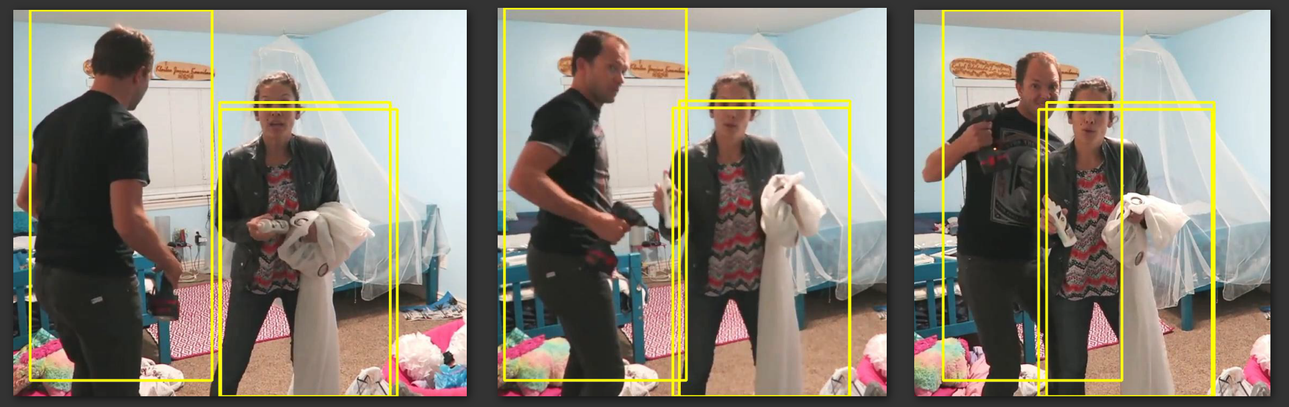


Processing that requires multiple videos (e.g., depth estimation from disparity)

Challenges unique to video domain

1. Striding/gathering frames from compressed video streams
2. Temporal dependencies in common video processing operations

Object tracking (stateful)

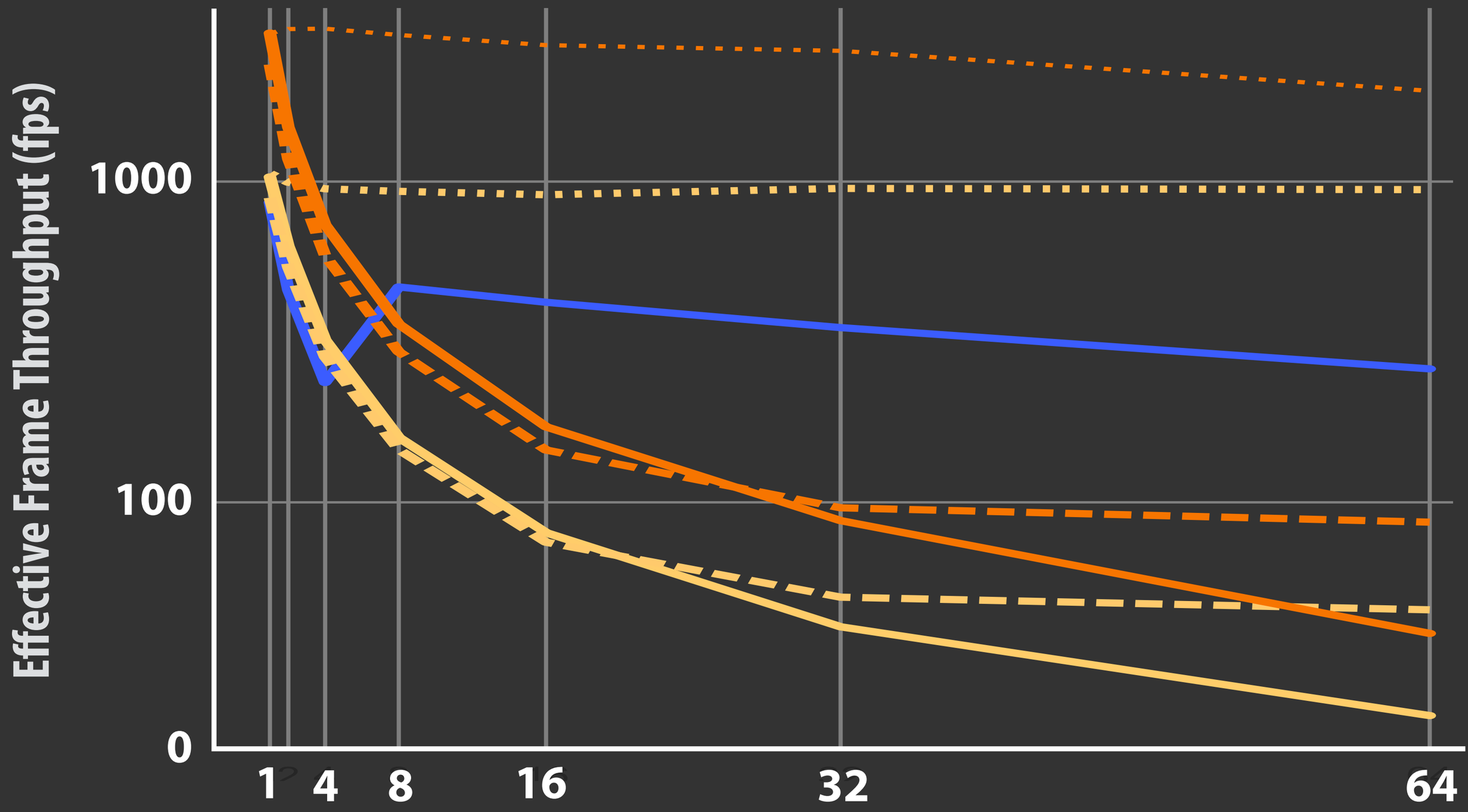


Activity recognition (must observe long sequence of frames)

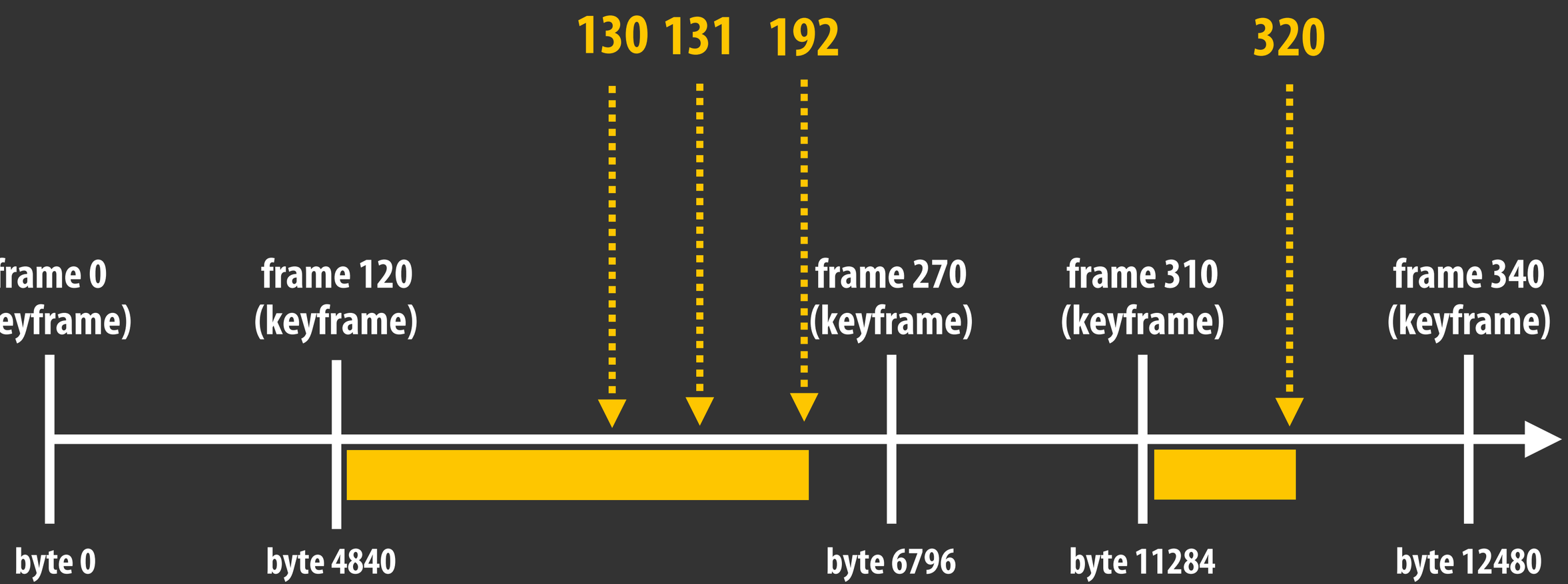
[Ma 2016]

Choosing a video storage format

16 core Xeon GPU + 1 Titan Xp GPU
1920x1080 video



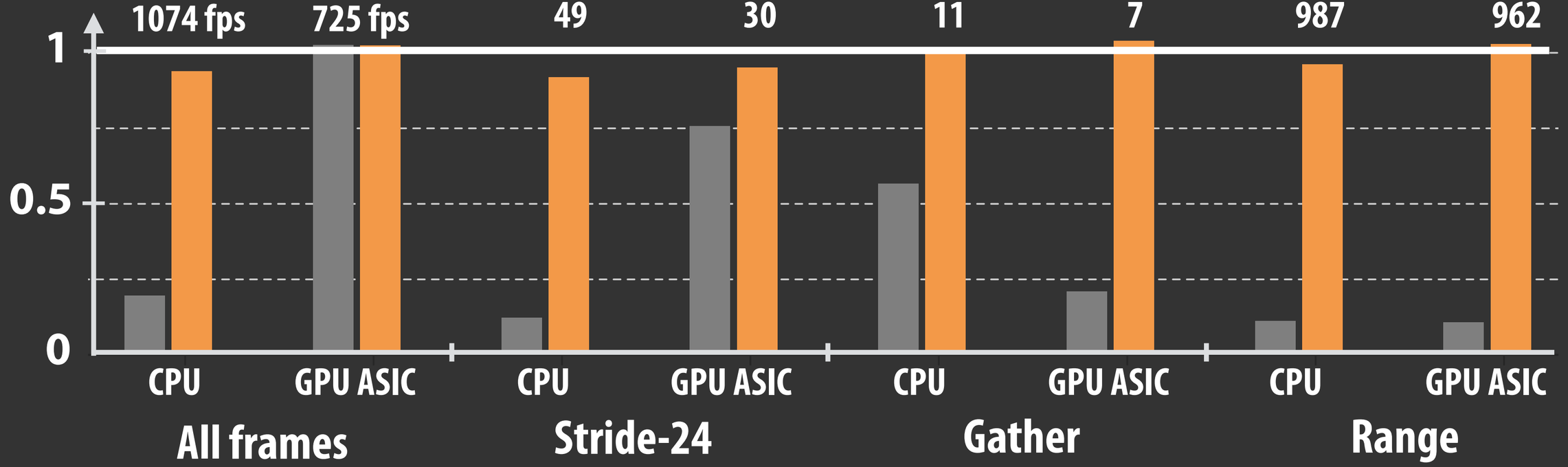
Maintain index of keyframe locations to accelerate parallel decode



Scanner maintains index of keyframe locations to enable work-efficient parallel, gathered decode

Importance of well-optimized video decode

Decoded Frame Throughput (1080p)
(Relative to expert hand-tuned implementations)



CPU = 16-core CPU

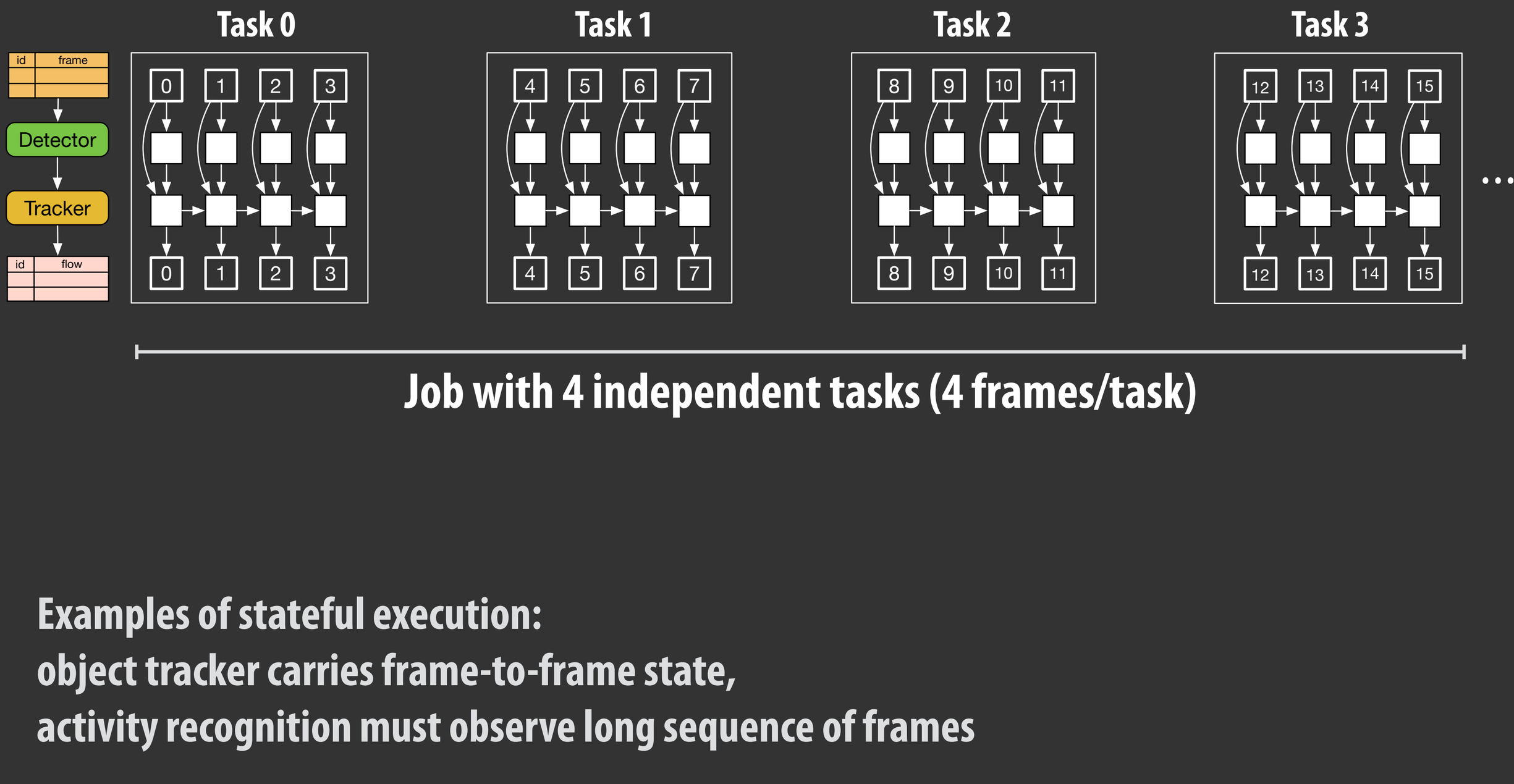
GPU = Titan Xp

Scanner (with keyframe index)

Baseline (off-the-shelf libraries and tools)

Handling temporal dependencies in video processing

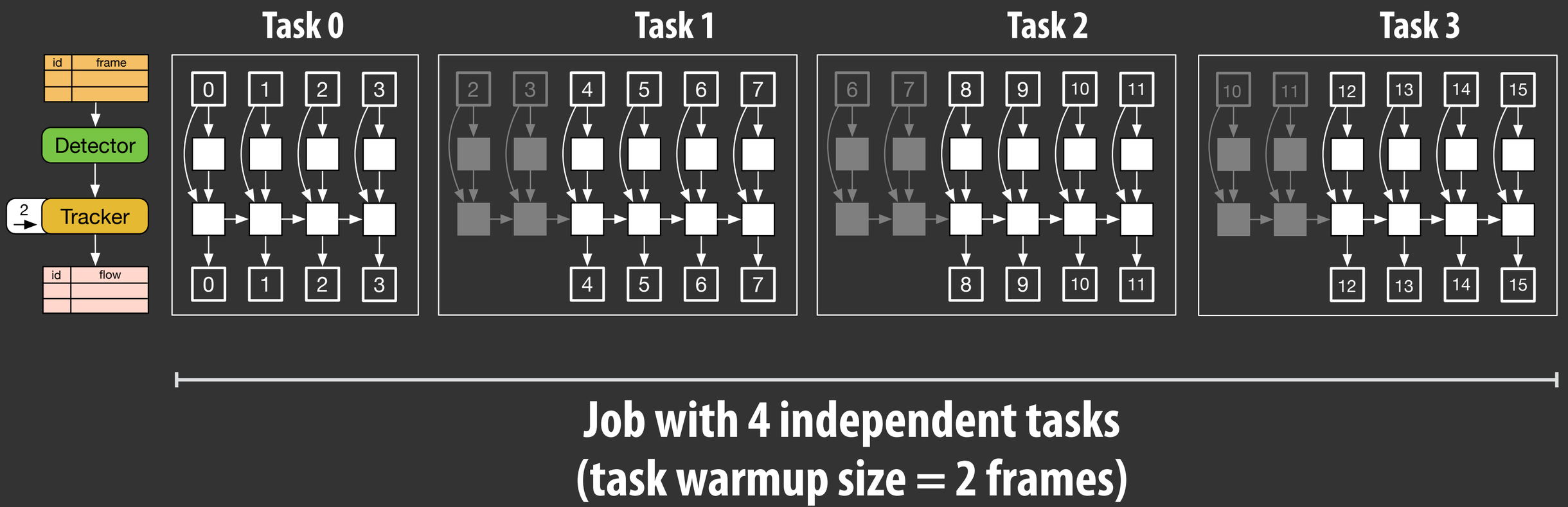
Two-level stream hierarchy: applications partition jobs into“tasks”
(Scanner ensures all elements in task are scheduled serially on same pipeline)



Examples of stateful execution:
object tracker carries frame-to-frame state,
activity recognition must observe long sequence of frames

Handling temporal dependencies in video processing

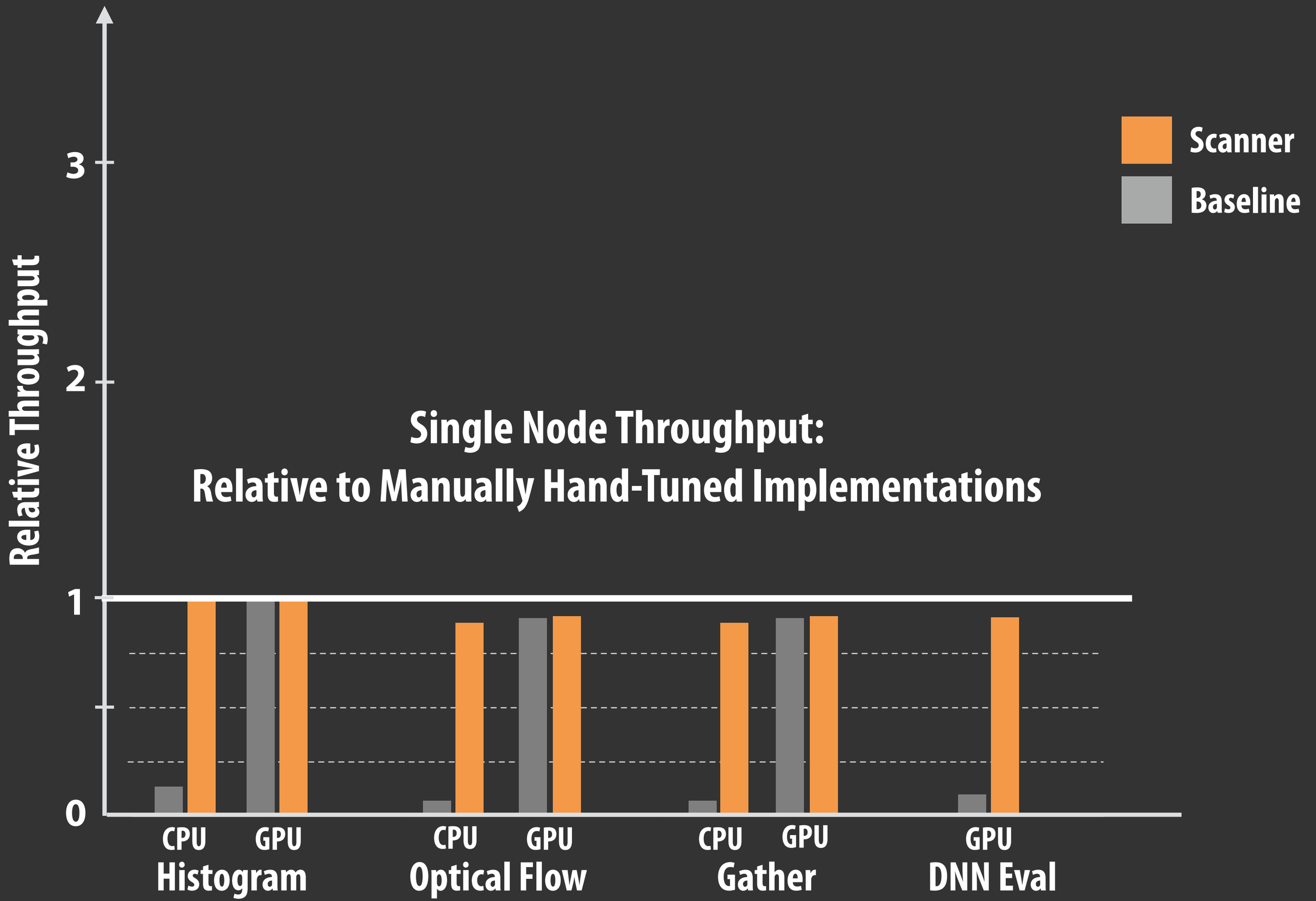
Tasks can receive "warmup" stream elements to initialize state
Pipeline generates no output for these elements.
(redundant computation across tasks to facilitate parallelism)



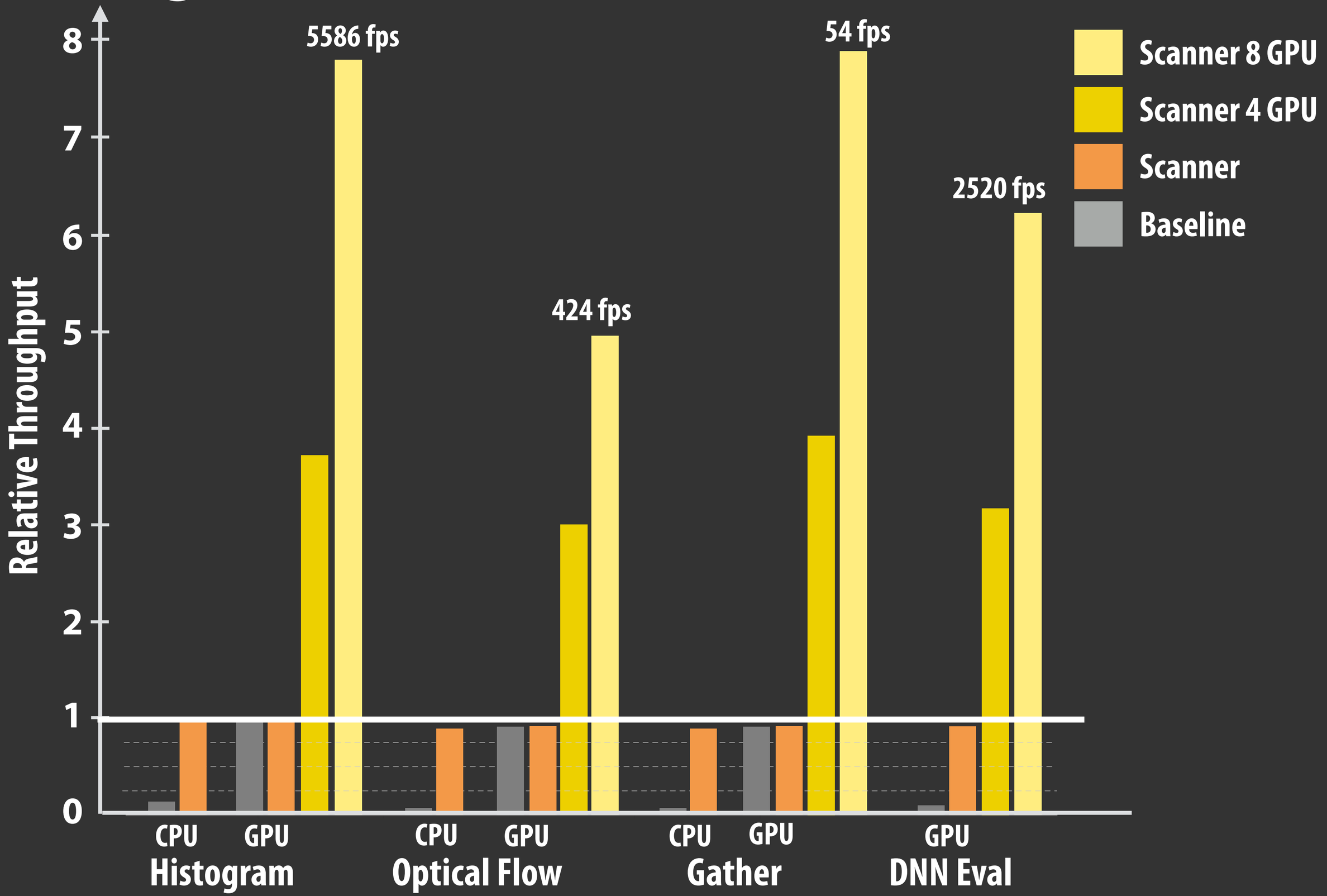
Intuition for warmup:
When "influence" of state is relatively local in time, warmup allows parallel execution with little change in output values
e.g. provide task additional 30 frames to initialize object tracker

Preliminary results

Single node: scanner has low overhead



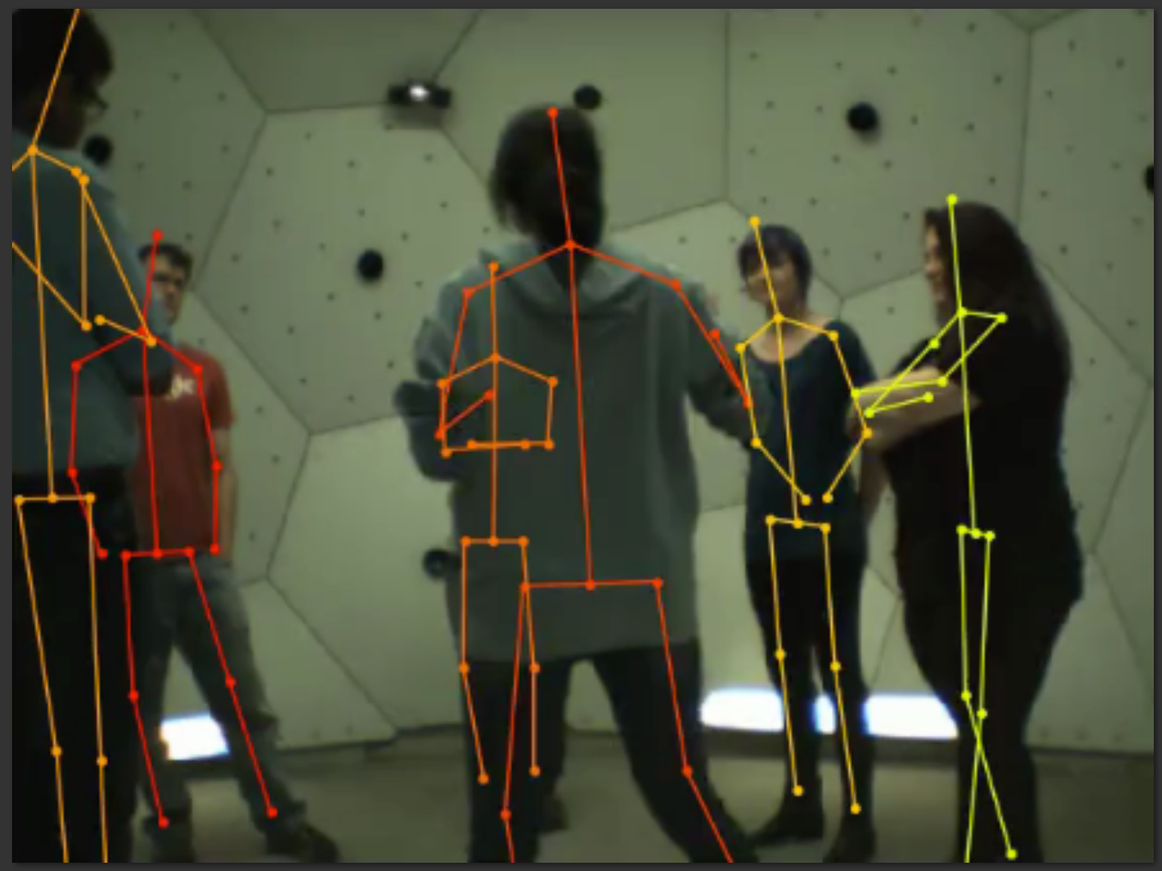
Scaling to 2 machines (8 GPUs)



16-core CPU, GPU=Titan Xp

3D human pose reconstruction

Processing 40 seconds of video
from CMU Panoptic studio



Grad student hand-tuned: 7 hrs (1 node x 4 Titan Xp GPUs)

Scanner: 2.6 hrs (1 node x 4 Titan Xp GPUs)

Scanner on cluster: 38 mins (4 node x 4 Titan Xp)

Approaching viability for extended capture sessions.

Shot segmentation (cinematography analysis)

608 feature length films (2.4 TB)
103M frames
Histogram-based shot segmentation of all films: 4.7 hrs (4 node cluster, 4 GPUs/node)

Facebook Surround 360 VR video generation

(omnidirectional stereo VR video)

2048 x 2048 PointGrey Camera @ 30 FPS

Preliminary Scanner results:

Single node (32-core CPU)

- 5 secs / frame

Multi-node on Google Compute Engine (8 x 32-core nodes)

- 0.7 secs/frame

14 cameras

8K x 8K stereo panorama output = 12.5 secs per frame on 32-core CPU

Scanner

- Open source compute engine for high-performance cluster-scale video analytics (attacks platform/infrastructure needs)
  - Integrates high performance video delivery to heterogeneous accelerated computing pipelines
- Hope: influence design of current future distributed systems
  - Spark/Hadoop ecosystem
  - APIs for cloud-based video analysis services (Microsoft Cognitive Services API, Google Cloud vision API, NVIDIA Intelligent Video Analytics)

Ongoing: American TV news analysis

- Dataset provided by Internet Archive
- 9 months of US election coverage (2012, 2016) on CNN, FOX, MSNBC
- 6.6 TB, 18,000 hours of video, 1.5 billion frames

Fareed Zakaria GPS

CNN Newsroom

Situation Room

CNN Newsroom with Poppy Harlow

The Lead with Jake Tapper

America News Headquarters

The Five

The Real Story With Gretchen Carlson

Shepard Smith Reporting

On the Record With Brit Hume

Inspiration

Geena Davis Inclusion Quotient (GD-IQ)

- Project between Google and The Geena Davis Institute on Gender in Media
- Uses computer vision to search for gender bias in blockbuster films

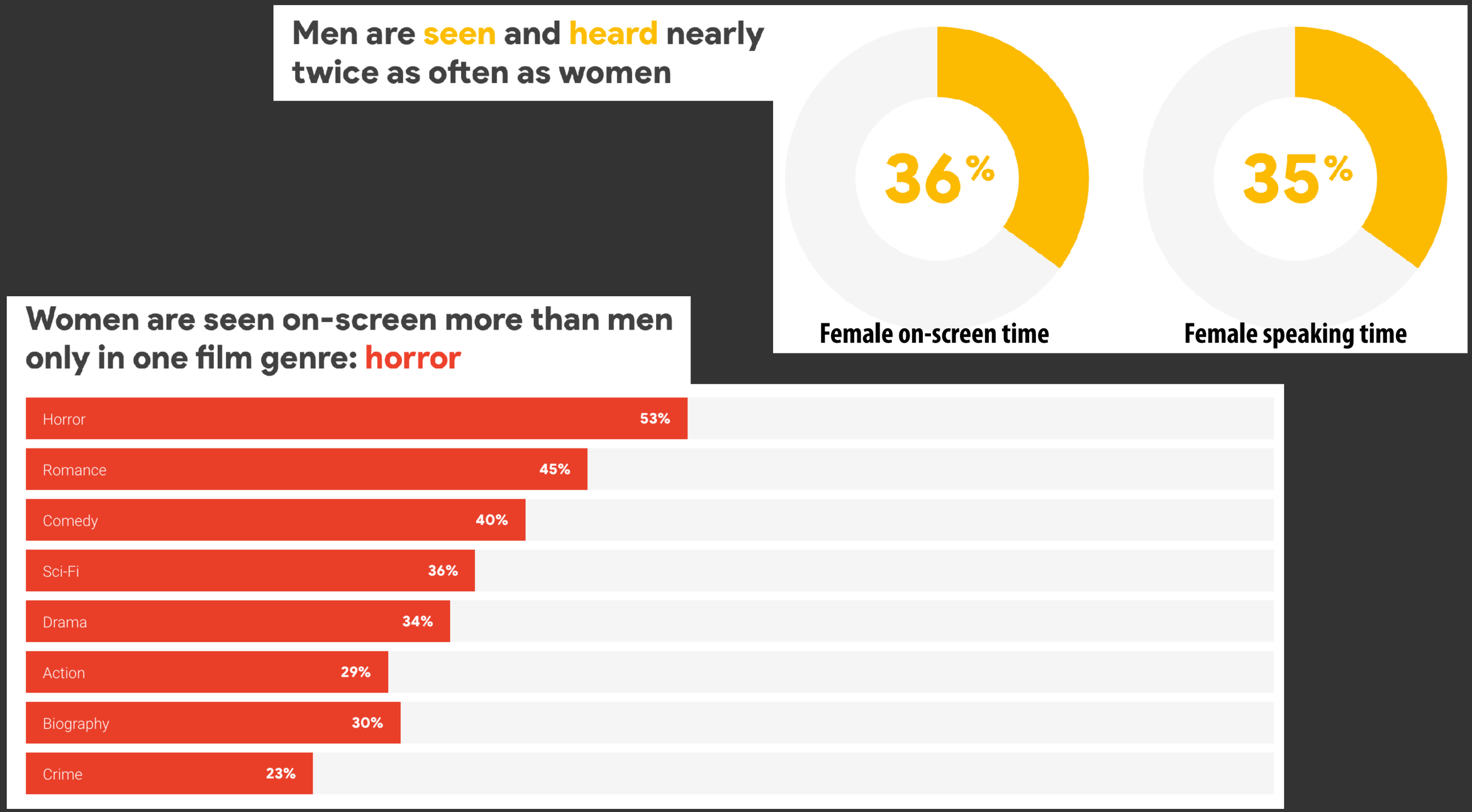


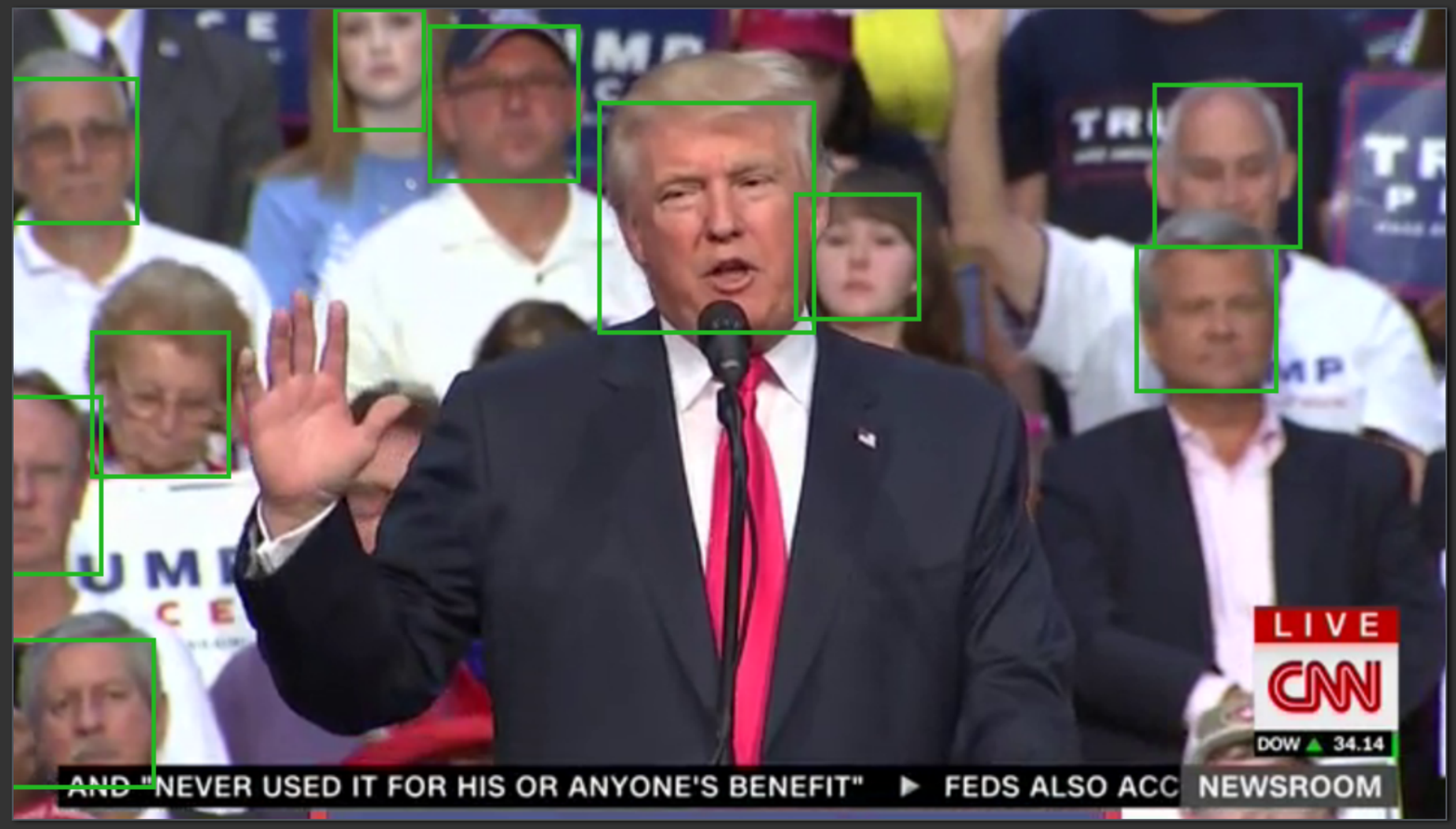
Image credit: https://www.google.com/intl/en/about/main/gender-equality-films/

Visual data mining process

Work by:
M. Perron
W. Crichton
S. Dulloor

100 hours (10 hours from each of 10 shows)
Sampled at 2 fps (every 12th frame) - 70K frames

MTCNN for face detection [Zhang 16]
"Rude Carnie" DNN for gender ID [Levi 16]



Refine filtering to include only the large faces

```
detection score > THRESHOLD1 && bbox_area > THRESHOLD2
```

Endless opportunities for innovation…

- Performance-centric algorithm innovation
  - Approximate high-quality detectors with cheaper ones
    - Manually via intelligent topology simplification?
    - Automatically via replacement or topology search?
  - Multi-resolution and/or adaptive detection techniques
    - What are most important frames to pay attention to in 18,000 hours of video?
  - Exploiting temporal coherence
    - Use results of prior frames to accelerate future processing
- Future hardware acceleration
  - Need for DNN acceleration widely recognized
  - ASIC video decoder interfaces might wish to support strided/gathered access

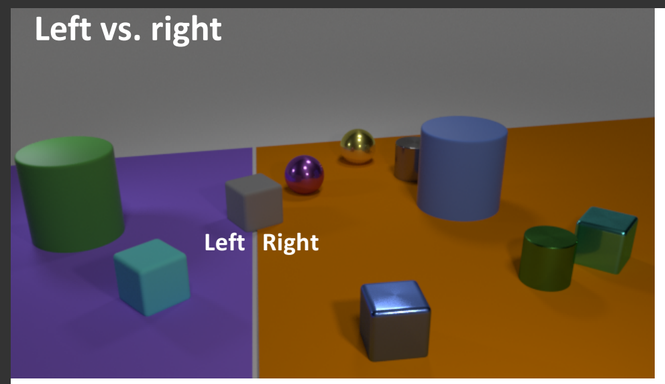
Endless opportunities for innovation…

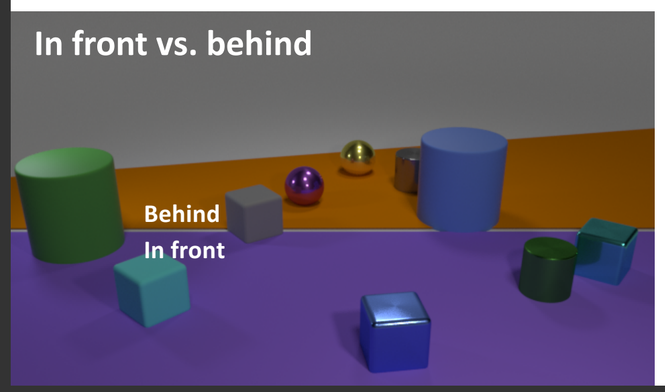
How to express visual data mining queries?

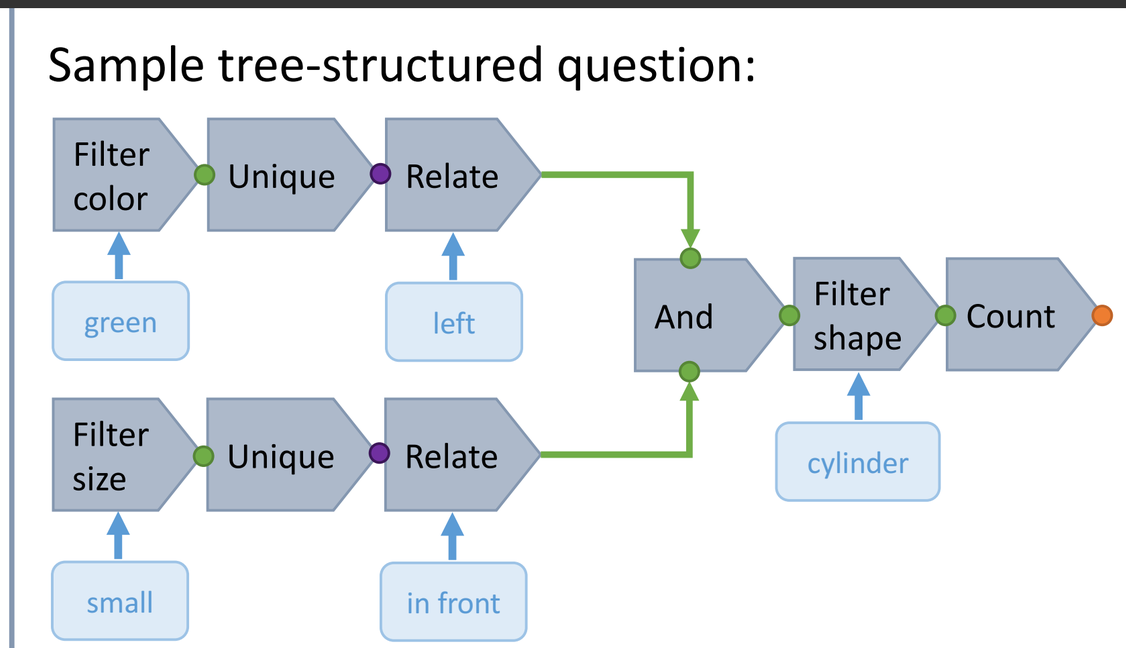
- **What is SQL for video or scenes?**

"Three cups to the left of the blue cup" [Ma 17]

```
count(left(filterbycolor(detect(cup), blue), detect(cup))) == 3
```







AI for visual reasoning CLEVR [Johnson 17]

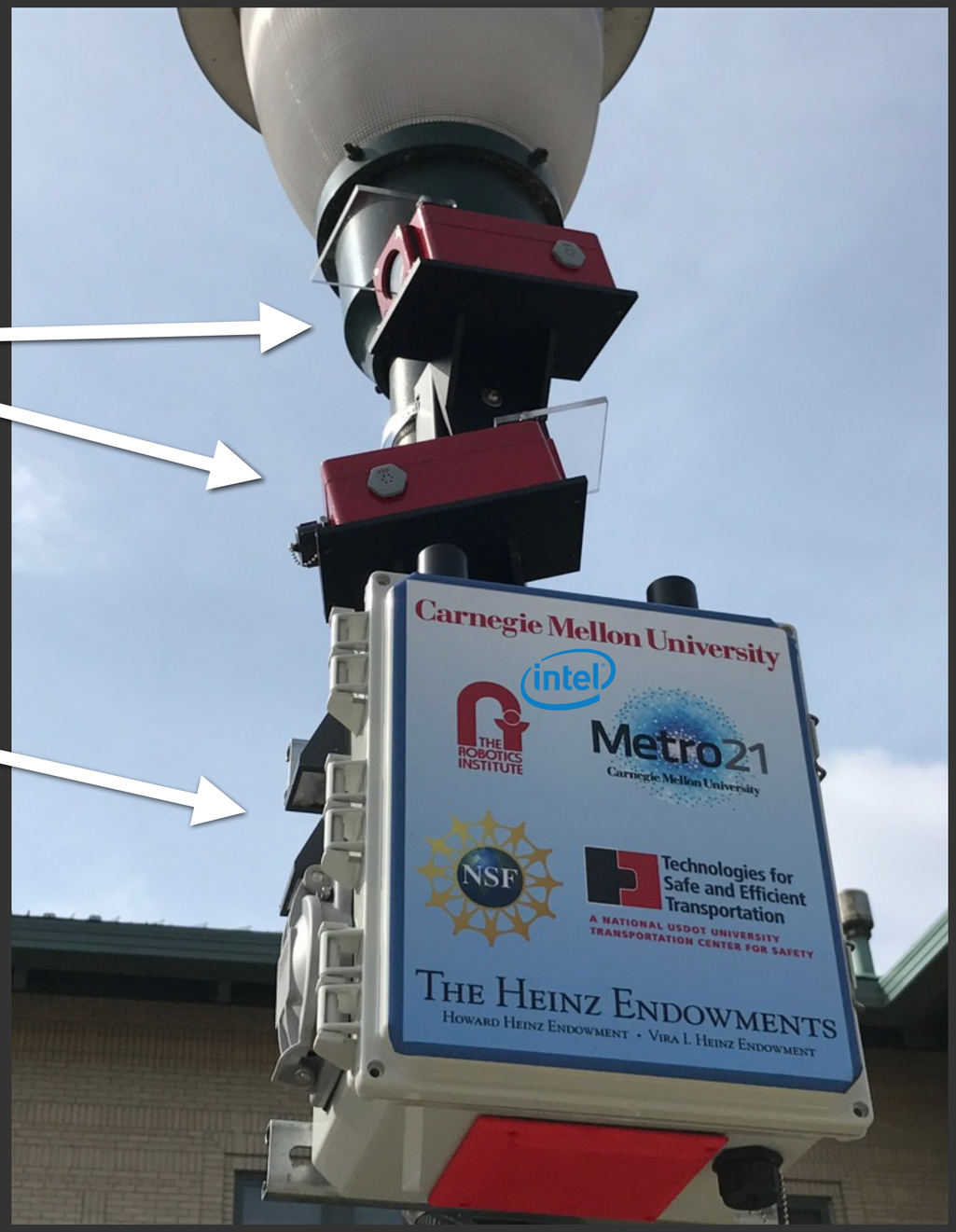
CMU urban video analytics testbed + Streamer

CMU urban video analytics testbed

Deployment of high-resolution cameras and edge compute nodes on campus at CMU and across a new city blocks nearby campus

Industrial computer vision cameras (emits RAW pixels)

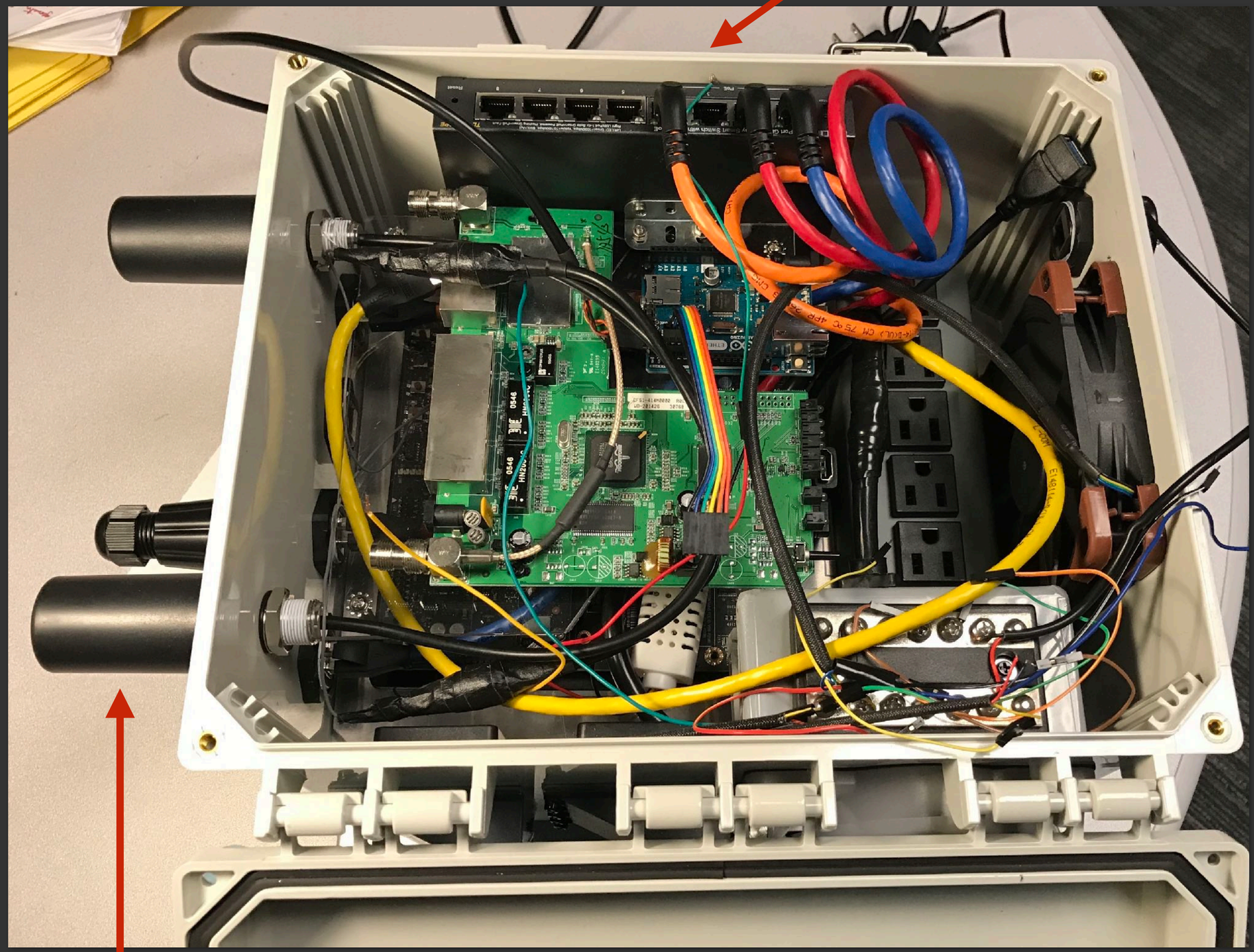
Compute node



More details at:
urbanvideo.cs.cmu.edu

Video analytics node

PoE Gigabit ethernet switch (power/data to cameras)



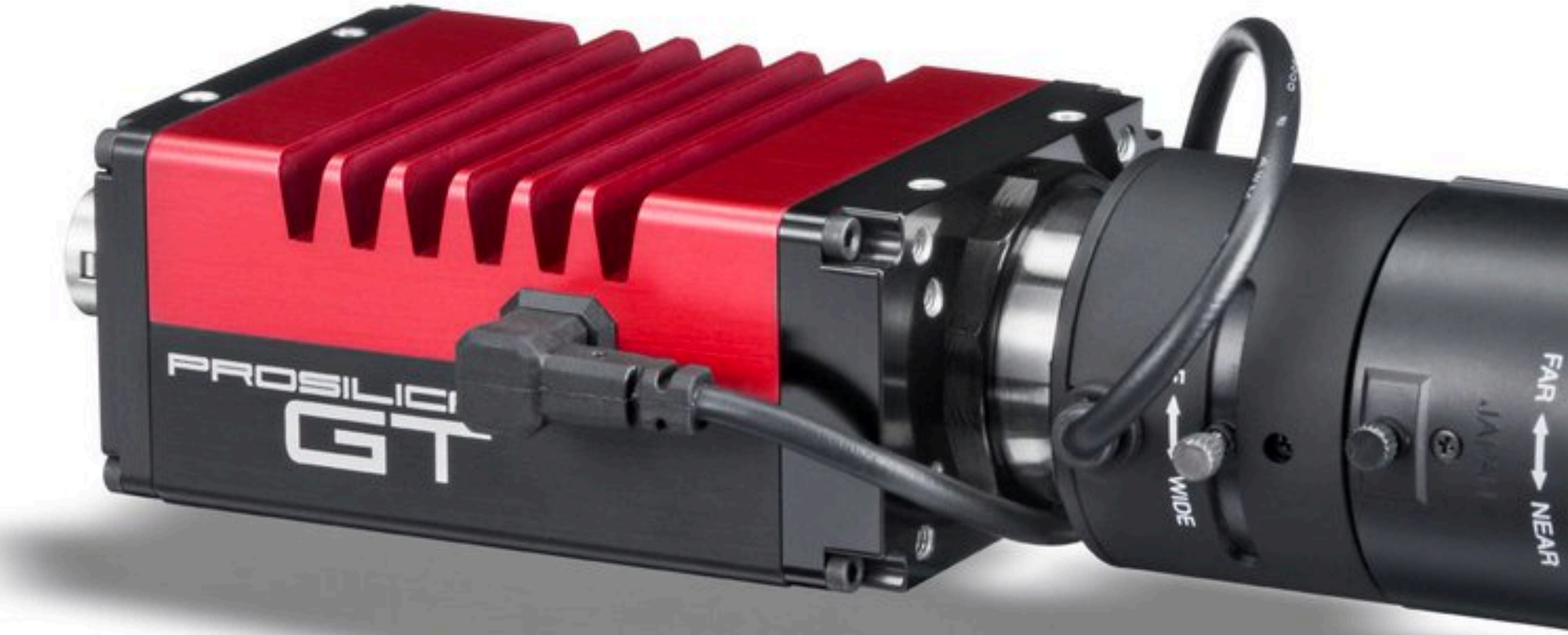
Significant compute capability on the near-camera node

Intel NUC / NVIDIA Tegra X1

1-2 TFLOPs of image processing hardware per node

Wi-fi antenna

Cameras

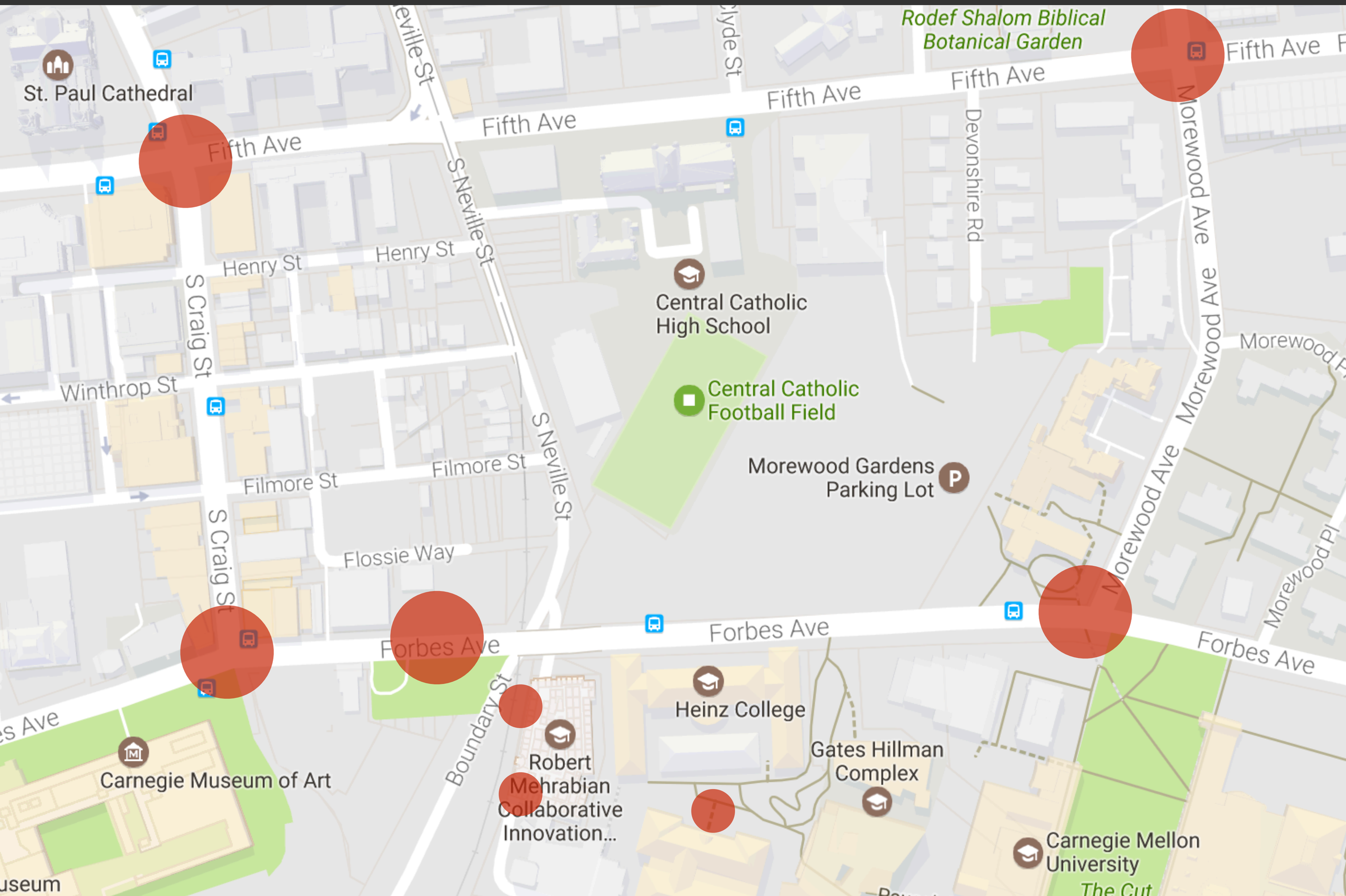


Max resolution: 2.4 MPixel
Up to 60 fps
Emits RAW pixels (uncompressed video signal)

Inside windows

One Intel NUC per camera
Intel Gen 9 Integrated graphics + media decode

Year 1 (~end of 2017)



Urban video analytics testbed goals

- Be a "living laboratory" for research in cloud-to-edge systems, computer vision, security, privacy, urban computing
  - Provide open platform for deploying streaming applications at scale
  - Facilitate easy deployment of applications to 10's-100's of cameras

- Tackle issues of privacy and policy head on
  - Start with small deployment, then grow
  - One output of project will be policy and technology guidelines for responsible capture, use, and retention of urban video data

Urban video analytics testbed: use cases

TRANSPORTATION / CITY DYNAMICS

Vehicle/pedestrian/bicyclist trajectories

Notable "event" counting: bike near bus, near collisions, pedestrian unexpectedly entering street

Detailed statistics of human and vehicle behavior at intersections (for autonomous vehicle development and training)

External validation of autonomous vehicle positioning/ decision making

CLIMATE / ENVIRONMENTAL MONITORING

Air-quality estimation from video data

Per-vehicle pollution estimation
(based on analysis of exhaust)

Frozen road detection

PUBLIC SAFETY

Students opt-in to automated tracking when walking home at night

NEW COMPRESSION TECHNIQUES

"Smart camera" that learns a viewpoint-specific compression scheme (reduce network requirements)

Compression for machines, not humans: preserve information needed for analysis tasks (rather than preserve image details that are salient to human eye)

PRIVACY

Video anonymization (cameras never output original images, but anonymized images)

Which analysis applications can remain effective while being performed on anonymized video sources?

Example: air-quality analysis

- Computer vision collaborators are interested if they can attribute pollution to individual vehicles
  - Large trucks, buses, trains, etc…
- Requires 24-7 recording at low frame rate
- Jump to high frame-rate resolution when potential polluter detected
- Capture setup: two time synced 12.4 Mpixel cameras emitting 12-bit RAW
  - Two cameras per NUC

“Streamer” software platform

Dataflow-based edge-to-cloud real-time video processing framework
Open source software infrastructure for CMU Visual Analytics testbed

Camera

RAW pixels or
H.264 encoded video

Intel QuickSync
HW Decoder

(TensorFlow / Caffe)
Gen 9 CPU
(Iris Pro 580)

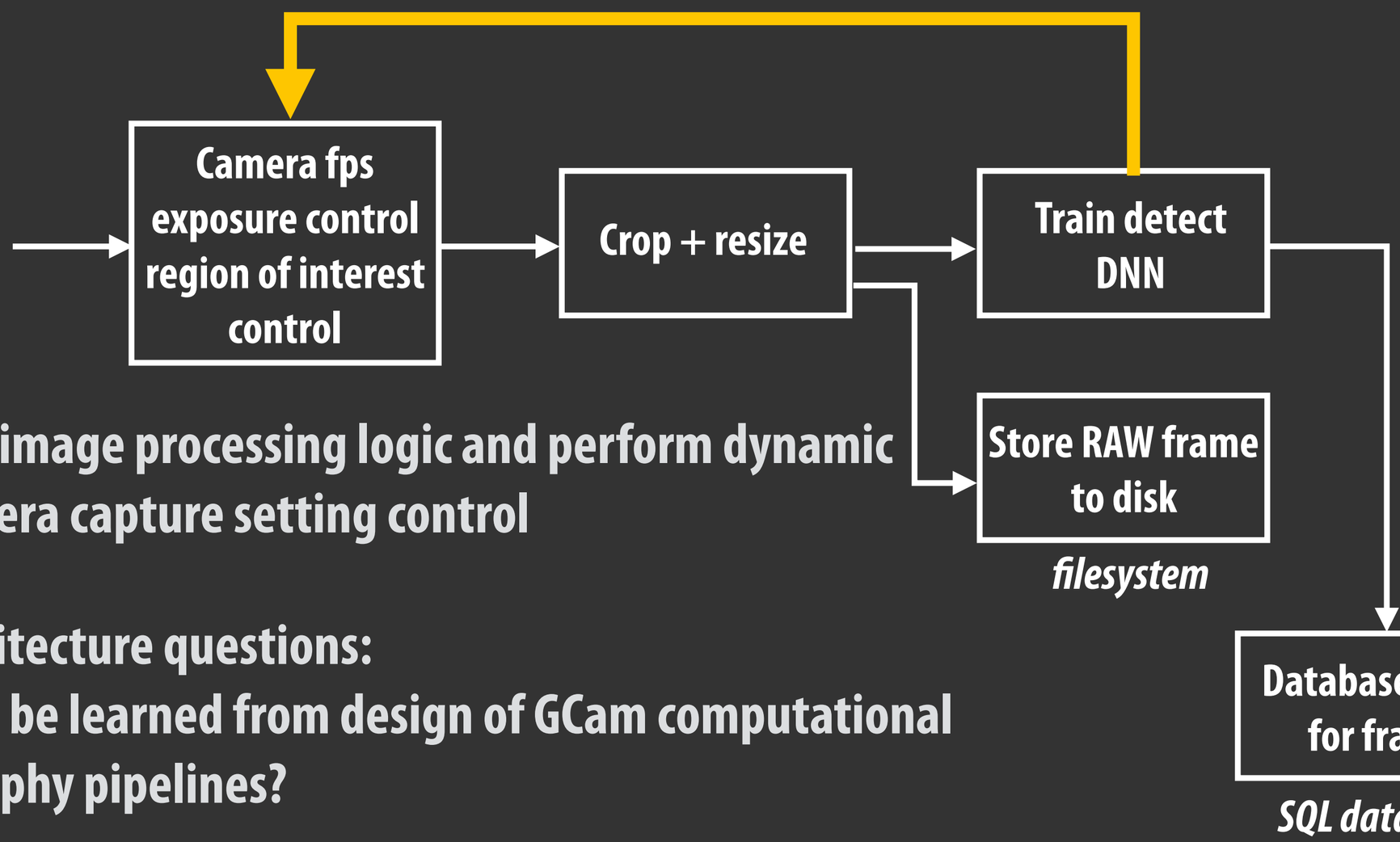
CPU core | CPU core | CPU core | CPU core

Intel i7-6770HQ (Skull Canyon NUC)

Servers in CMU datacenter

Work by:
Andersen
Canel
Kaminsky
Jiang
Xian

Streamer pipeline



- Specifies both image processing logic and perform dynamic control of camera capture setting control
- Streamer architecture questions:
  - What can be learned from design of GCam computational photography pipelines?
- Streaming implementation questions:
  - Many of the same algorithmic opportunities as Scanner apps (what frames to pay attention to? How to exploit temporal context?)
  - New forms of video compression: Learn camera-viewpoint specific compression?
  - Edge-to-cloud scheduling: What decisions should be made automatically by the system and which decisions must be made by the programmer?

Big visual computing systems needs

1. **Techniques for efficiently mapping image analysis algorithms to accelerated computing platforms**
   (Efficiently generating kernels for CPUs, GPUs, FPGAs, ASICs)

2. **Distributed computing support for scalable accelerated computing**
   (Connecting efficient processing pipelines to data stores, distribution across many machines)

3. **Performance-centric algorithmic innovation/approximation**
   (New work efficient algorithms and approximations)

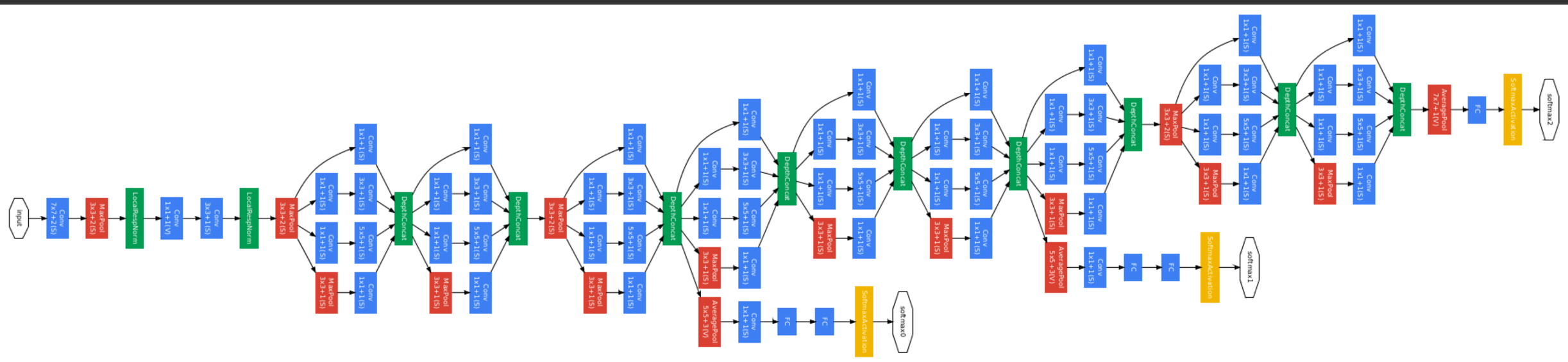
4. **Good abstractions for authoring scalable visual data analysis applications**
   (Considering higher-level primitives for authoring future applications e.g., SQL for video DBs?)

Generating efficient pixel-processing code for CPUs/GPUs/accelerators:

Scheduling image analysis pipelines

with Ravi Mullapudi (CMU), Andrew Adams (Google), Dillon Sharlet (Google), Jonathan Ragan-Kelley (Stanford)

Code generation for deep learning



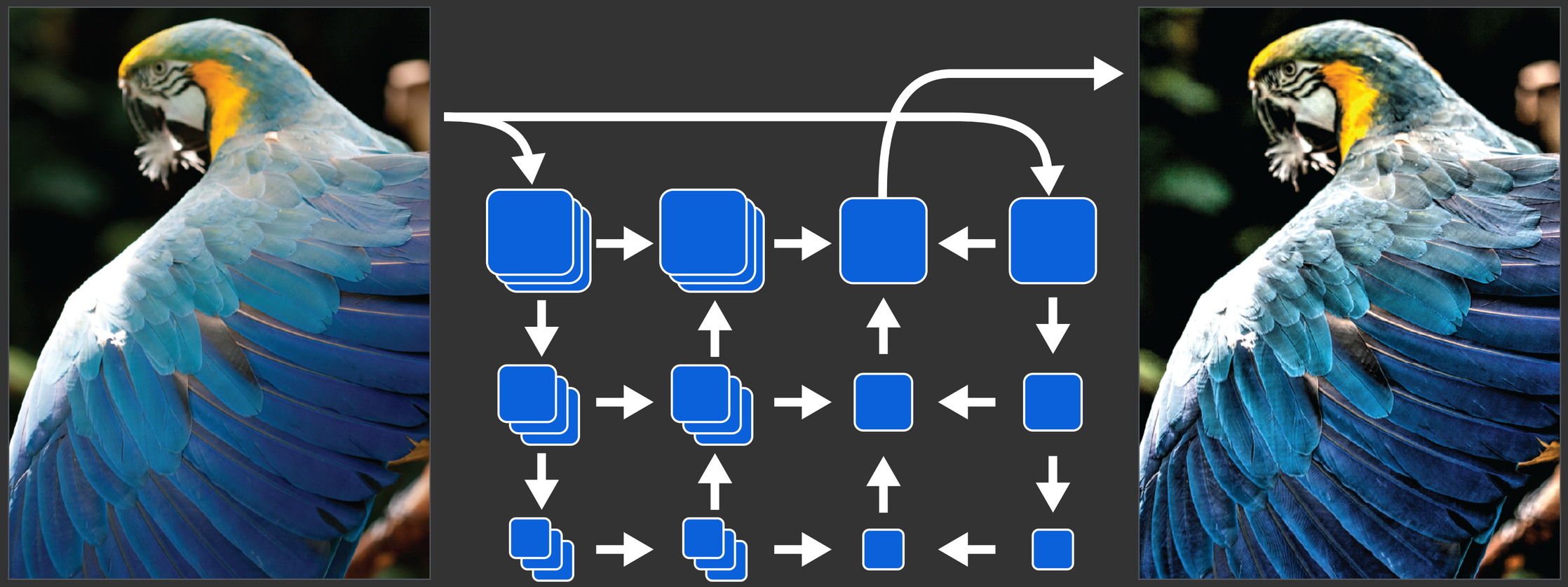
Trend: new compiler intermediate representations (IR) for optimization of deep learning data flow graphs

TensorFlow
Google XLA

nervana
Graph Compiler

mxnet
NNVM / TVM

Real-world computational photography pipelines are complex dataflow graphs



100 stages

Local Laplacian filter
[Paris 2010, Aubry 2011]

Google Nexus HDR+ mode: over 2000 stages!

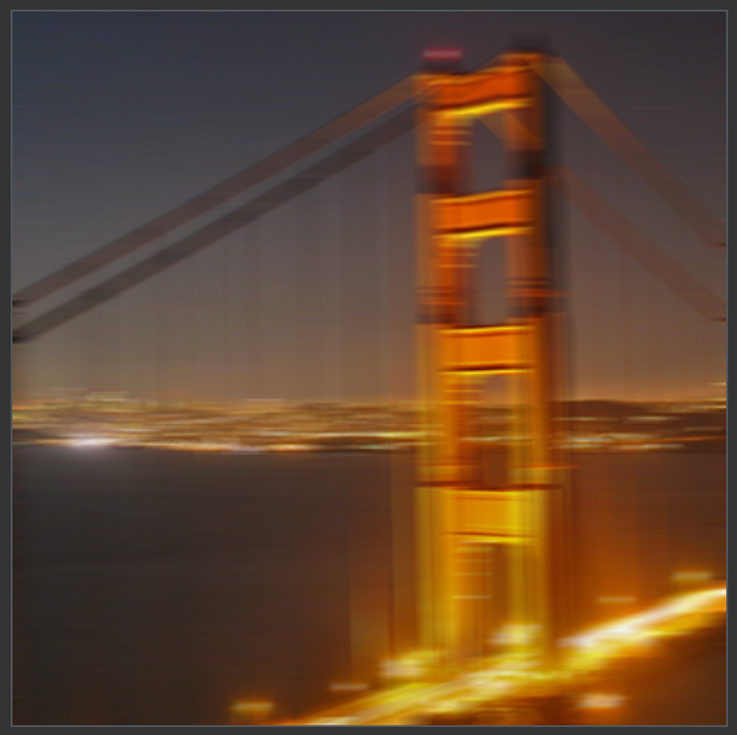
Halide DSL

[Ragan-Kelley, Adams et al. 2012]

Raised level of abstraction for developing high-performance image processing algorithms

```
blurx(x,y) = (in(x-1,y)    + in(x,y)    + in(x+1,y))    / 3;
out(x,y)   = (blurx(x,y-1) + blurx(x,y) + blurx(x,y+1)) / 3;
```

in



blurx

out

Halide DSL

[Ragan-Kelley, Adams et al. 2012]

Raised level of abstraction for developing high-performance image processing algorithms

Functional pipeline description:

```
blurx(x,y) = (in(x-1,y)    + in(x,y)    + in(x+1,y))    / 3;
out(x,y)   = (blurx(x,y-1) + blurx(x,y) + blurx(x,y+1)) / 3;
```

Schedule: DSL for mapping pipeline stages to a parallel machine

```
output.tile(x, y, xi, yi, 256, 32);

output.vectorize(xi, 8);
output.parallelize(y);

blurx.compute_at(xi);
blurx.vectorize(x, 8);
```

compute output in tiled order

vectorize innermost loop

parallelize loop across cores

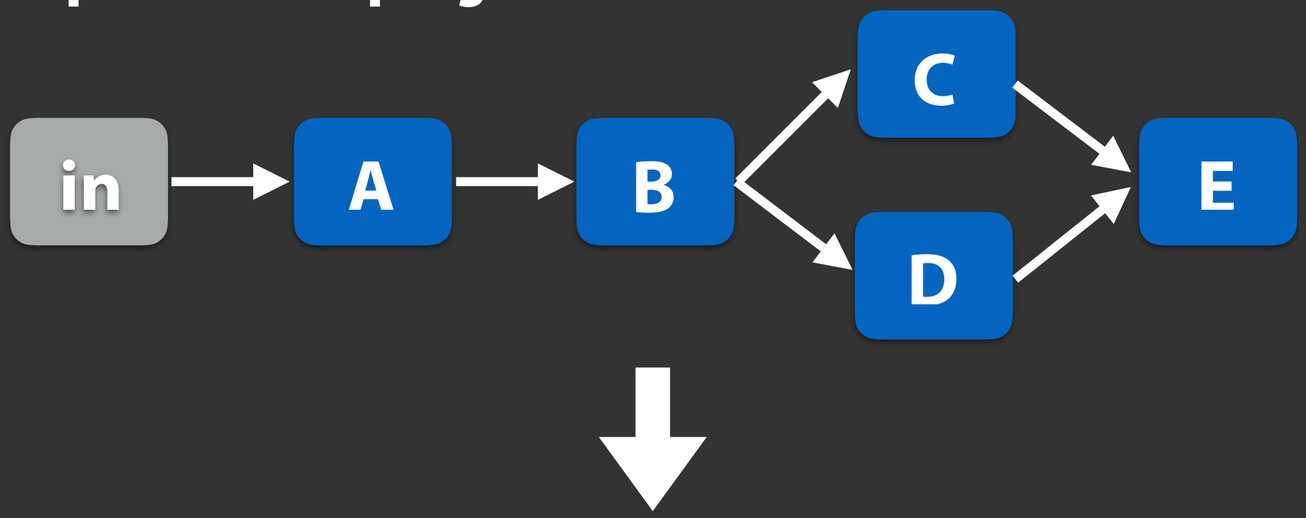
loop fusion

vectorize innermost loop

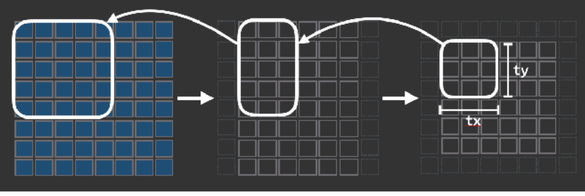
[Mullapudi 2016]

Automatically scheduling Halide

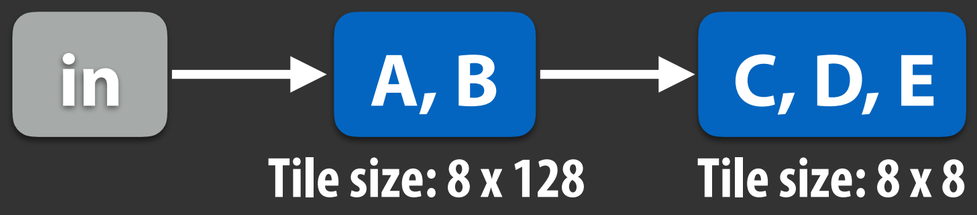
Input: Halide program DAG



Halide Autoscheduler



Output: optimized schedule

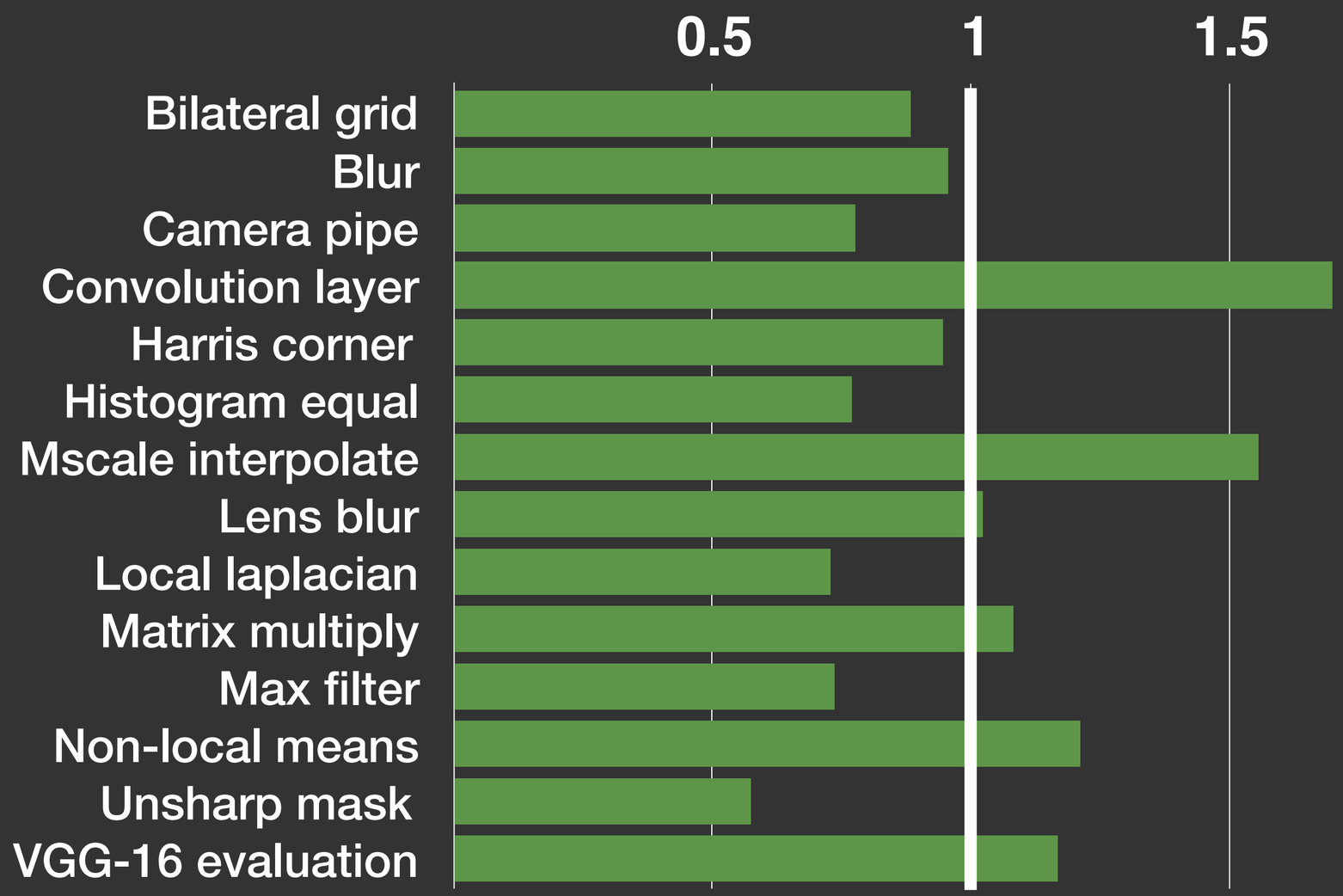



```
for each 8x128 tile in parallel
    vectorize compute required pixels of A
              unroll x by 4
    vectorize compute required pixels of B
    vectorize compute pixels in tile of D

for each 8x8 tile in parallel
    vectorize compute required pixels of C
              unroll y by 2
    vectorize compute pixels in tile of E
```

Autoscheduled Halide performs now comparably to experts

Performance relative to expert schedules

(6-core Xeon CPU)

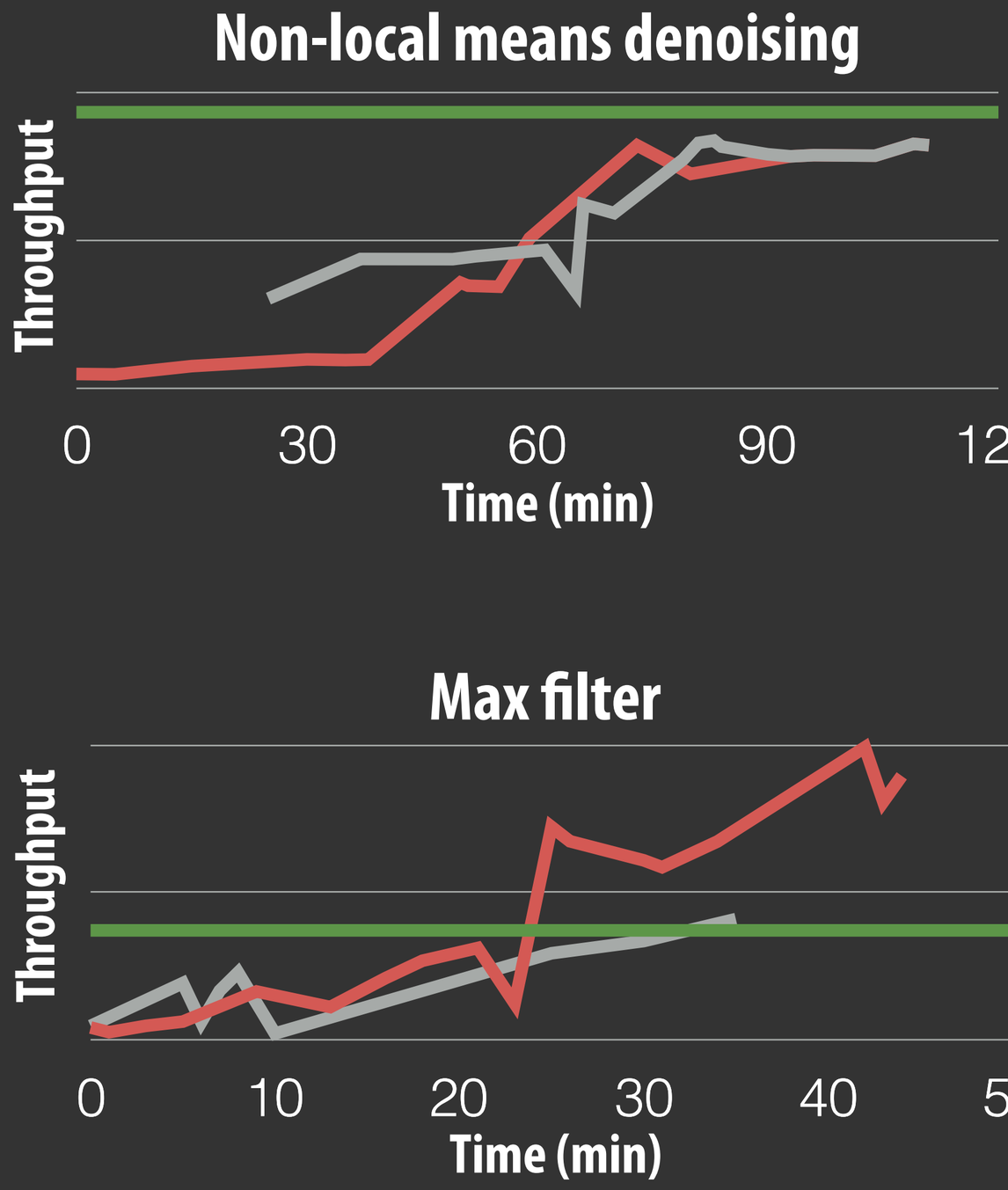


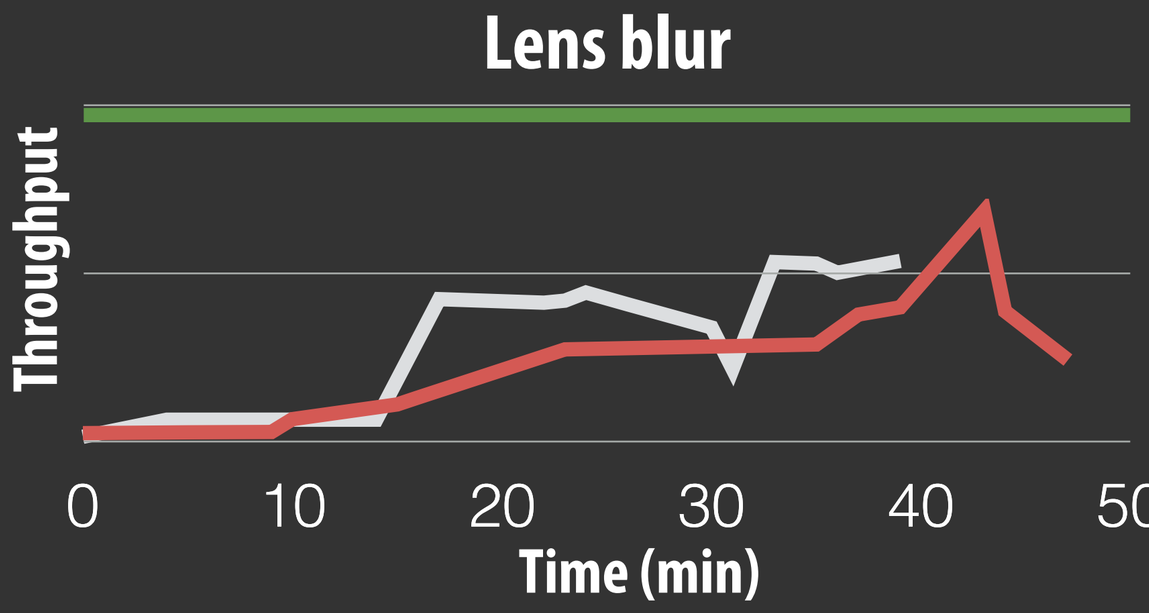
On 8 of the 14 benchmarks performance within 10% of experts or better

Auto scheduler

[Mullapudi 2016]

Autoscheduler saves time for experts





[Mullapudi 2016]

What can we contribute to scheduling DNN frameworks?

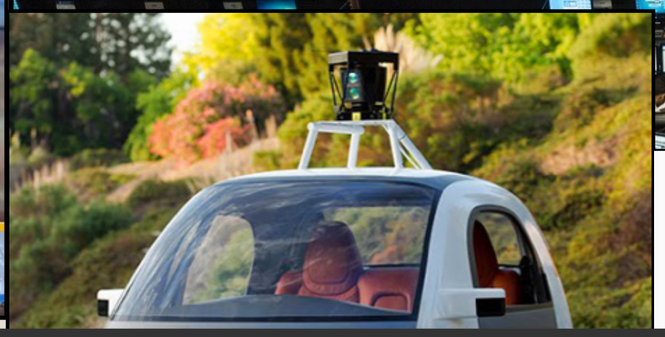
Google XLA

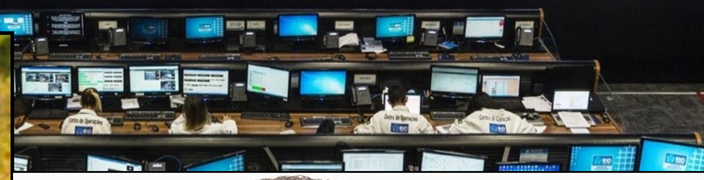
nervana
Graph Compiler

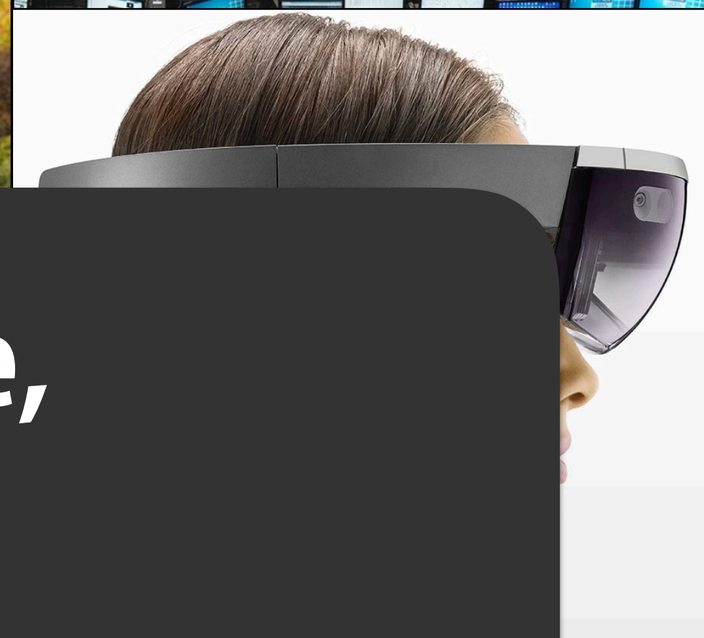
mxnet

NNVM / TVM

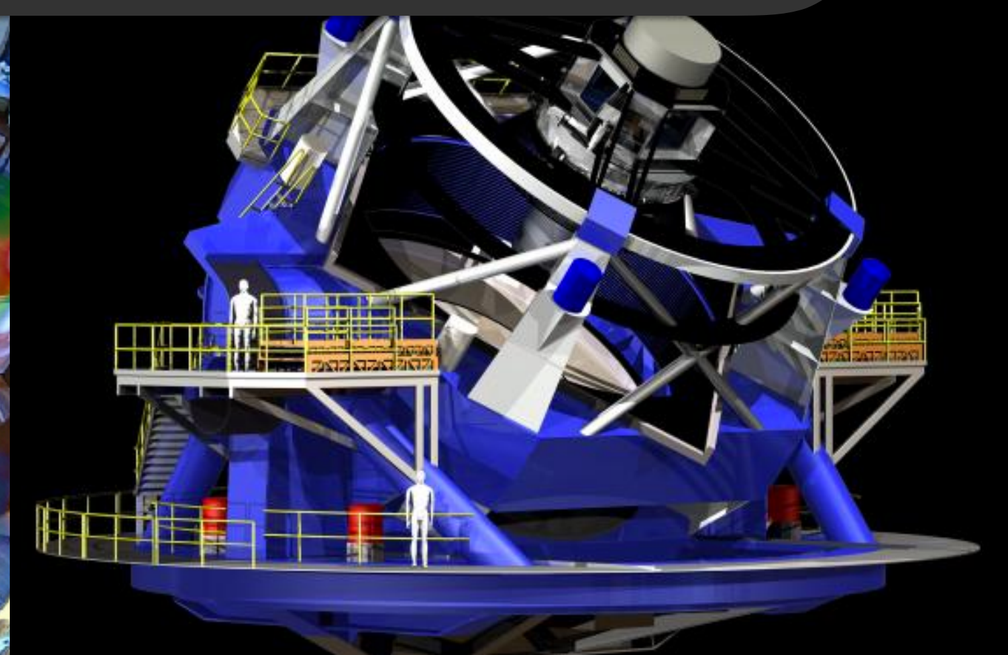
- New challenges that do not exist in Halide:
  - Stateful computation (recurrent networks)
  - Data-dependent execution
  - Auto-differentiation service
  - Expect diversity in DNN hardware accelerators







How do we create flexible, high-efficiency systems for analyzing the world's visual signal?



Rich space of high-impact applications

(space is being defined as we go!)

Applications convert new performance into new value

Use every flop systems can provide!

CPUs, GPUs, ASICs…

Large opportunities for performance-minded algorithm design (orders of magnitude available)

In addition to huge body of fundamental computer vision/AI/ML algorithms work to solve problems previously not solvable

Familiar need for domain-specific programming abstractions to impose useful structure (for productivity and performance)

Thank you

Collaborators:

Alex Poms
Ravi Mullapudi
Krishna Kumar Singh
Karima Ma
Ran Xian
Satya Tangirala
Christopher Canel
Angela Jiang
Tomas Kim
Hanbyul Joo
Dave Andersen
Srinivas Narasimhan
Yaser Sheikh
(CMU)

Will Crichton, Jonathan Ragan-Kelley,
Maneesh Agrawala, Pat Hanrahan (Stanford)
Alyosha Efros (Berkeley)
Andrew Adams, Dillon Sharlet, Bill Mark (Google)
Matt Perron, Dulloor Subramanya, Michael Kaminsky (Intel)

Support:
Intel Science and Technology Center for Visual Cloud Systems (ISTC-VCS)
Intel VEC, ML ISRAs
National Science Foundation
Google Faculty Fellowship
Heinz Foundation