Scheduling Constrained Dynamic Applications on Clusters

Arun Chauhan

joint work with

Kathleen Knobe

Jim Rehg

Nikhil Rishiyur

Umakishore Ramachandran

October 11, 1999

Student Pizza Talk

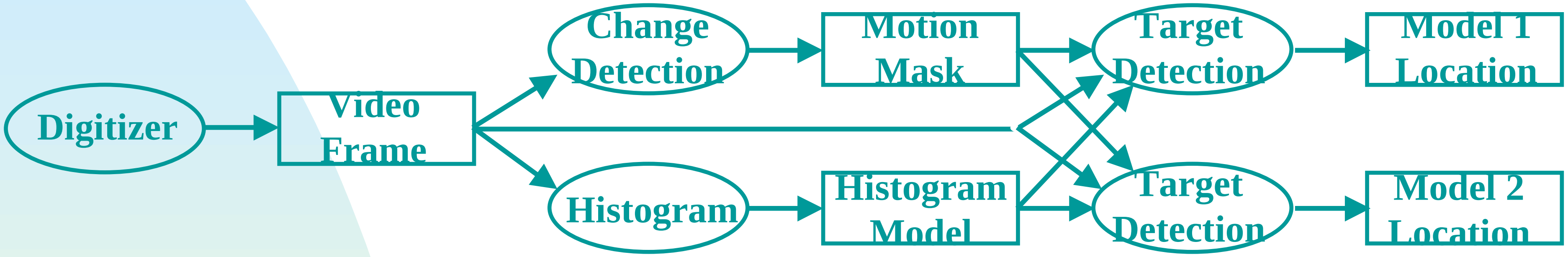
Context: user's view

- Free Standing Smart Kiosk
- Automatically detects approaching customers
- Animated face exhibits natural gaze behavior
- Interacts through synthesized voice and touch-screen

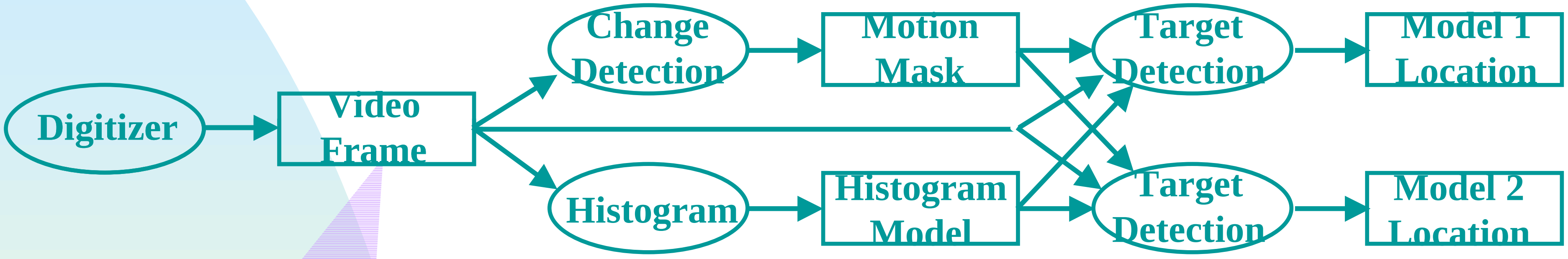
Context: programmer's view

- Multi-media applications
  - $\Rightarrow$ streaming data
- Interactive
  - $\Rightarrow$ response time (latency)
- Needs to be compelling
  - $\Rightarrow$ natural gaze behavior (people tracking)
- Kiosk has other background apps
  - $\Rightarrow$ dynamic environment

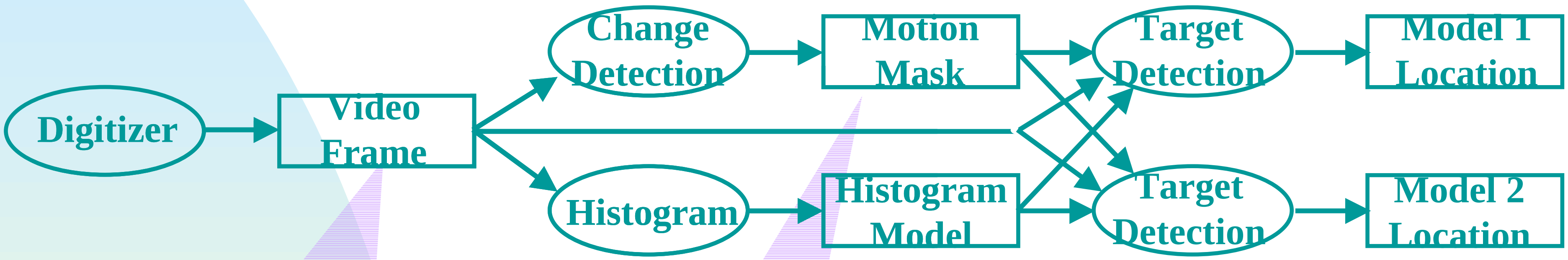
Operation of Color Tracker



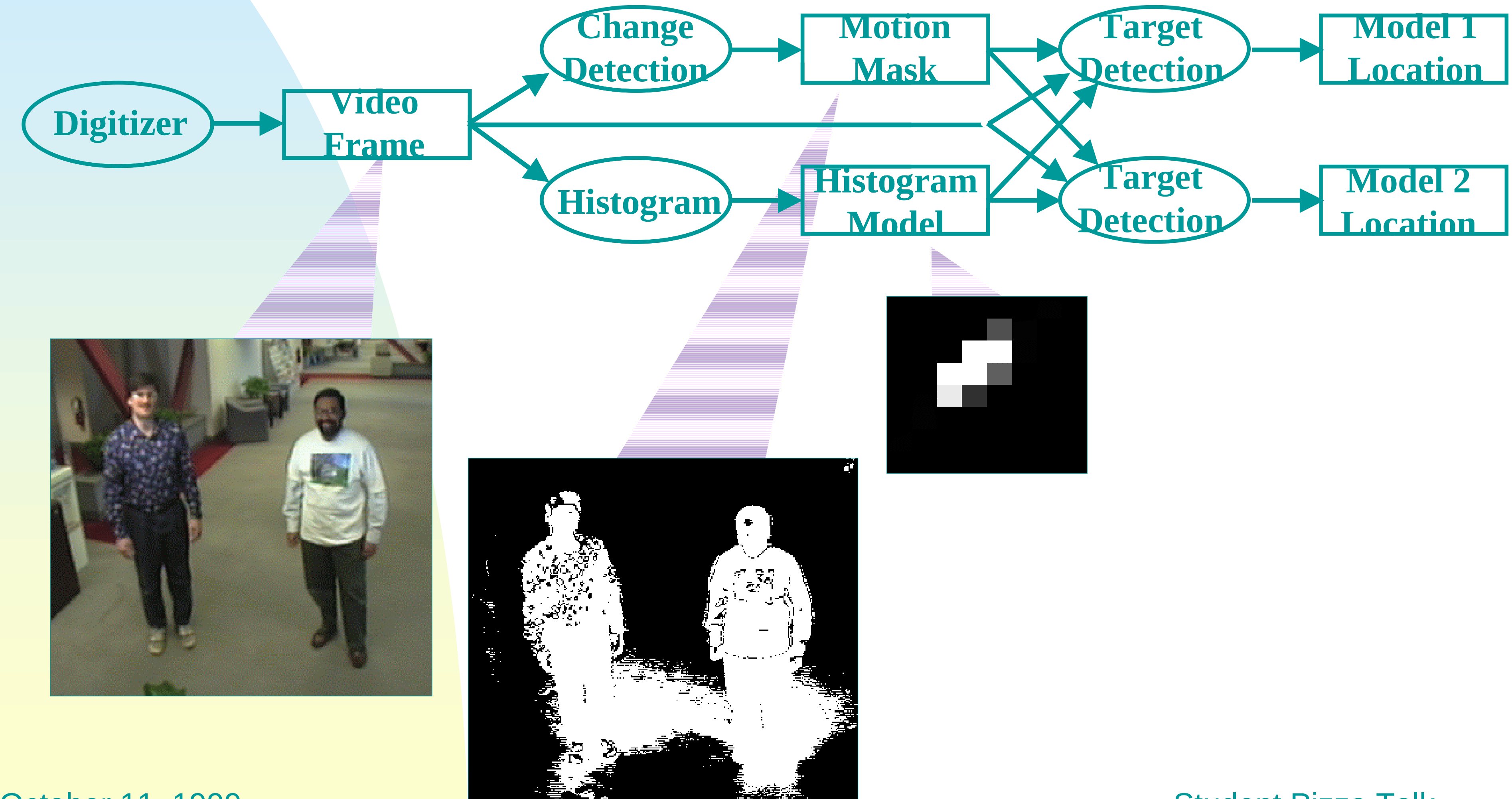
Operation of Color Tracker



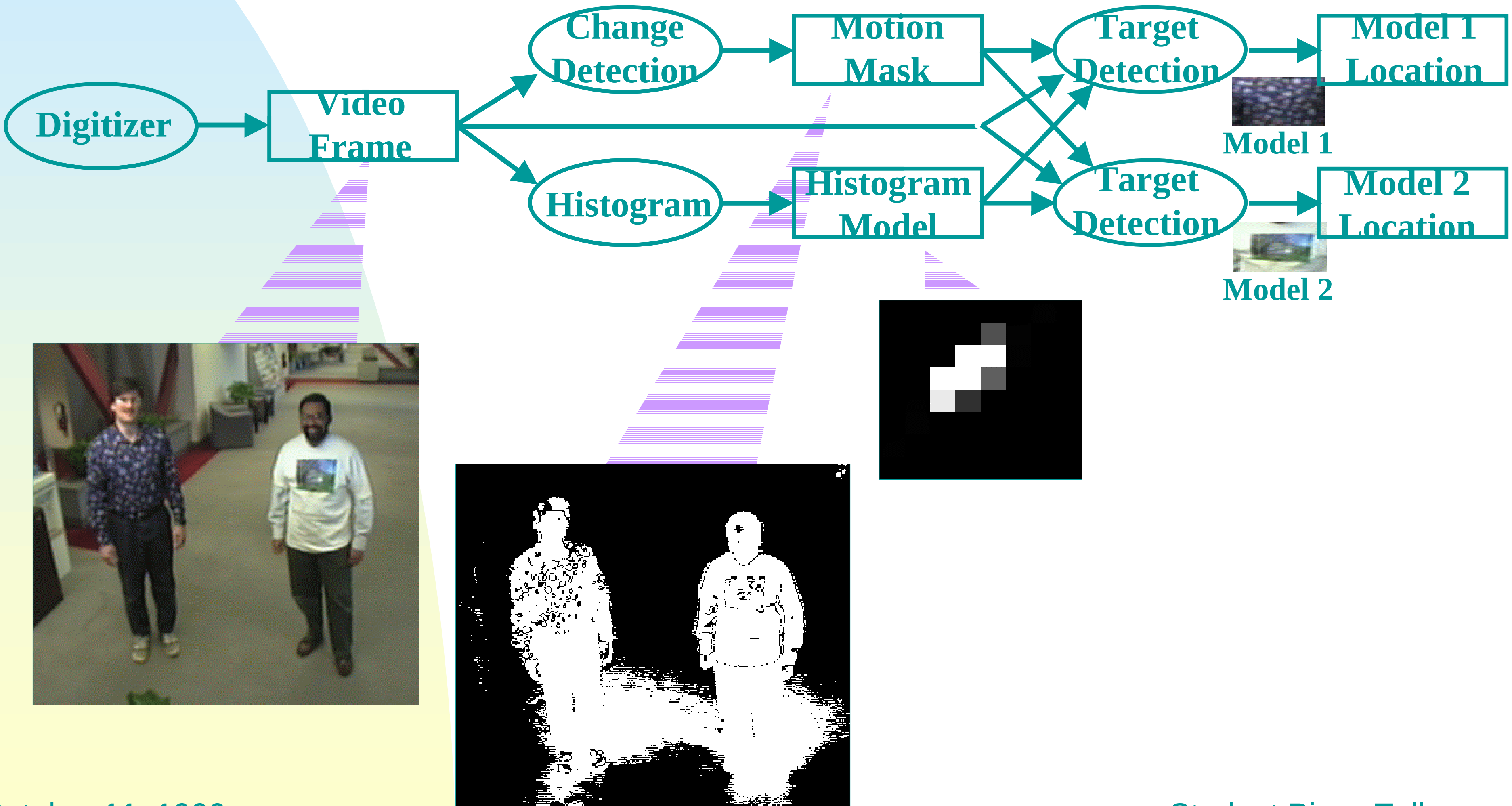
Operation of Color Tracker



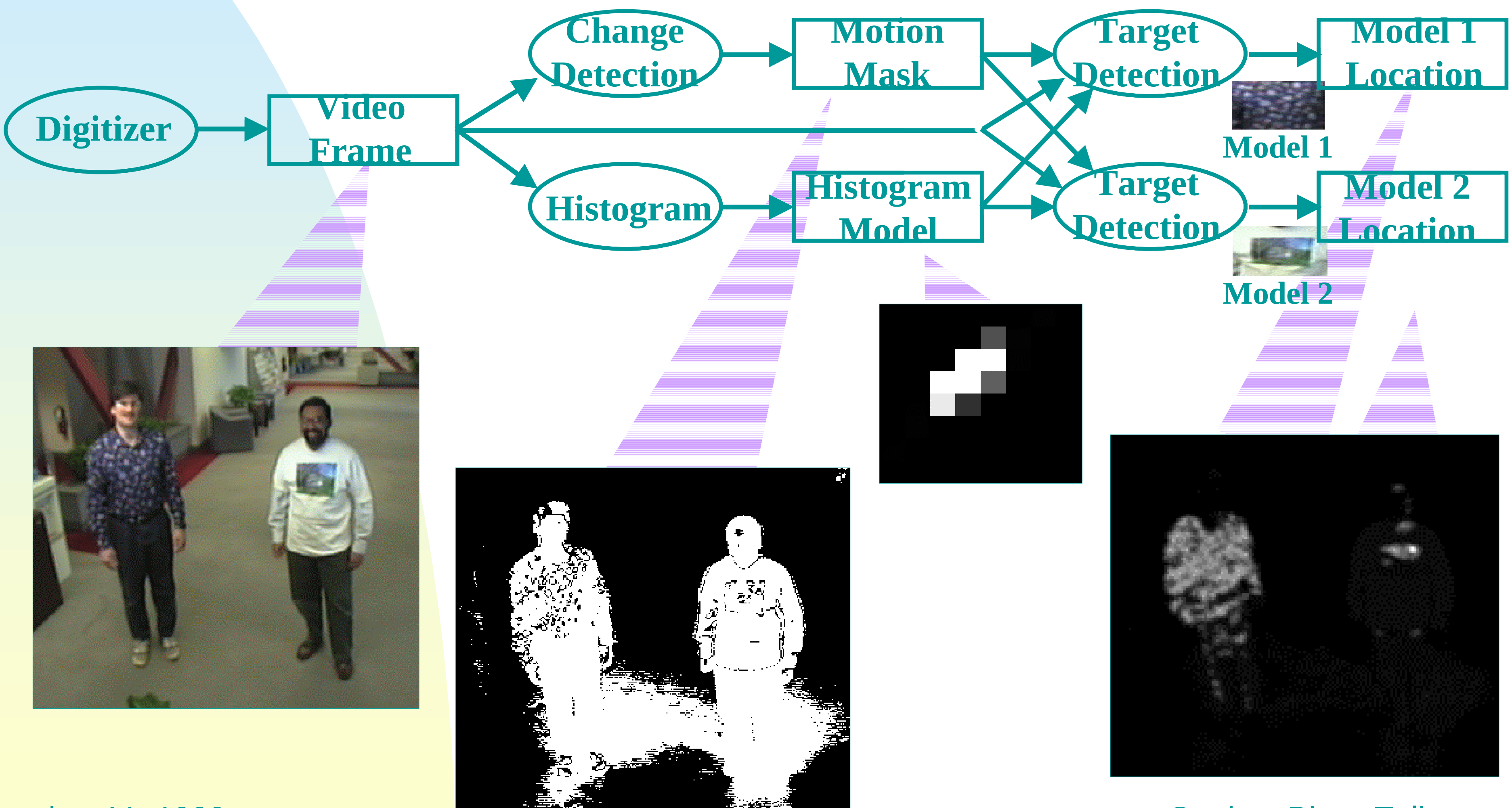
Operation of Color Tracker



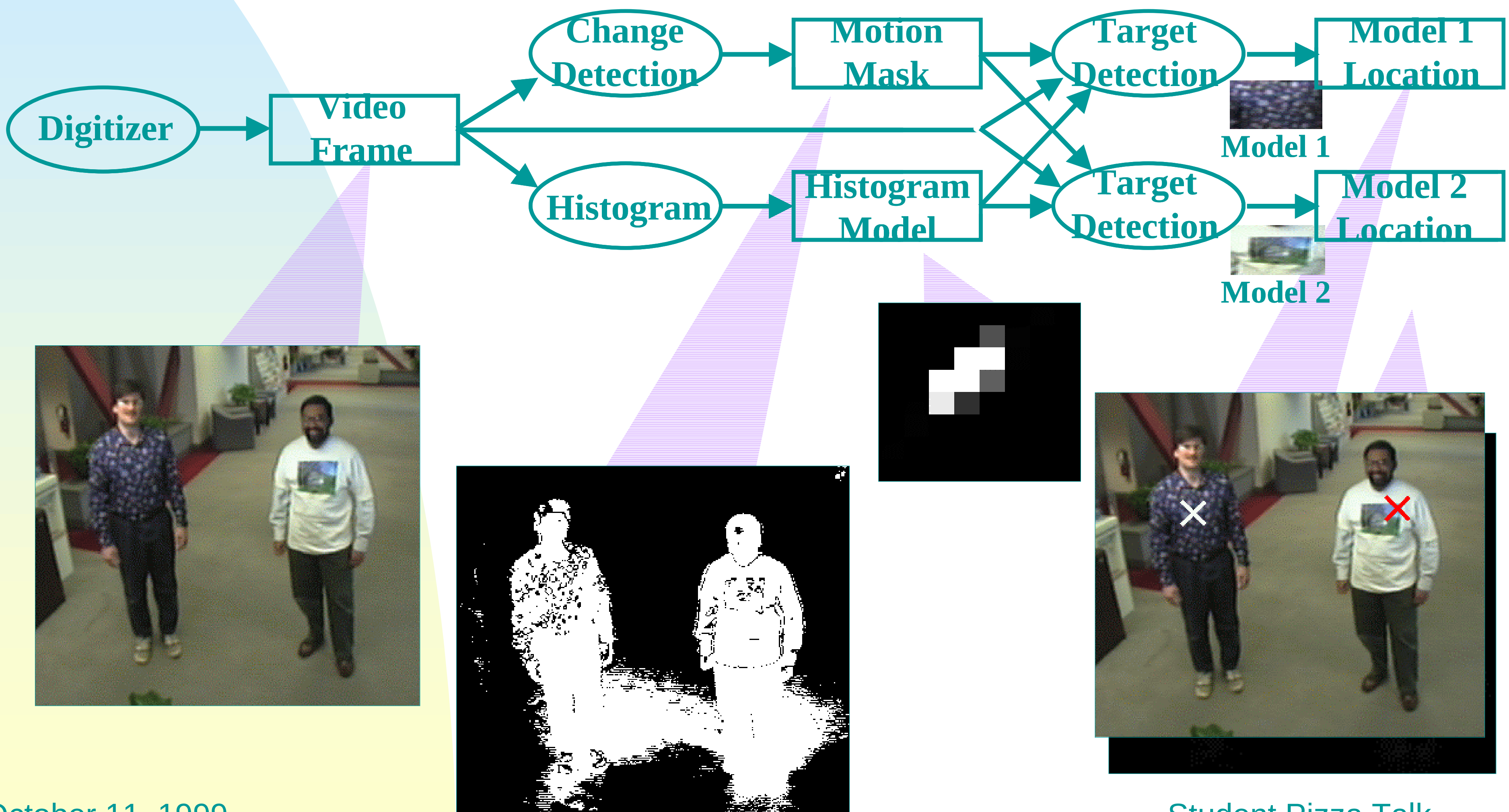
Operation of Color Tracker



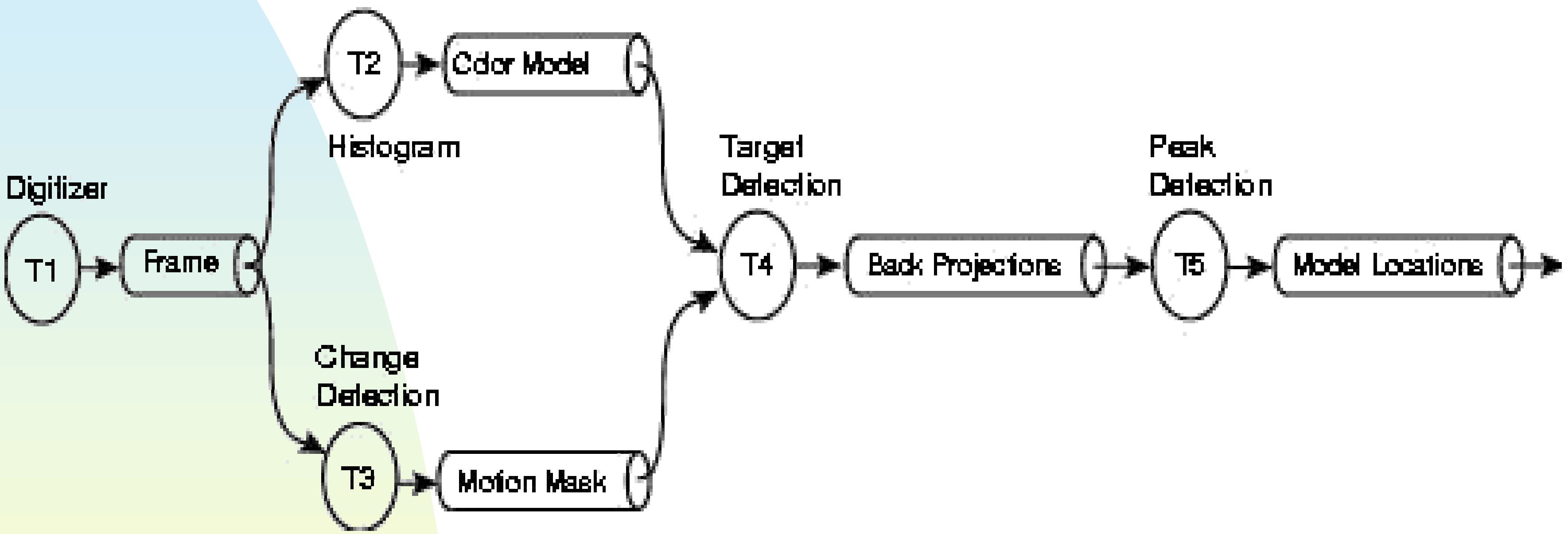
Operation of Color Tracker



Operation of Color Tracker



Closer look at Tracker



Characteristics

- Downstream tasks are more compute intensive
- Tasks “sample” the stream at varying rates
- Fundamental ability to sample at varying rates provided by Space-Time Memory

Characteristics

- Downstream tasks are more compute intensive
- Tasks “sample” the stream at varying rates
- Fundamental ability to sample at varying rates provided by Space-Time Memory

Raises scheduling questions

Metrics

- Kiosk must be compelling and interactive
  - low latency per frame
  - avoid “dead” periods
- Good use of resources
  - good throughput
- Use off-the-shelf OS and hardware

Metrics

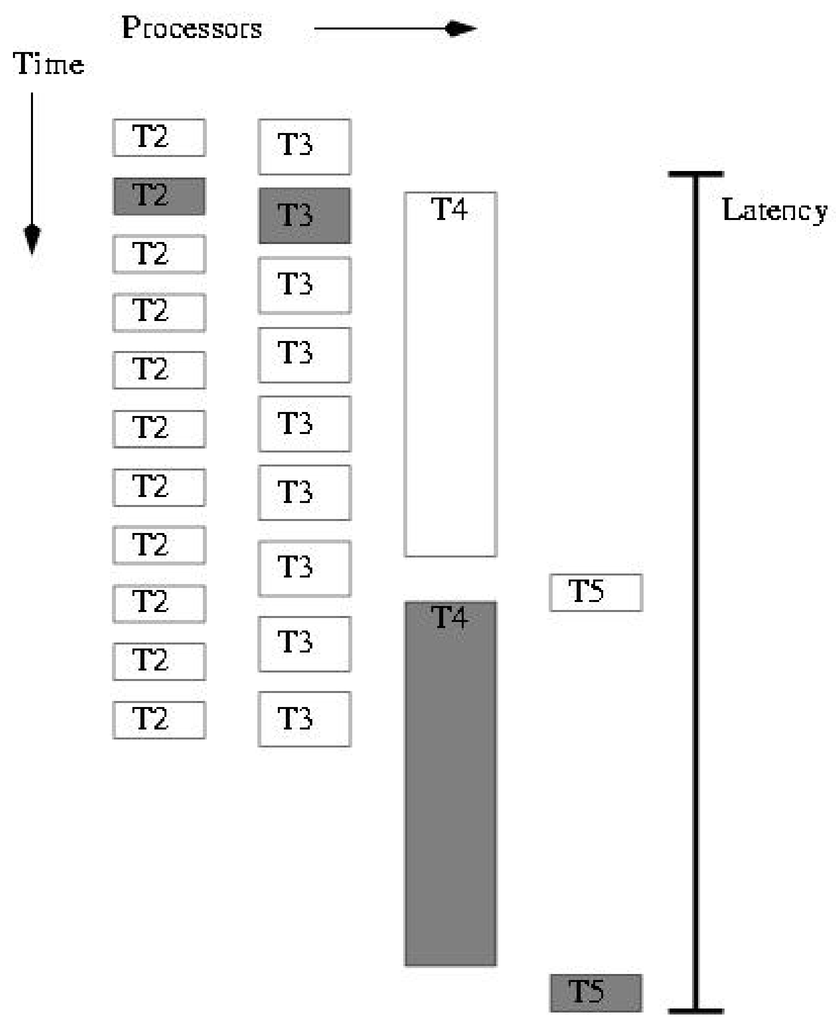
- Kiosk must be compelling and interactive
  - low latency per frame
  - avoid “dead” periods
- Good use of resources
  - good throughput
- Use off-the-shelf OS and hardware

Do this in a dynamic environment

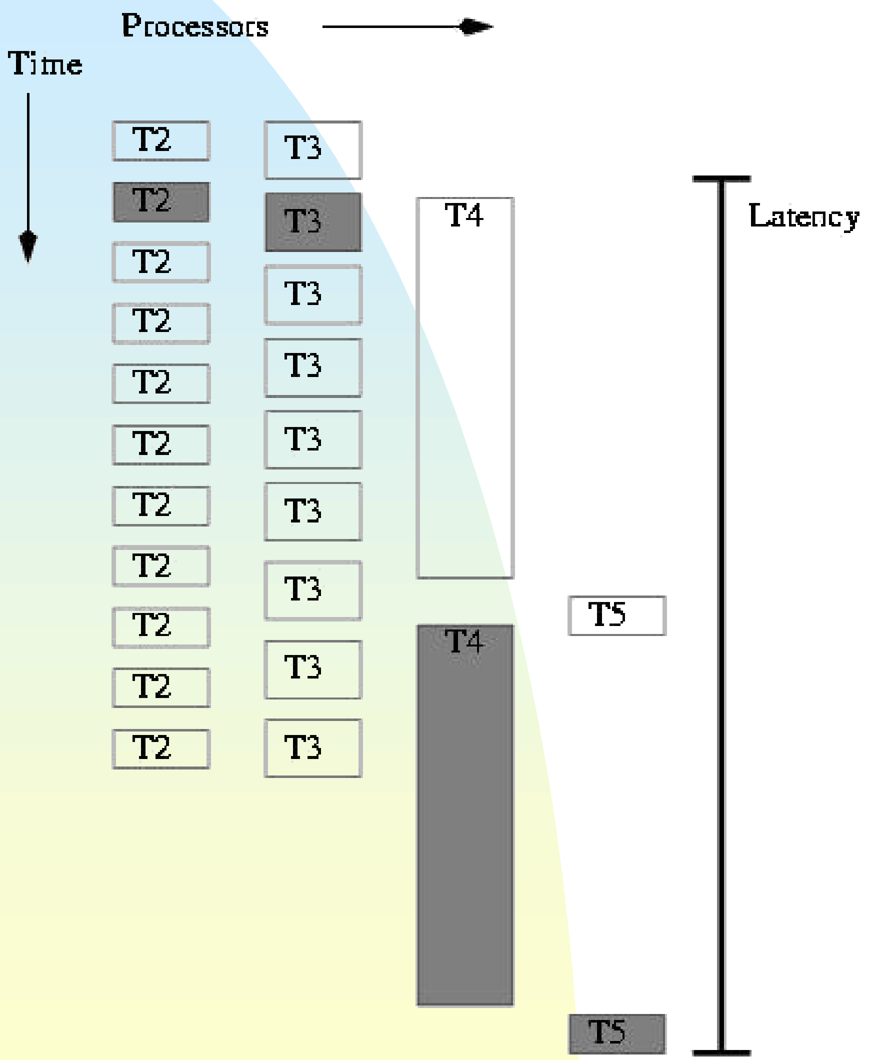
Constrained Dynamism

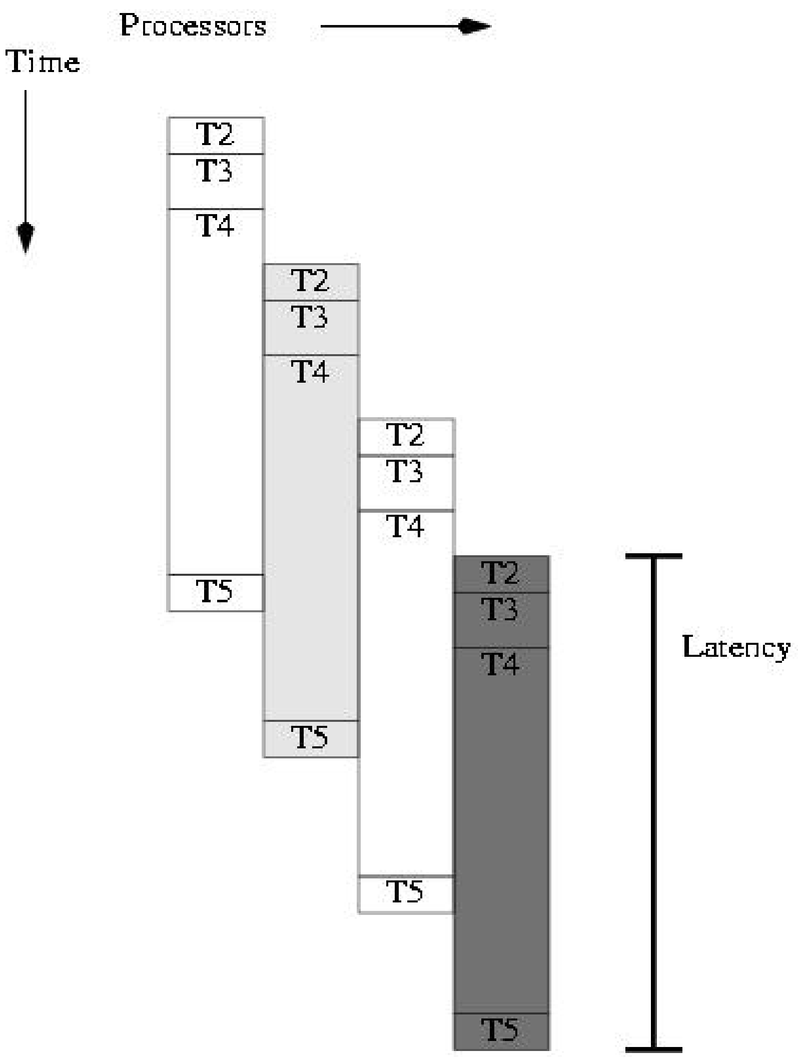
- The system changes among a small number of states
  - run-time environment, e.g., number of processors available, or load
  - input dependent
- State changes are infrequent
- State changes are detectable

Generic Thread Scheduler

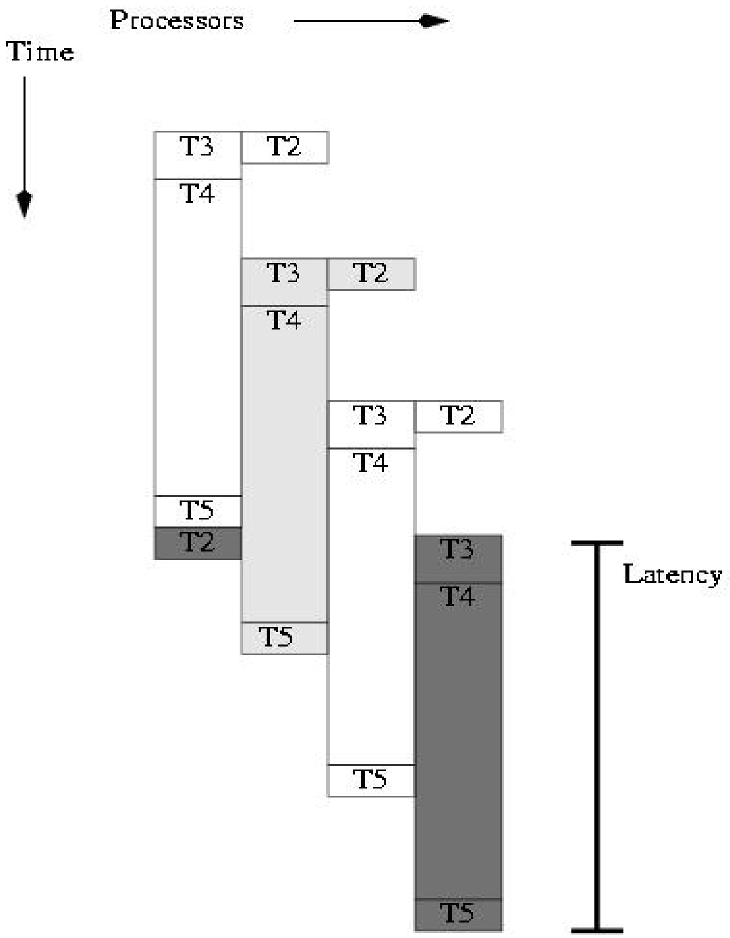


Generic Thread Scheduler

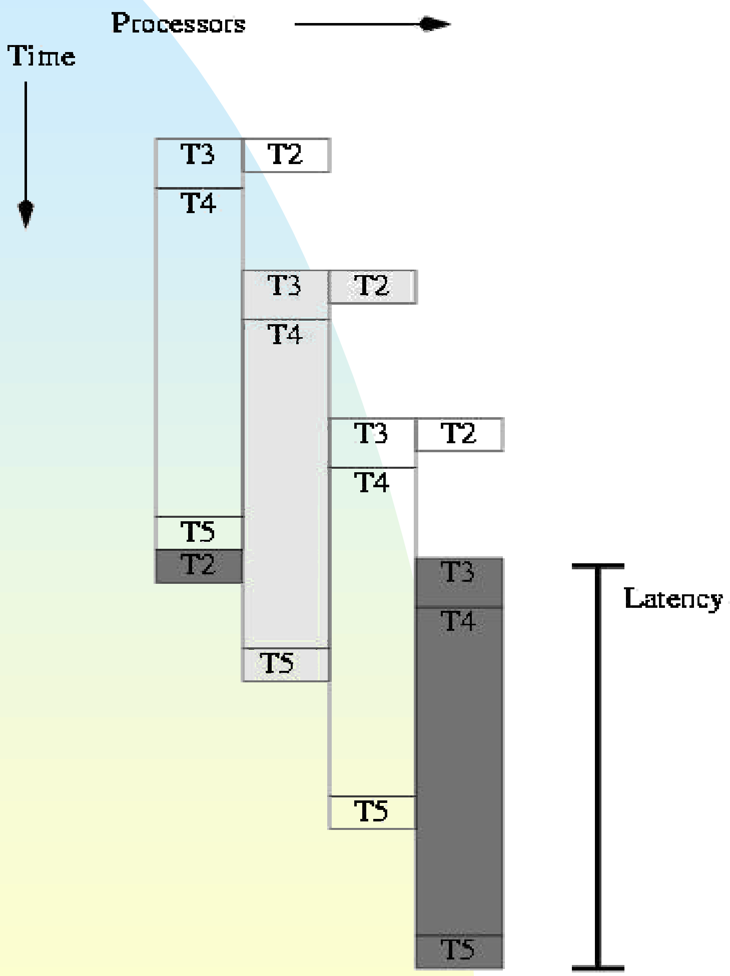


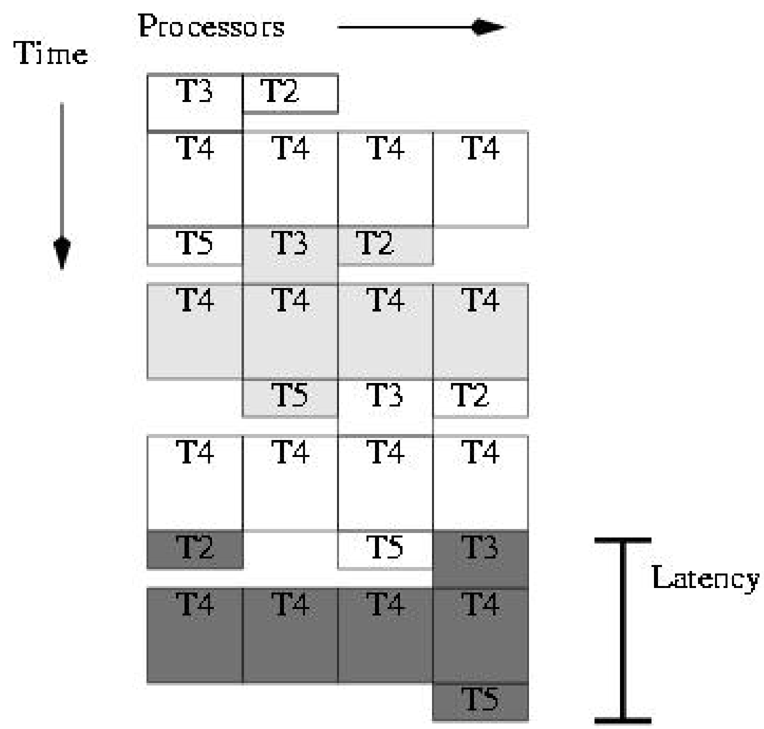


Better Schedules



Better Schedules





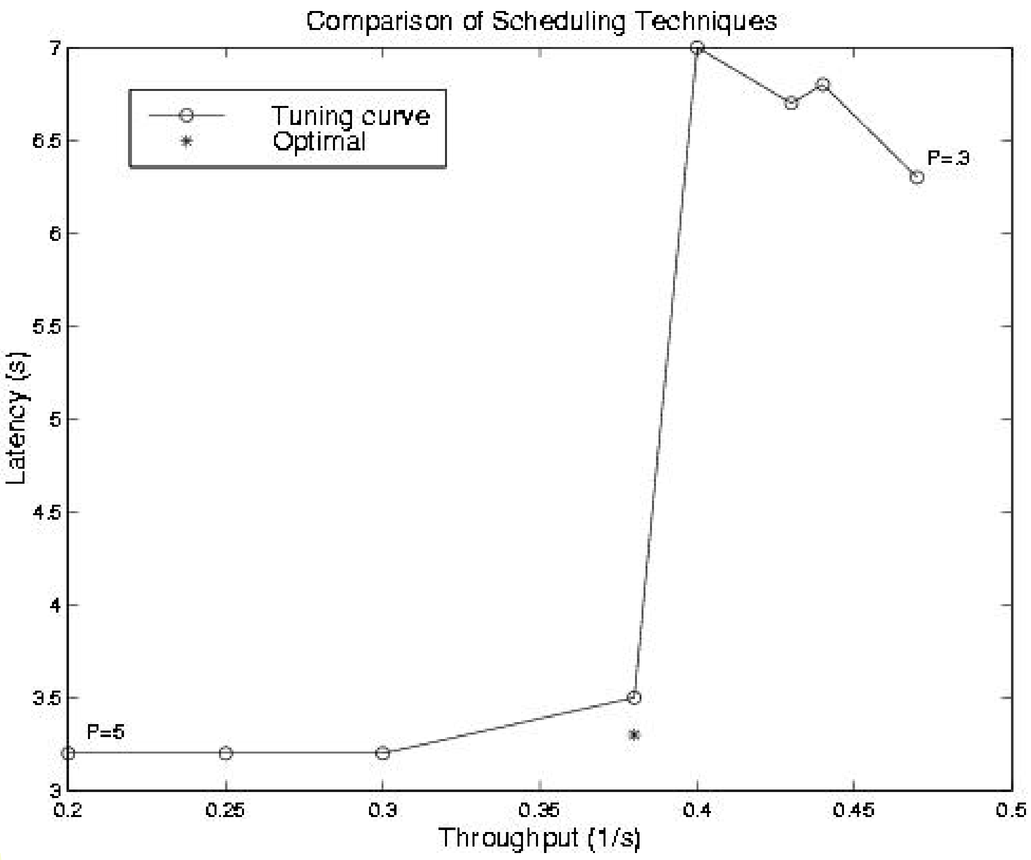
How does this work?

- Compute optimal schedules for each state
  - input: execution time, communication time
  - compute: minimal latency, single iteration schedule for minimal latency, and finally multiple iteration schedule
- Detect the current state at run-time and choose the best schedule

Why does this work?

- Alternatives
  - Do nothing!
  - Control rate of frame generation
  - Control the size of inter-task “channels”
- A limited number of states is the key

Comparison



Comparison of Scheduling Techniques
Tuning curve
Optimal
P=.3
P=5
Latency (s)
Throughput (1/s)
7
6.5
6
5.5
5
4.5
4
3.5
3
0.2
0.25
0.3
0.35
0.4
0.45
0.5

Benefits

- No extra work (good resource utilization)
- Reduces “live” time (smaller space req.)
- Simplifies garbage collection
- Implicitly solves flow control

Benefits

- No extra work (good resource utilization)
- Reduces “live” time (smaller space req.)
- Simplifies garbage collection
- Implicitly solves flow control

Works on off-the-shelf systems

Data Decomposition

	Total Models		
	1	8	
Partitions	MP=1	MP=8	MP=1
FP=1	0.876	1.857	6.850
FP=4	0.275	2.155	2.033

- Number of input models defines a state
- Small number of states ⇒ constrained dynamism

Conclusion

- A class of applications exhibits the property of “constrained dynamism”
- The property enables state-based approach to obtain good schedules in the face of dynamic environment
- Constrained Dynamism also helps in other aspects of application tuning, like, parallelization strategy