Networked Control Systems for Robotics and Autonomy

Richard M. Murray
California Institute of Technology

Presentation to
Committee on Autonomy Research for Civil Aviation

28 August 2013

Goals
• Important trends in control over the last decade
• Networked control systems for robotics and autonomy
• Potential challenges for autonomy in civilian aviation
Some Important Trends in Control in the Last Decade

(Online) Optimization-based control
- Increased use of online optimization (MPC/RHC)
- Use knowledge of (current) constraints & environment to allow performance and adaptability

Layering and architectures
- Command & control at multiple levels of abstraction
- Modularity in product families via layers
- Platform-based design; contract-based design

Cyber-physical systems (CPS)
- Systems that combine information systems/physics
- Better coupling between computer science, controls, communications and networking

Formal methods for analysis, design and synthesis
- Formal methods from computer science, adapted for cyberphysical systems
- Horizontal & vertical contracts for layering, modularity

Components → Systems → Enterprise
- Movement of control techniques from “inner loop” to “outer loop” to entire enterprise (eg, supply chains)
Modern Networked Control System Architecture
(following P. R. Kumar, UIUC/Texas A&M)

- External Environment
- Actuation
- Process
- Sensing
- Online Model
- State Server (KF, MHE)
- Inner Loop (PID, $H_\infty$)
- Mode and Fault Management
- Trajectory Generation
- Online Optimization (RHC, MILP)
- Goal/Req’s Management
- Attention & Awareness
- Memory and Learning

- Feeder: Reliable
- 1-3 Gb/s
- Command:FIFO
- Actuator State: Unreliable
- 100 Kb/s
- Map: Causal
- 10 Mb/s
- State: Unreliable
- 100 Kb/s
- State Server (KF, MHE)
- Online Model
- Trajectory: Causal
- Mode and Fault Management
- Trajectory: Causal
- Online Optimization (RHC, MILP)
- Goal/Req’s Management
- Attention & Awareness
- Memory and Learning
Recent Example: Alice (DGC07)

Alice
- 300+ miles of fully autonomous driving
- 8 cameras, 8 LADAR, 2 RADAR
- 12 Core 2 Duo CPUs + Quad Core
- ~75 person team over 18 months

Software
- 25 programs with ~200 exec threads
- 237,467 lines of executable code
Application of existing controls technology in Alice

- Receding horizon (optimization-based) control for path planning with obstacles; ~100 msec iteration rate
- Multi-layer sensor fusion: sensor “bus” allows different combinations of sensors to be used for perceptors + fusion at “map” level
- Low level (inner loop) controls: PID w/ anti-windup (but based on a feasible trajectory from RHC controller)

Properties

- Highly modular
- Rapidly adaptable
- Constantly viable
- Robust ???
Protocol stack based architecture
- Planners use directives/responses to communicate
- Each layer is isolated from the ones above and below
- Had 4 different path planners under development, two different traffic planners.

Engineering principle: layered protocols isolate interactions
- Define each layer to have a specific purpose; don’t rely on knowledge of lower level details
- Important to pass information back and forth through the layers; a fairly in an actuator just generate a change in the path (and perhaps the mission)
- Higher layers (not shown) monitor health and can act as “hormones” (affecting multiple subsystems)

Hybrid system control methodology
- Finite state automata control interactions between layers and mode switches (intersection, off road, etc)
- Formal methods for analysis of control protocol correctness (post race)
  - Eg: make sure that you never have a situation where two layers are in conflict
Analysis vs Design: Optimization-Based Control

Nonlinear design
- global nonlinearities
- input saturation
- state space constraints

Local design

- Plant
- Local Control

Trajectory Generation
- $x_d$
- $u_d$
- noise

δ $u$

$\Delta$

Actual state

Computed state

Ref

time

state

$u_{[t,t+\Delta T]} = \arg \min_{t} \int_{t}^{t+T} L(x(\tau), u(\tau)) \, d\tau + V(x(t + T))$

$x_0 = x(t) \quad x_f = x_d(t + T)$

$\dot{x} = f(x,u) \quad g(x,u) \leq 0$

Offline design + analysis $\rightarrow$ online design

- Traditional: design (simple) controller, analyze performance, check with constraints
- Modern: specify performance and constraints, design trajectory/control to satisfy
- Problem: overall space too large; online optimization allows simplification
- Example of a “correct by construction” technique. Cost function = Lyapunov function

Links to complexity management

- Correct by construction allows “automatic” verification
- Still limited by our ability to formally specify behavior, computational tractability, etc
Challenge: Verification of Control Logic

Function: respond to control commands + DARPA pause/emergency stop

Verification using temporal logic (Lamport’s TLC, TLA+)
- Model follower, Actuation Interface, DARPA, accModule, transModule in TLC
- Shared variables: state, estop, acc, acc_command, trans, trans_command

Verify the following properties
- □((estop = DISABLE) ⇒ ◇□(state = DISABLED ∧ acc = -1))
- □((estop = PAUSE) ⇒ ◇(state = PAUSED ∨ estop = DISABLE))
- □((estop = RUN) ⇒ ◇(state = RUNNING ∨ state = RESUMING))
- □((state = RESUMING) ⇒ ◇(state = RUNNING ∨ estop = DISABLE ∨ estop = PAUSE))
- □((state ∈ {DISABLE, PAUSED, RESUMING, SHIFTING} ⇒ acc = -1))
Lessons Learned from Alice

Online optimization solves nonlinear control problems
• Modern computation allows constrained optimization problems to be solved online
• Solutions exist for situations with more limited computation (multi-parametric optimization)

Layered control architectures allow more efficient design
• Allows for “separation of concerns” between subsystems
• Provided a very modular design, capable of rapid (human-controlled) adaptation

Verification of control protocols is necessary, but hard
• Traditional methods of simulation and testing not sufficient
• Formal methods not easily applied to “hand designed” control protocols

New tools for “correct by construction” design are needed
• Temporal logic(s) are powerful language for specifying desired behavior (combined with traditional measures)
• New tools are becoming available, but not yet ready for prime time
Current Work: Design of Control Protocols

How do we design control protocols that manage behavior

- Mixture of discrete and continuous decision making
- Insure proper response external events, with unknown timing
- Design input = specification + model (system + environment)
- Design output = finite state machine implementing logic

Approach: rapidly explore all trajectories satisfying specs

- Search through all possible actions and events, discarding executions that violate a set of (LTL) specifications
- Issue: state space explosion (especially due to environment)
- Good news: recent results in model checking for class of specs
Temporal Logic Planning (TuLiP) toolbox
http://tulip-control.sourceforge.net

Python Toolbox
- GR(1), LTL specs
- Nonlin dynamics
- Supports discretization via MPT
- Control protocol designed using JTLV
- Receding horizon compatible

Applications of TuLiP in the last year
- Autonomous vehicles - traffic planner (intersections and roads, with other vehicles)
- Distributed camera networks - cooperating cameras to track people in region
- Electric power transfer - fault-tolerant control of generator + switches + loads
Example: Autonomous Navigation in Urban Environment

Traffic rules
- No collisions with other vehicles
- Stay in the travel lane unless there is an obstacle blocking the lane
- Only proceed through an intersection when it is clear

Assumptions
- Obstacle may not block a road
- Obstacle is detected before the vehicle gets too close to it
- Limited sensing range
- Obstacle does not disappear when the vehicle is in its vicinity
- Obstacles may not span more than a certain number of consecutive cells in the middle of the road
- Each intersection is clear infinitely often
- Each of the cells marked by star and its adjacent cells are not occupied by an obstacle infinitely often
Example: Autonomous Navigation in Urban Environment

- JTLV returns 900 state FSA in about 1.5 seconds
- $\Phi = \text{start in proper lane if no obstacle present } \wedge \text{no collision}$

**Use response mechanism to replan if no feasible solution exists**

- Trajectory planner sees blockage and fails to find strategy satisfying specification
- Trajectory planner reports failure to goal generator
- Goal generator re-computes a (high level) path to the goal state
Potential challenges for autonomy in civilian aviation

Two comparisons to previous areas of interest
• Control of UAVs in military operations (AFSAB study, 2003)
• Challenges in control of autonomous vehicles in urban environments

References
• “UAVs in Perspective,” Air Force. Scientific Advisory Board Summer Study, June 2003
With the possible exception of air to air combat, UAVs are capable of executing all current air force missions.

Advances in autonomy and human-system integration technologies are required to conduct increasingly complex missions.

The challenge is to optimally integrate human and machine abilities.
Challenges in Autonomous Driving

Systems integration
- Integration of complex sensing, actuation and decision-making subsystems
- Need to insure assumptions are consistent across algorithms (and teams)

Prediction and trust
- How do we anticipate the actions of other systems (autonomous or human-controlled)
- How do we make sure that autonomous systems behave in “understandable” manner

Interactions between agents
- Exploit the ability for autonomous (or semi-autonomous) systems to communicate
- Unfortunately, cannot assume that all vehicles will cooperate...

Learning
- Can autonomous systems learn from prior mistakes, other vehicles, online data?

Scaling up
- How do we “scale up” (speed, complexity) from operation in controlled environments

Verification and validation
- How do we verify and certify that the system can operate safely and robustly
- Particularly hard since we know that this cannot be done 100% of the time