Towards Accessible Real-Time Distributed Embedded Vision Middleware

Cheng-Yao Chen, Jason Schlessman, Wayne Wolf
Embedded System Group
Department of Electrical Engineering
Princeton University

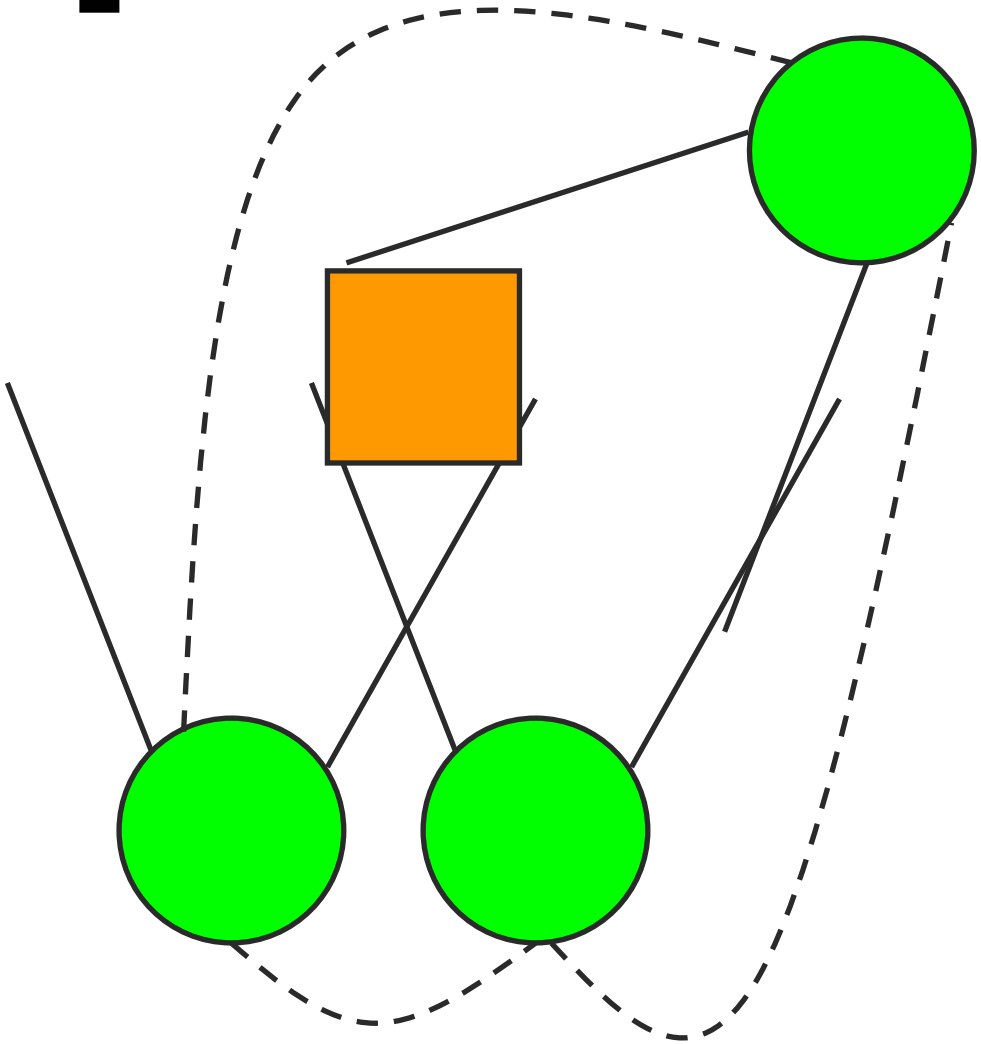
INTRODUCTION

- Visual computing with multiple video sensors can achieve a higher degree of robustness and reliability
- A peer-to-peer fashion of communication can be more efficient with respect to computation and communication
- However, peer-to-peer vision systems have unique communication patterns and requirements compared to other types of sensor networks
- Moreover, strict deadline requirements for real-time vision system further complicate the design
- Without careful evaluation of system resource constraints, the entire system could prove disastrous

Research Goal

- Most computer vision researchers focus on developing algorithms to perform the task on a single sensor
- Little or no consideration is given to distributed networks of cameras
- To provide a design tool in the form of layered middleware to assist in distributed computer vision system development is insightful

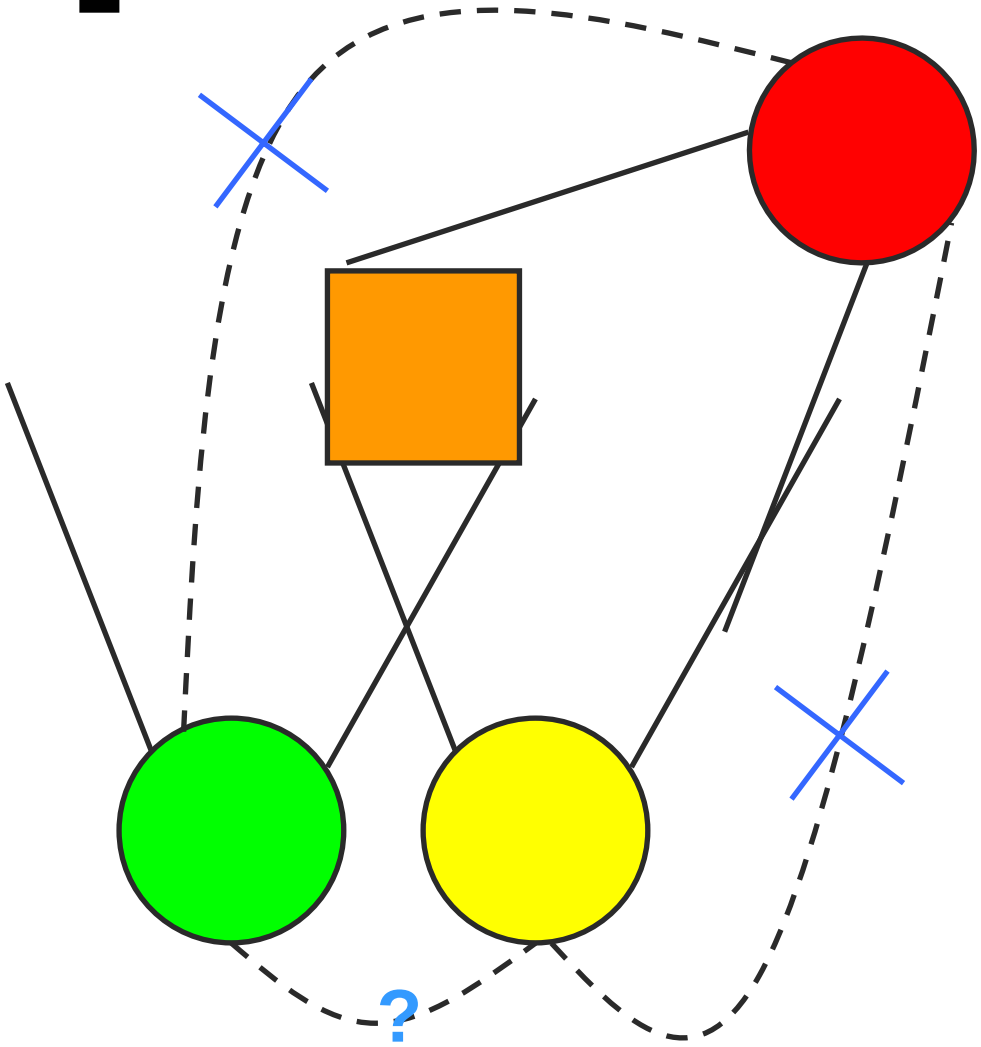
System Scenarios



Each camera is communicating with other overlapped cameras for objects within their overlapped areas

Green: not busy – the camera has plenty of room for extra computation made by other cameras' requests

Decision SITUATION



Connection to extremely busy camera should be temporarily disabled to preserve system reliability

Connection to somewhat busy camera can be either on or off depending on users' preferences

Green: not busy – the camera have plenty of room for extra computation made by other cameras' requests

Red: extremely busy – no room for extra computation

Yellow: somewhat busy – can process requests under certain circumstances

Middleware Overview

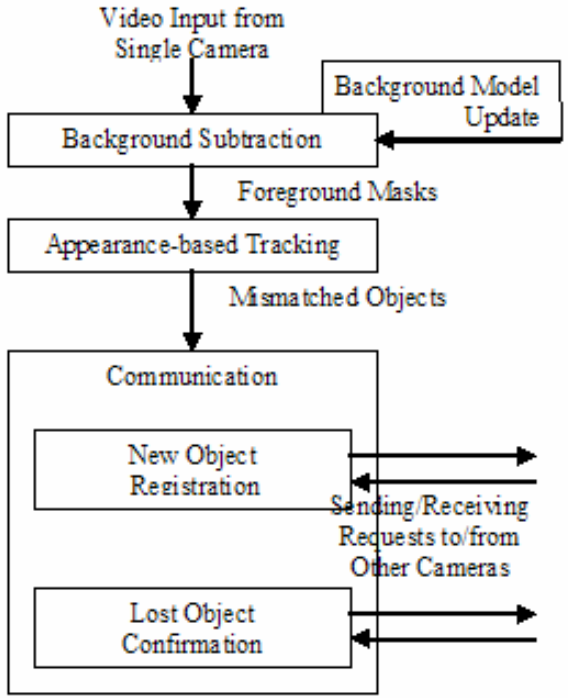
- Target peer-to-peer multiple object tracking system
- Small clusters of processing nodes could be made manifest within the system in a hierarchical fashion
- The decision criteria are described by six models, including, alternative model, system specification model, priority model, system failure model, error correction model, and resource constraint model

Alternative Model

- Defines the network configuration alternatives at run time
- In the multiple camera system, it usually refers to the cameras with which a given node need to communicate
- Each tracked object can also be descried by a set of overlapped sensors
- Object label coherence should be preserved within the overlapped sensor set

System Specification Model

- Describes each software and hardware component within the network system
- It could include descriptions of processor speed, memory size, and state machine of vision software task



An example of software components of a vision system

Priority Model

- Characterizes performance priorities as specified by the system user
- Vision-precision oriented – every node to try to talk to every other overlapped node if possible
- Power-efficiency oriented – where communication between nodes is minimized with correct functions

System Failure Model

- Three types of failures to characterize different faulty situations
- Parameter failures – due to inappropriate thresholds or random noises
- Vision failures – refer to internal algorithm limitations
- Communication failures – refer to errors result arising from a lack of or incorrect communication decisions
- Network link failure are not addressed since this class of failure can typically be alleviated by existing lower layer of network middleware

Error Correction Model

- Methods of failure recovery or correction for the system
- Vision systems can recover from either using extra communications or history updates
- Cost varies among different methods and preferences are determined by the user

Resource Constraint Model

- Describes the system resource constraints
- Typical examples – power consumption, processing thread limitations, process deadline requirements, and communication bandwidth boundaries

Preliminary Progress

- An activity model of characterizing higher level vision tracking activities in terms of lower level system computation and communication states
- Model the camera states by Markov Chain
- Predict the transition of states (e.g. not busy to busy) and make corresponding changes (e.g. temporarily break the link)
- Provide certain operation modes to manage different situations

Experiment Results

- Less than 5% prediction error can achieved when communicating 30 frame/second
- System failure rate with our proposed model can be reduced to at least lower than 1/3 of the original failure rate

Discussion

- A 100% prediction is difficult to achieve while the vision activity is complicated
- Time to collect necessary information and to calculate the operation mode can be the bottleneck of real-time processing

Conclusion

- A middleware that bridge higher level vision activities and lower level system specifications is proposed
- With this layered middleware, system reliability and efficiency are enhanced
- Users can also decide the configuration by their preferences
- A more accurate model needs to be explored for complex vision applications