MultiSense: Fine-grained Multiplexing for Steerable Camera Sensor Networks

Navin Sharma, David Irwin, Prashant Shenoy, and Michael Zink
Camera Sensor Network

- Network of Camera Sensors
  - Useful for remote monitoring
  - E.g., doorways, intersections, equipment

- Key Attribute: Programmable Actuation
  - Pan-tilt-zoom ("steer") camera’s lens
  - Continuous loop: steer → sense → steer → sense
    - Sense == capture image, Steer == pan-tilt-zoom
Multi-user Camera Sensor Network

- Sharing is Advantageous
  - High deployment and maintenance cost
  - Per-user networks economically infeasible

- Sharing in today’s camera networks
  - Actuation controlled by single user
  - Data sharing is the only option
    - E.g., live traffic cameras

- Need for multi-user “timeshared” camera networks
Outline

- Motivation
- Overview
  - Challenges of Shared Actuation
  - Virtualization-based Approach
- MultiSense Design
  - Camera-specific Optimizations
  - Proportional-share Scheduling
- Implementation and Evaluation
  - Xen-based using Sony PTZ
  - Synthetic Workloads + Case Study
- Conclusion
Sharing Actuation: Desirable Properties

Challenge

How do we multiplex a camera network to efficiently service conflicting demands of competing users?

- **Transparency**: users feel like they control dedicated cameras
- **Isolation**: users don’t impact performance of other users
- **Arbitration**: control over each user’s QoS
Virtualization-based Approach

- **Our Approach: Virtualization**
  - Virtual camera looks like physical camera
  - Extend VMM isolation mechanisms
  - **Goal:** hide + mitigate complexity of state transitions

![Diagram]

- **MultiSense**
  - Alice
  - Bob
  - Carol
  - Alice’s vcamera
  - Bob’s vcamera
  - Carol’s vcamera
  - Physical Camera
Design Overview

- **Performance**
  - Minimize slow mechanical actuation overhead

- **Fairness**
  - Proportionally partition sensing resources among users
Naïve Time Sharing

- Divide time into slots
- Round-robin assign slots to users

Naïve interleaving leads to wrong results!
- Cameras are “stateful”: PTZ position encodes state
Sharing with State Restoration

Interleaving actuations can be wasteful

**Insight:** actuation without capturing an image does not change an application’s control flow
Request Groups

- Prevent waste by grouping requests
  - All actuation requests grouped with a sense
  - Execute group atomically

Alice Request Queue
- Pan: 30°
- grab

Bob Request Queue
- Pan: -30°
Actuator Fair Queuing

- Adapt Start-time Fair Queuing
  - Well-known algorithm from networks/CPUs
    - Partition actuation time based on weights
  - Direct adaptation is wasteful
    - Schedules based on fairness not performance

- Use request batching to adjust tradeoff between fairness and performance
Request Batching Example

Batch of size 3

12

30° -> 75° -> 40° = 80°

11

30° -> 40° -> 75° = 45°
MultiSense Implementation

- Leverage Xen Hypervisor
  - Augment VMs with virtual cameras
  - Write split-drivers for camera

```
Domain-0
MultiSense
  Camera Driver
  Xen Hypervisor
  Physical camera
```

```
VM-1
App 1
  vcamera 1

VM-2
App 2
  vcamera 2
```
**Evaluation: Setup**

- **Network of 2 Sony RZ50N PTZ Cameras**
  - Attached to nodes running Xen
  - Each node runs 5 VMs with virtual cameras

- **Workloads**
  - **Synthetic:** Random actuations and senses
  - **Case Study:** Tracking and monitoring requests
Benefits of Request Grouping

- **AFQ Scheduling with Batch Size of 5**

![Graph showing the benefits of request grouping with AFQ scheduling with a batch size of 5. The graph compares the number of requests serviced over time between request grouping and no request grouping. The graph indicates a 2x improvement with request grouping.](image-url)
Actuator-Fair Queuing

- 5 Users with different weights

Inefficient but Fair

Efficient but Unfair
Case Study: Tracking and Monitoring

- **Tracking**: scan path and capture image every 10°
- **Monitoring**: Continuously capture image at fixed-point

![Graphs showing Tracking Latency and Sensing Rate vs Weight Ratio](image-url)
Case Study: Tracking and Monitoring

- **Tracking**
  - For object at 300ft away...
  - ...capture image of object every 23 feet
  - ...equates to object moving at 2.66mph or a walking person

- **Monitoring**
  - Concurrently capture image of fixed point every 3 sec.

Refer to the paper for results on camera coordination
Related Work

- Virtualization
  - Extend Xen hypervisor
  - Virtualizing I/O devices
- Proportional-share Scheduling
  - Adapt SFQ for cameras
  - Combine with camera-specific optimizations
- Sensor Networks
  - Focus on high-power steerable sensors
  - Orthogonal to low-power sensor networks
Conclusion

- Cameras capable of supporting multiple concurrent applications
  - Challenges
    - Slow mechanical actuation, stateful actuators
    - Key techniques for steerable cameras
      - Request grouping, AFQ, and batching

- Future Work
  - Extend to other steerable sensors
  - E.g., weather radars