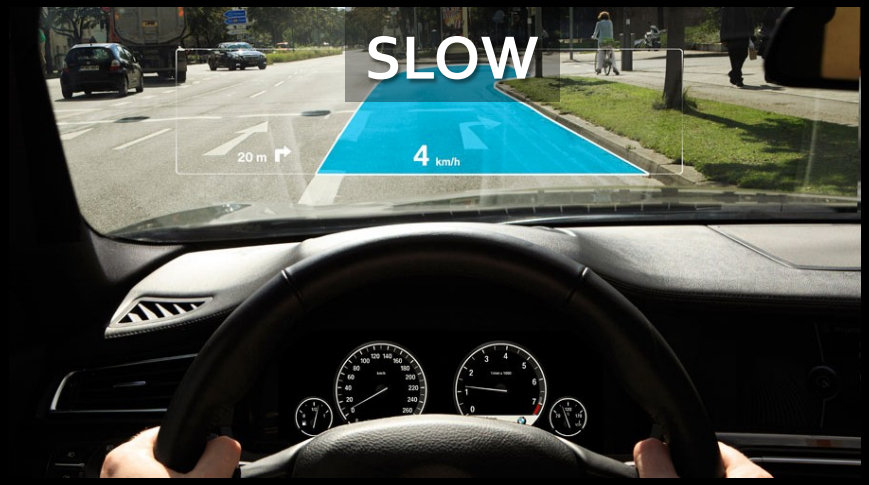
Glimpse

Continuous, Real-Time Object Recognition on Mobile Devices

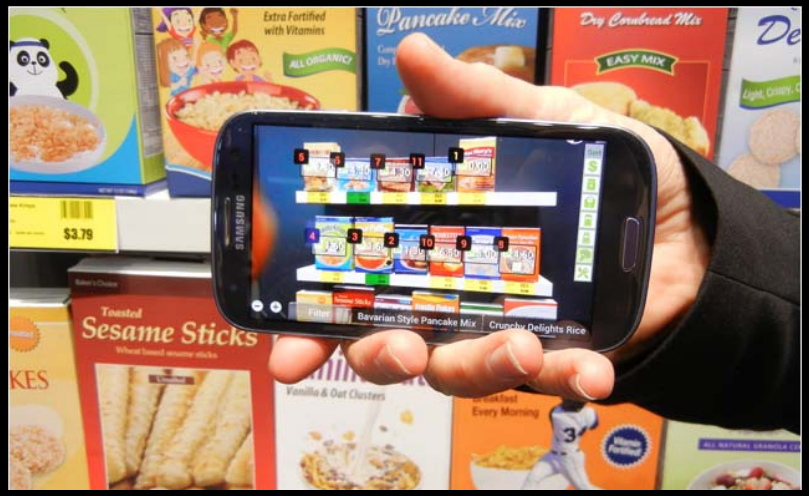
Tiffany Chen
Lenin Ravindranath
Shuo Deng
Victor Bahl
Hari Balakrishnan

CSAIL Microsoft Research

Continuous, Real-Time Recognition Apps



Driver Assistance



Augmented Reality Shopping

Face Recognition

Augmented Reality Tourist App

Today: Picture-Based Object Recognition

Today: Picture-Based Object Recognition

Today: Picture-Based Object Recognition

Video-Based Object Recognition

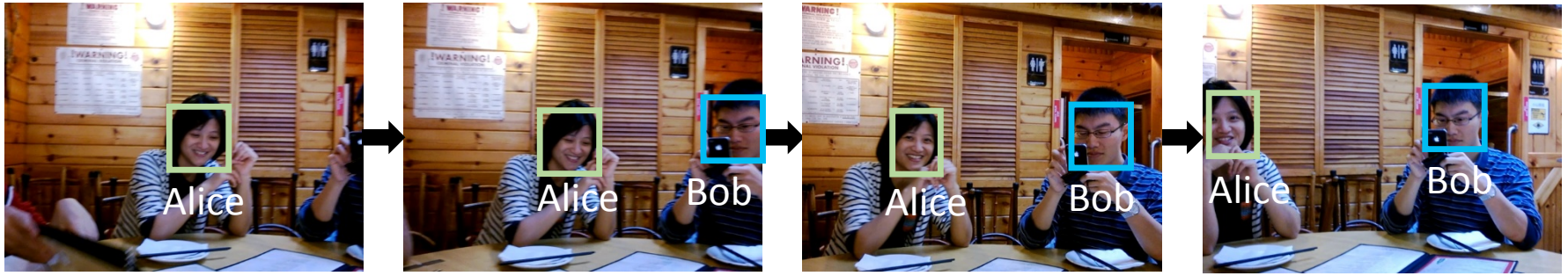
Video-Based Object Recognition

Glimpse

- Continuous, real-time object recognition on mobile devices in a video stream

Glimpse

- Continuous, real-time object recognition on mobile devices in a video stream
- Continuously ***identify*** and ***locate*** objects in each frame



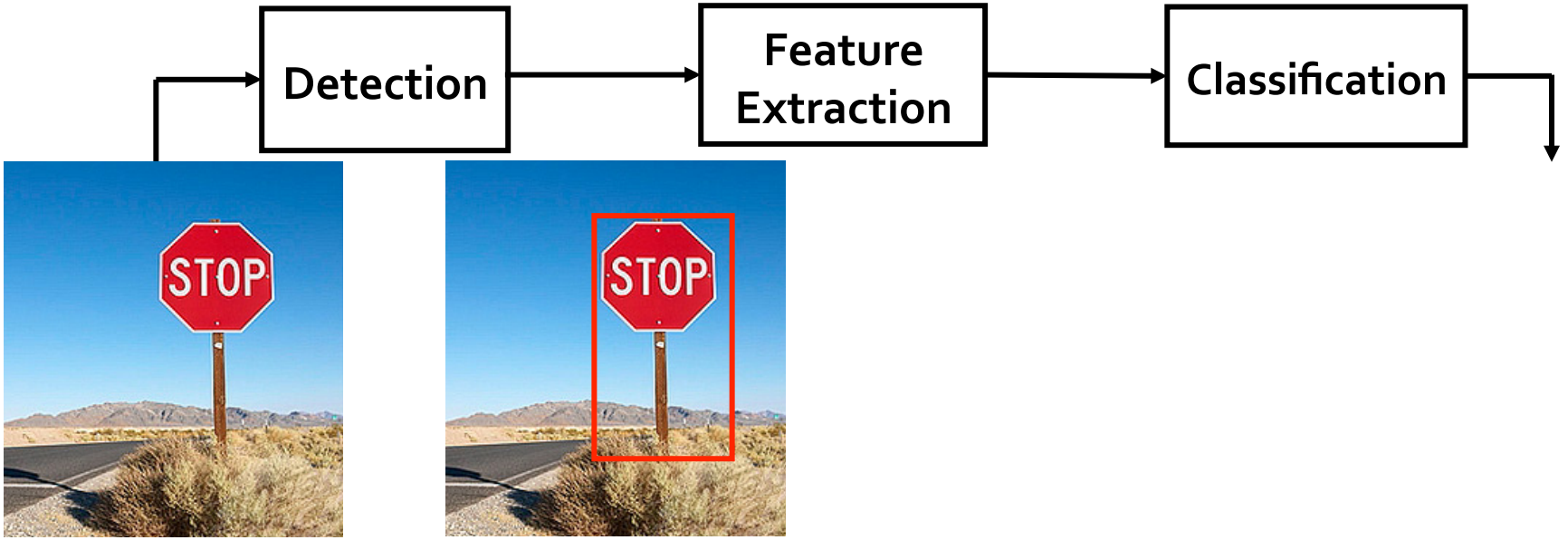
Object Recognition Pipeline

Object Recognition Pipeline

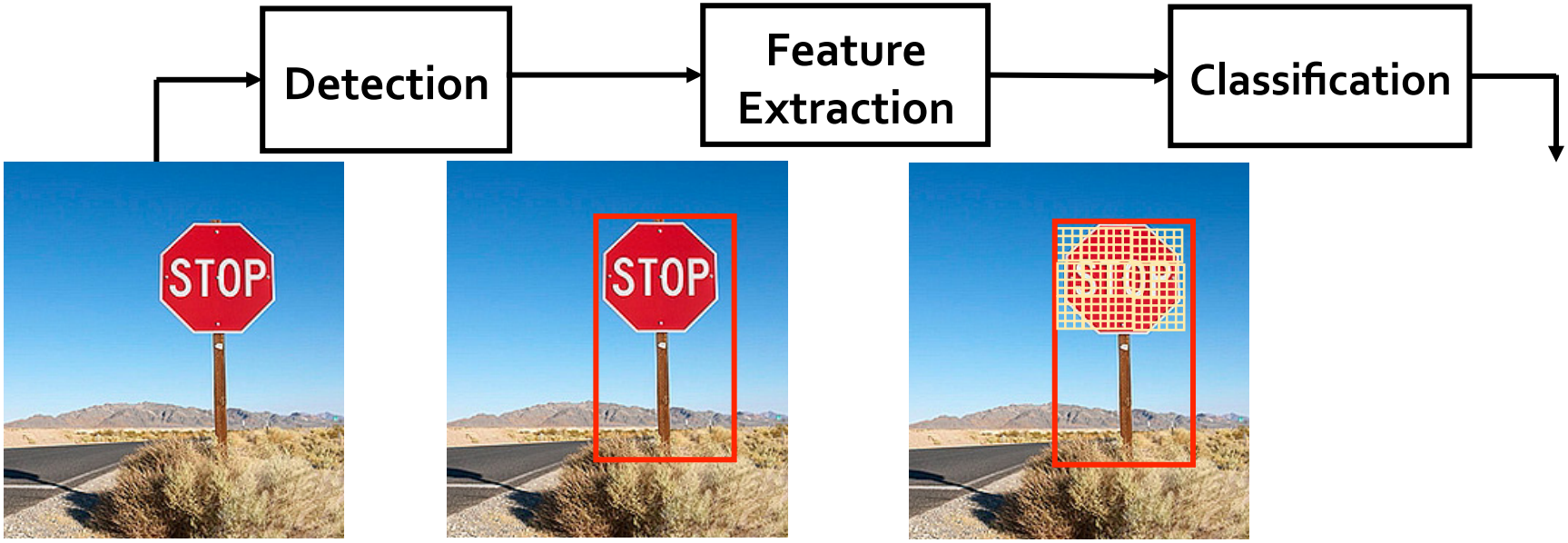
Object Recognition Pipeline

Detection → Feature Extraction → Classification

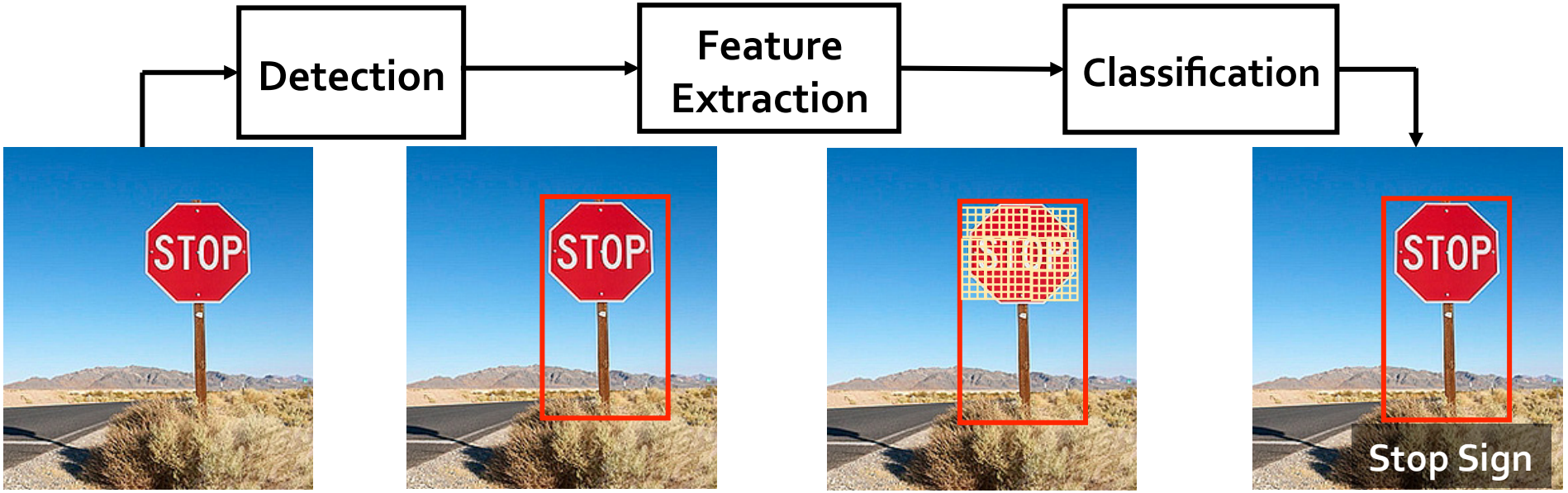
Object Recognition Pipeline



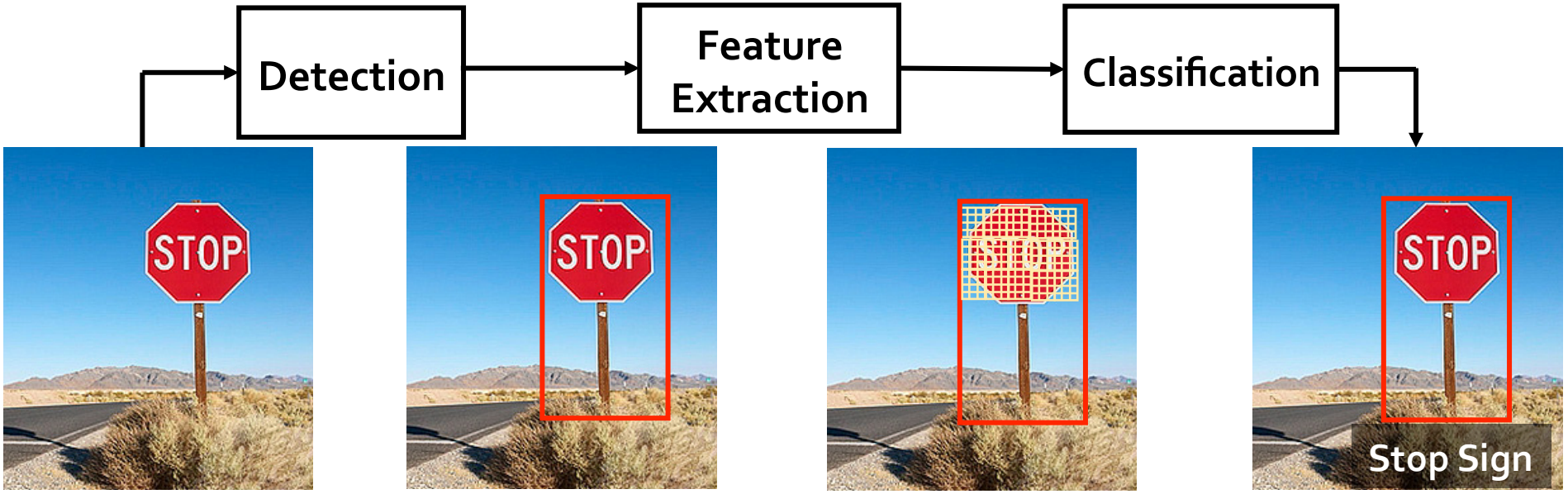
Object Recognition Pipeline



Object Recognition Pipeline



Object Recognition Pipeline



- Computationally expensive and memory-intensive
  - Server is 700x faster than Google Glass
  - Scalability
- We need to offload the recognition pipeline to servers

Object Detection

Feature Extraction

Object Recognition/Labeling

Wrong Location Tracking Issue

Frame 1 (t = 0 ms) Frame 20 (t = 660 ms)

Figure 1: Offloading every frame to a server reduces trackability (right): the stop sign's location is wrong.

Performance using Other Systems

Stage	Google Glass Execution Time (ms)	Mobile Client Execution Time (ms)	Server Execution Time (ms)	Model Memory Usage (MB)	Settings
Road Sign Detection [2]	-	2353 ± 242.4	110 ± 32.1	-	Server uses 4 cores.
Road Sign Feature Extraction	-	1327.73 ± 102.4	69 ± 15.2	0.21/object	Using convolutional neural networks with the BVLC GoogleNet model [29, 26, 51]. Server uses a GPU.
Road Sign Recognition	793.3 ± 102	162.1 ± 73.2	11 ± 1.6	0.03/object	Using linear SVM [18] to classify 1K objects with 4K features. Server uses a GPU.
OpenCV Face Detection	3130.18 ± 800.1	2263.71 ± 478.15	197.77 ± 10.56	0.89	Using the frontal face classifier [56, 55]; the minimum size of the detected face is 30×30 pixels.
FaceSDK Face Detection	-	1129 ± 239.5	92.26 ± 21.79	0.12	Mobile client: Nokia Lumia 928.
Hardware-based Face Detection [4]	-	174.6 ± 70.0	-	-	Mobile client: HTC One M8.
Facial Feature Extraction	309.8 ± 101.2	84.55 ± 25.57	19 ± 3.15	35	Extracting 57K features around 27 landmarks [9, 10].
Face Recognition	2912.3 ± 448	537.8 ± 224.1	41.13 ± 3.11	1.25/object	Using linear SVM to classify 224 objects with 57K features.
Tracking	43.22 ± 9.1	37.7 ± 11.5	1.2 ± 0.4	0	Lucas-Kanade tracking with 3 pyramid levels and 30 feature points [35].

Table 2: Performance of object detection, feature extraction, recognition, and tracking. Unless stated otherwise, the Mobile Client is a Samsung Galaxy Nexus Android smartphone and Server is a Intel Core i7 with 3.6GHz, 4-core CPU. The performance is averaged across 1293 frames with resolution 640 × 480.

Performance Degradation

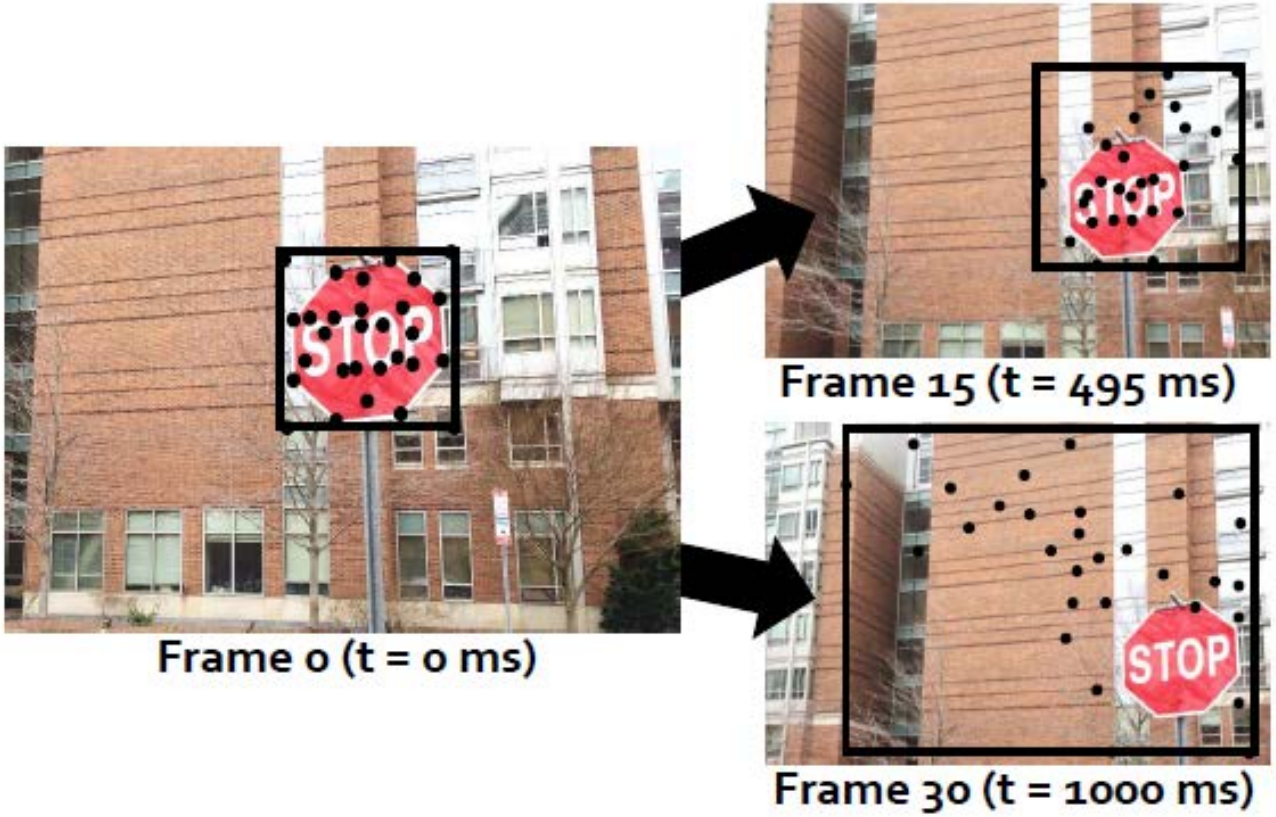
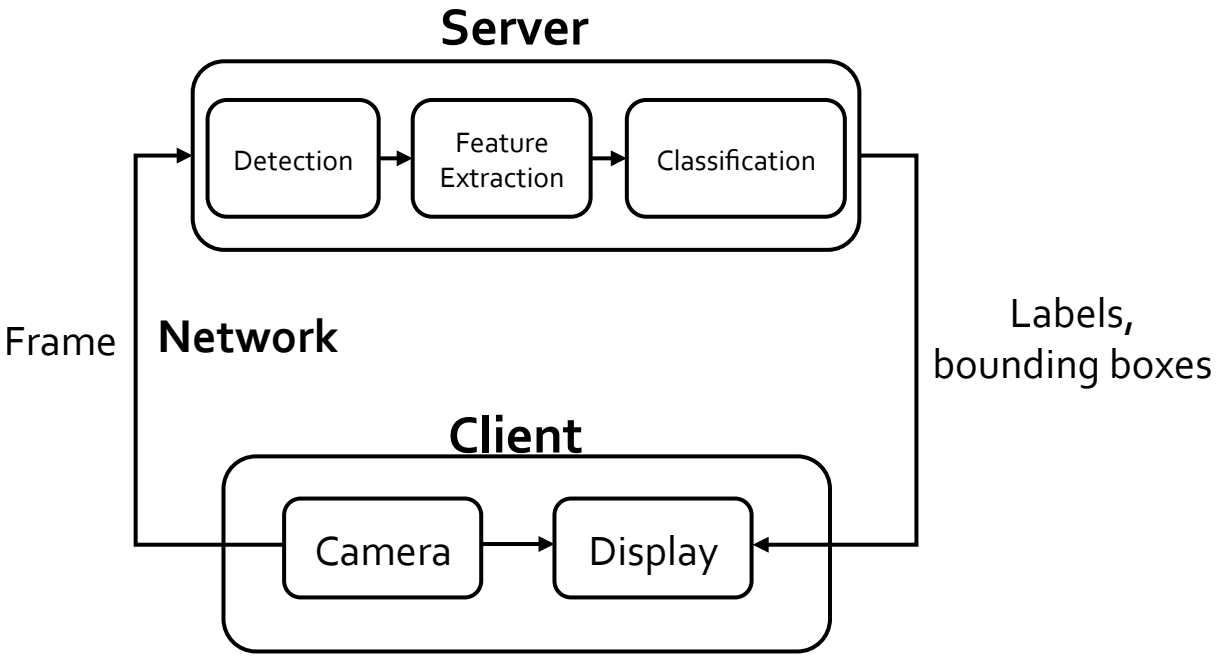
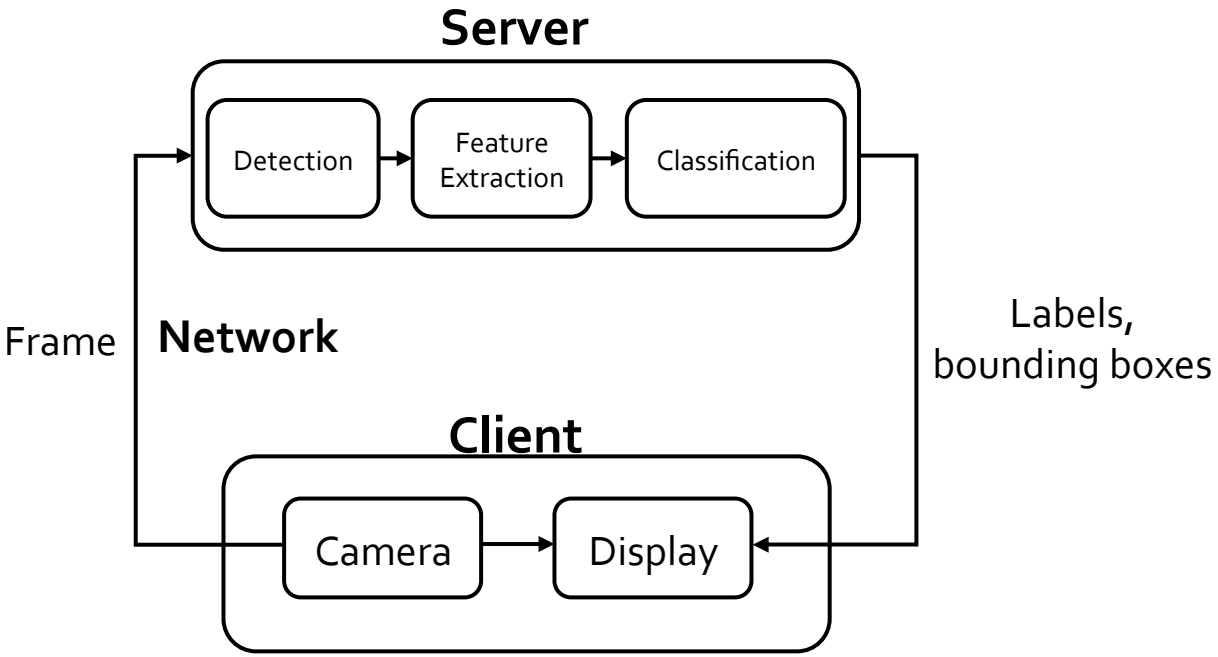


Figure 5: Tracking performance degrades as the displacement of the object increases.

Client-Server Architecture



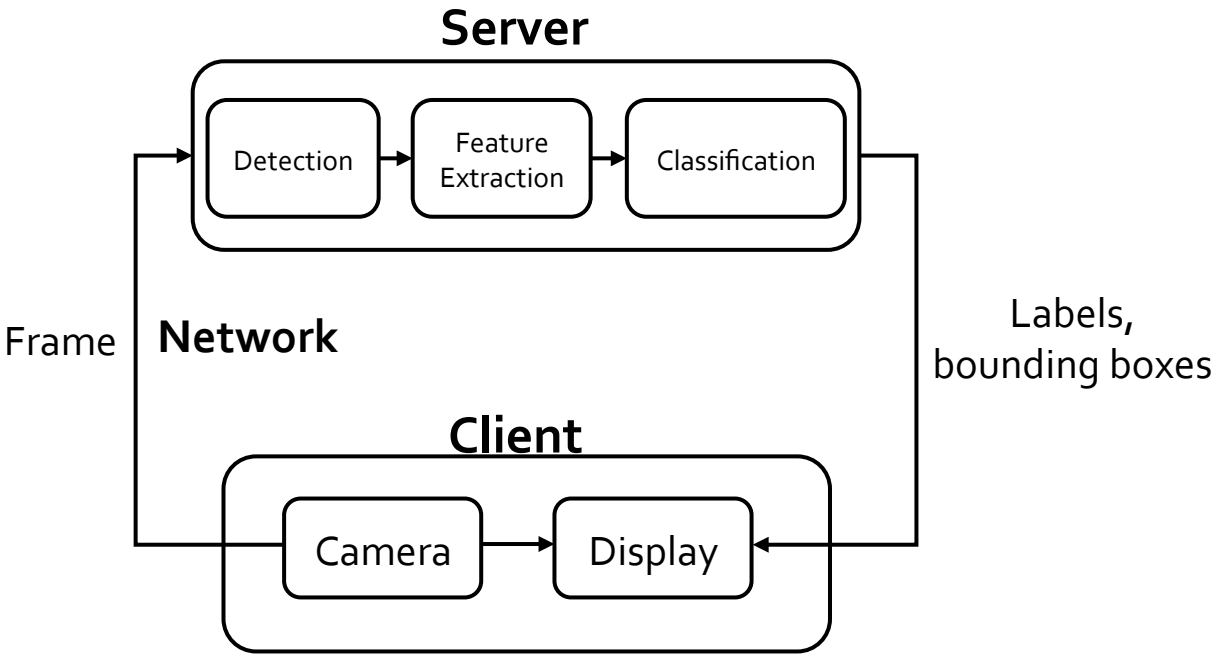
Client-Server Architecture



Challenges

1. **End-to-end latency** lowers object recognition accuracy

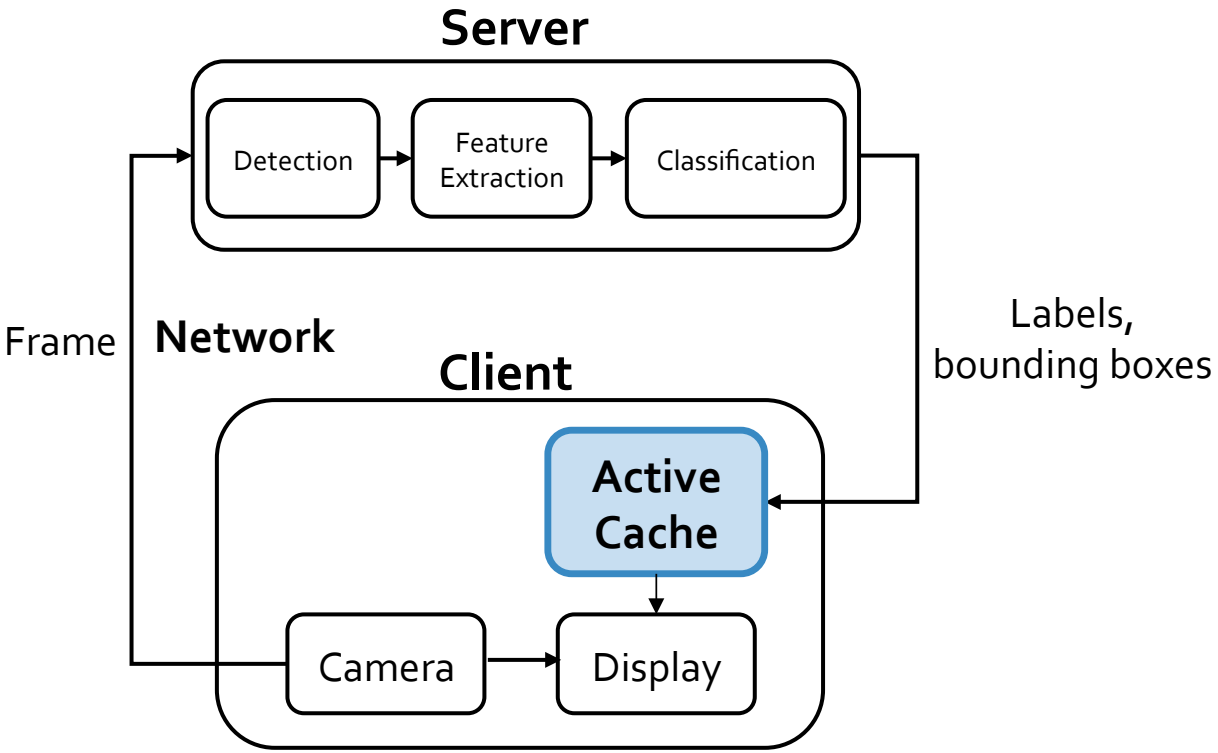
Client-Server Architecture



Challenges

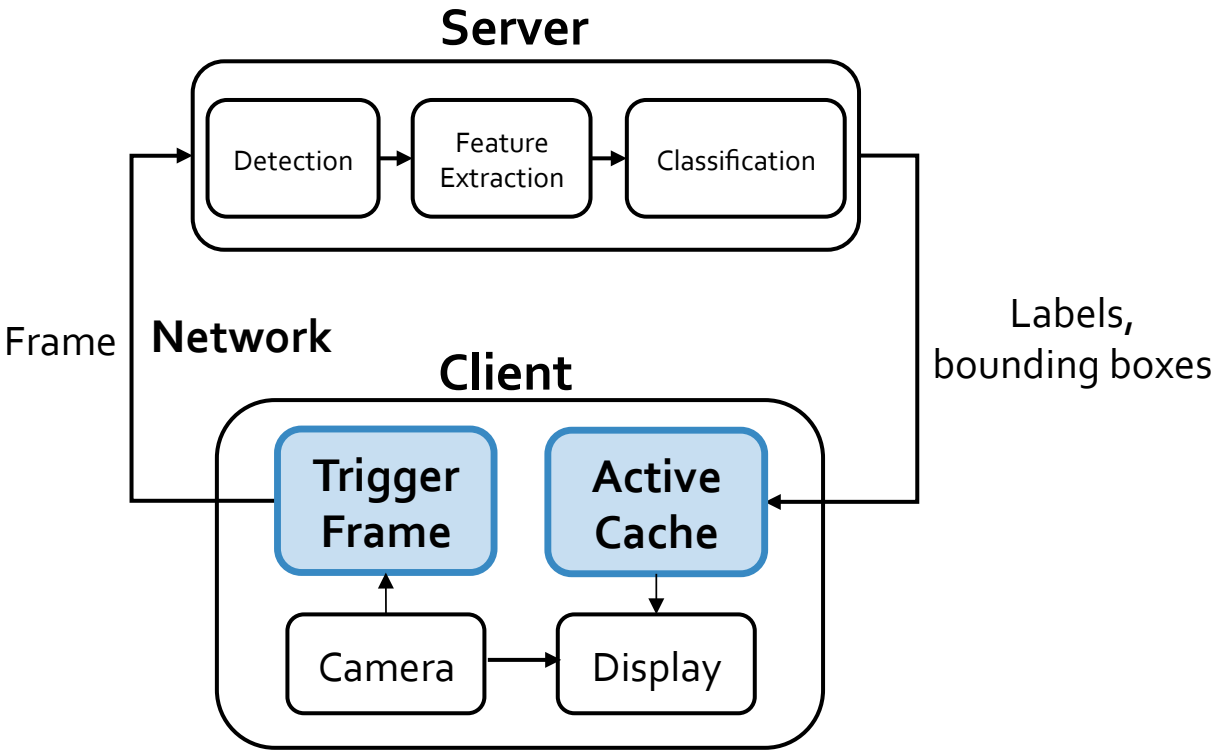
1. **End-to-end latency** lowers object recognition accuracy
2. **Bandwidth and battery** efficiency

Glimpse Architecture



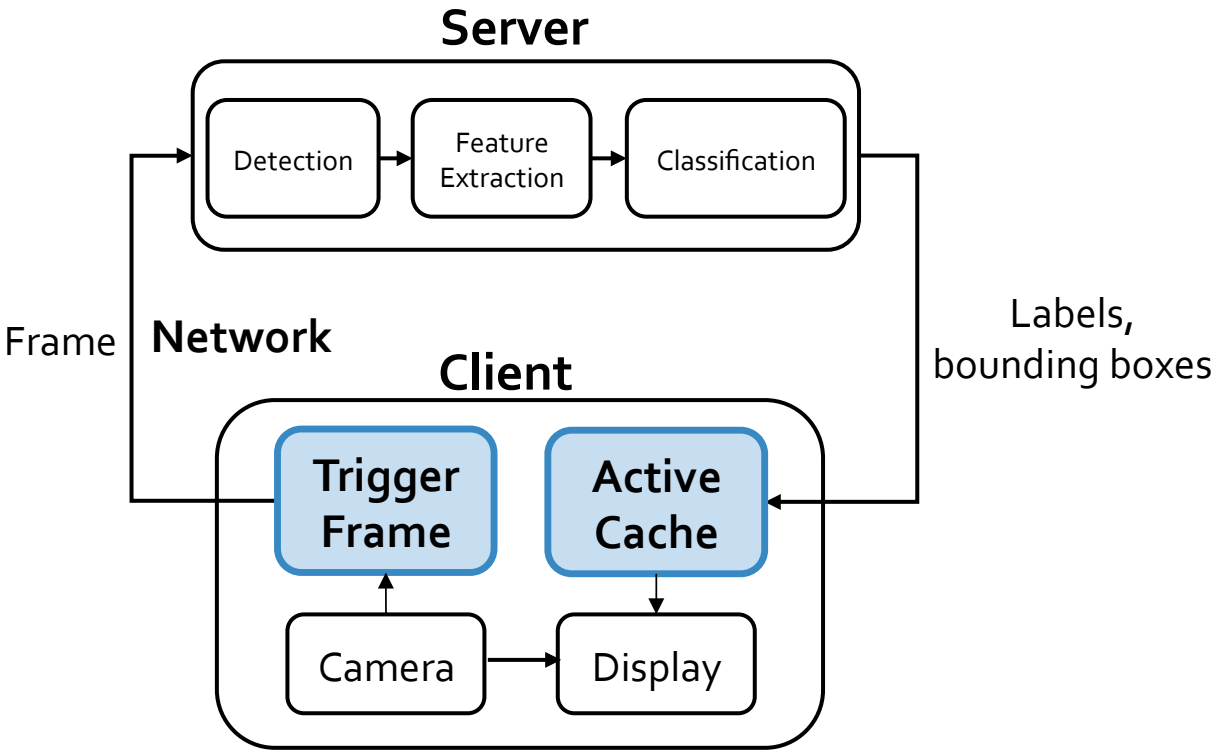
1. **Active Cache** combats e2e latency and regains accuracy

Glimpse Architecture



1. **Active Cache** combats e2e latency and regains accuracy
2. **Trigger Frame** reduces bandwidth usage

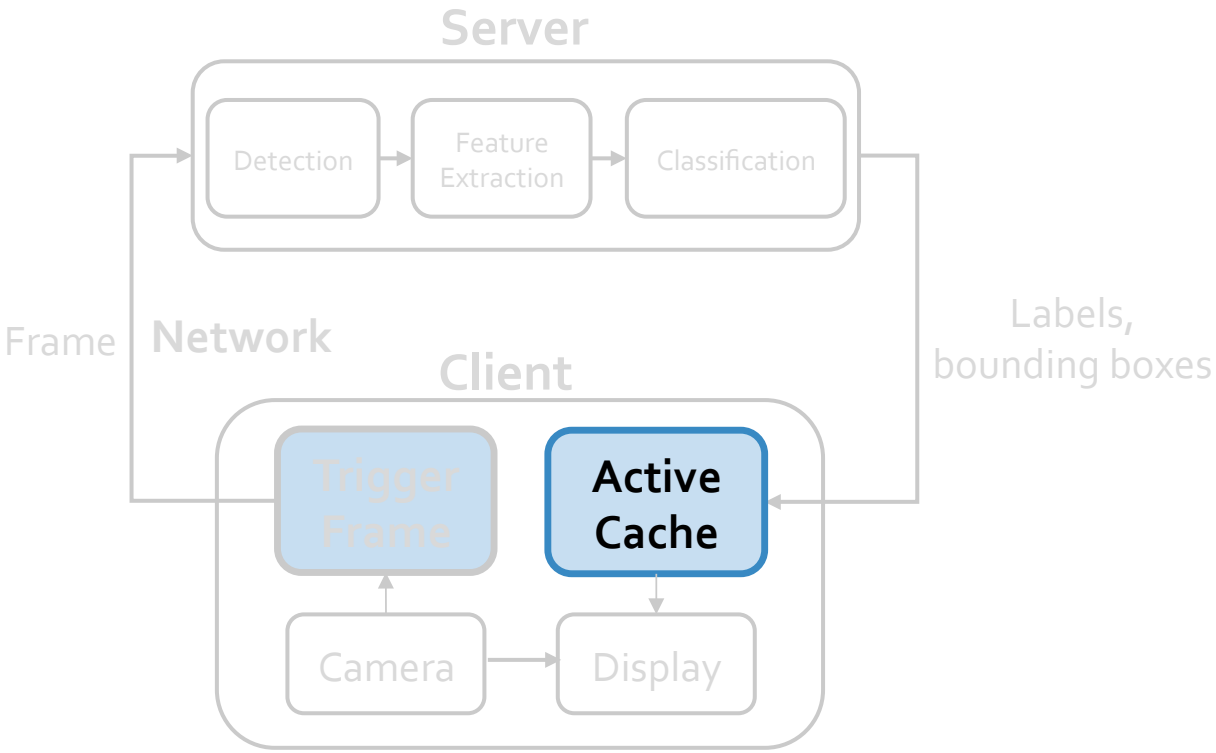
Glimpse Architecture



1. **Active Cache** combats e2e latency and regains accuracy
2. **Trigger Frame** reduces bandwidth usage

Wishbone [NSDI'09], Maui [MobiSys'10], Odessa [MobiSys'11], Gabriel [MobiSys'13]

Glimpse Architecture



1. **Active Cache** combats e2e latency and regains accuracy

Relocate Moving Object with Tracking

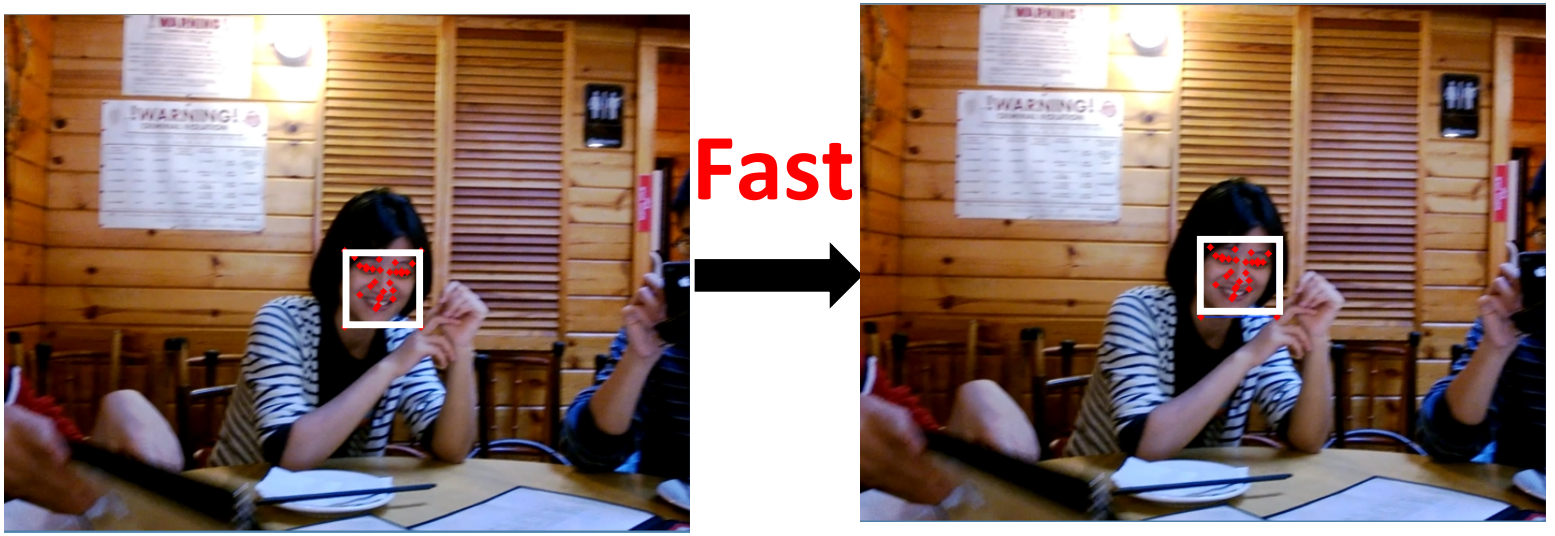
- Object tracking on the client to re-locate the object

Frame 0

Frame 12 (delay = 360 ms)

Relocate Moving Object with Tracking

- Object tracking on the client to re-locate the object



Frame 0

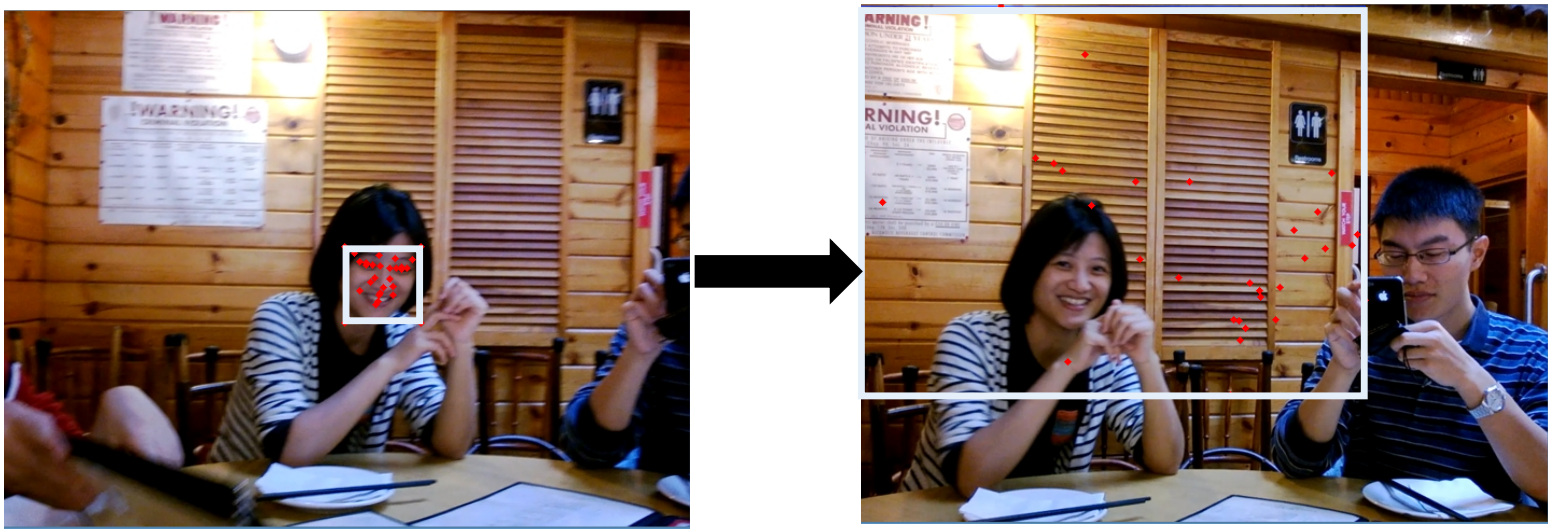
Frame 12 (delay = 360 ms)

Relocate Moving Object with Tracking

- Object tracking on the client to re-locate the object
- Fails to work when object displacement is large

Relocate Moving Object with Tracking

- Object tracking on the client to re-locate the object
- Fails to work when object displacement is large



Frame 0

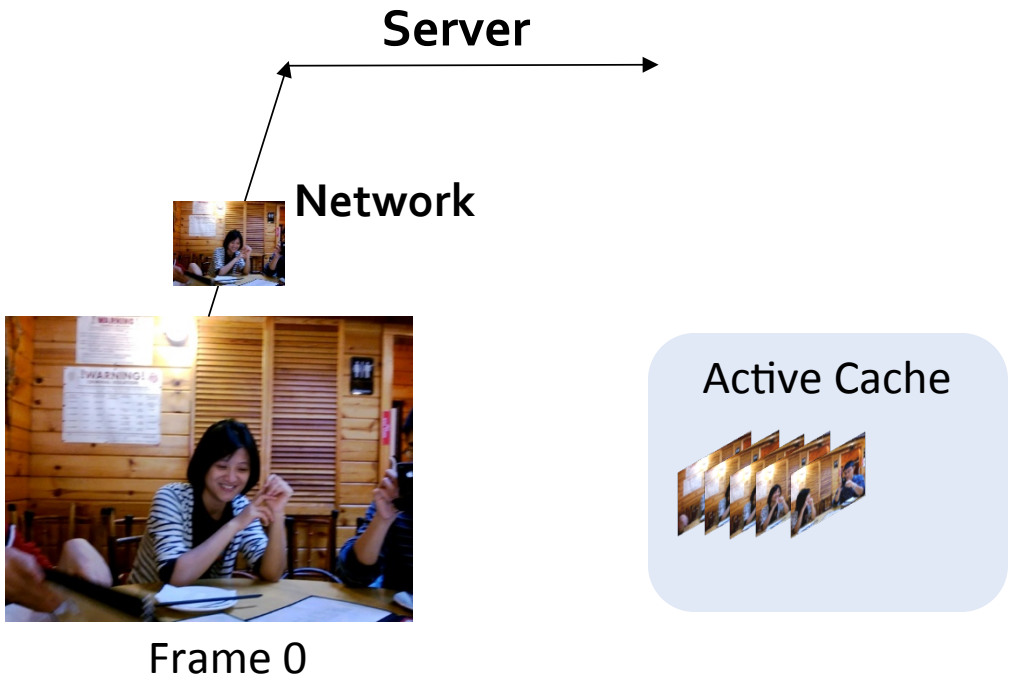
Frame 30 (delay= 1 sec)

Regain Accuracy with *Active Cache*

- Cache and run tracking through the cached frames

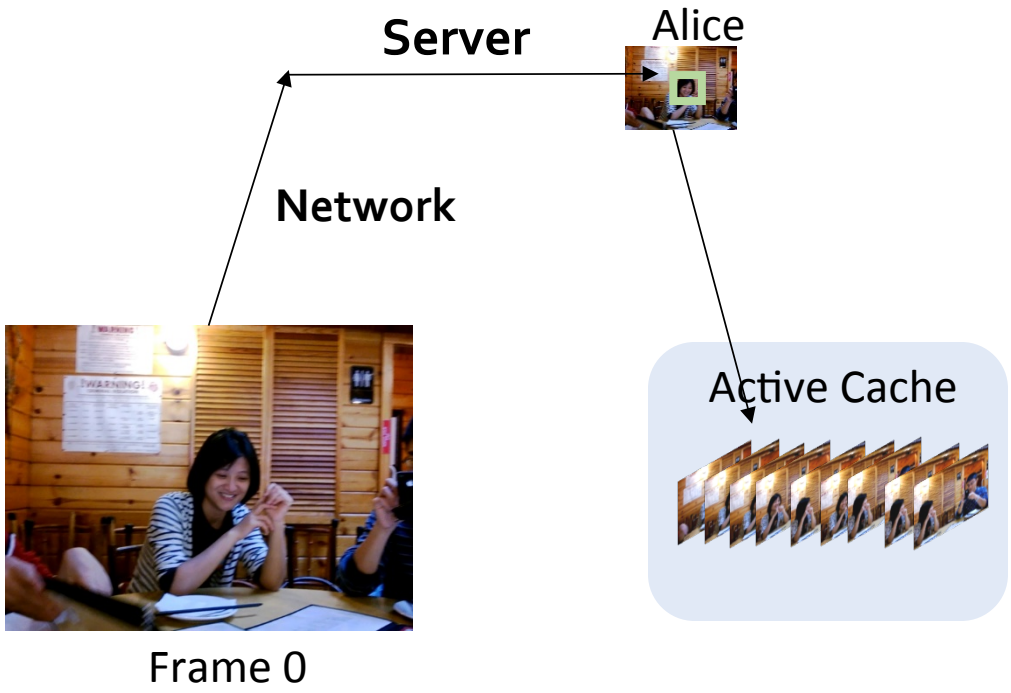
Regain Accuracy with *Active Cache*

- Cache and run tracking through the cached frames



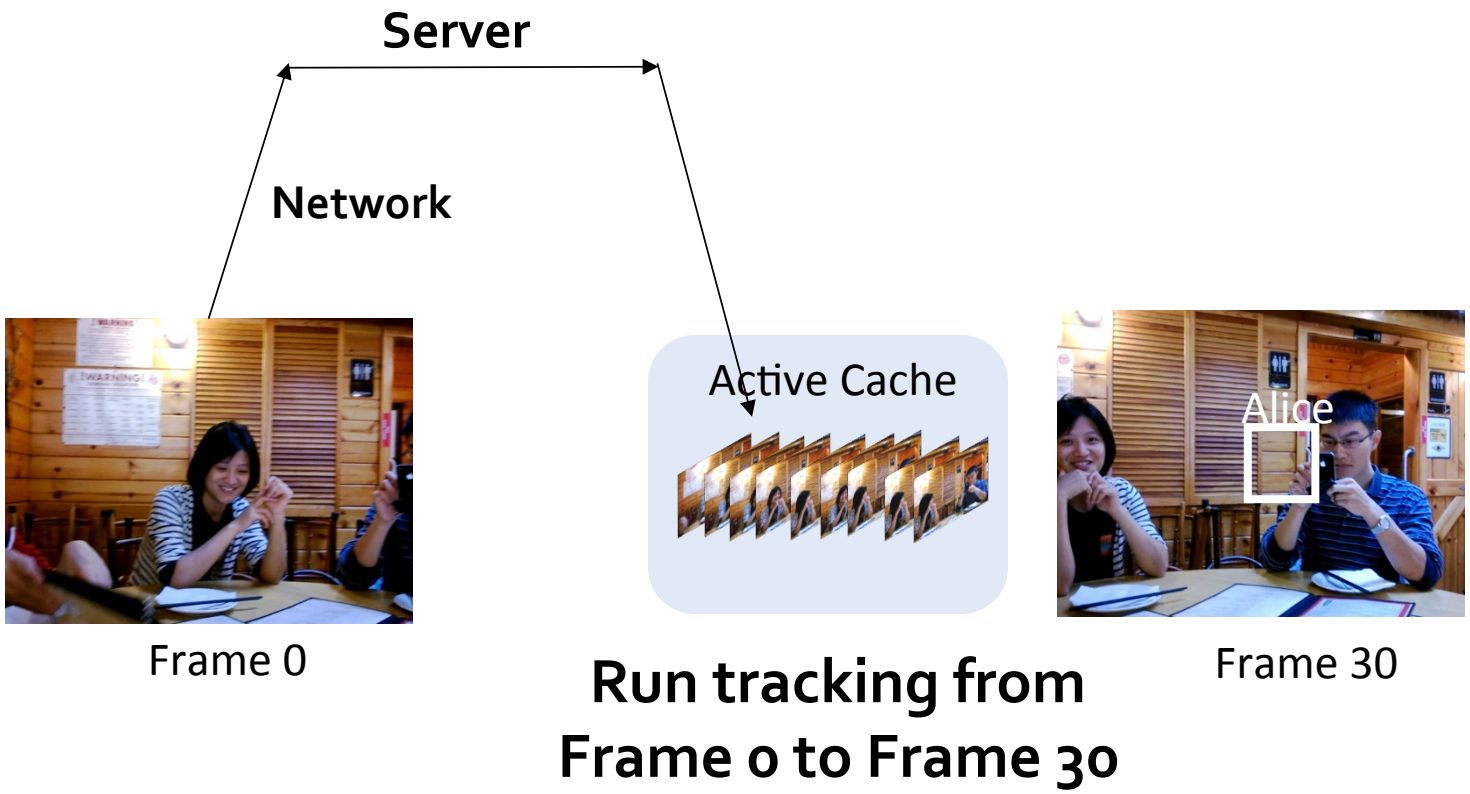
Regain Accuracy with *Active Cache*

- Cache and run tracking through the cached frames



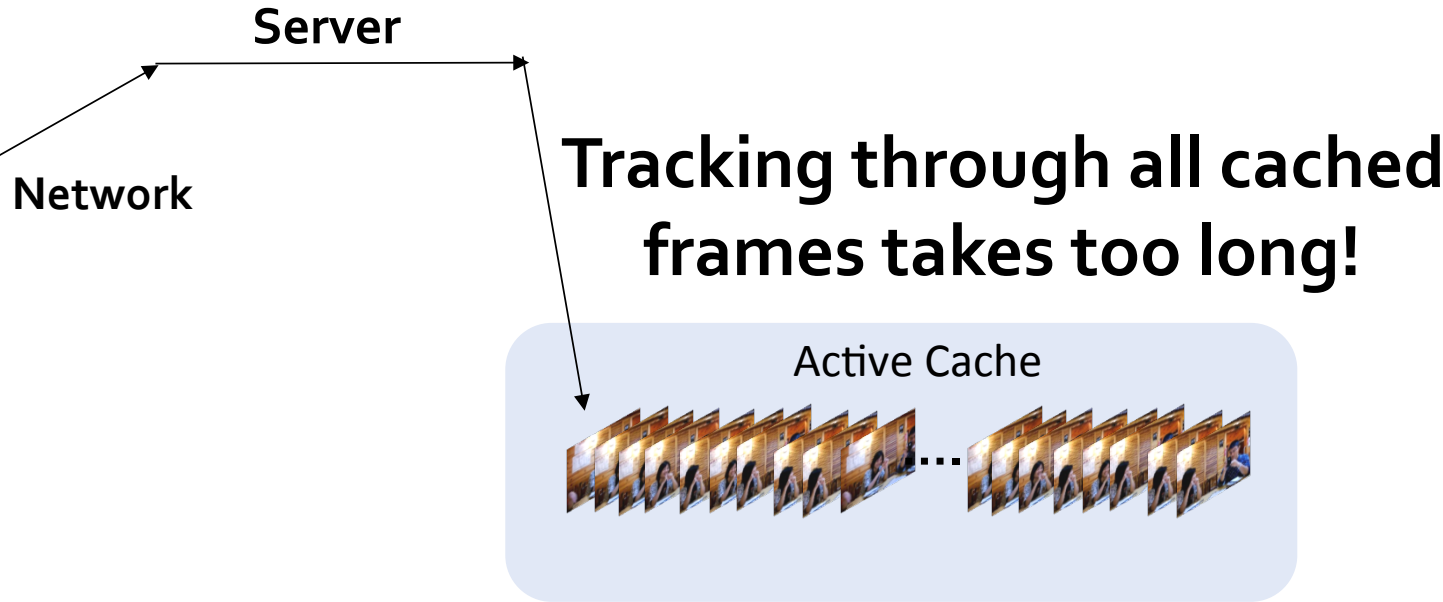
Regain Accuracy with *Active Cache*

- Cache and run tracking through the cached frames



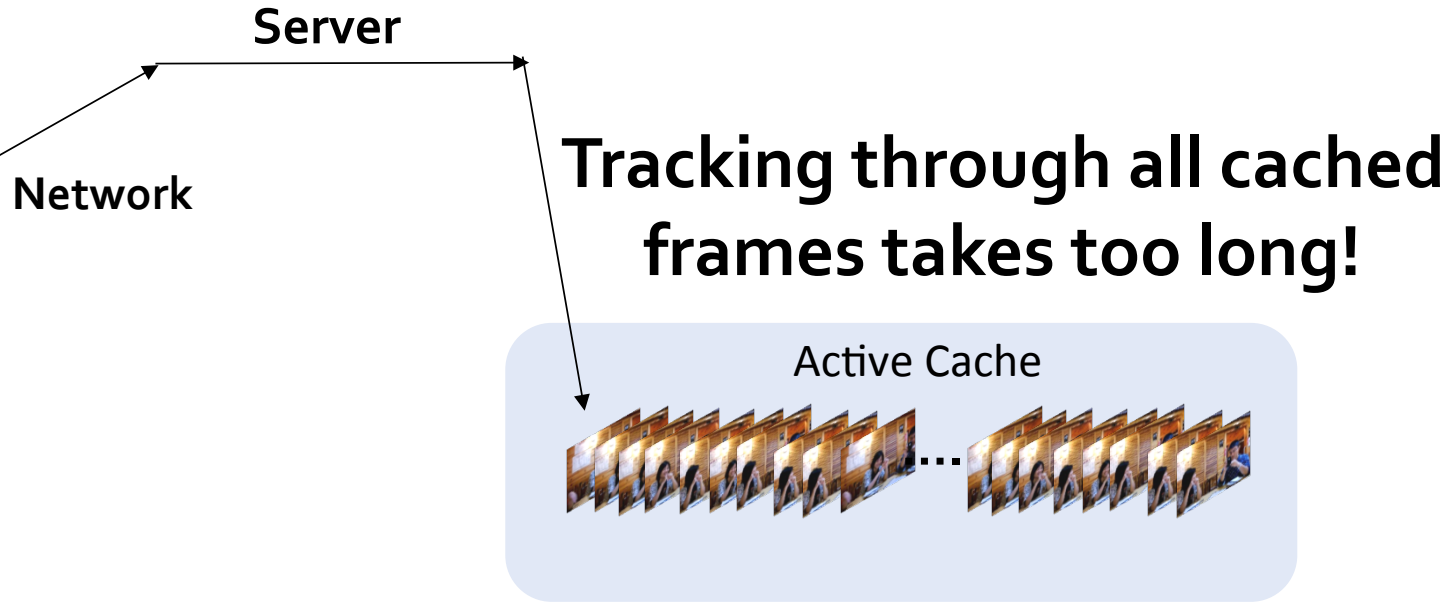
Regain Accuracy with *Active Cache*

- Cache and run tracking through the cached frames



Regain Accuracy with *Active Cache*

- Cache and run tracking through the cached frames



Adaptive Frame Selection

Given ***n_cached*** frames, select ***s_selected*** frames so that we can catch up without sacrificing tracking performance

Adaptive Frame Selection

Given ***n_cached*** frames, select ***s_selected*** frames so that we can catch up without sacrificing tracking performance

1. How many frames to select?
2. Which frames to select?

Adaptive Frame Selection

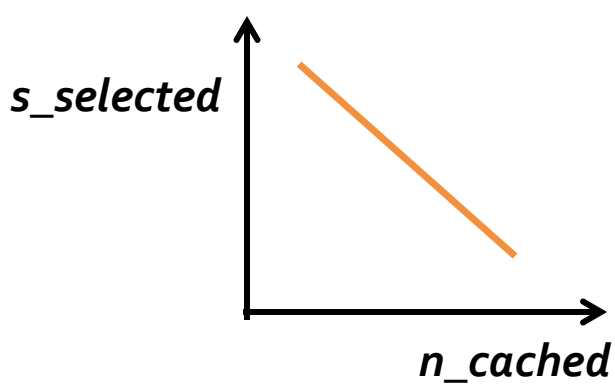
Given ***n_cached*** frames, select ***s_selected*** frames so that we can catch up without sacrificing tracking performance

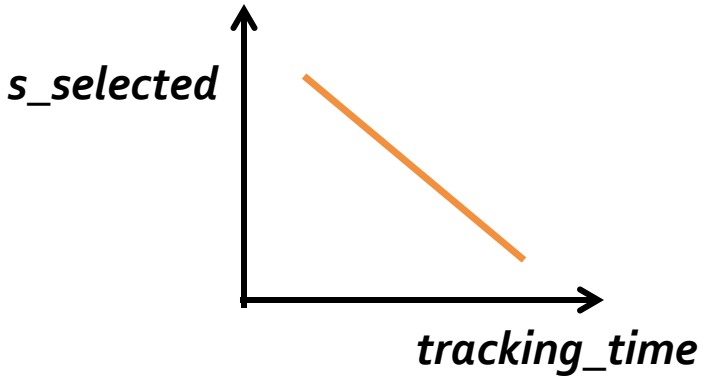
1. **How many frames to select?**
   - ***s_selected:*** active cache processing time vs. tracking accuracy

Adaptive Frame Selection

Given ***n_cached*** frames, select ***s_selected*** frames so that we can catch up without sacrificing tracking performance

1. **How many frames to select?**
   - ***s_selected:*** active cache processing time vs. tracking accuracy
   - ***s_selected*** depends on
     a. The end-to-end delay -- ***n_cached***
     b. The exec time of tracking on the client-- ***tracking_time***





Adaptive Frame Selection

Given ***n_cached*** frames, select ***s_selected*** frames so that we can catch up without sacrificing tracking performance

1. **How many frames to select?**
   - ***s_selected:*** active cache processing time vs. tracking accuracy
   - ***s_selected*** depends on
     a. The end-to-end delay -- ***n_cached***
     b. The exec time of tracking on the client-- ***tracking_time***
   - Simulate *n_cached*, *tracking_time*, and *s_selected,* and pick the *s_selected* that maximizes the accuracy

Adaptive Frame Selection

Given ***n_cached*** frames, select ***s_selected*** frames so that we can catch up without sacrificing tracking performance

2. Given *s_selected*, which frames to select?

- Temporal redundancy between frames

Adaptive Frame Selection

Given ***n_cached*** frames, select ***s_selected*** frames so that we can catch up without sacrificing tracking performance

2. Given *s_selected*, which frames to select?

- Temporal redundancy between frames
- Use *frame differencing* to quantify movement and select frames to capture as much movement as possible

Frame Differencing Function

$$a_{i,j}(x,y) = |f_i(x,y) - f_j(x,y)|$$

$$d_{i,j}(x,y) = \begin{cases} 1 & a_{i,j}(x,y) > \phi \\ 0 & otherwise \end{cases}$$

Frame Difference (Movement Metric):

$$d_{i,j} = \sum_{x,y} d_{i,j}(x,y), d_{i,j} \geq 0$$

Which frames to select?

Given a sequence of frame differences D = {d_i, d_{i+1}, …. , $d_{i+(n-1)}$}, divide D into (l+1) partitions

Linear partition problem – $O(ln^2)$

Define *H[n,l]* as the optimum value of a partition arrangement with n frame differences and l partitions.

Dynamic Programming Formulation:

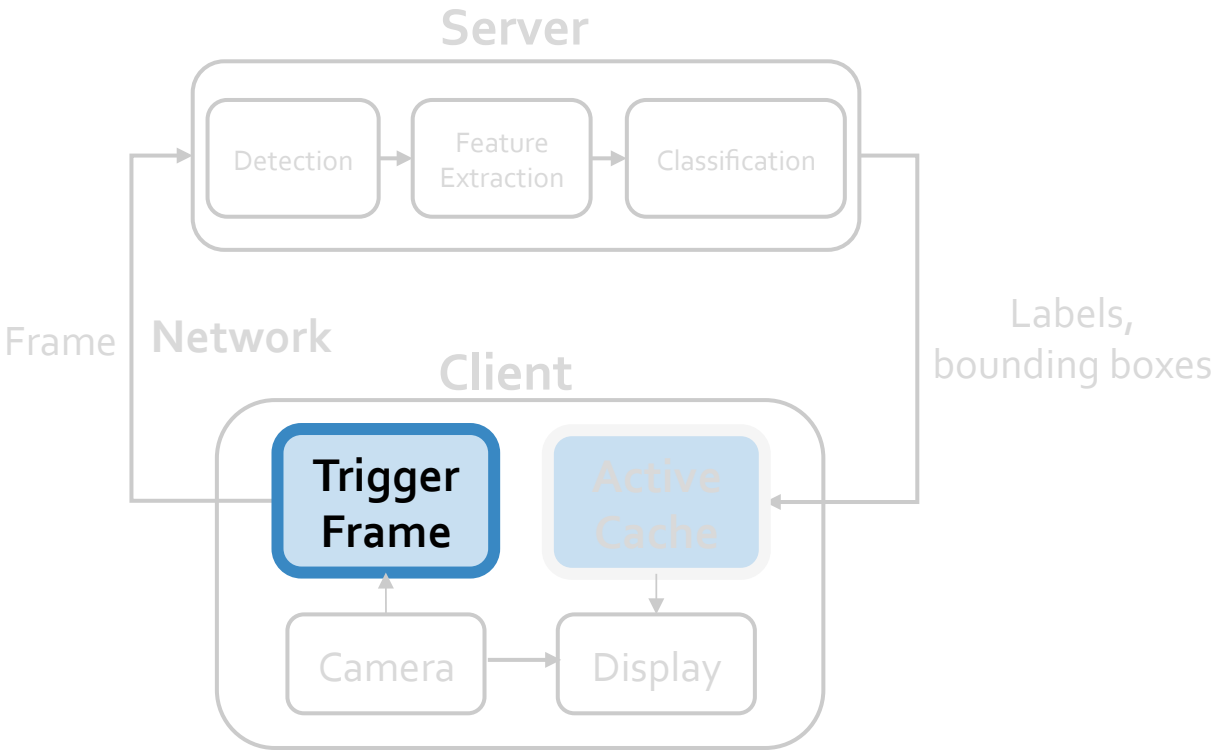
$$H[n,l] = \min_{j=i}^{i+n}(\max\{H[j,l-1], \sum_{k=j}^{i+n} d_{k,k+1}\})$$

Active Cache Achieves Higher Accuracy

Before Active Cache **After Active Cache**

- Active Cache can be applied to any objects
- Active Cache can be used to hide any end-to-end delay

Glimpse Architecture



1. **Active Cache** combats e2e latency and regains accuracy
2. **Trigger Frame** reduces bandwidth usage

Reduce Bandwidth Usage with Trigger Frames

- Strategically send certain trigger frames to the server

Reduce Bandwidth Usage with Trigger Frames

- Strategically send certain trigger frames to the server

1. Measuring scene changes

Reduce Bandwidth Usage with Trigger Frames

- Strategically send certain trigger frames to the server

1. Measuring scene changes
2. Detecting tracking failure

- The standard deviation of distance of all tracked points between two frames

Reduce Bandwidth Usage with Trigger Frames

- Strategically send certain trigger frames to the server

1. Measuring scene changes
2. Detecting tracking failure

- Limiting the number of frames in-flight

Evaluation

- **Object recognition pipelines**
  1. Face recognition
  2. Road sign recognition

Evaluation

- **Object recognition pipelines**
  1. Face recognition
  2. Road sign recognition
- **Datasets**

  1. Face Dataset:

  - 26 videos recorded with a smartphone
  - 30 minutes, 54K frames, and 36K faces
  - Scenarios: shopping with friends and waiting at a subway station

  2. Road Sign Dataset:

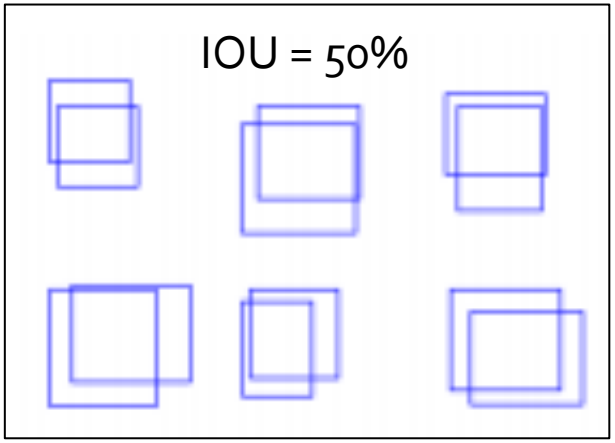
  - 4 walking videos recorded using Google Glass from YouTube
  - 35 minutes, 63K frames, and 5K road signs

Evaluation

- **Evaluation Metrics**
  - Intersection over union (IOU) to measure localization accuracy

$$IOU_i = \frac{area\,|O_i \cap G_i|}{area\,|O_i \cup G_i|}$$

O*i:* bounding box of the detected object *i*
G*i:* bounding box of object *i*'s ground truth



  - **Correct if <u>IOU > 50%</u> and the <u>label matches</u> ground truth**

Evaluation

- **Evaluation Metrics**
  - Precision

$$\frac{\text{\# of objects correctly labeled and located}}{\text{total \# of objects detected}}$$

  - Recall

$$\frac{\text{\# of objects correctly labeled and located}}{\text{total \# of objects in the ground truth}}$$

Evaluation

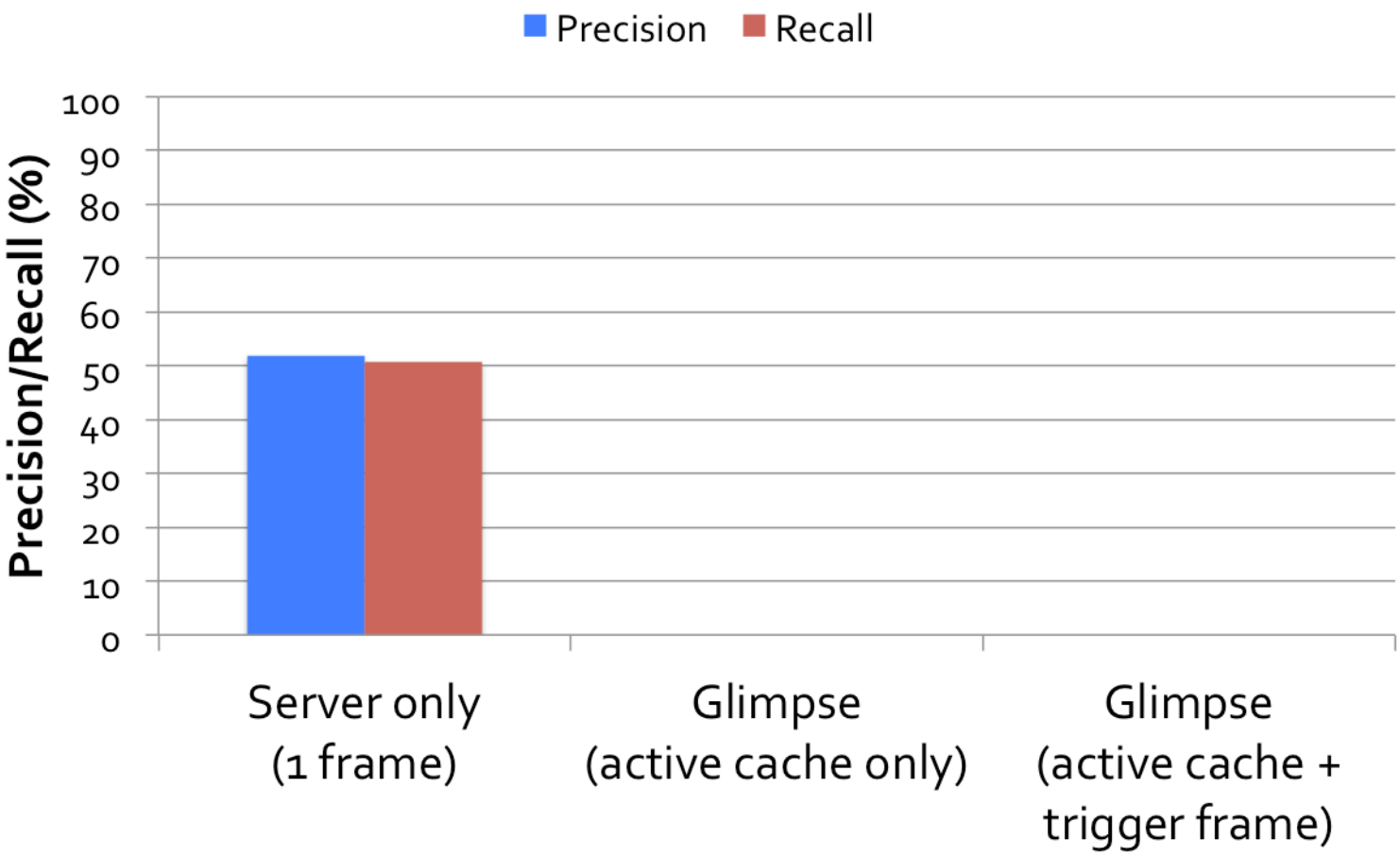
- **Network conditions**
  - Wi-Fi, Verizon's LTE, and AT&T's LTE network

Results Outline

1. Face recognition
2. Road sign recognition
3. Face recognition with hardware-assisted face detection

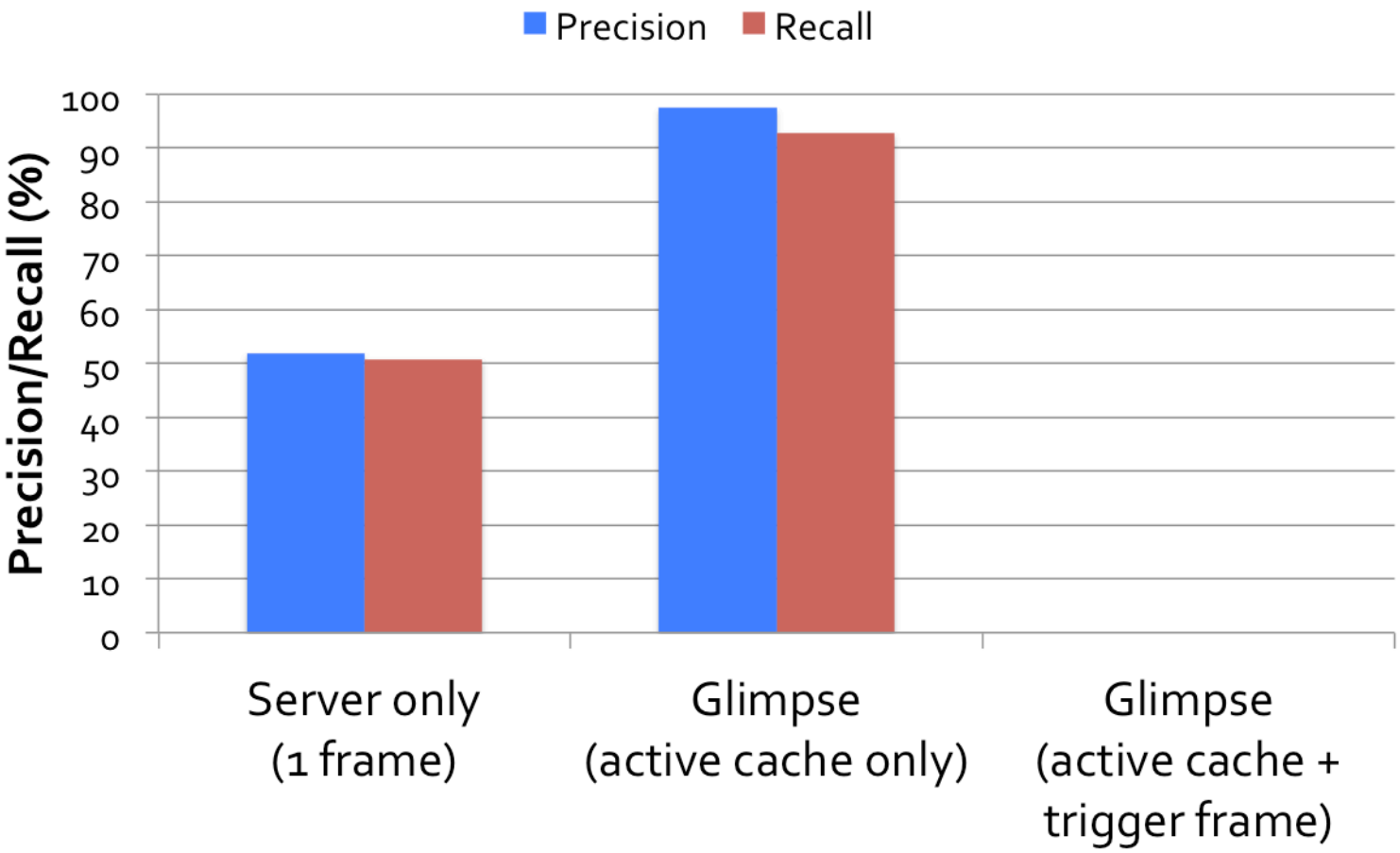
Active Cache Achieves High Accuracy

- Face dataset
- Wi-Fi (End-to-end delay: 430 ms)



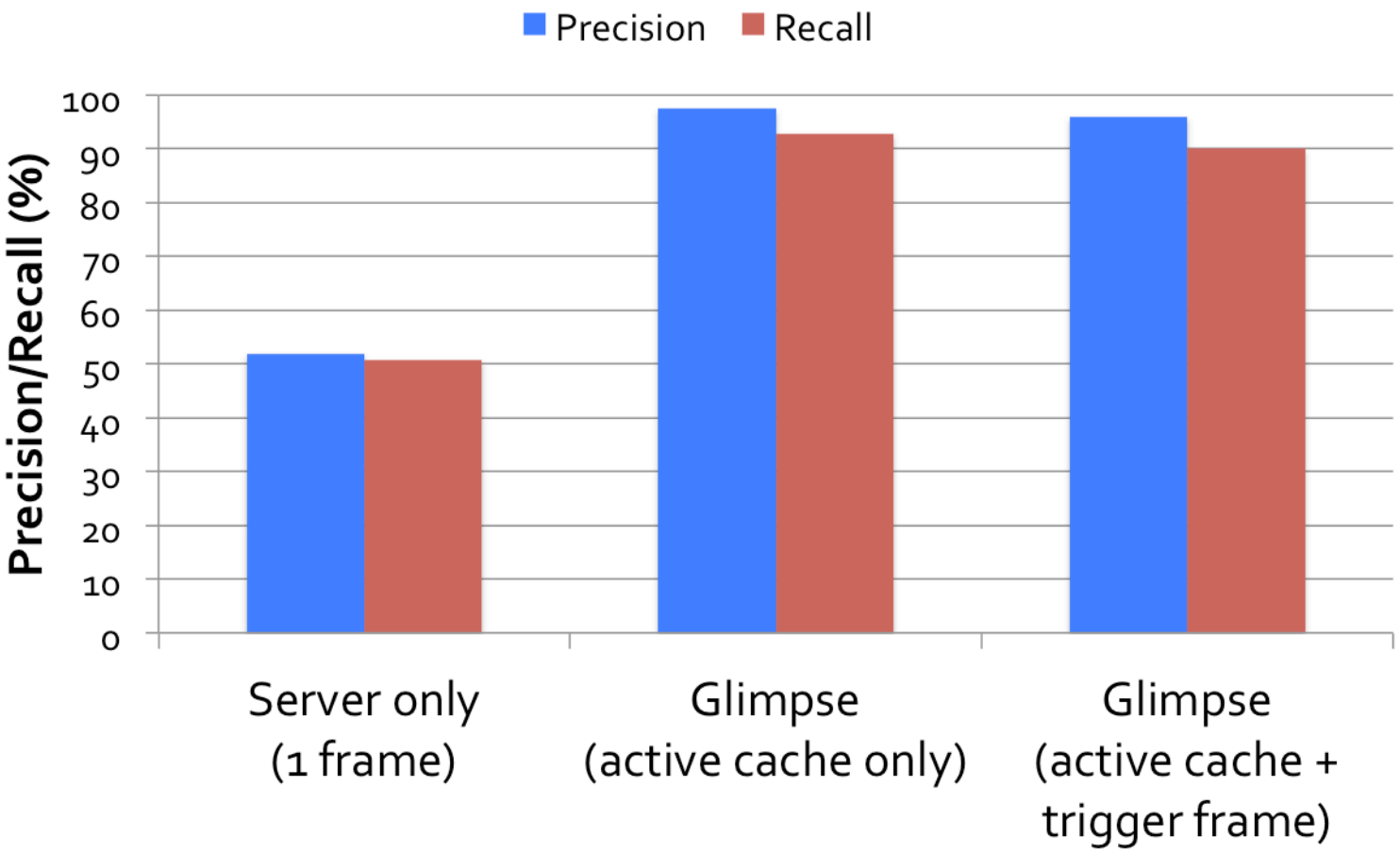
Active Cache Achieves High Accuracy

- Face dataset
- Wi-Fi (End-to-end delay: 430 ms)



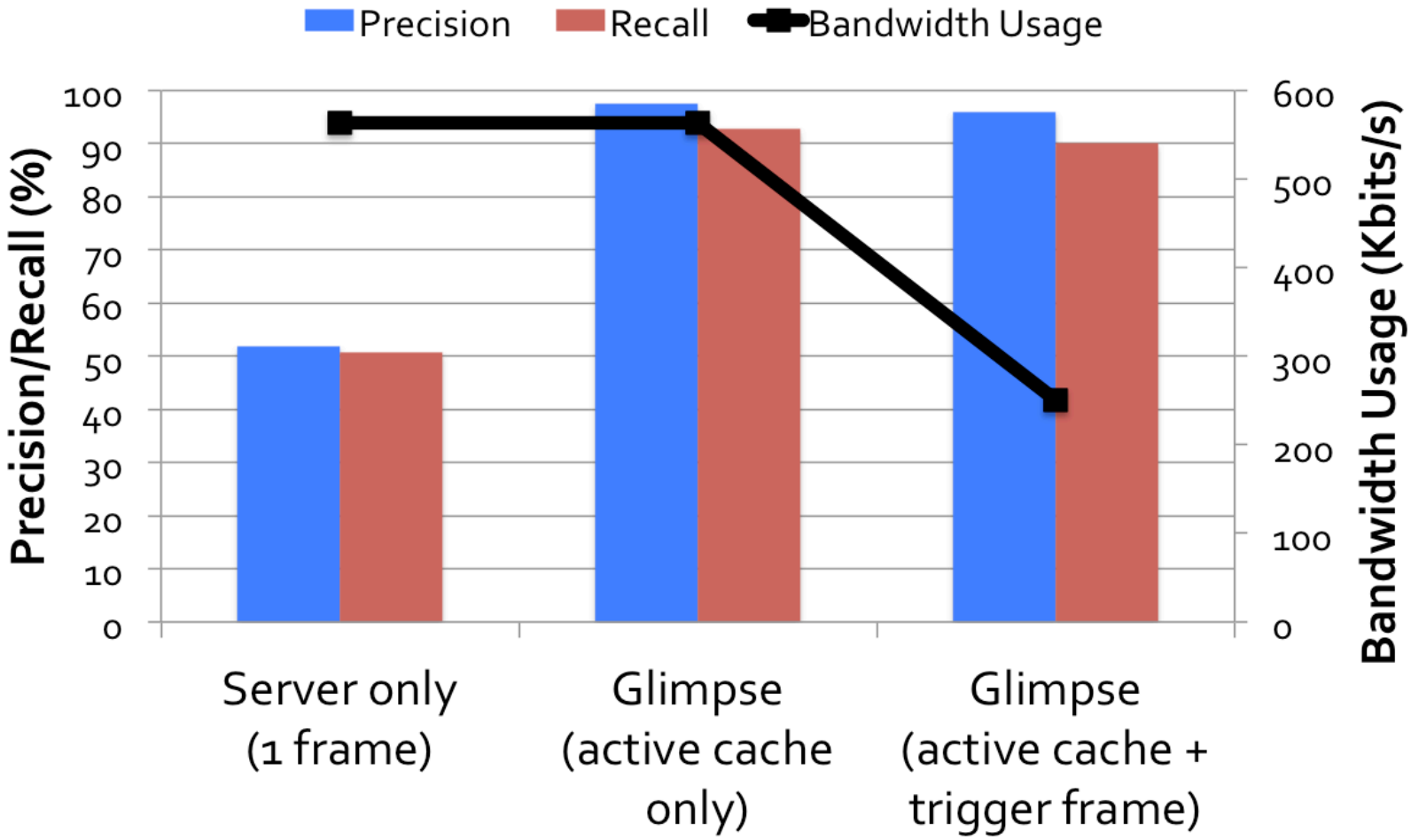
Trigger Frame Reduces Bandwidth Usage without Sacrificing Accuracy

- Face dataset
- Wi-Fi (End-to-end delay: 430 ms)



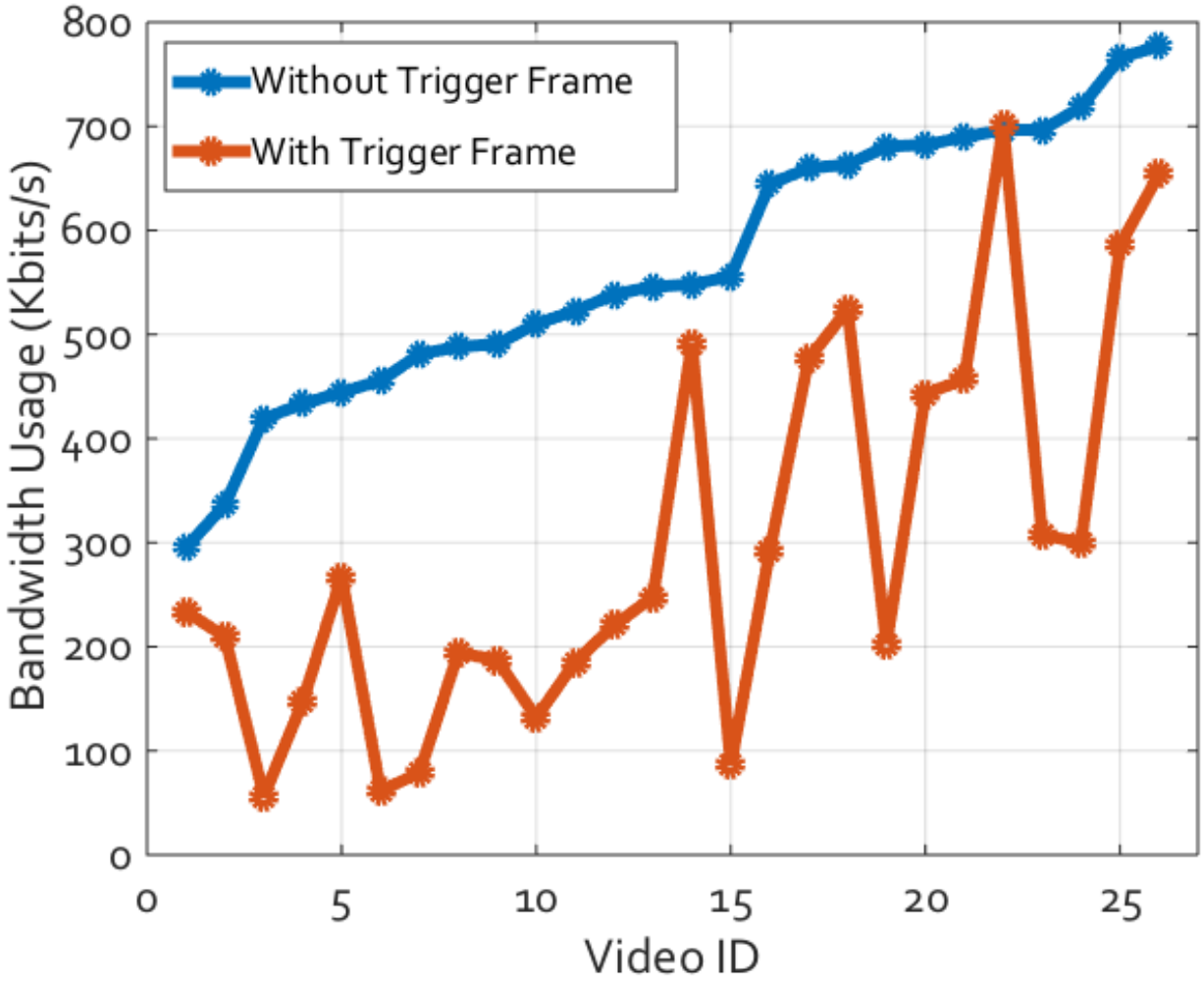
Trigger Frame Reduces Bandwidth Usage without Sacrificing Accuracy

- Face dataset
- Wi-Fi (End-to-end delay: 430 ms)



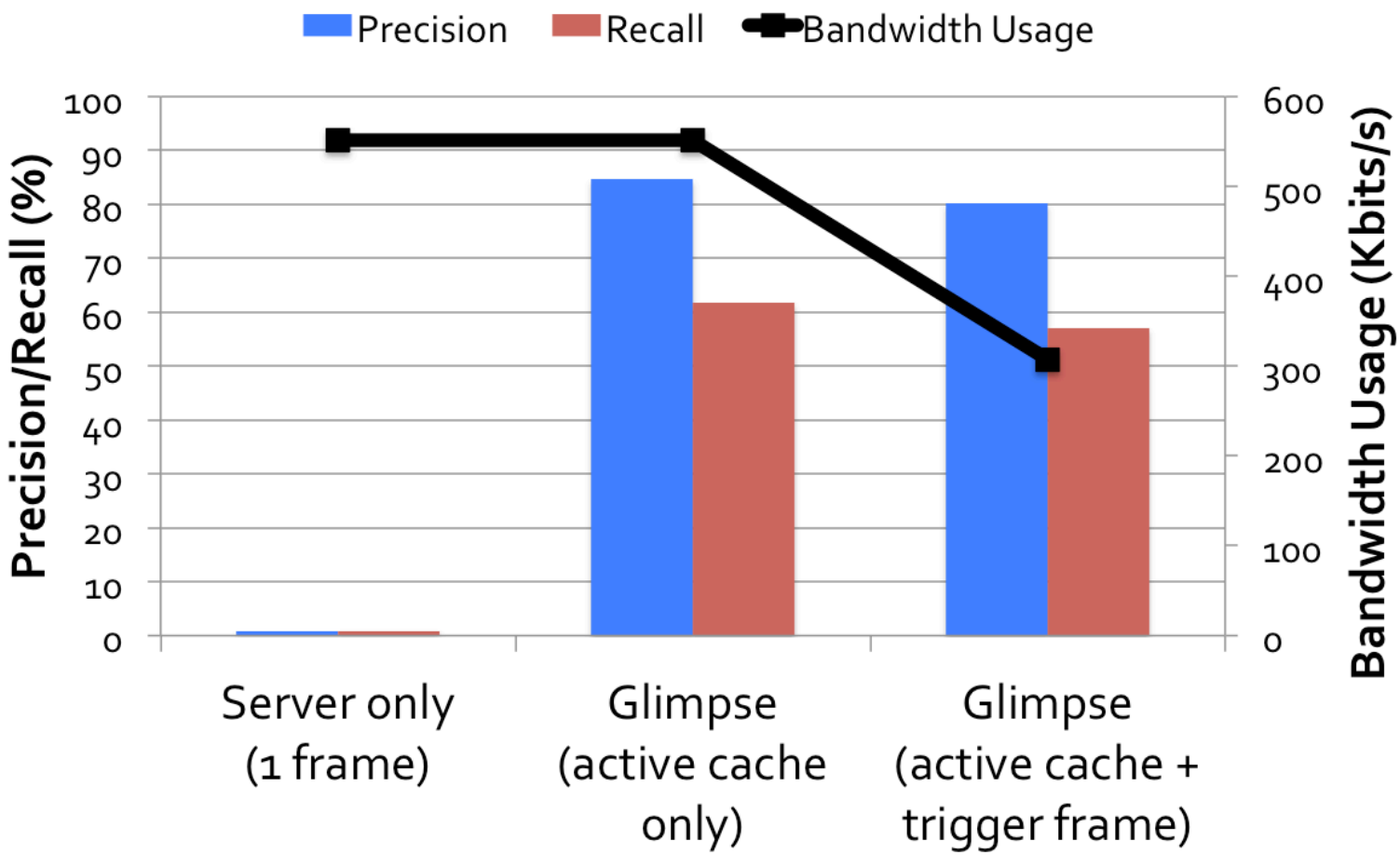
Trigger Frame Consistently Reduces Bandwidth Usage

- Face Dataset (Wi-Fi)



Glimpse Achieves Higher Accuracy and Lower Bandwidth Usage

- Road sign dataset
- Wi-Fi (End-to-end delay: 520 ms)

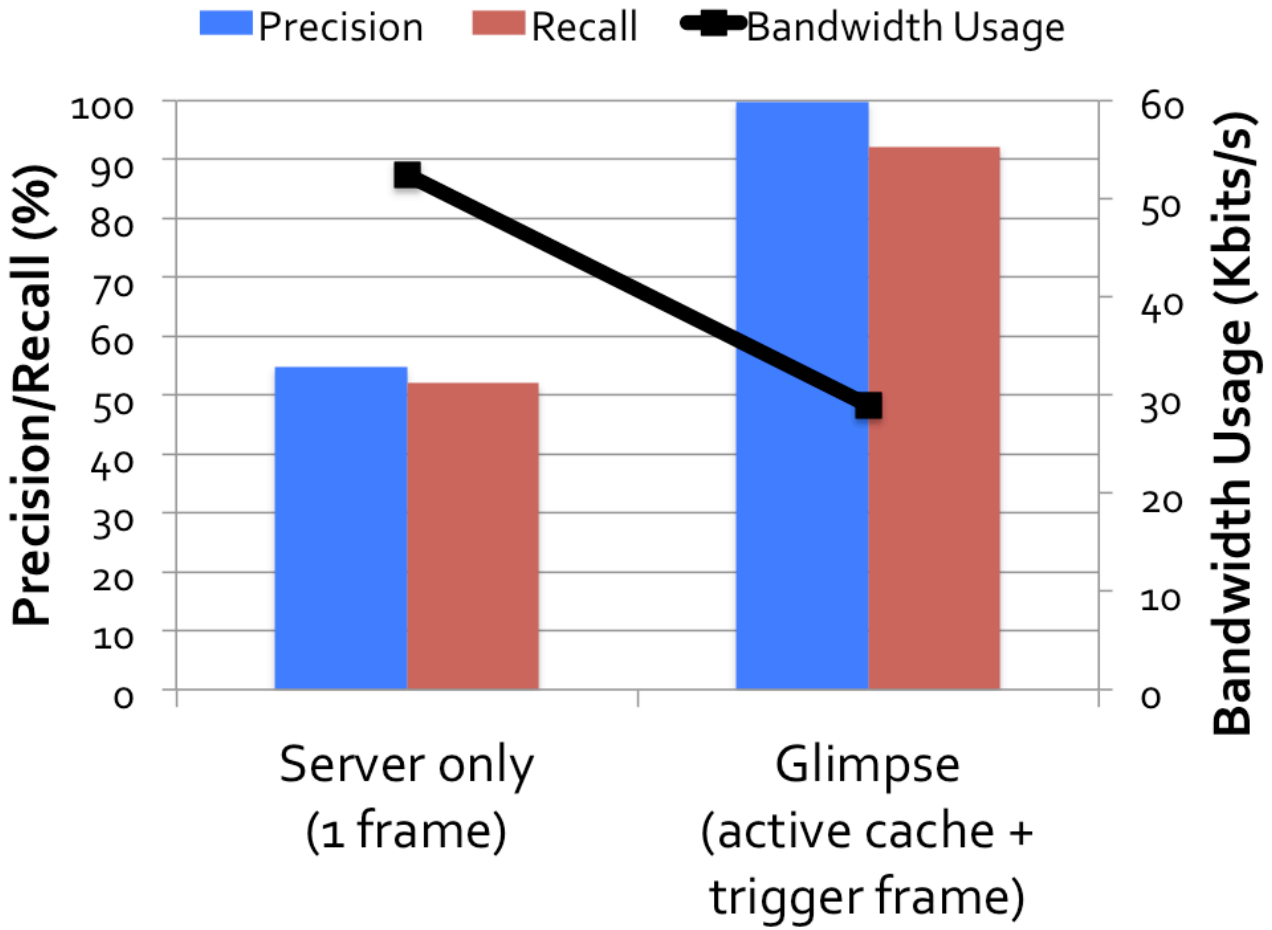


Hardware-Assisted Object Detection

- Mobile devices are now equipped with object detection hardware
- Is Glimpse still helpful?

Glimpse Improves Accuracy even with Detection Hardware on Devices

- Face dataset (Wi-Fi)
- Face detection in hardware



Glimpse

- Glimpse enables continuous, real time object recognition on mobile devices
- Glimpse achieves high recognition accuracy by maintaining an *active cache* of frames on the client
- Glimpse reduces bandwidth consumption by strategically sending only certain *trigger frames*

Active Cache and Trigger Frame are Generic

- Latency caused performance degradation and excessive resource usage are fundamental problems to object recognition
- *Active Cache* can hide any end-to-end latency
- *Trigger Frame* can reduces resource consumed

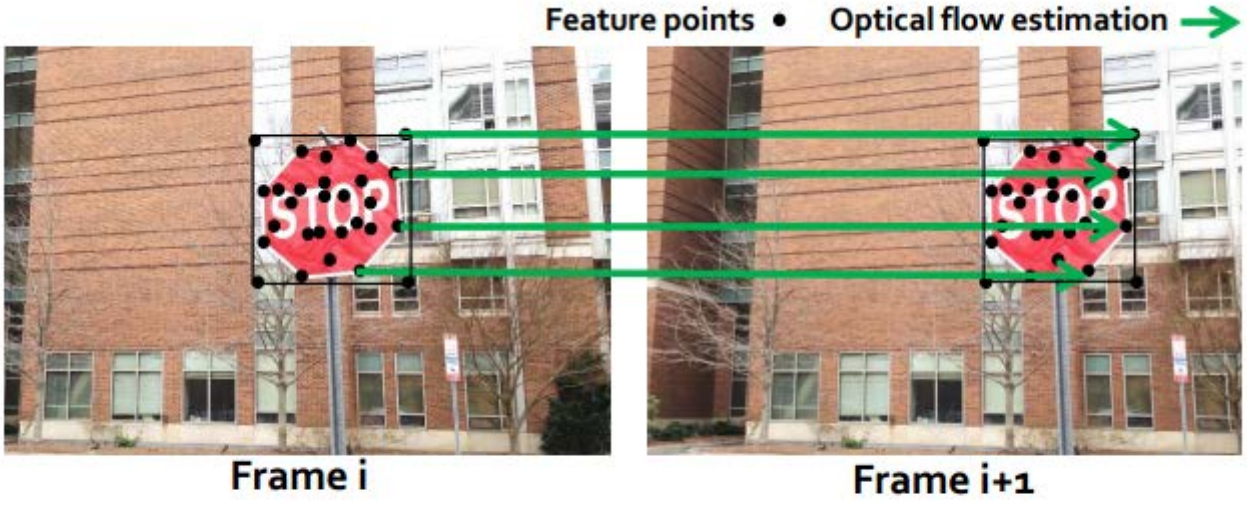
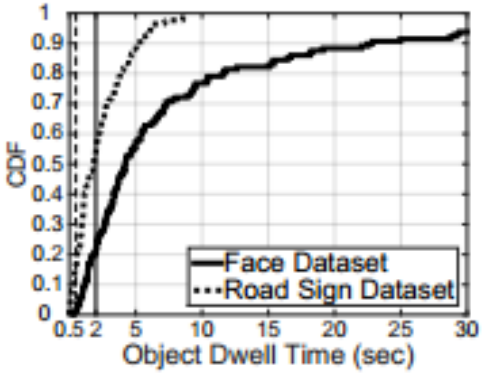


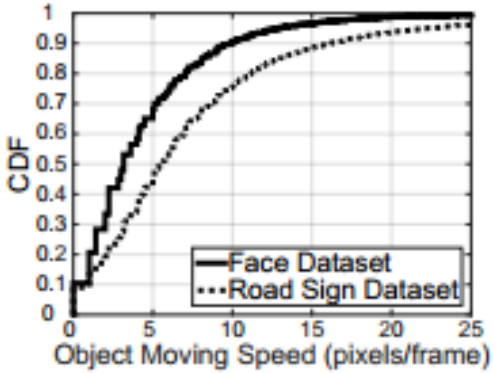
Figure 3: Extracting feature points on a road sign in frame i, computing the optical flow for each of the points, and locating the points in frame $i + 1$.

Scheme	Device-only	Offload to Server (Wi-Fi)
Road sign recognition	11.32 J	0.54 J
Face recognition	5.16 J	0.44 J

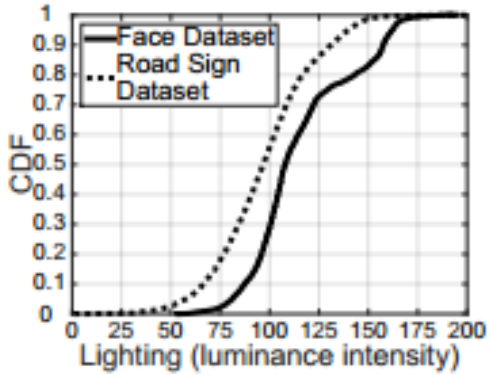
Table 1: Energy consumption of the object recognition pipeline for a single frame on a Samsung Galaxy Nexus.



(a) The CDF of the dwell time of each object.



(b) The CDF of object moving speed.



(c) The CDF of luminance intensity of frames.

Figure 8: Dataset characteristics.

Networks	Schemes	Precision (%)	Recall (%)	Bandwidth Usage (Kbits/s)
Wi-Fi	Server only	56.5	52.2	616.1
	Server only (1 frame in flight)	54.7	52	52.5
	Glimpse	**99.8**	**92.1**	**28.9**
Verizon's LTE	Server only	50.8	47.4	605.2
	Server only (1 frame in flight)	47	44.3	40.2
	Glimpse	**99.4**	**90.7**	**25**
AT&T's LTE	Server only	38.4	31.6	335.8
	Server only (1 frame in flight)	41.3	39	32.1
	Glimpse	**96.4**	**85.5**	**20.7**

Table 3: Glimpse with face detection in hardware.

Datasets	Networks	Glimpse (active cache only) F1 score (%)	No active cache F1 score (%)
Road Sign	Wi-Fi	71.4	48.5
	Verizon's LTE	50.9	26.1
Face	Wi-Fi	95.1	87.2
	Verizon's LTE	91.9	82.1
	AT&T's LTE	88	78.1

Table 4: The F1 score of Glimpse (active cache only) vs. without maintaining an active cache.

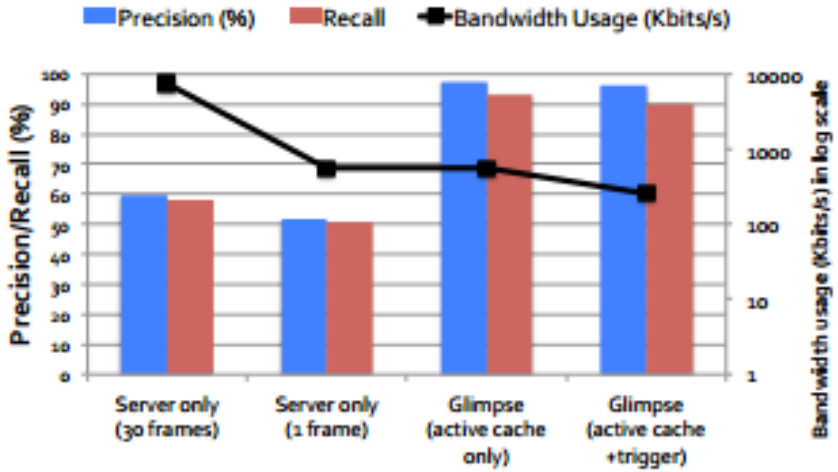


Figure 9: Performance of Glimpse on Wi-Fi for tracking faces. The end-to-end delay (the latency from mobile to server and obtain a response) is 425-455 ms in all schemes, except for 30 frames in flight (542.2 ms).

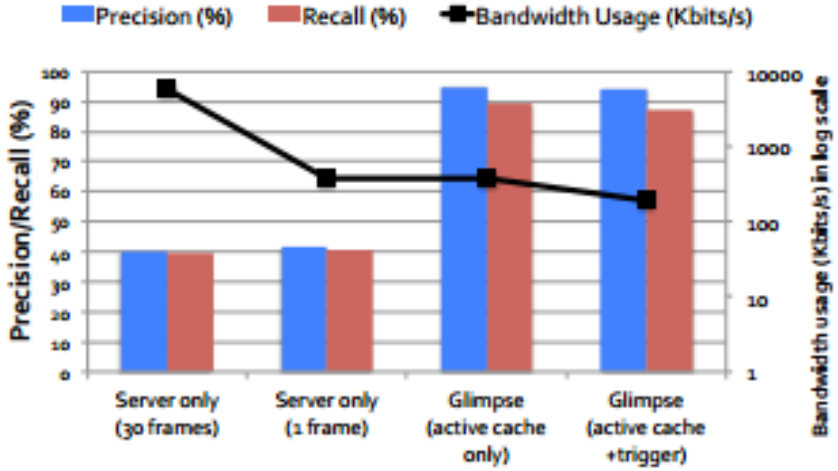


Figure 10: Performance of Glimpse on Verizon's LTE for tracking faces. The end-to-end delay is 656-721 ms in all schemes, except for 30 frames in flight (1102.5 ms).

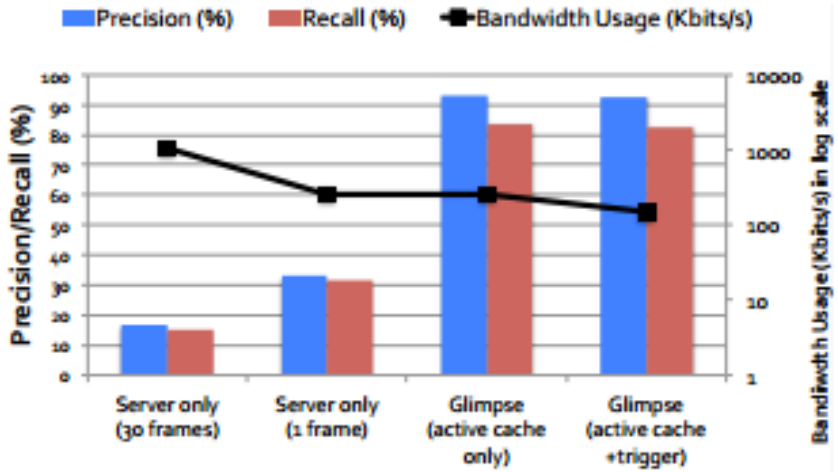


Figure 11: Performance of Glimpse on AT&T's LTE for tracking faces. The end-to-end delay is 927-1041.2 ms in all schemes, except for 30 frames in flight (7391.7 ms).

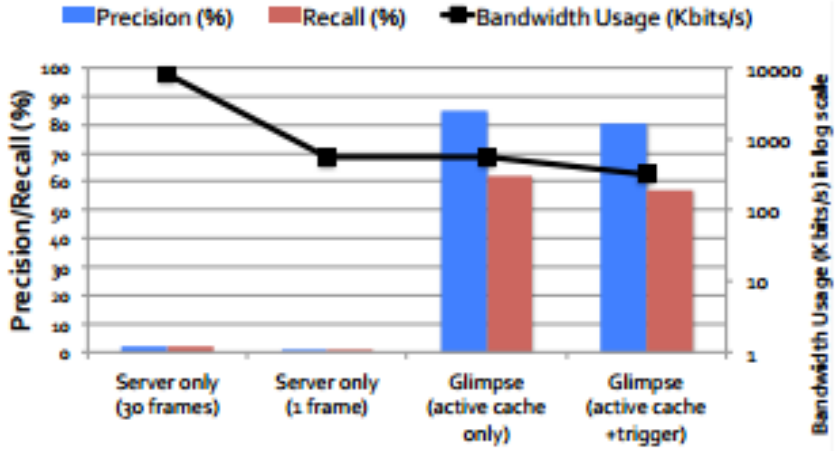


Figure 12: Performance of Glimpse on Wi-Fi for tracking road signs. The end-to-end delay is 510-548 ms in all schemes, except for 30 frames in flight (683.1 ms).

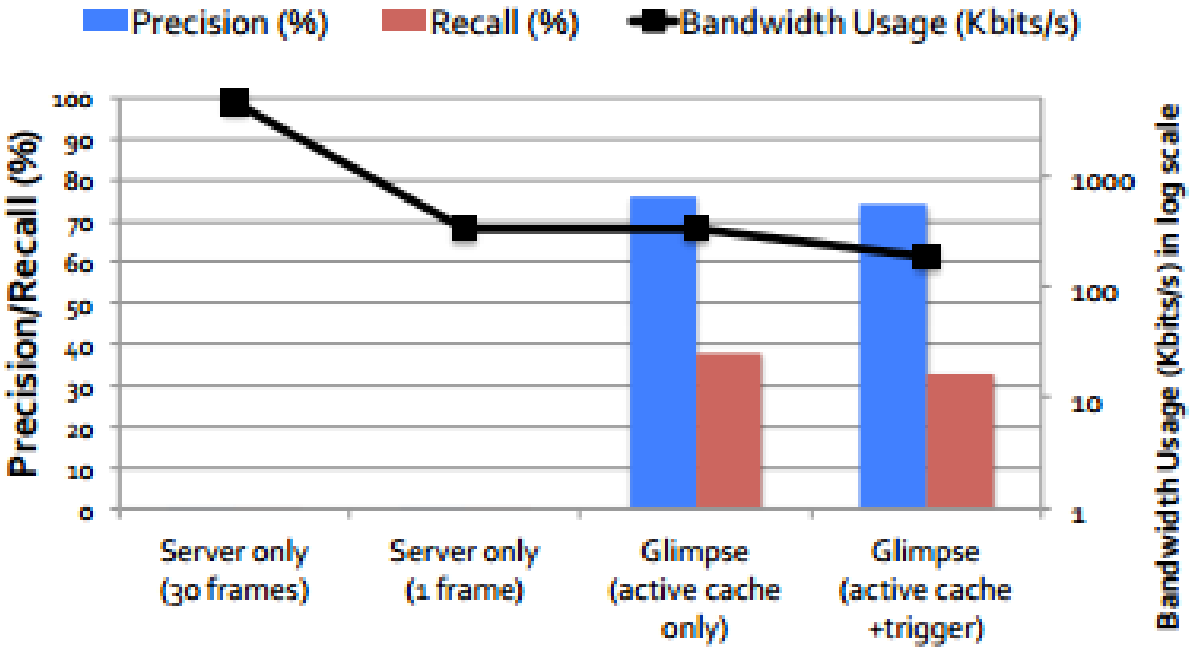


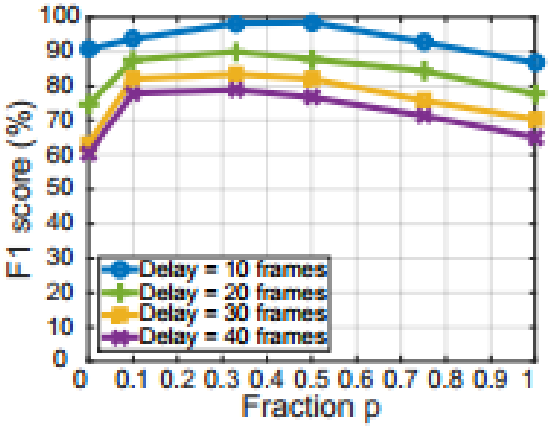
Figure 13: Performance of Glimpse on Verizon's LTE for tracking road signs. The end-to-end delay is 901.1-963.2 ms in all schemes, except for 30 frames in flight (1765.2 ms).

Networks	Schemes	Precision (%)	Recall (%)	Bandwidth Usage (Kbits/s)
Wi-Fi	Server only	56.5	52.2	616.1
	Server only (1 frame in flight)	54.7	52	52.5
	Glimpse	**99.8**	**92.1**	**28.9**
Verizon's LTE	Server only	50.8	47.4	605.2
	Server only (1 frame in flight)	47	44.3	40.2
	Glimpse	**99.4**	**90.7**	**25**
AT&T's LTE	Server only	38.4	31.6	335.8
	Server only (1 frame in flight)	41.3	39	32.1
	Glimpse	**96.4**	**85.5**	**20.7**

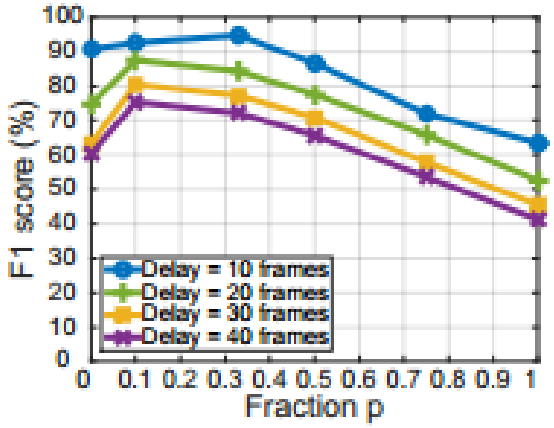
Table 3: Glimpse with face detection in hardware.

Datasets	Networks	Glimpse (active cache only) F1 score (%)	No active cache F1 score (%)
Road Sign	Wi-Fi	71.4	48.5
	Verizon's LTE	50.9	26.1
Face	Wi-Fi	95.1	87.2
	Verizon's LTE	91.9	82.1
	AT&T's LTE	88	78.1

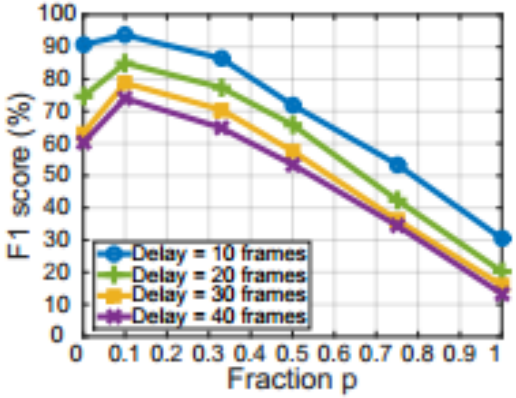
Table 4: The F1 score of Glimpse (active cache only) vs. without maintaining an active cache.



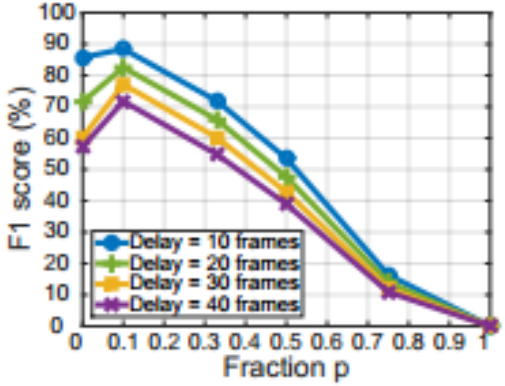
(a) $e = 10$ ms.



(b) $e = 20$ ms.



(c) $e = 30$ ms.



(d) $e = 40$ ms.

Figure 14: Performance of tracking based on the fraction of frames picked from the active cache using both datasets.

Network	DP-based F1 score (%)	Fixed-interval F1 score (%)
Wi-Fi	82.2 ± 7.5	73.3 ± 12.4
Verizon's 3G	77 ± 9.4	59 ± 11.7

Table 5: F1 score for hard videos in the dataset. For this experiment, we selected 50% of all videos that had the worst F1 score. For Wi-Fi, we use both the face and road sign datasets. For Verizon's 3G network, we use only the face dataset.

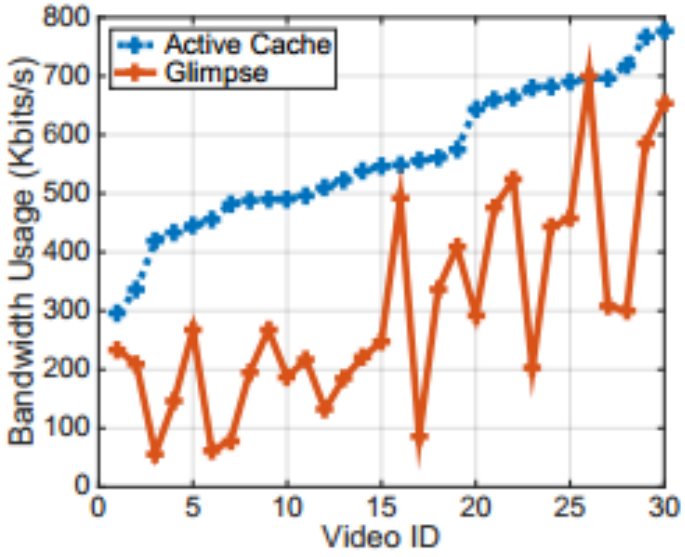
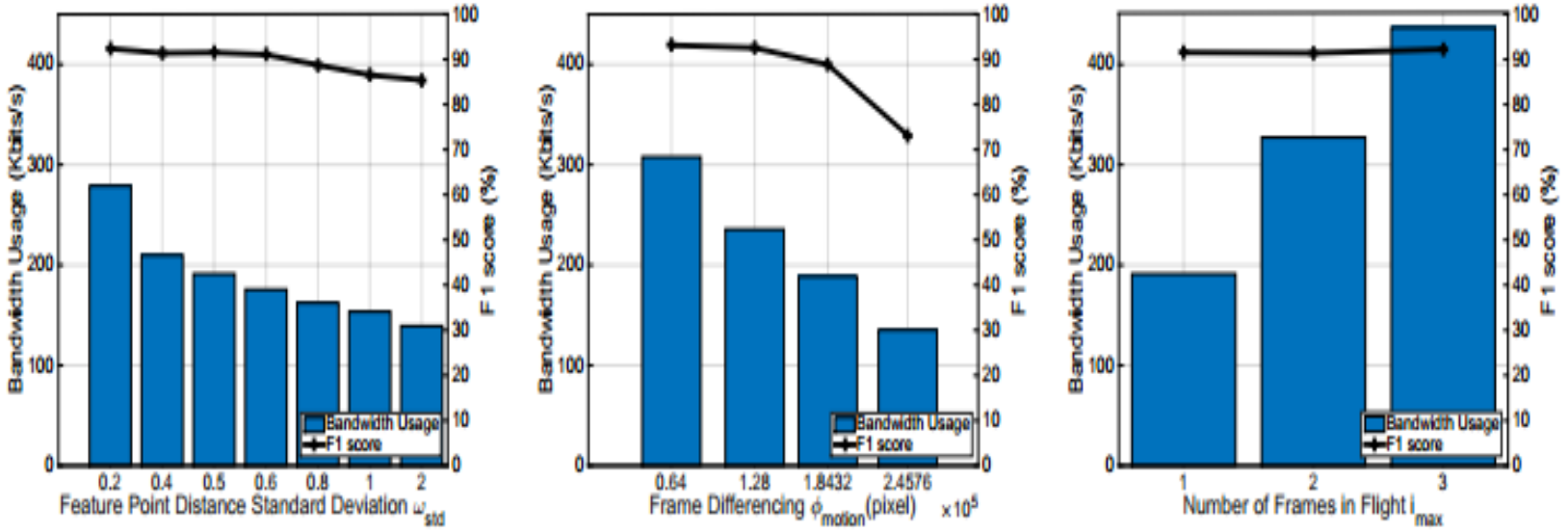


Figure 15: Bandwidth usage of each video by Glimpse and the *Active cache* scheme without trigger frames. The *Active cache* line is sorted by bandwidth usage. The Glimpse line shows the bandwidth usage of the corresponding video.



(a) ω_{std}, the standard deviation of feature point distance.

(b) ϕ_{motion}, the frame difference threshold.

(c) i_{max}, the number of frames in flight.

Figure 16: Bandwidth usage and F1 score for various parameters on identifying trigger frames for the face dataset. We see the same trends for the road sign dataset.

Mode	Power (mW)	Current (mA)	Battery Life (h)
Sleep	56	15.1	122.5
Idle	1009.2	272.9	6.8
Screen on + Frame Capturing	2124.6	574.2	3.2
Verizon's LTE uplink (Active)	2928.2	791.4	2.3
Verizon's LTE downlink (Active)	1737	496.3	3.7
Verizon's LTE (Idle)	1324.6	358	5.2
Wi-Fi uplink	1871.5	506	3.7
Wi-Fi downlink	1289.9	347.8	5.3

Table 6: Energy measurement for various operations on the Samsung Galaxy Nexus with an 1850 mAh battery.

Component	Execution Time (ms)
Frame differencing	7
Active cache (DP frame selection)	1.7
Tracking failure detection	< 1
Tracking	37.7

Table 7: Average execution time of each of Glimpse's components on the Samsung Galaxy Nexus.

Network	Glimpse battery life time (h)	1 frame/RTT battery life time (h)
Wi-Fi	1.9 ± 0.1	1.4
Verizon's LTE	1.5 ± 0.1	1.2

Table 8: Estimated battery life time.

by 35% on Wi-Fi and 25% on LTE. For Google Glass (Wi-Fi), based on the energy measurement numbers reported in [33], compared to a scheme that sends 1 frame every RTT, Glimpse can improve the battery life by 24%.