
Workshop SC841

Unmanned Systems 101

Presenters: Kevin L. Moore, Colorado School of Mines
Nicholas Flann, Utah State University
Matthew Berkemeier, Autonomous Solutions, Inc.

SPIE 2007 Security & Defense Symposium
Orlando Florida

9 April 2007

Workshop Schedule

Time	Topic	Presenter
8:30-8:45	Introductions and Course Overview	Moore
8:45-10:00	Unmanned Systems: Components and Architectures	Moore
10:00-10:30	Break	
10:30-12:30	Control Algorithms for Unmanned Systems	Berkemeier
12:30-1:30	Lunch	
1:30-3:30	Intelligent Behavior Generation	Flann
3:30-4:00	Break	
4:00-5:15	Future Directions in Unmanned Systems	All
5:15-5:30	Wrap-up	Moore

Workshop Overview

- This course is intended to give a basic introduction to the technical aspects of unmanned systems.
- The course will emphasize:
 - Systems engineering perspective on the conceptual design and integration of the components of an unmanned system for a specific application.
 - Specify the components needed to implement an unmanned system
 - Task-appropriate architectures to integrate and coordinate the algorithms in an unmanned system.
 - Servo-level and path-tracking controllers, based on actuator modeling and system-level kinematics.
 - Navigation, motion planning, and intelligent behavior generation problems using high-level planning techniques based on searching.
- Concepts in the course will be illustrated by design examples and their implementation using Matlab™ and Mobius™.
- Course content will be generally applicable to all types of unmanned systems, but will focus on autonomous unmanned ground systems.

Presenters' Contact Information

Kevin L. Moore, Ph.D., P.E.

G.A. Dobelman Distinguished Chair and Professor of Engineering
Colorado School of Mines
Golden, Colorado
Email: kmoore@mines.edu
Web: egweb.mines.edu/faculty/kmoore/



Matthew Berkemeier, Ph.D.

Research Projects Manager and Senior Controls Engineer
Autonomous Solutions, Inc.
Wellsville, Utah
Email: mattb@autonomoussolutions.com



Autonomous Solutions, Inc.™

Nicholas, Flann Ph.D.

Associate Professor, Computer Science
Utah State University
Logan, Utah
Email: nick.flann@usu.edu
Web: <http://www.cs.usu.edu/~flann/>



Colorado School of Mines

Located in Golden, Colorado, USA
10 miles West of Denver



CSM sits in the foothills of the Rocky Mountains

CSM has about 300 faculty and 4000 students

CSM is a public research institution devoted to engineering and applied science, especially:

- Discovery and recovery of resources
- Conversion of resources to materials and energy
- Utilization in advanced processes and products
- Economic and social systems necessary to ensure prudent and provident use of resources in a sustainable global society

Utah State University

Located in Logan, Utah, USA
80 miles North of Salt Lake City



18,000 students study at USU's Logan campus, nestled in the Rocky Mountains of the inter-mountain west

CSOIS is a research center in the Department of Electrical and Computer Engineering



Autonomous Solutions, Inc. (ASI)

Located in Wellsville, Utah, USA
80 miles North of Salt Lake City



“Robots that Work!”



Workshop Schedule

Time	Topic	Presenter
8:30-8:45	Introductions and Course Overview	Moore
8:45-10:00	Unmanned Systems: Components and Architectures	Moore
10:00-10:30	Break	
10:30-12:30	Control Algorithms for Unmanned Systems	Berkemeier
12:30-1:30	Lunch	
1:30-3:30	Intelligent Behavior Generation	Flann
3:30-4:00	Break	
4:00-5:15	Future Directions in Unmanned Systems	All
5:15-5:30	Wrap-up	Moore

Workshop SC841 Unmanned Systems 101

Components and Architectures

Presenter: **Kevin L. Moore**, Colorado School of Mines

SPIE 2007 Security & Defense Symposium
Orlando Florida

9 April 2007

Acknowledgments

Professor D. Subbaram Naidu

- Idaho State University
- Measurement and Control Engineering Research Center

Professor YangQuan Chen

Professor Nicholas Flann

- Utah State University (USU)
- Center for Self-Organizing and Intelligent Systems (CSOIS)

Mr. Mel Torrie

- Autonomous Solutions, Inc.

David Watson

David Schiedt

Dr. I-Jeng Wang

Dr. Dennis Lucarelli

- Johns Hopkins Applied Physics Lab (APL)

ALL MY STUDENTS OVER THE YEARS!

Idaho State
UNIVERSITY



Utah State
UNIVERSITY

CSOIS
CENTER FOR SELF-ORGANIZING
AND INTELLIGENT SYSTEMS



Autonomous Solutions, Inc.™



Outline

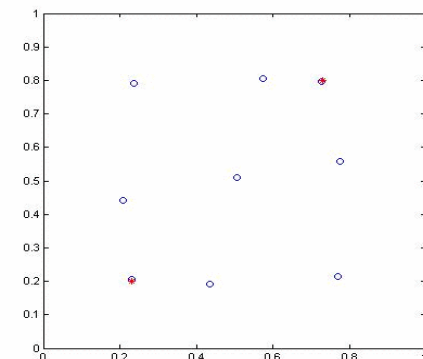
- **What is an Unmanned System?**
- Unmanned system components
 - Motion and locomotion
 - Electro-mechanical
 - Sensors
 - Electronics and computational hardware
- Unmanned system architectures
 - Multi-resolution approach
 - Software Architecture
 - Reaction, adaptation, and learning via high-level feedback

Unmanned Systems

- What is an **unmanned** system?
- What is an unmanned **vehicle**?
- Is an unmanned system a **robot**?
- Is a robot an unmanned system?
- Is an unmanned system an **autonomous** system?
- What about **unmanned sensors**?
- What about **mobile sensors**?
- What about telepresence or **tele-operation**?
- What about teams of unmanned vehicles, or **swarms**?



DARPA Crusher 1.0



Robots

From Wikipedia:

- A **robot** is a mechanical or virtual, artificial agent. A robot is usually an electro-mechanical system, which, by its appearance or movements, conveys a sense that it has intent or agency of its own. The word *robot* can refer to both physical robots and virtual software agents, but the latter are often referred to as *bots*.
- A robot may have the following properties:
 - Is not natural/has been artificially created
 - Can sense its environment
 - Can manipulate things in its environment
 - Has some degree of intelligence, or ability to make choices based on the environment, or automatic control / preprogrammed sequence
 - Is programmable
 - Can move with one or more axes of rotation or translation
 - Can make dexterous coordinated movements
 - Appears to have intent or agency

Robots

- The word “robot” was introduced in *R.U.R. (Rossum's Universal Robots)*, a 1921 science fiction play by Karl Čapek
 - The play begins in a factory that makes 'artificial people' - they are called robots, but are closer to the modern idea of androids or even clones, creatures who can be mistaken for humans. They can plainly think for themselves, though they seem happy to serve. At issue is whether the "robots" are being exploited and, if so, what follows?
- Typically, people think of “robots” as either manipulator arms or humanoid systems



Unmanned System

- Let us define:
 - **Unmanned system:** any electro-mechanical system which has the capability to carry out a prescribed task or portion of a prescribed task automatically, without human intervention
 - **Unmanned vehicle:** a vehicle that does not contain a person
 - Can be tele-operated
 - Can be autonomous
 - Typically deploys a payload (sensor or actuator)
- Focus today will be on **unmanned vehicles**
- Unmanned vehicles can come in several flavors: **UxV**
 - Land: UGV
 - Air: UAV
 - Maritime: UUV, USV
 - Sensors: UGS

Example UGVs



Bombot



Hydrema



Robo-Trencher



REDCAR



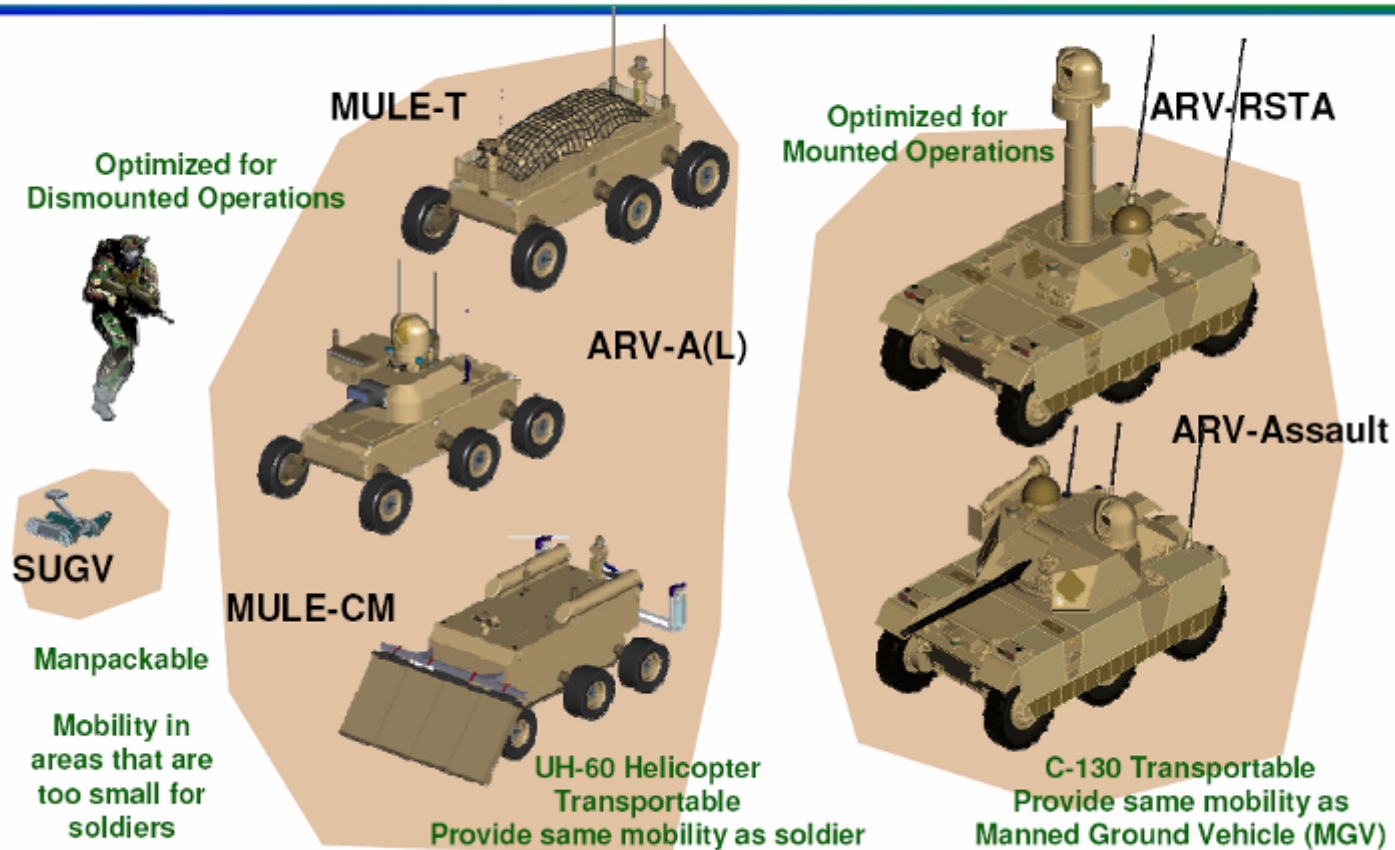
ARTS



Robotic ATV Carrier

The Future Combat System UGVs

FCS Unmanned Ground Vehicles Three Classes of UGVs



Approved for Public Release, Distribution Unlimited, PM FCS 23 JAN 2007, case 07-030

Slide due to Dr. Scott Fish, SAIC

Example UAVs



Fixed Wing



Rotary



Ducted Fan

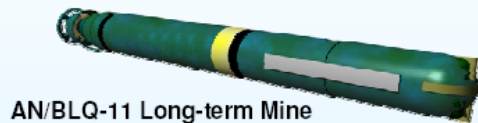
Example UUVs, USVs



Unmanned Maritime Vehicle Systems



Mission: Develop, acquire and maintain operationally superior and affordable surface and submarine launched Unmanned Maritime Vehicle Systems for the Warfighter



AN/BLQ-11 Long-term Mine Reconnaissance System (LMRS)

- MRUUVS risk reduction; SSN688/I torpedo tube launch and recovery
- Clandestine Mine Counter-Measures (MCM)
- ACAT II



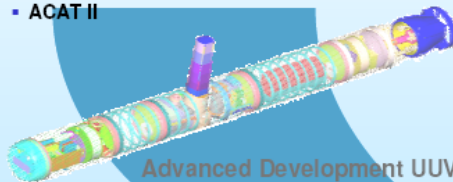
Battlespace Prep Autonomous Undersea Vehicle (BPAUV)

- MCM search and localization capability
- LCS Flight 0
- Direct leverage of ONR investment



Large Displacement MRUUVS

- Future development (ACAT Undesignated)
- Significantly improved range and payload capability
- SSGN, VIRGINIA, Littoral Combat Ship compatible
- Modular payloads open systems architecture
- Extended Anti-Submarine Warfare, Mine Countermeasure, Intelligence Surveillance and Reconnaissance (ISR), and Special Operations Forces missions



Advanced Development UUV

- MRUUVS risk reduction
 - Modular payloads
 - Reconfigurable vehicle design
 - Test-bed for future payloads
- Program Terminated FY06

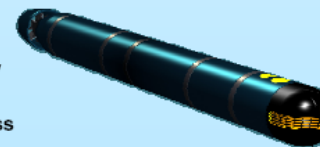


Surface Mine Countermeasure UUV (SMCM UUV)

- MCM search and localization capability
- UOES – User Operational Evaluation System
- MCM-1, LCS, Craft of Opportunity
- ACAT Program will detect buried mines

21" Mission Reconfigurable UUV System

- Modular payloads
- Open systems architecture
- Clandestine SSN launch and recovery
- Pre-MDAP
- Current Program – MCM 688/688I Class
- Future Capability – ISR and 774 Class



USV

- MCM Sweep, Delivery, and Neutralization
- ISR / Gun Payloads
- SOF Support



2

Mr. Mike Alperi
Deputy Program Manager
PMS403

Slide due to

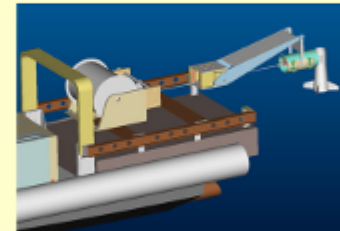
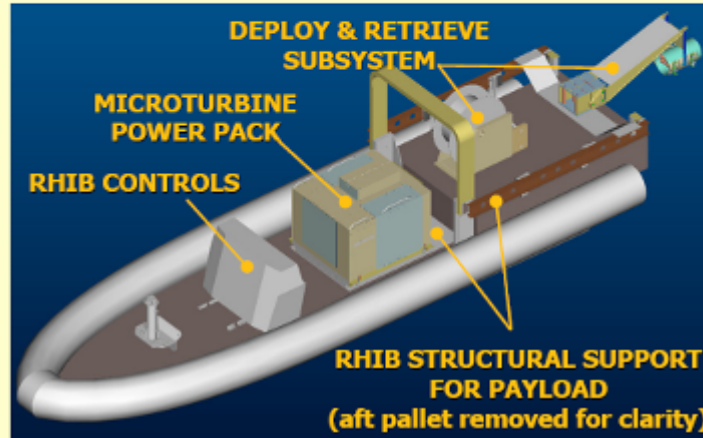
USV Application



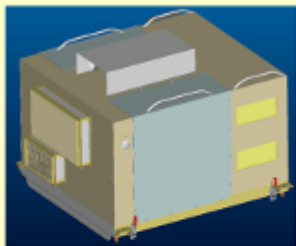
USV – Minesweeping



RHIB Host Platform:
Remote Control Piloting
While Sweeping



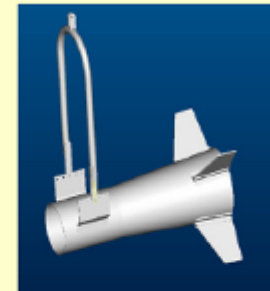
Deploy & Retrieve:
Automated Handling of
Influence Sweep



Power Pack:
Provides All Influence
Sweep Electrical Power



Magnetic Sweep:
Generates Subsurface Magnetic Field



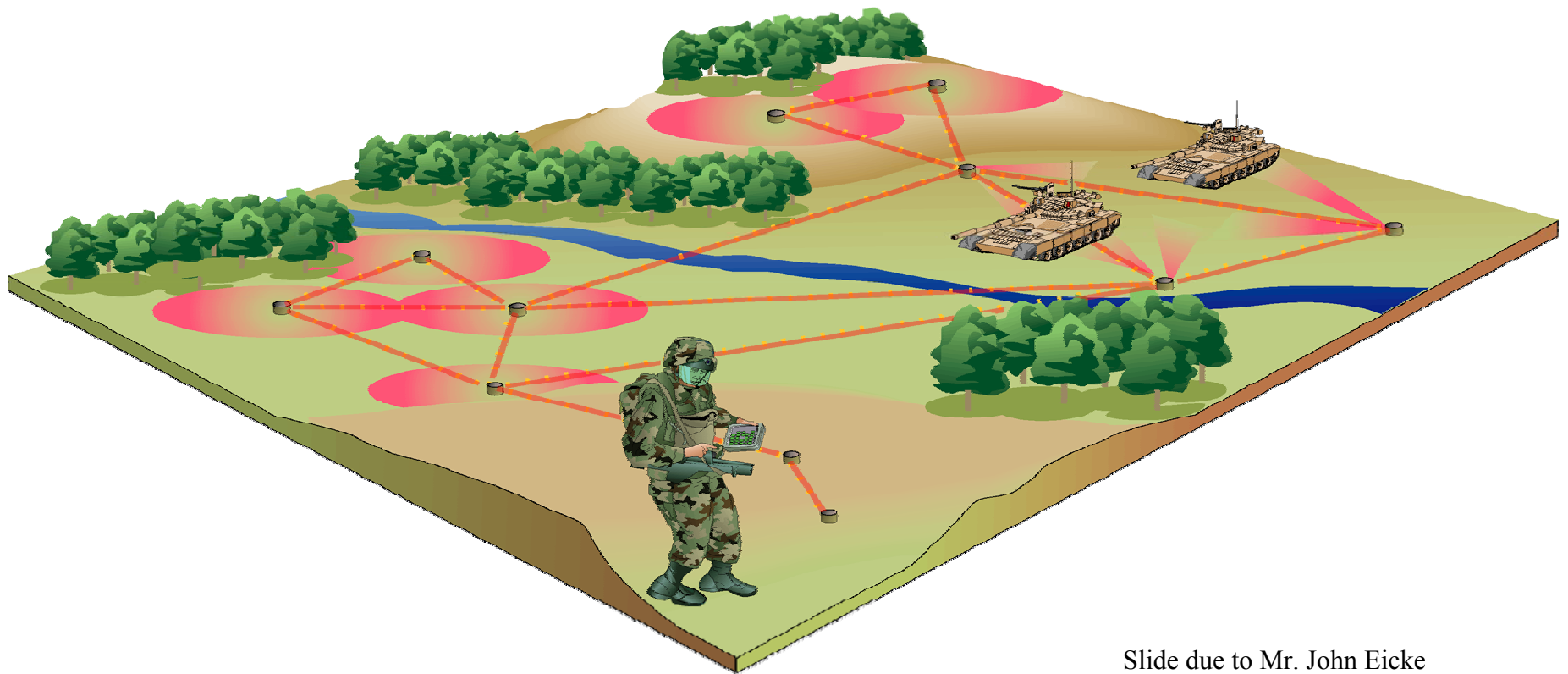
Acoustic Sweep:
Generates Subsurface
Acoustic Field

In Testing December 2006; Demonstration in AUV Fest June 2007

AUVSI Unmanned Systems Prog Rev_8Feb07.ppt

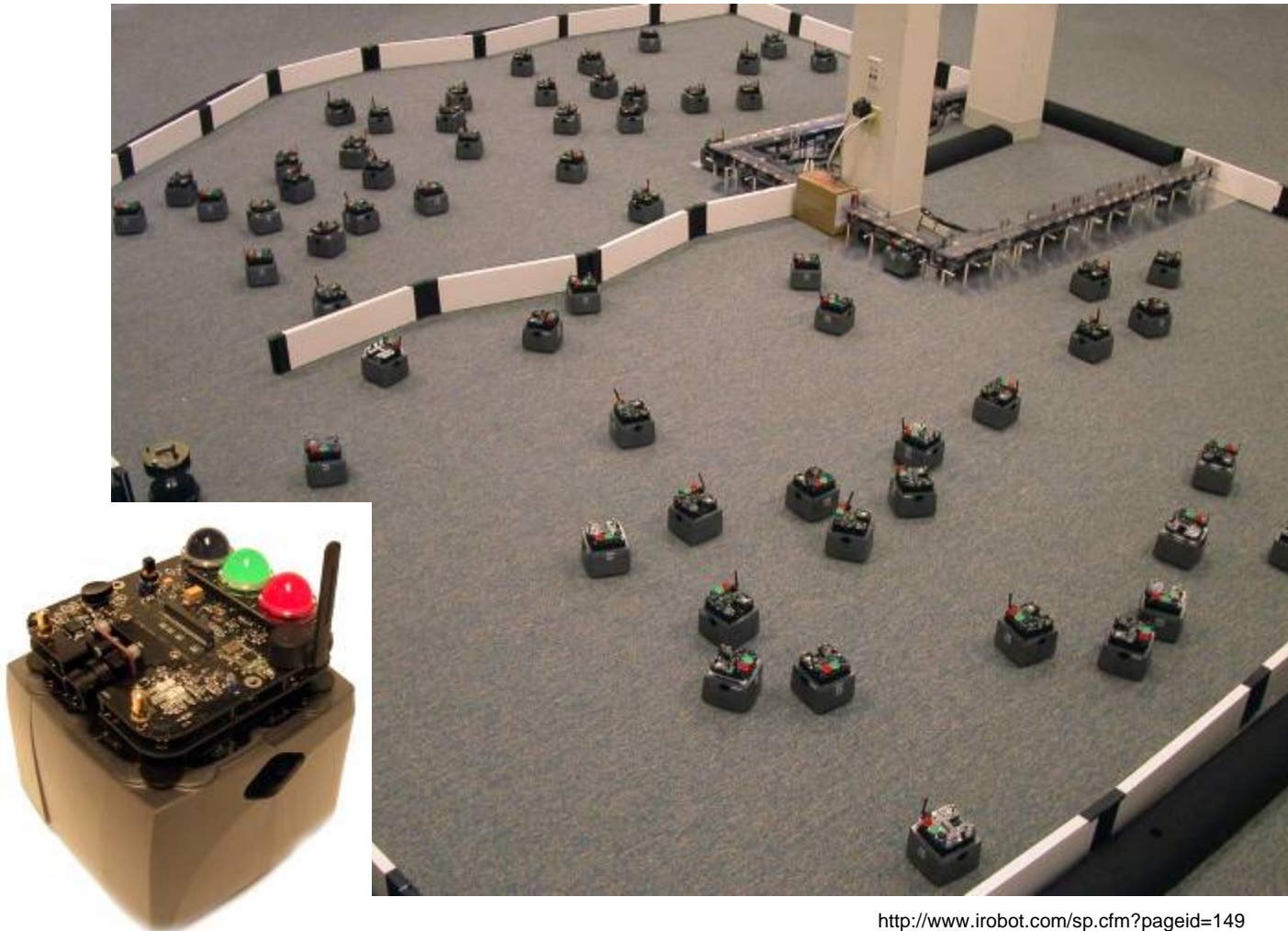
Slide due to
Dr. Tom Swain, Jr.
Office of Naval Research

Unattended Ground Sensors



Slide due to Mr. John Eicke
U.S. Army Research Laboratory

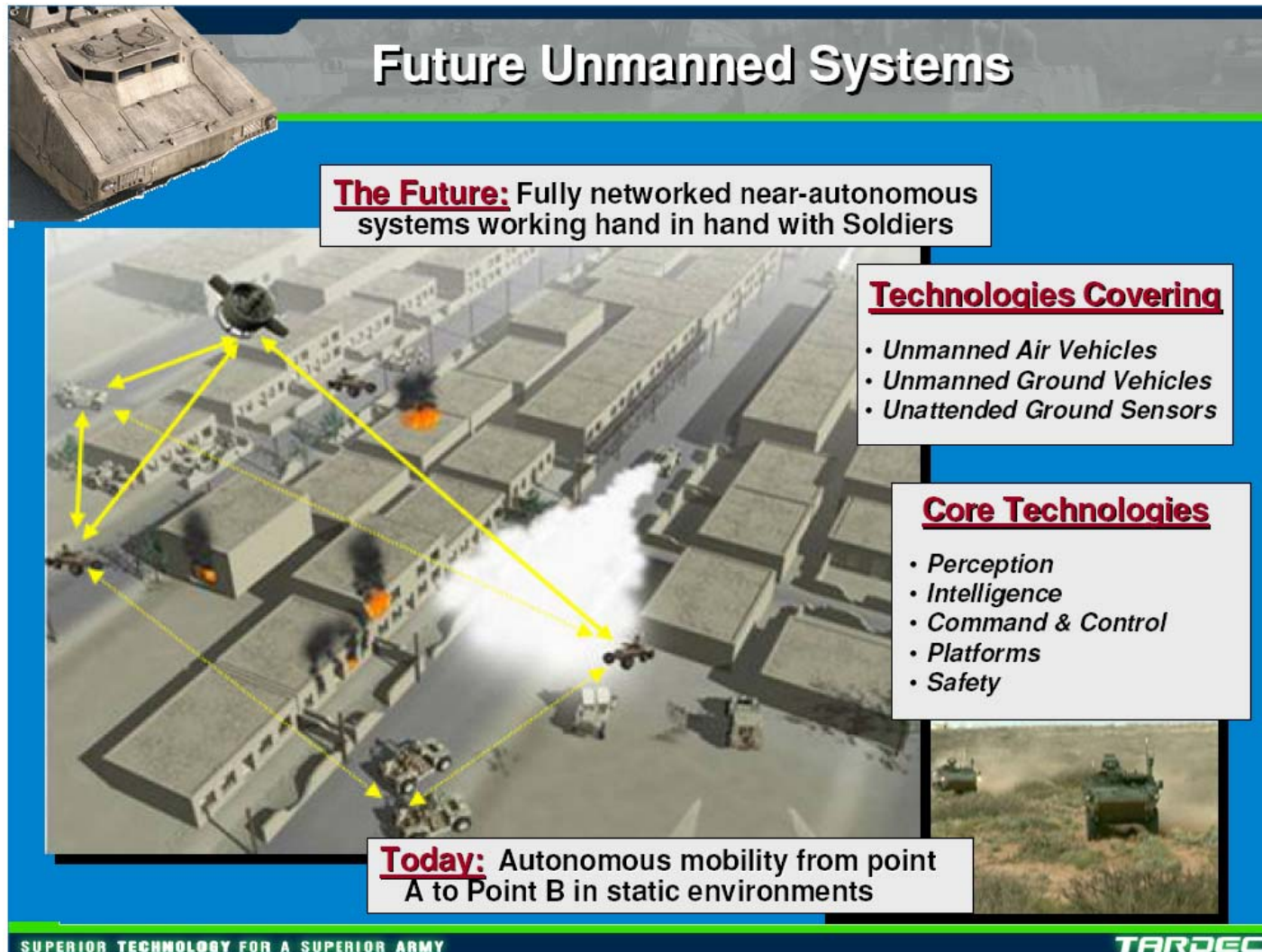
Mobile Sensors/Swarms of Today?



<http://www.irobot.com/sp.cfm?pageid=149>

The SwarmBot™ used at MIT built by IRobot

Networked UxVs – Swarms of the Future?



Future Unmanned Systems

The Future: Fully networked near-autonomous systems working hand in hand with Soldiers

Technologies Covering

- Unmanned Air Vehicles
- Unmanned Ground Vehicles
- Unattended Ground Sensors

Core Technologies

- Perception
- Intelligence
- Command & Control
- Platforms
- Safety

Today: Autonomous mobility from point A to Point B in static environments

SUPERIOR TECHNOLOGY FOR A SUPERIOR ARMY

TARDEC

Slide due to

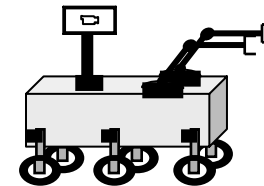
Paul D. Rogers, Ph. D. - Executive Director of Research
TARDEC - Warren, MI

GiG-Level UxVs Networks – Meta-Swarms of the Future?

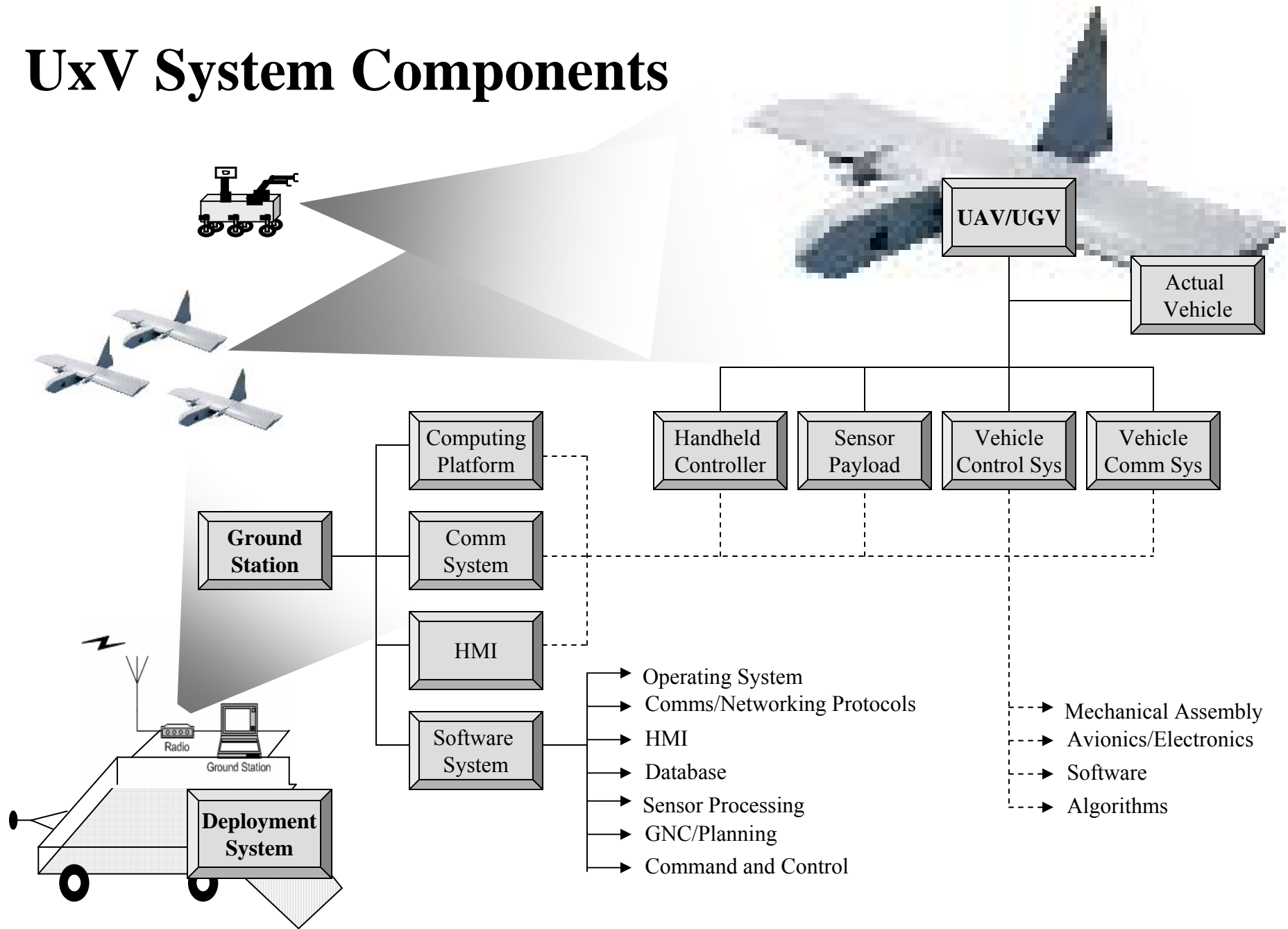


What Makes a UxV?

- All UxVs have common elements:
 - Mechanical components (drive, power, chassis)
 - Electronics
 - Sensing/mission payloads
 - Communication systems
 - Control
 - “Smarts”
 - Interface to user



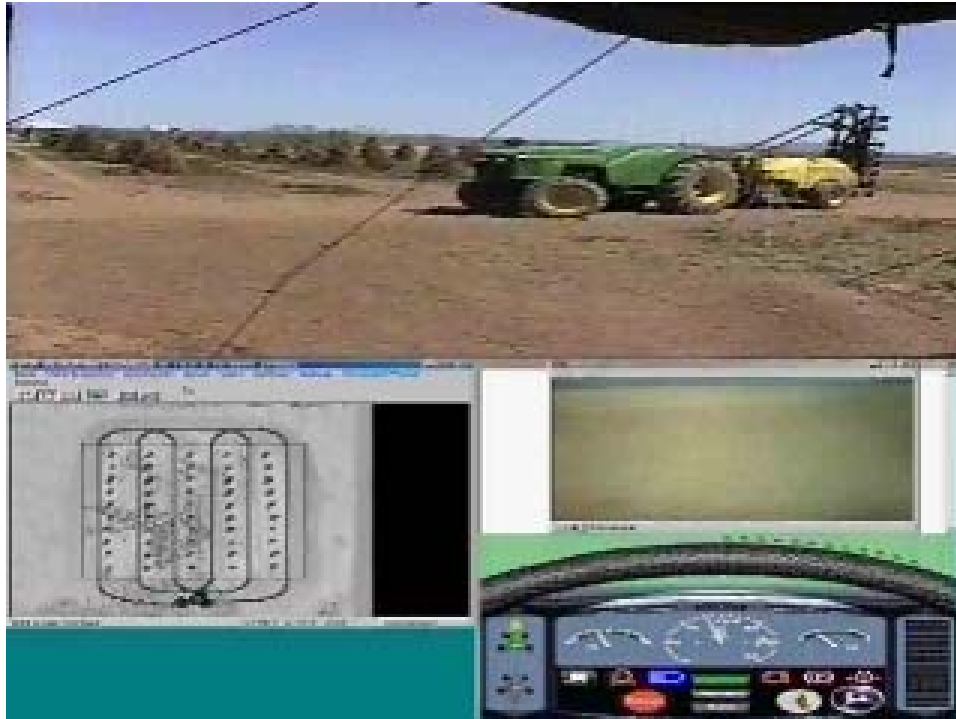
UxV System Components



What Makes a UxV?

- All UxVs have common elements:
 - Mechanical components (drive, power, chassis)
 - Electronics
 - Sensing/mission payloads
 - Communication systems
 - Control
 - “Smarts”
 - Interface to user
- However, our perspective is that all unmanned systems should be developed from the perspective of its concept of operations (CONOPS)

Example CONOPS - Automated Tractors

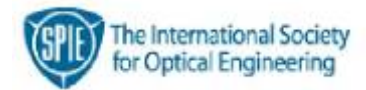


Curved Path Following

Example CONOPS -Unique
Mobility Robots →
(Autonomous Solutions, Inc.)



Joint Ground Robotics Enterprise (JGRE) CONOPS



Workshop SC841:
Unmanned Systems 101

Mission	Company	BCT	Division	Ranking
Reconnaissance	1	1	1	1
Mine Detection/CM	2	2	2	2
Precision Target Location and Designation	3	3	5	3
Chem/Bio Reconnaissance	6	4	3	4
Weaponization/Strike	4	6	6	5
Battle Management	8	5	4	6
Communications/Data Relay	5	7	7	7
Signals Intel	7	8	8	8
Covert Sensor Insertion	9	9	10	9
Littoral Warfare	13	10	9	10
Counter Cam/Con/Deception	10	11	11	11
SOF Team Resupply	11	12	12	12
Combat SAR	12	13	13	Slide due to Mrs. Ellen M. Purdy Enterprise Director, Joint Ground Robotics OUSD(ATL)/PSA/LW&M
Information Warfare	14	14		
Decoy/Pathfinder	15	15		



What Makes a UxV?

- All UxVs have common elements:
 - Mechanical components (drive, power, chassis)
 - Electronics
 - Sensing/mission payloads
 - Communication systems
 - Control
 - “Smarts”
 - Interface to user
- However, our perspective is that all unmanned systems should be developed from the perspective of its concept of operations (CONOPS)
- Once a CONOPS has been defined, then systems engineering is used to flow-down requirements for subsystems.

Unmanned Systems: Capabilities and Control



COLORADO SCHOOL OF MINES

- We consider two key aspects of unmanned vehicles and autonomy:
 - Inherent physical capabilities built into the system
 - Intelligent control to exploit these capabilities
 - **Inherent physical capabilities**
 - Mechanisms for mobility and manipulation
 - Power
 - Sensors for perception
 - Proprioceptive
 - External
 - Computational power
 - **Intelligent control to exploit these capabilities**
 - Machine-level control
 - Perception algorithms
 - Reasoning, decision-making, learning
 - Human-machine interfaces
- These are driven by your CONOPS**
- These are driven by your CONOPS and by your inherent physical capabilities**

Outline

- What is an Unmanned System?
- **Unmanned system components**
 - **Motion and locomotion**
 - **Electro-mechanical**
 - **Sensors**
 - **Electronics and computational hardware**
- Unmanned system architectures
 - Multi-resolution approach
 - Software Architecture
 - Reaction, adaptation, and learning via high-level feedback
- Toward an algorithmic framework for autonomous UxVs

Case Study/Illustrative Example

- In following, we discuss
 - Inherent capability in unmanned ground systems
 - Exploitation of these inherent capabilities
- Primarily use the Omni-Directional Inspection System (ODIS) as a case study
 - Inherent capability in ODIS
 - Exploitation of these inherent capabilities



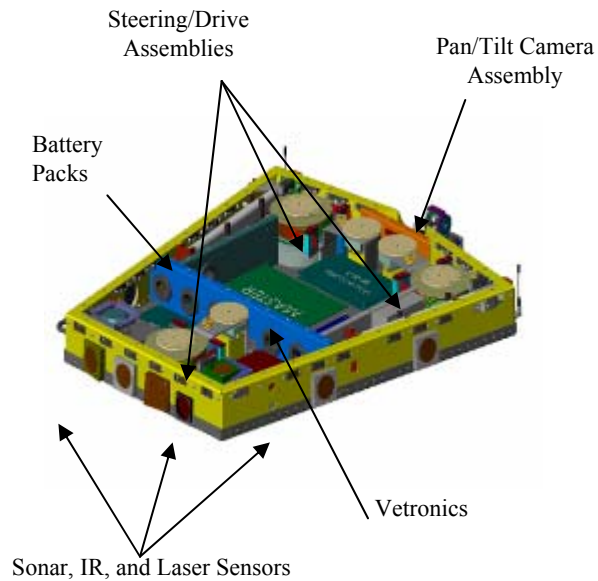
ODIS I – An Autonomous Robot Concept



ODIS I Description



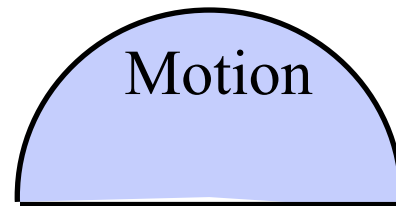
- Laser Rangefinder
- IR Sensors
- Sonar
- FOG Gyro
- 3 Wheels



“Mirror-on-a-Stick” vs. ODIS



UGV Technology



Motion and Locomotion for Unmanned Systems

Except for UGS, most unmanned systems must move:

- UGV: wheels and tracks
- UAV: fixed wing, rotary wing, VTOL
- USV/UUV: propeller based, jetted

In general the motion and locomotion aspects of an unmanned vehicle are not remarkably different than that of their manned counterparts:

- Design of motion and locomotion system becomes “only” an engineering task!



Typical UGV Mobility Platforms

199

- Ackerman
- Skid-Steer
- Unicycle
- Unique Mobility

over

Autonomous
wheelchair

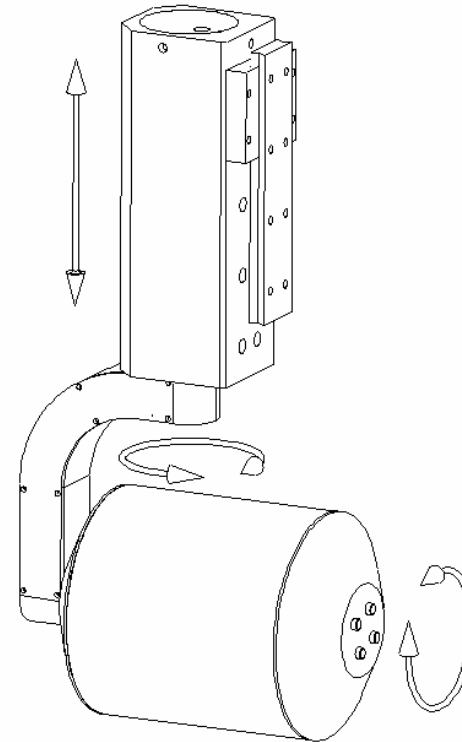
1996-1998
Arc III

1997-1998
Predator

Predator with
ARC II ³⁹

Mobility Example: USU ODV Technology

- USU developed a mobility capability called the “smart wheel”
- Each “smart wheel” has two or three independent degrees of freedom:
 - Drive
 - Steering (infinite rotation)
 - Height
- Multiple smart wheels on a chassis creates a “nearly-holonomic” or omni-directional (ODV) vehicle



T1 Omni Directional Vehicle (ODV)



ODV steering gives improved mobility compared to conventional steering



Smart wheels make it possible to simultaneously

- Translate
- Rotate

Some ODV Robots Built At USU

T1 -1998



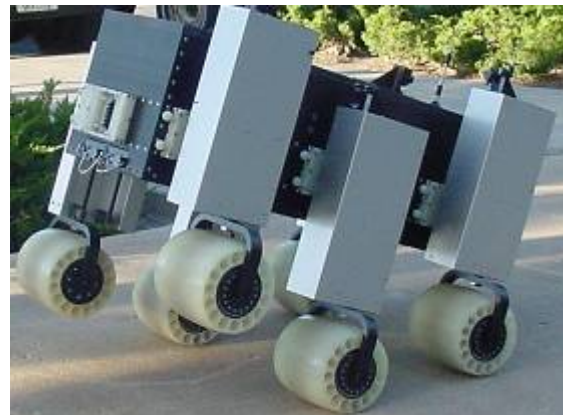
T2 -1998



ODIS I -2000



T4 -2003



T3 -1999

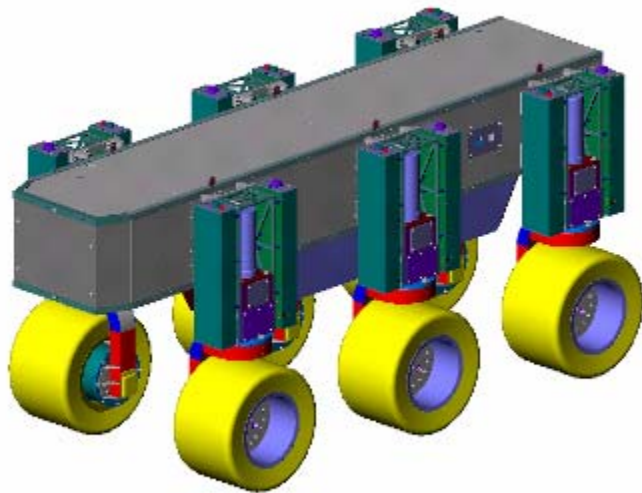
(Hydraulic drive/steer)



C
S
O
I
S

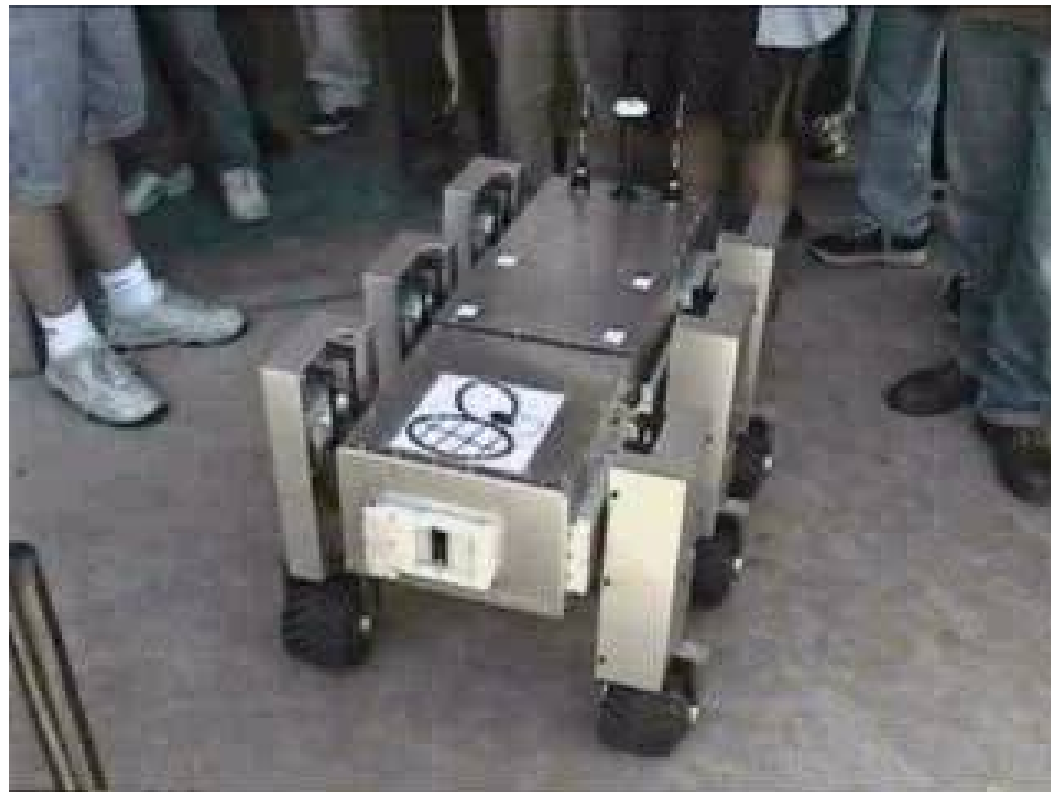


T2 Omni Directional Vehicle



T2 can be used for military scout missions, remote surveillance, EOD, remote sensor deployment, etc.

T3 ODV Vehicle





Omni-Directional Inspection System (ODIS)

- First application of ODV technology
- Man-portable physical security mobile robotic system
- Remote inspection under vehicles in a parking area
- Carries camera or other sensors
- Can be tele-operated, semi-autonomous, or autonomous

“Putting Robots in Harm’s Way So People Aren’t”

ODIS – the Omni-Directional Inspection System

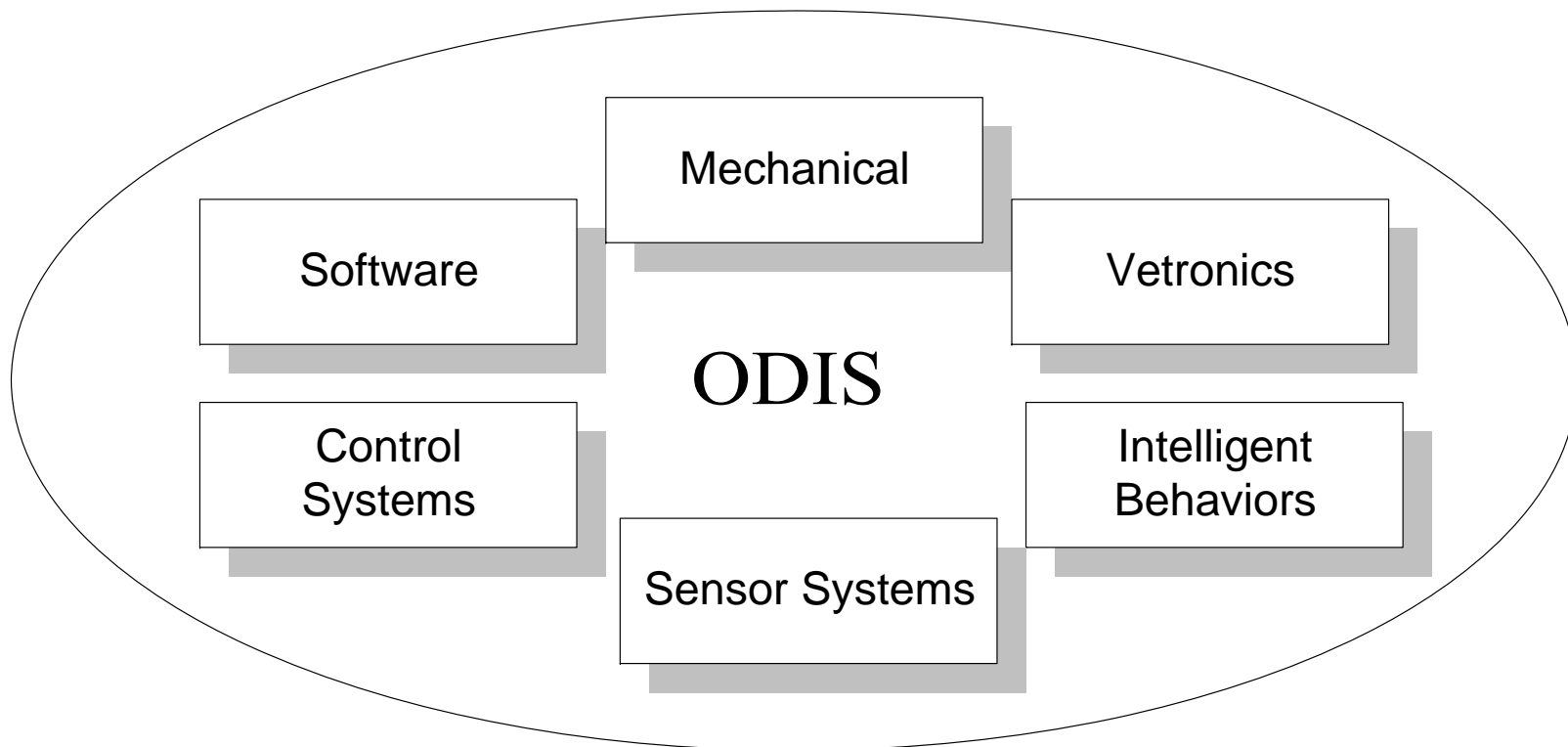
An ODV Application: Physical Security



Under joystick control¹⁷

Systems Engineering a UGV: Case Study

ODIS Design and Implementation

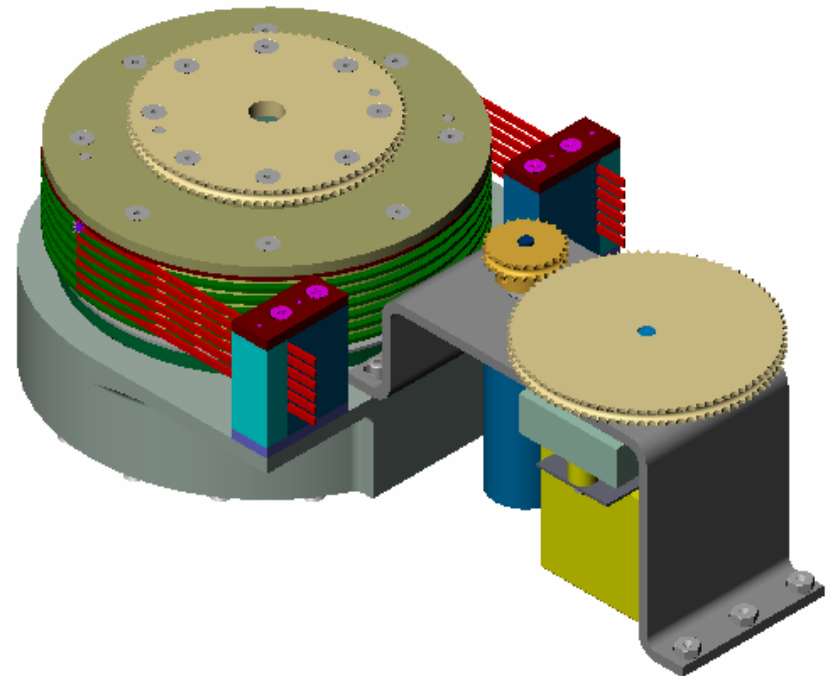


Overall Specifications

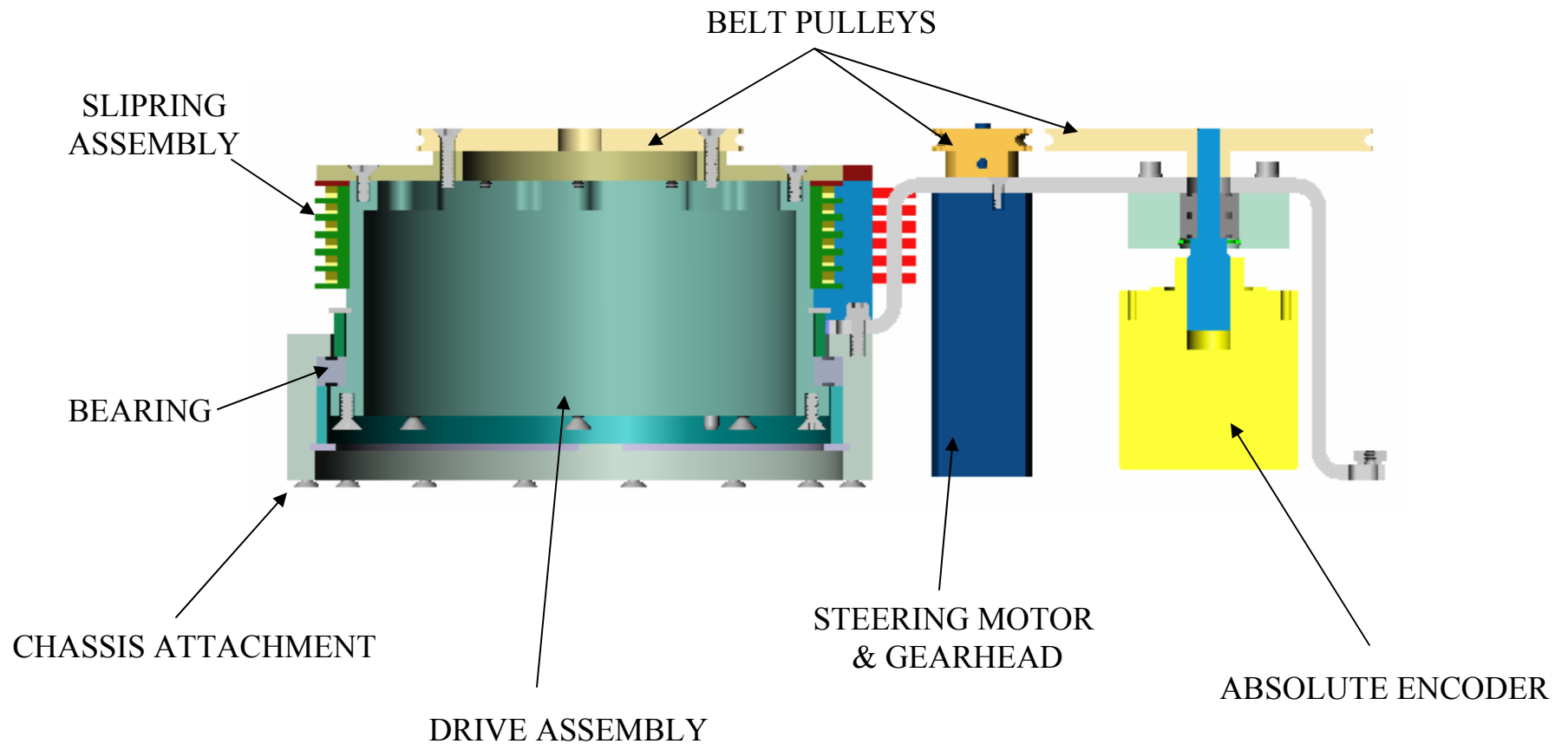
- Weight: approx. 40 lb.
- Height: 3.75 inches
- Footprint: 25" X 32"
- Velocity: 2.5 ft/sec
- Power Source: Battery
- Number of Wheels: 3
- Number of Processors: 8
- Environmental Sensing: Sonar, IR, Laser
- Position Sensing: dGPS, FOG
- Vehicle Runtime: 1 Hour
- Number of Battery Packs: 2 (12 V and 24 V)
- Ground Clearance: 0.5 inches

Steering Characteristics

- 24 Volt Maxon 110125 Motor
- 98:1 Gear Reduction
- Integrated 6 Contact Custom Slipring Assembly
- Steering Rate of 1 rev/sec max.
- Overall Weight = 3.24 lb.
- Computer Optics 10 bit Absolute Encoder

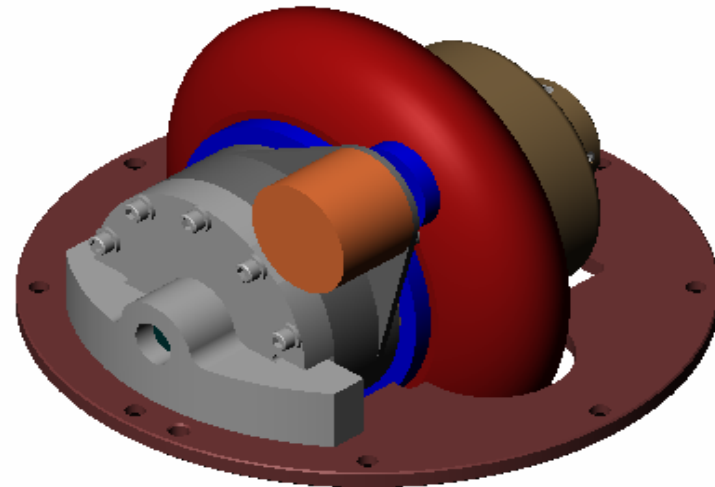


ODIS Steering Layout



Drive Characteristics

- QT 1221A 17 Volt
Kollmorgen Frameless
Torquer Motor
- 43:1 Micro-Mo Gearbox
- 2.6 Feet/Second Top Speed
- 25 Pounds Max Drive
Force Per Wheel
- 80 mm Wheel Diameter
- CUI Stack Incremental
Encoder



Power for Unmanned Systems

- Power for UxVs is one of today's limiting issues.
- Battery-based systems
 - Lead-acid
 - Nickel-Metal Hydride
 - Lithium-Ion
 - Silver-Zinc
- Combustion-engines
 - Gasoline
 - Diesel
- Fuel cells
- Novel: wind/water



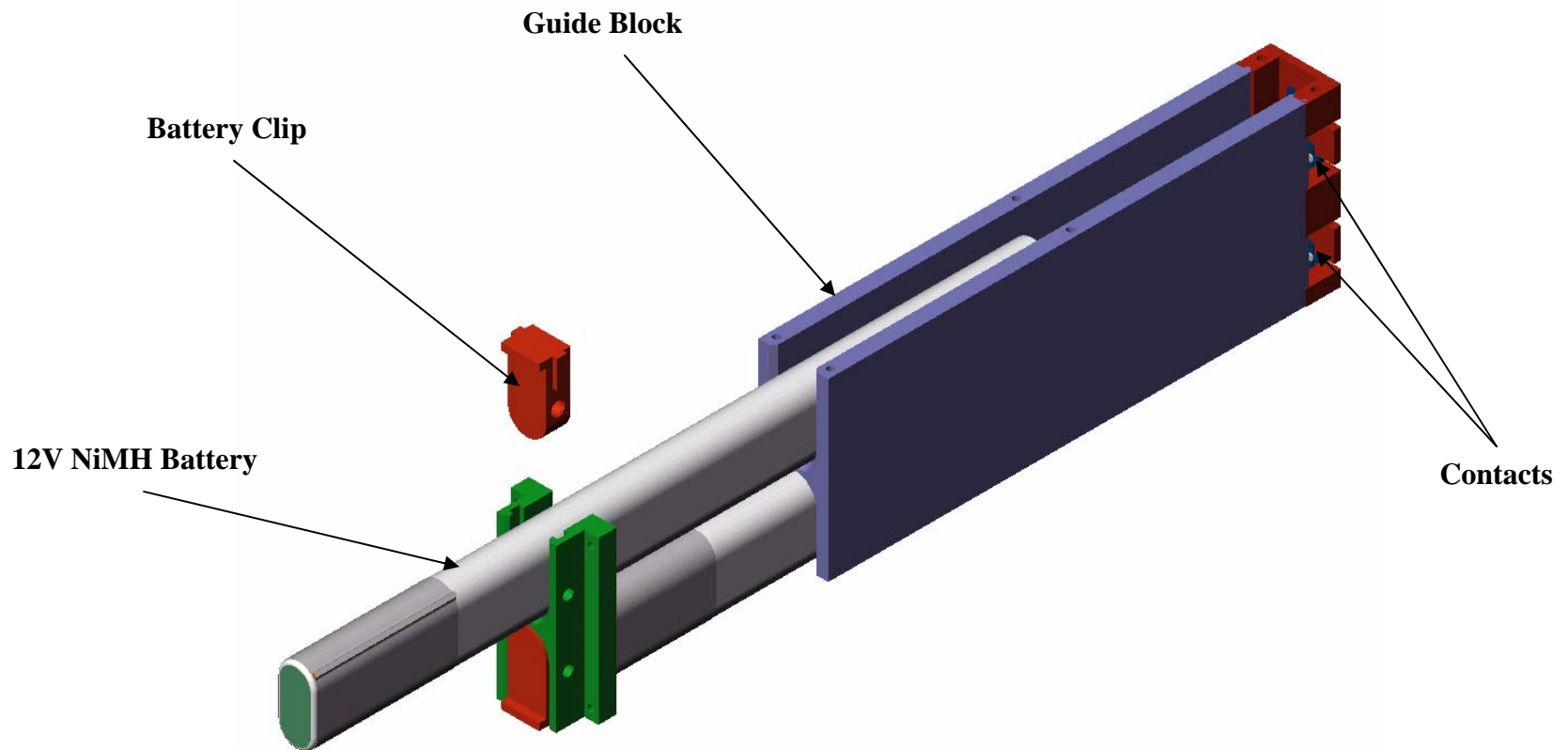
Station-keeping
sailboat



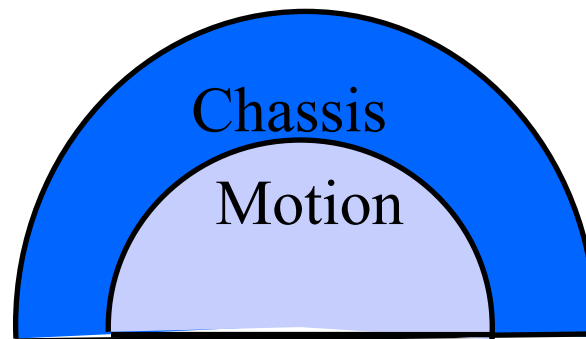
Fuel-cell powered ODIS developed
by Kuchera Defense Systems



ODIS Battery Assembly

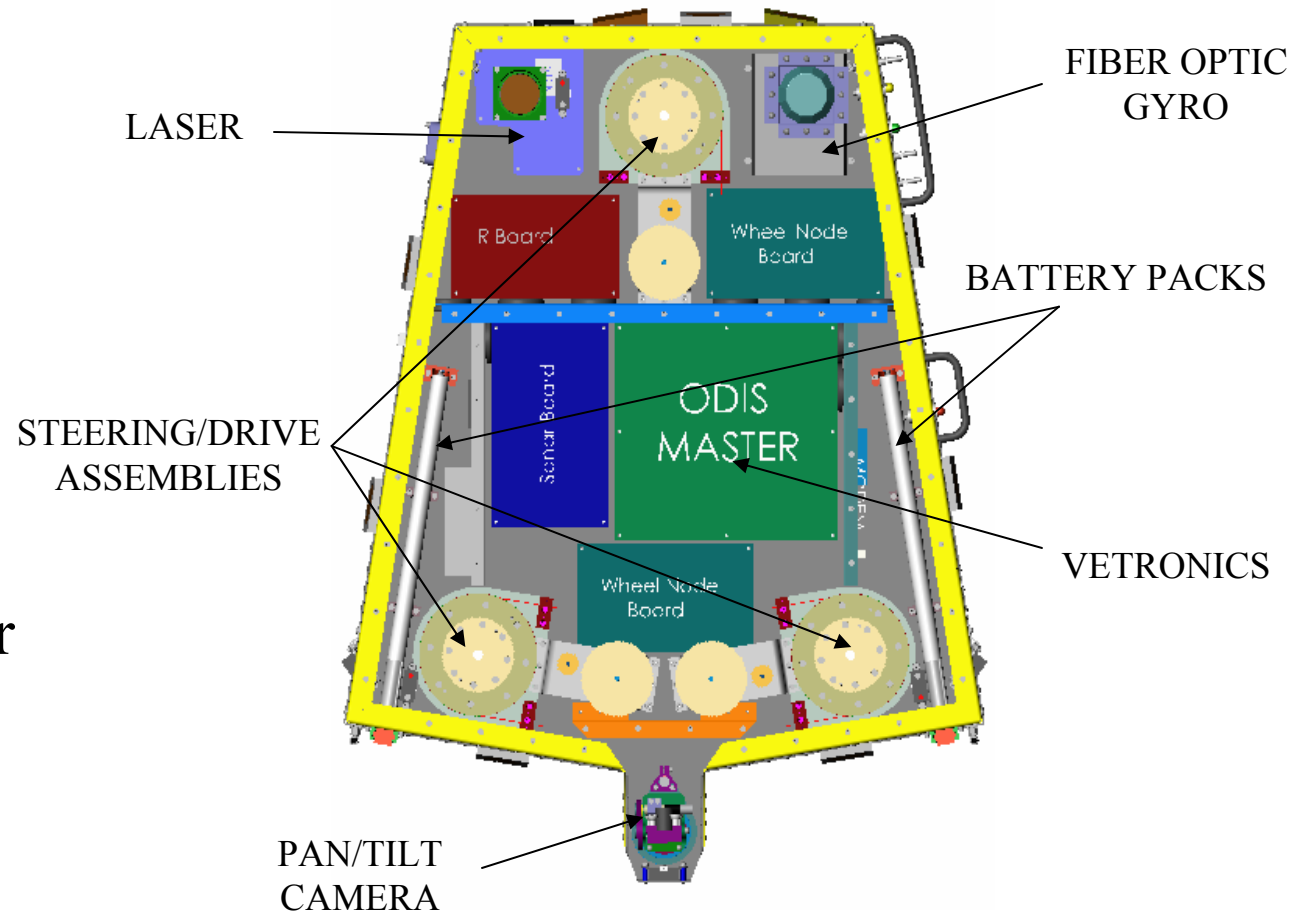


UGV Technology

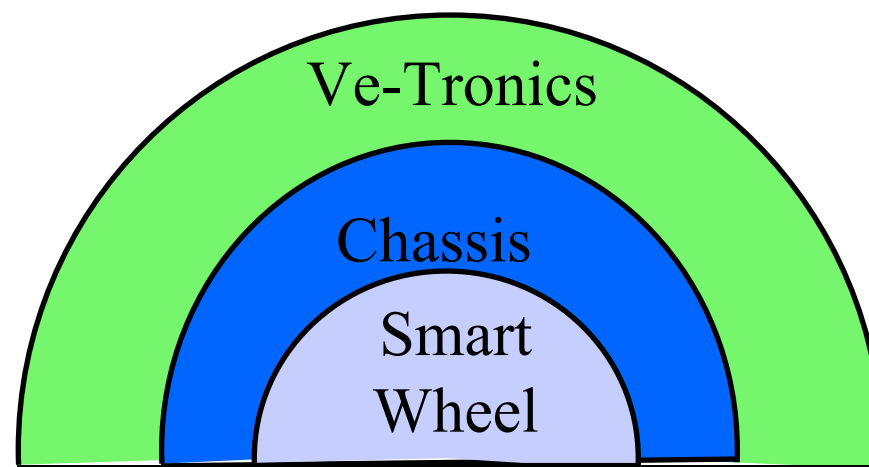


ODIS Chassis Layout

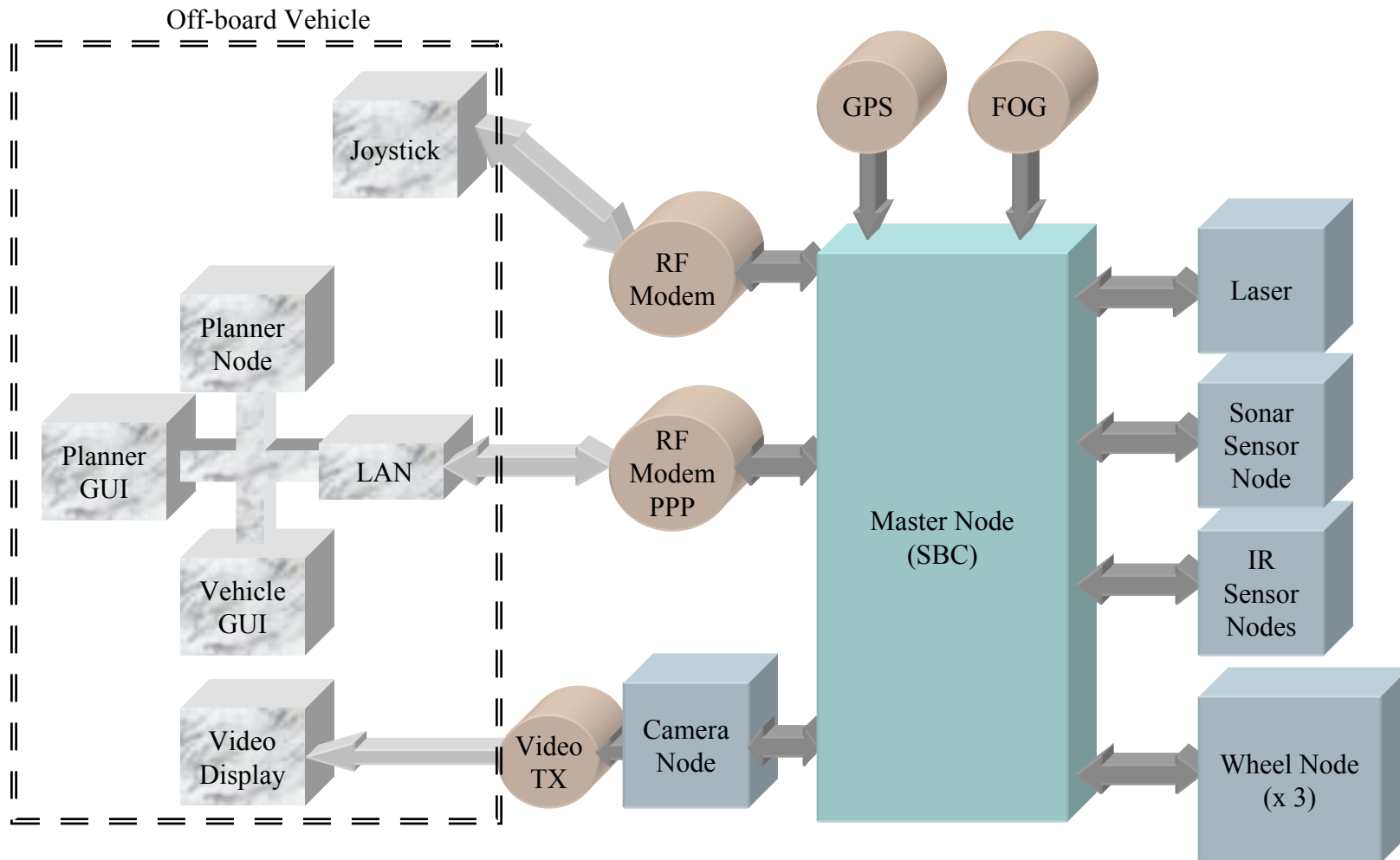
- 1/16” Aluminum Panels
- Glued & Riveted on Joints
- Interior Shear Panels



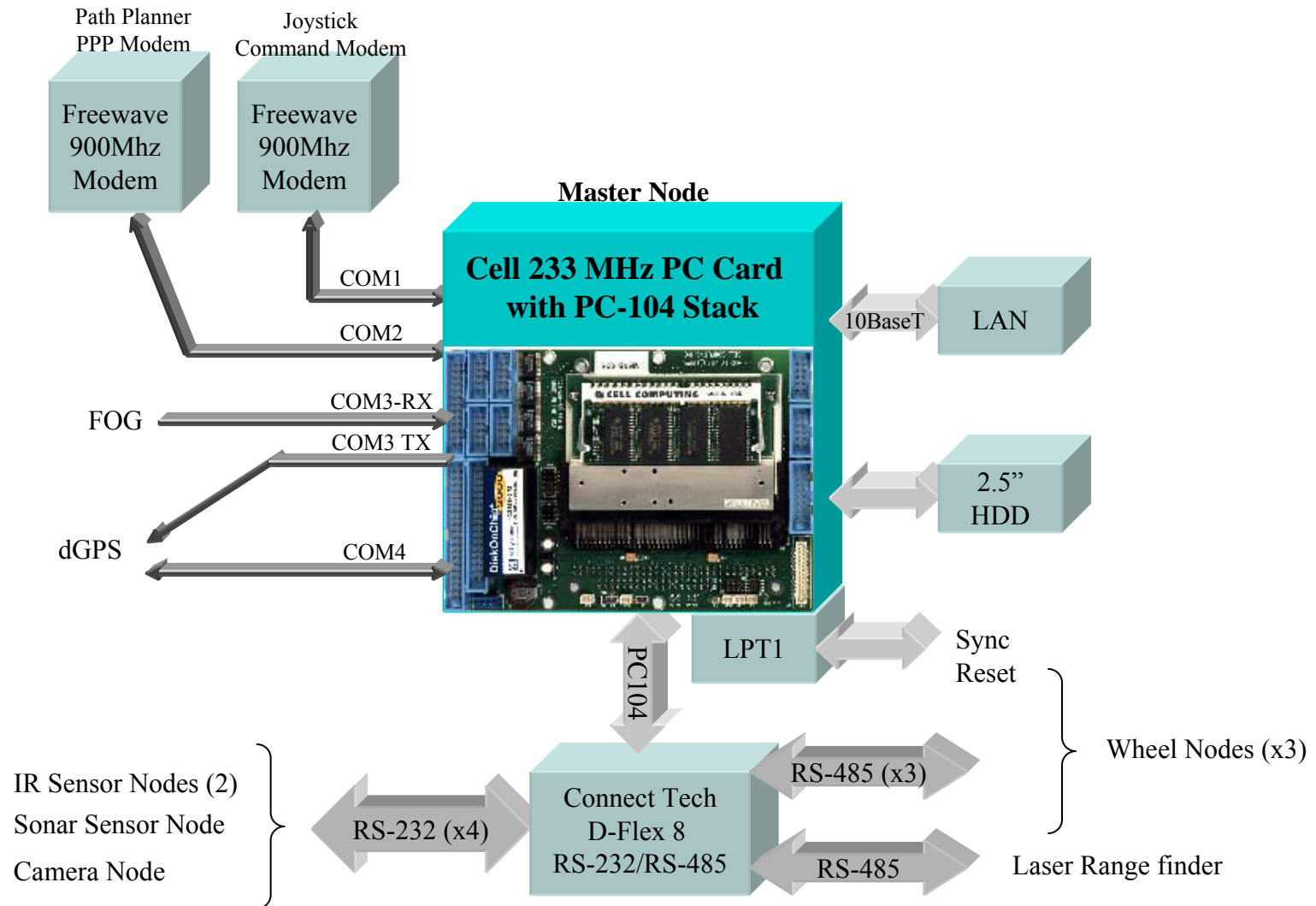
UGV Technology



Vetronics Block Diagram

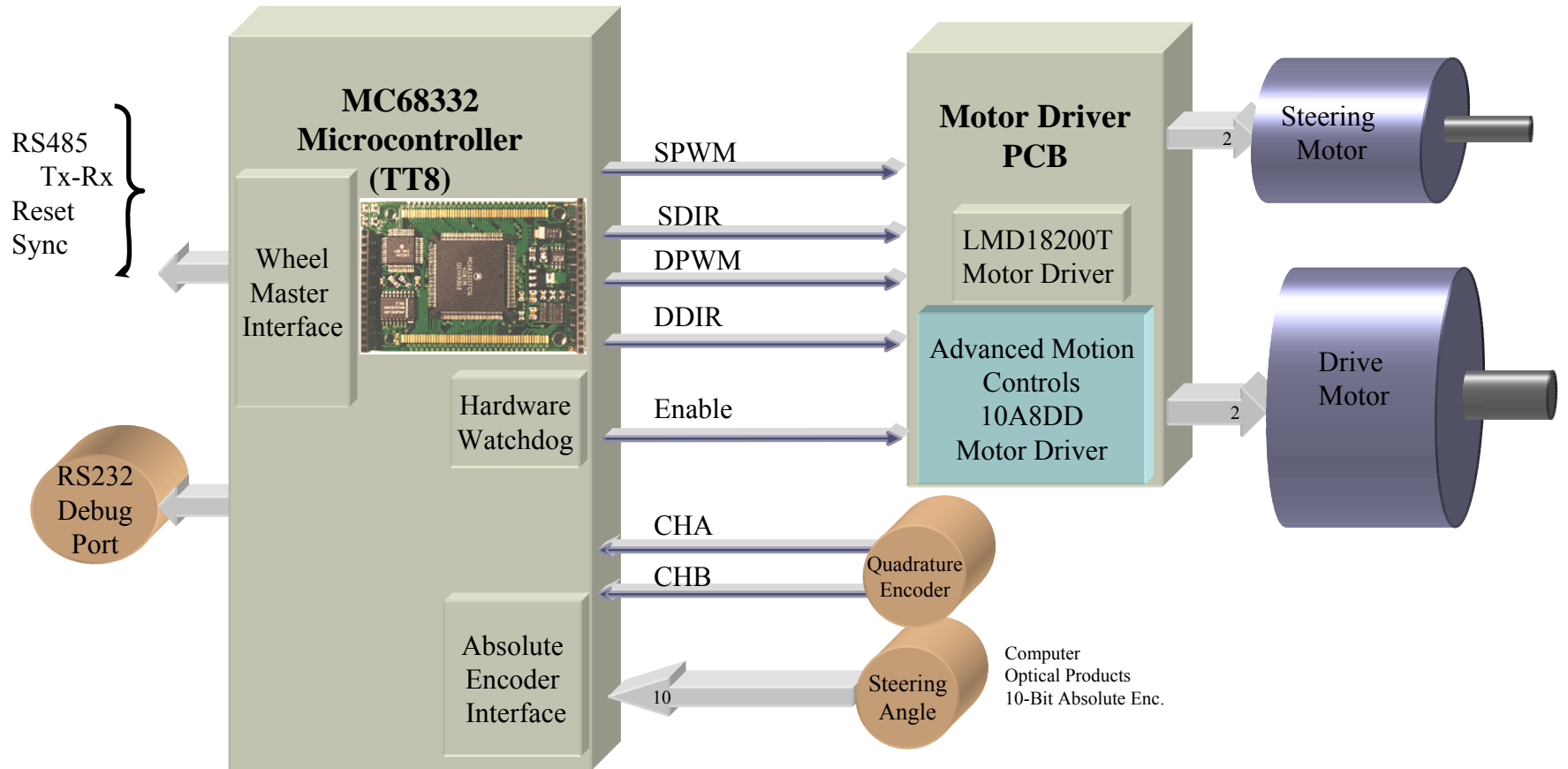


Master Node



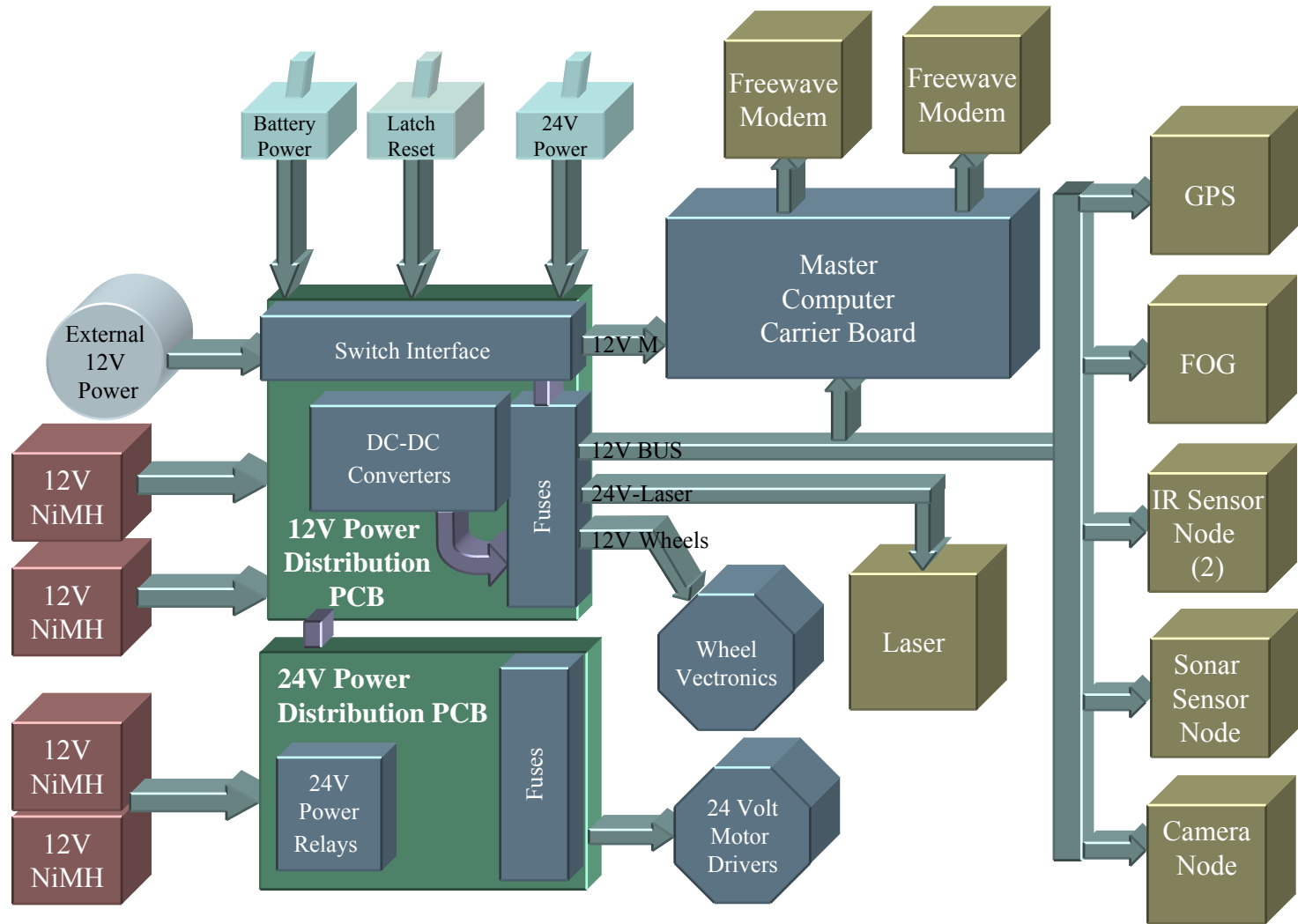
Master Node Subsystem

Wheel Node Block Diagram



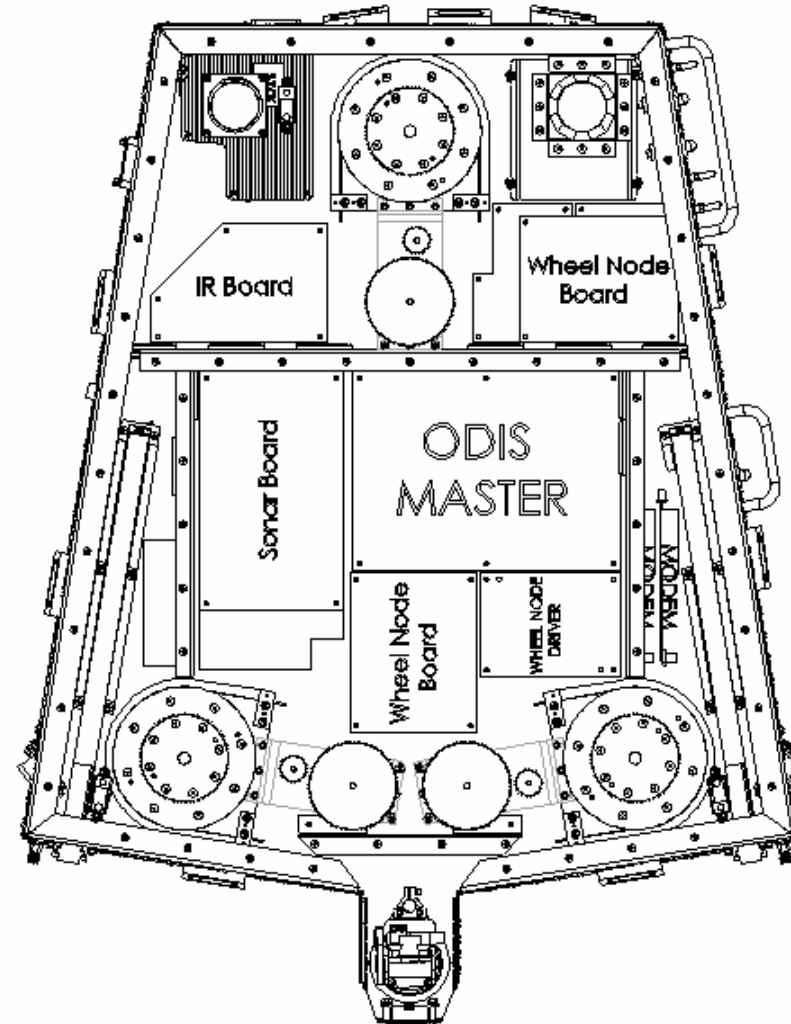
Wheel Node Subsystem

Vehicle Power System

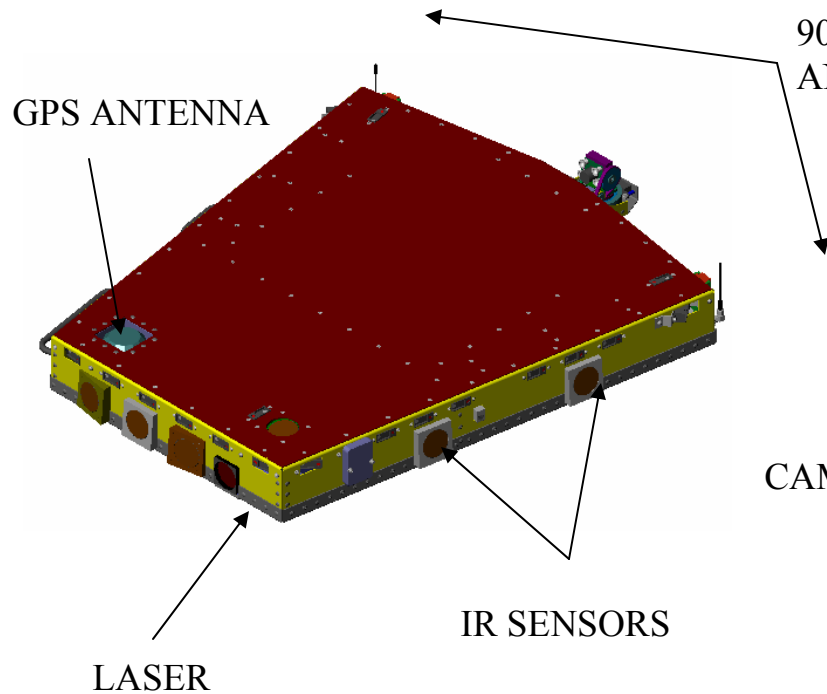


Vetronics Board Layout

- Master Node
 - Navigation Sensors
- Communications
 - Modem (Off vehicle)
 - Modem (Off Vehicle PPP)
 - RS-232/485 Onboard
- Sensor Node
 - Laser Rangefinder
 - IR Sensor Nodes
 - Sonar Sensor Node
 - Camera Node
- Wheel Node
- Power system

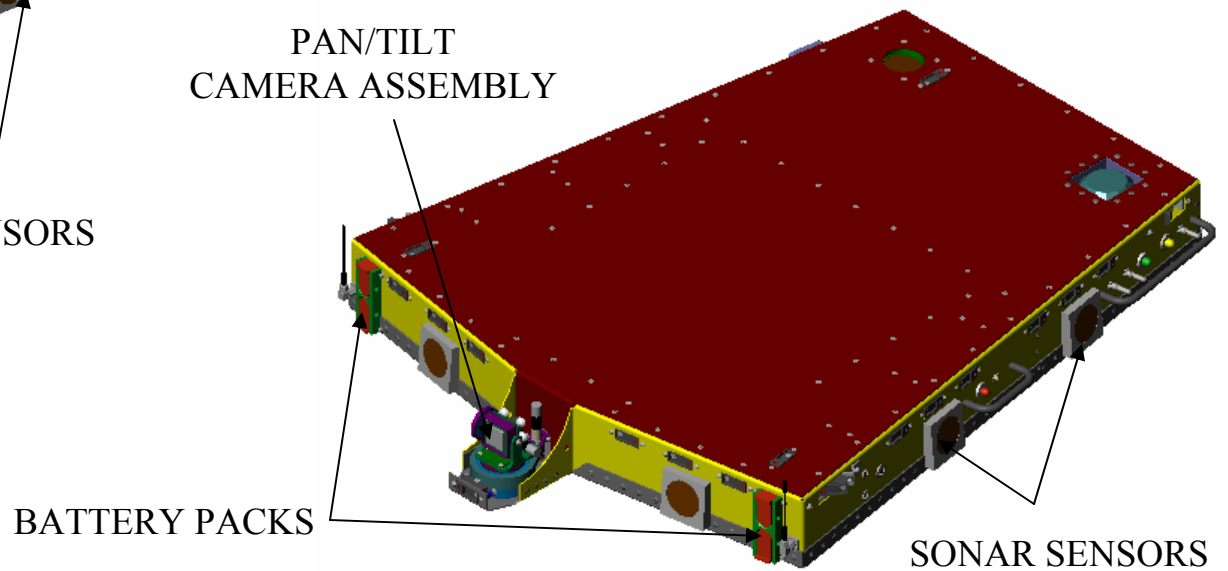


ODIS Weight Budget



- Chassis = 11.28 lbs
- Vetrronics = 9.53 lbs
- Drive/Steering = 14.11 lbs
- Batteries = 5.80 lbs

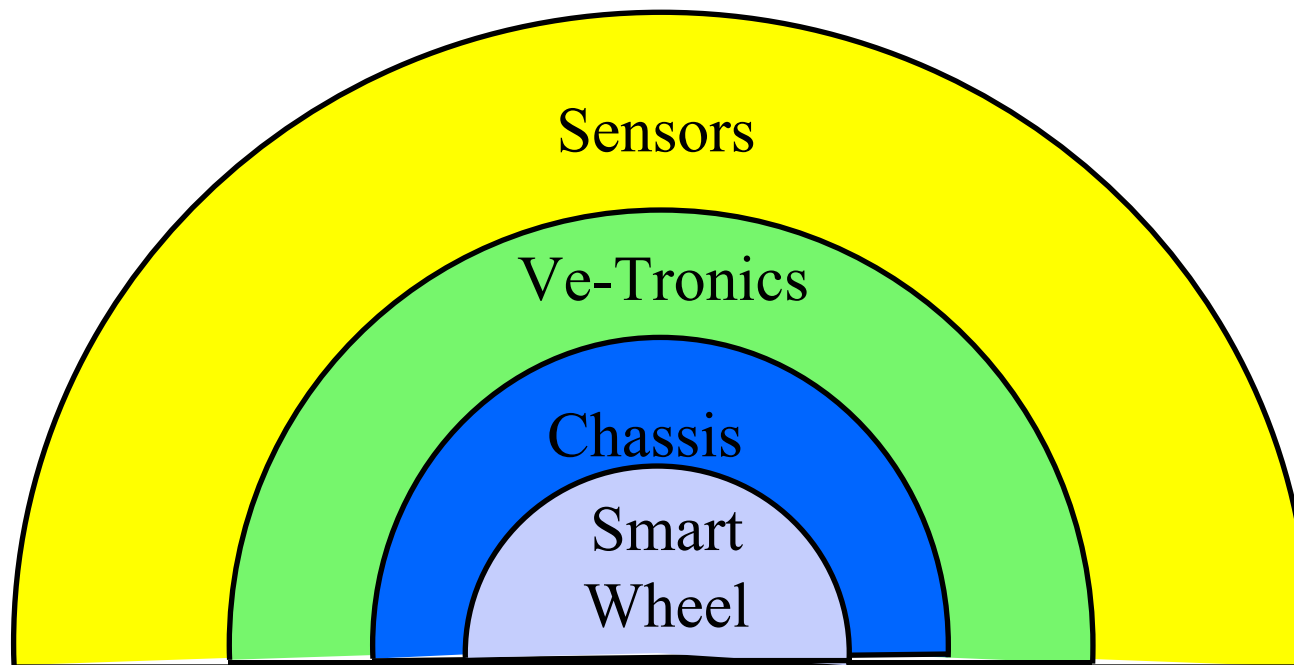
- ODIS = 40.72 lbs



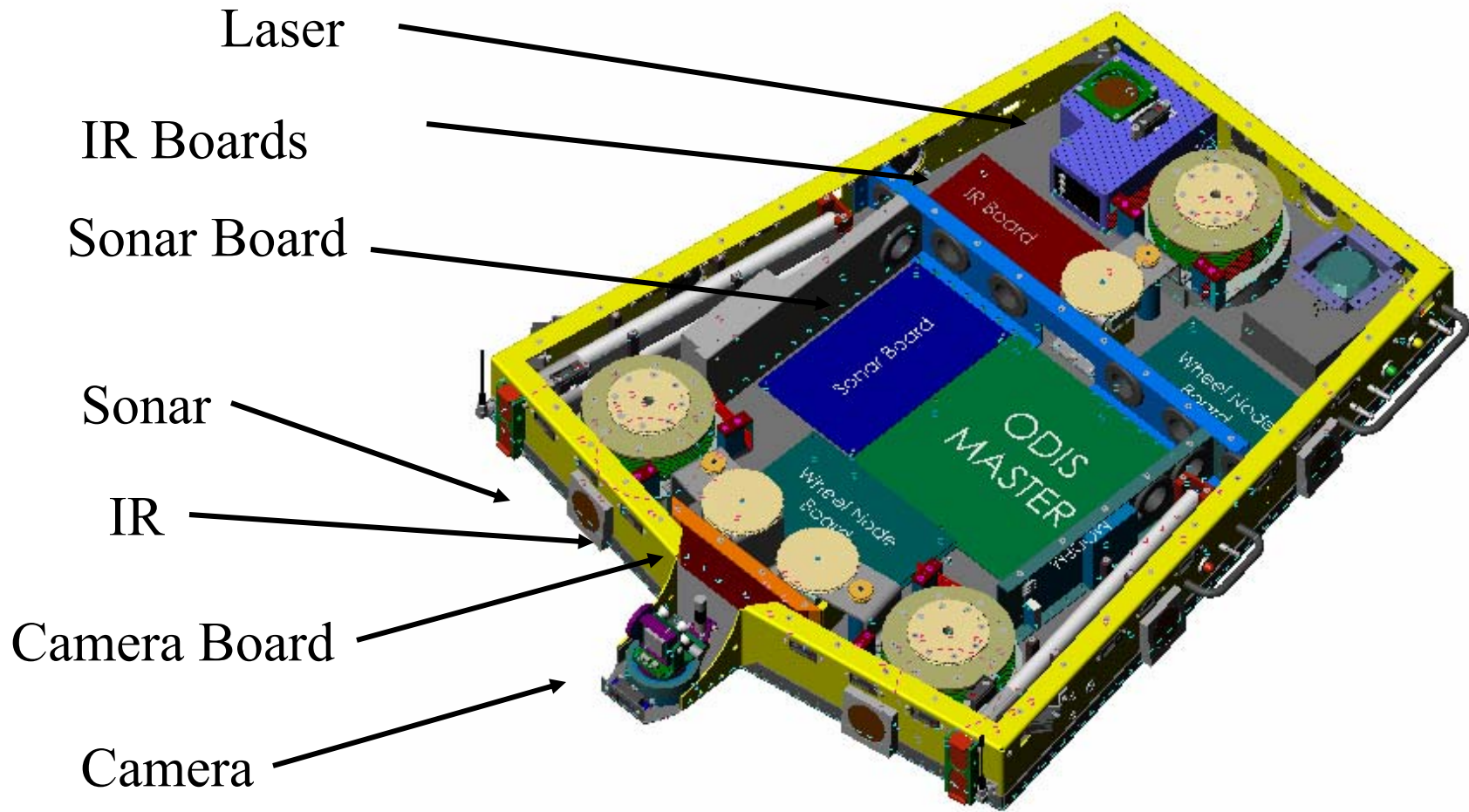
ODIS Vetronics System



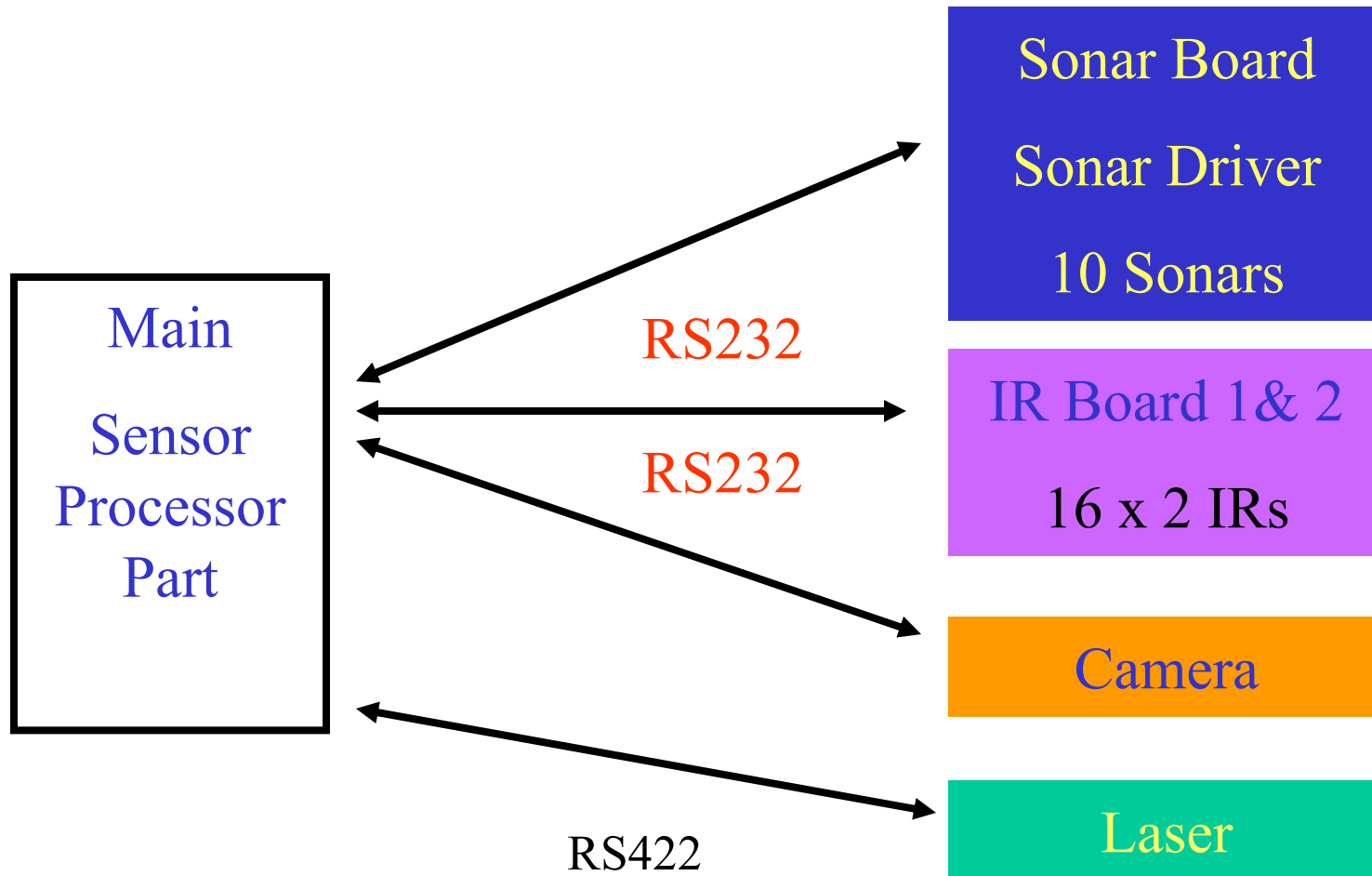
UGV Technology



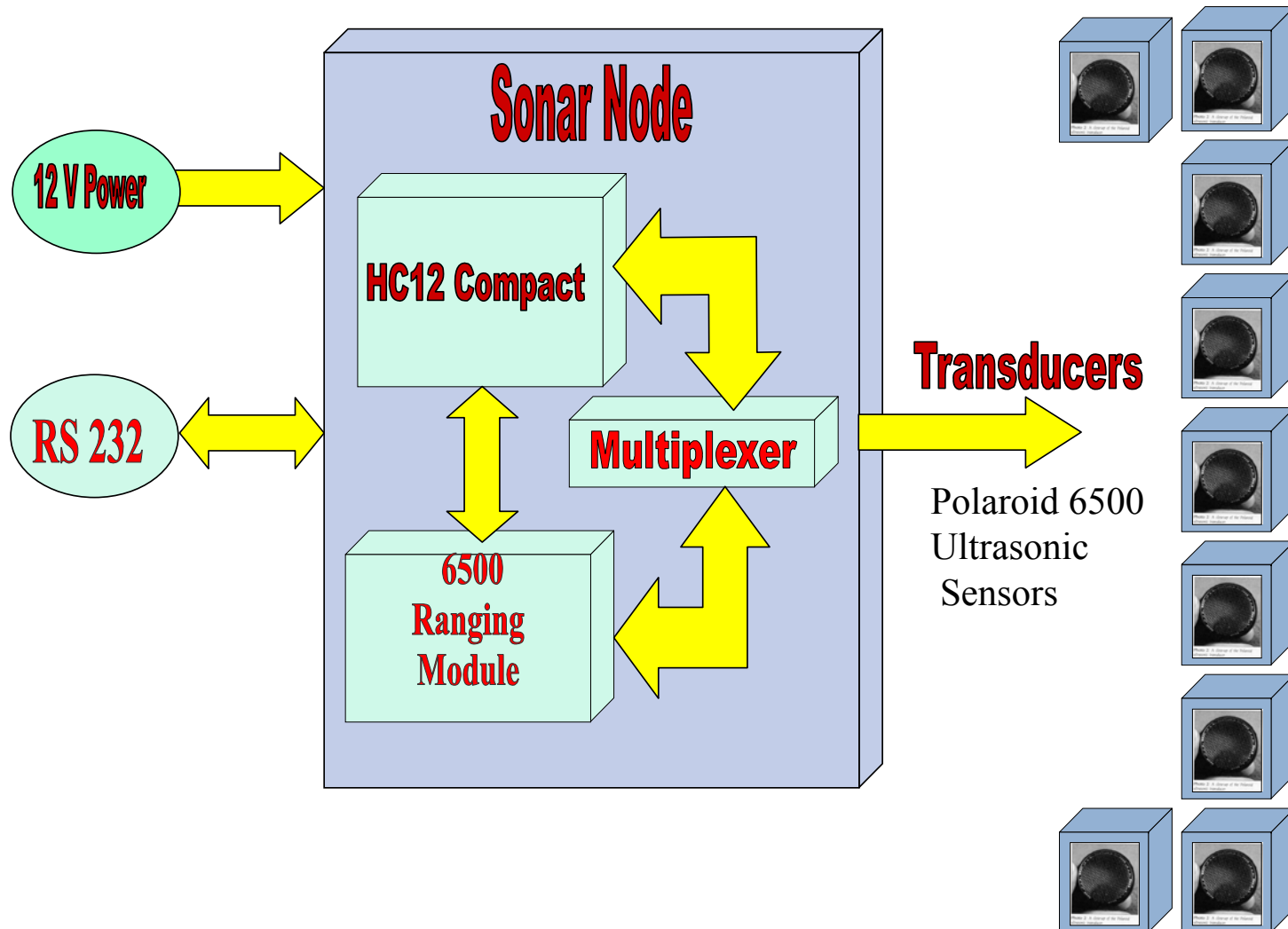
ODIS Sensor Suite



Sensor Drivers and Sensor Processor



Sonar Node Block Diagram

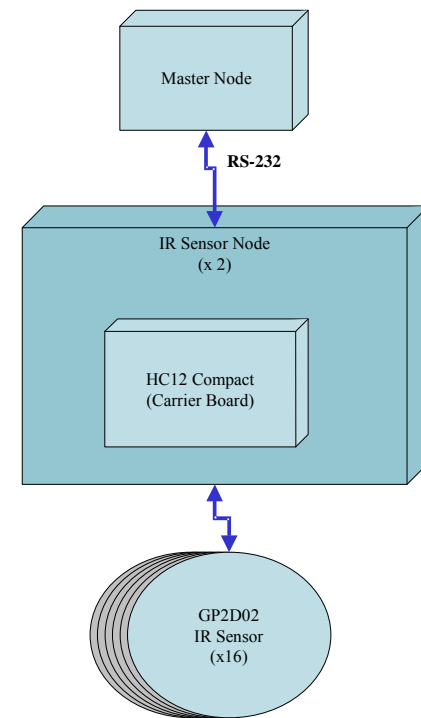


Sonar Summary

- Sonar was installed at $(1.313+0.5)$ " from the ground.
- Elevation angle: 5 degree to avoid detecting ground.
- Range :
 - 16" ~ 15 feet,
 - 3" ~ 16" (Target appearance only)
- Resolution < 1 "
- Error
 - Error in near range : $(1.0$ " + actual distance)
 - Error in far range is ± 1 "
- Beam angle is about 20 degree, ($w=6$ " at 16" distance.)

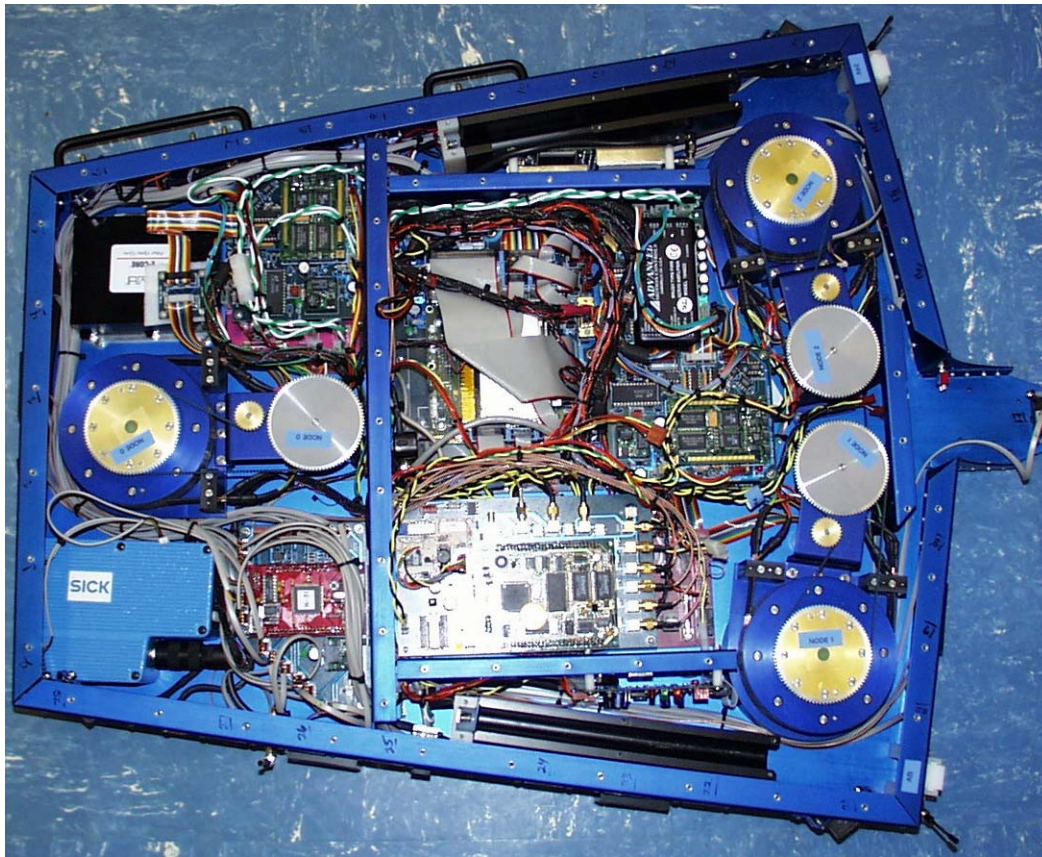
IR Sensor Summary

- Maximum range: 14 inch, (43.18Cm)
- Minimum Range:3 inch (7.62 Cm)
- Resolution < 1 Cm.
- Error range:
 - 3 ~ 13 inch distance: +/- 0.5 inch
 - 14 ~ 15 inch distance: +/- 1 inch
- IR beam width is very narrow < 0.3”
- 32 individual look up tables
- Works under 17000 lx



Sharp GP2G02
IR Sensors

SICK Laser DME3000

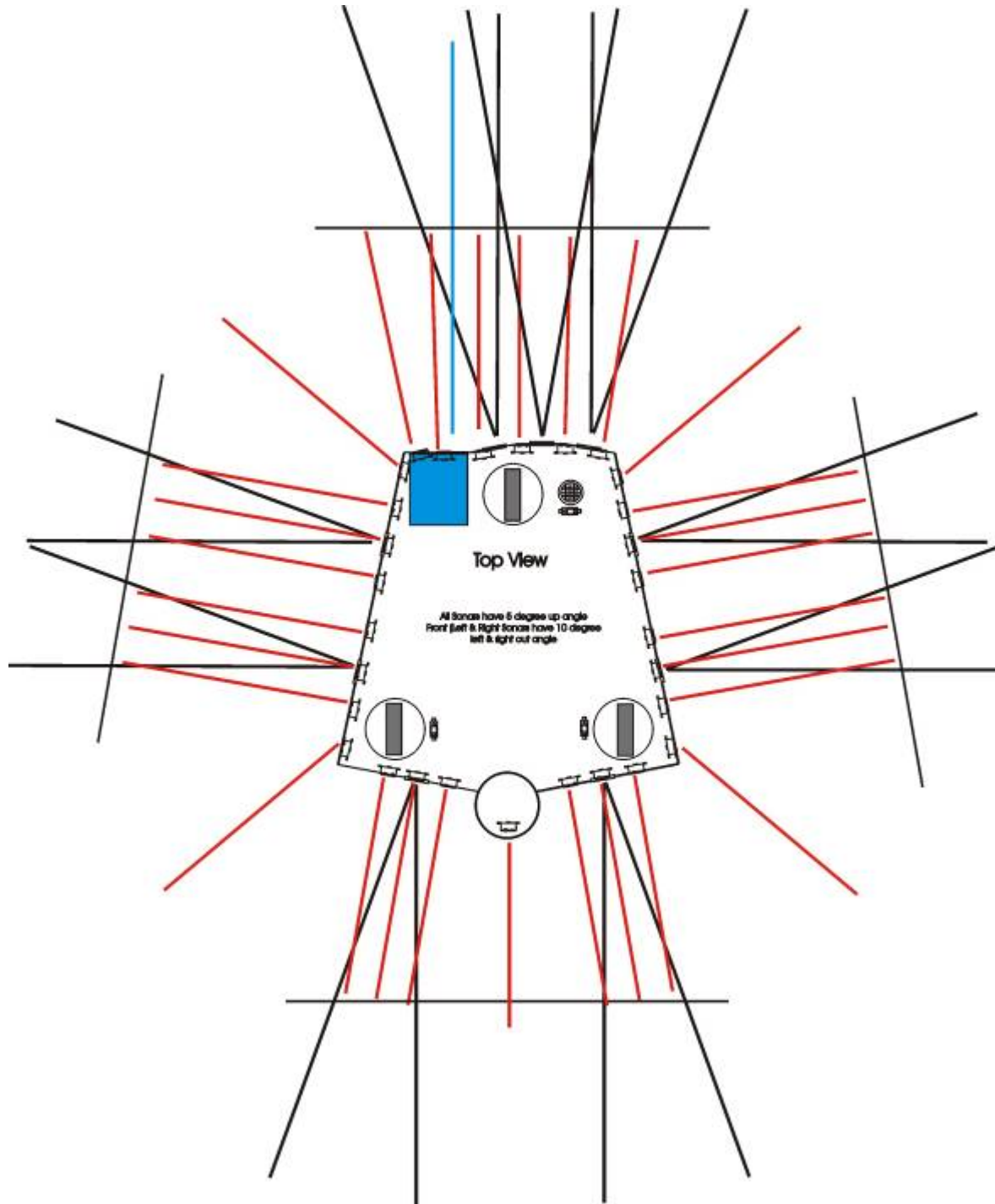


SICK Laser DME3000

- Dimensions: (137.5 x 105 x 53.7mm), 980g
- Depending on reflectivity of an object.
- Maximum Range for Target Acquisition
 - White Paper: 8 meters (maximum range in Spec.)
 - Average Vehicle Tire: 2.5 meter
 - Worst-Case Tire: 1.8 meters**
- Distance error < 1 cm
- Interface to Main board : RS422

B
E
A
M

P
A
T
T
E
R
N

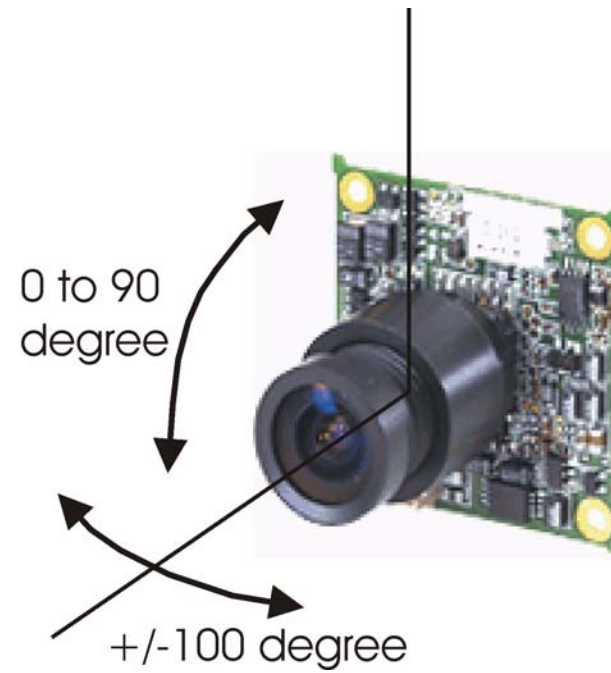
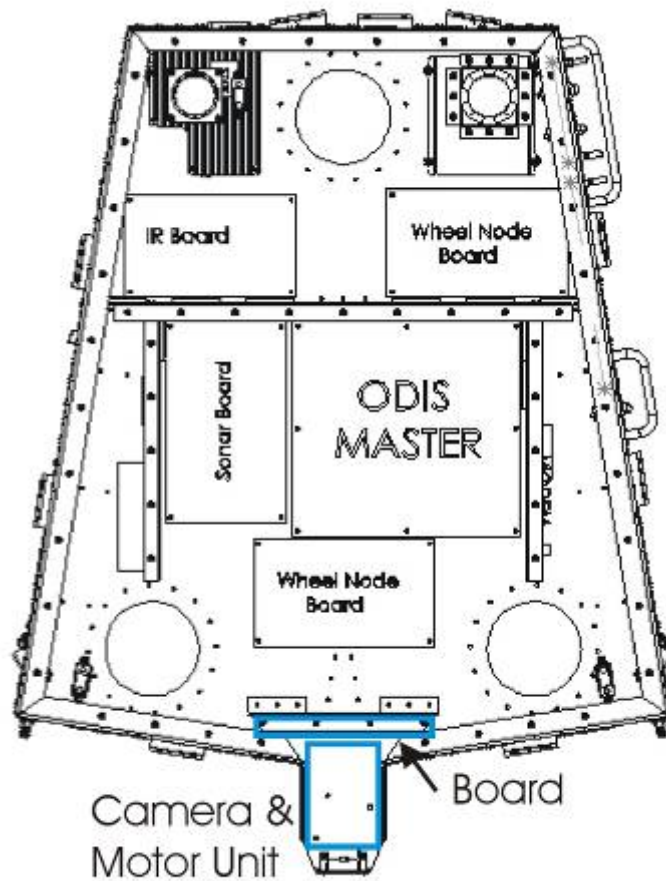


|
IR

|
Sonar

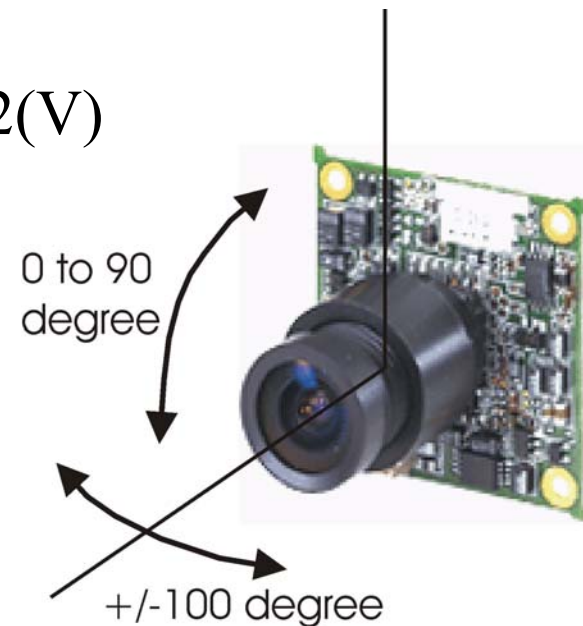
|
Laser

Camera GANZ CM3000



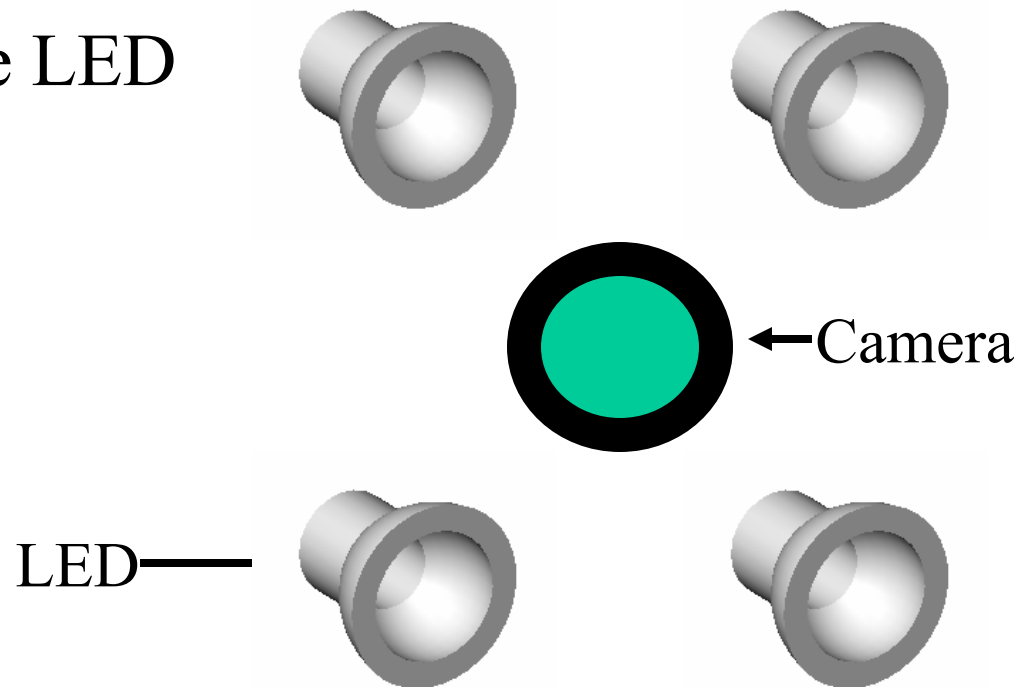
Camera GANZ CM3000

- Board Size: 1.25" x 1.25"
- Lens: FL=2.9mm , 330 H-line.
 - TV System: NTSC
 - Image Format: IT format 1/4" CC
 - Effective Picture Element 510(H) x 492(V)
 - Resolution 380 (H) / 350(V) TV Lines
 - 69 degree F.O.V.
 - 12V, 120mA
- Rotation Angle:
 - +/-100 degree
 - 0 to 90 degree elevation angle



Camera Lighting

- High Output White LED
- 4 V DC
- 50 mA
- 5.6 LUX Output
- Array of 4

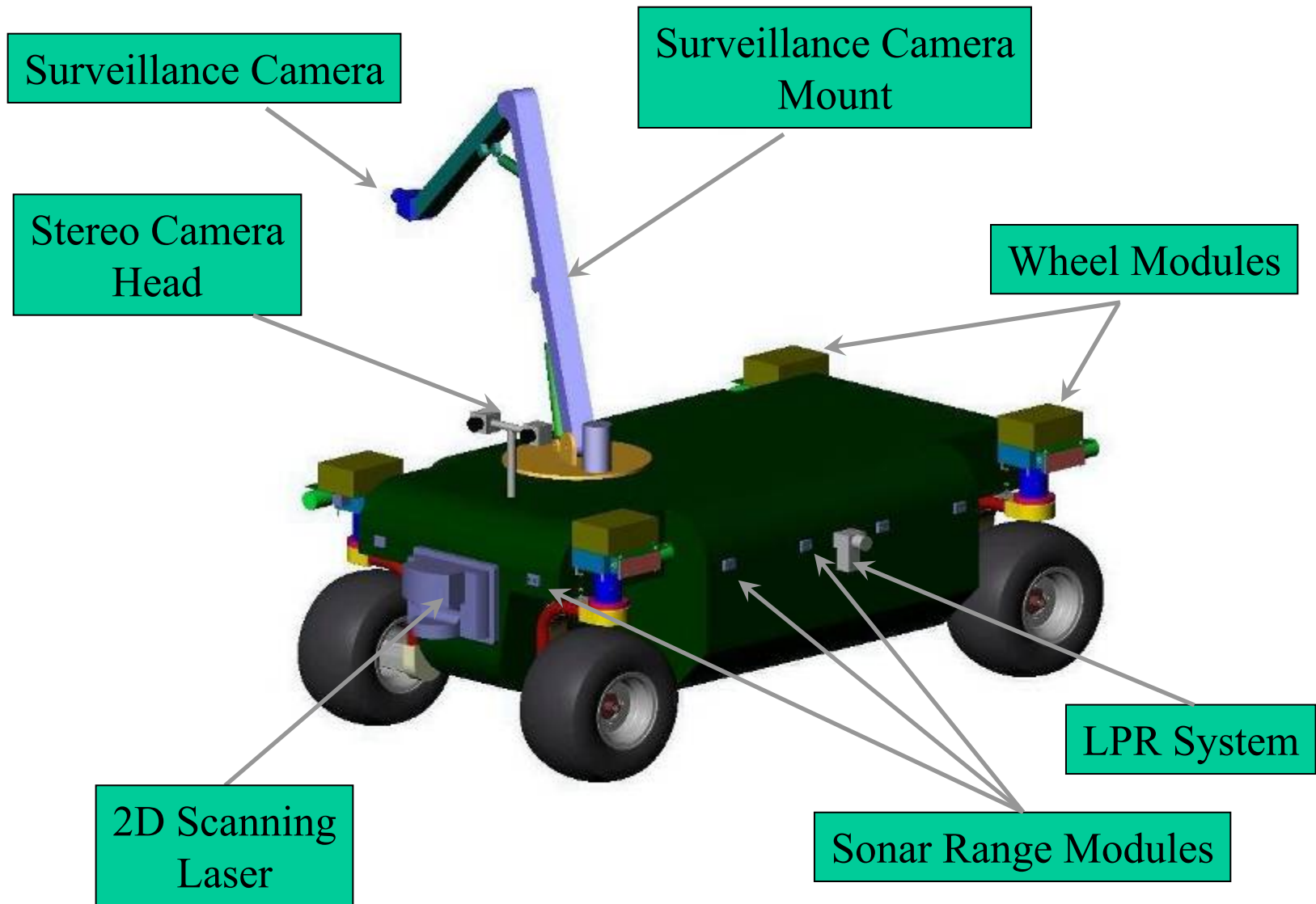


Camera: Video Tx & Rx

- Transceiver: CVT-M By Coherent
 - Size: (3.9 x 4.7 x 1.4 cm)
 - NTSC or PAL
 - Range: 1000 ft (305m) line of sight
 - 902-928 MHz in 10 user selectable channels.
 - Antenna: 50 Ohm subminiature MMCX type
- Receiver: CVR-1000 By Coherent
 - Size: (98 x 155 x 30mm)
 - NTSC or PAL
 - Range: 1000 ft (305m) line of sight
 - 902-928 MHz in 10 user selectable channels.
 - Up to 3 channels in same area
 - Antenna: 50 Ohm BNC.



T4 Sensors - Artist's Rendition



Sensors and Safety: Automated Tractor Project Example





Autonomous
Solutions Inc.

Safety Scenarios

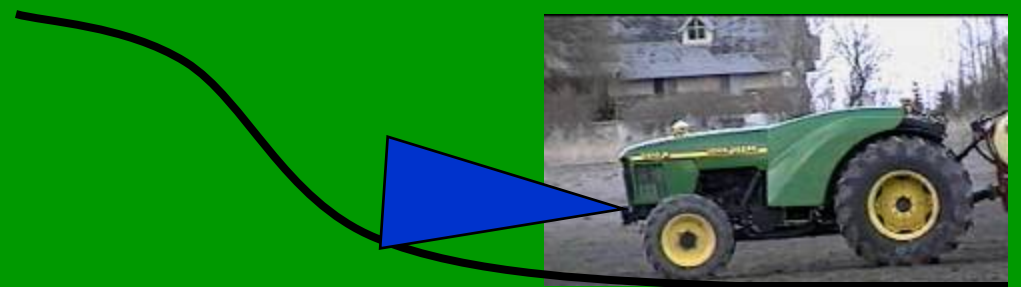


- We need to halt the vehicle if:
 - Tractor leaves field boundary or deviates from path
 - Unavoidable obstacle within given threshold
 - Communication disrupted or lost
 - d-GPS dropout corrupts position information
 - Computer failures occur
 - Emergency stop button
 - Vehicle halt computer command
 - Mission complete
- Some safeguards include
 - Sensor suite for detecting vehicle path obstructions
 - Redundant radio link to protect against wireless communication dropout or corruption.
 - Use of odometry to complement/supplement dGPS



Autonomous
Solutions Inc.

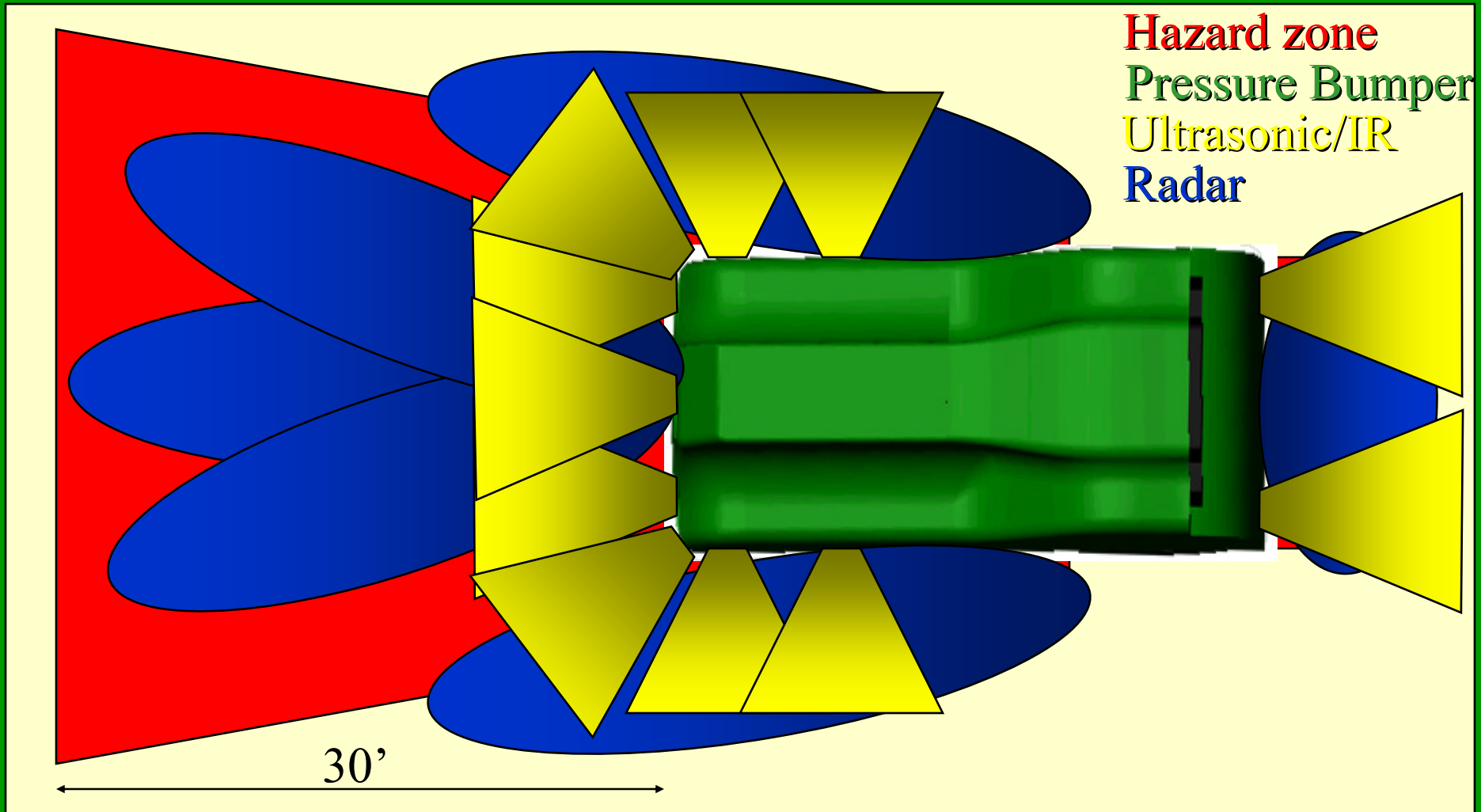
Awareness Issues





Autonomous
Solutions Inc.

3 Tiered Proximity Detection





Autonomous
Solutions Inc.

Operational Lessons



- 200+ hours in the McMullin Orchard
- 800 acres of cherries, peaches, apples, & pears
- Orchard owner/manager is very cooperative
 - Cost is the key to owner acceptance
 - Safety is important
 - No problem accepting the technology if it works
 - Very interested in EPA report generation
- Localization (where am I?) is a key issue



**Autonomous
Solutions Inc.**

Localization Issues



- **Poor Radio Communications**
 - **High power antennas**
 - **Lower Frequency**
- **Intermittent GPS**
 - **Dead-reckoning**
 - **Reactive positioning**
 - **Hole-following with range data**
 - **Row sensing with laser**

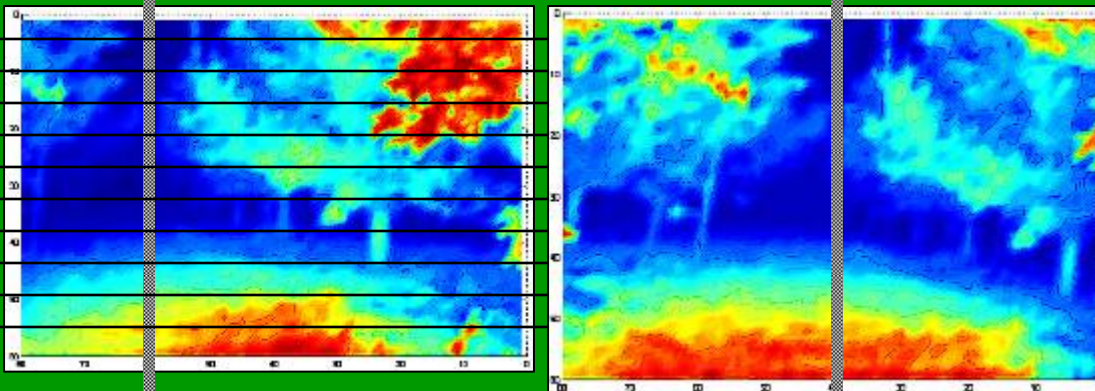


Infrared

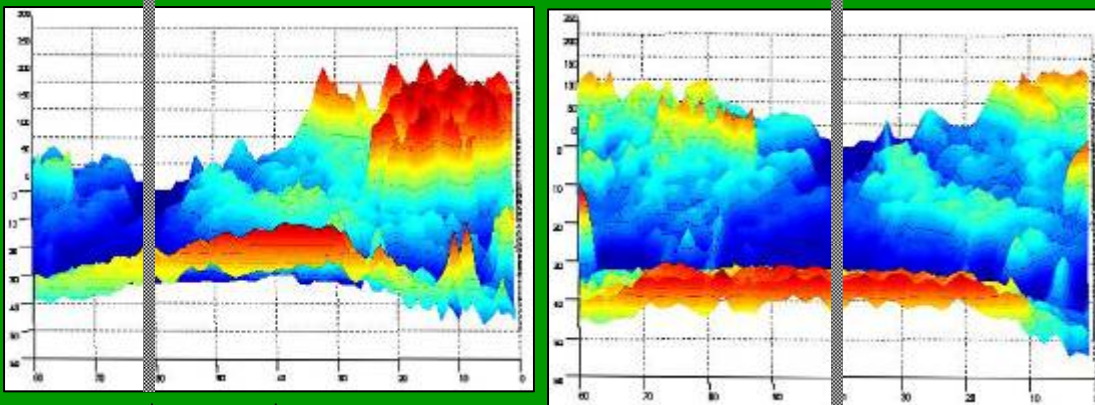


Steer to the hole
(minimize error)

Range
map



3D range
Map



Error

Pro:

- Sensor independent
- Fast
- Simple

Con:

- Environment dependent

Scope Creep and Design Evolution

- Inevitably, all projects experience either
 - Scope creep: customer keeps changing things and/or adding to the requirements
 - Product evolution
 - Both!
- In the case of the ODIS project,
 - Customer requested tele-op only version (ODIS-T)
 - Later, different customer request another full-up autonomous version to be used for semi-autonomous operation (ODIS-S)!

ODIS-T – A Tele-operated Robot

- Replaces traditional “mirror on a stick” at security checkpoints
- Joystick-driven; video/other sensor feedback to operator
- Ideal for stand-off inspection, surveillance, hazard detection



Stand-off is the main benefit



Mission Payloads for UxVs

- Different CONOPS will produce different mission payload requirements.
- ODIS-T Sensor Suites:
 - Visual – pan/tilt imaging camera
 - Passive & active thermal imaging
 - Chemical sniffers – i.e. nitrates, toxic industrial chemicals
 - Night vision sensors
 - Acoustic sensors
 - Radiation detectors – i.e. dirty bombs
 - Biological agents detection
 - MEMS technology – multiple threats
 - License plate recognition

Mission Packages - IR

IR Image – Warm Brake



IR Image – Recently Driven Vehicle



Some Mission Packages Actually Deployed



1. LCAD Chem “Sniffer”

2. Radiation Detector (not shown)

3. IR Thermal Imaging Camera
(recently driven vehicle)

- Continuous, real-time detection of CW Agents
- Enhanced IMS technology using a non-radioactive source
- Communication port for use with computer, ear piece or network systems
- Small and lightweight
- Audio and / or visual alarm
- 40 + hours on AA type batteries
- Data logging capabilities
- Detection of TIC'S (Toxic Industrial Compounds)



Mission Payloads for UxVs

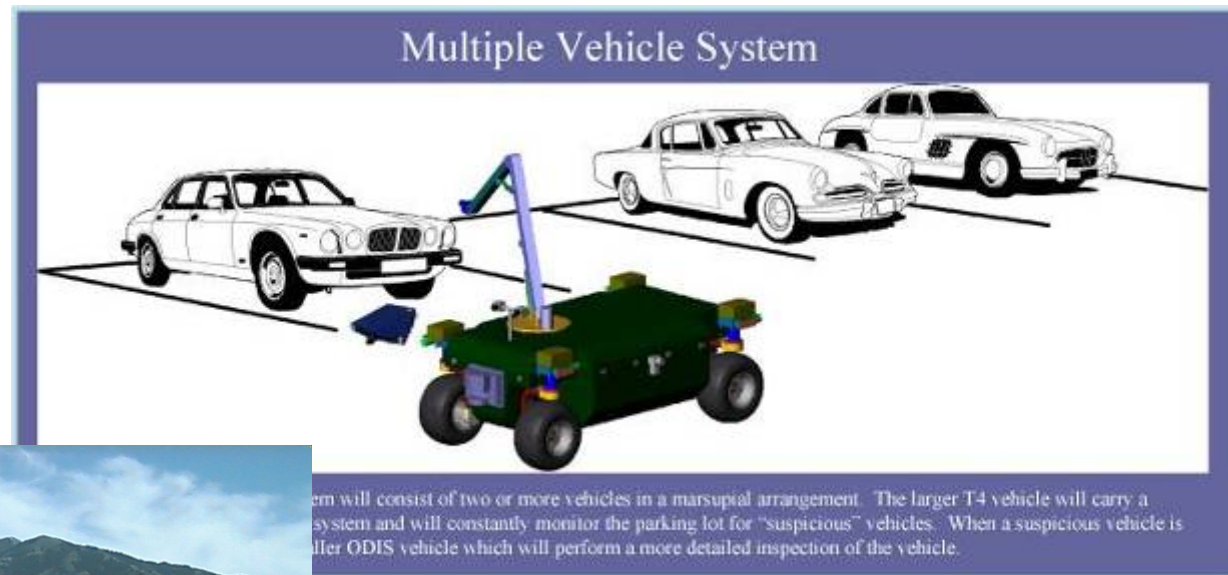
- Different CONOPS will produce different mission payload requirements
- ODIS-T Sensor Suites:
 - Visual – pan/tilt imaging camera
 - Passive & active thermal imaging
 - Chemical sniffers – i.e. nitrates, toxic industrial chemicals
 - Night vision sensors
 - Acoustic sensors
 - Radiation detectors – i.e. dirty bombs
 - Biological agents detection
 - MEMS technology – multiple threats
 - License plate recognition
- **Mission payload can be actuators as well as sensors**



Samsung Robot Sentry

Other UxV Concepts

- Coordinated and Marsupial Behaviors





T4 Parking Lot Surveillance Robot

- Omni-directional
- Hydraulically driven
- Gasoline Powered
- Designed to work in cooperation with ODIS



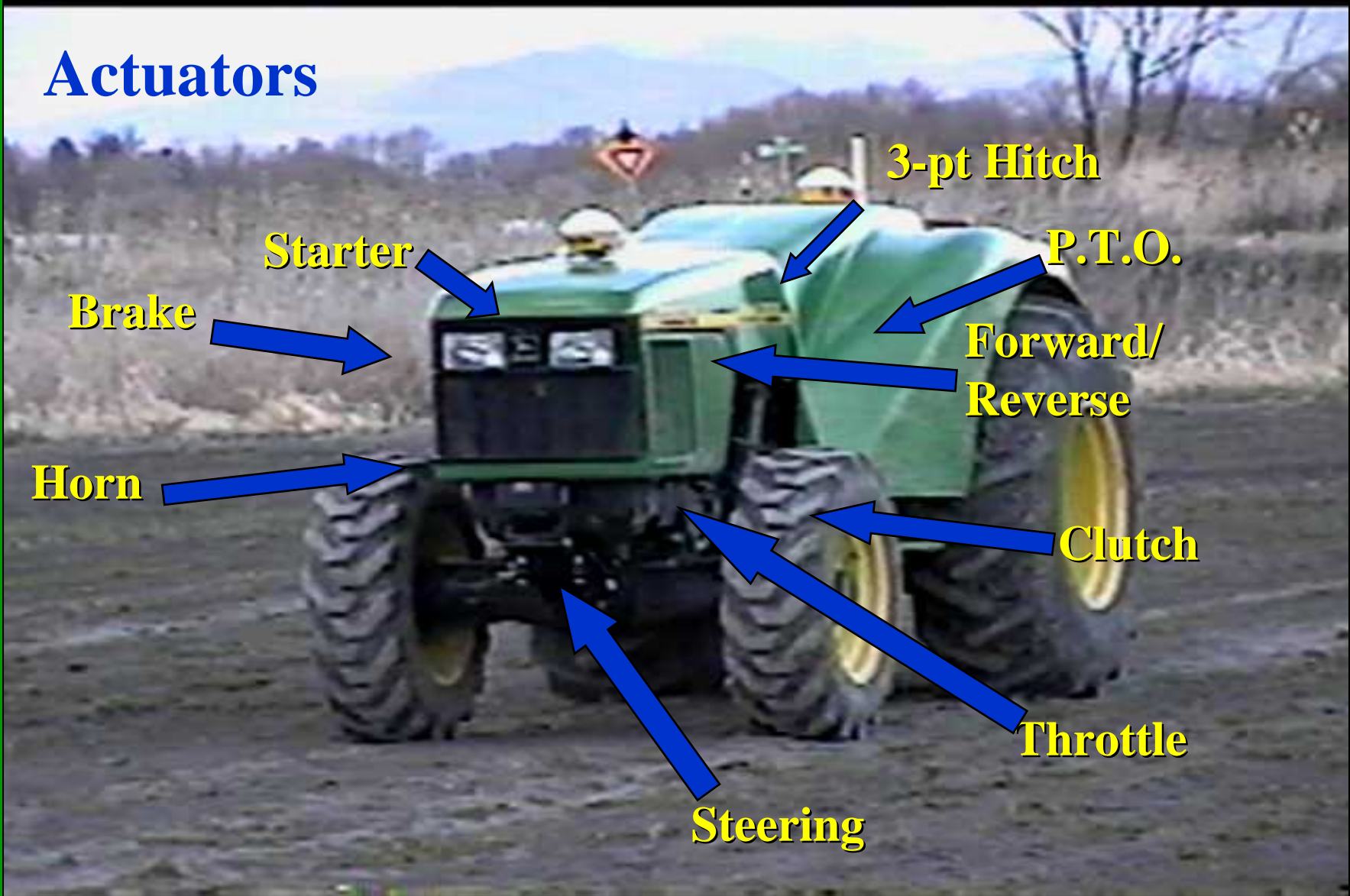
Retrofitting Existing Vehicles

- Larger automated vehicles (tractors, construction equipment) can be used by security and law enforcement personnel for
 - Fire-fighting
 - Road-block and debris clearing
 - Building breaching
 - Crowd control
 - Explosive ordinance disposal
- Typically such vehicles are retro-fitted using an “automation” kit
- This requires adding actuation as well as sensing

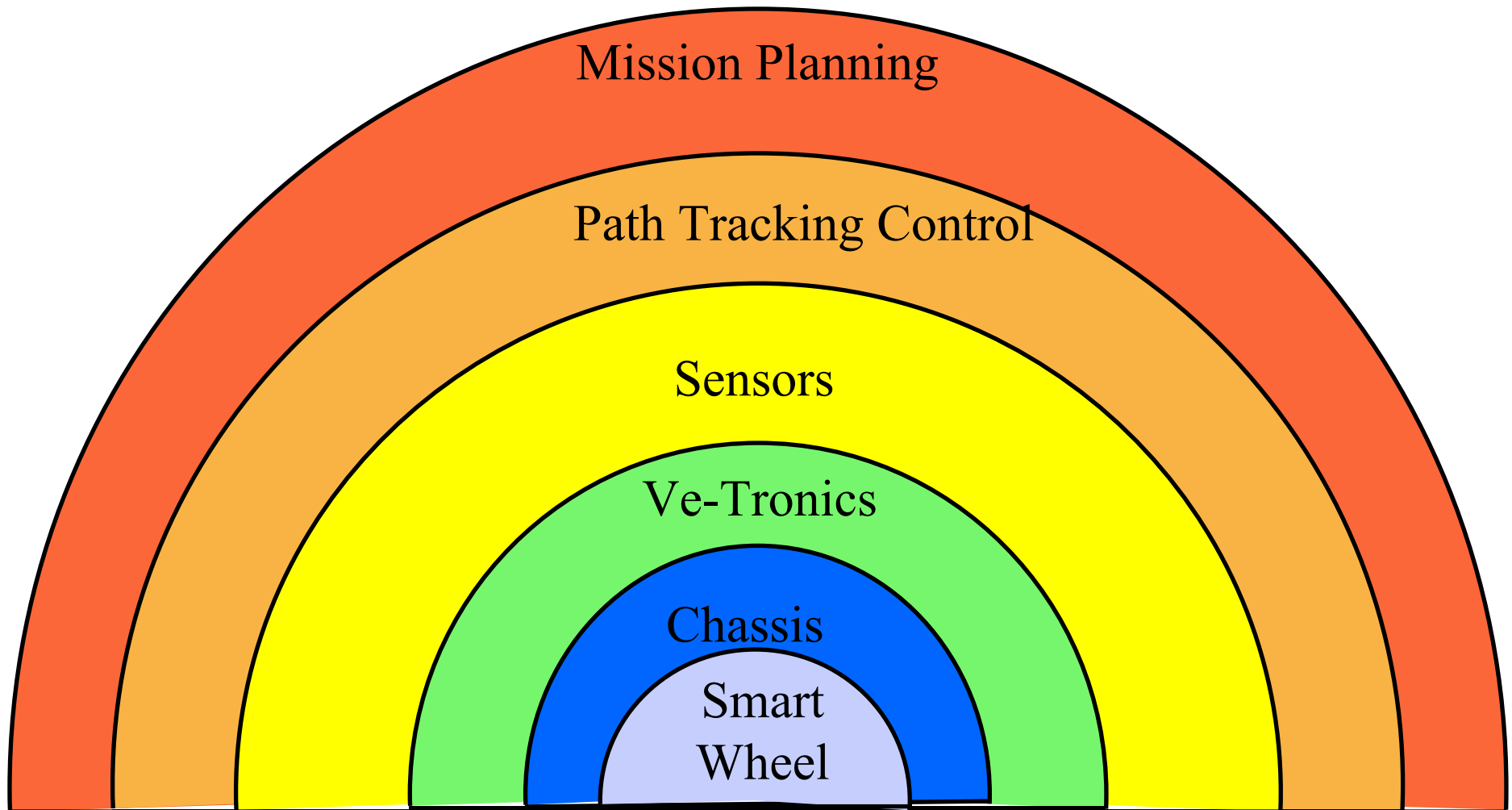


Automated Gator ATV
developed by Logan,
Utah-based Autonomous
Solutions, Inc.

Actuators



UGV Technology



Outline

- What is an Unmanned System?
- Unmanned system components
 - Motion and locomotion
 - Electro-mechanical
 - Sensors
 - Electronics and computational hardware
- **Unmanned system architectures**
 - **Multi-resolution approach**
 - **Software Architecture**
 - **Reaction, adaptation, and learning via high-level feedback**

Mission Planning and Control System

- Transforms a collection of smart wheels into a smart, mobile vehicle
- Smart mobility is achieved by coordinating and executing the action of multiple smart wheels:
- The Mission Planning and Control System must be organized in a suitable architecture.
- Here we will emphasize a multi-resolution system to implement a “task decomposition” approach

Architectures for Unmanned Systems

- A large number of architecture and strategies have been proposed for intelligent behavior generation for UxVs:
 - Subsumption (Brooks).
 - Input-output relations to solve small problem; prioritization of actuation

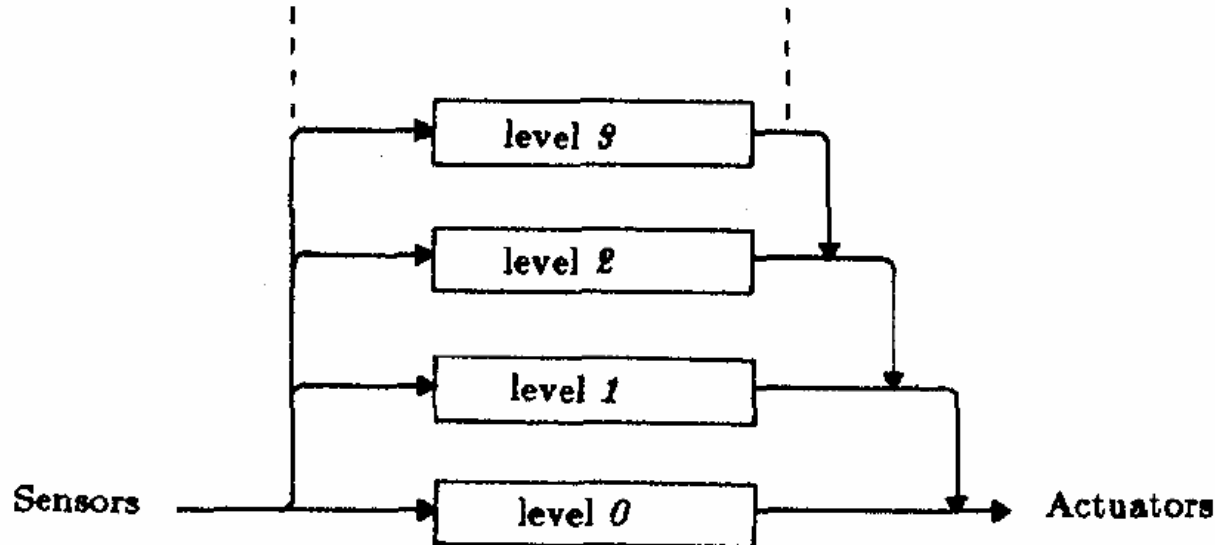
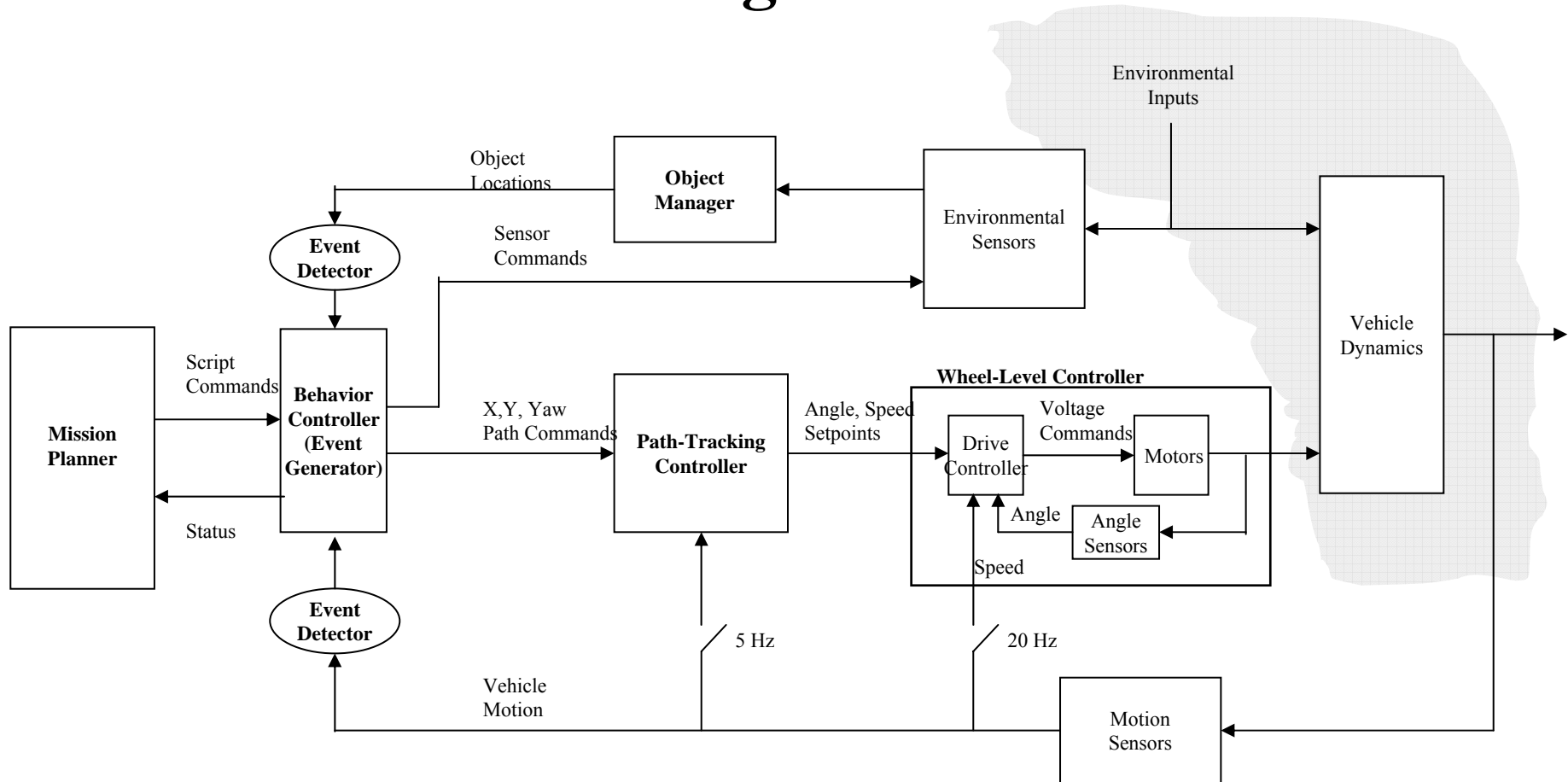


Figure from Rodney Brooks

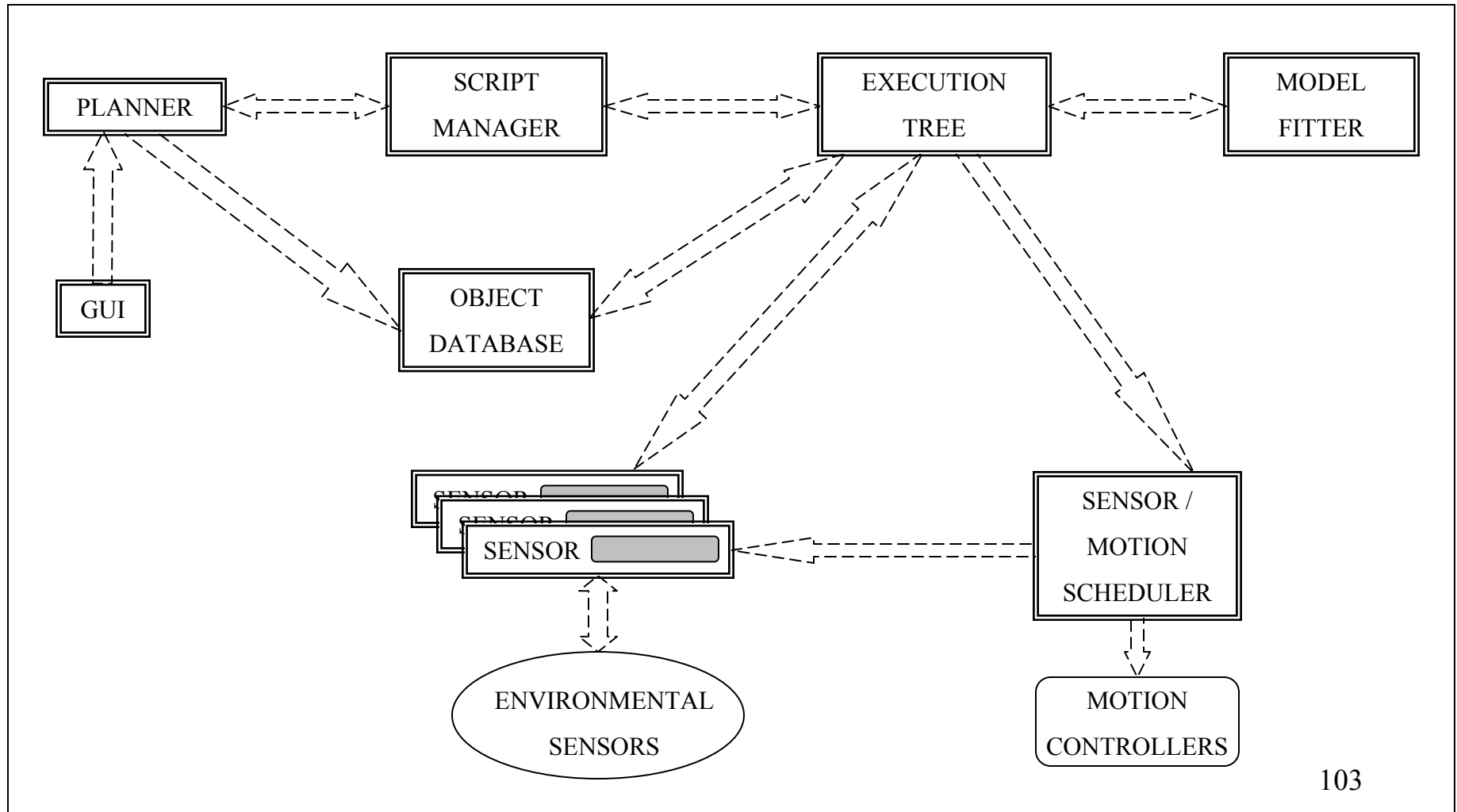
Architectures for Unmanned Systems

- A large number of architecture and strategies have been proposed for intelligent behavior generation for UxVs:
 - Subsumption (Brooks).
 - Input-output relations to solve small problem; prioritization of actuation
 - Behavior-based, hierarchical (Arkin)
 - Uses specific motor schemas for specific behaviors
 - Behavior-based reinforcement learning (Barto/Sutton/Anderson)
 - Output of a difference engine measuring state-goal mismatch and taking action to minimize that mismatch (Minsky)
 - Deliberative/Reactive
 - AI-based planning approach
 - Formal Codes
 - 4D/RCD (Albus from *Outline of a Theory of Intelligence*)
 - JAUS (now an SAE standard)
 - Standards for UAVs

Architectures: Signal Flow/Functional Block Diagrams

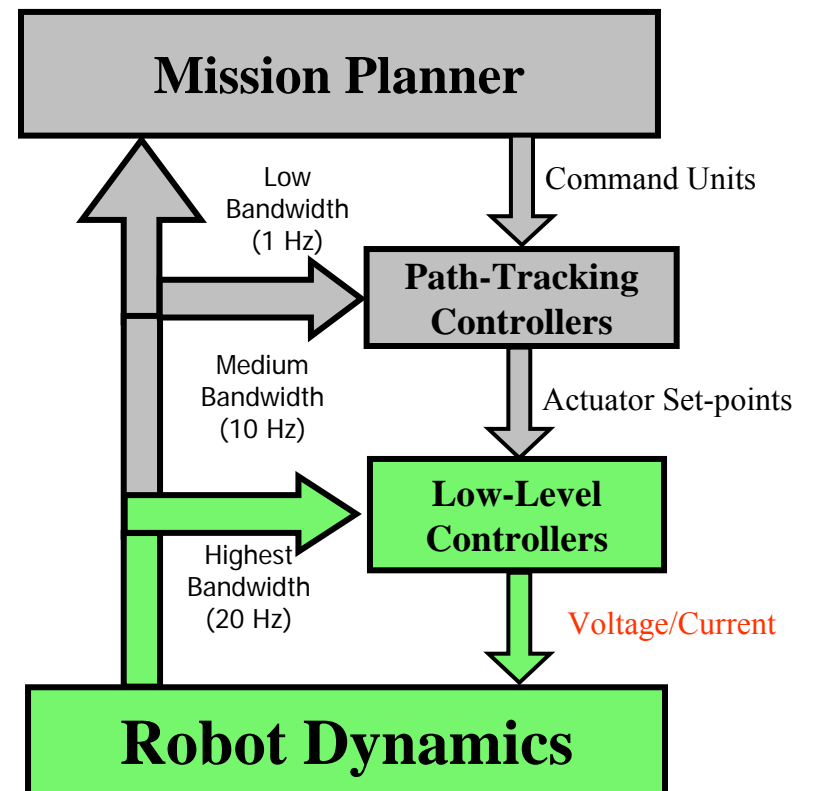


Software Architectures



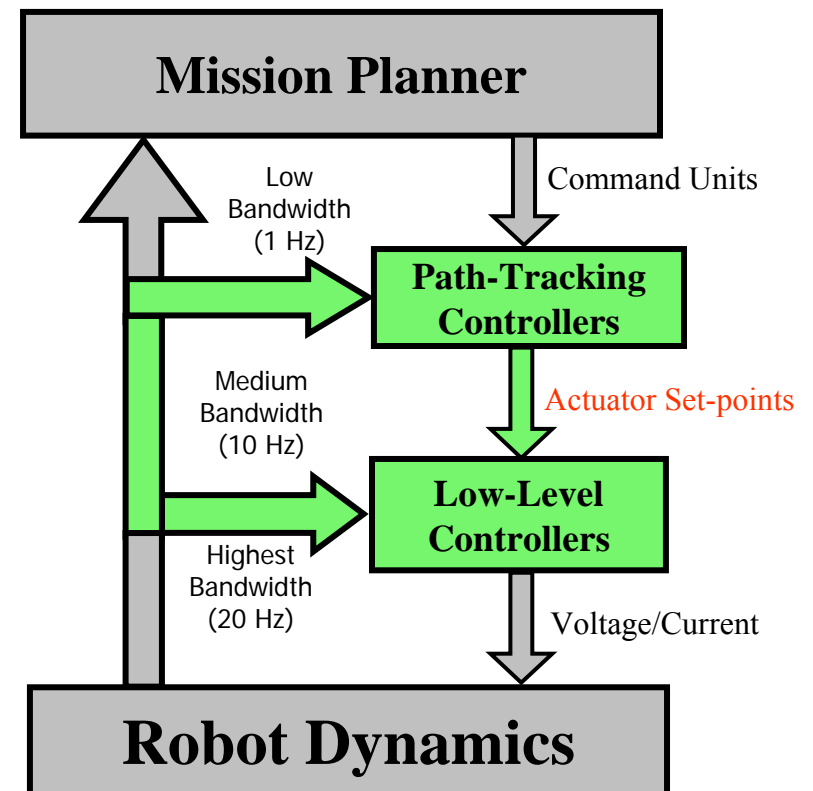
Multi-Resolution Control Strategy

- At the lowest level:
 - Actuators run the robot



Multi-Resolution Control Strategy

- At the middle level:
 - The path tracking controllers generate set-points (steering angles and drive velocities) and pass them to the low level (actuator) controllers



Path Tracking Strategies

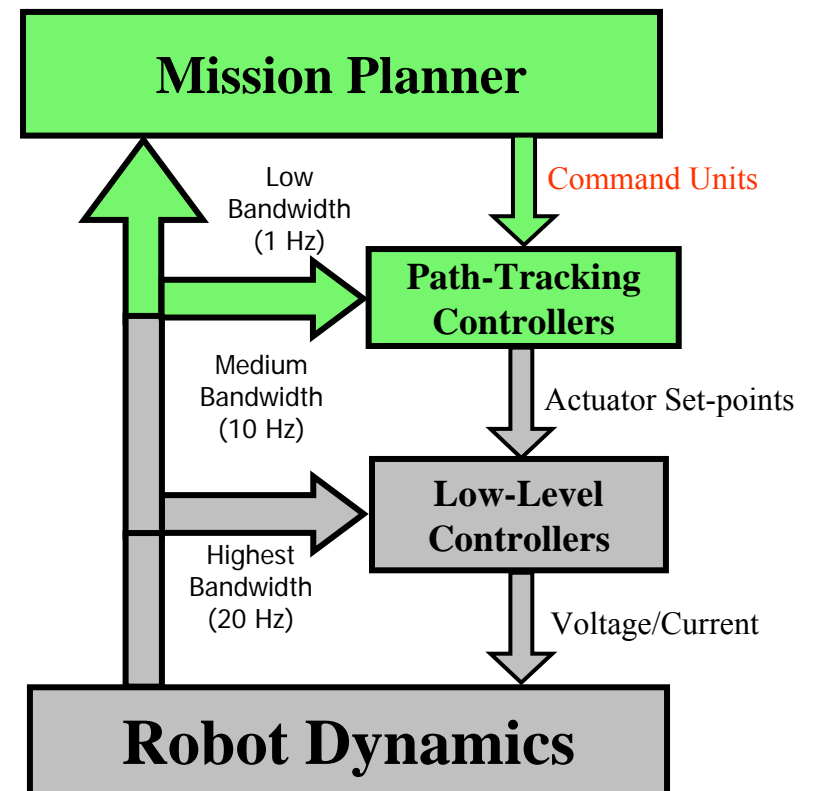
- Fundamental for behavior generation
- Can be broadly classified into two groups
 1. Time trajectory based (temporal)
 - Desired path is parameterized into time-varying set-points
 - Locus of these set-points follow (in time) the desired trajectory (in space)
 2. Spatial
- We have implemented a variety of each type of controller on our robots

T2 Path-Tracking Control

Path Tracking

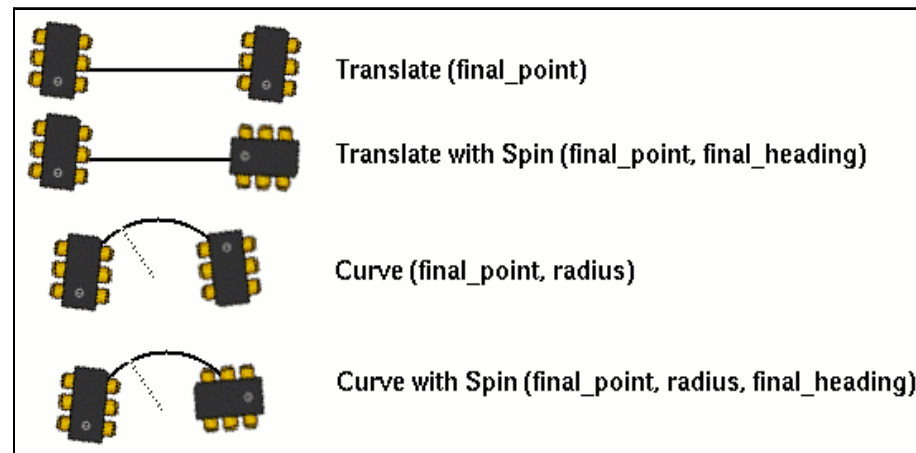
Multi-Resolution Control Strategy

- At the highest level:
 - The mission planner decomposes a mission into atomic tasks and passes them to the path tracking controllers as command-units



Behavior Generation Strategies

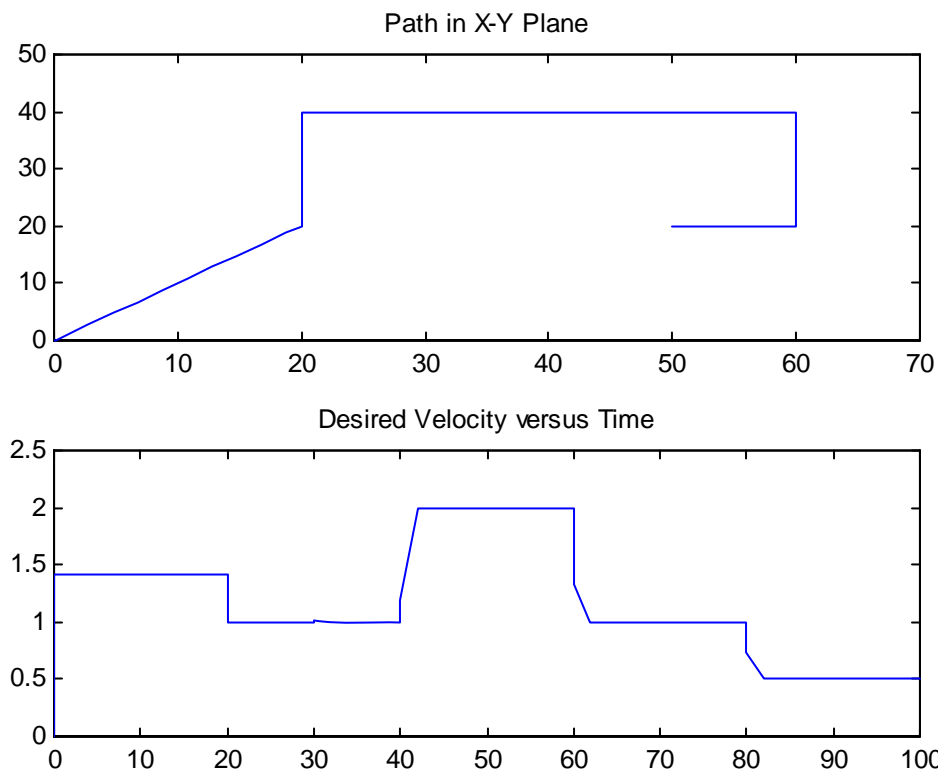
- First Generation: pre-T1
 - Waypoints fit using splines for path generation
 - **User-based path generation**
- Second Generation: T1, T2
 - decomposition of path into primitives
 - fixed input parameters
 - **open-loop path generation**



Example: “z-commands”

- Mission and Path Planner: Generates an appropriate sequence of tasks (basic maneuvers, or “z commands”) to achieve a goal. For example:
 1. Translate to (20,20) with a velocity of 1.414.
 2. Translate to (20,40) with a velocity of 1.
 3. Translate to (60,40) with a velocity of 2.
 4. Translate to (60,20) with a velocity of 1.
 5. Translate to (50,20) with a velocity of 0.5.
- “z-commands” include translate, curve, spin, and combinations of these.

Example: Result of a Series of “z-commands”

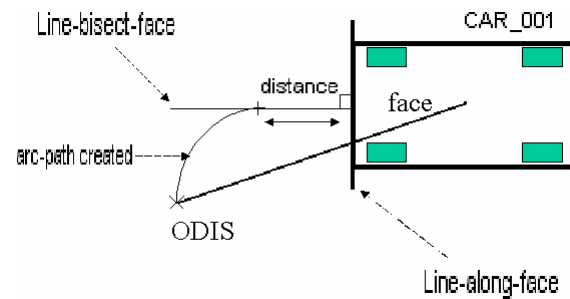
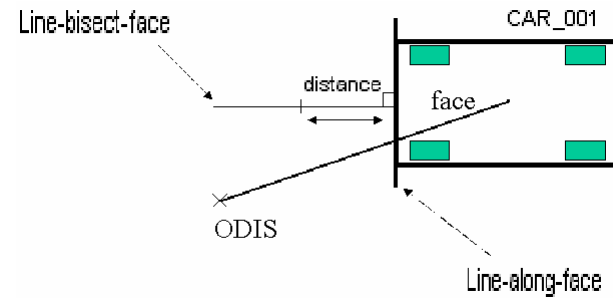
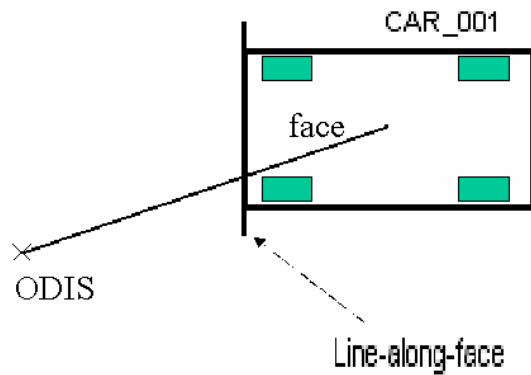


Behavior Generation Strategies

- First Generation: pre-T1
 - Waypoints fit using splines for path generation
 - **User-based path generation**
- Second Generation: T1, T2
 - decomposition of path into primitives
 - fixed input parameters
 - **open-loop path generation**
- Third Generation: T2, T3, ODIS
 - decomposition of paths into primitives
 - variable input parameters that depend on sensor data
 - **sensor-driven path generation**

3rd Generation Maneuver Command: Sensor-Driven, Delayed Commitment Strategy

(ALIGN-ALONG (LINE-BISECT-FACE CAR_001) distance)



ODIS Command Environment

- Developed command environment called MoRSE to implement our delayed commitment approach (Mobile Robots in Structured Environments)
- Has a high degree of orthogonality:
 - a number of small orthogonal constructs
 - mixed and matched to provide almost any behavior
 - effectively spans the action space of the robot
- *Variables* include standard integer, floating point, and geometric data types, such as:
 - Points, lines, arcs, corners, pointsets
 - Data constructs for objects in the environment, to be fit and matched to data
- *Geometric computation* functions:
 - Functions for building arcs and lines from points
 - Functions for returning points on objects
 - Functions for extracting geometry from environment objects
 - Functions to generate unit vectors based on geometry
 - Fitting functions to turn raw data into complex objects
 - Vector math

ODIS Command Environment - 2

- A key feature of MoRSE is the command unit:
 - Set of individual commands defining various vehicle actions that will be executed in parallel
- *Commands for XY movement:*
 - `moveAlongLine(Line path, Float vmax, Float vtrans = 0)`
 - `moveAlongArc(Arc path, Float vmax, Float vtrans = 0)`
- *Commands for Yaw movement:*
 - `yawToAngle(Float angle_I, Float rate = max)`
 - `yawThroughAngle(Float delta, Float rate = max)`
- *Commands for sensing:*
 - SenseSonar
 - SenseIR
 - SenseLaser
 - Camera commands
- A set of rules defines how these commands may be combined

Rules for Combining Commands to Form a Command-Unit

- At most one command for XY movement
- At most one command for yaw movement
- Only one Rapid-stop command
- At most 1 of each sense command (laser, sonar, IR)
- At most 1 command for camera action
- No XY, yaw movement, and senseLaser commands allowed with Rapid-stop command
- No yaw movement command when a senseLaser command is used

Example Macroscript - 1

findCar() script

- If there is a car, find bumper and move closer.
- Fit the open-left tire.
- Fit the open-right tire.
- Move up the centerline of car.
- Fit the closed-left tire.
- Fit the closed-right tire.
- Fit the entire car and prepare for inspection.

Example Macroscript - 2

The detailed structure of the first two steps is as follows:

If (car) fit bumper and move in

fire sonar at rear of stall

if there is something in the stall

fire sonar at front half of stall

fit bumper_line

move to \cap of bumper_line with c.l. of stall

fit tire_ol

coarse scan of ol and or_quadrants

move to the line connecting two data centroids

arc and detail scan around the ol data centroid

fit tire_ol with the resulting data

else go to next stall

Example Macroscript - 3

Actual Code

```

If (car) fit bumper and move in
sense_sonar_duration = 1.0;
sense_radius = max_sonar_range;
<<<
// fires sonar to see if there is a car in the stall.
senseSonar( my_stall.p_cl, my_stall.p_cr,
sonar_cutoff_radius, sense_sonar_duration );
>>>
sonar_data = getSonarData();
// If there is a car.
if ( sonar_data.size > 5 &&
    pointsInsideStall ( sonar_data.mean(),my_stall ))
{
    Line stall_centerline;
    Line line_to_bumper;
    Line bumper_line;
    Vector stall_x_axis; // Unit vector pointing toward
                        //the face_c of stall.
    Vector stall_y_axis; // Unit vector 90 degrees from
                        //stall_x_axis.
    Point stall_cline_and_bumper_inter;
    sense_sonar_duration = 4.0;
    sonar_cutoff_radius = dist_from_stall +
                        my_stall.face_r.length() * 0.5;

```

```

if( fitLineSegLMS( sonar_data, bumper_line ) <=
    minimumConfidence )
{
    // Fit is not good.
    ?return 0;
}
stall_centerline=makeLine(my_stall.face_o.midPoint()
,my_stall.center() );
stall_x_axis = stall_centerline.unitvec();
stall_y_axis = rotateVec( stall_x_axis, 90 );
stall_cline_and_bumper_inter = LineIntersection(
bumper_line,stall_centerline );
line_to_bumper = makeLine( entry_point,
stallcl_and_bumper_int );
<<<
// moves in to the intersection of the bumper line with
// the stall centerline.
moveAlongLine( line_to_bumper, max_velocity );
>>>
...
}

```

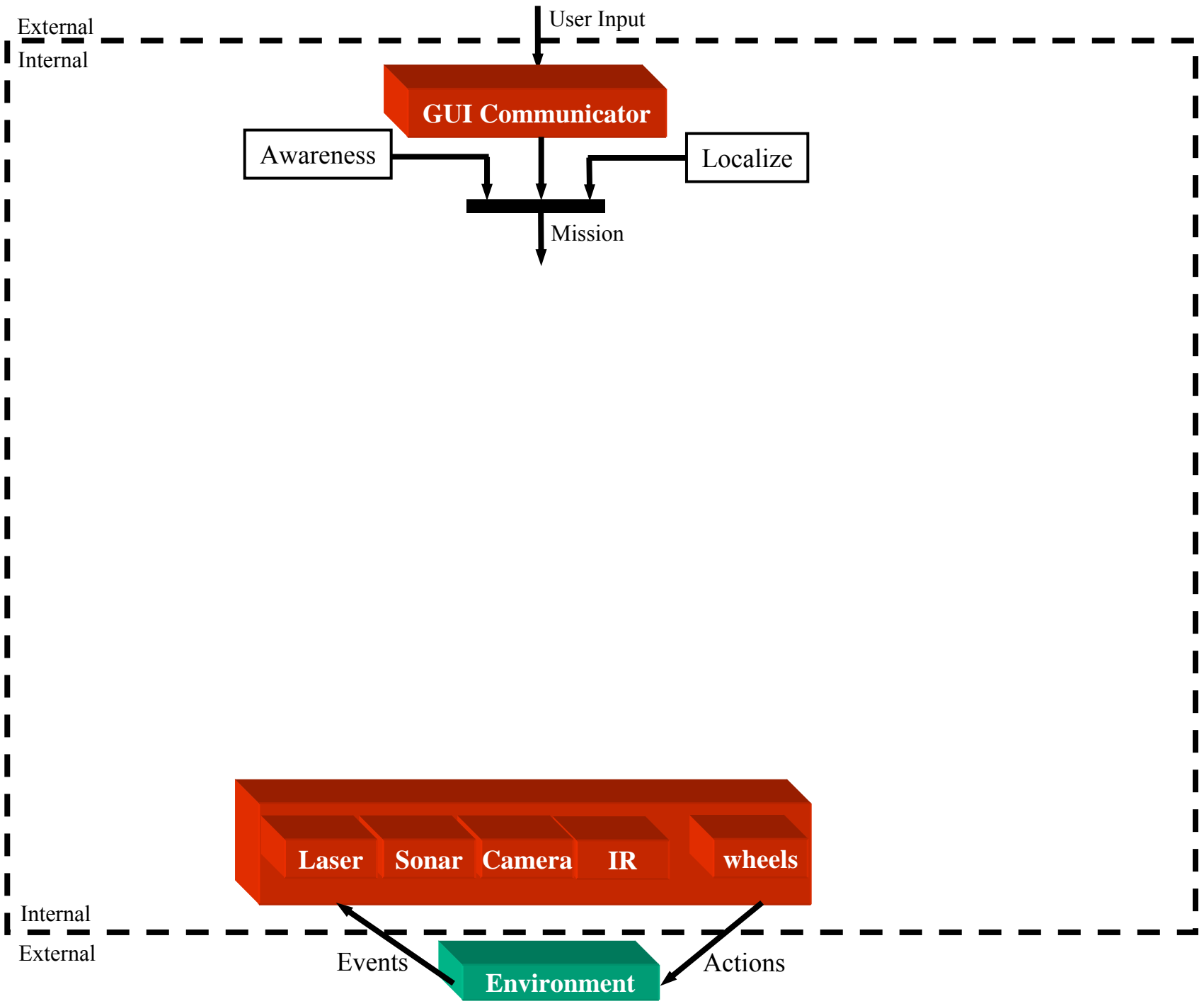
Example: ODIS FindCar() Script



ODIS
DEMONSTRATION

- Command actions are the lowest-level tasks allowed in our architecture that can be commanded to run in parallel
- For planning and intelligent behavior generation, higher-level tasks are defined as compositions of lower-level tasks
- In our hierarchy we define:

Mission	}	User-defined
Tasks		
Subtasks	}	Variable (planned)
Atomic Tasks (Scripts)		
Command Units		
Command Actions	}	Hard-wired (but, (parameterized and sensor-driven)

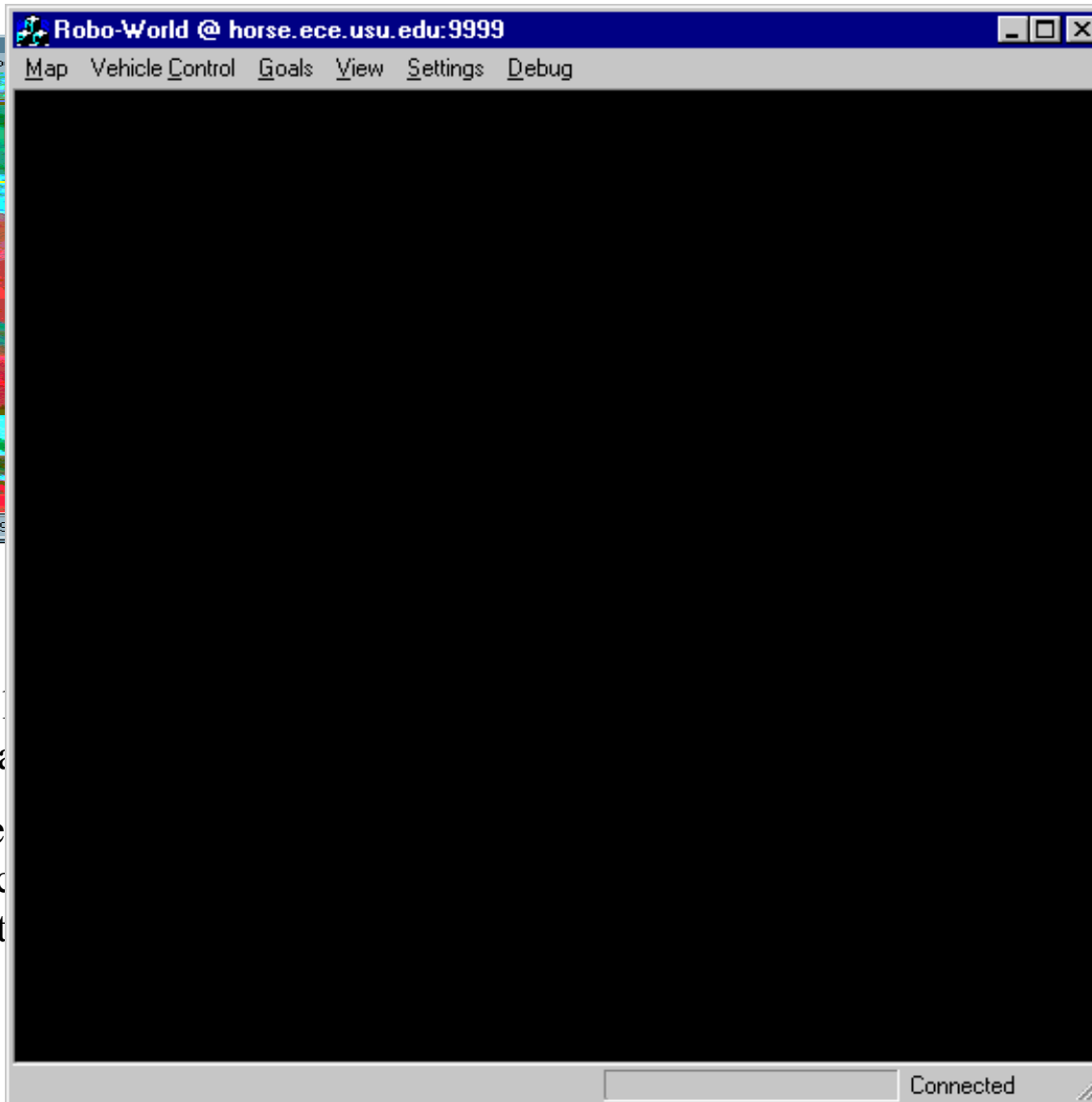
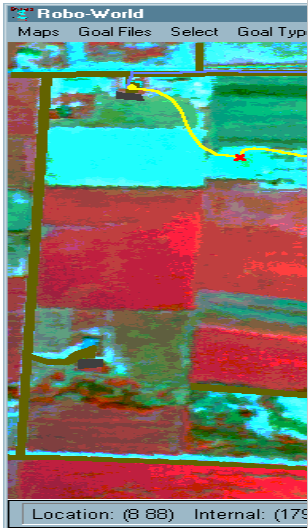


Operator Interfaces

- Like architectures, the topic of operator interfaces has attracted an enormous amount of activity:
 - Currently 22 different government-sponsored studies on common OCUs
- For tele-operated vehicles the design of an OCU may differ significantly from that for an autonomous vehicle
 - Tele-operation requires coordination of many degrees of freedom
 - Autonomous systems need a mission-specific context driving the OCU, with an emphasis on sensor fusion



Farming Automation Projects



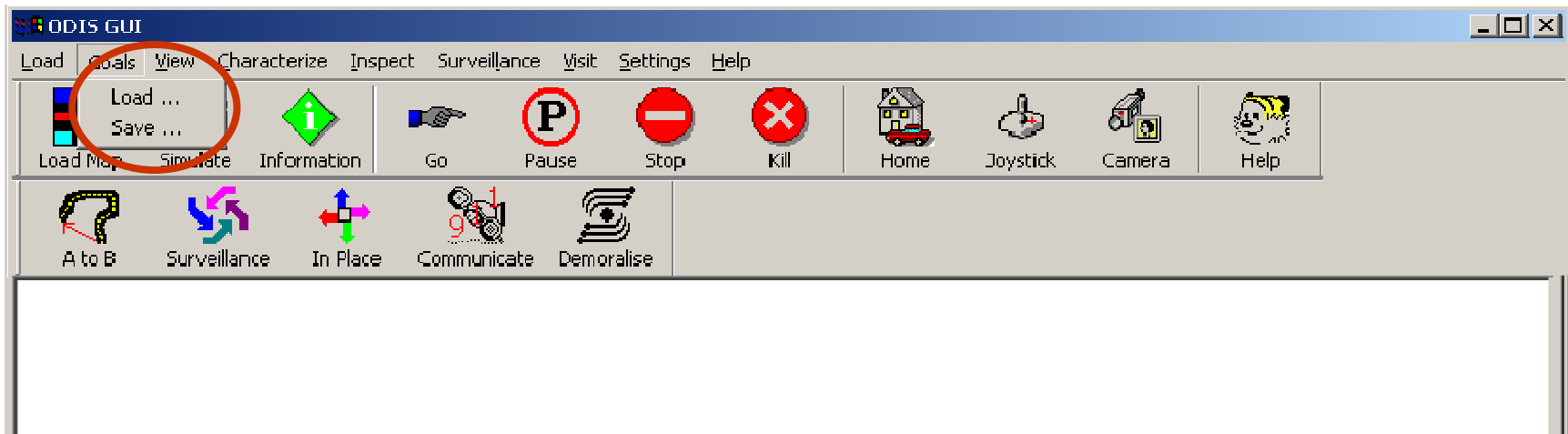
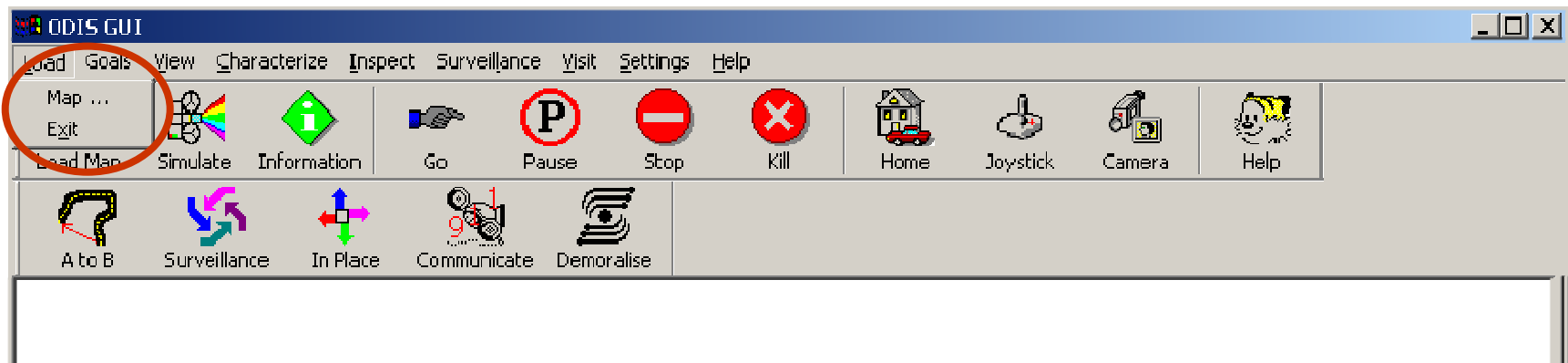
Co-operative
ng
llite map of the
sks are
lligent path and

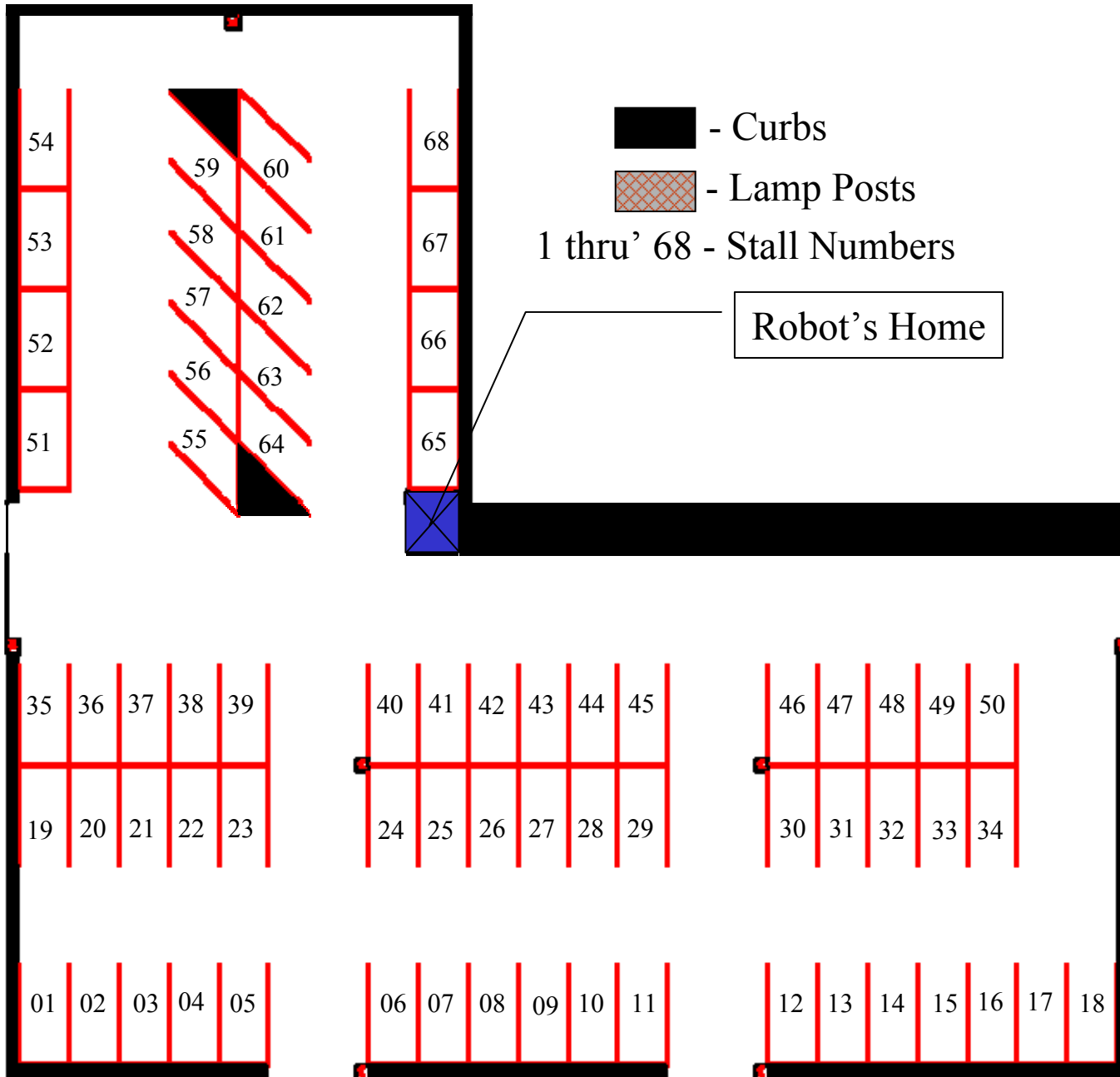
expected
ures by re-
n and path

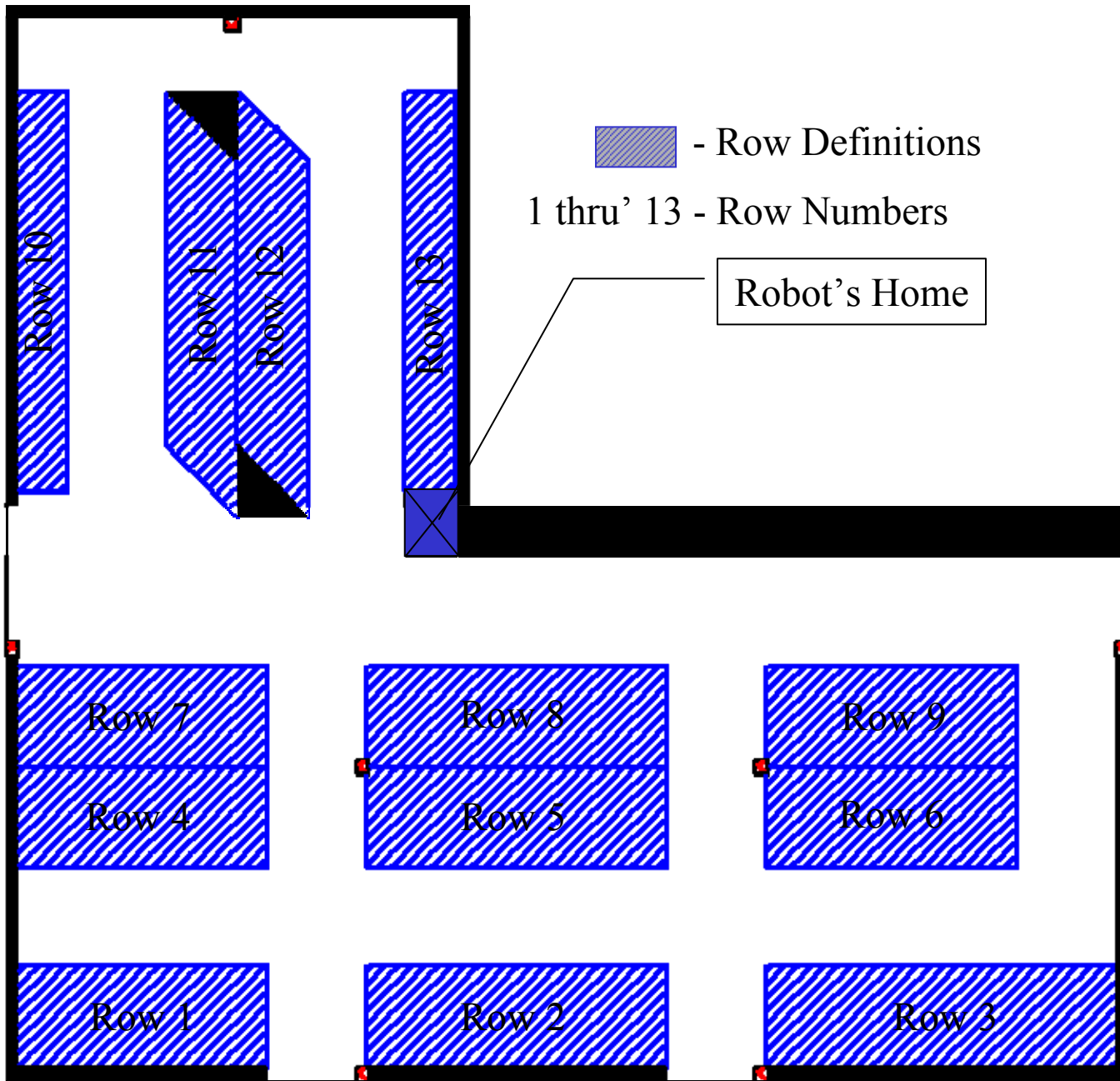


- Technology autonomous dGPS navigation
- Prototypes equipment, radiation detection

An ODIS Mission Command GUI

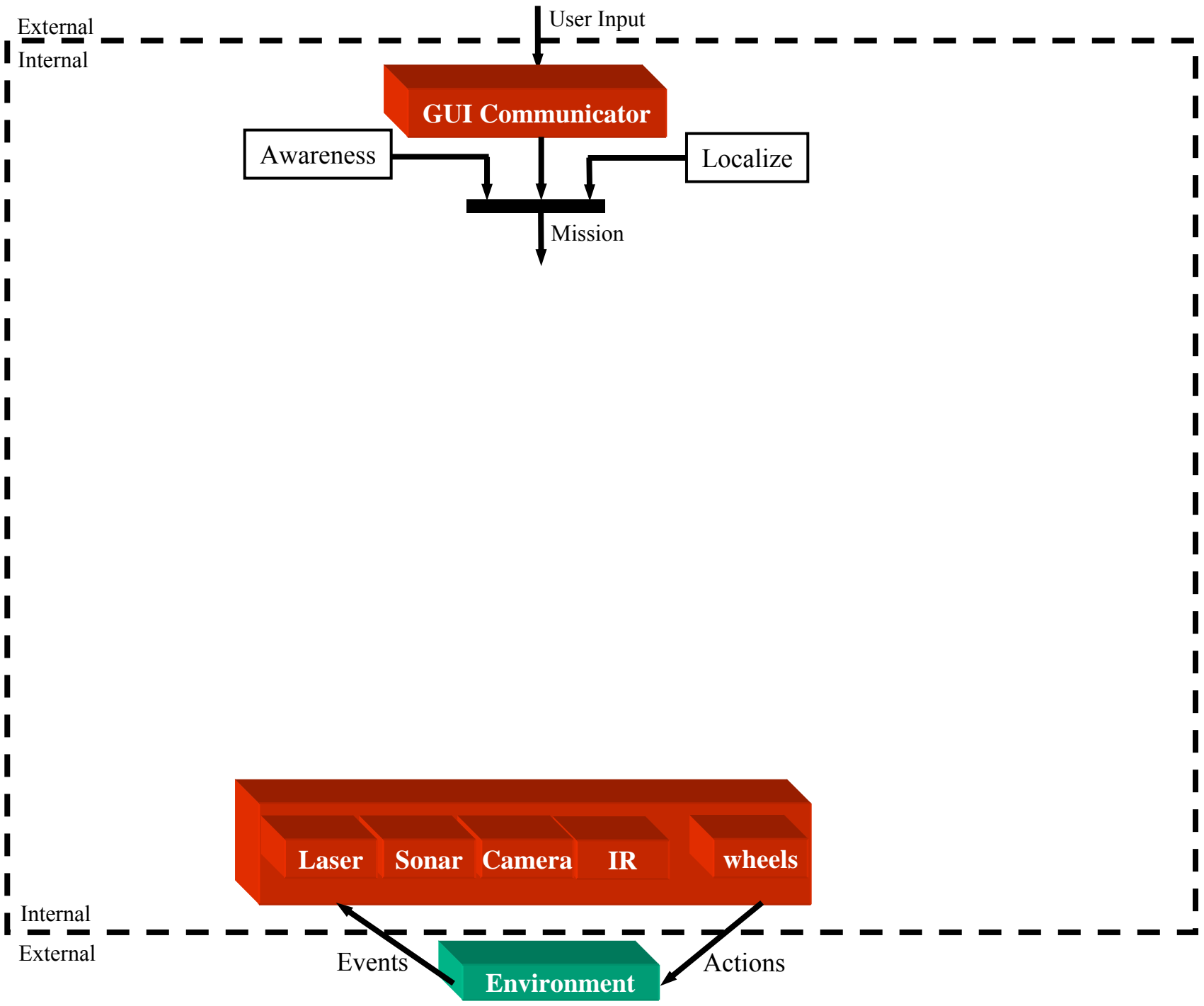


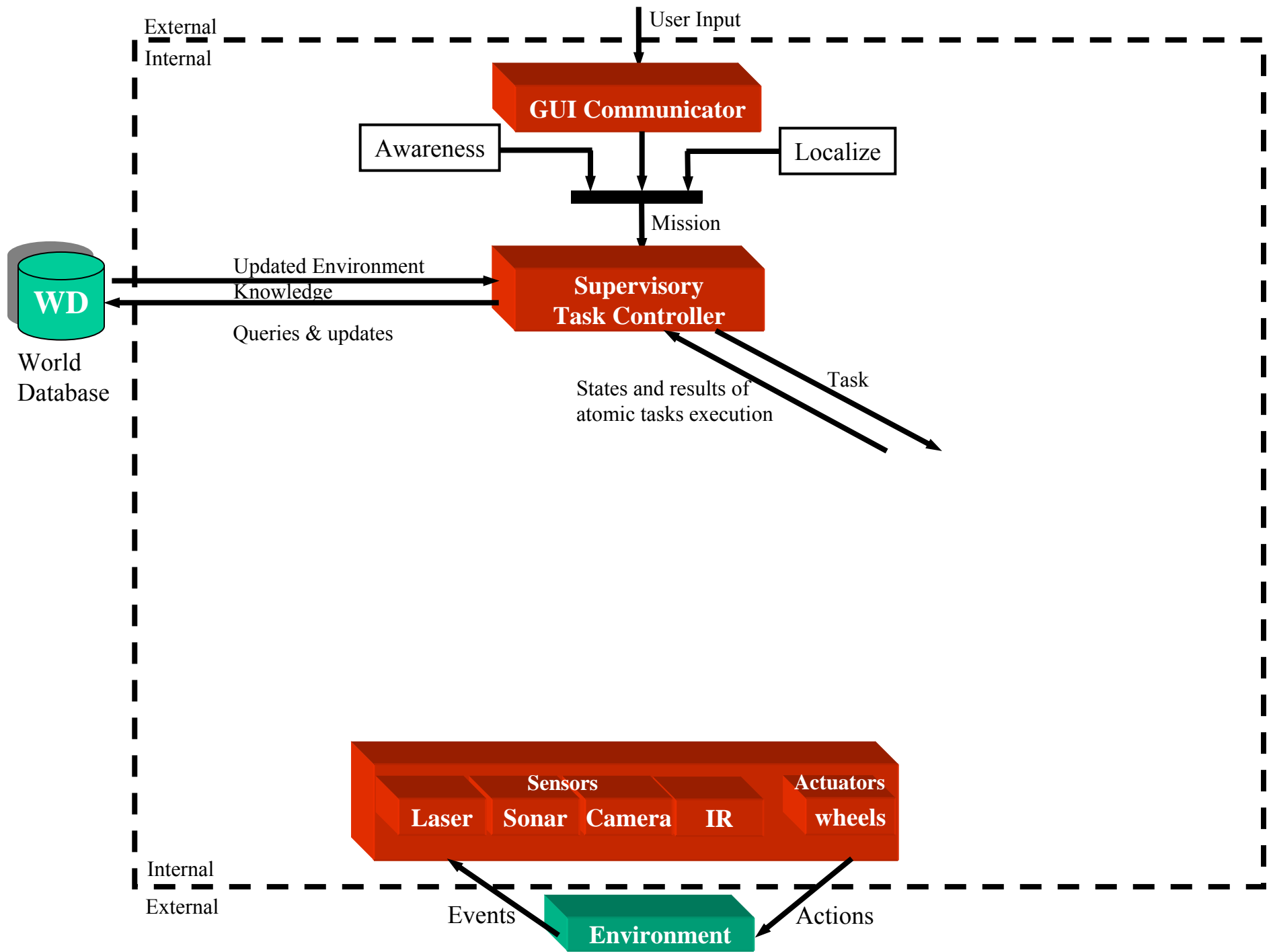


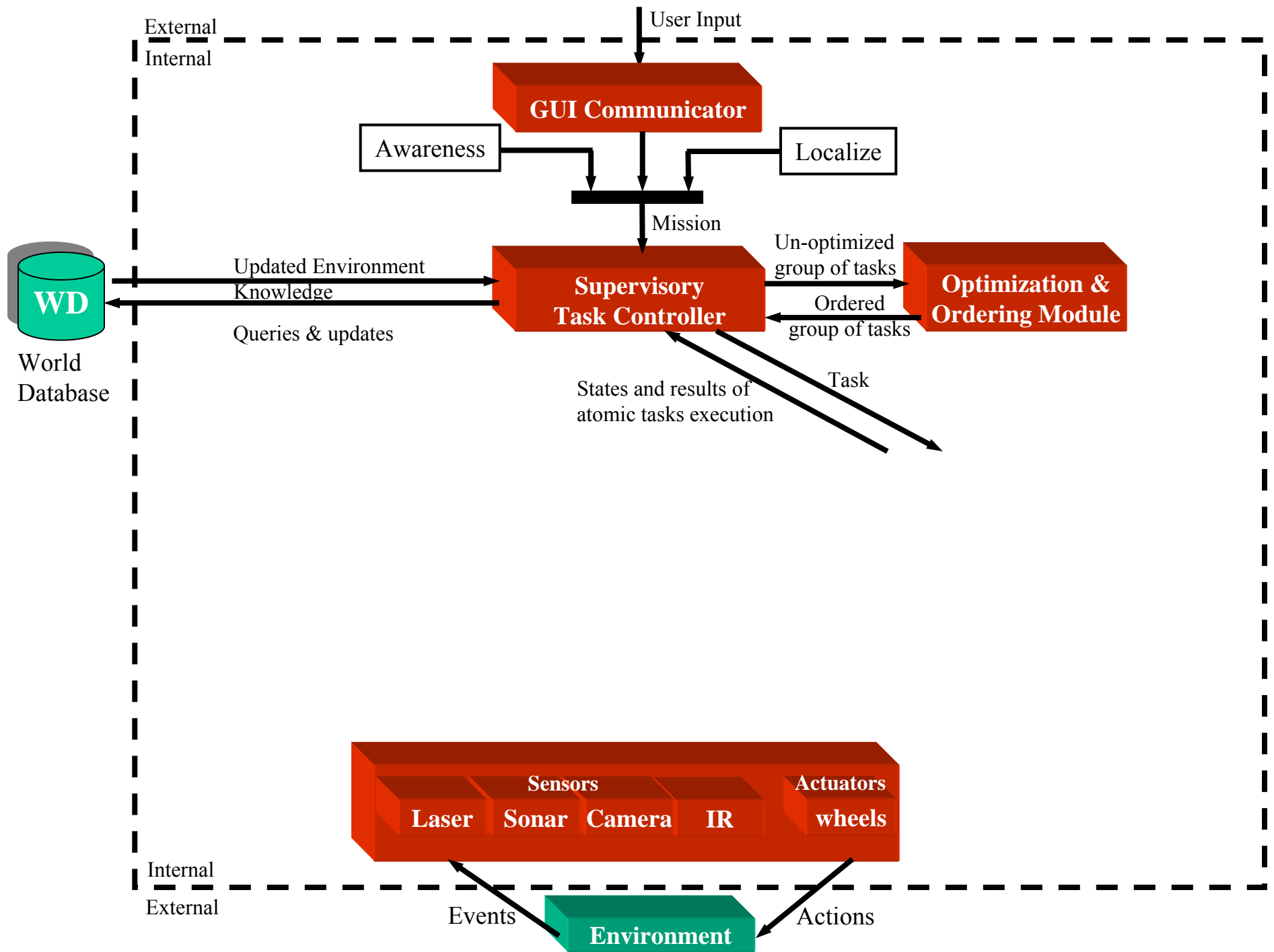


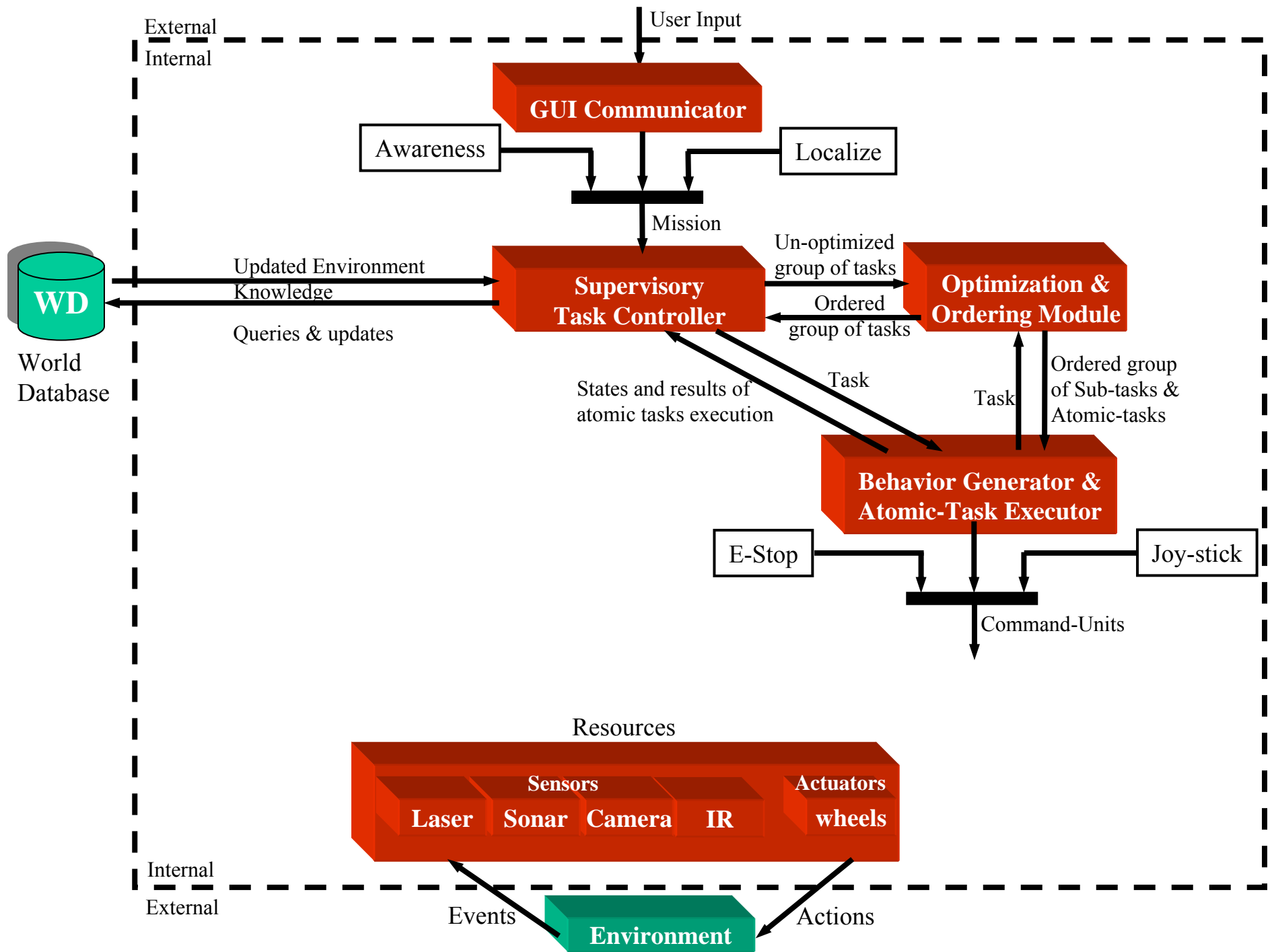
User-tasks in the environment

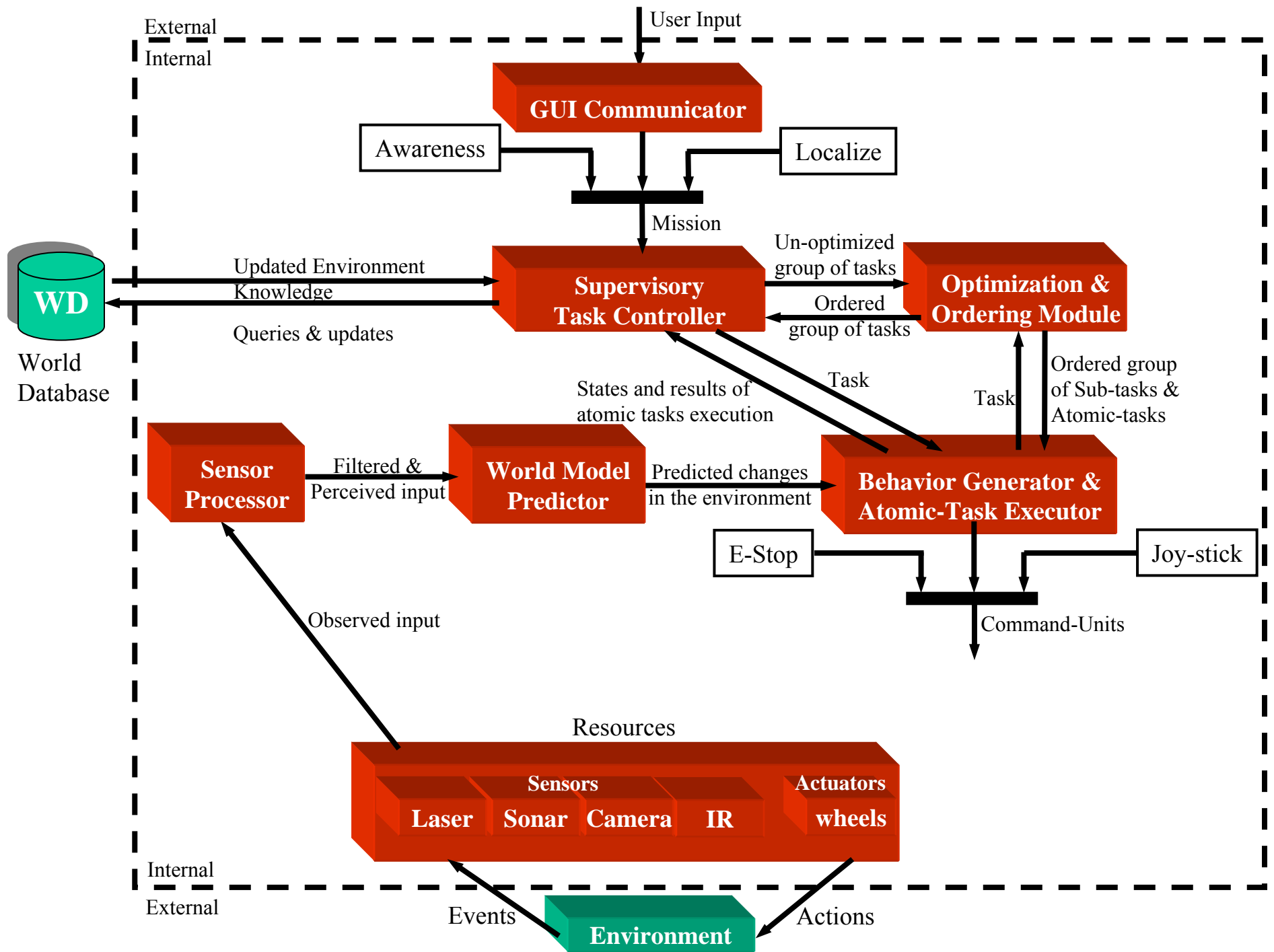
- {MoveTo Point}
- {Characterize a stall}
- {Inspect a stall}
- {Characterize a row of stalls}
- {Inspect a row of stalls}
- {Localize}
- {Find my Car}
- {Sweep the parking lot}
- {Sweep Specific area of the parking lot}

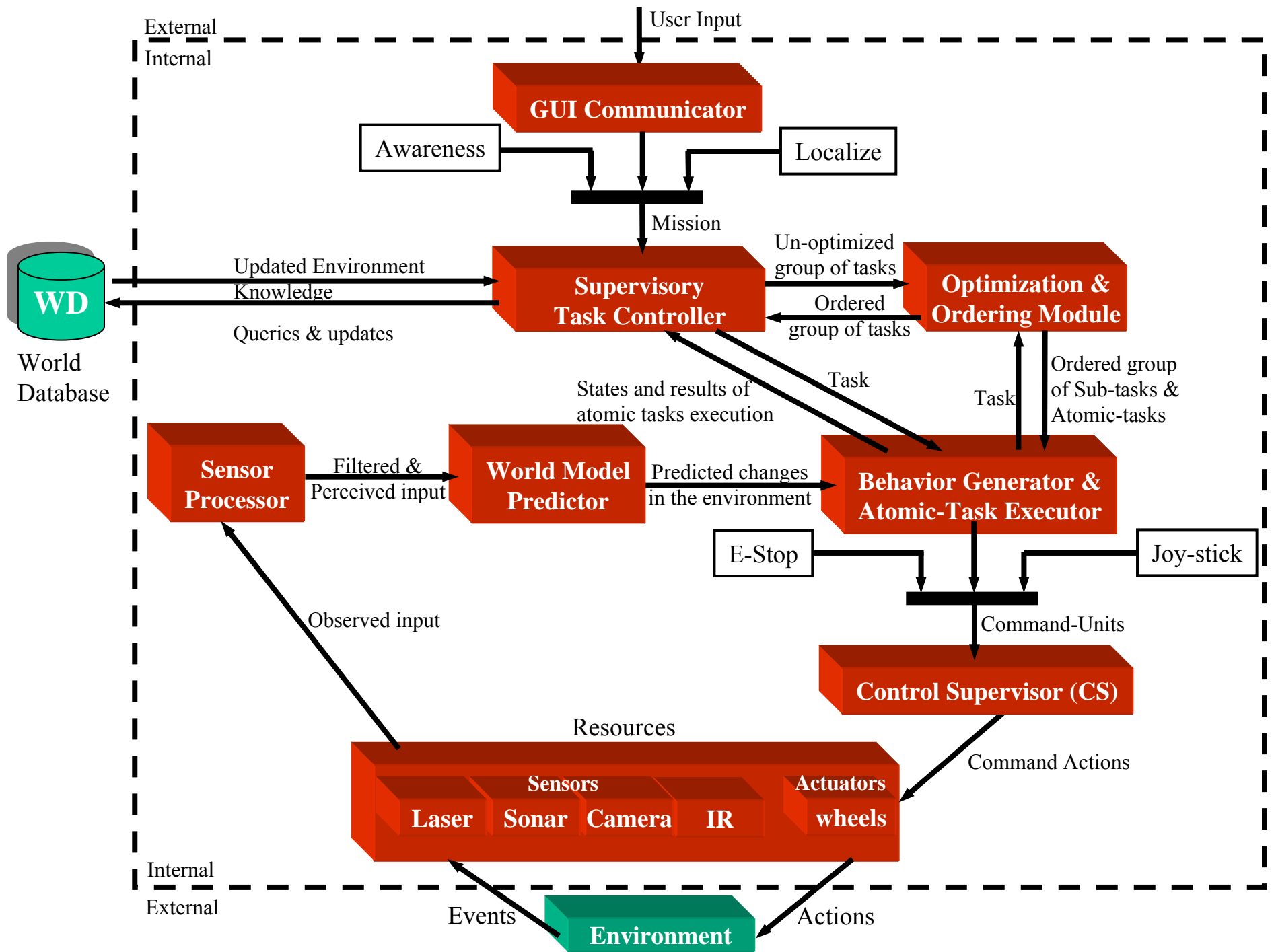












Reactive Behaviors

Reactive behaviors are induced via:

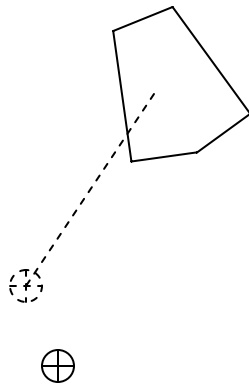
1. Localization thread
 - Compares expected positions to actual sensors data and makes correction to GPS and odometry as needed
2. Awareness thread
 - Interacts with the execution thread based on safety assessments of the environment
3. Logic within the execution thread
 - Scripted adaptive behaviors
 - Exit conditions at each level of the hierarchy determine branching to pre-defined actions or to re-plan events

Key Capability Requirement- Localization

- ODIS's missions require lots of motion, including rotation.
- Problems occur in “knowing where we are” due to:
 - GPS dropout
 - Drift in fiber-optic gyro
- Two solutions:
 - Landmark-based localization using ODIS's scripting language and ODIS's laser
 - Wireless visual-servoing using ODIS's camera

Method 1 - Example Script: Localization to a Curb

Actions: Move to what ODIS *thinks* is the origin,
and yaw to what ODIS *thinks* is // curb.



```
Line temp_line = Line(position(), Point(0.0, 0.0, 0.0));  
Float temp_angle = Atan2( curb.leg2.y, curb.leg2.x );
```

```
<<<<
```

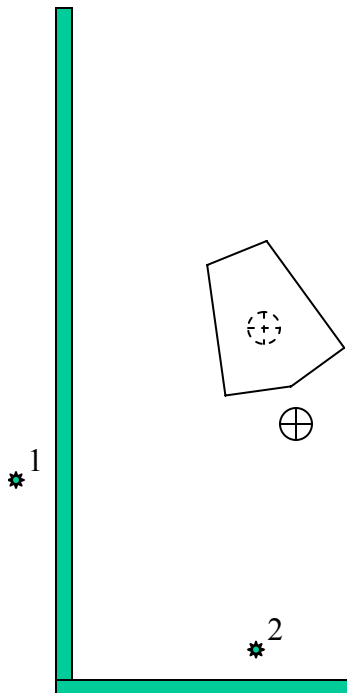
```
moveAlongLine(temp_line, v_max, v_trans);
```

```
yawToAngleCoupled(temp_angle);
```

```
>>>>
```

Method 1 - Example Script: Localization to a Curb

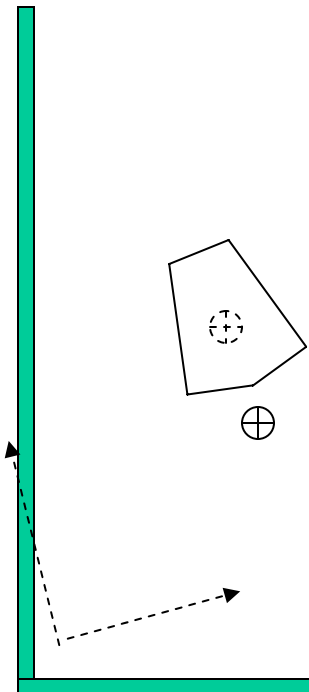
Actions: Scan where *ODIS expects* the curb to be



```
Point scan_point_1 = curb.vertex + curb.leg2;  
Point scan_point_2 = curb.vertex + curb.leg1;  
  
<<<  
senseLaser( scan_point_1, scan_point_2, cutoff_radius );  
>>>
```

Method 1 - Example Script: Localization to a Curb

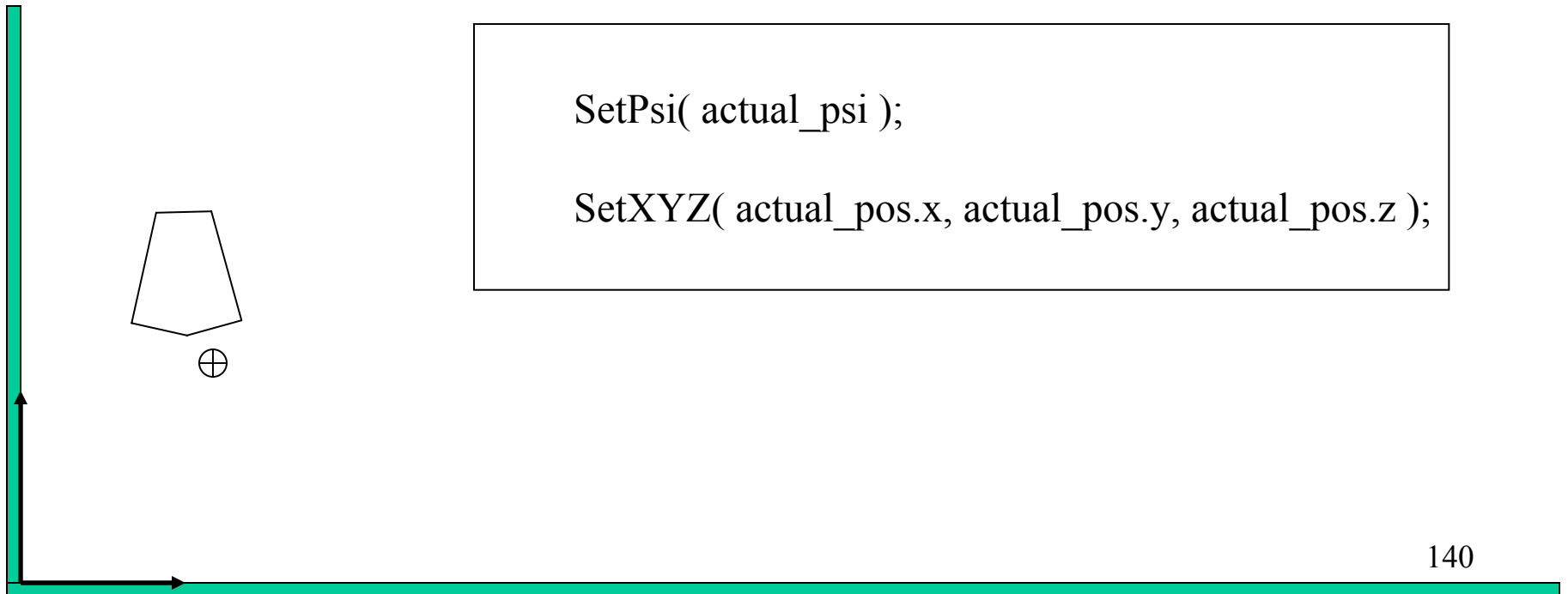
Actions: Get the laser data and fit it to a curb



```
PointSet scan_data = getLaserData();  
  
if ( fitCorner(scan_data, curb_est) >= corner_fit_conf )  
{  
    Print("The Corner Is Fit.");  
}
```

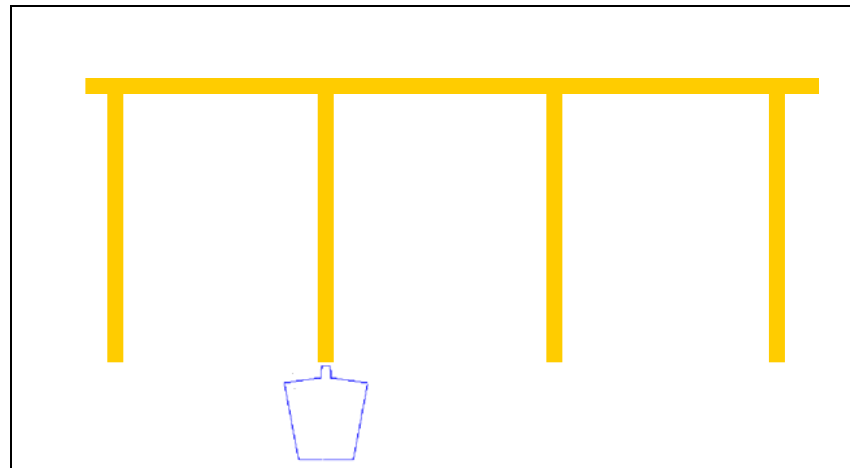
Method 1 - Example Script: Localization to a Curb

Actions: Reset yaw and position, based on curb data



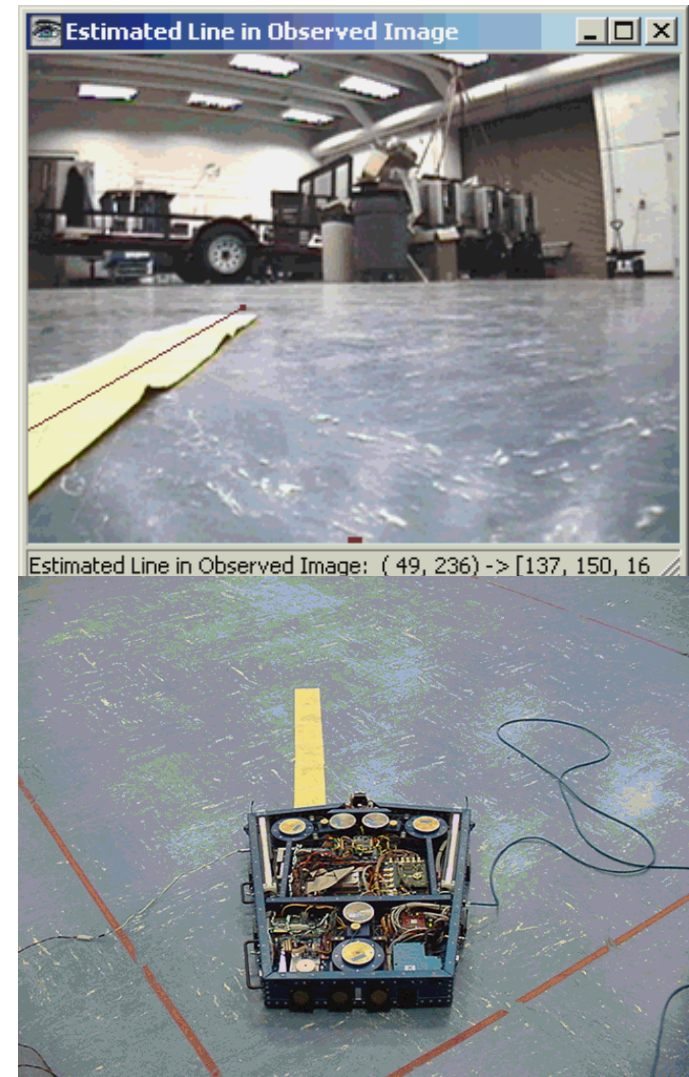
Summary of Approach

- The idea can be summarized as follows:
 - Actions can be described relative to objects or features of objects that exist or are expected to exist
- In the second approach, we apply this idea again, but now using a camera to (iteratively) align to yellow lines in the parking lot
- Goal is to correct ODIS's orientation

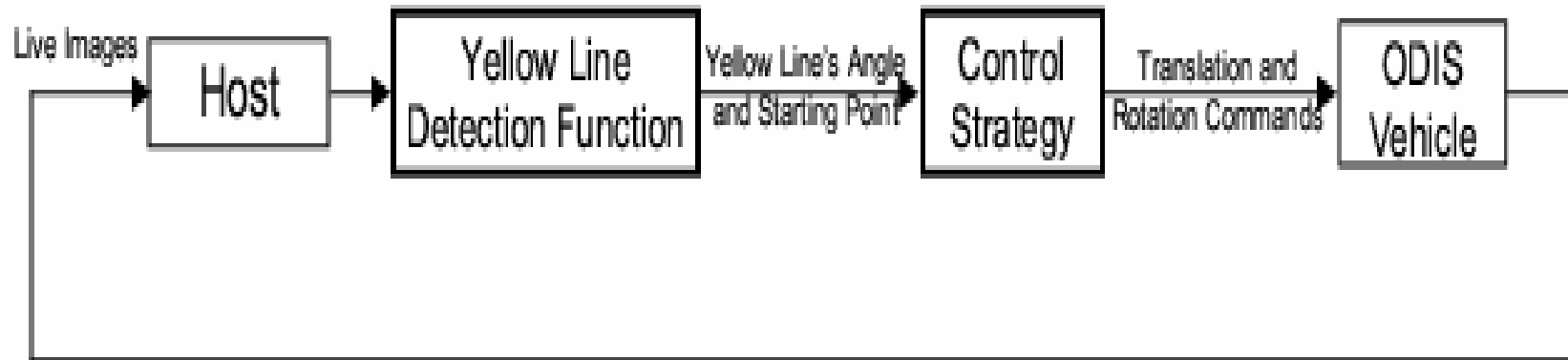


Example: Localization to yellow lines

- Periodically the fiber-optic gyro is reset
- Yellow line is identified in camera image
- Vehicle is rotated to align its body-centered axis with identified line
- Process repeats



Visual Servoing Strategy

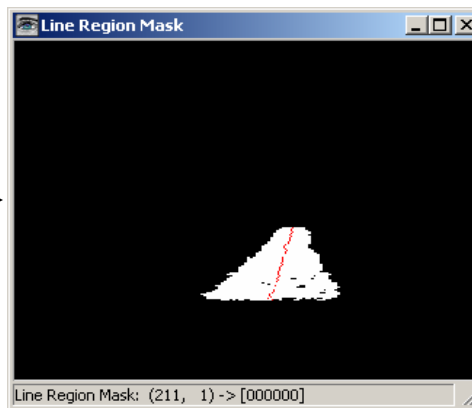
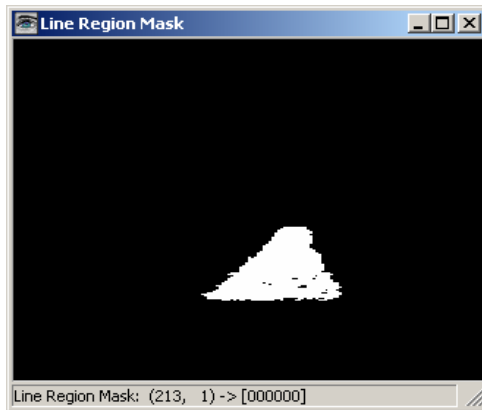
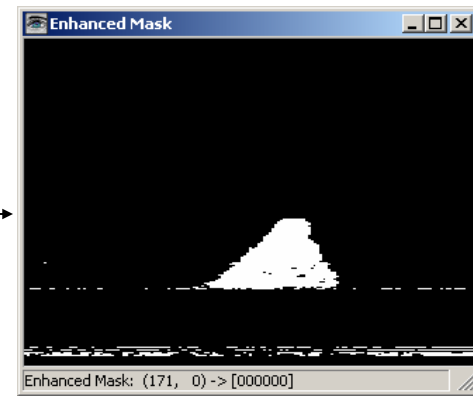
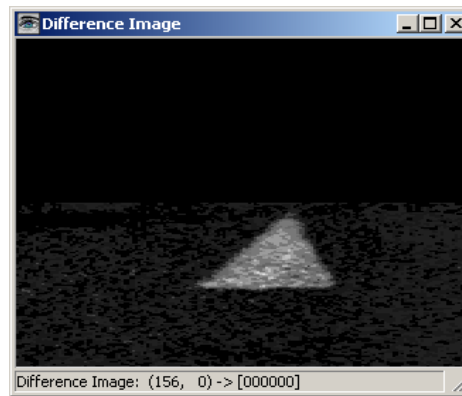
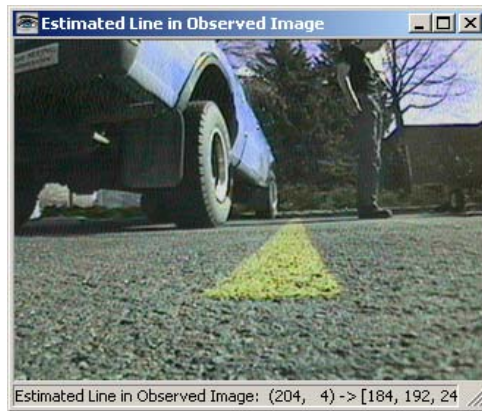


- Two key algorithms needed:
 1. Yellow Line Detection
 2. Control Strategy (visual servoing algorithm)

Yellow Line Detection

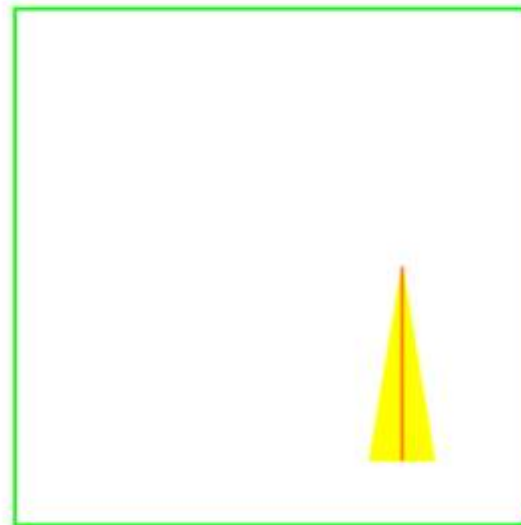
- Five Steps:
 - Yellow Color Segmentation
 - Connected Component Labeling
 - Select Maximal Region
 - Locate Line Points
 - Line Fitting

Yellow Line Detection

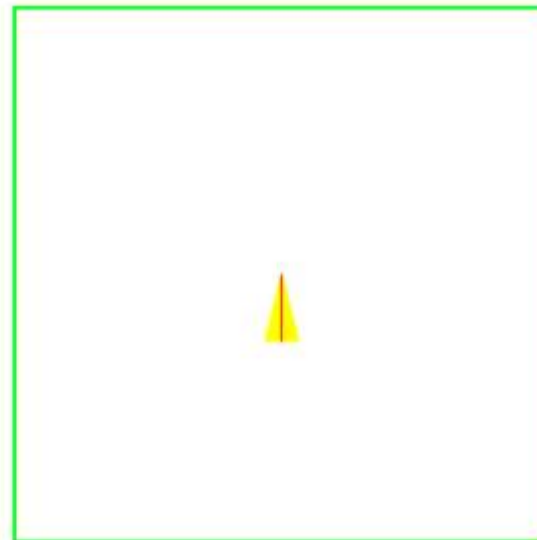


Control Strategy : Intuition

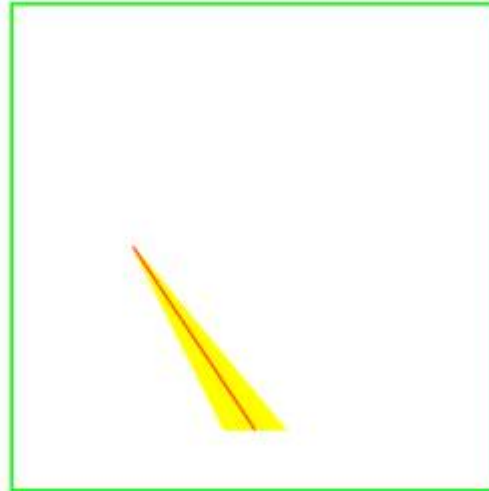
- If the line is too far from the left in image plane
 - move ODIS to the left
- If the line is too high in image plane
 - move ODIS forward
- If the line needs to rotate clockwise to be vertical in image plane
 - rotate ODIS anti-clockwise.



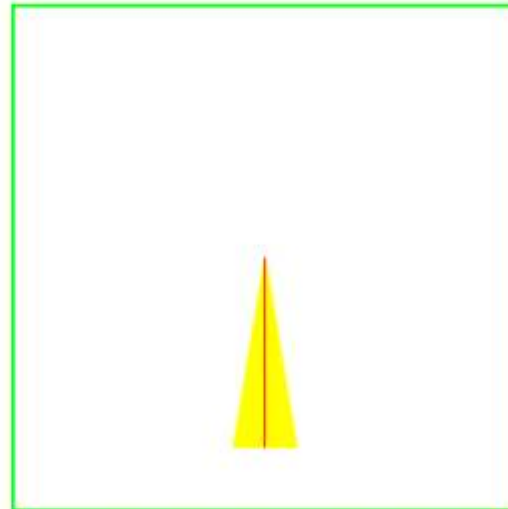
Vehicle should move right.



Vehicle should drive forward.



Vehicle should rotate counter-clockwise.

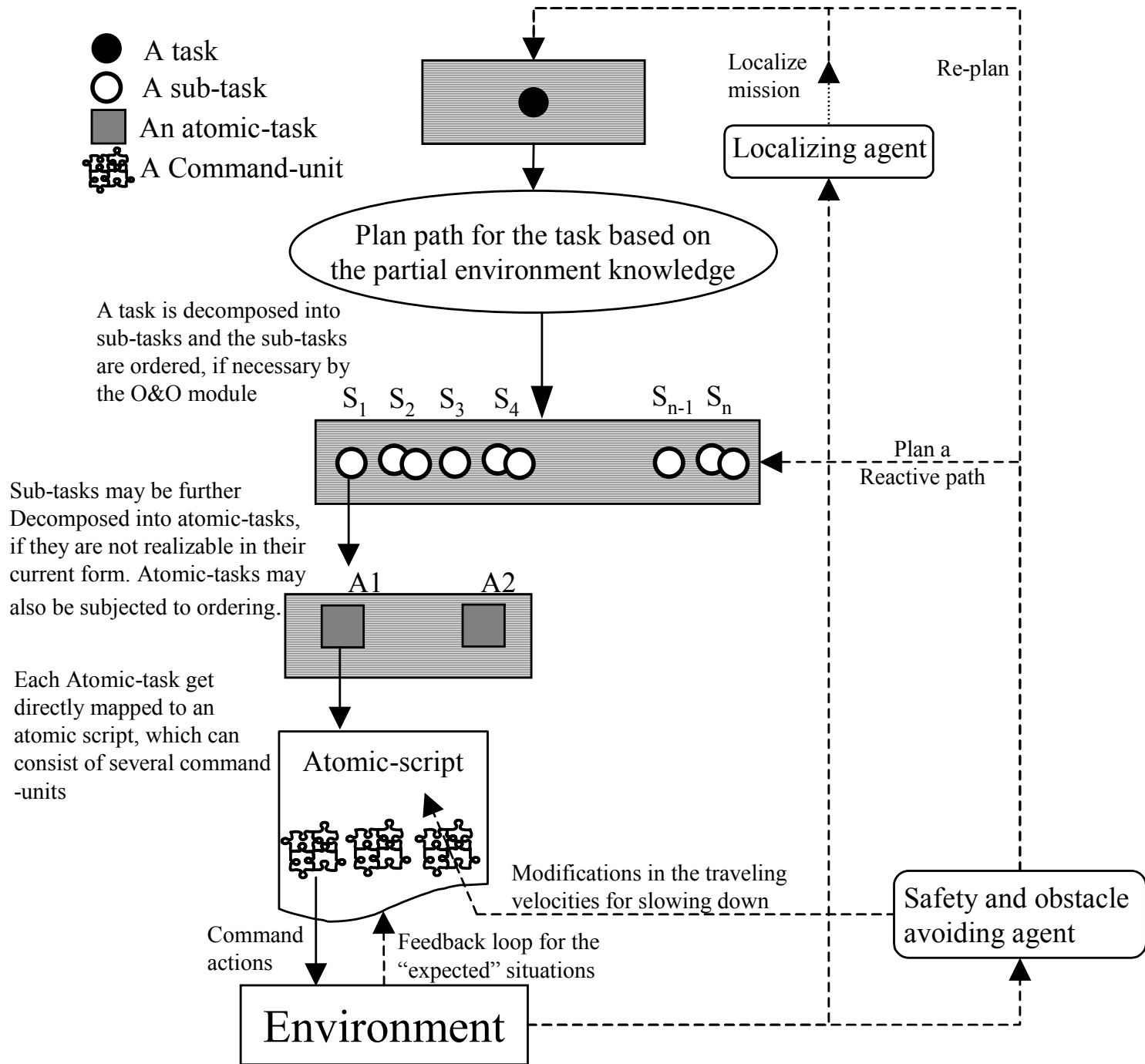


Vehicle is in the desired location.

Reactive Behaviors

Reactive behaviors are induced via:

1. Localization thread
 - Compares expected positions to actual sensors data and makes correction to GPS and odometry as needed
2. Awareness thread
 - Interacts with the execution thread based on safety assessments of the environment



Awareness Thread



Reactive Behaviors

Reactive behaviors are induced via:

1. Localization thread
 - Compares expected positions to actual sensors data and makes correction to GPS and odometry as needed
2. Awareness thread
 - Interacts with the execution thread based on safety assessments of the environment
3. Logic within the execution thread
 - Scripted adaptive behaviors

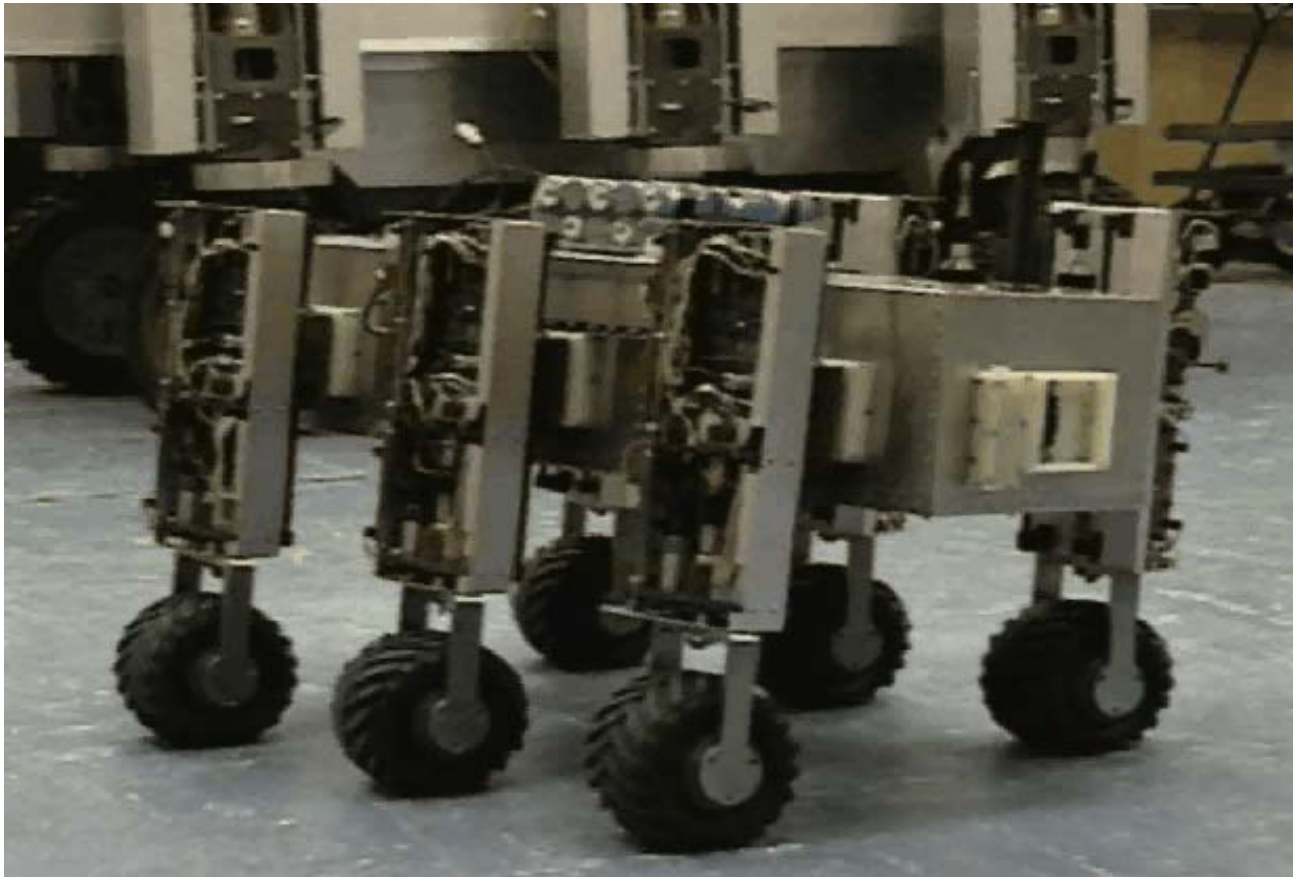
T2 Adaptive/Reactive Hill-Climbing

Hill Climb Maneuver Under Path Planner Control

algorithm HillClimbingMethod

1. Set drive motor voltage $V_D = V_{D_{max}}$ (throughout).
2. Head for goal.
3. **if** AtBoundary
 Flip-flop steering angle.
 else if TooSlow
 Steer downhill a little.
 else if TooFast
 Steer toward goal a little.
 else
 Continue in the same direction.
4. **if** AtGoal
 endalgorithm
 else
 do Line 3.

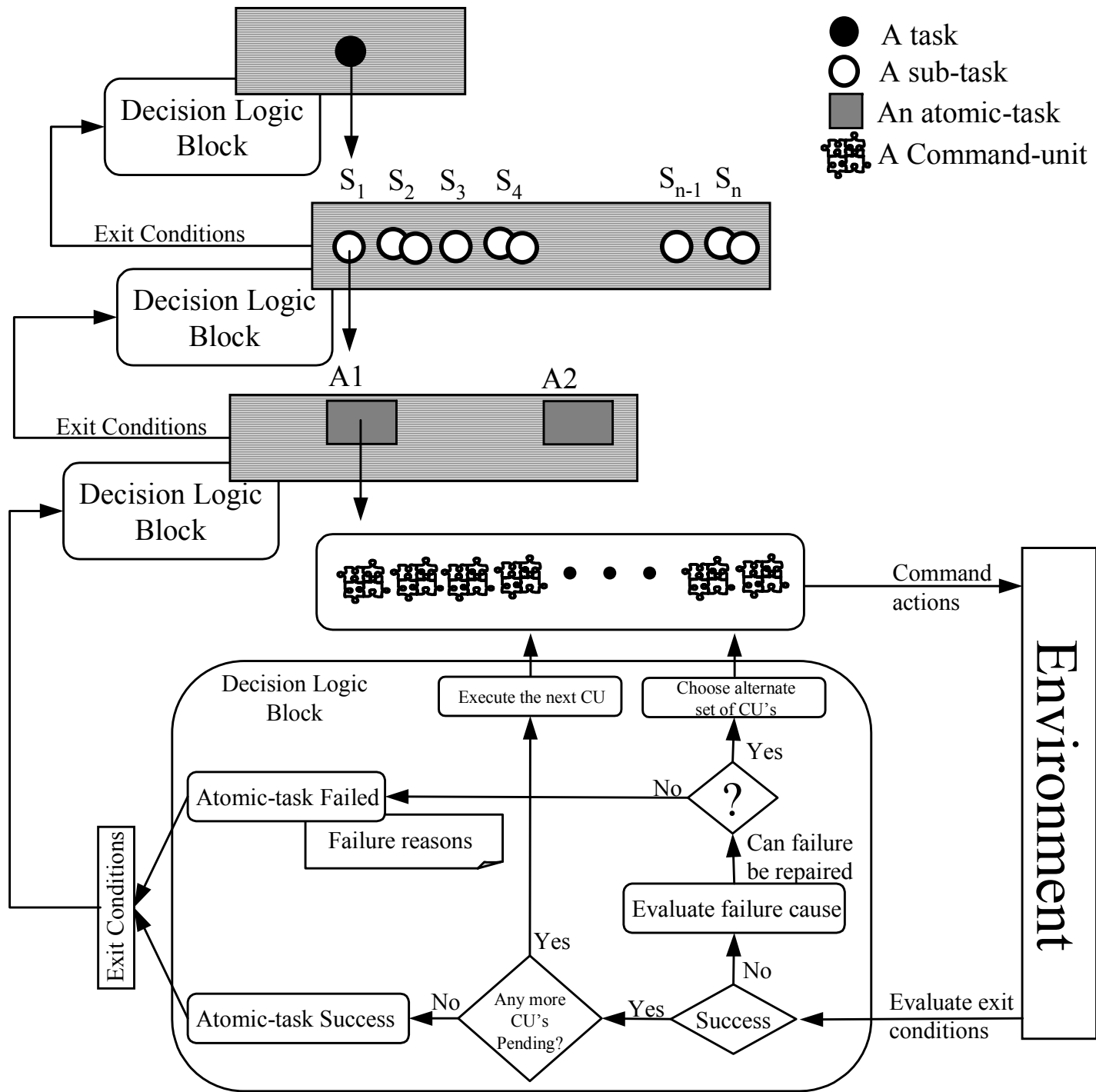
T3 Step-Climb Using a Rule-Based Controller



Reactive Behaviors

Reactive behaviors are induced via:

1. Localization thread
 - Compares expected positions to actual sensors data and makes correction to GPS and odometry as needed
2. Awareness thread
 - Interacts with the execution thread based on safety assessments of the environment
3. Logic within the execution thread
 - Scripted adaptive behaviors
 - Exit conditions at each level of the hierarchy determine branching to pre-defined actions or to re-plan events

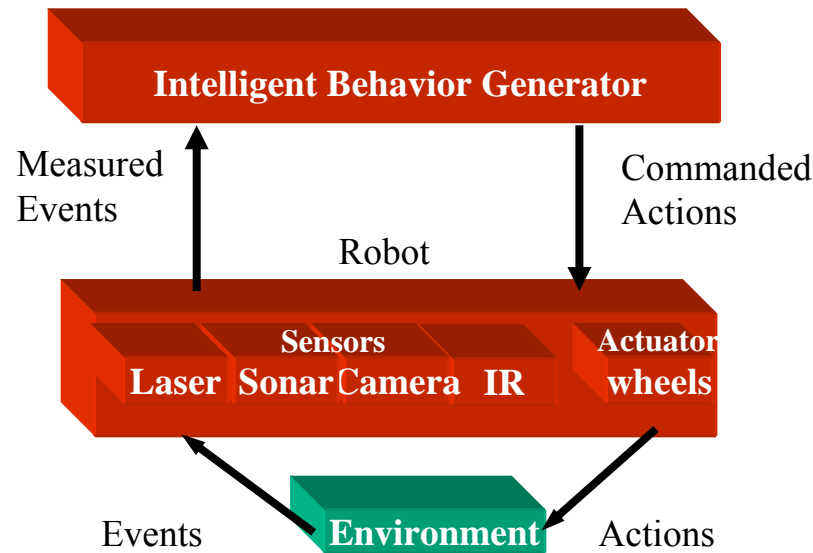


More on Architectures

- New Paradigms for Architectures
 - Formal approach to grammar-based strategies

DEDS Approach

- The mobile robot behavior generator can be interpreted as a discrete-event dynamic system (DEDS)



- In this interpretation commands and events are symbols in an alphabet associated with a (regular) language
- This formalism can be used for synthesis of scripts
- Other suggested approaches for synthesis include Petri nets and recent results on controller design for finite state machine model matching

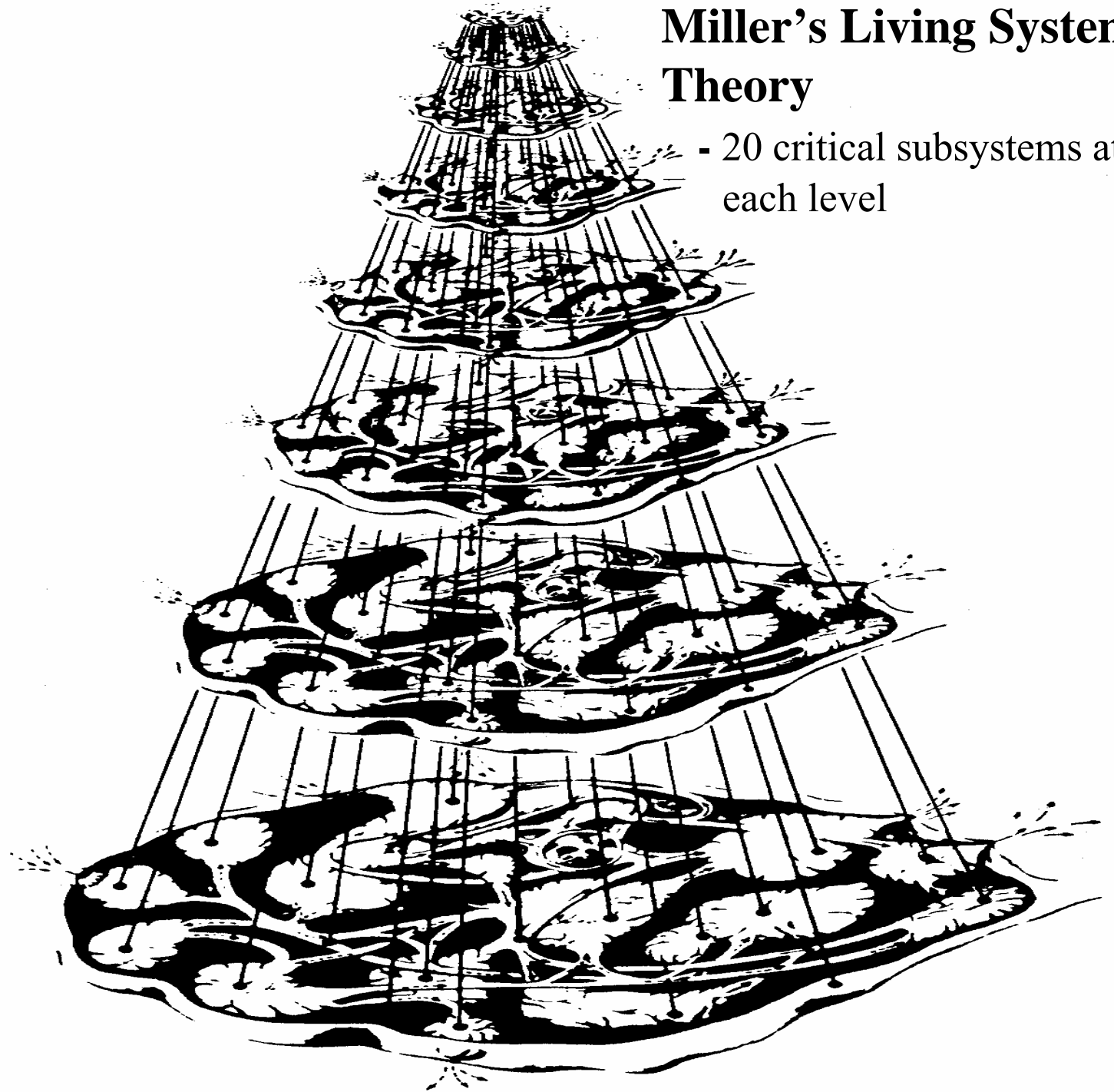
More on Architectures

- New Paradigms for Architectures
 - Formal approach to grammar-based strategies
 - Role of memory in intelligence needs to be understood
 - Architectures must allow “growth” of knowledge (e.g., evolution of language)
- Other biologically-inspired modeling and architecture ideas:
 - Swarms (e.g., bacterial foraging, ants/bees, etc. – more later)
 - Multi-agent systems
 - Self-organization, complex adaptive system (e.g., membrane formation)
 - Architecture of cell functionality (e.g., genomics, proteomics)
 - Cybernetic views (e.g., Miller’s Living System Theory)

Miller's Living System Theory

- 20 critical subsystems at each level

Cell
Organ
Organism
Group
Organization
Community
Society
Supranational System



Outline

- What is an Unmanned System?
- Unmanned system components
 - Motion and locomotion
 - Electro-mechanical
 - Sensors
 - Electronics and computational hardware
- Unmanned system architectures
 - Multi-resolution approach
 - Software Architecture
 - Reaction, adaptation, and learning via high-level feedback

Workshop Schedule

Time	Topic	Presenter
8:30-8:45	Introductions and Course Overview	Moore
8:45-10:00	Unmanned Systems: Components and Architectures	Moore
10:00-10:30	Break	
10:30-12:30	Control Algorithms for Unmanned Systems	Berkemeier
12:30-1:30	Lunch	
1:30-3:30	Intelligent Behavior Generation	Flann
3:30-4:00	Break	
4:00-5:15	Future Directions in Unmanned Systems	All
5:15-5:30	Wrap-up	Moore

Workshop SC841

Unmanned Systems 101

Unmanned Control Systems

Matthew Berkemeier, Autonomous Solutions, Inc.

SPIE 2007 Security & Defense Symposium
Orlando Florida

9 April 2007

Autonomous Solutions Overview

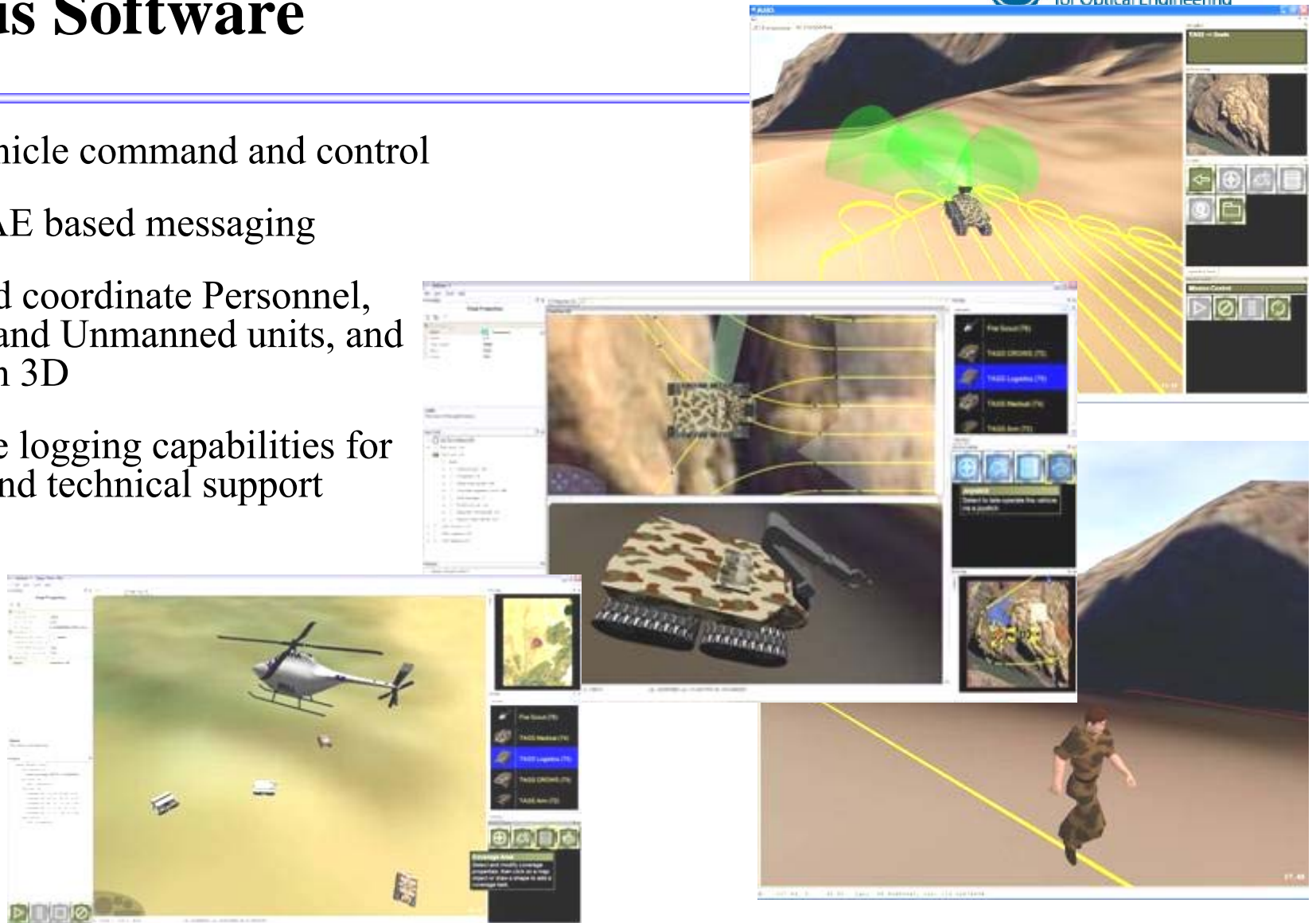
New Location, March, 2007

100 acres



Mobius Software

- Multi-vehicle command and control
- JAUS/SAE based messaging
- Track and coordinate Personnel, Manned and Unmanned units, and sensors in 3D
- Extensive logging capabilities for records and technical support



Chaos

- Developed by ASI
- Funded by TARDEC
- Ultra Mobile Reconnaissance Robot
- Applications:
 - Military
 - Homeland Security
 - Physical Security
 - Police
 - Fire
 - EMS



High Speed Unmanned Targets

▪ Customers:

- Raytheon \ Army (Fort Polk)
- JT3 \ Airforce (NTTR)
- PEO STRI



- Fielded at MOUT sites
- Operated from 40 Miles away
- 50 MPH top speed
- Multiple units deployed

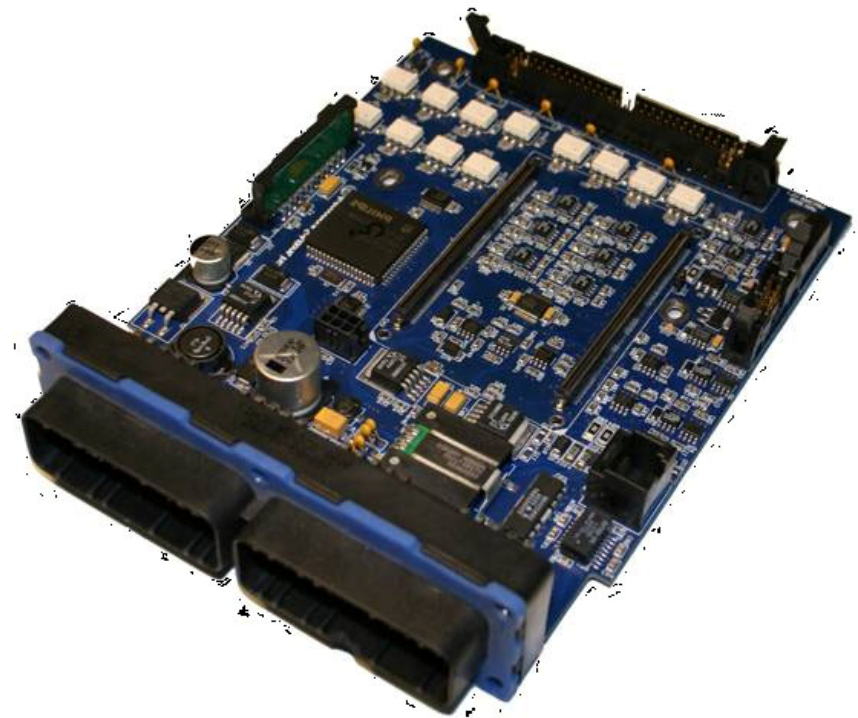


Unmanned Control Systems

- Processing Hardware
- Low-Level Servos
- Open-Loop Approaches for Vehicle Motion
- Intelligent Behaviors for a High-Mobility Vehicle
- Closed-Loop Path Tracking

Processing Hardware

- Even teleoperated vehicles need some processing power on board.
- Equipment designed for rugged environments should be used.
- Microcontrollers with many I/O pins and timers, etc. are appropriate.
- Real-time OS/kernels should be considered so that events (e.g, range sensor) receive a deterministic response.



Low-Level Servos

- DC motor motion controller
 - Very nice, low-level DC motor controllers are available.
 - Configurable and programmable.
 - Sophisticated firmware supports many different modes, like position, velocity, torque control.
 - PC software allows selection of gains, testing with different periodic inputs, graphing of output, logging output, etc.
 - Different feedback devices supported, like optical encoders, potentiometers, etc.
 - Controlled by CAN bus, which leads to simplified wiring: Actuator generally needs only power and CAN signals (4 wires total).

Low-Level Servos (cont.)

- DC motor motion controller (cont.)
 - General purpose inputs
 - Useful for vehicle drive, steering, robotic arm joints, etc.
 - Accept a wide range of power supply voltages. Supply a wide range of currents.
 - Can be difficult to understand capabilities/ease-of-use without actually buying one and setting it up.
 - Low-level servos shouldn't be built from scratch!

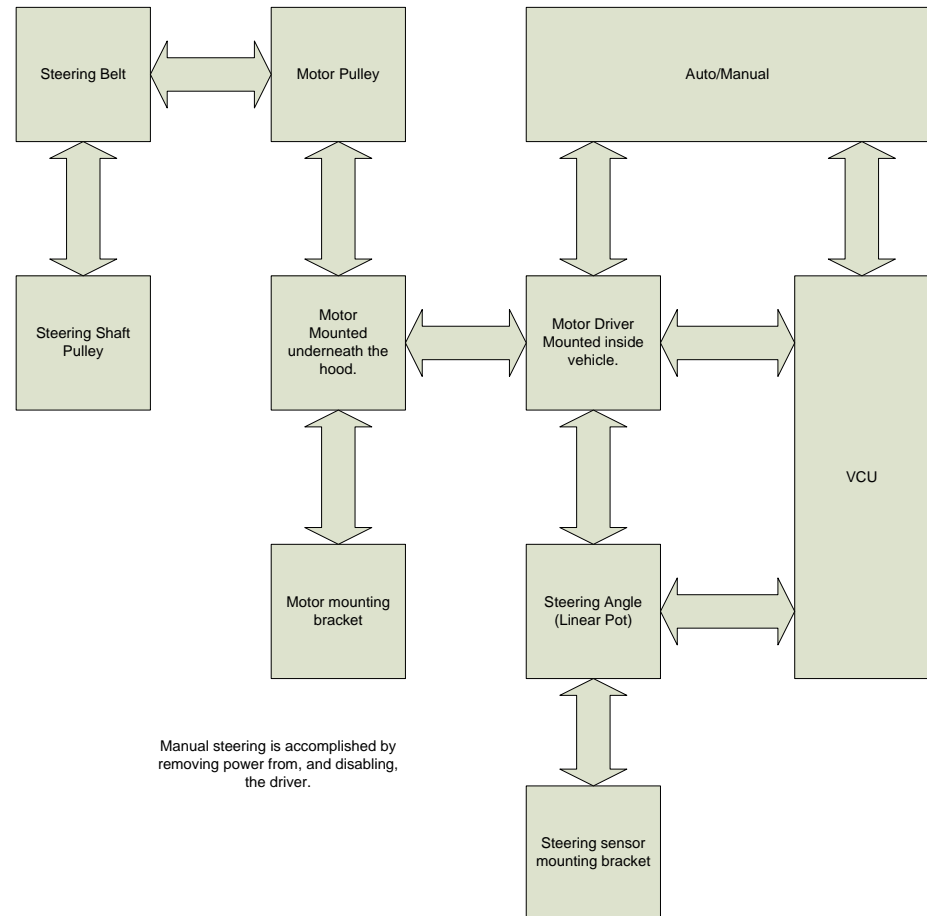
Low-Level Servos (cont.)

- Copley
 - One popular brand. Seen on some NC machine tools.
 - 102 x 69 x 25 mm (4 x 2.7 x 1 in) for this particular model.
- Elmo
 - Another popular brand.
 - 150 x 25.4 x 105 mm (5.9 x 1 x 4.1 in).
- AMC is another brand. There are undoubtedly others.



Steering Servo

- Automobile hydraulic steering system is not well-behaved and does seem to require a custom solution for high-performance.
- Many nonlinearities, and much testing and calibration is necessary.
 - Rate limit
 - Deadband
 - Sinusoidal frequency response

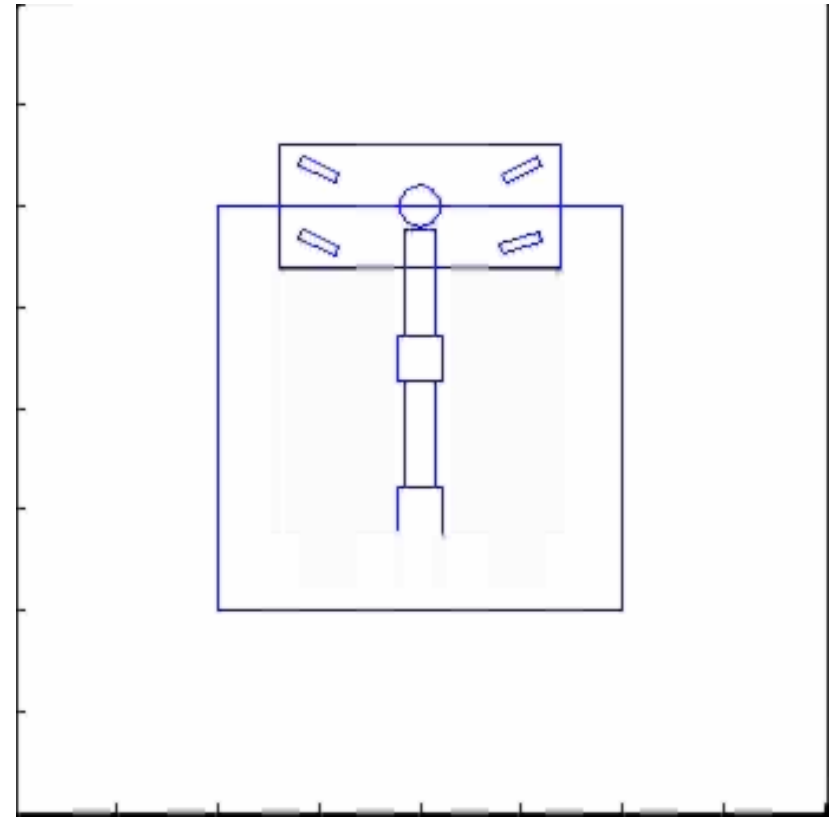
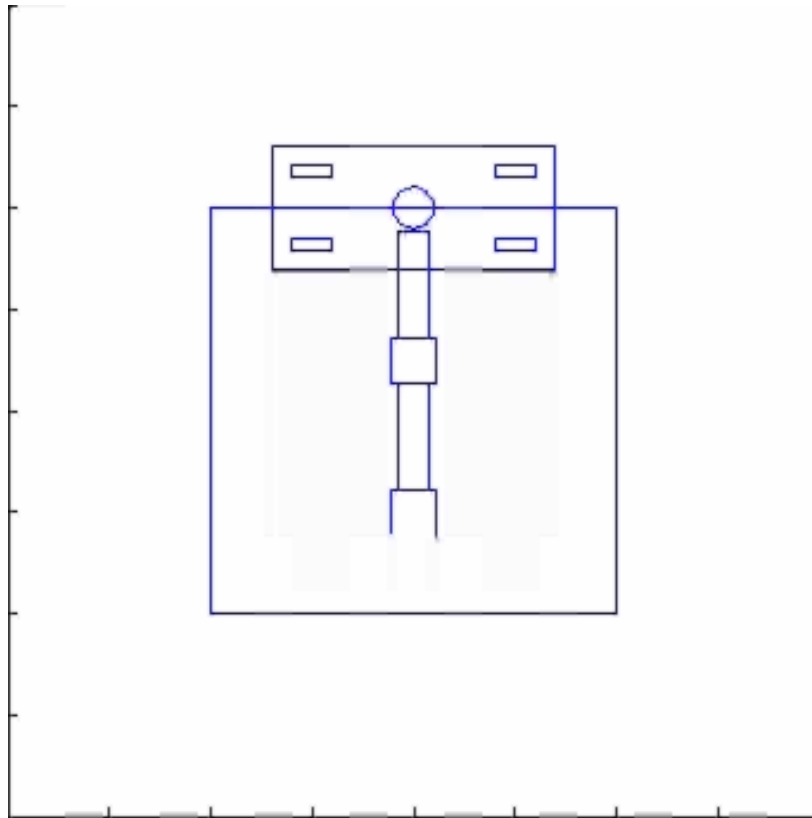


Open-Loop Approaches

- Kinematics
- Ackermann Vehicle Model
- Functionally Omni-Directional Vehicle
 - Omni mode
 - Travel mode
- Omni-Directional Vehicle

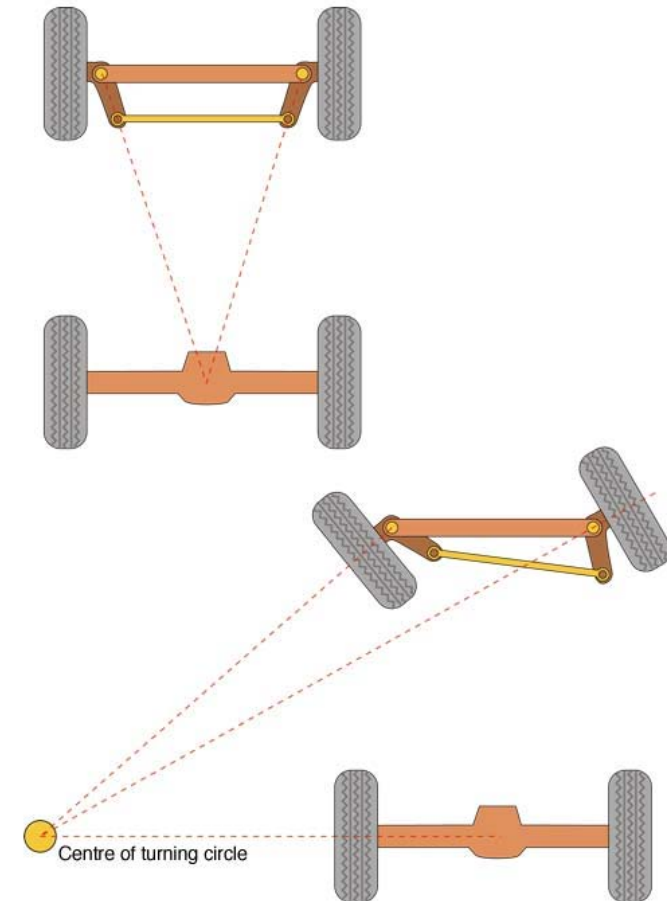
Kinematics

- Motivating Animations



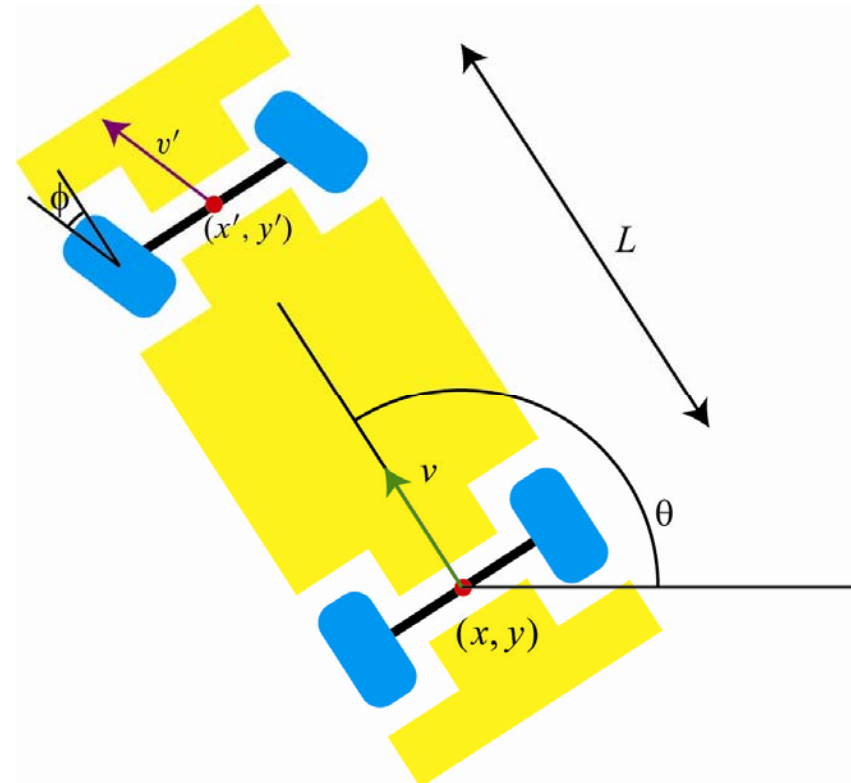
Ackermann Vehicle Model

- Refers to a conventionally-steered, car-like vehicle
- Rudolf Ackermann patented a mechanism to orient the front wheels correctly so lines drawn through all wheel centers intersect at a common point.
- This common point is the center of the turning circle.
- Development is illustrative of general method for deriving kinematics of other types of vehicles



Ackermann Vehicle Model

- To simplify analysis, the “bicycle” model is often used.
- In this case, the 2 front wheels are approximated by a single wheel.
- We often still draw a four-wheeled vehicle, but the bicycle idea is implicit.
- For rear-wheel drive we put the origin at center of the rear axle.



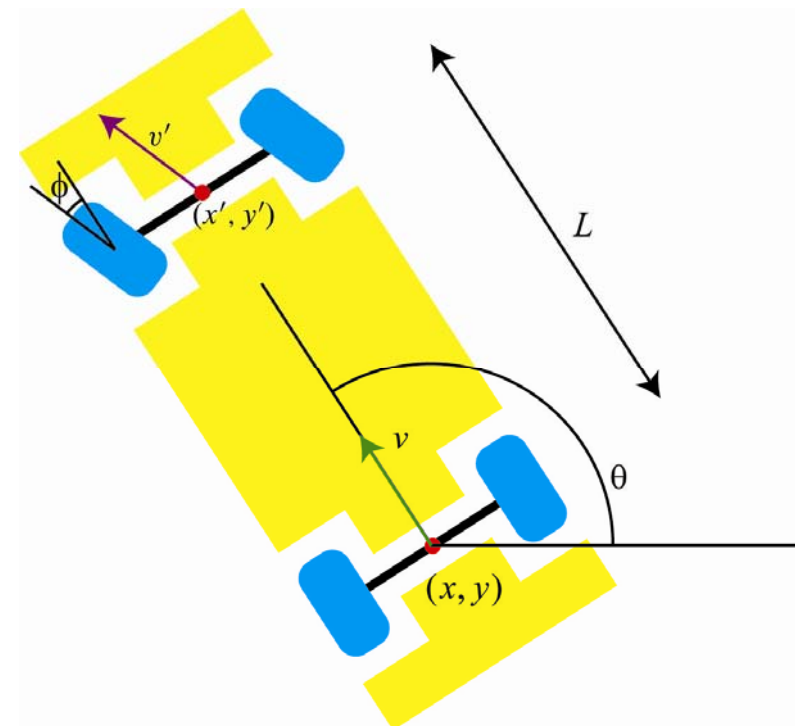
Ackermann Vehicle Model

- The bicycle model can be represented in state-space form.
- v is the forward speed of the rear wheels. v' is the speed of front wheels.
- θ is vehicle orientation (counter-clockwise with respect to positive x -axis).
- L is wheelbase.
- ϕ is angle of front wheels (counter-clockwise with respect to forward-directed vehicle centerline).
- The control inputs are v and ϕ for a rear-wheel drive model.

$$\dot{x} = v \cos \theta$$

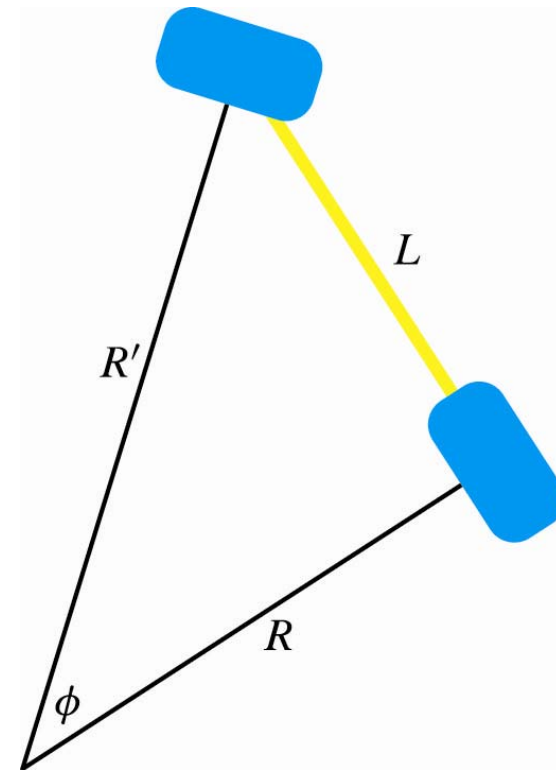
$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = \frac{v}{L} \tan \phi$$



Ackermann Vehicle Model

- It is often useful to have expressions for the turning radius, R , or if measured from the front wheel, R' .
- $R = L / \tan \phi$
- $R' = L / \sin \phi$



Ackermann Vehicle Model

- For a front-wheel drive model we are interested in the speed of the front axle.
- We can derive equations similar to the rear-wheel drive model.

$$\dot{x} = v' \cos \theta$$

$$\dot{y} = v' \sin \theta$$

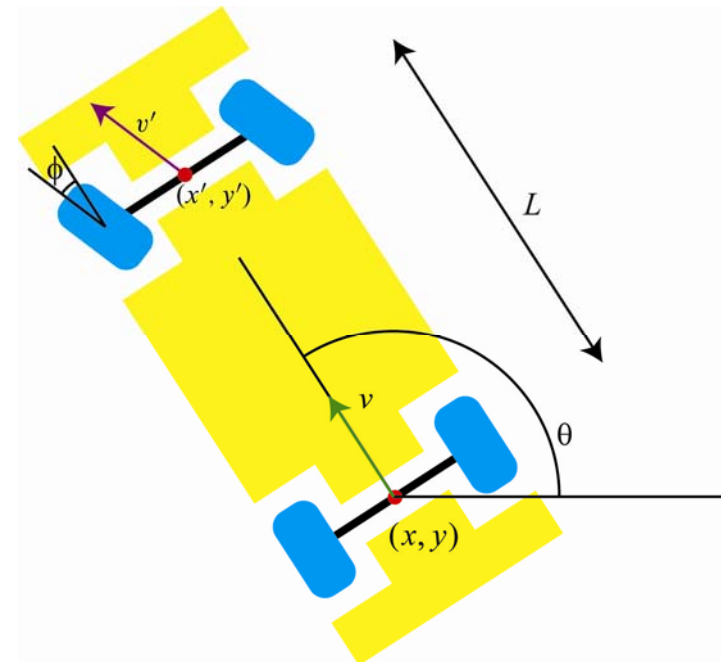
$$\dot{\theta} = \frac{v}{L} \sin \phi$$

- We can also write our equations from the point of view of a point on the front axle.

$$\dot{x}' = v' \cos(\theta + \phi)$$

$$\dot{y}' = v' \sin(\theta + \phi)$$

$$\dot{\theta} = \frac{v'}{L} \sin \phi$$



High Mobility Vehicle

- Chaos is a high-mobility vehicle with 4 track arms.
- For flat terrain, it can be modeled approximately as a differential drive robot.
- v_L, v_R are the speeds of the left and right tracks.
- w is the width between the tracks.
- θ is the heading.

$$\dot{x} = \frac{v_L + v_R}{2} \cos \theta$$

$$\dot{y} = \frac{v_L + v_R}{2} \sin \theta$$

$$\dot{\theta} = \frac{v_R - v_L}{w}$$



Functionally Omni-Directional Vehicle

- Swabby is a functionally omni-directional vehicle, in that its arm base can be rotated and commanded in x and y directions.
- It is tele-operated, and most control is open-loop.
- One of its purposes is to swab surfaces of cars so that explosive chemical residues can be detected, hence its name.
- Swabby is essentially a differentially-drive robot.
- 2 wheelchair motors drive the rear wheels.
- The front wheel is not drive but is actively steered. This provides more precise motions than if we had used a caster.

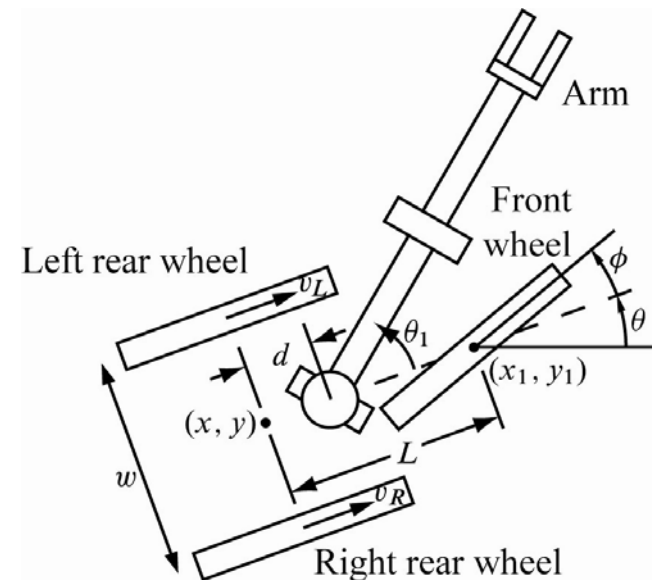


Functionally Omni-Directional Vehicle

$$\dot{x} = \frac{v_L + v_R}{2} \cos \theta$$

$$\dot{y} = \frac{v_L + v_R}{2} \sin \theta$$

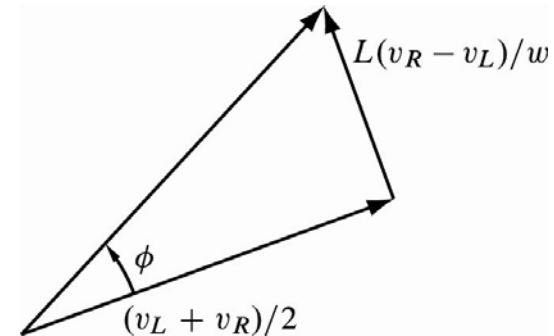
$$\dot{\theta} = \frac{v_R - v_L}{w}$$



Functionally Omni-Directional Vehicle

$$\phi = \tan^{-1} \left(\frac{2L}{w} \frac{v_R - v_L}{v_R + v_L} \right)$$

- To derive front steering wheel angle, simple geometrical reasoning is used.
- Forward speed is $(v_L + v_R)/2$.
- Lateral speed is $(v_R - v_L)L/w$.



Omni Mode

- Operator moves a joystick to specify the 2 axes of translation and 1 axis of rotation of the arm base.
- Let (\dot{a}, \dot{b}) be the operator's commanded arm-base velocity.
- Let $\dot{\theta}_{1u}$ be the user's specified arm-base angular velocity.
- The velocity of the arm base joint is obtained by differentiation of its position.

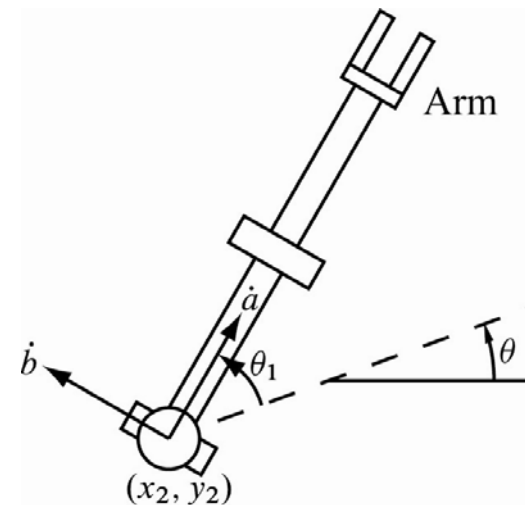
$$\dot{x}_2 = v \cos \theta - d \omega \sin \theta$$

$$\dot{y}_2 = v \sin \theta + d \omega \cos \theta$$

- where

$$v = (v_L + v_R)/2$$

$$\omega = (v_R - v_L)/w$$

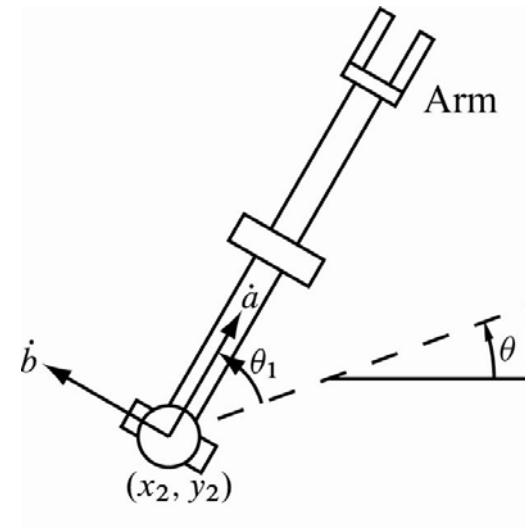


Omni Mode

$$\dot{x}_2 = v \cos \theta - d \omega \sin \theta$$

$$\dot{y}_2 = v \sin \theta + d \omega \cos \theta$$

- This expression for the base velocity is rotated into the vehicle-fixed frame.
- Thus, all operator commands are with respect to the vehicle.
- This makes the most sense when the user is remotely operating Swabby while looking through one of its cameras.

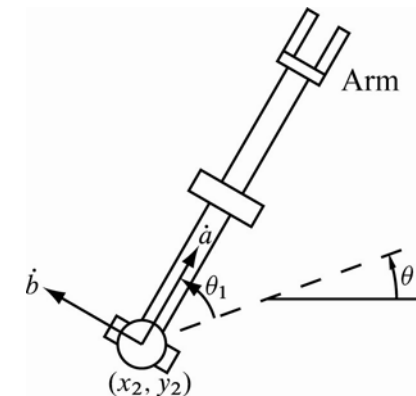
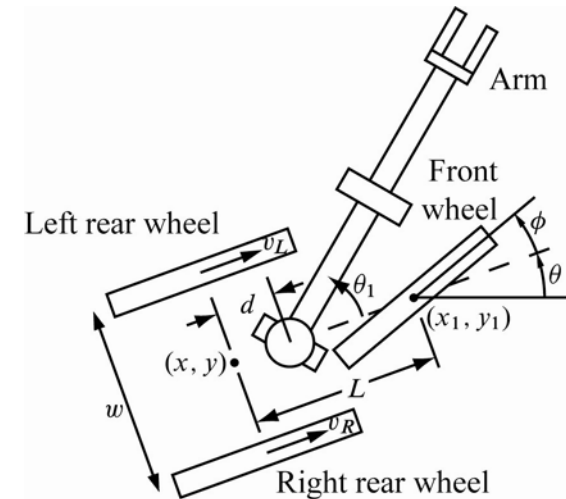


Omni Mode

- The final result, which gives left and right wheel speeds as functions of \dot{a} and \dot{b} is obtained after some algebra.

$$v_L = -\frac{1}{2d} \left((-2\dot{a}d + \dot{b}w) \cos \theta_1 + (2\dot{b}d + \dot{a}w) \sin \theta_1 \right)$$

$$v_R = \frac{1}{2d} \left((2\dot{a}d + \dot{b}w) \cos \theta_1 + (-2\dot{b}d + \dot{a}w) \sin \theta_1 \right)$$



Swabby Video Clips



Swabby
Functionally
Omni-Directional
Robot

Travel Mode

- Swabby also has a travel mode, in which it behaves like a front-wheel drive car.
- Travel mode is used for quickly moving from some point A to another point B , which are somewhat distant.
- Omni mode is intended for more precise movements.
- To find formulas for travel mode, we compare Swabby's kinematic equations with those of a front-wheel-drive car.



Travel Mode

- Front-wheel-drive car equations have been previously given

$$\dot{x}' = v' \cos(\theta + \phi)$$

$$\dot{y}' = v' \sin(\theta + \phi)$$

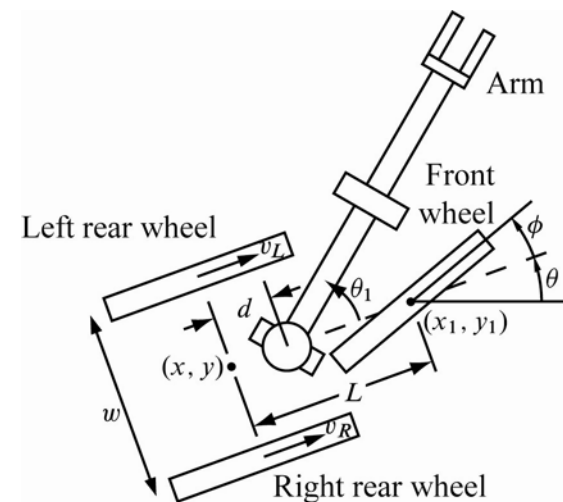
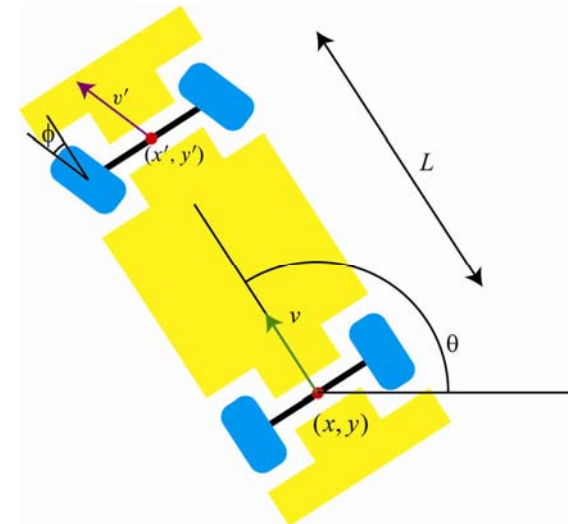
$$\dot{\theta} = \frac{v}{L} \sin \phi$$

- Swabby's kinematic equations must be modified to use (x', y') (front wheel) instead of (x, y) (rear wheel).

$$\dot{x}' = \frac{v_R + v_L}{2} \cos \theta - L \frac{v_R - v_L}{w} \sin \theta$$

$$\dot{y}' = \frac{v_R + v_L}{2} \sin \theta + L \frac{v_R - v_L}{w} \cos \theta$$

$$\dot{\theta} = \frac{v_R - v_L}{w}$$

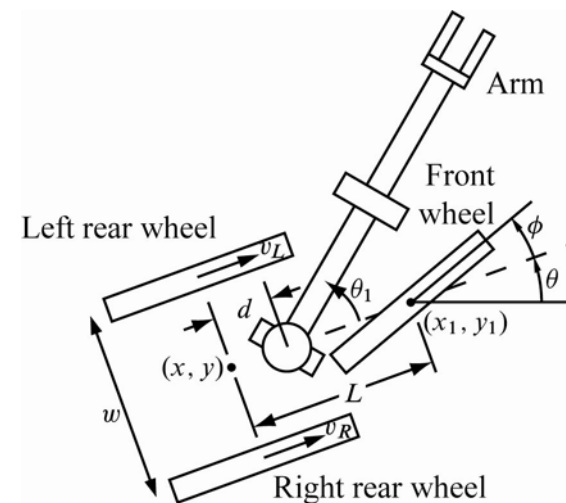
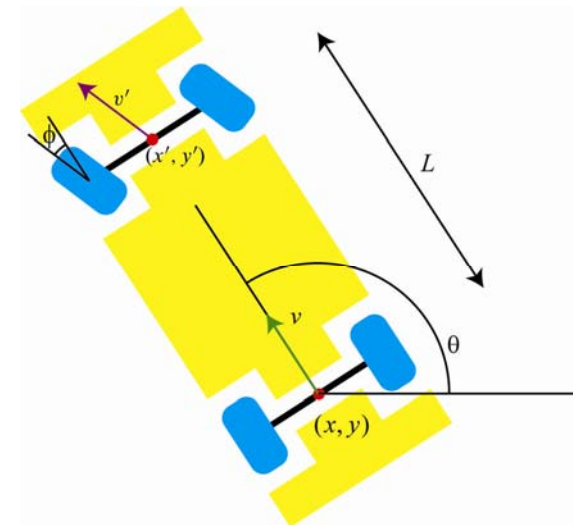


Travel Mode

- To find the desired result, equate the first 2 equations, and solve for v_R, v_L .
- This makes Swabby drive like a car!

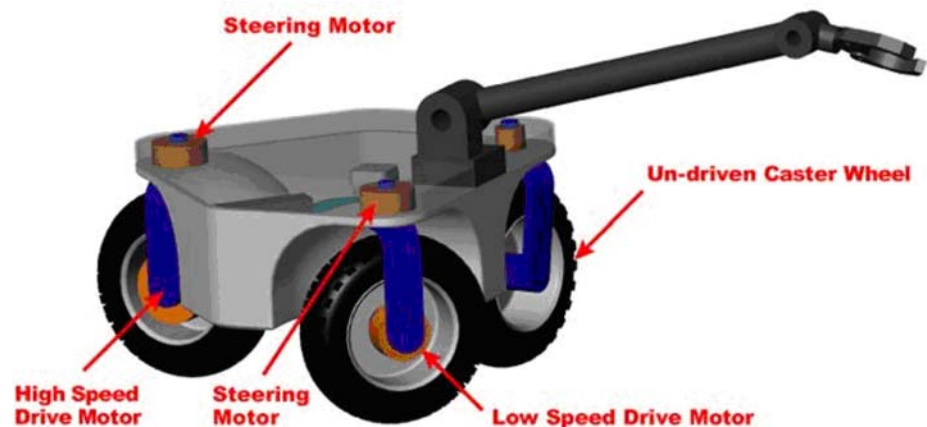
$$v_R = v' \left(\cos \phi + \frac{w}{2L} \sin \phi \right)$$

$$v_L = v' \left(\cos \phi - \frac{w}{2L} \sin \phi \right)$$



Omni-Directional Vehicle

- Taz is a vehicle under development with 2 actively drive and steered wheels.
- The third wheel is a caster.
- Unlike Swabby, it will be fully omni-directional.
- Taz has 4 inputs: 2 wheel speeds and 2 wheel angles.
- The user will specify only 3 quantities: a velocity vector (2 parameters) and a rotational speed.
- The wheel speeds and angles must be such that they would not tend to “stretch” or “compress” the base.
- This provides a unique solution to the kinematic equations.



Omni-Directional Vehicle

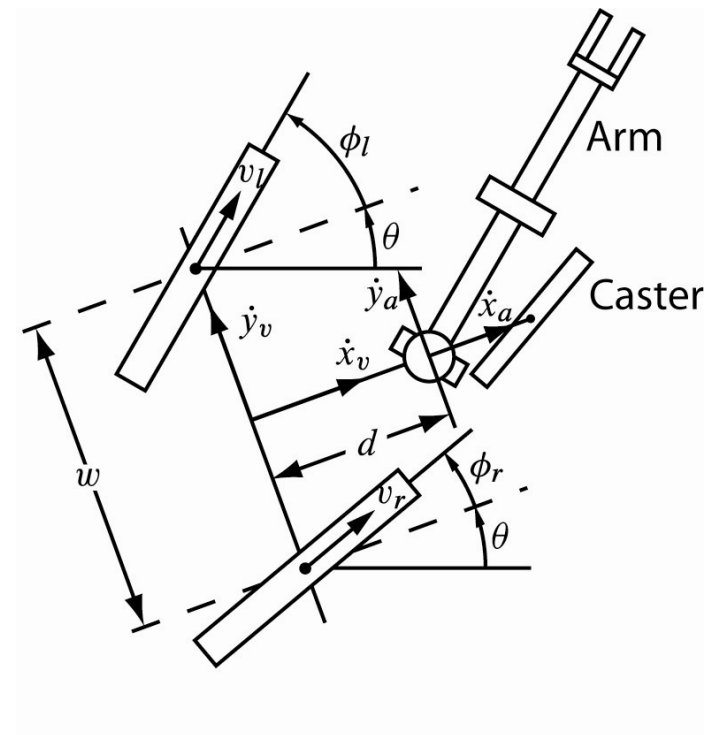
- The equations are written this time with respect to a frame in the vehicle.
- Like with Swabby, we expect the user will be driving the vehicle while looking through a camera fixed to the vehicle.

$$\dot{x}_v = \frac{v_r \cos \phi_r + v_l \cos \phi_l}{2}$$

$$\dot{y}_v = \frac{v_r \sin \phi_r + v_l \sin \phi_l}{2}$$

$$\dot{\theta} = \frac{v_r \cos \phi_r - v_l \cos \phi_l}{w}$$

$$0 = v_r \sin \phi_r - v_l \sin \phi_l$$



Omni-Directional Vehicle

- These 4 equations can be solved for the 4 inputs: v_r , v_l , ϕ_r , ϕ_l in terms of user-specified \dot{x}_v , \dot{y}_v , $\dot{\theta}$.
- This gives the user the ability to specify the motion of the point midway between the wheels.
- Alternatively, new equations for the location of the point where the arm is located could be derived, and this process repeated.
- It may make more sense to drive the arm base, rather than the point midway between the wheels.

$$\dot{x}_v = \frac{v_r \cos \phi_r + v_l \cos \phi_l}{2}$$

$$\dot{y}_v = \frac{v_r \sin \phi_r + v_l \sin \phi_l}{2}$$

$$\dot{\theta} = \frac{v_r \cos \phi_r - v_l \cos \phi_l}{w}$$

$$0 = v_r \sin \phi_r - v_l \sin \phi_l$$

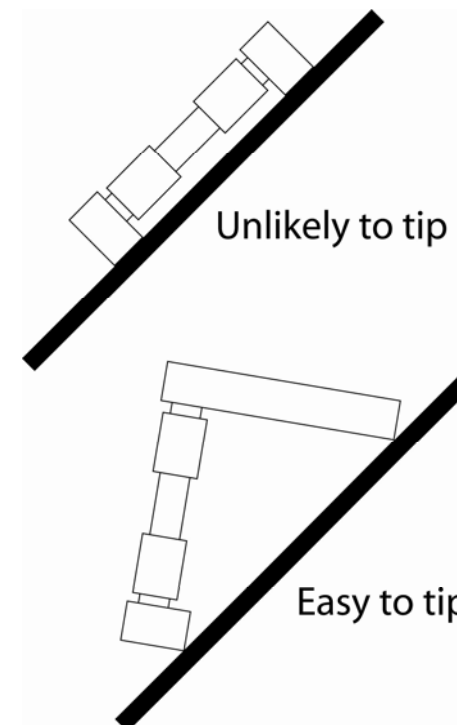
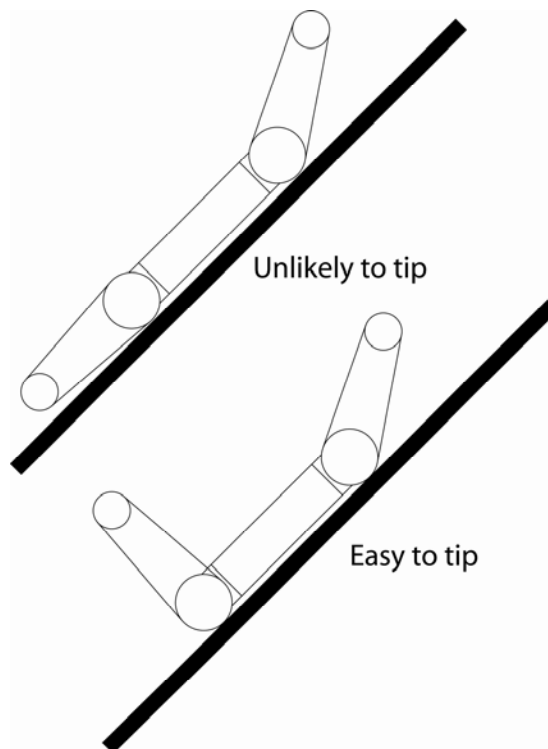
Intelligent Behaviors for a High-Mobility Robot

- The user can specify 4 inputs to Chaos:
 - Forward/reverse drive speed
 - Clockwise/counter-clockwise drive turn rate
 - Speed of front arms
 - Speed of rear arms
- It takes a skilled operator to handle Chaos well.
- We installed an IMU on Chaos to sense linear accelerations and rotation rates.
- With the IMU, we implemented 2 “Intelligent Behaviors”:
 - Anti-Rollover
 - Heading stabilization

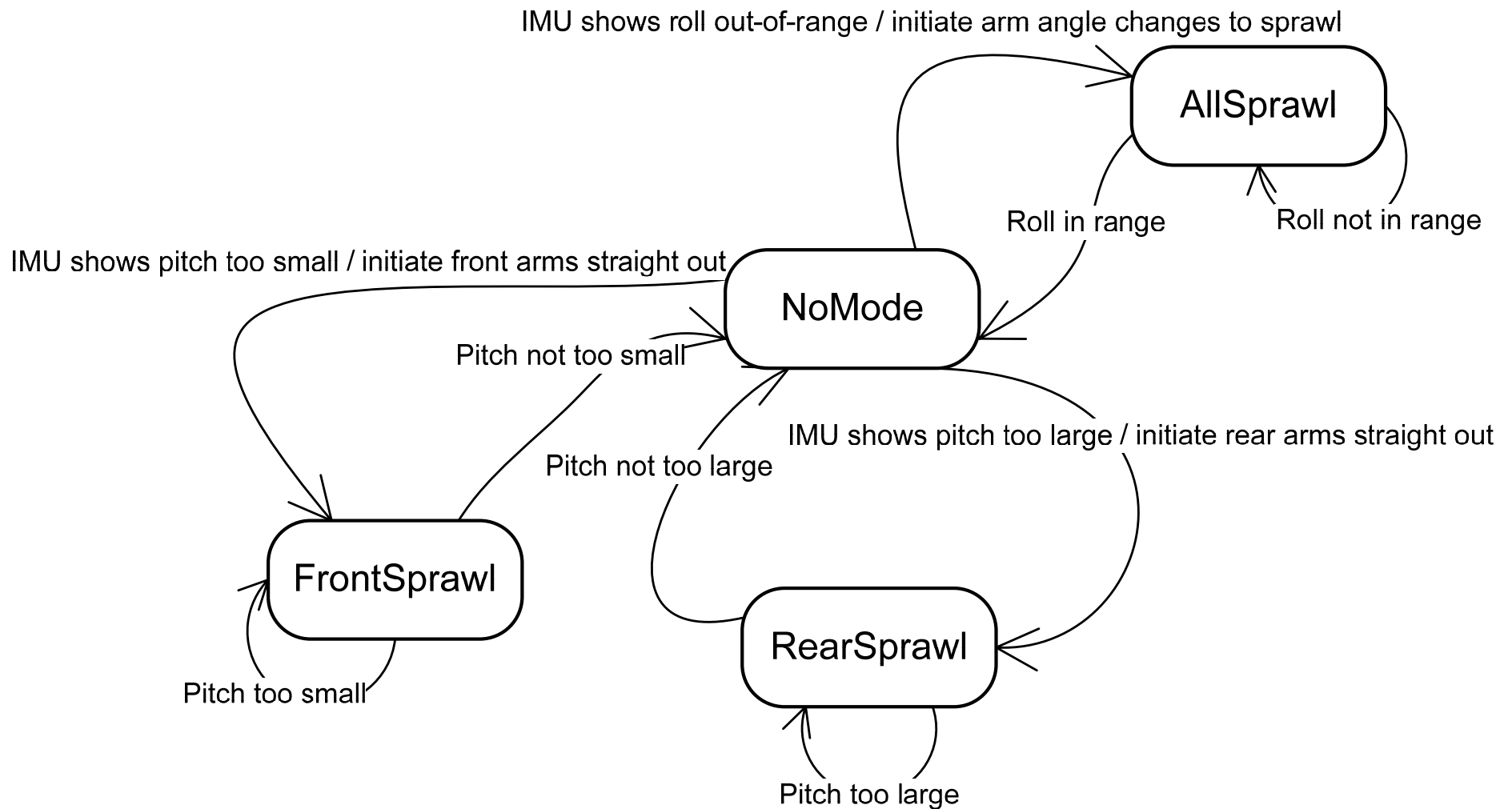


Anti-Rollover Behavior

- When going up a steep hill, it is easy for Chaos to tip over backward.
- However, if rear tracks are positioned straight out, it is difficult to tip.
- Similarly, when traversing a steep hill, it is easy for Chaos to roll over.
- If all tracks are positioned out straight, the risk is minimized.



Anti-Rollover Statechart



Anti-Rollover Test Results

- When climbing a steep hill with loose dirt without anti-rollover behavior enabled, it was easy to tip Chaos over.
- With the anti-rollover behavior, Chaos could not be tipped!
- Some difficulty with different behaviors “fighting” each other:
 - Anti-collision (highest priority)
 - Anti-rollover
 - User commands (lowest priority)

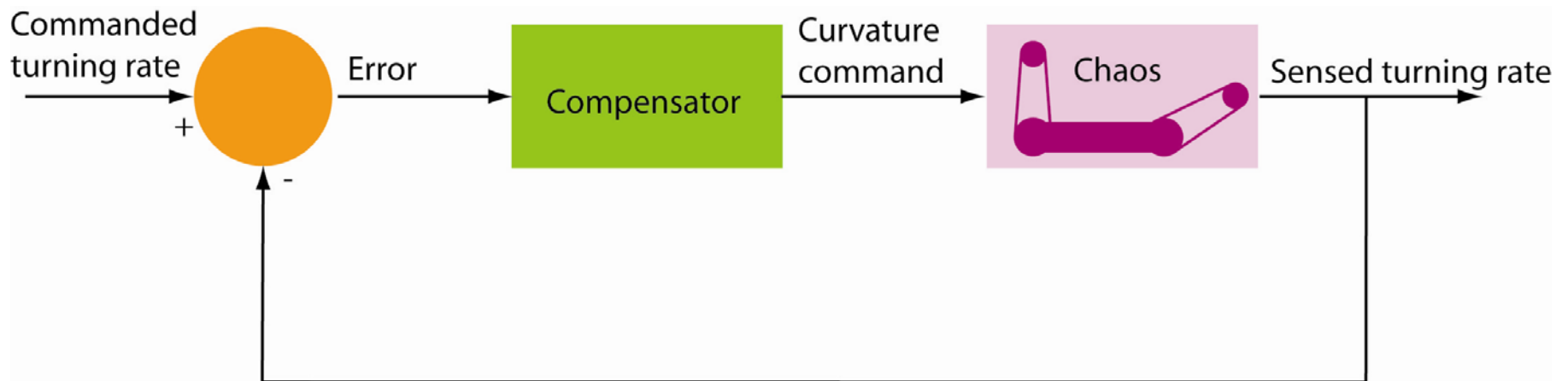


Anti-Rollover Video



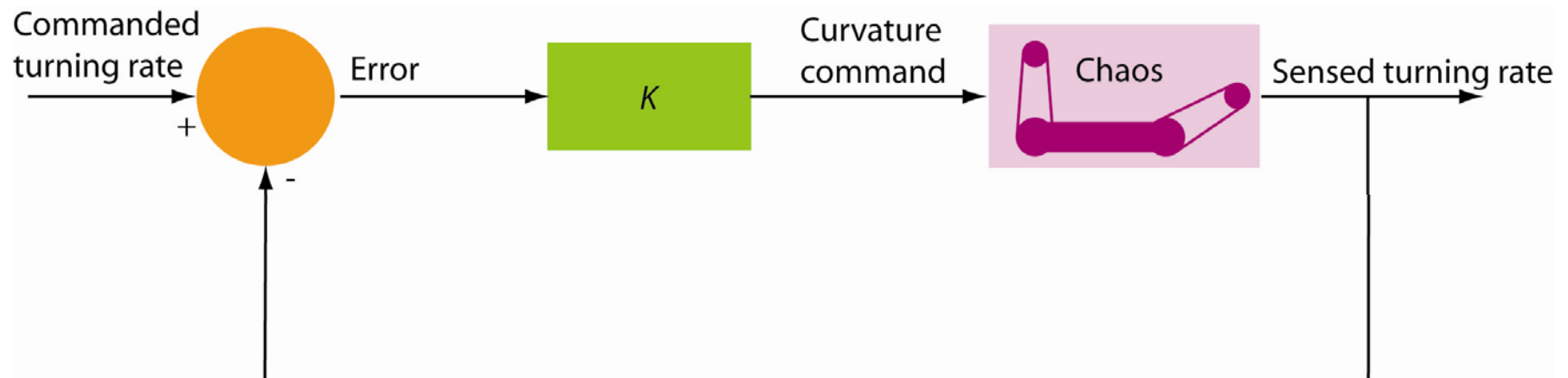
Heading Stabilization

- When Chaos is being driven on rough terrain, it tends to veer off course from time to time.
- The user must constantly be monitoring the heading.
- With the IMU it is possible to assist the user with angular rate control.
- Under this control scheme, Chaos rotates at the rate commanded by the user and resists disturbances.



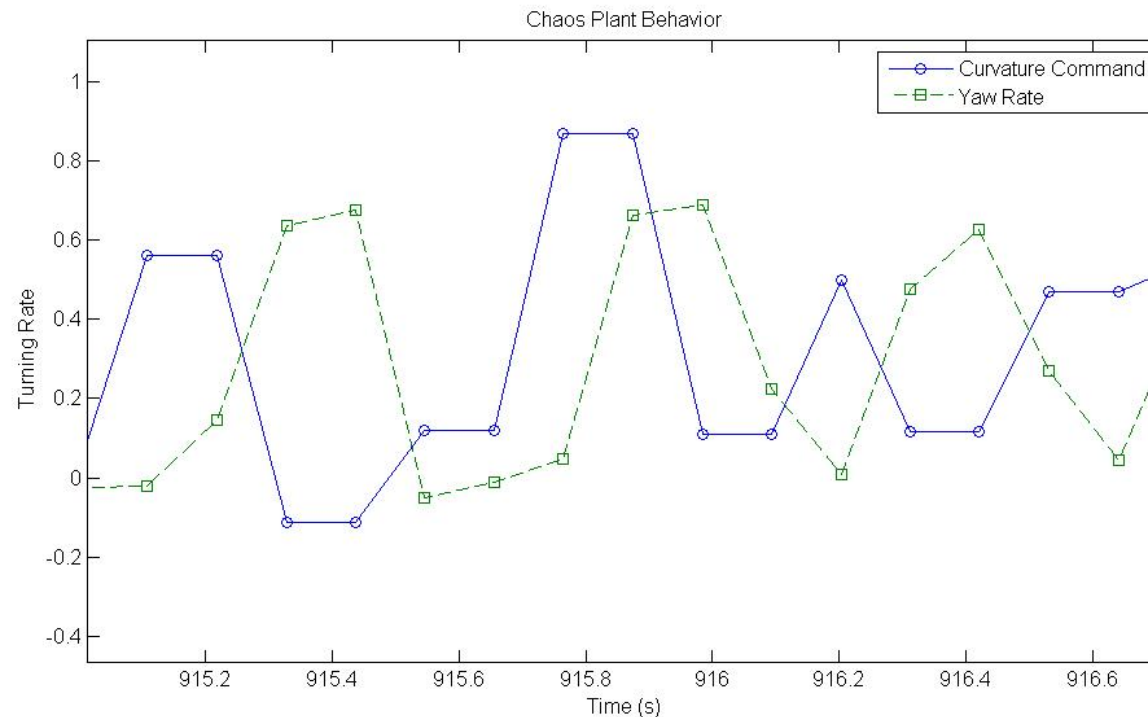
Heading Stabilization

- For the first attempt, a simple gain was used for the compensator.
- As the gain was increased, the system went unstable.
- An unstable Chaos is not a good thing!
- It was not possible to make the gain large enough to achieve good performance.



Heading Stabilization

- In order to use a slightly more sophisticated compensator, it was necessary to attempt a system identification.
- A pseudo-random input was used to excite the Chaos dynamics.



Heading Stabilization

- A second-order plant model was assumed with unknown coefficients:

$$G(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

- A least-squares fit with 201 data points was used.
- The data and unknown coefficients can be put into the form

$$y(N) = C(N)x(N) + \varepsilon(N)$$

with $y(N)$ a vector representing the output, $C(N)$ a matrix containing output and input data, $x(N)$ a vector representing the unknown coefficients, and $\varepsilon(N)$ a vector representing the error.

Heading Stabilization

- The least-squares solution is then given by

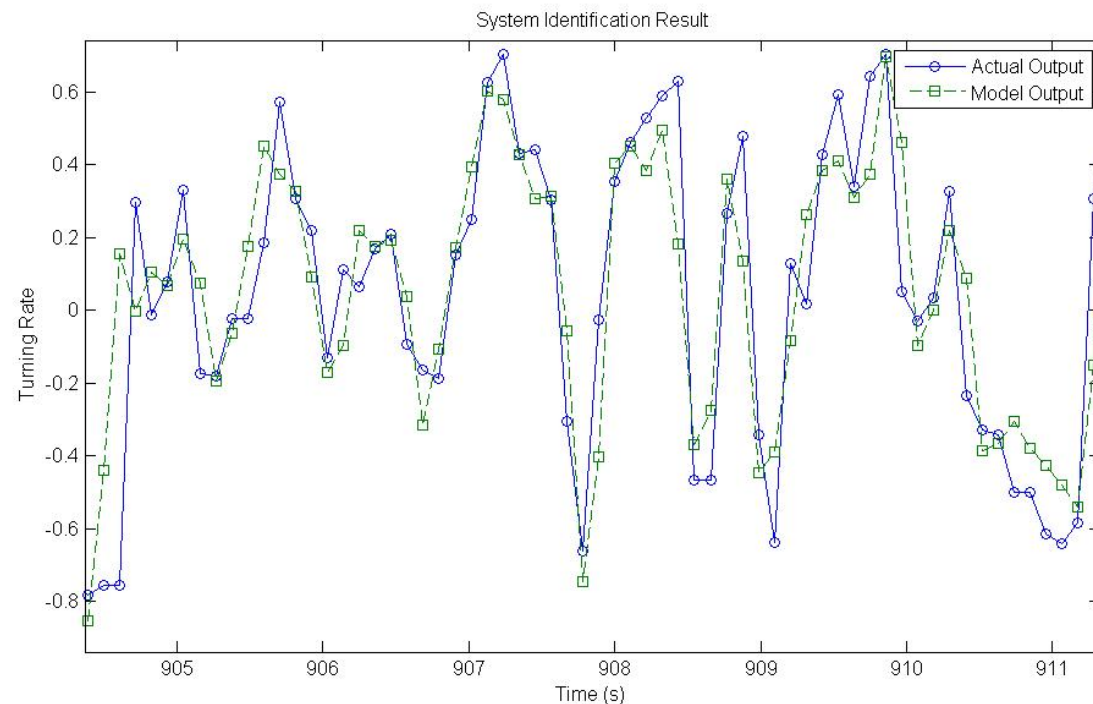
$$\tilde{x}(N) = [C^T(N)C(N)]^{-1} C^T(N)y(N)$$

- Our result was

$$G(z) = \frac{-0.04165 + 0.2692z^{-1} + 0.3548z^{-2}}{1 - 0.3085z^{-1} + 0.0001579z^{-2}}$$

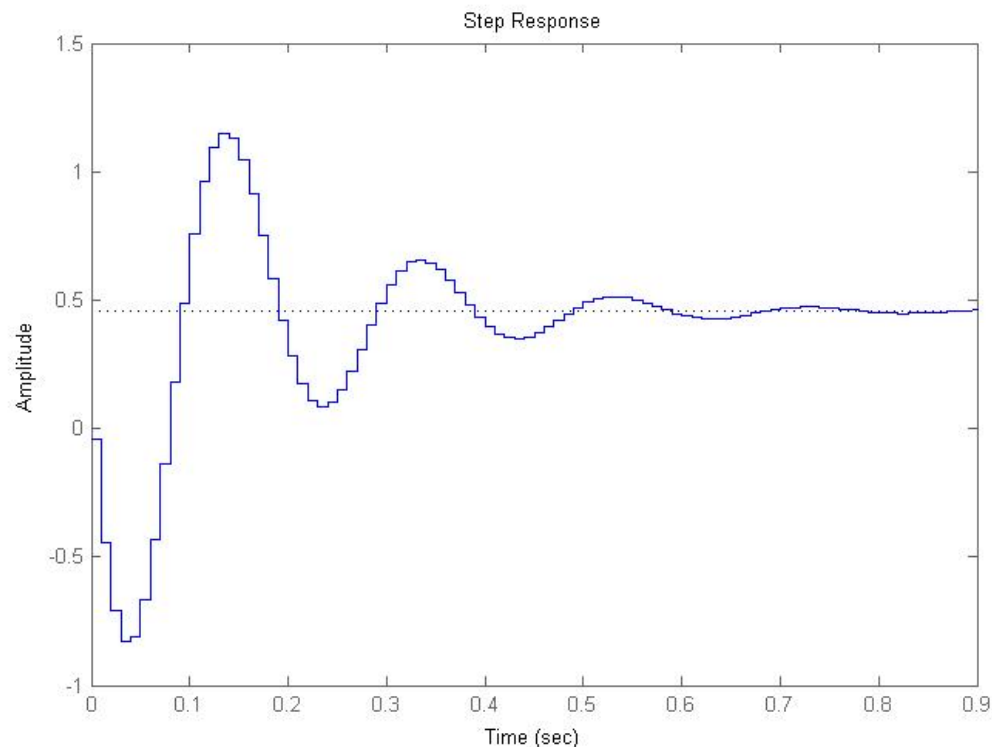
Heading Stabilization

- Model seemed reasonably good but there was significant deviation.
- Unmodeled (nonlinear) behavior:
 - Deadband due to track design
 - Not-quite-periodic control loop time
 - Track slippage
 - Gyro sensor noise



Heading Stabilization

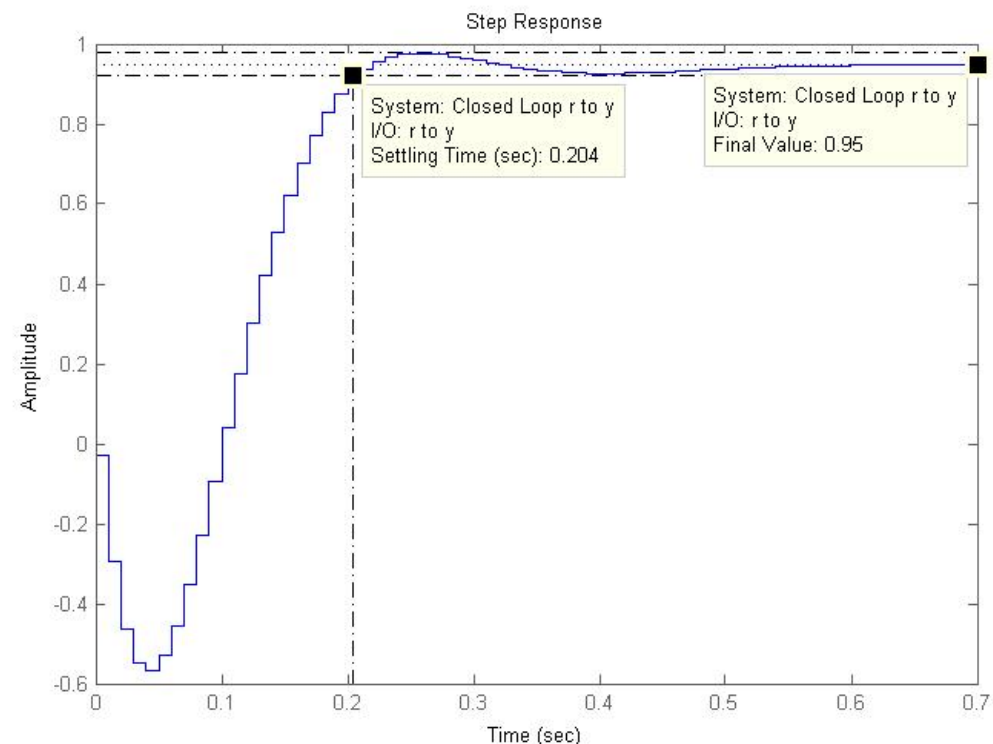
- When we plotted a root locus of our model, we found that the system went unstable with a gain of about 1.25.
- This was very similar to what was observed experimentally.
- With a gain of 1, a step response produced a large steady-state error.



Heading Stabilization

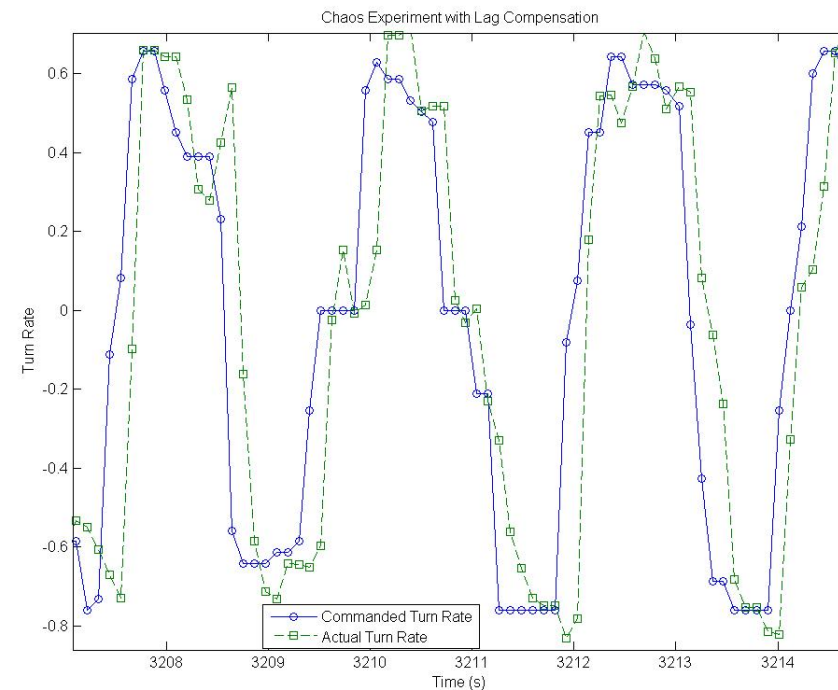
- Since a simple gain could not provide adequate performance, a first-order compensator design was performed.
- Following standard design techniques, the following lag filter and closed-loop step response were obtained:

$$C(z) = 0.67367 \frac{1 - 0.9071z^{-1}}{1 - 0.9972z^{-1}}$$



Heading Stabilization

- Chaos performed very well with the lag compensator in place.
- The compensator produced slightly better performance than the open-loop system when no disturbances were present.
- However, the compensated system was very robust with respect to disturbances in heading rate.



Heading Stabilization Video



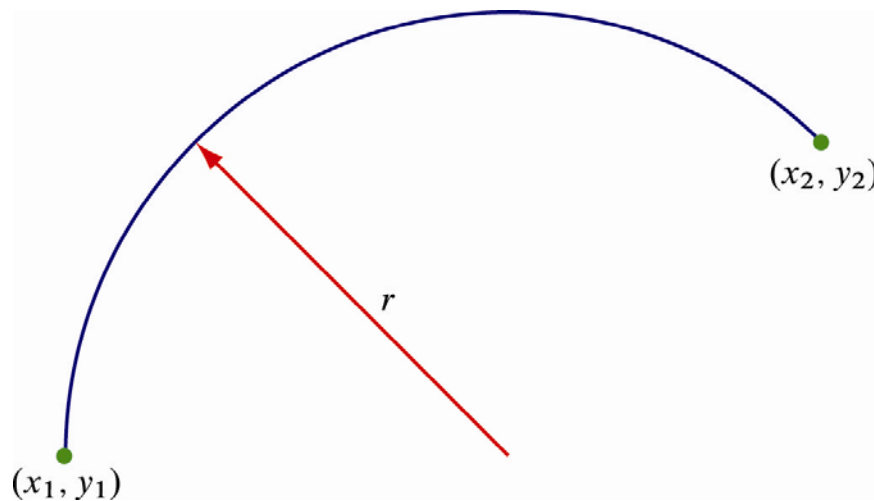
Heading Stabilization
Control

Closed-Loop Path Tracking

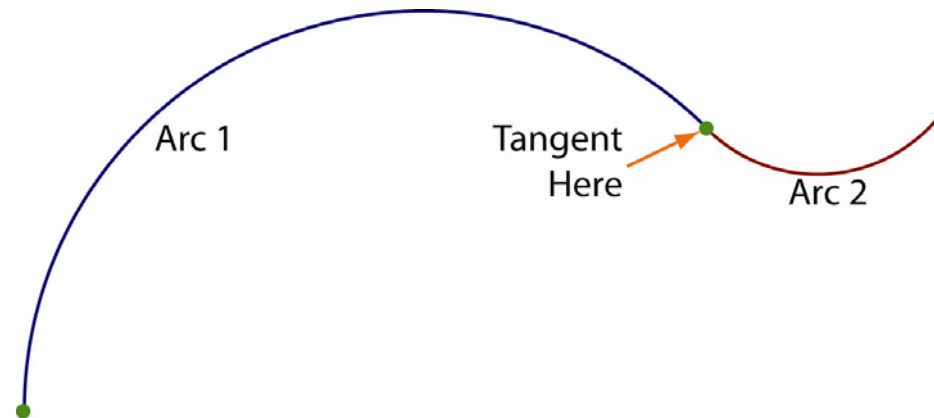
- Path Representation
 - Straight segments and arcs
 - It is convenient for 2D path planning to represent vehicle paths by straight segments (arcs with infinite radius) and arcs.
 - An arc can be represented, for example, by its start point, end point, and radius (5 parameters).
 - A single arc is driveable by an Ackermann vehicle.

Path Representation

- Sign of radius can be used to indicate clockwise or counter-clockwise.
- To this we add a desired speed for the arc.
- Consecutive arcs must be tangent at their common point.
- There is a problem: Paths are not actually drivable!



Path Representation



- Paths are not drivable because they require instantaneous changes in steering angle and speed at the point of tangency.
- This means the closed-loop controllers must deal with step-type disturbances at each tangency point.
- For slow speeds this might be ok.
- At higher speeds there will be problems.
- This can be handled by “pre-biasing” the steering in anticipation of an upcoming arc of different radius.

Path Representation

- Spirals
 - A solution to the instantaneous-steering-angle-change problem is to insert spiral transitions.
 - Clothoid spirals have curvatures which vary linearly with time.
 - Clothoid spirals have no solution in elementary functions and require the introduction of Fresnel sine and cosine integrals.
 - One concern is that the new path may deviate too much from the original path: Original path may clear obstacles but adjusted path may not.

Pure-Pursuit

- Developed at Carnegie-Mellon University Robotics Institute, beginning in 1985.
- Modified versions used in CMU vehicles in 2006 DARPA Grand Challenge.



Pure Pursuit

- The basic method:
 - Basic idea: Calculate curvature that will move a vehicle to some goal position.
 - Goal is some distance l ahead.
 - l is termed the *lookahead distance*.
 - The above is executed periodically.

Pure Pursuit

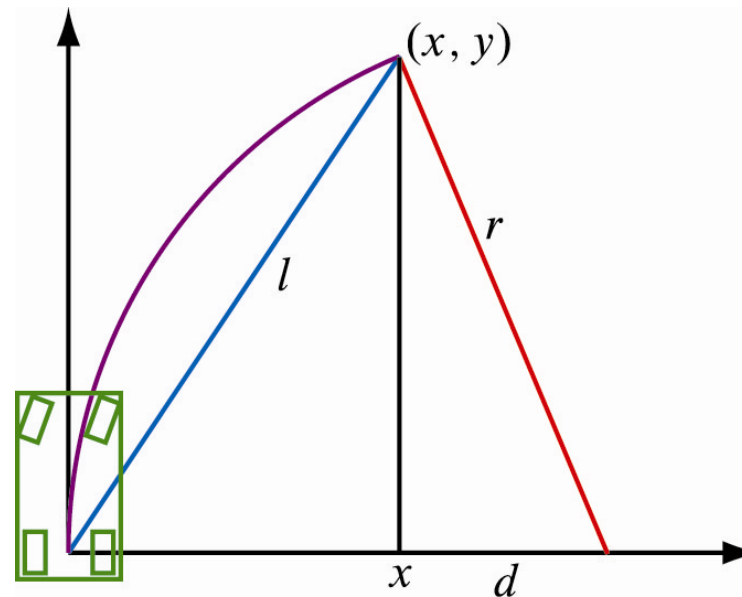
- Goal: (x,y)
- Lookahead distance: l
- Derivation:

$$x^2 + y^2 = l^2$$

$$x + d = r$$

$$d = r - x$$

$$(r - x)^2 + y^2 = r^2$$



$$r^2 - 2rx + x^2 + y^2 = r^2$$

$$2rx = l^2$$

$$r = l^2/2x$$

Pure Pursuit

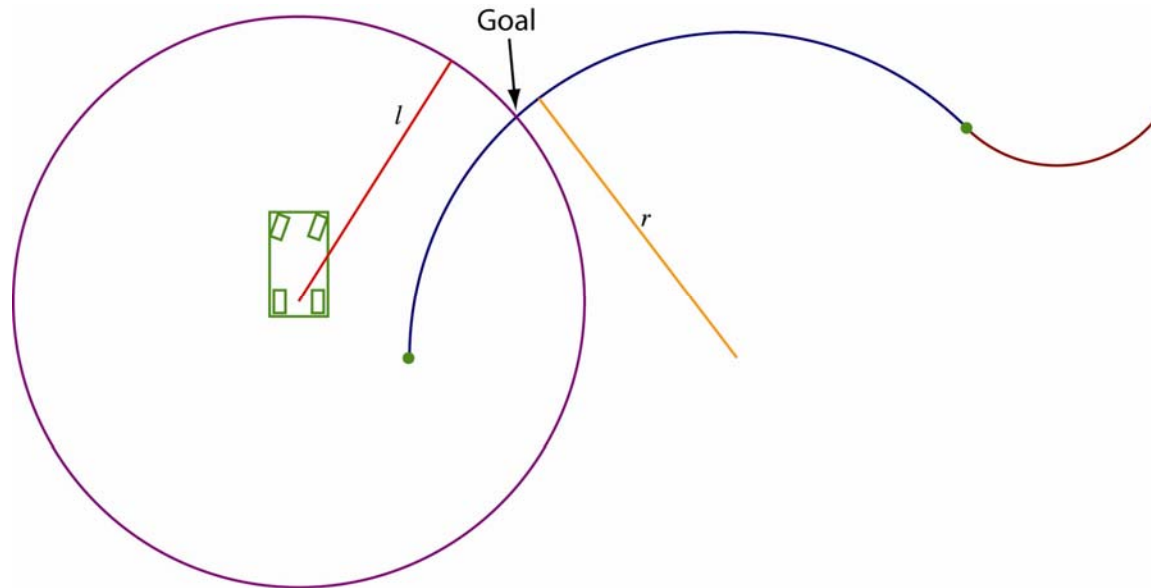
- To calculate steering wheel angle from

$$r = l^2 / (2x)$$

recall previous Ackermann model with $r=L/\tan\phi$.

- L is wheelbase, ϕ is steering wheel angle.
- So, $\phi = \tan^{-1}(L/r)$
- To use Pure Pursuit with arcs, must intersect 2 circles.
- This results in 0, 1, or 2 solutions.

Pure Pursuit



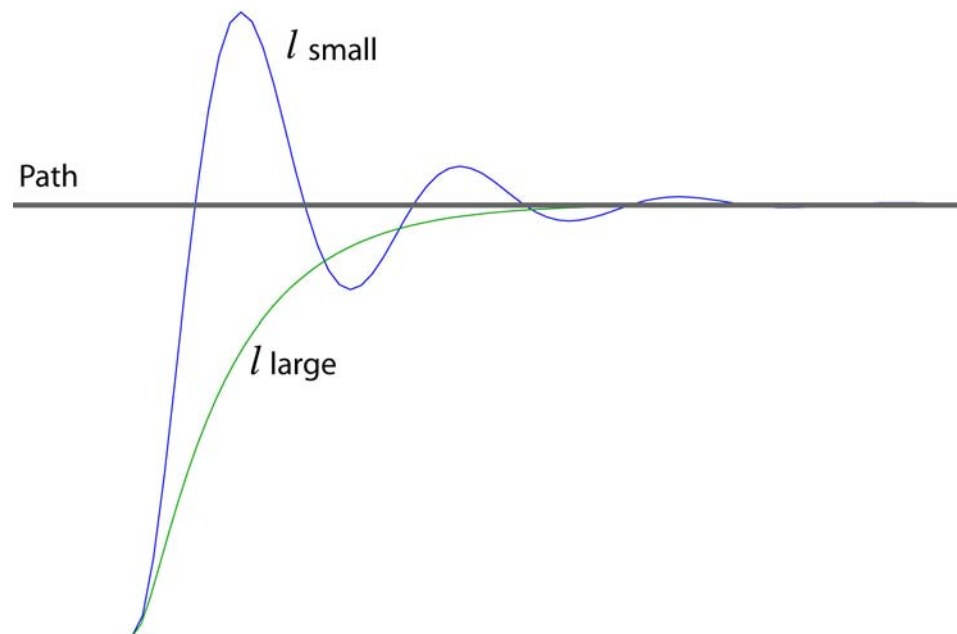
- However, when a full path is considered, there can be more than 2 solutions.
- Some solutions may be behind the vehicle.
- Clearly, there are some non-trivial additional issues involved with this method.

Pure Pursuit

- A 1990 CMU-RI technical report suggests the following algorithm:
 - Determine the current location of the vehicle (GPS, dead reckoning, etc.)
 - Find the path point closest to the vehicle. This is not the goal. This is the point on the path, which is closest to the vehicle.
 - Find the goal point. Since there are potentially multiple solutions, start at previous step and move up path until first solution is found.
 - Calculate the curvature and steering wheel angle.
 - Update the vehicle's position.

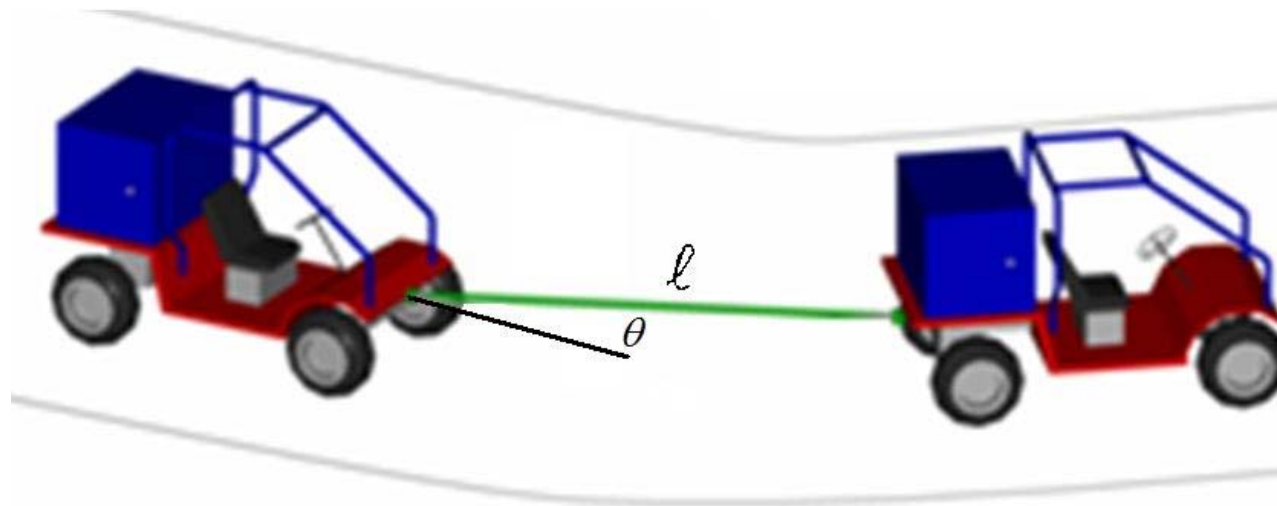
Pure Pursuit

- There is only 1 parameter in the basic Pure Pursuit algorithm: the lookahead distance l .
- Longer lookahead values tend to produce more gradual convergence to the path with less oscillation.
- The longer the lookahead value is chosen, the less “curvy” a path that can be followed.

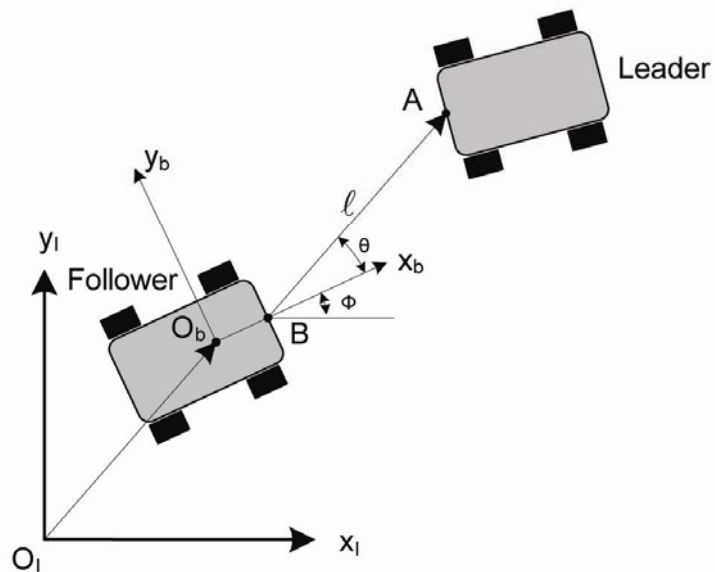
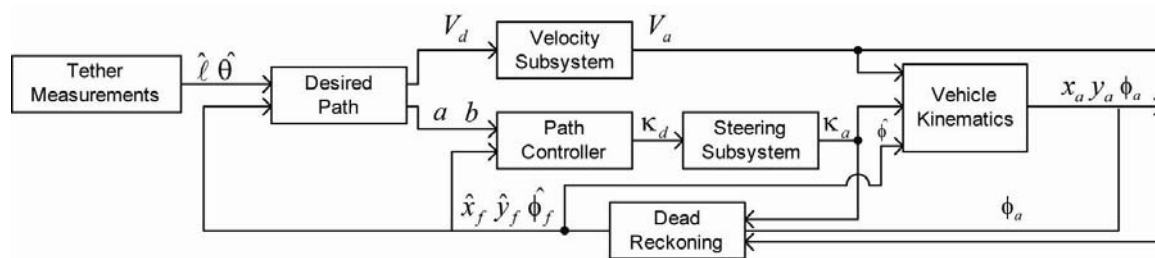


Follow-the-Leader Using a Tether

- In the Robotic Convoy project, a variation of the Pure Pursuit method will be used.
- The lead vehicle is driven by a person, and the second vehicle must automatically follow.
- The follower will record positions of the lead vehicle and then track the path using Pure Pursuit.



Follow-the-Leader Using a Tether



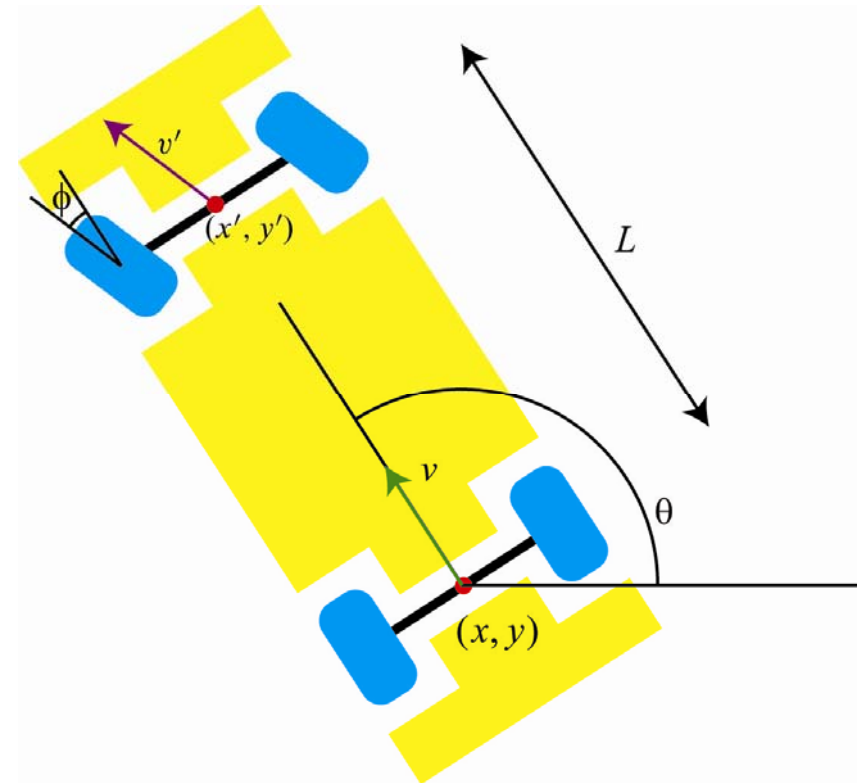
Normal Error

- JT-3
 - At its Nevada Test & Training Range (NTTR), the Air Force runs several Jeep Cherokees automated by ASI for use as target and training vehicles.
 - Path tracking is achieved using a linear controller, whose input is the normal error of the vehicle from the path.



Normal Error

- Normal (orthogonal) error is the distance from the path, normal to the path.
- Heading error is the angular difference between the vehicle orientation and the path orientation. Often it is not used since measurement of the vehicle heading is difficult.



Normal Error

- Linear control design methods can be used by linearizing the Ackermann kinematic model.
- Let κ be the vehicle path curvature (inverse of turn radius).
- From before, $\kappa = \tan \phi / L$.

$$\dot{x} = v \cos \theta$$

$$\dot{y} = v \sin \theta$$

$$\dot{\theta} = v \kappa$$

- We assume the path to be tracked is the x -axis. Then, the normal error is simply y .

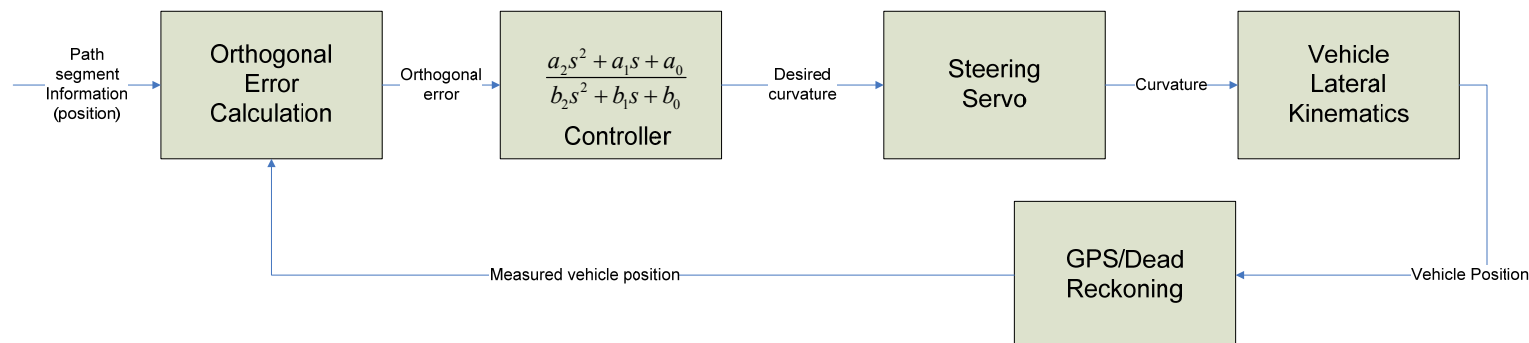
Normal Error

- x no longer has any relevance in our modeling for lateral control.
- We assume $\kappa \approx 0$, $v \neq 0$, $\theta \approx 0$. Then,

$$\dot{y} = v\theta$$

$$\dot{\theta} = v\kappa$$

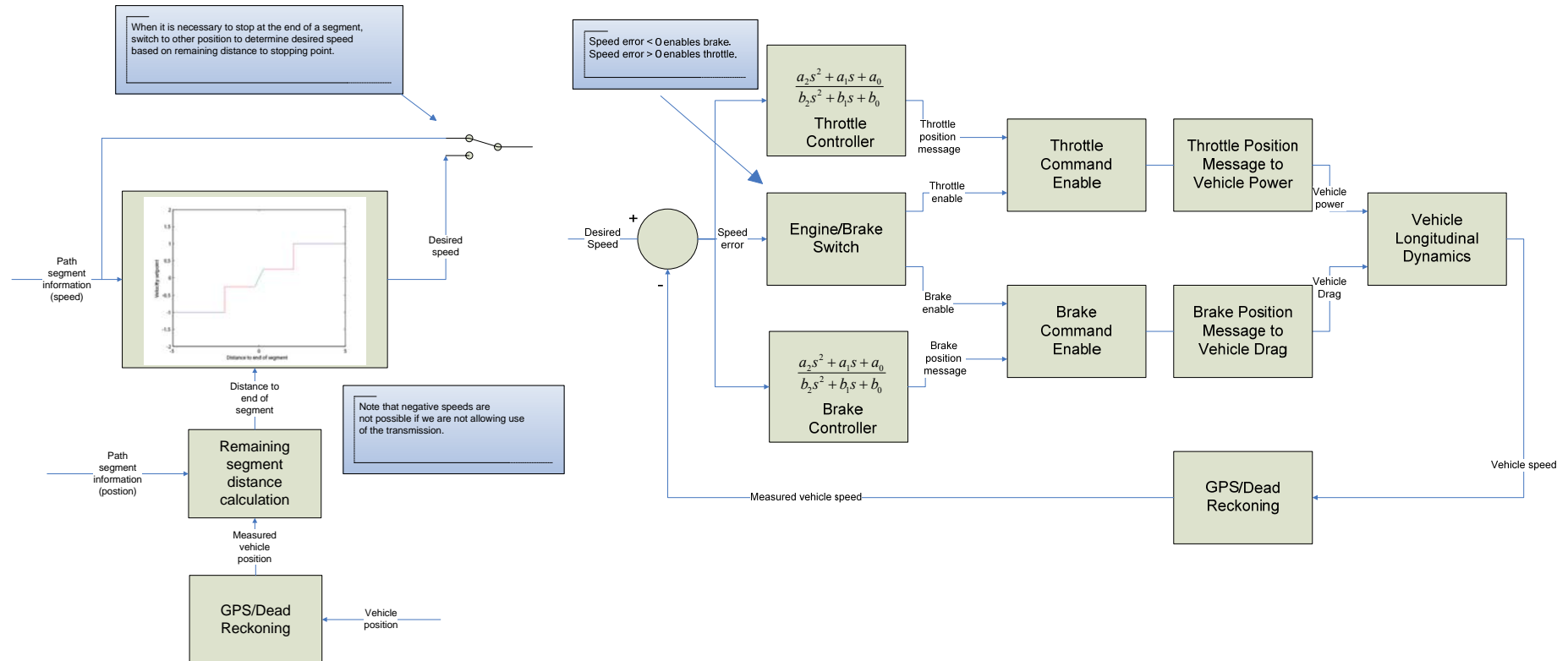
- So, $\ddot{y} = v^2\kappa$, where κ is the input, and we pretend v is a parameter, rather than a time varying input.
- Standard linear control design techniques can be used on $\ddot{y} = v^2\kappa$.



Normal Error

- Speed control will only be mentioned in passing.
- It is complicated slightly because the throttle and the brake usually must both be used.
- If both forward and backward speeds are needed, then the transmission must also be involved.

Speed Control



Summary and Conclusions

- The following topics were covered:
 - Processing hardware
 - Low-level servos
 - Open-loop approaches for vehicle motion
 - Intelligent behaviors for a high-mobility vehicle
 - Closed-loop path tracking
- Open-loop methods work well for tele-operated vehicles since humans close the loop.
- Often open-loop makes the most sense when there isn't a reliable sensor (e.g., GPS is not very reliable).
- Closed-loop path tracking has many challenges but is now well-established.

Workshop Schedule

Time	Topic	Presenter
8:30-8:45	Introductions and Course Overview	Moore
8:45-10:00	Unmanned Systems: Components and Architectures	Moore
10:00-10:30	Break	
10:30-12:30	Control Algorithms for Unmanned Systems	Berkemeier
12:30-1:30	Lunch	
1:30-3:30	Intelligent Behavior Generation	Flann
3:30-4:00	Break	
4:00-5:15	Future Directions in Unmanned Systems	All
5:15-5:30	Wrap-up	Moore

Workshop SC841

Unmanned Systems 101

Intelligent Behavior Generation

Nicholas Flann, Utah State University

SPIE 2007 Security & Defense Symposium
Orlando Florida

9 April 2007

Intelligent Behavior Generation

- Solve Tasks (application independent)
 - Navigate along a route
 - Visit a goal location
 - Tour a series of goal locations
 - Cover a region
- Given
 - Static
 - Map terrains obstacles roads
 - Goal location/regions
 - Dynamic
 - New tasks
 - Map updates due to sensor

Fundamental Approach

- Construct and maintain a representation of the environment
 - Goal locations
 - Possible commands, actions, path segments
- Apply searching over the representation
 - Find safe and near-optimal paths between task locations
 - Find near-optimal ordering of task sequences

Inputs/Interfaces

- Maps with terrain information
- Human/machine task designator
- Situational sensors
 - Proprioception
 - External such as stereo vision, radar, laser
- Actuator/control system takes path commands and executes them on the vehicle

Complexity Factors

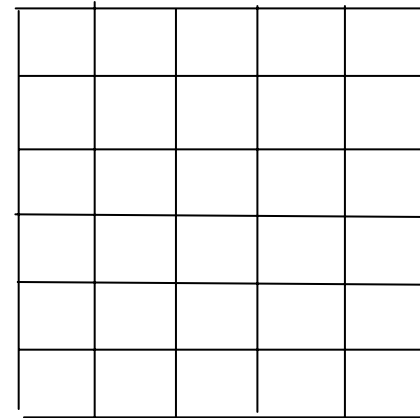
- Tasks: Visit vs. coverage
- Environment: static known vs. dynamic unknown
- Environment: structured vs. unstructured
- Vehicle: Ackermann, Omni-directional, tracks
- Control system interface: path commands vs. motor controls

Environment Representation

- Models the state of the environment
 - Map knowledge
 - Sensor updates
 - Vehicle configuration position, heading, state
 - Goal/task locations
- Two principal approaches
 - Grid
 - Graphs

Environment Representation: Grids

- Each cell represents a square region of environment
- Typically 100cm to 2m resolution depending on vehicle/task
- Costs estimated from terrain information in map
- Good for unstructured environments
- Good for omni-directional vehicles
- Good for small areas ($< 500\text{m}^2$)



Environment Representation: Graphs

- Each vertex represents a specific location in environment
- Each edge represents a drivable path between locations
- Costs estimated from path and terrain
- Good for structured environments
- Good for Ackermann vehicles
- Good for large scale areas (1km²--100km²)

Simplest Problem: Known static environment

- Given:
 - A static terrain map with obstacles
 - A single point goal
 - An omni-directional vehicle
- Find:
 - A drivable path from the vehicle location to the point goal
- Such that:
 - Costs (distance, time, energy) are minimized

Solution: Grid representation

- g_0 be the goal cell
 - $c(a,b)$ be the estimated *cost* (distance or time etc.) between adjacent cells a and b in the grid
 - $f(a)$ be the *cost* of the optimal path to g_0 from grid cell a
 - $n(a)$ return the set of neighbor grid cells of cell a
 - $\pi(a)$ be the neighbor cell of a that follows the optimal route to the goal g_0
- Then:

$$f(g_0) = 0$$

$$f(a) = \text{Min}_{x \in n(a)} (c(a, x) + f(x))$$

$$\pi(a) = \text{Argmin}_{x \in n(a)} (c(a, x) + f(x))$$

Solution: Find $f(a)$ for all cells

- To find optimal path from any a to g_0 :
 - Solve the simultaneous set of equations for $f(a)$ for all grid cells
 - Given a start location s_0 then:
 - $f(s_0)$ is the cost of the optimal path from s_0 to g_0
 - The path from s_0 to g_0 is
 - $\text{Path}(s)=$
 - If $s==g_0$ then $\text{path}=\{\}$
 - Else $\text{path}=\{\pi(s),\text{Path}(\pi(s))\}$

Solution: Solve $f(a)$ for all a

- Simplest method: Value Iteration

until a fixed point

do for each a in grid

do if not($a == g_0$)

then $f(a) = \underset{x \in n(a)}{\text{Min}}(c(a, x) + f(x))$

for each a in grid

do if not($a == g_0$)

then $\pi(a) = \underset{x \in n(a)}{\text{Argmin}}(c(a, x) + f(x))$

- Fixed point is no change in values of $f(a)$
- Uses definition of $f(a)$ as an *update* rule (called relaxation)
- Value iteration is slow, but useful as a conceptual tool to understand $f(a)$ and $\pi(a)$, especially when we get dynamic changes to $c(a, b)$

Solution: Search out from the goal

- Efficient method
 - Dijkstra's algorithm
 - $O(n \log n)$ where n is the number of cells in the grid
 - n grows as a square of the grid dimensions
 - Grid dimensions grow with increase in resolution
 - Good for local area around vehicle

```
Dijkstra( $g_0$ )
for each cell  $i$  in grid
     $f[i] := \infty$ 
 $f[g_0] := 0$ 
for each cell  $i$  in grid
    do Push( $i, Q$ ) //  $Q$  is a priority queue
while  $Q$  is not empty
     $u := \text{pop}(Q)$ 
    for each  $v$  in  $n(u)$ 
        do if  $f[u] + c(u, v) < f[v]$ 
            then  $f[v] := f[u] + c(u, v)$ 
                 $\pi[v] := u$ 
```

Summary Grid representation

- Advantages
 - Fast for small areas around vehicle
 - Good for unstructured environments
 - Costs based on terrain types
 - Can be incrementally updated due to sensor information (see later)
- Disadvantages
 - Path will comprise of straight segments with 0-radius turns at angle multiples of 45°
 - Path cannot be driven by an Ackermann vehicle
 - Does not scale up to large environments
 - Does not exploit any structure in the environment
- Extensions
 - Use Quad tree representation of grid

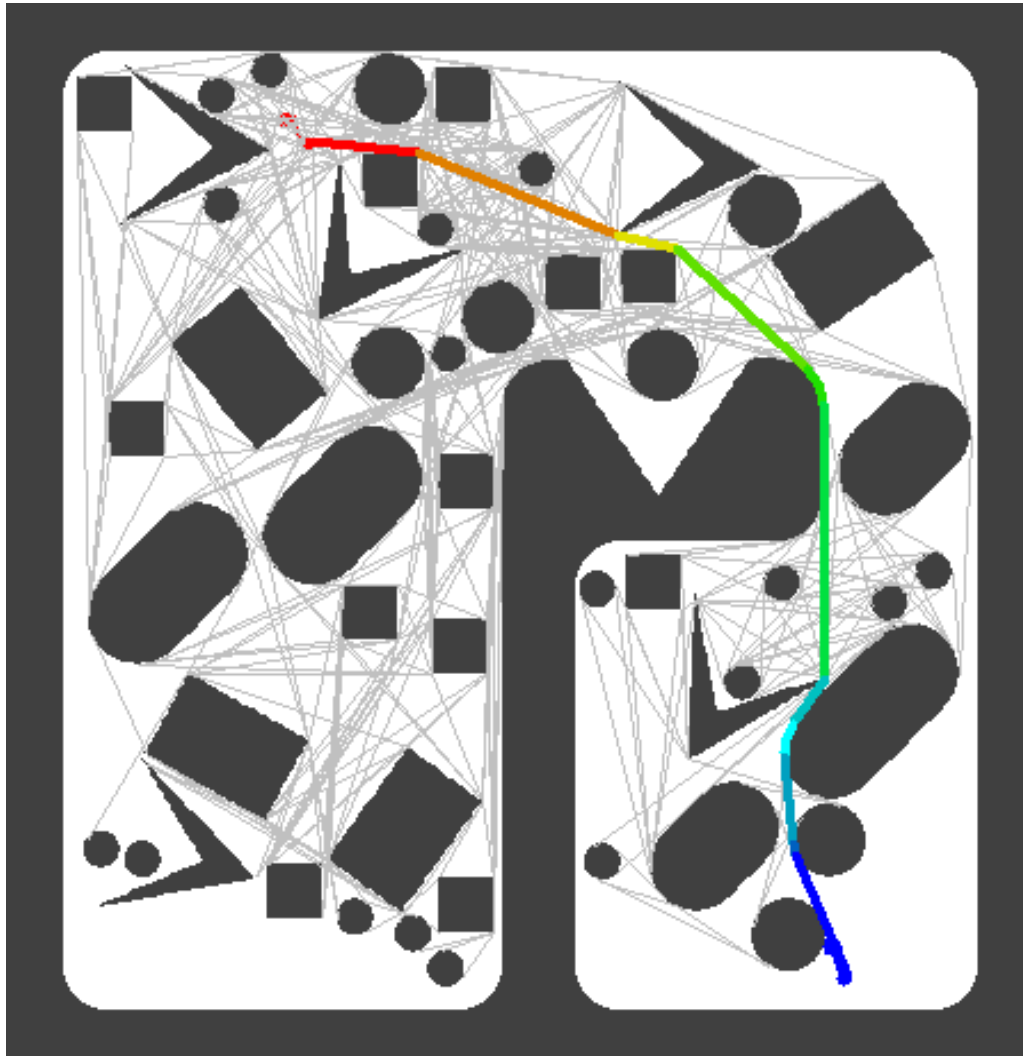
Simplest Problem: Known static environment

- Given:
 - A static terrain map with obstacles and road network
 - A single point and heading (configuration) start goal
 - An Ackermann steer vehicle
 - A start configuration
- Find:
 - A *drivable* path from the vehicle start configuration to the goal configuration
- Such that:
 - Path never intersects any obstacles
 - Path maximally utilizes given road network
 - Path costs (distance, time, energy) are minimized

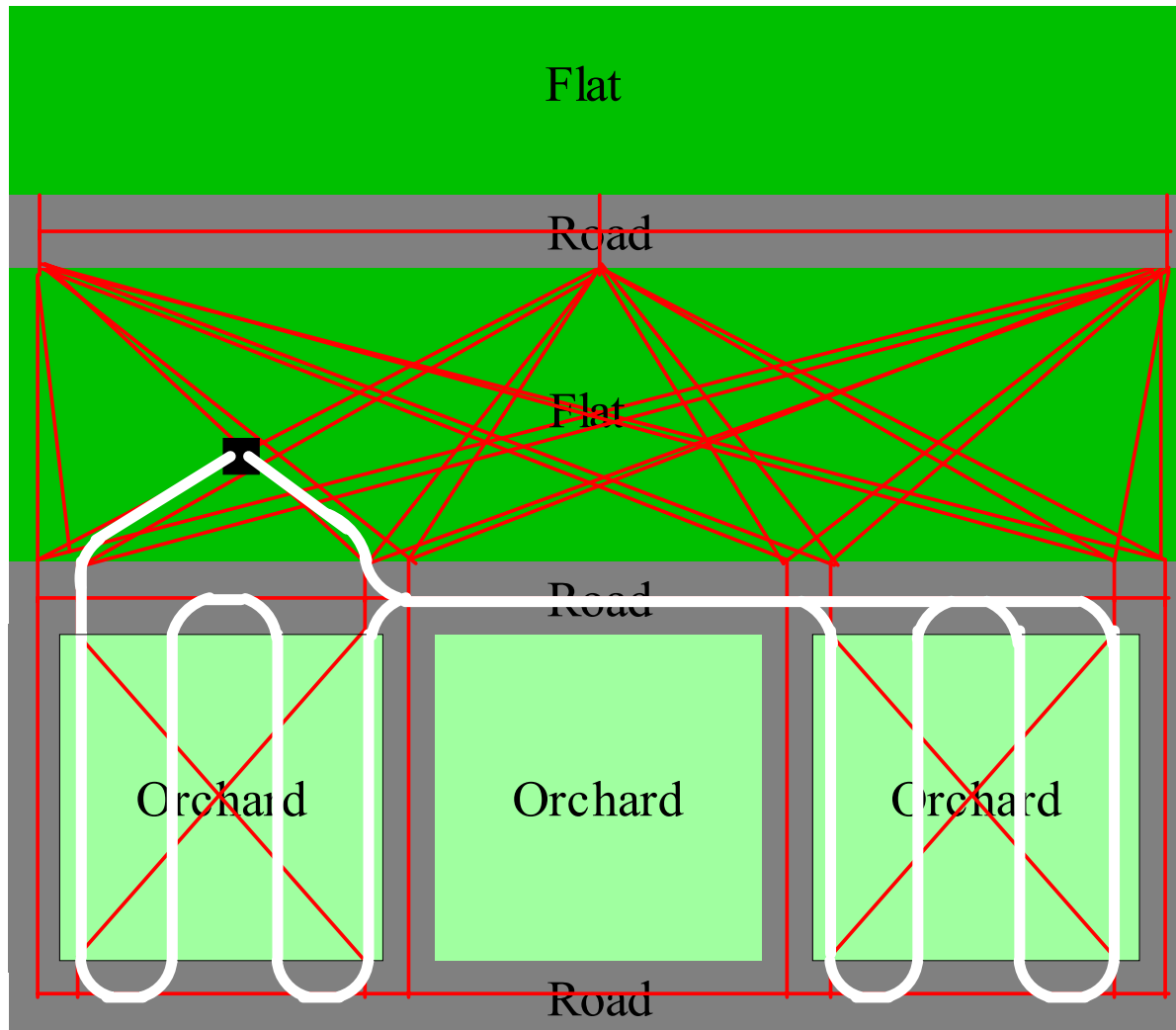
Environment Representation: Graphs

- Nodes represent vehicle configurations (location, heading, state)
- Edges represent drivable paths
- Which nodes? Which edges?
 - Exploit the structure in the environment
 - Exploit the structure in the tasks
- Visibility graphs
 - Obstacles “grow” to take into account vehicle footprint and turning radius
 - Nodes placed at “corners” of obstacles
 - Edges traverse the border of (convex) obstacles
- Mobility graph
 - Nodes placed at useful locations along borders and within terrain objects
 - Edges placed along known travel routes such as roads, borders of terrain objects

Visibility Graph

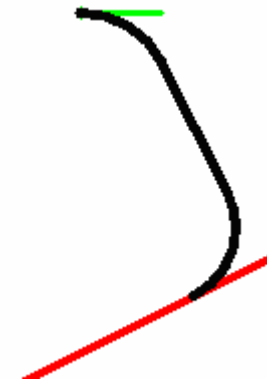


Mobility Graph

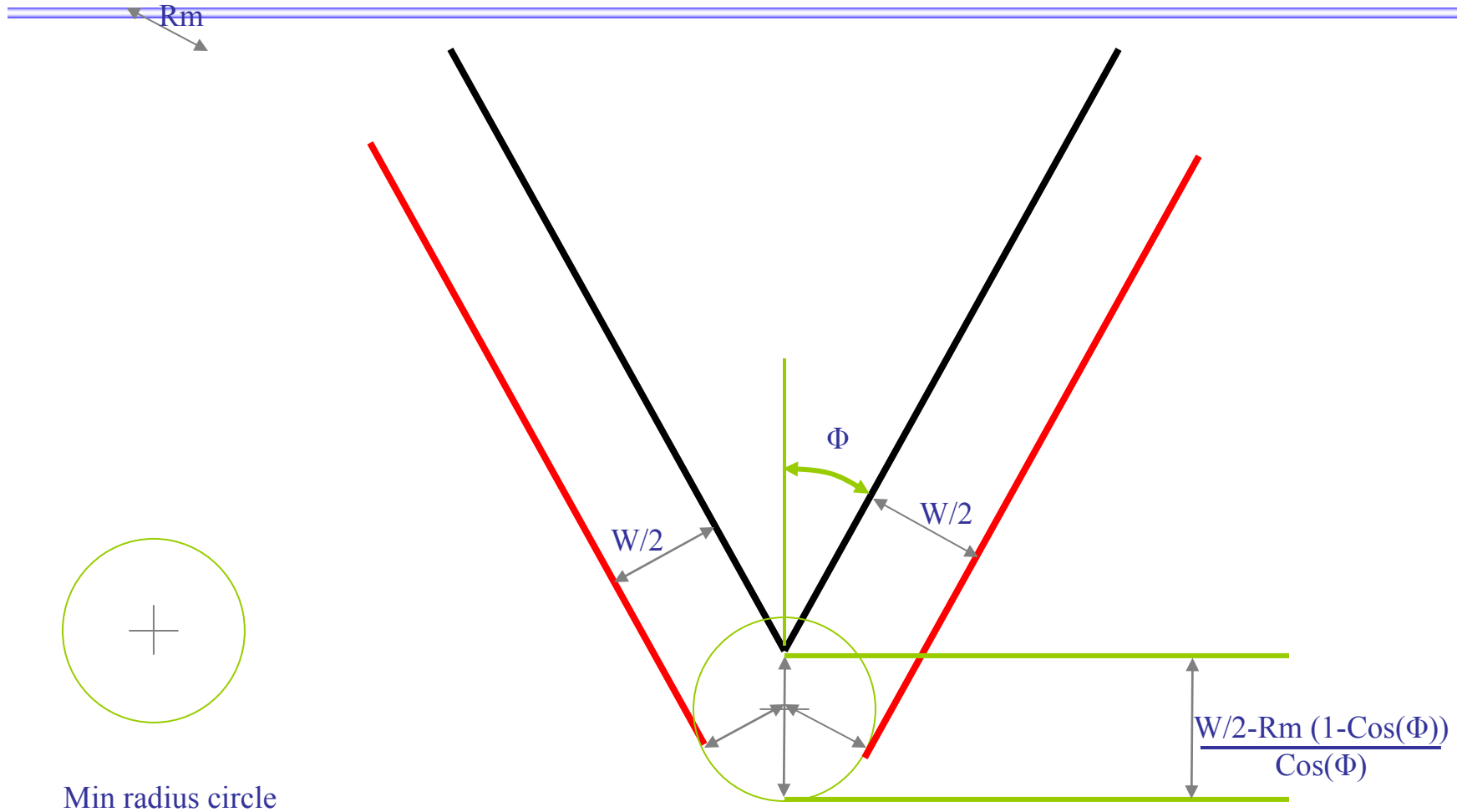


Drivable Paths

- Paths assembled from
 - Borders of grown obstacles
 - Borders of traversable regions (e.g. fields)
 - Along roads and trails
 - Connecting paths between borders and roads
- All paths must be drivable by Ackermann vehicle
 - Sequence of straight and arc path segments
 - Good approximation of drivable path for slow velocities
- Geometry engine
 - All objects and paths are
 - arc or straight sequences
 - Grows or shrinks arc/straight paths
 - Connects between configurations (location, heading) and paths

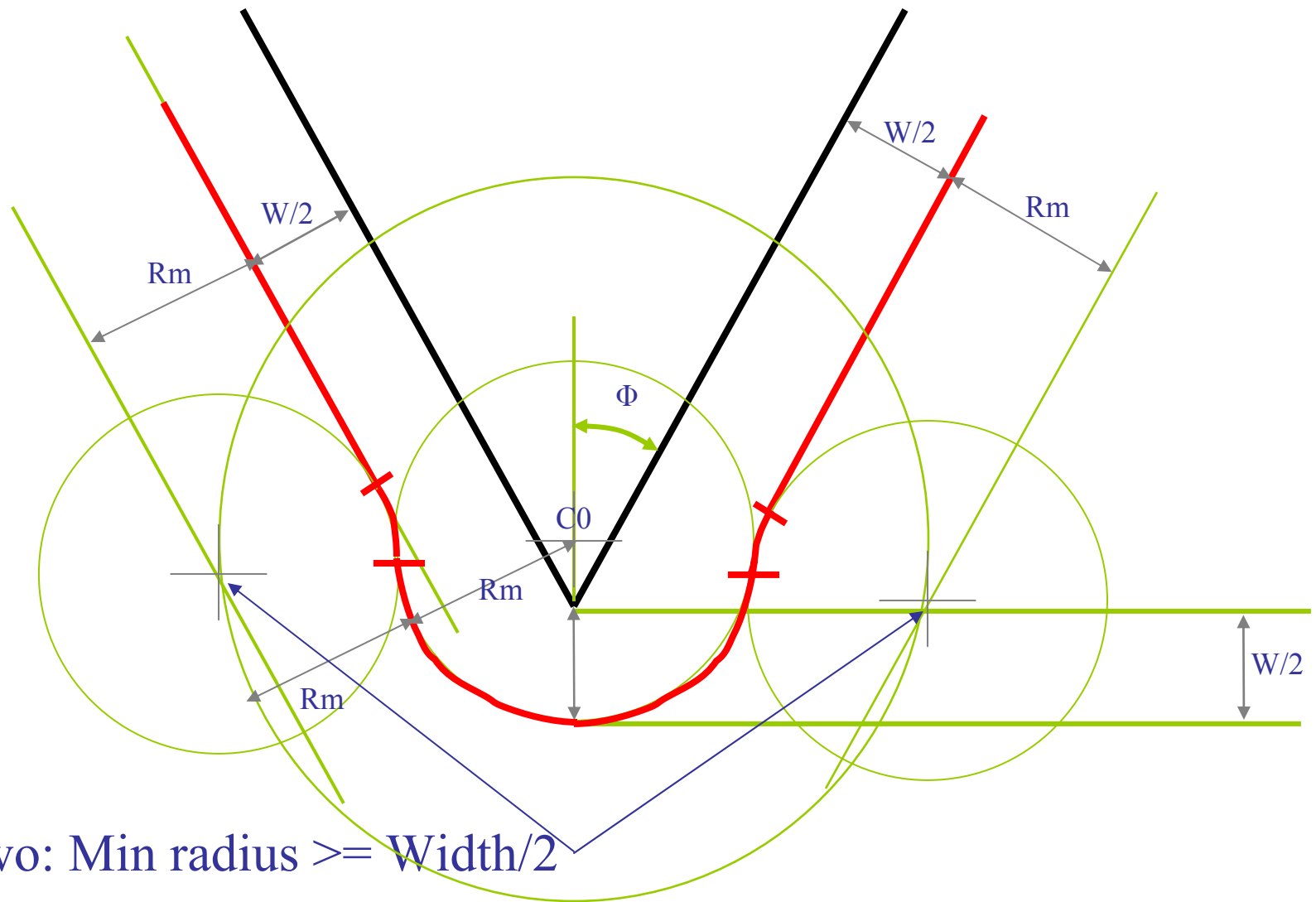


Visibility Graph: Growing Obstacles

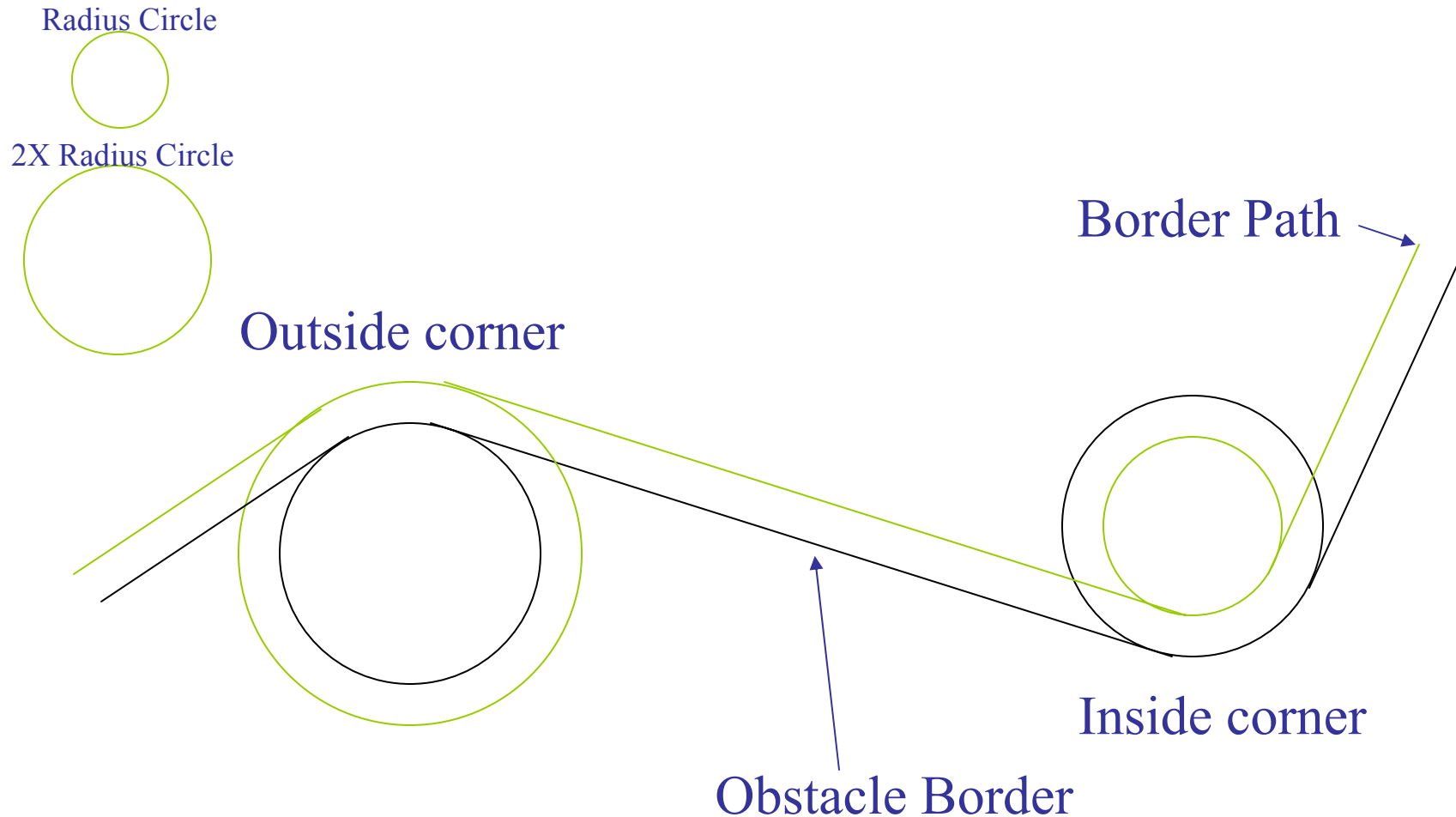


Case One: Min radius \leq Width/2

Visibility Graph: Growing Obstacles

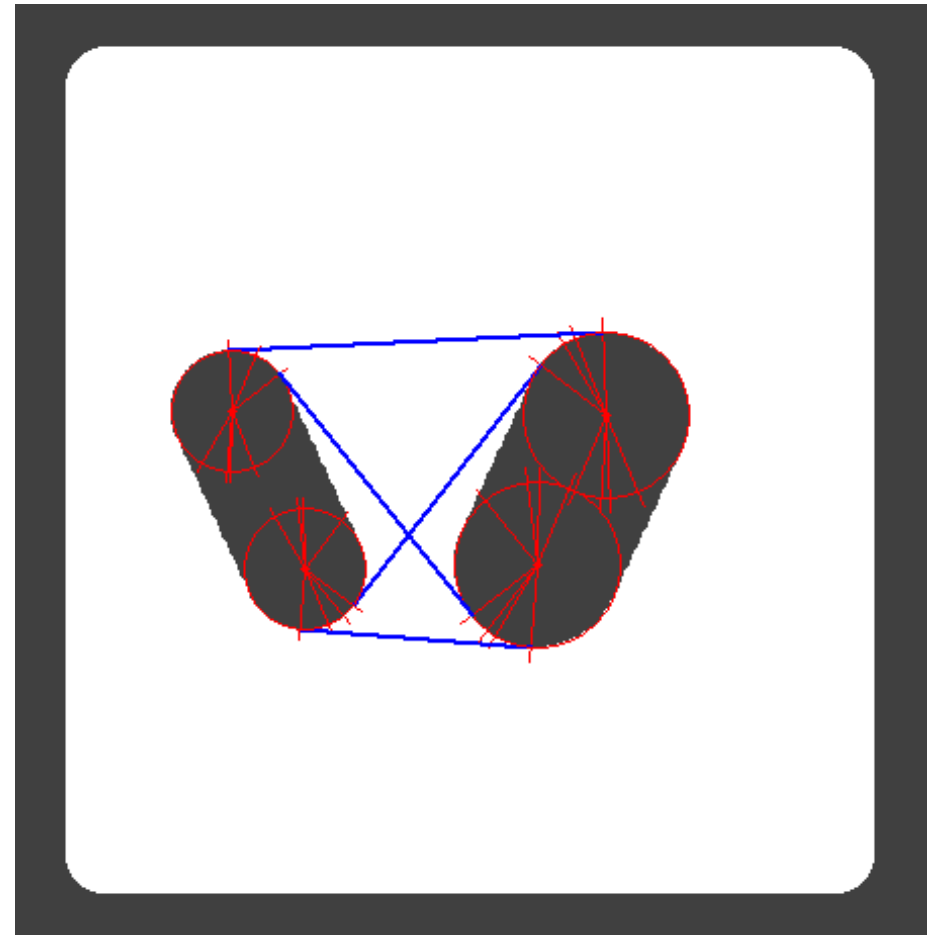


Visibility Graph: Growing Obstacles



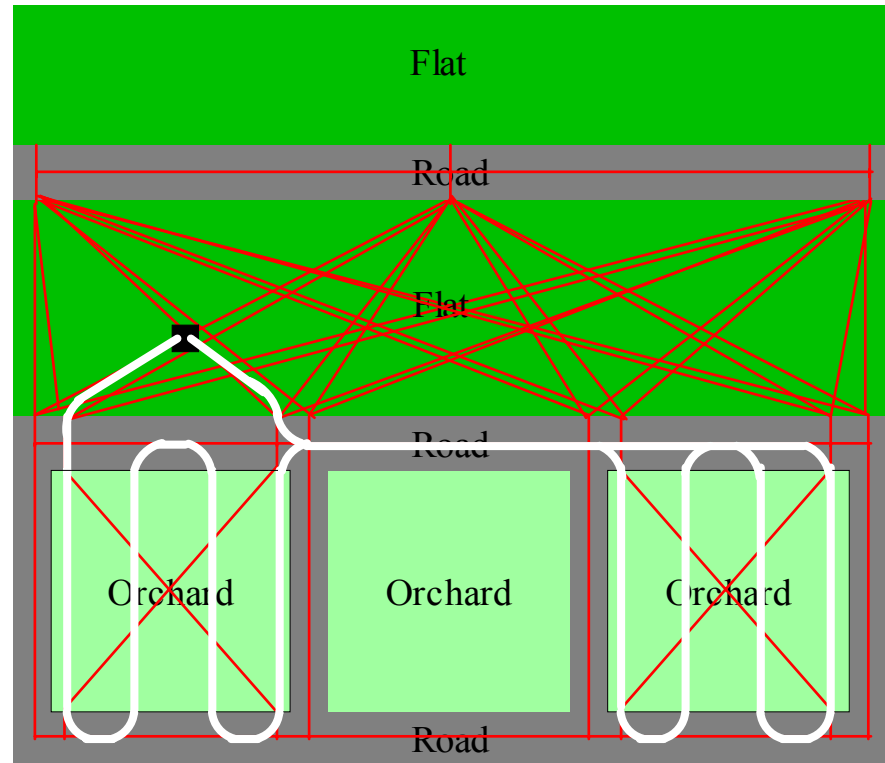
Visibility Graph: Connecting Obstacles

- All pairs of obstacles are connected
- Paths that intersect any obstacle are not added to the graph

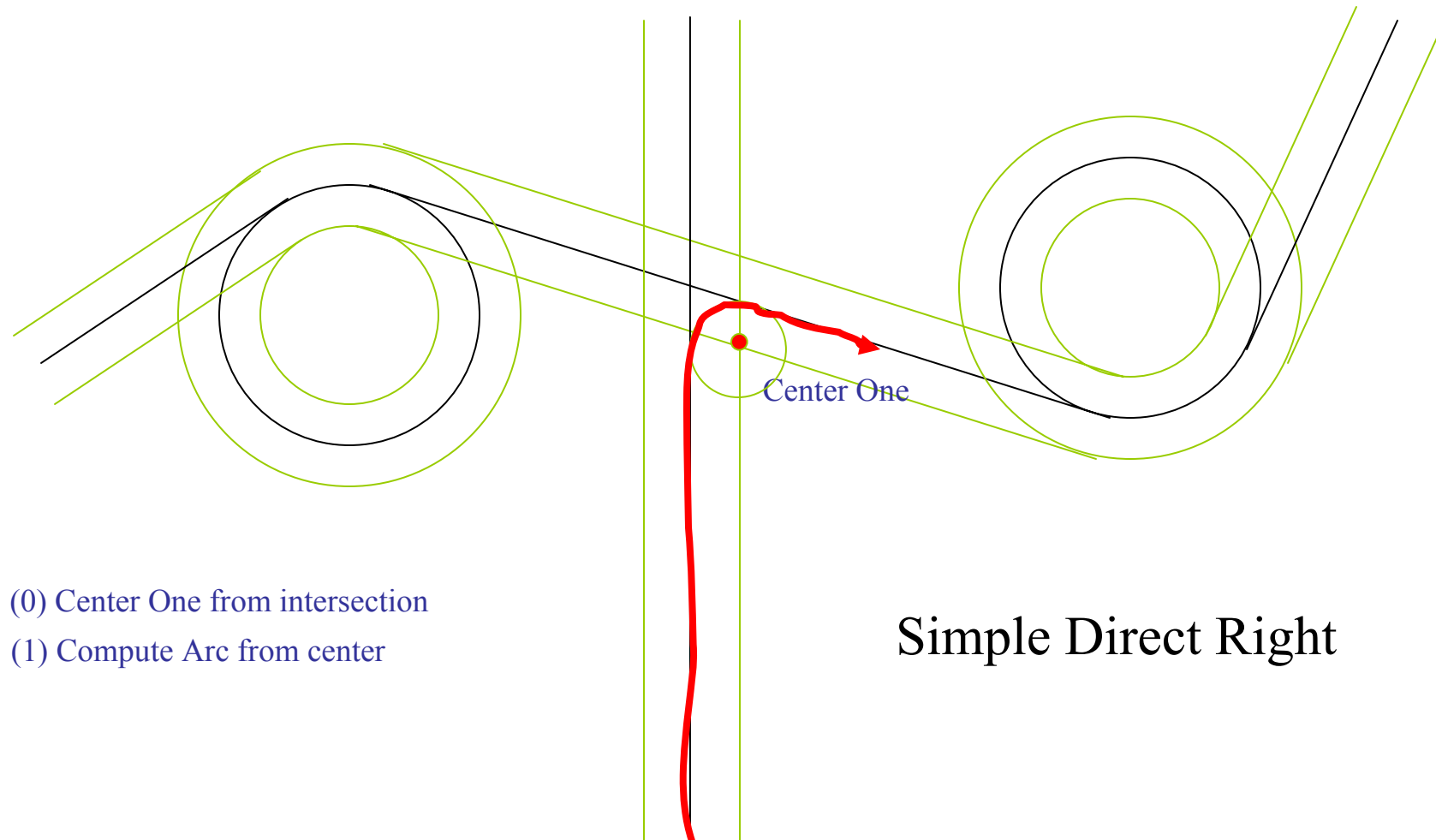


Mobility Graph: Connecting Shapes

- Around shapes: follow border
- Along Routes: follow lane path
- Between Shapes/routes?
 - Heuristically identify
 - Potentially useful enter/exit locations, headings
 - Arrival and departure angles
 - Run geometry engine to connect
 - Eliminate high cost edges

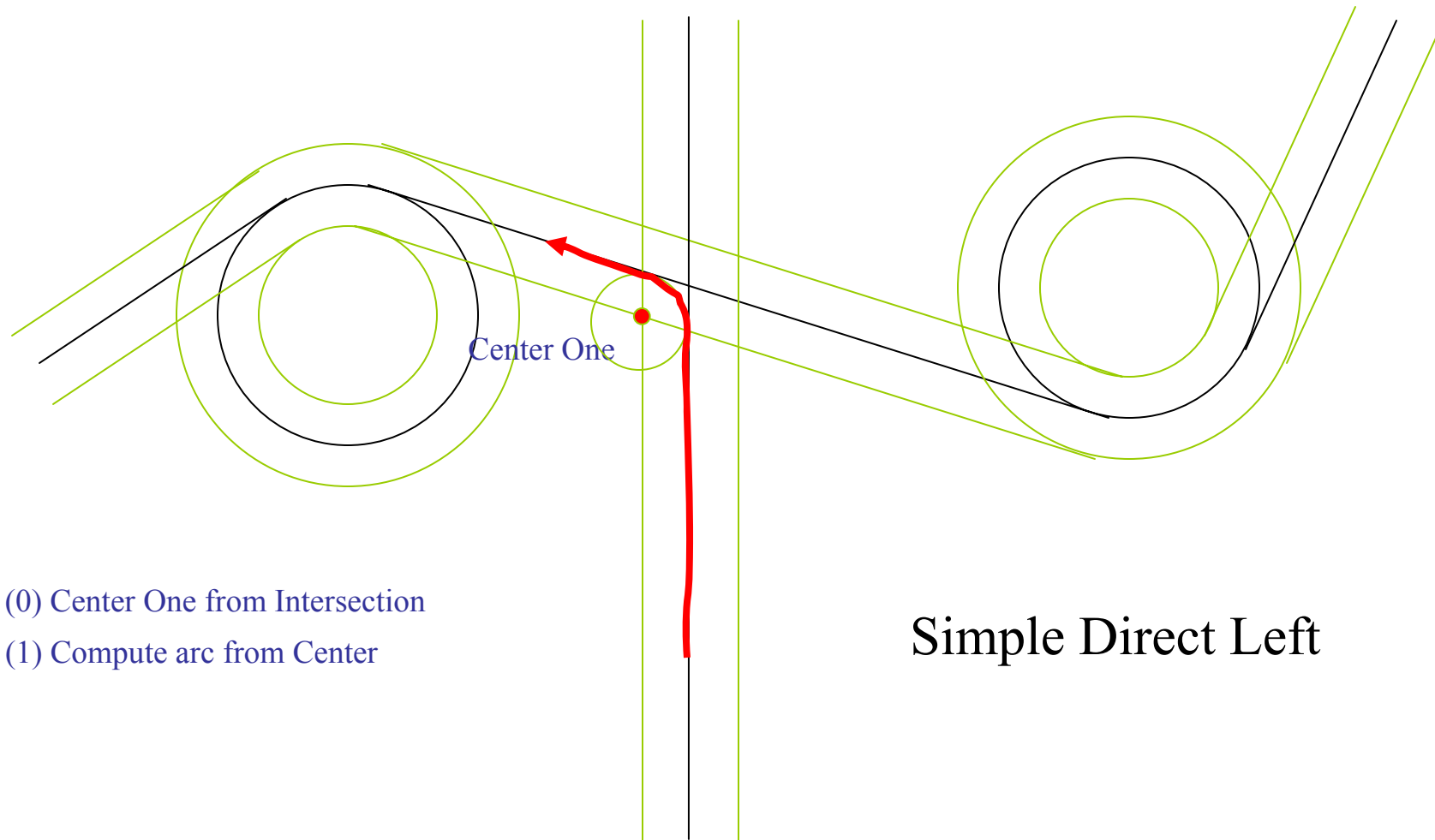


Mobility Graph: Geometry



- (0) Center One from intersection
- (1) Compute Arc from center

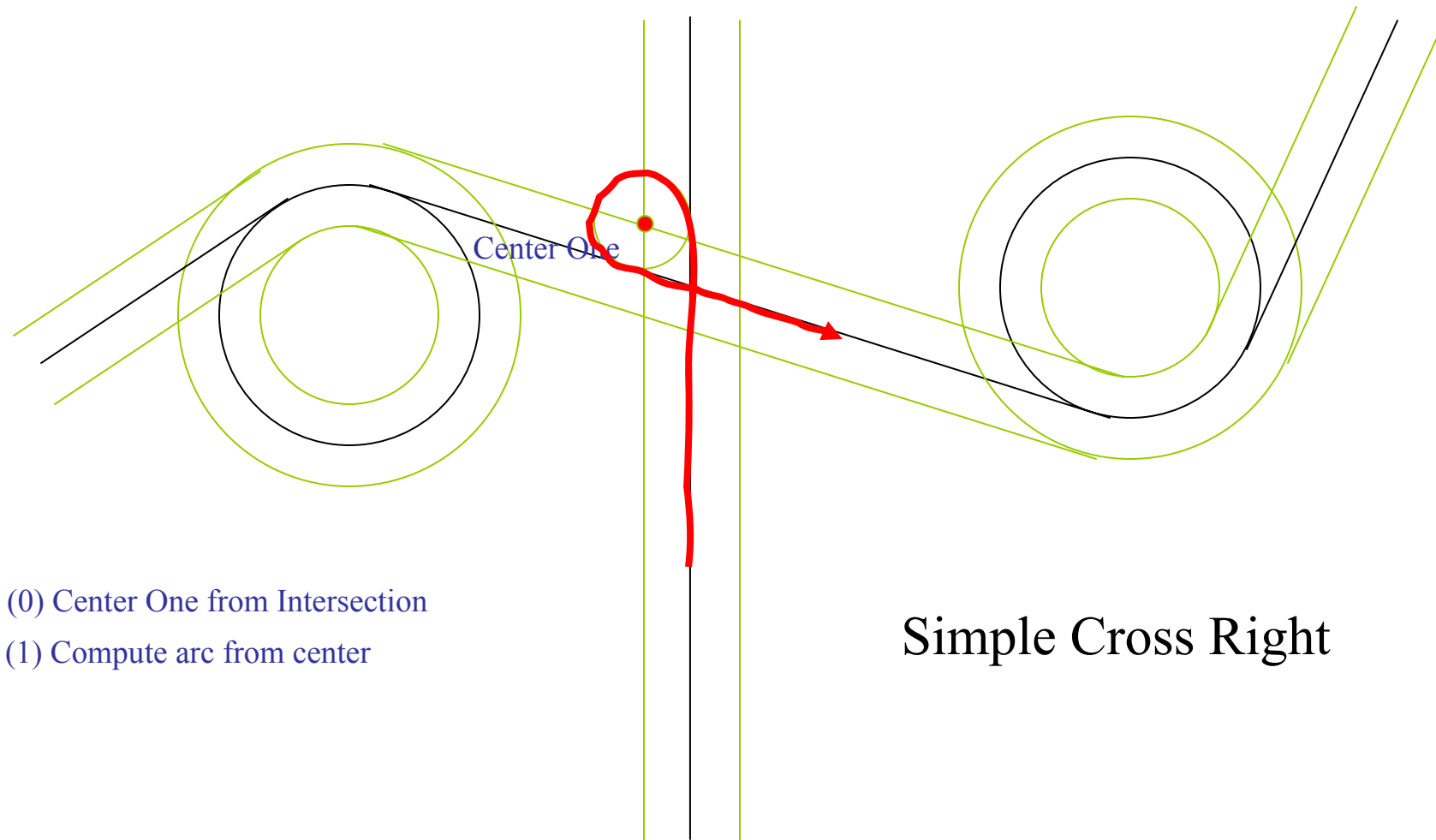
Mobility Graph: Geometry



(0) Center One from Intersection

(1) Compute arc from Center

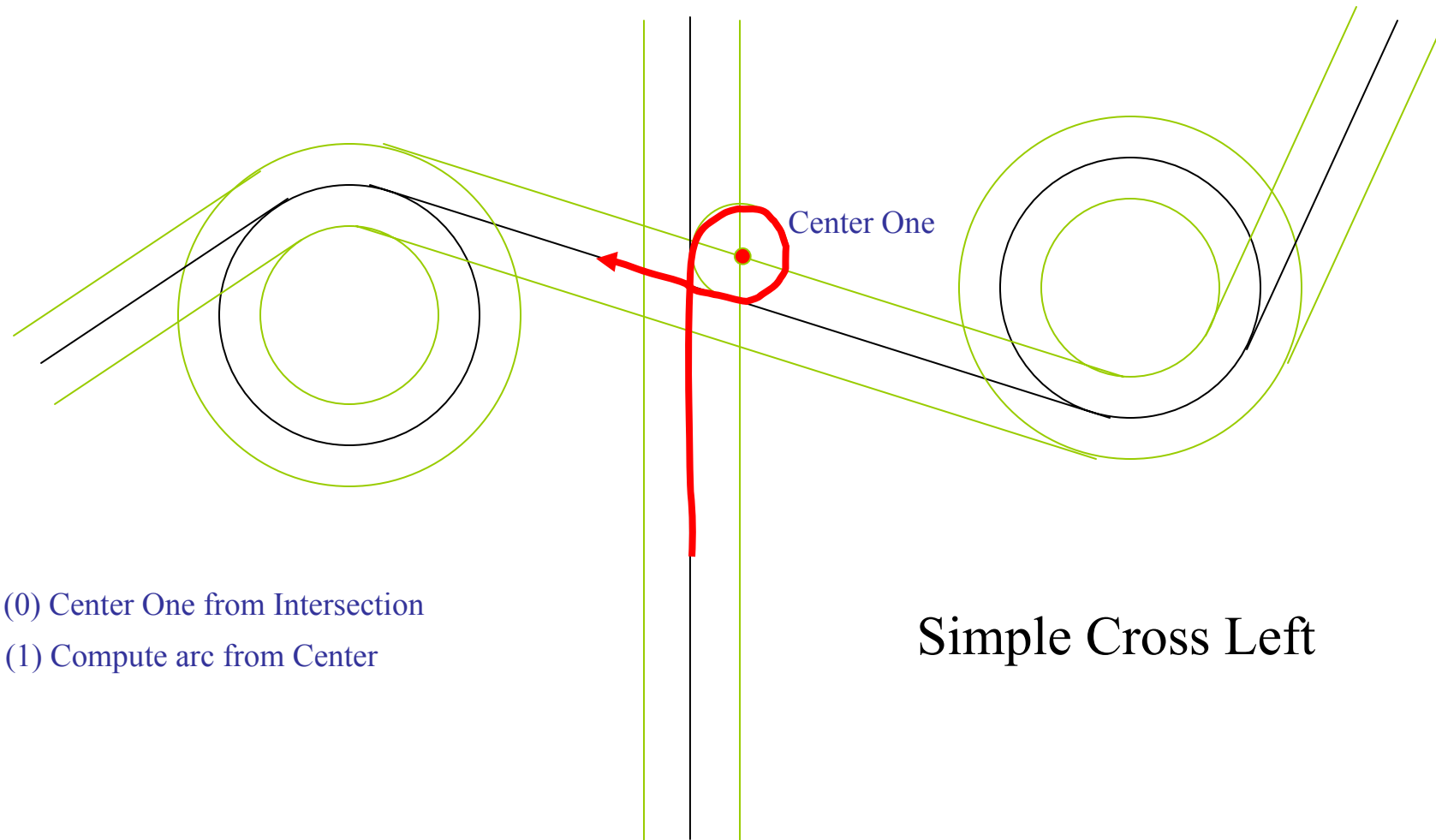
Mobility Graph: Geometry



(0) Center One from Intersection

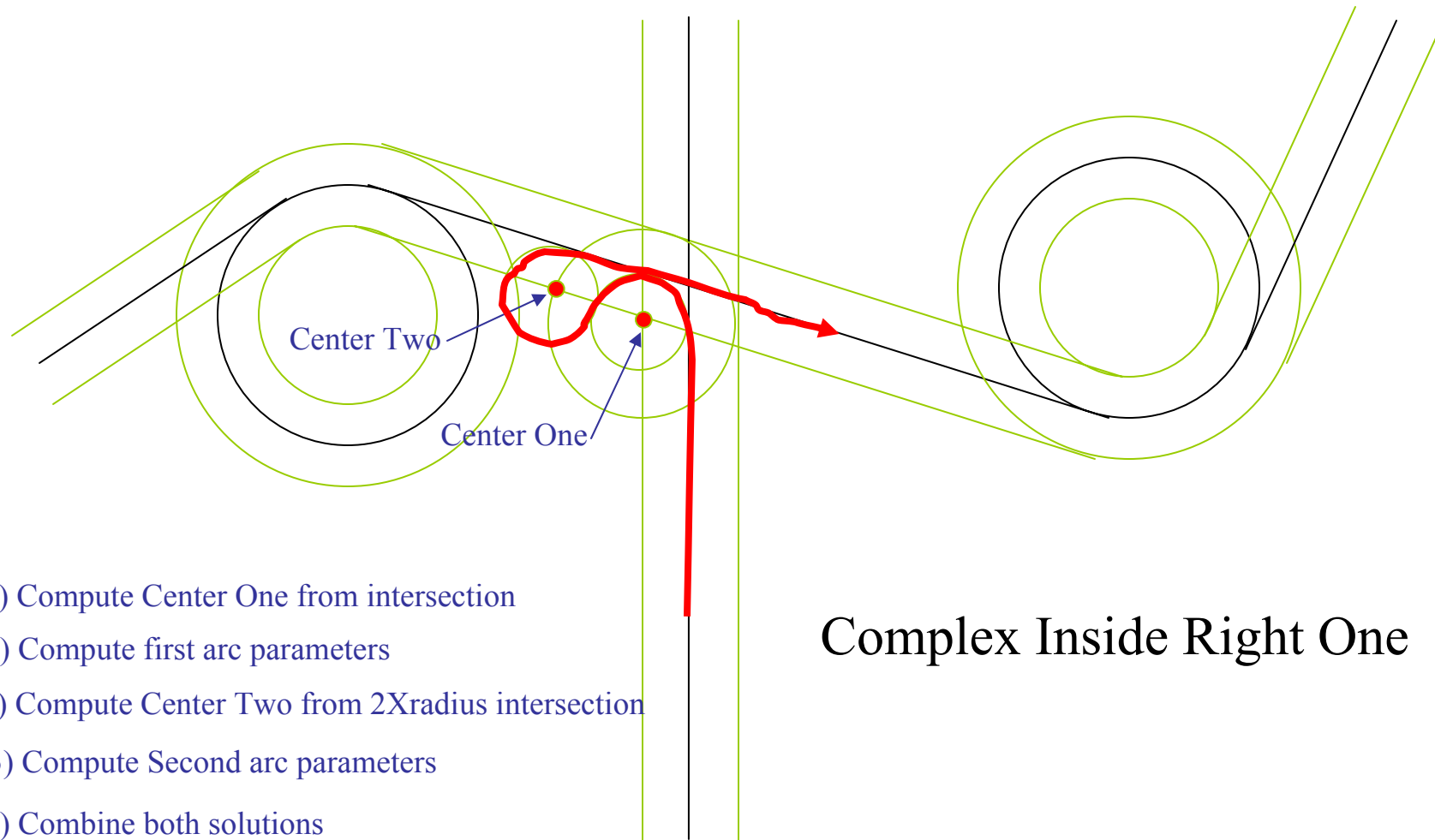
(1) Compute arc from center

Mobility Graph: Geometry

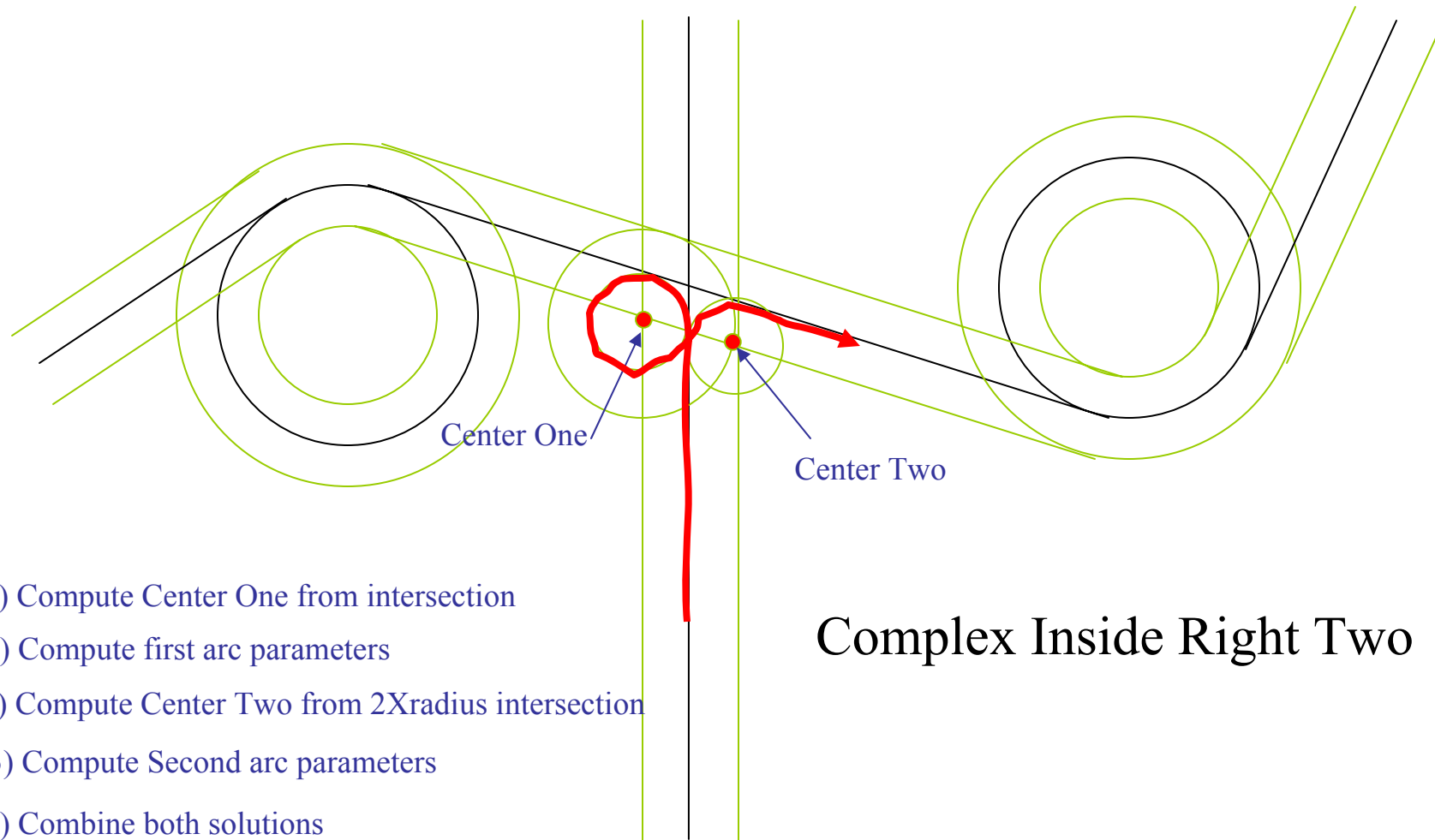


- (0) Center One from Intersection
- (1) Compute arc from Center

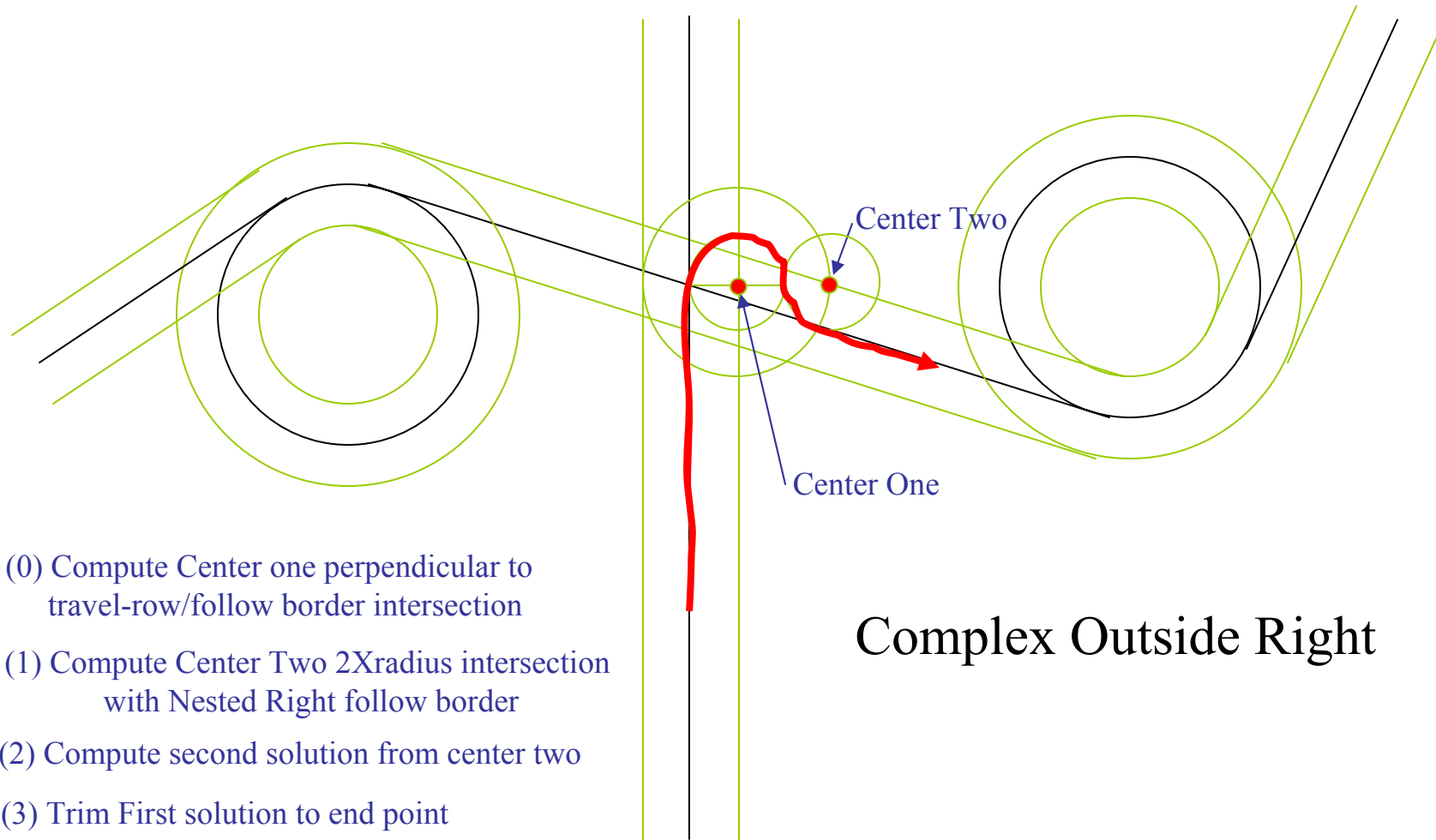
Mobility Graph: Geometry



Mobility Graph: Geometry



Mobility Graph: Geometry

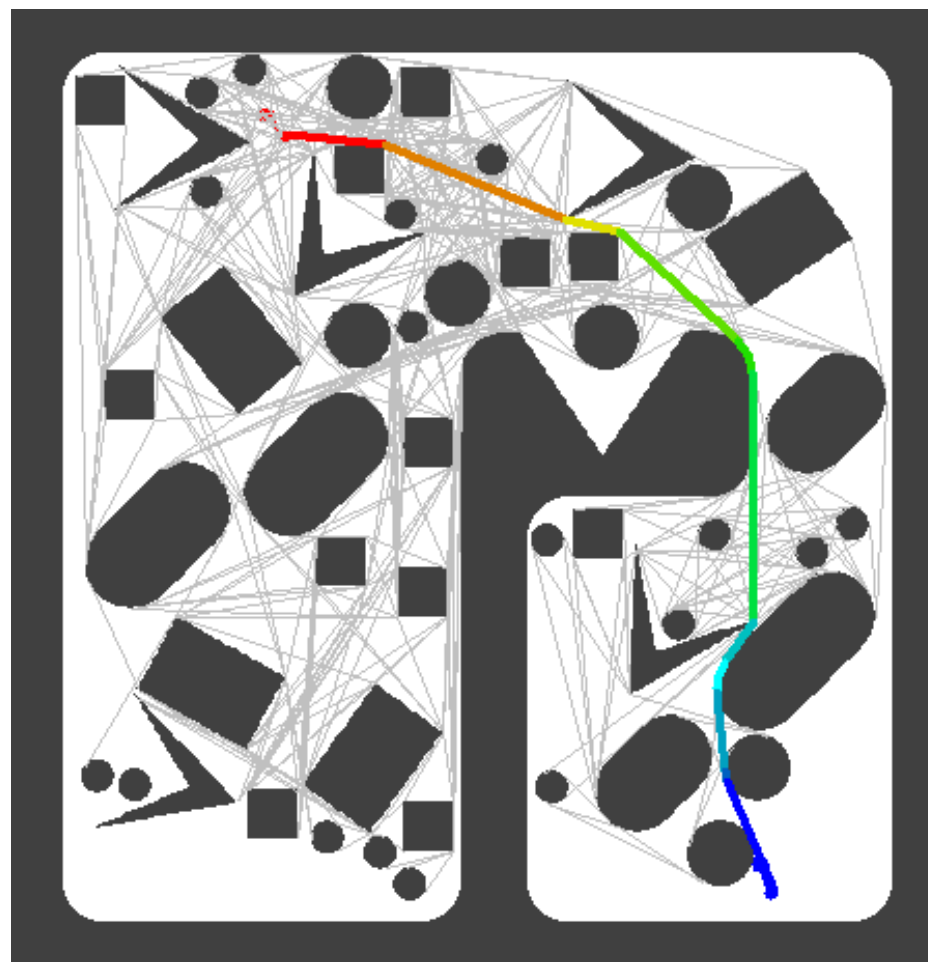


- (0) Compute Center one perpendicular to travel-row/follow border intersection
- (1) Compute Center Two 2Xradius intersection with Nested Right follow border
- (2) Compute second solution from center two
- (3) Trim First solution to end point

(4) Merge both solutions

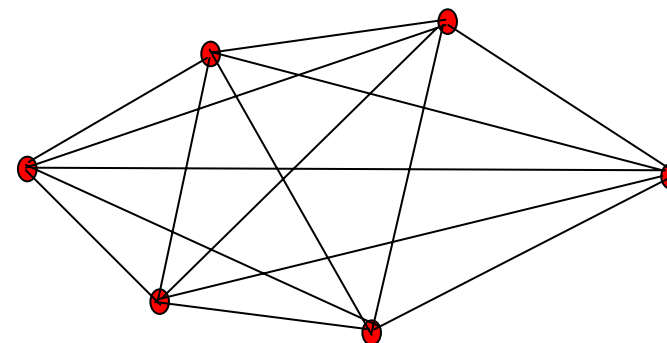
Visibility + Mobility Graph

- Once graph is constructed
- Find the optimal path
- Same approach as before
 - Grid cells map to graph nodes
 - $n(a)$ is all outgoing edge
 - $cost(a,b)$ is function of distance/time and terrain shapes crossed
- Additional heuristics possible
 - Limit obstacle/shape visits
 - Build graph on-the-fly



Solved: Single Goal location

- Upgrade to Missions
 - Multiple visit goals
 - Multiple coverage goals (see later)
- Compute paths between each goal location and vehicle location
- Extract a cost matrix between each location
- Solve the Traveling Sales Person Problem



	g_1	g_2	g_3	g_4	g_5	g_6
g_1						
g_2						
g_3						
g_4						
g_5						
g_6						

Traveling Sales Person Problem

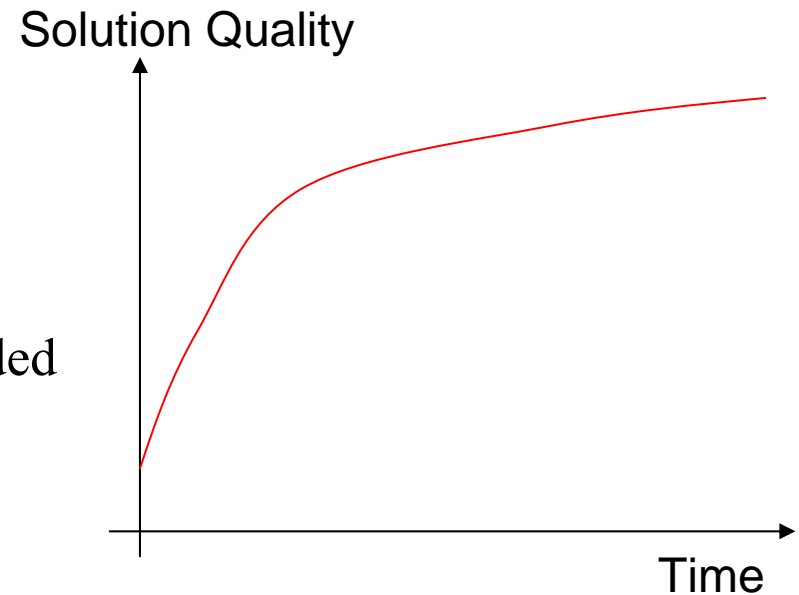
- Given
 - A set of goal locations
 - A cost matrix defining the cost to travel from each goal to each goal
- Find
 - A tour that visits each goal exactly once
- Such that
 - The total cost is minimized

TSP: Combinatorial Optimization

- Classic optimization problem
- $n!$ possible tours
- Known NP-Complete problem
- Well established near-optimal solution methods
 - Branch and Bound
 - Branch: expand solution with unvisited cities
 - Bound:
 - Global upper-bound best-so-far solution
 - Local lower-bound on partial solution
 - A*
 - Iterative repair (simulated annealing)
 - Genetic algorithm

Anytime Combinatorial Optimization

- In unmanned systems
 - Important that solution available quickly
 - Must use an “Anytime Algorithm”
 - Improve quality with thinking time
 - Always have a solution ready when needed
- Good Methods:
 - Branch and Bound
 - Iterative repair
- Poor Methods
 - A*



Branch and Bound

- Depth first search
 - Start at vehicle position
 - Expand current solution to all unvisited goals
 - Recursive backtracking
 - Terminate (success) with complete solution
 - Terminate (failure) when best-cost for current solution exceeds best-so-far (Bound)
 - Backtrack to previous choice
 - Try next city (Branch)
 - Can use best-first (greedy) expansion order
 - First solution is greedy
 - Improved with time through backtracking

Iterative Repair

- Start with random solution
- Pick random repair (mutation)
 - Swap order of two goals
 - Reverse subsequence
- Accept repairs that reduce the cost
- Probabilistically accept repairs that increase the cost
 - Use Metropolis criteria with temperature
- Maintain best-so-far solution
- Return when needed

Summary: Static Mission Planning

- Considered only visit goals (position and heading)
- Two representations of the environment
 - Grid
 - Good for unstructured, omni-directional vehicles
 - Flexible but slow without quad trees
 - Graph
 - Good for structured, Ackermann vehicles
 - Needs a geometry engine process paths and shapes
- Use Dijkstra's algorithm to compute optimal paths
- Use Anytime Combinatorial Optimization to find goal order
 - Branch and Bound
 - Simulated Annealing

Upgrade: Add Coverage Goals

- Coverage Goals
 - Multiple applications:
 - Land Mine Clearing
 - Farming: Spraying, plowing, harvesting
 - Strip Mining
 - Lawn Mowing
 - Search and rescue

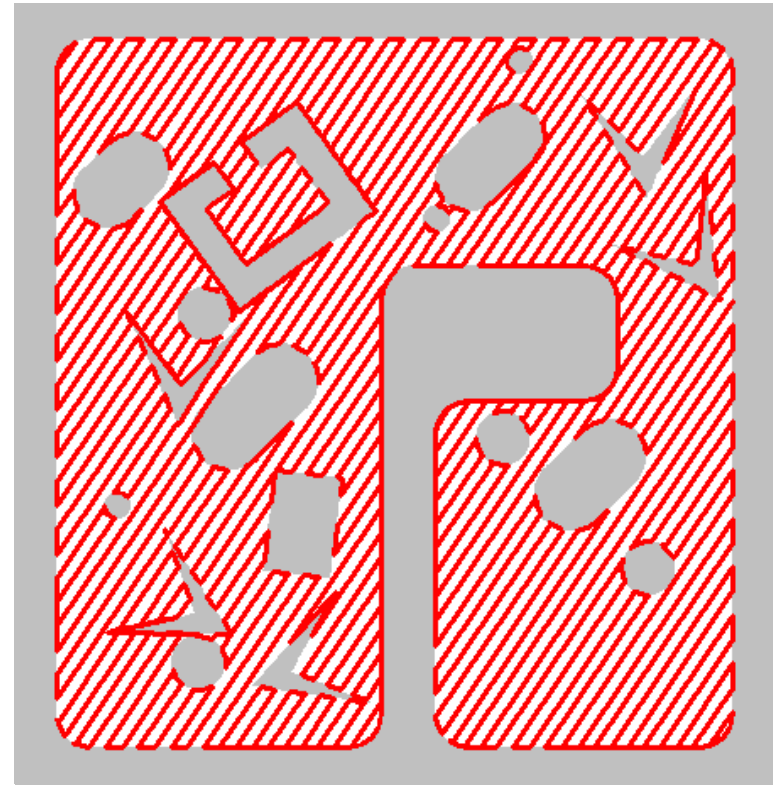
Problem: Coverage Goal

- Given:
 - Region as a polygon, which may include mapped obstacles
 - Ackermann steer vehicle
 - Operation width
- Find:
 - Continuous path that covers the region with adjacent sweeps
- Such that:
 - Path cost is minimized
 - Gaps in coverage minimized
 - Overlaps in coverage minimized



Problem: Coverage Goal

- Given:
 - Region as a polygon, which may include mapped obstacles
 - Ackermann steer vehicle
 - Operation width
- Find:
 - Continuous path that covers the region with adjacent sweeps
- Such that:
 - Path cost is minimized
 - Gaps in coverage minimized
 - Overlaps in coverage minimized



Solution: Combinatorial Optimization

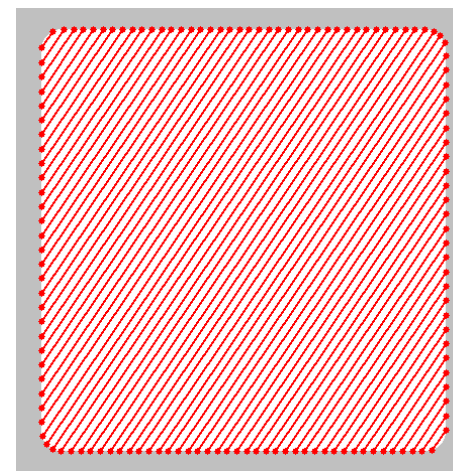
- Use geometry engine to create travel rows across region
- Connect travel rows together along common borders of shapes
- Reduce to TSP
 - Map each m travel-row to m visit goals (or cites)
 - Cost matrix $m \times m$ from the end of each travel row to the beginning of each other travel row
 - Tour corresponds to complete coverage

Unregistered HyperCam

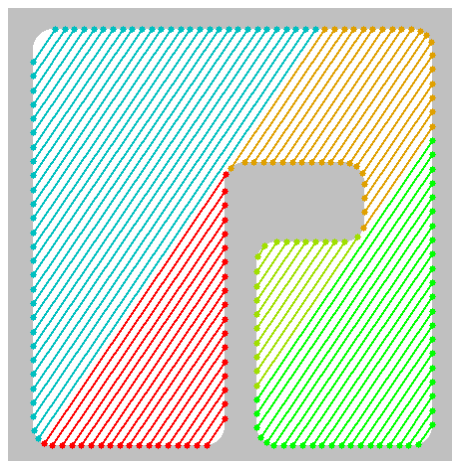


TSP Reduction?

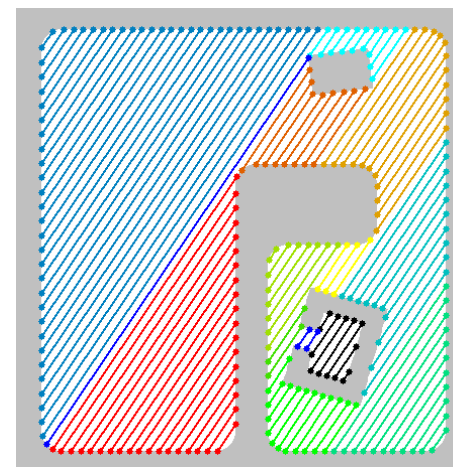
- Inadequate Solution:
 - Number of cities is often in the 100's
 - Cost matrix is sparse
 - Many poor quality local min
 - Many problems appear unsolvable



Greedy
solution
found quickly

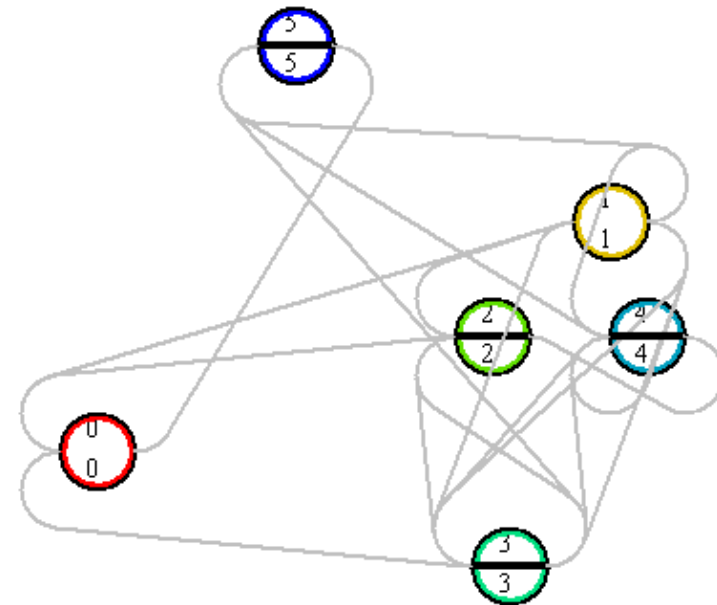
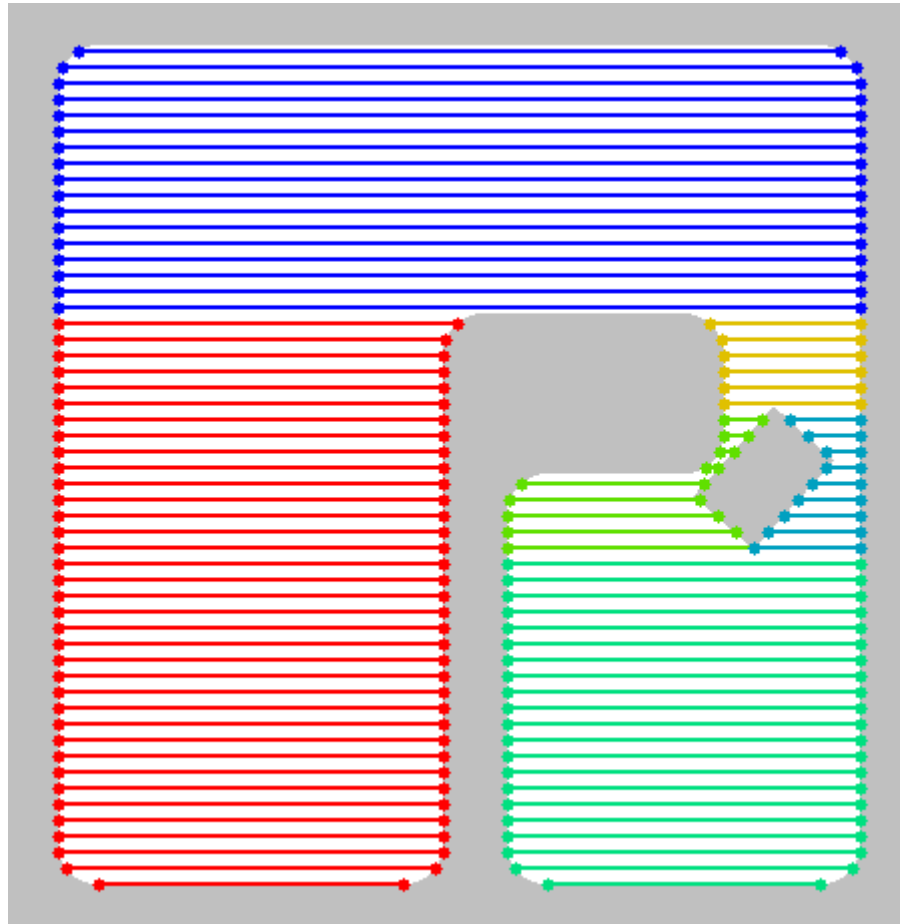


Many low
quality
solutions



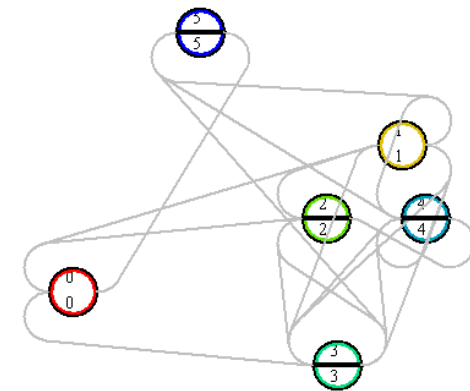
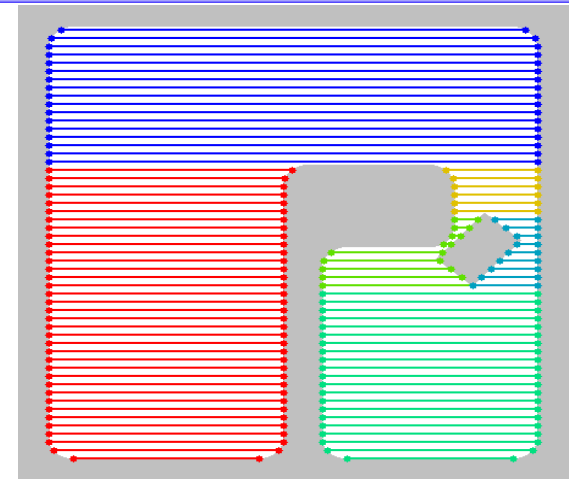
Unsolvable

Abstraction into Partitions



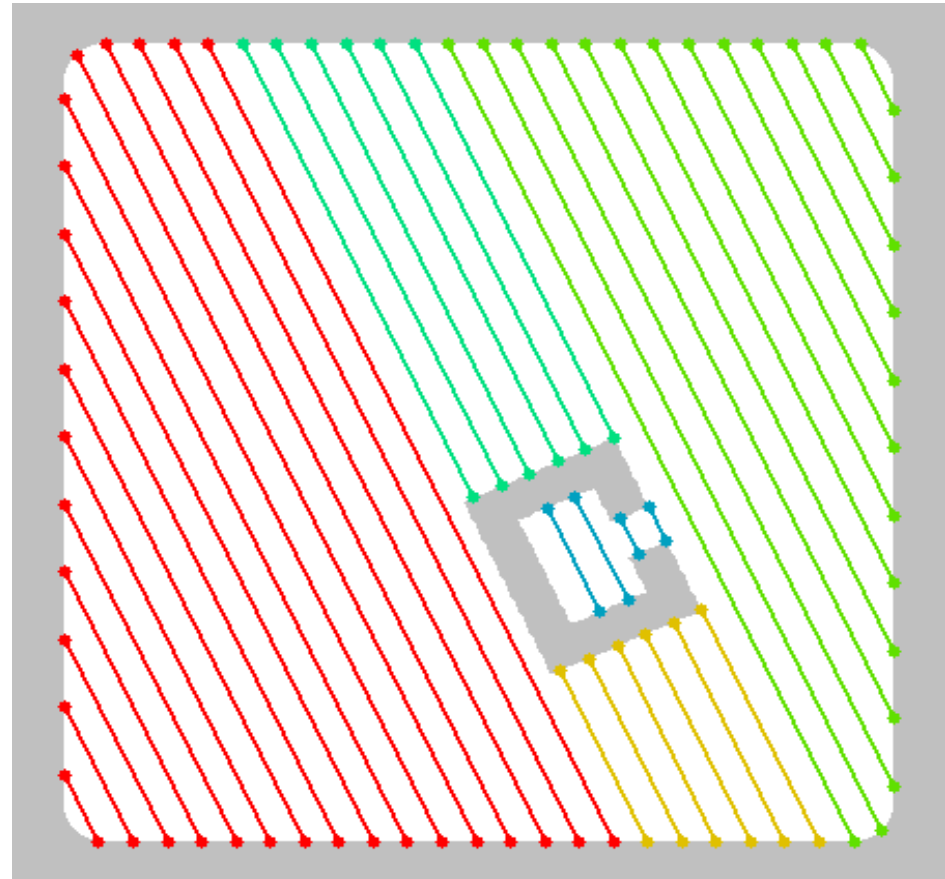
Partition into Equivalence Classes

- Travel edges are equivalent if they
 - Connect to the same two shapes
 - Adjacent at each shape connection
- Search space is significantly reduced
 - Greedy solutions are often near-optimal
- Analysis of Odd and Even partitions reduce space further
 - Unsolvable problems have unreachable partitions!



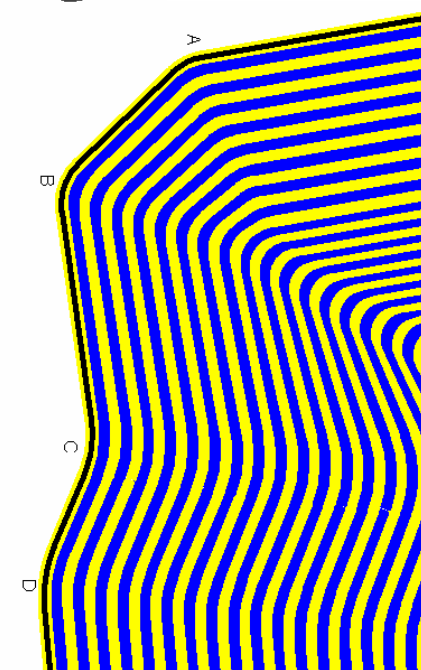
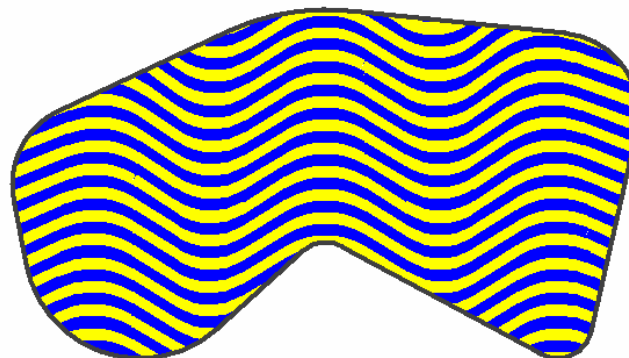
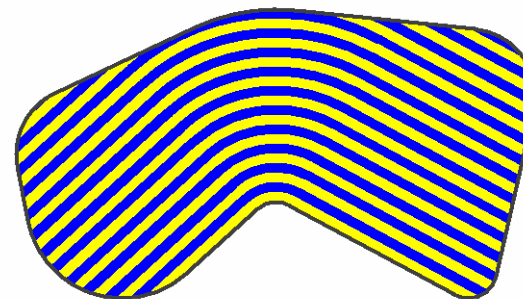
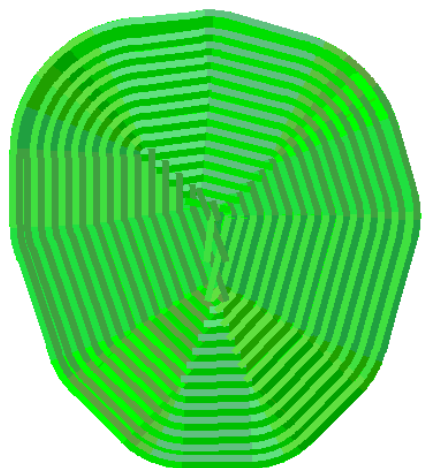
Unreachable Partition P:

- If P touches one shape S:
 - If all other partitions touching S are even
- If P touches two shapes S1 and S2
 - If P is odd
 - And the number of other odd partitions touching S1 is even
 - Or the number of other odd partitions touching S2 is even
 - If P is even



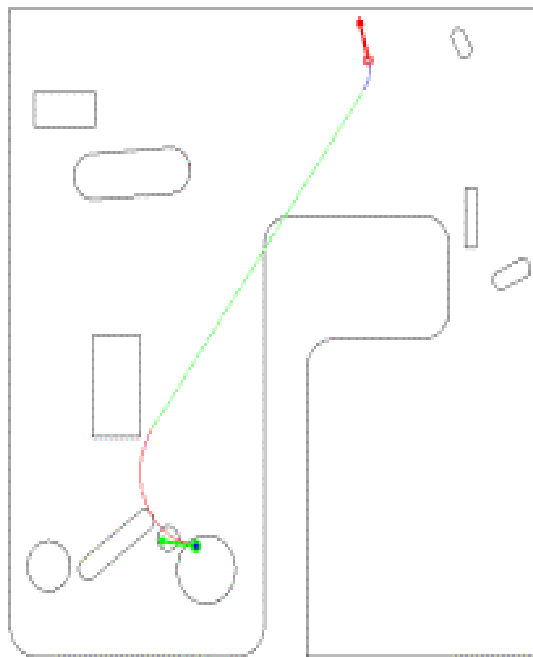
Contour Coverage Goals

- Many coverage tasks require specialized patterns
 - Contours: across slope, aesthetics
 - Spirals: harvesting/sensor constraints

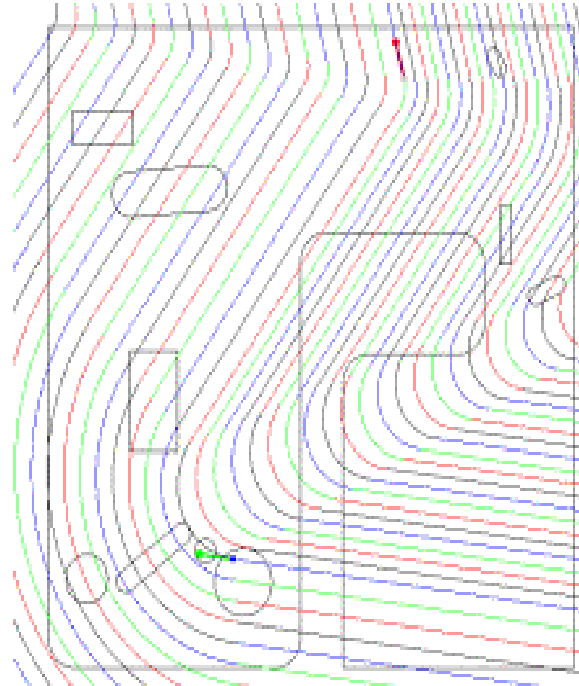


Constructing Travel Rows

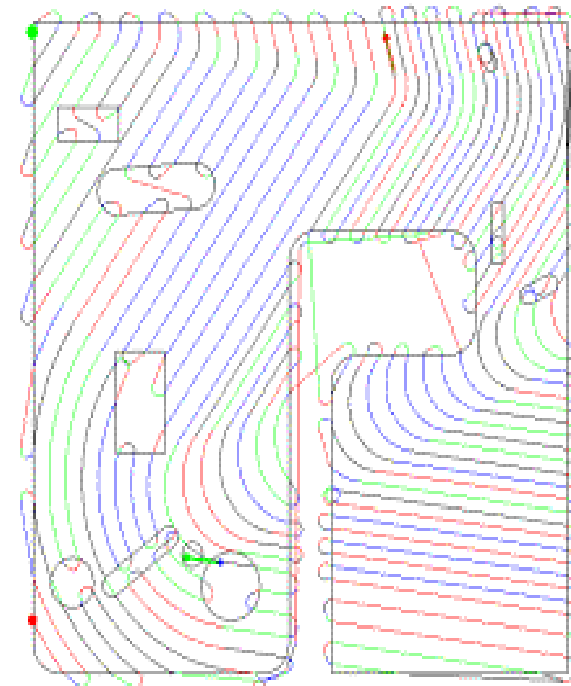
- Given target contour line
- Contour on both sides to form transparency
- Intersect with region and internal obstacles



1. Target Line



2. Transparency



3. Coverage Solution

Summary: Coverage Tasks

- Each Coverage tasks generates alternative solutions
- Mission planner integrates each coverage solution into tour with visit goals
- Comprehensive UMV tasking with near optimal solutions
- Principal Limitation: STATIC!
- Rest of talk upgrades these methods to work in dynamic partially known environments
 - New map data from sensors
 - New tasks from user
 - Dynamic obstacles (other vehicles)
 - No global location reference

Dynamic Sensor Updates

- Grid-based representation
 - Sensor information updates both increase and decrease costs
 - Example D* (Tony Stentz)
 - No local minima (traps)
 - Each iteration
 - Collect changed $c(a,b)$ from sensors
 - Update $f(a)$ and $\pi(a)$
 - Follow $\pi(a)$



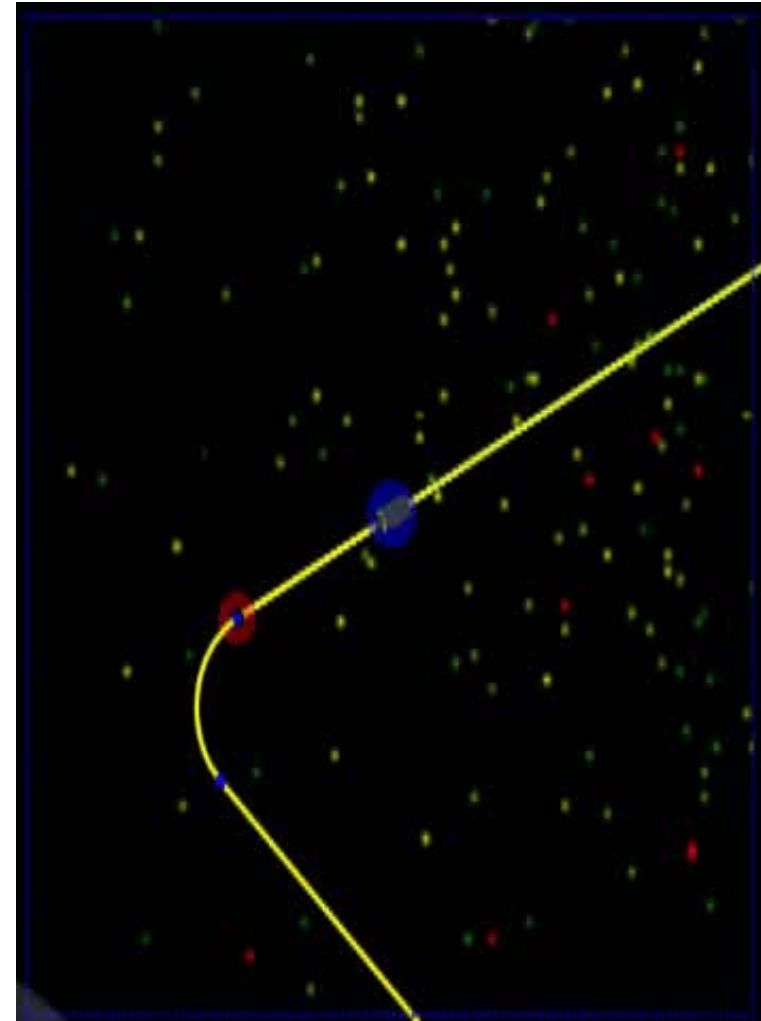
From the Mars Autonomy Project

D* Dynamic Grid Updates

-
- If new $c(a,b)$ decrease the current cost
 - Update $c(a,b)$ from sensor data
 - Relax $f(a)$ using the modified Dijkstra's algorithm
 - priority queue initialized to cells connecting changed costs
 - If new $c(a,b)$ increase the current cost
 - Update $c(a,b)$ from sensor data
 - Clear $f(x)$ for all cells x , where $\pi(x)$ leads through a,b (set to ∞)
 - Relax $f(a)$ using a modified Dijkstra's
 - priority queue initialized to cells connecting a cleared cell

Dynamic Sensor Updates

- Graph based representation
 - Sensor information updates both increase and decrease region costs
 - Each iteration
 - Collect changed cost regions from sensors
 - Update graph connectivity and costs
 - Update $f(x)$ and $\pi(x)$
 - Follow $\pi(x)$



Graph Dynamic Updates

- Add new mobility graph edges through new cheaper regions
- Relax $f(x)$ using the modified Dijkstra's algorithm
 - priority queue initialized to graph nodes connecting either new edges or edges with reduced costs
- Add new visibility graph edges around new obstacles
- Clear $f(x)$ for all nodes whose $\pi(x)$ leads through cost increasing edges (set to ∞)
- Relax $f(x)$ using a modified Dijkstra's
 - priority queue initialized to nodes connecting a cleared node

Dynamic Updates

- With reliable referenced location and orientation
 - Search-based dynamic re-planning can avoid local minima
 - Quickly re-direct the vehicle to a new path around obstacle or through a “short-cut”
 - Ackermann vehicles may require backing-up and 3-point turn capability to avoid traps
 - Control system may become unstable due to changing the current path command during execution
- Integrating Mission re-planning may cause the vehicle to re-order goals
 - Importance of anytime mission planning algorithms

Unknown Structured Tasks

- Task structured but map not known at planning time?
- Delayed commitment approach
 - Abstract plan is instantiated during running from sensor data
 - Delayed commitment avoids re-planning
- Example: ODIS under-car
 - Inspection plan is specified without knowing the position of the car
 - Sensing actions are incorporated into the plan
 - Path commands are created as functions of objects found during execution

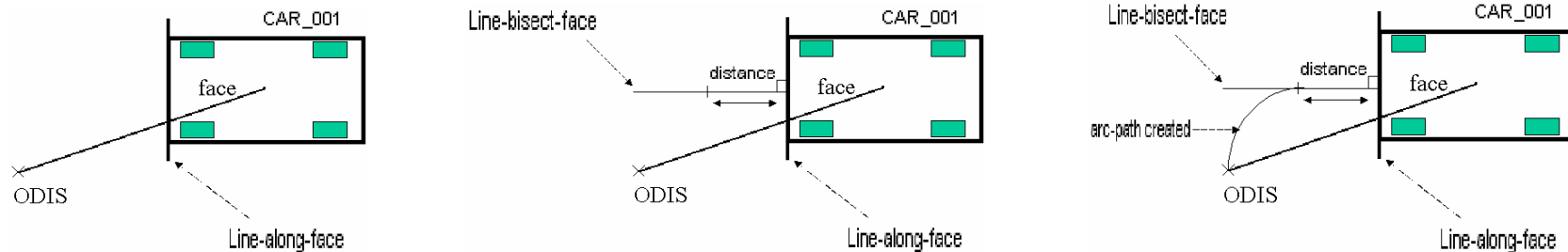
Delayed Commitment Planning

- **findCar() script**
 - If there is a car, find bumper and move closer.
 - Fit the open-left tire.
 - Fit the open-right tire.
 - Move up the centerline of car *//now estimated from tires found*
 - Fit the closed-left tire.
 - Fit the closed-right tire.
 - Fit the entire car and prepare for inspection
 - Determine a sweep path that covers the underside of car *// now estimated from car footprint*

Delayed Commitment Planning

Path command computed as a function of the shape of CAR_001, initially a variable at planning time.

CAR_001 is instantiated during run time from model fitting of sensor data
(ALIGN-ALONG (LINE-BISECT-FACE CAR_001) distance)



Summary

- Planning paths, tours and coverage
 - Utilizing interface with vehicle system:
 - location information and situational awareness
 - Path command interface with vehicle control
 - Representation of the environment: grids vis. Graphs
 - Search used to find paths that optimize goal achievement
 - Dijkstra's or A* search for paths
 - Combinatorial optimization for tours or coverage
 - Abstraction makes search tractable
 - Dynamic sensor or task updates
 - Modify representation
 - Incremental search update guaranteed goal achievement
 - Delayed commitment planning good for structured unknown environments

Workshop Schedule

Time	Topic	Presenter
8:30-8:45	Introductions and Course Overview	Moore
8:45-10:00	Unmanned Systems: Components and Architectures	Moore
10:00-10:30	Break	
10:30-12:30	Control Algorithms for Unmanned Systems	Berkemeier
12:30-1:30	Lunch	
1:30-3:30	Intelligent Behavior Generation	Flann
3:30-4:00	Break	
4:00-5:15	Future Directions in Unmanned Systems	All
5:15-5:30	Wrap-up	Moore

Workshop SC841 Unmanned Systems 101

Future Directions in Unmanned Systems

Panelists: Kevin L. Moore, Colorado School of Mines
Nicholas Flann, Utah State University
Matthew Berkemeier, Autonomous Solutions, Inc.

SPIE 2007 Security & Defense Symposium
Orlando Florida

9 April 2007

Workshop SC841

Comments on the Future for Unmanned Systems

Matthew Berkemeier, Autonomous Solutions, Inc.

SPIE 2007 Security & Defense Symposium
Orlando Florida

9 April 2007

Future Perspectives

- Mobility
 - Chaos
 - Big Dog
- Communications
 - Wireless video in challenging NLOS situations
- Intelligent Behaviors
 - Adding simple autonomous behaviors to
 - Make unmanned systems easier to operate
 - Make unmanned systems safer to operate
- Control Systems
 - Increasing robustness by fusing data from multiple sensors
 - Cameras becoming a more important sensor
- Multiple Vehicles, Multiple Users
 - Multiple UGVs
 - Cooperating UGVs, USVs, UUVs, and UAVs
 - Multiple users for 1 vehicle

Mobility



- Chaos
 - High mobility for short-term future.



- Big Dog
 - High mobility for long-term future.

Communications

- It is critical to have high-quality video transmission/reception in many situations where robots are operated remotely.
- This is obvious for tele-operation but also for reconnaissance/surveillance.
- Huge challenges due to
 - Long distances
 - Structures
 - Vehicle movement
 - Power restrictions on small UGVs
- Certain promising solutions are expensive or unavailable (for small, low-cost UGVs):
 - COFDM
 - UWB

Intelligent Behaviors

- Seems to be a strong desire to add small autonomous behaviors to existing UGVs
 - One purpose is to make UGVs easier, safer to operate.
 - EOD robots are currently difficult to use in many circumstances.
 - Add behaviors to automate movement of arm to within x cm of object of interest.
 - Add behavior to automate return travel of EOD robot after detonating an IED.
 - Increase dexterity of manipulators, and coordinate the base and arm movements.

Control Systems

- Fusing data from multiple sensors is becoming more and more common.
 - GPS alone is unreliable but GPS+IMU (dead reckoning) is much better.
 - GPS+IMU+vision is even better.
- Vision is being used more and more in a feedback loop for controlling unmanned systems
 - Sarnoff fuses visual data with IMU data to obtain data which is better than either system could provide alone.
 - Visual tracking, recognition, etc. getting better and better.

Multiple Vehicles, Multiple Users

- UAV and UGV can work together to keep on eye on an elusive dismount.
- Each has certain capabilities the other does not have.
- Sometimes more than one user will want to use an unmanned vehicle at the same time, and this may be possible without conflict.
- Need software to support multiple users.
- Similarly, sometimes a user will need to control multiple vehicles at the same time, while also perform his or her primary mission. Software is needed to assist the user.

Workshop SC841 Unmanned Systems 101

Future Directions in Unmanned Systems Cooperative Autonomy Concepts

Kevin L. Moore, Colorado School of Mines

**SPIE 2007 Security & Defense Symposium
Orlando Florida**

9 April 2007

Networked Sensors

- Through history, many technologies have become ubiquitous:
 - Microprocessors, motors
- Today a new technology has the same promise;
 - Networked sensors
- Advances in sensors, communications, nanotechnology, and biology enable the the concept
 - **“information about everything available everywhere”**
- Information is no longer a simple sensor output, but becomes multi-modal/multi-media
- Result is more complex engineered systems, with implications for
 - Hardware: distributed (wireless) and embedded
 - Software: parallel, distributed, intelligent algorithms
 - Systems: distributed and embedded, issues include:
 - Information management
 - Information processing
 - Decision making

Paradigm Change Brings Challenges

- Key challenge: the classical, lumped parameter ODE/PDE paradigm of systems and control is inadequate for future progress.

“... Essentially every thing done in the last [50] years of control theory rests on a common presumption of centrality [that all the information available about the system, and the calculations based on this information, take place at a single location]... ” “Survey of decentralized control methods for large scale systems,” Sandell, Varaiya, Athans, Sarnov, *IEEE Transactions on Automatic Control*, April 1978.

- From an algorithmic perspective, we need to turn to spatial-temporal methodologies

Paradigm Change Brings Opportunity

- One perspective on future challenges and opportunities for sensor networks can be summarized by:

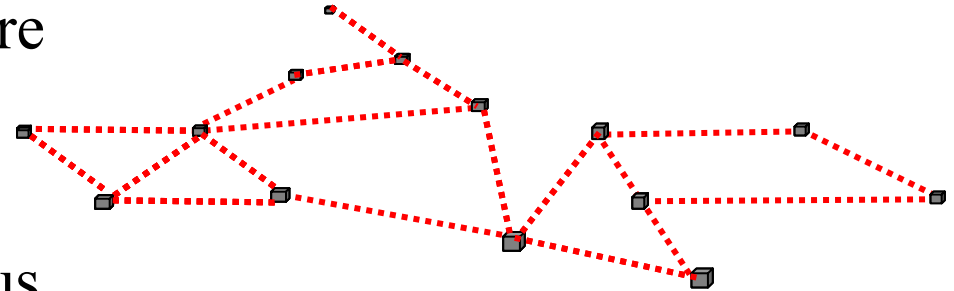
Only “...*through control will sensor networks achieve their value ...*,” Michael Bruns, VP A&D AS Process Automation, Siemens AG, DE, comments during Plenary Lecture, Wednesday, July 6, 2005 IFAC World Congress, Prague.

- From this perspective, I would like to comment on ideas related to what I call “Dynamic Resource Networks,” which includes the idea of “Mobile Actuator/Sensor Networks,” or MASNET

Dynamic Resource Networks

- **Network of “entities”**

- Communication infrastructure
- Entity-level functionality
- Implied global functionality
- Not necessarily homogeneous



- **Resource:**

- Primarily considering entities that are sensors
- “Bigger picture” includes actors as well

- **Dynamic**

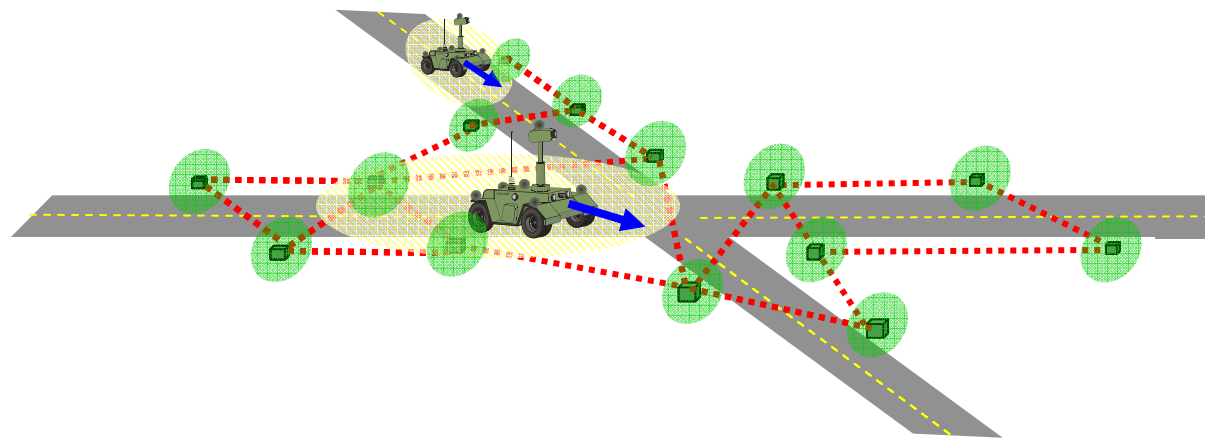
- Entities may be mobile
- Communication topology might be time-varying
- Data actively and deliberately shared among entities
- Decision-making and learning

A Prototypical Problem



COLORADO SCHOOL OF MINES

- Given:
 - Network of distributed, static sensors
 - Each instantiated with a perfect “classifier” that allows successful target identification if given perfect measurements
 - Each receives corrupted signatures due to sensor placement in the environment
 - Each sensor has a limited detection range and a limited communication range and BW between sensors
 - Each sensor is connected to some of the other sensors; assume at least one spanning tree exists in the graph; also assume that at least range between communicating sensors is known
 - Each sensor can compute the bearing and possible classification of three dominant targets in its region of detection
 - Multiple moving targets
 - Motion vector (velocity, attitude)
 - Characteristic signature (e.g., tank, car, motorcycle, etc.) with spatially-limited range of influence
- Find:
 - Estimated motion vector and classification of all targets moving through the sensor field

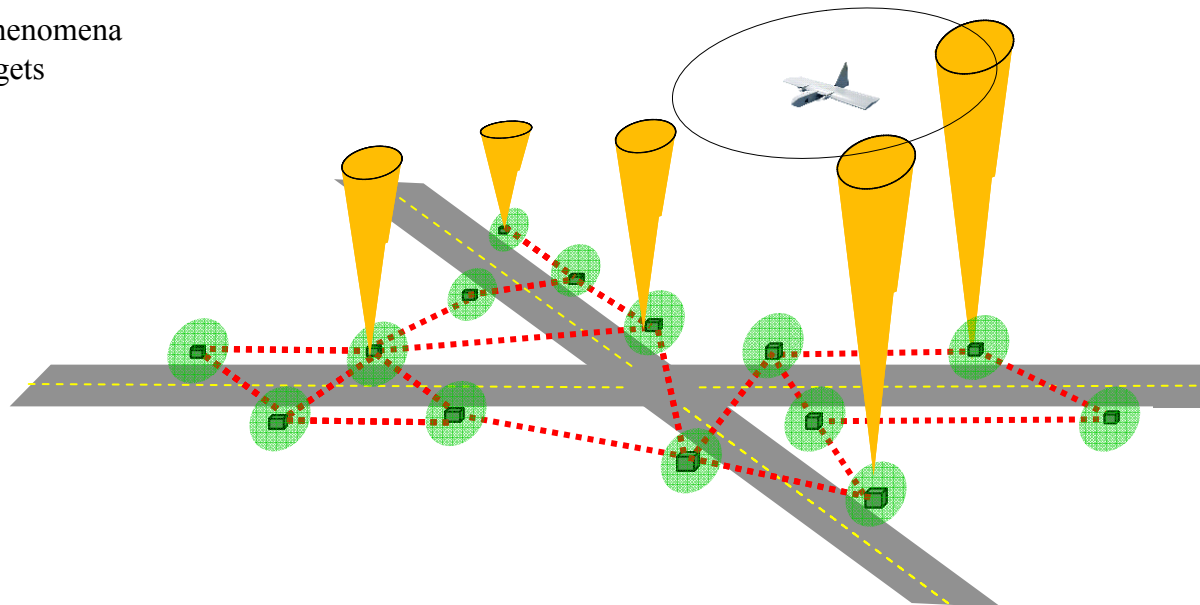


A Dual to the Prototypical Problem



COLORADO SCHOOL OF MINES

- Given:
 - Network of distributed, static targets
 - Each has a characteristic signature with spatially-limited range of influence, each related to the other in some (known) way (e.g., fixed network of weather stations for ecological monitoring in the wilderness)
 - Each target is connected to some of the other sensors; assume at least one spanning tree exists in the graph
 - Some targets have higher-level communications
 - Single moving sensor
 - Instantiated with a perfect “classifier” that allows successful classification of the phenomena represented by the targets if given perfect measurements
 - Receives corrupted signatures due to sensor placement in the environment
- Find:
 - Classification of the phenomena represented by the targets

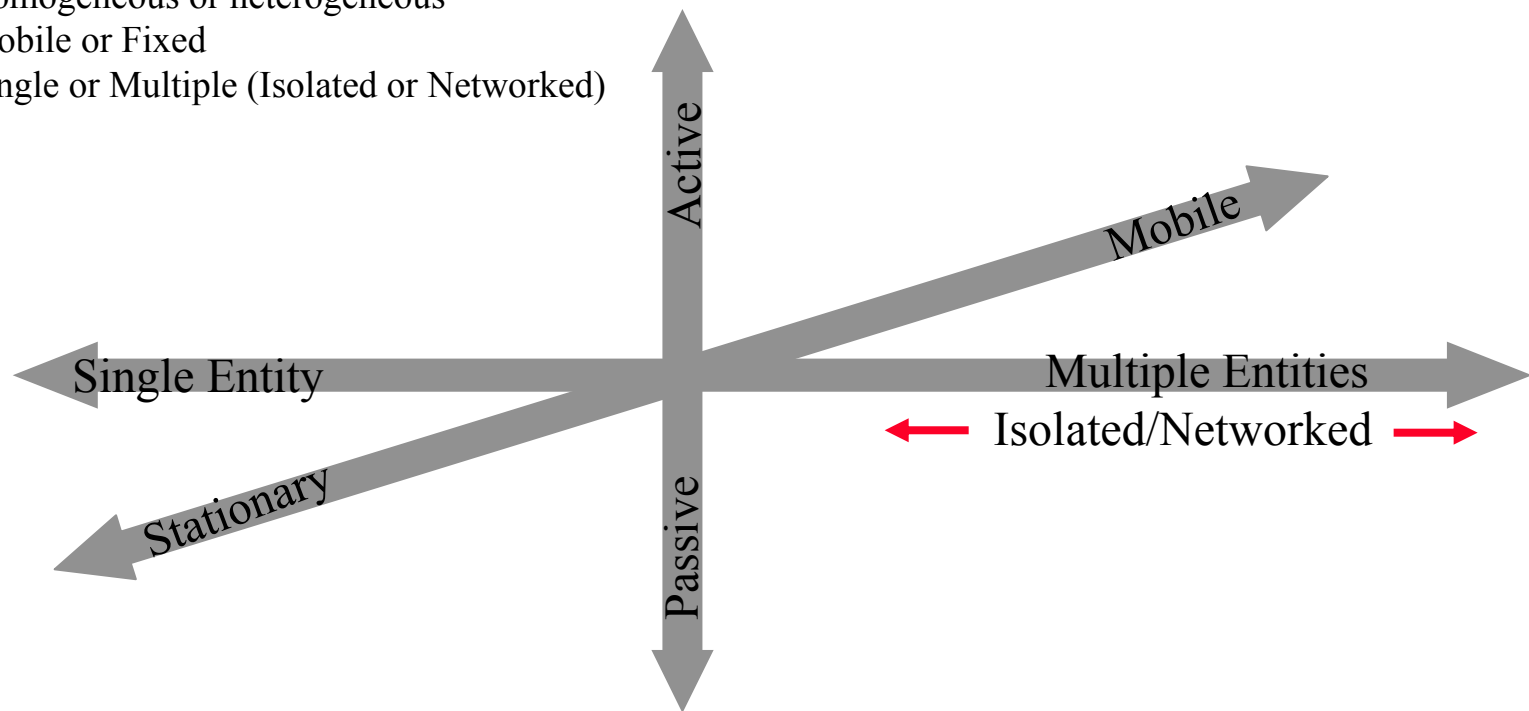


A Set of Related Problems



COLORADO SCHOOL OF MINES

- Sensors:
 - Passive (listen only) or Active (illuminates and listens)
 - Homogeneous or heterogeneous
 - Mobile or Fixed
 - Single or Multiple (Isolated or Networked)
- Target:
 - Passive (must be illuminated) or Active (generates signature)
 - Homogeneous or heterogeneous
 - Mobile or Fixed
 - Single or Multiple (Isolated or Networked)

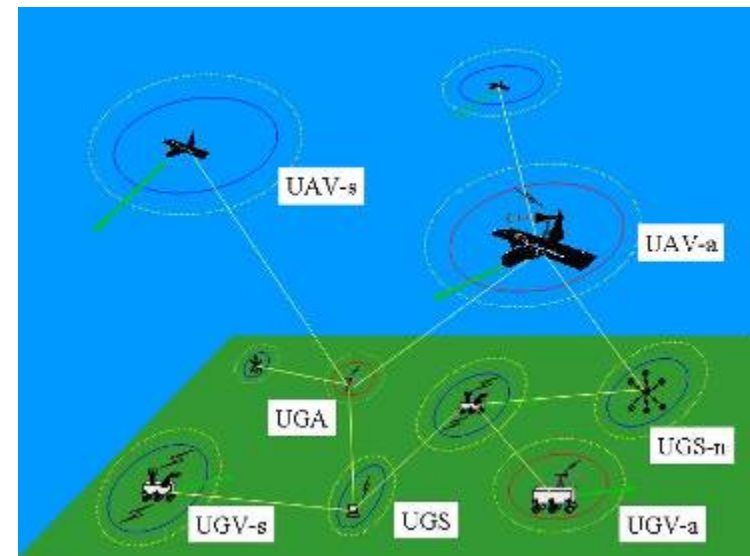


Dynamic Resource Networks

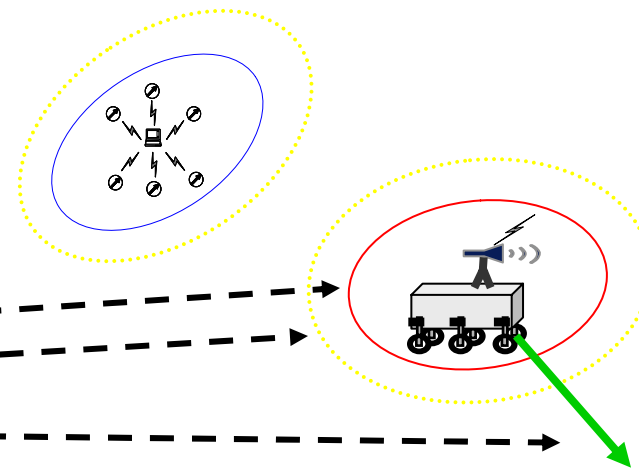


COLORADO SCHOOL OF MINES

- **Define seven types of resources:**
 - UGS: Unattended ground sensor
 - UGA: Unattended ground system with direct action capability (e.g., autonomous artillery)
 - UGS-n: Local network of multiple UGS
 - UAV-s: UAV used as a sensor
 - UAV-a: UAV with direct action capability (e.g., strike)
 - UGV-s: UGV used as sensor
 - UGV-a: UGV with direct action capability



- **Each entity is characterized by**
 - Type
 - Action Region
 - Sensor region of attraction (blue)
 - Actor region of influence (red)
 - Communication Envelope (yellow)
 - Mobility Vector (green)

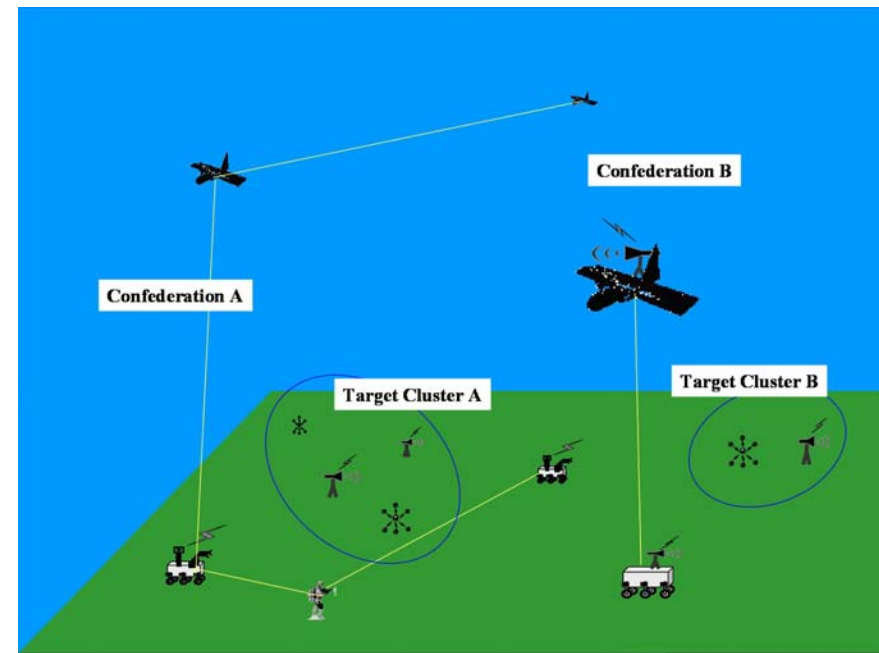
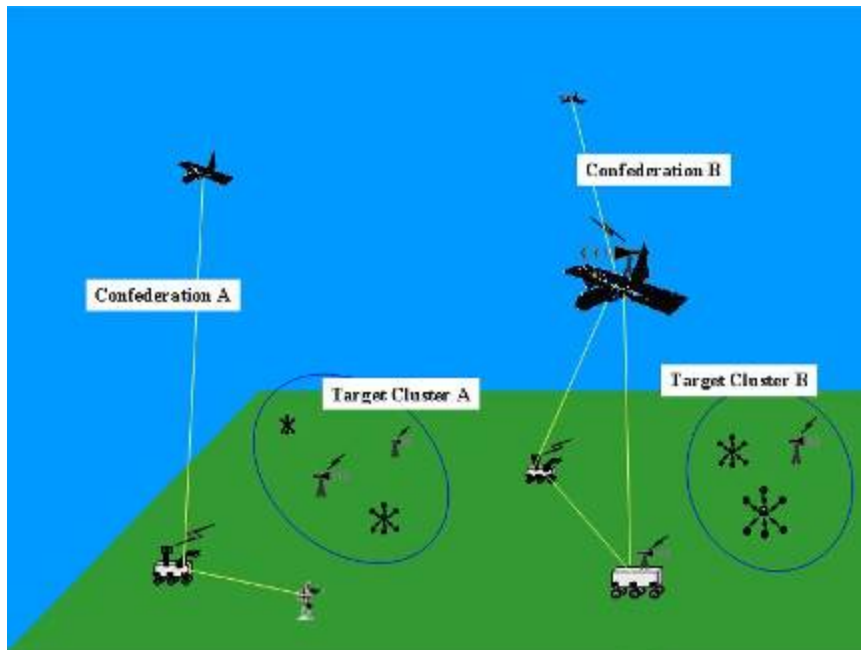


Motivating Example 1



COLORADO SCHOOL OF MINES

- **Autonomous confederation building, cooperative search, etc., adaptive to changes in environment**

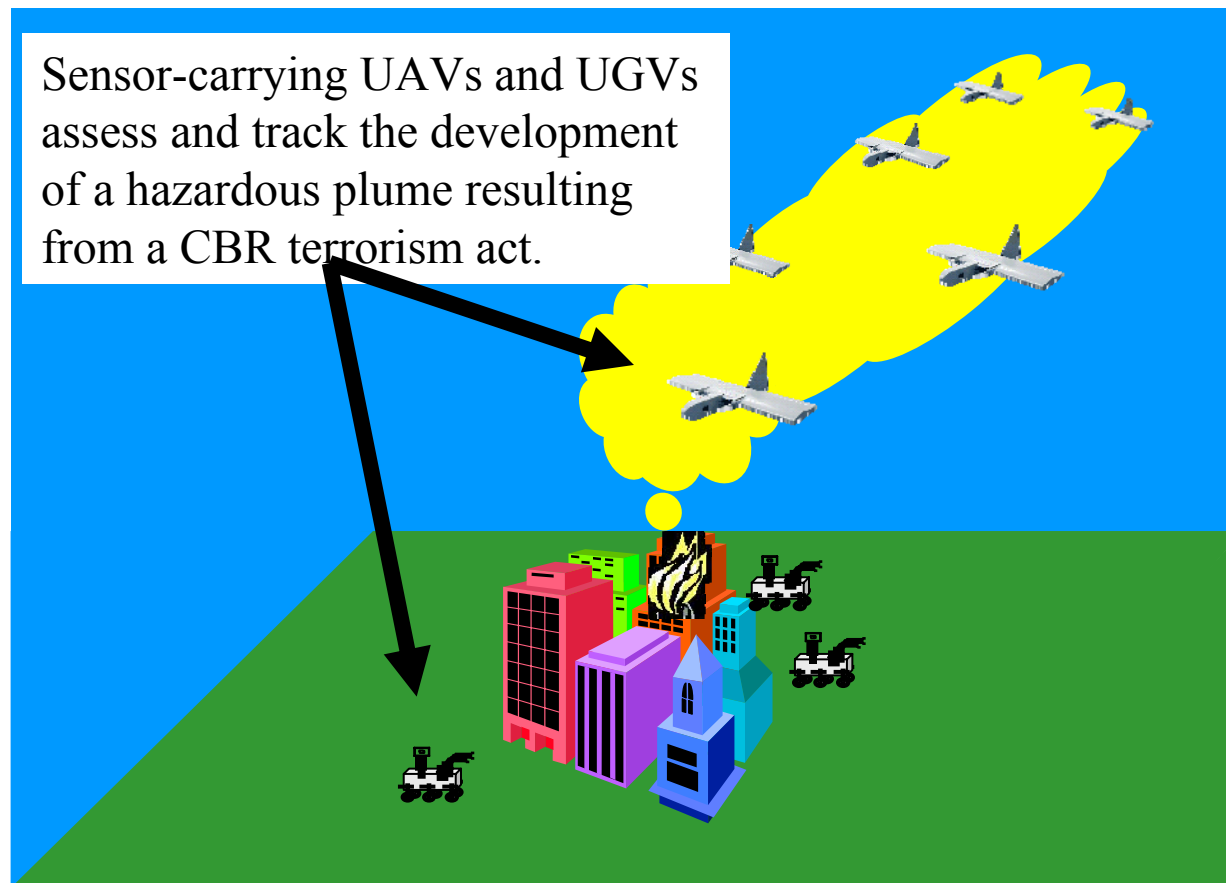


Motivating Example 2

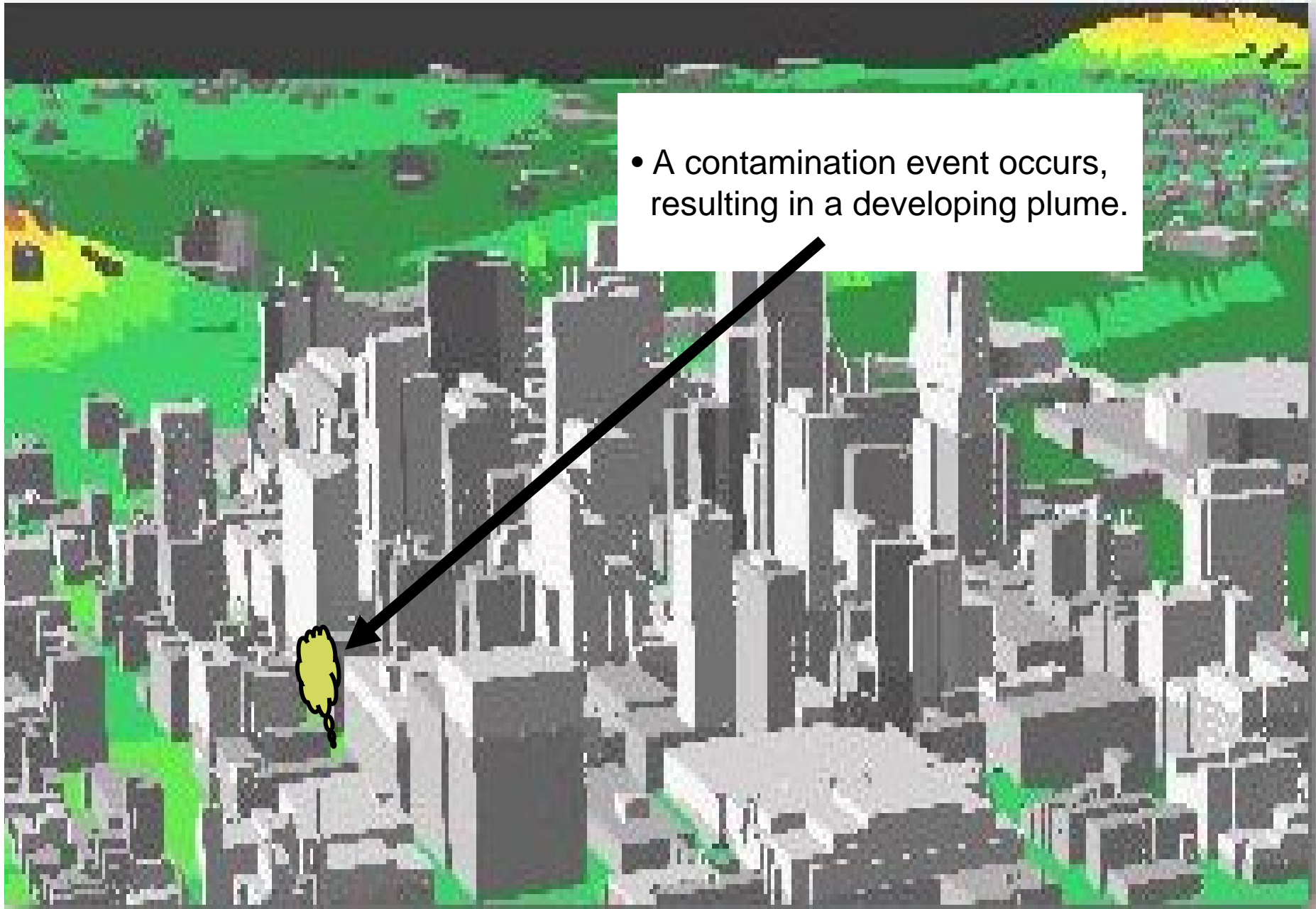


COLORADO SCHOOL OF MINES

- **Autonomous swarm for plume tracking**

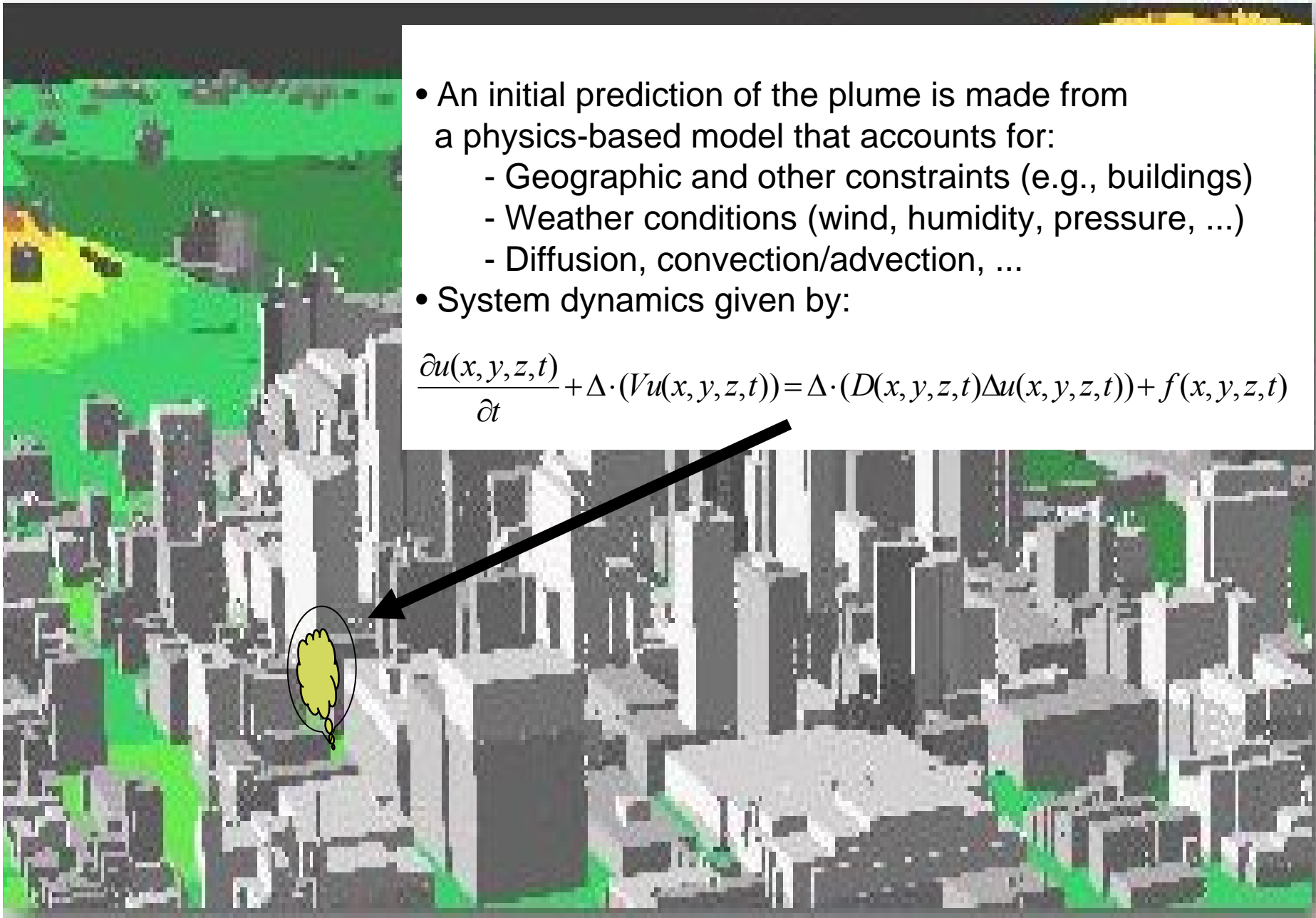


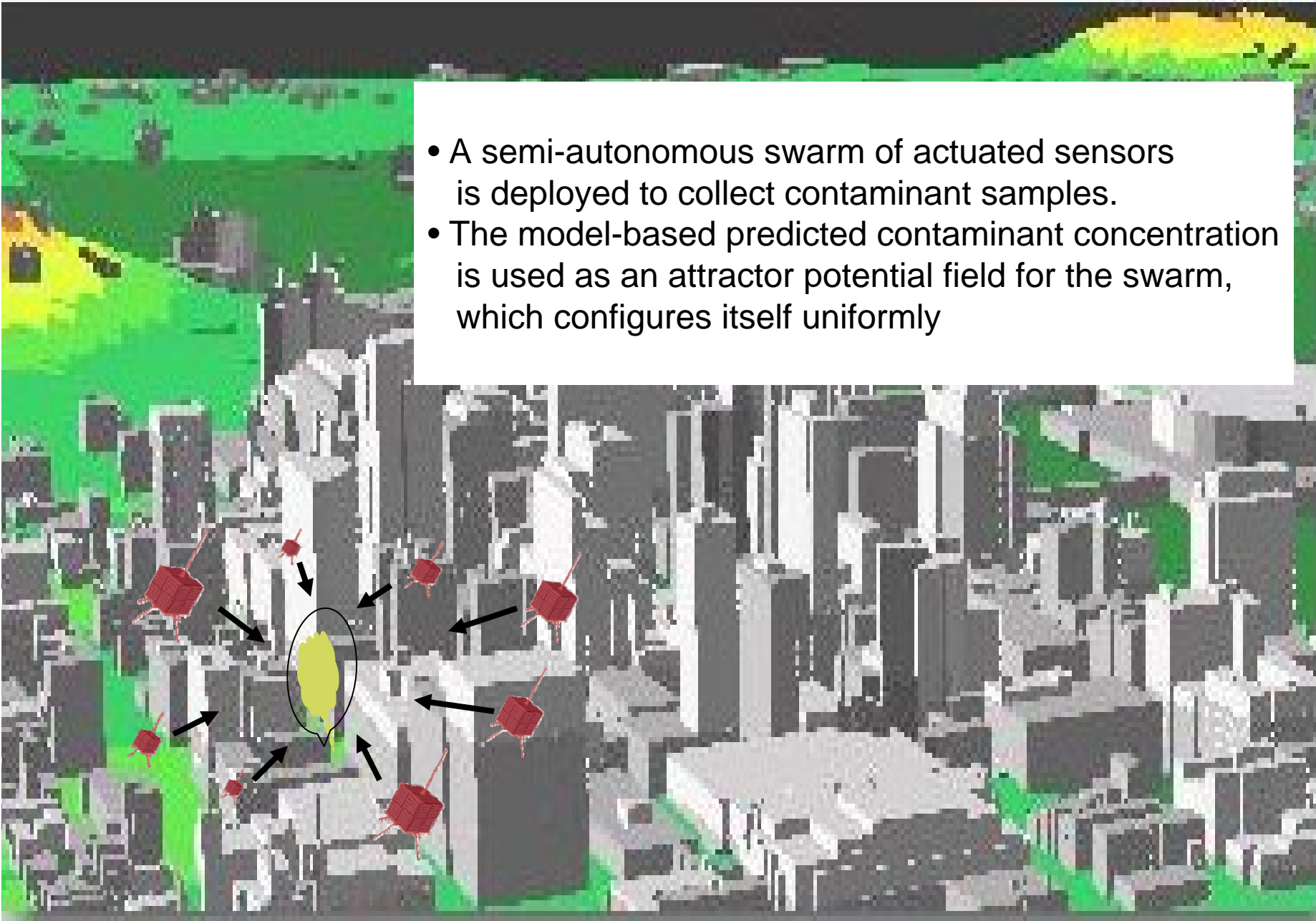
- A contamination event occurs, resulting in a developing plume.

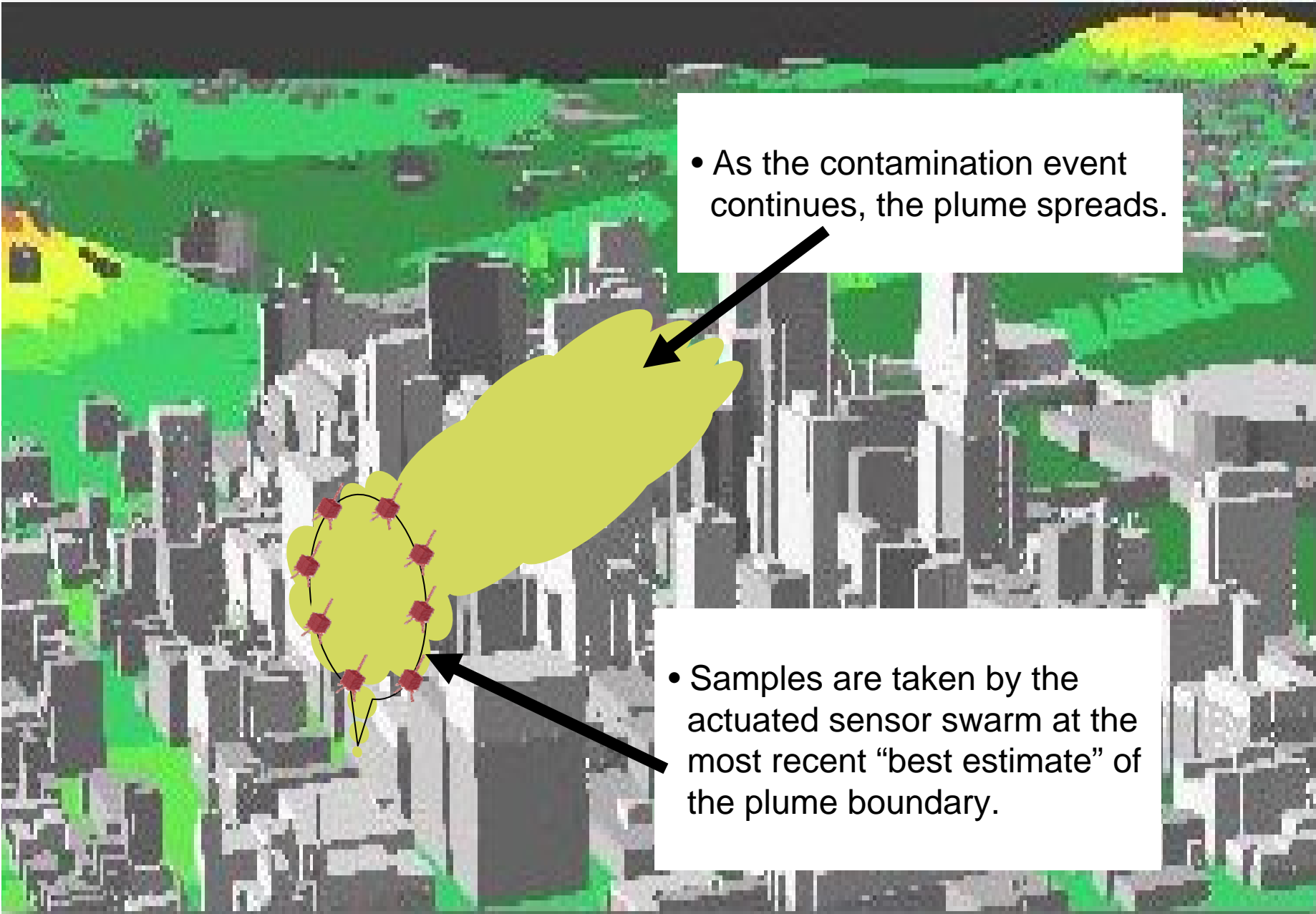


- An initial prediction of the plume is made from a physics-based model that accounts for:
 - Geographic and other constraints (e.g., buildings)
 - Weather conditions (wind, humidity, pressure, ...)
 - Diffusion, convection/advection, ...
- System dynamics given by:

$$\frac{\partial u(x, y, z, t)}{\partial t} + \Delta \cdot (Vu(x, y, z, t)) = \Delta \cdot (D(x, y, z, t)\Delta u(x, y, z, t)) + f(x, y, z, t)$$



- 
- A semi-autonomous swarm of actuated sensors is deployed to collect contaminant samples.
 - The model-based predicted contaminant concentration is used as an attractor potential field for the swarm, which configures itself uniformly

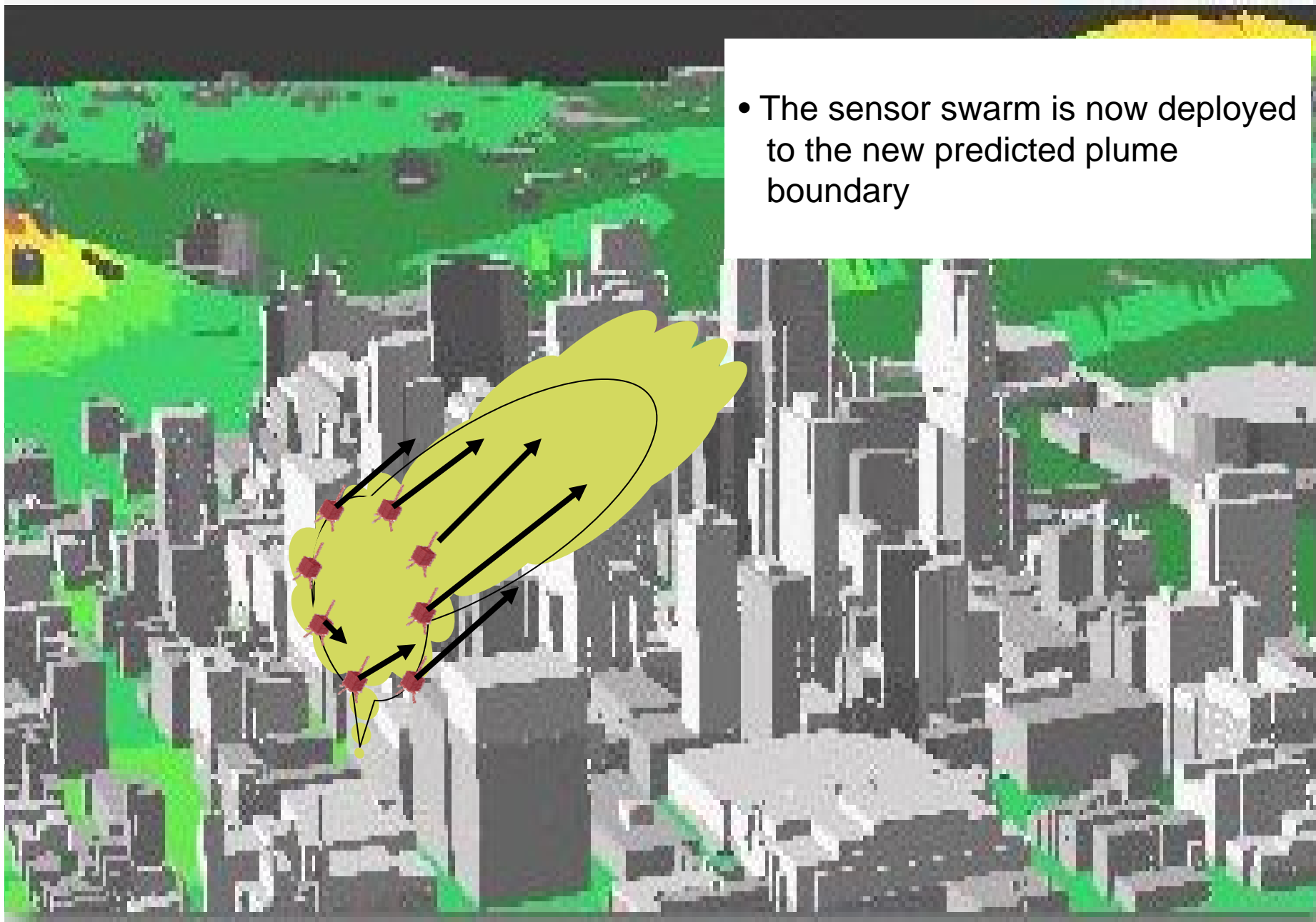
- 
- A 3D model of a city with a yellow plume spreading from a source. A swarm of red sensor nodes is positioned at the edge of the plume. Two text boxes with arrows point to the plume and the sensor swarm respectively.
- As the contamination event continues, the plume spreads.

- Samples are taken by the actuated sensor swarm at the most recent “best estimate” of the plume boundary.

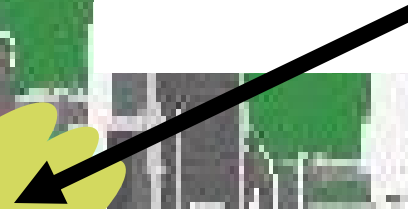
- Using the samples taken by the actuated sensor swarm, a new plume estimate is computed.



- The sensor swarm is now deployed to the new predicted plume boundary

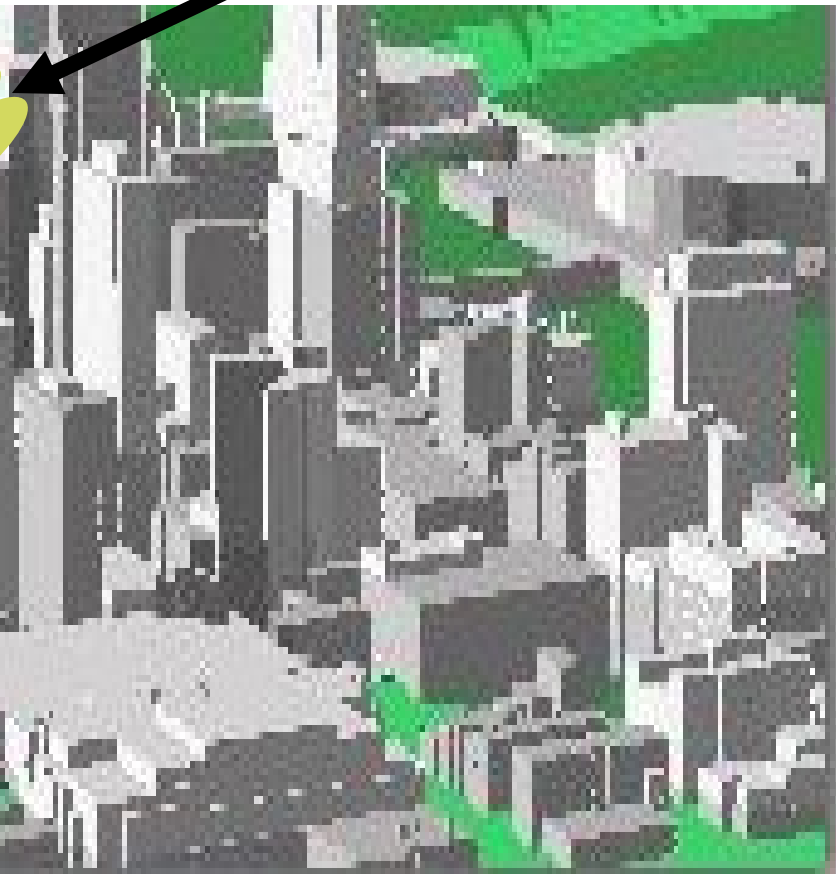


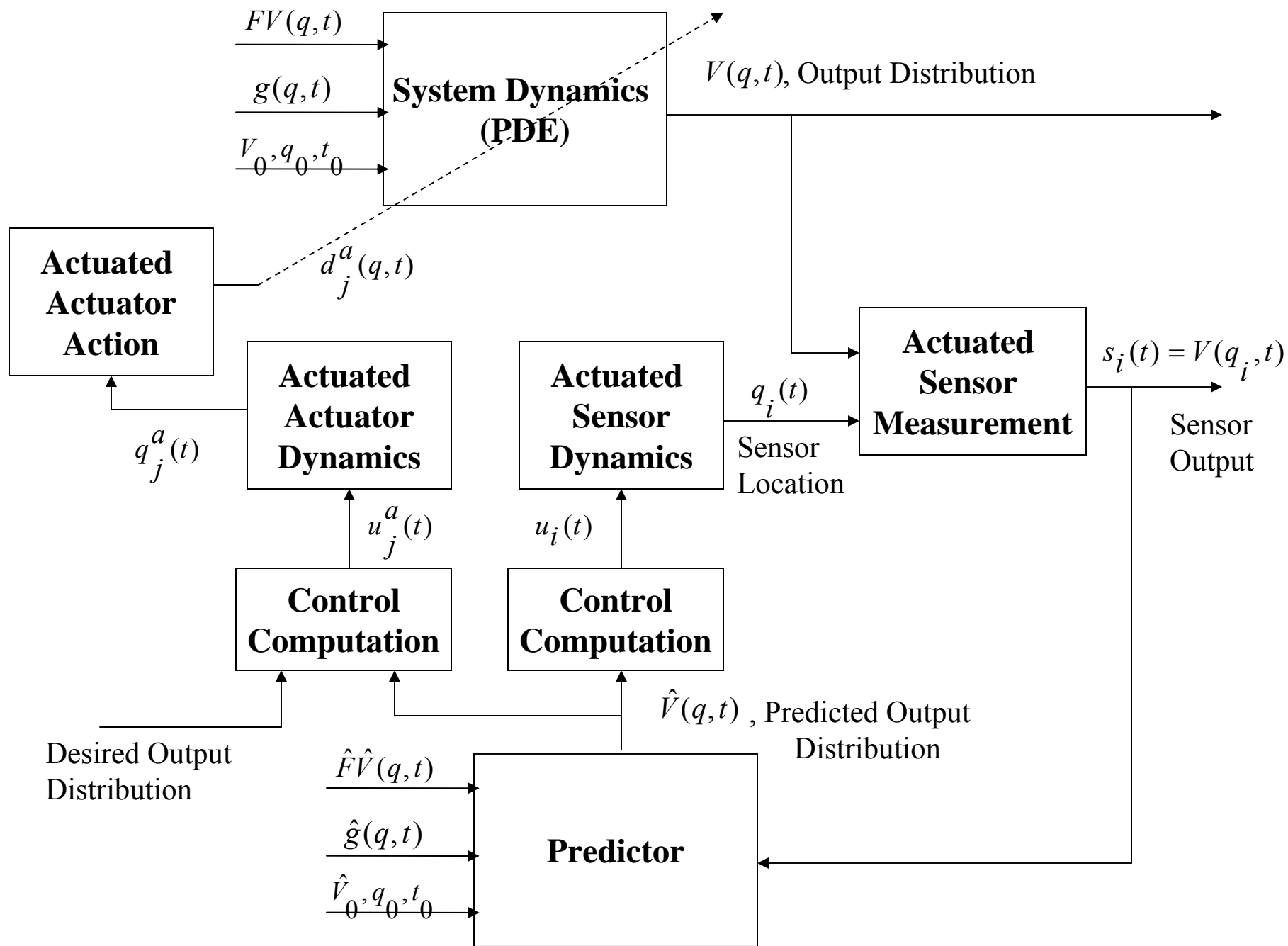
- The sensor swarm is now deployed to the new predicted plume boundary, where new samples are taken.





- Based on the plume features predicted by the swarm's measurements, a network of mobile actuators is deployed to apply dispersal agents to counteract the effect of the contaminant.





Preliminary results using Centroidal Voronoi Tessellations (due to YangQuan Chen at USU)

Simulation and Results

Diff-MAS2D is used as the simulation platform for our implementation. The area concerned is given by $\Omega = \{(x, y) | 0 \leq x \leq 1, 0 \leq y \leq 1\}$.

The system with control input is modelled as

$$\frac{\partial \rho(x, y, t)}{\partial t} = k \left(\frac{\partial^2 \rho(x, y, t)}{\partial x^2} + \frac{\partial^2 \rho(x, y, t)}{\partial y^2} \right) + f_c(x, y, t) + f_d(x, y, t), \quad (9)$$

where $k = 0.01$ and the boundary condition is given by

Plant Dynamics

$$\frac{\partial u}{\partial n} = 0.$$

Control Input

Disturbance

The stationary pollution source is modelled as a point disturbance f_d to the the PDE system (9) with its position at $(0.75, 0.35)$ and

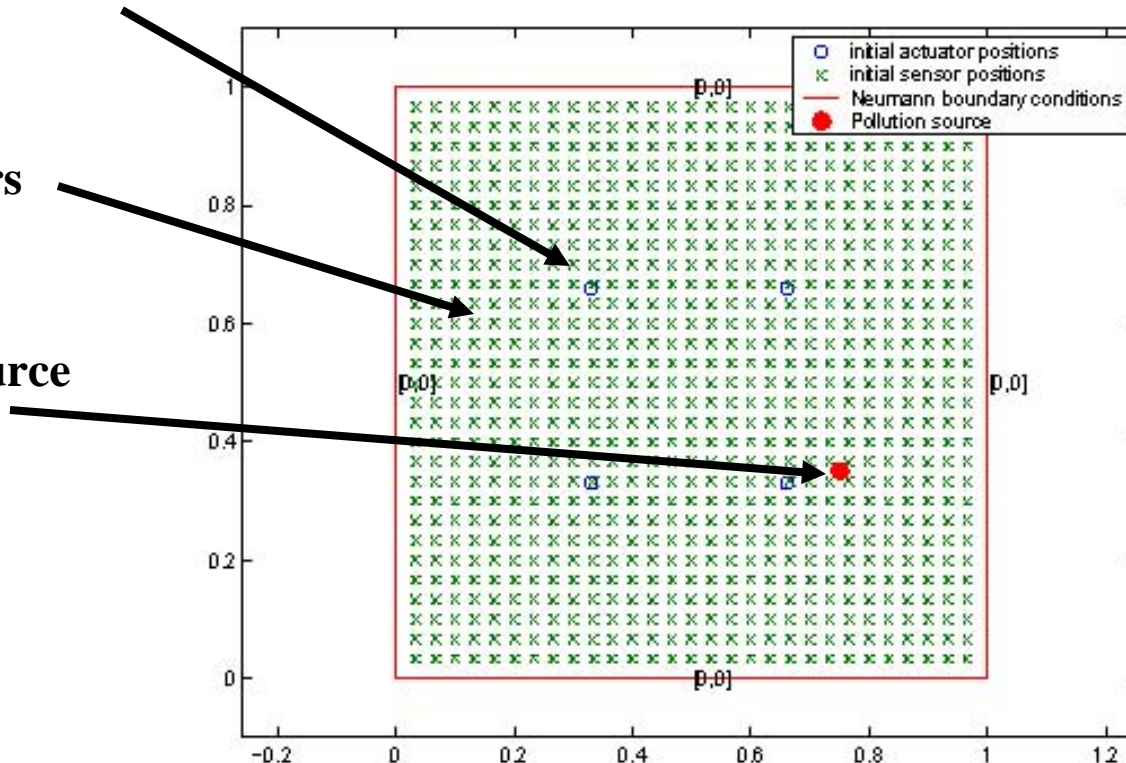
$$f_d(t) = 20e^{-t}|_{(x=0.75, y=0.35)}.$$

In our simulation, we assume that once deployed, the sensors remain static. There are 29×29 sensors evenly distributed in a square area $(0, 1)^2$ and they form a mesh over the area. There are 4 robots that can release the neutralizing chemicals. The amount of chemicals each robot releases is proportional to the average pollutant concentration in the Voronoi cell belonging to that robot.

Mobile Robots (Actuators)

Fixed Sensors

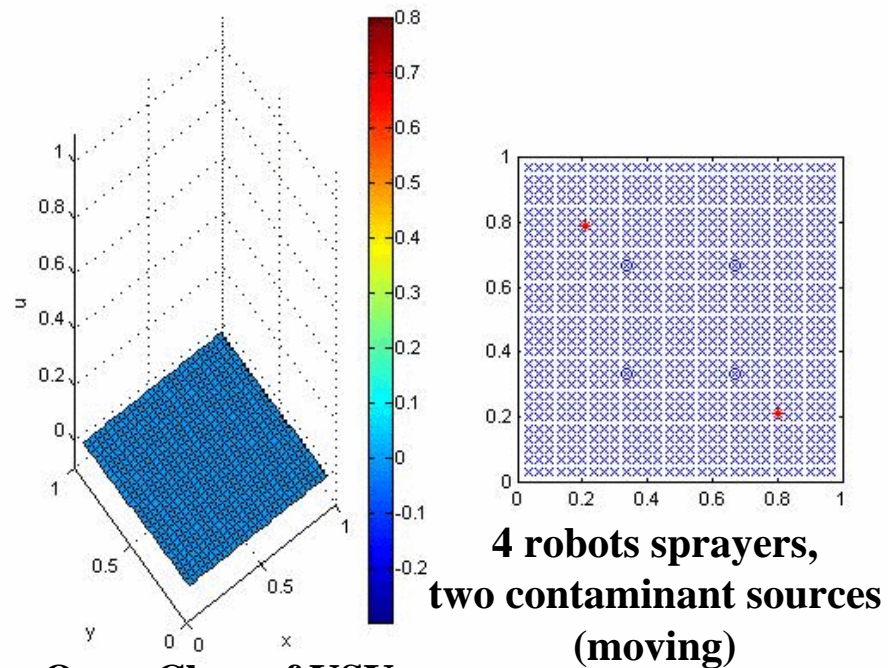
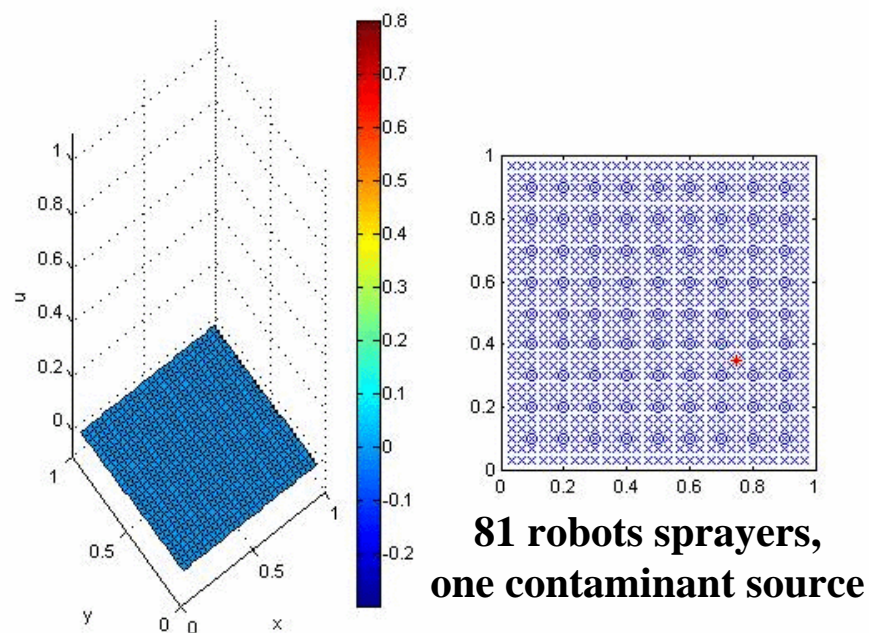
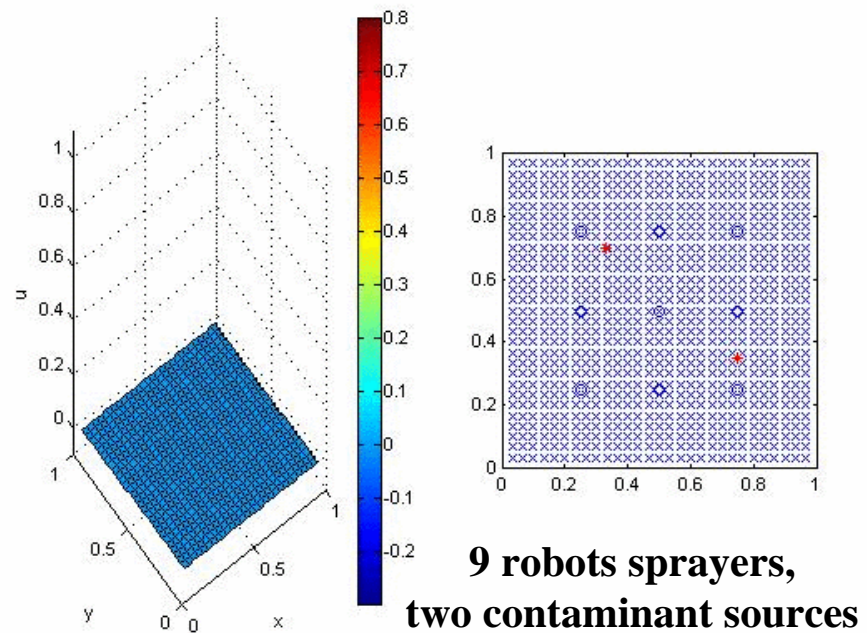
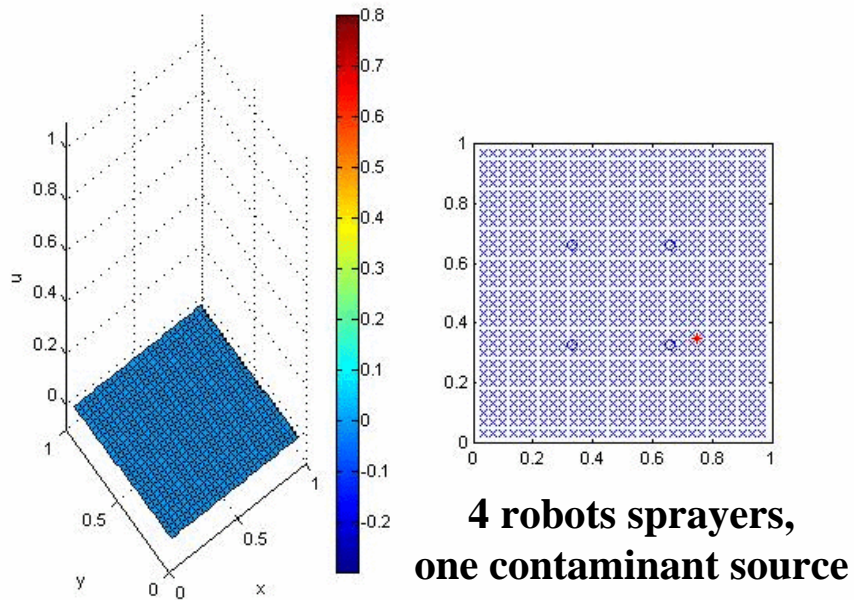
Diffusion Source



Initial layout of actuators and sensors.

Strategy:

- 1) Form Voronoi tessellation
- 2) Move each robot to the mass centroid of its region
- 3) Spray neutralizing chemical in amount proportional to concentration in region



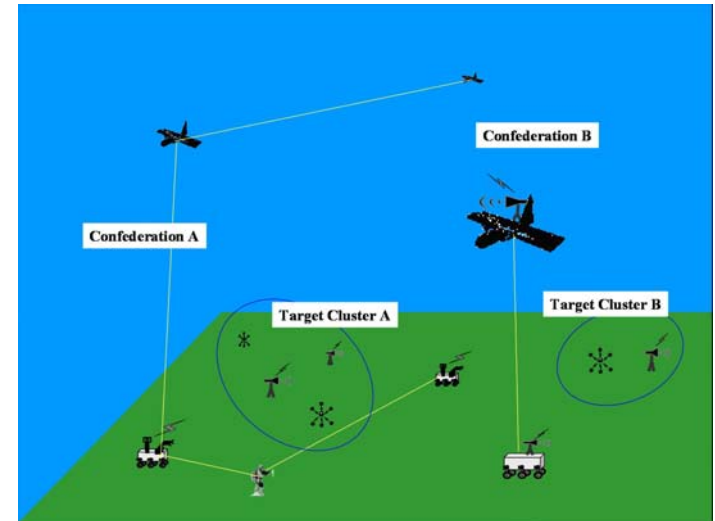
Simulations thanks to Prof. YangQuan Chen of USU

Consensus Variable Approach to Distributed Consensus



COLORADO SCHOOL OF MINES

- **Assertion:**
 - Multi-agent coordination requires that *some* information must be shared
- **The idea:**
 - Identify the essential information, call it the *coordination or consensus variable*.
 - Encode this variable in a distributed dynamical system and come to consensus about its value
- **Examples:**
 - Heading angles
 - Phase of a periodic signal
 - Mission timings
- **In the following we build on work by Randy Beard (BYU), Wei Ren (USU), others to use consensus variables for cooperative autonomy**





Consensus Variables

COLORADO SCHOOL OF MINES

- Suppose we have N agents with a shared *global* consensus variable ξ
- Each agent has a *local* value of the variable given as ξ_i
- Each agent updates their local value based on the values of the agents that they can communicate with

$$\dot{\xi}_i(t) = - \sum_{j=1}^N k_{ij}(t) G_{ij}(t) (\xi_i(t) - \xi_j(t))$$

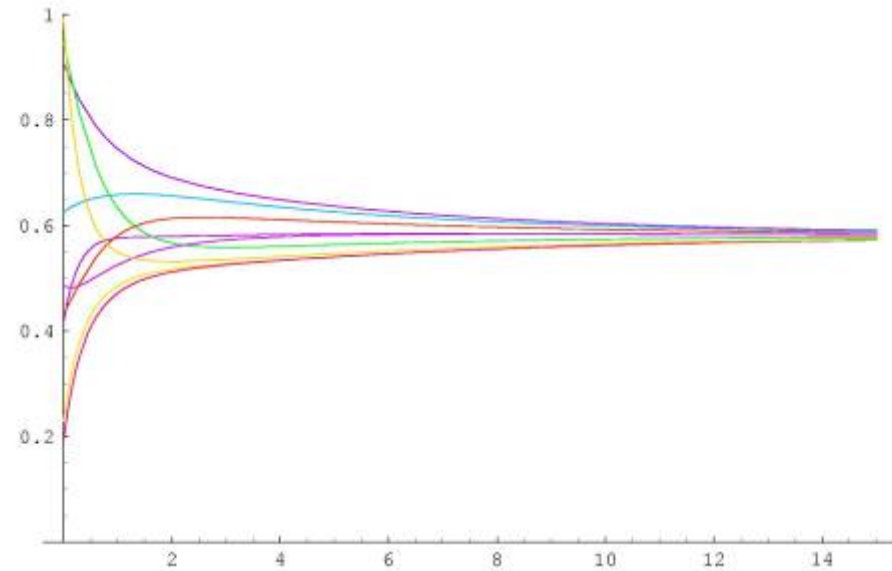
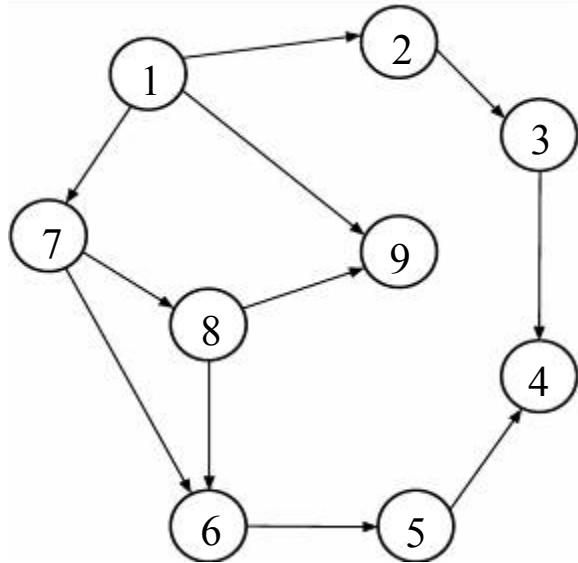
where k_{ij} are gains and G_{ij} defines the communication topology graph of the system of agents

- Key result from literature: If the graph has a spanning tree then for all i $\xi_i \rightarrow \xi^*$

Example: Single Consensus Variable



COLORADO SCHOOL OF MINES



$$\begin{pmatrix} \dot{\xi}_1 \\ \dot{\xi}_2 \\ \dot{\xi}_3 \\ \dot{\xi}_4 \\ \dot{\xi}_5 \\ \dot{\xi}_6 \\ \dot{\xi}_7 \\ \dot{\xi}_8 \\ \dot{\xi}_9 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ k_{21} & -k_{21} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & k_{32} & -k_{32} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{43} & -k_{43} - k_{45} & k_{54} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -k_{56} & k_{56} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -k_{67} - k_{68} & k_{67} & k_{68} & 0 \\ k_{71} & 0 & 0 & 0 & 0 & 0 & -k_{71} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & k_{87} & -k_{87} & 0 \\ k_{91} & 0 & 0 & 0 & 0 & 0 & 0 & k_{98} & -k_{91} - k_{98} \end{bmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \\ \xi_5 \\ \xi_6 \\ \xi_7 \\ \xi_8 \\ \xi_9 \end{pmatrix}$$

Extension 1 – Forced Consensus

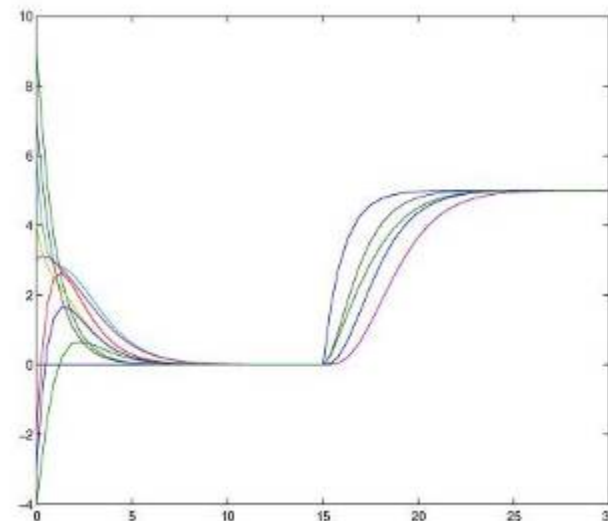
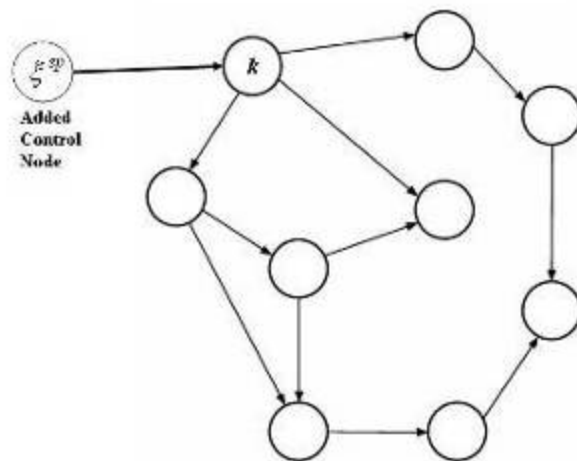
(work with Dennis Lucarelli at APL)



COLORADO SCHOOL OF MINES

- Forced Consensus
 - Sometimes we may like to force all the nodes to follow a hard constraint
 - This can be done by injecting an input into a node and introducing feedback:

$$\dot{\xi}_i(t) = -\sum_{j=1}^N k_{ij}(t)G_{ij}(t)(\xi_i - \xi_j) - k_p(\xi^{SP} - \xi_i)$$

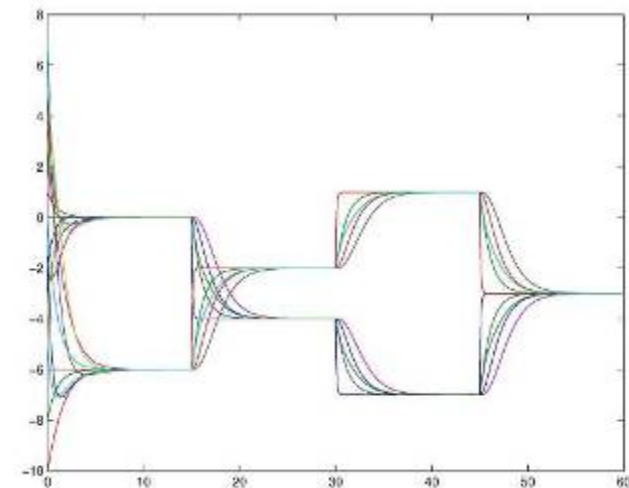
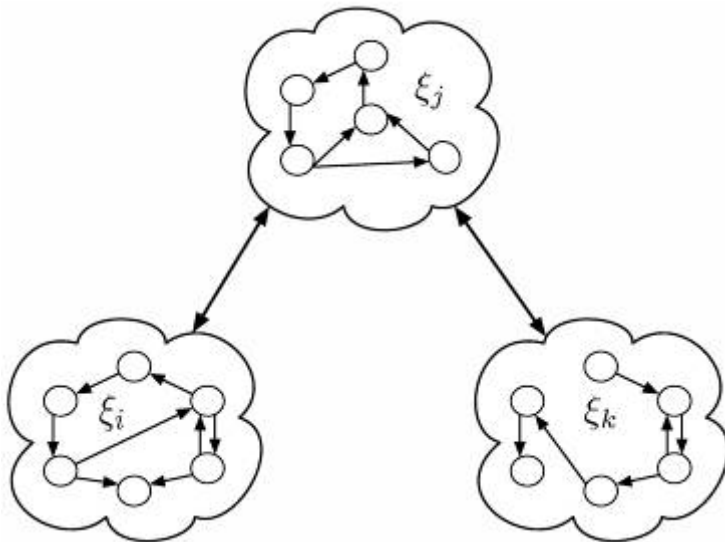


Extension 2 – Multiple, Constrained Consensus (work with Dennis Lucarelli at APL)



COLORADO SCHOOL OF MINES

- Often we will have multiple consensus variables in a given problem
- It can be useful to enforce constraints between these variables, specifically, to have $\xi_i = \xi_j + \Delta_{ij}$
- Again we can give a feedback control strategy to achieve this type of constrained consensus between groups of agents



Step changes in Δ_{ij}

Extension 3 – Higher-Order Consensus

(work with Wie Ren, YangQuan Chen at USU)



- Can generalize to higher-order as follows. Let

$$\dot{\xi}_i^{(1)} = \xi_i^0$$

$$\dot{\xi}_i^{(2)} = \xi_i^1$$

⋮

$$\dot{\xi}_i^{(l)}(t) = -\sum_{j=1}^N k_{ij}(t)G_{ij}(t)\left[\sum_{k=0}^{l-1} \gamma_k (\xi_i^{(k)} - \xi_j^{(k)})\right]$$

- Then, if there is a spanning tree (and system is stable) you will get

$$\xi_i^{(l)} \rightarrow \xi_i^{(l)*} \quad \text{for all } i$$

$$\xi_i^{(l-1)} \rightarrow t \xi_i^{(l-1)*} + \gamma \quad \text{for all } i$$

$$\xi_i^{(l-2)} \rightarrow t^2 \xi_i^{(l-2)*} + t\gamma + \beta \quad \text{for all } i$$

⋮

- Stability depends on the gains γ_k



Example – Higher-Order Consensus

COLORADO SCHOOL OF MINES

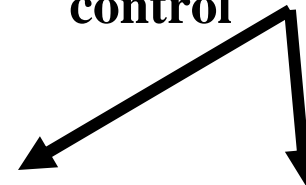
- For example, consider a third-order consensus problem, applied to a formation control problem with five vehicles, with one vehicle having an acceleration setpoint input

$$\dot{x}_i = v_i$$

$$\dot{v}_i = a_i$$

$$\dot{a}_i = - \sum_{j=1}^n g_{ij} k_{ij} \{ \gamma_0 [(x_i - \delta_i) - (x_j - \delta_j)] + \gamma_1 (v_i - v_j) + \gamma_2 (a_i - a_j) \} - \alpha (a_i - a_i^*)$$

Enables formation control



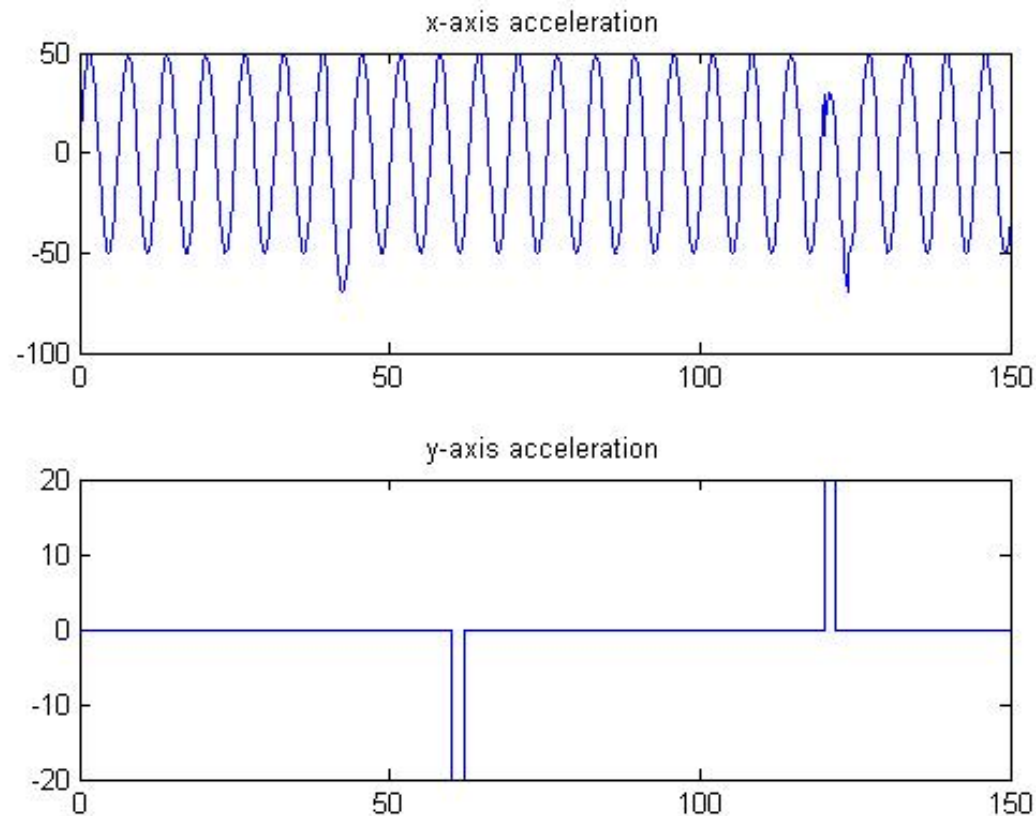
Acceleration Input

Example – Higher-Order Consensus



COLORADO SCHOOL OF MINES

- Suppose the “leader node” sees the following acceleration input profile:

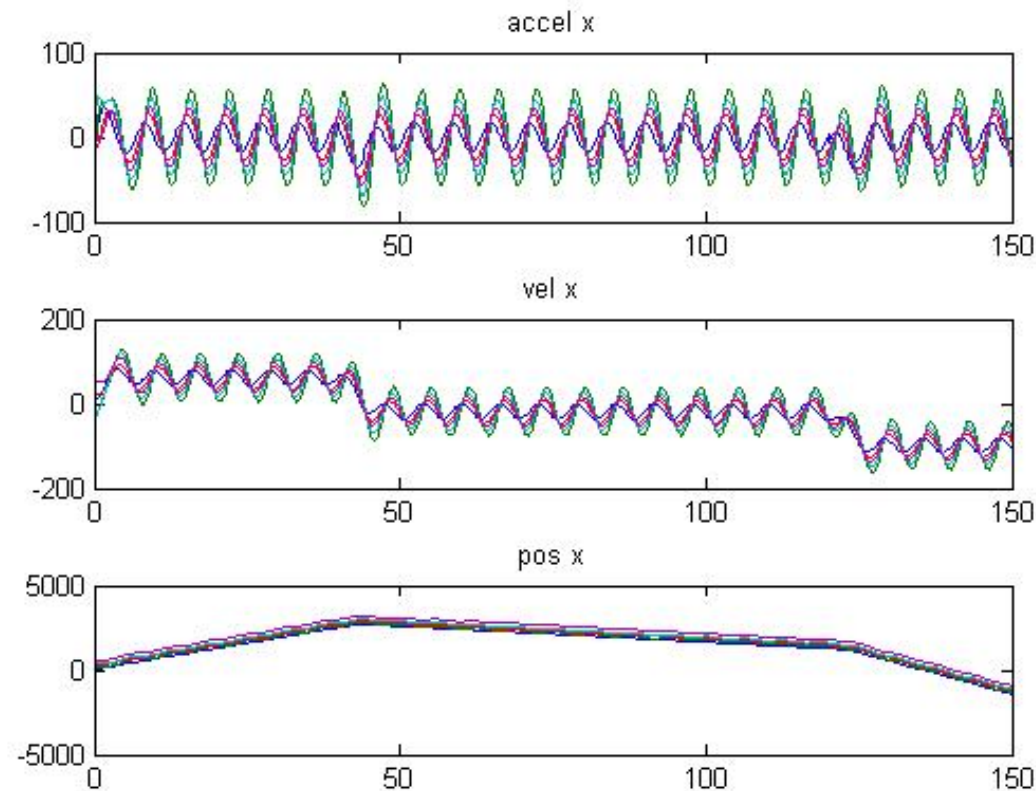


Extension 3 – Higher-Order Consensus



COLORADO SCHOOL OF MINES

- The resulting x-axis trajectory shows the effect of the higher-order consensus algorithm

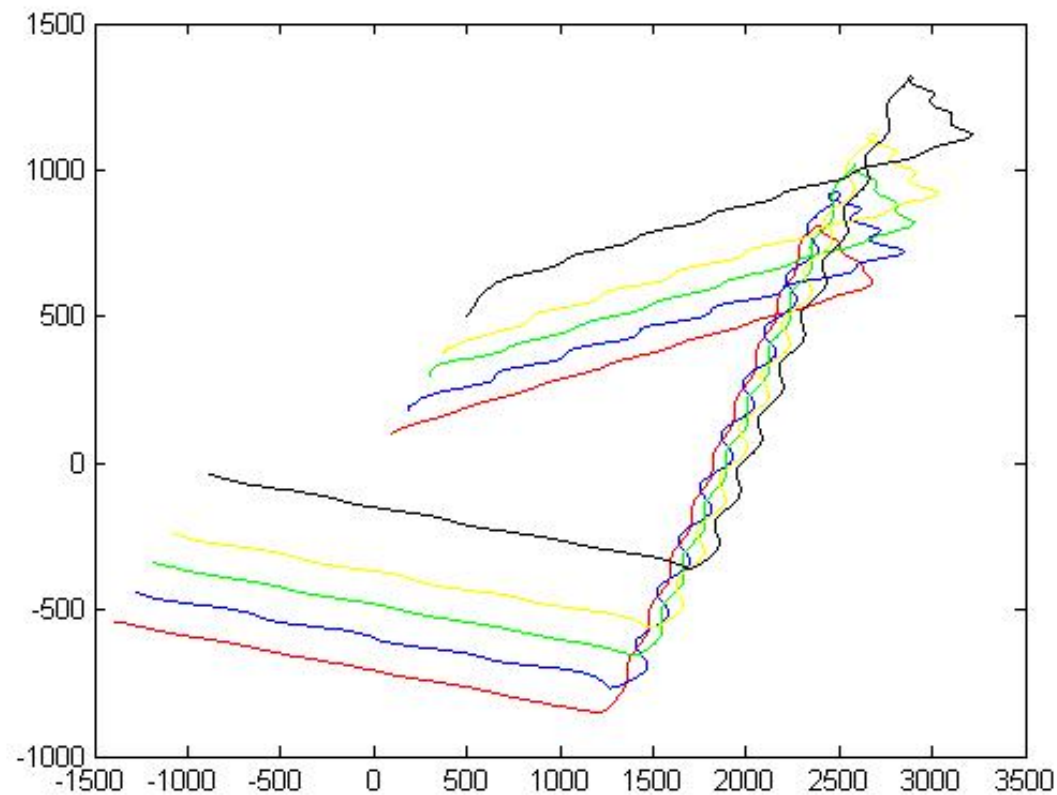


Example – Higher-Order Consensus



COLORADO SCHOOL OF MINES

- The resulting paths look like:

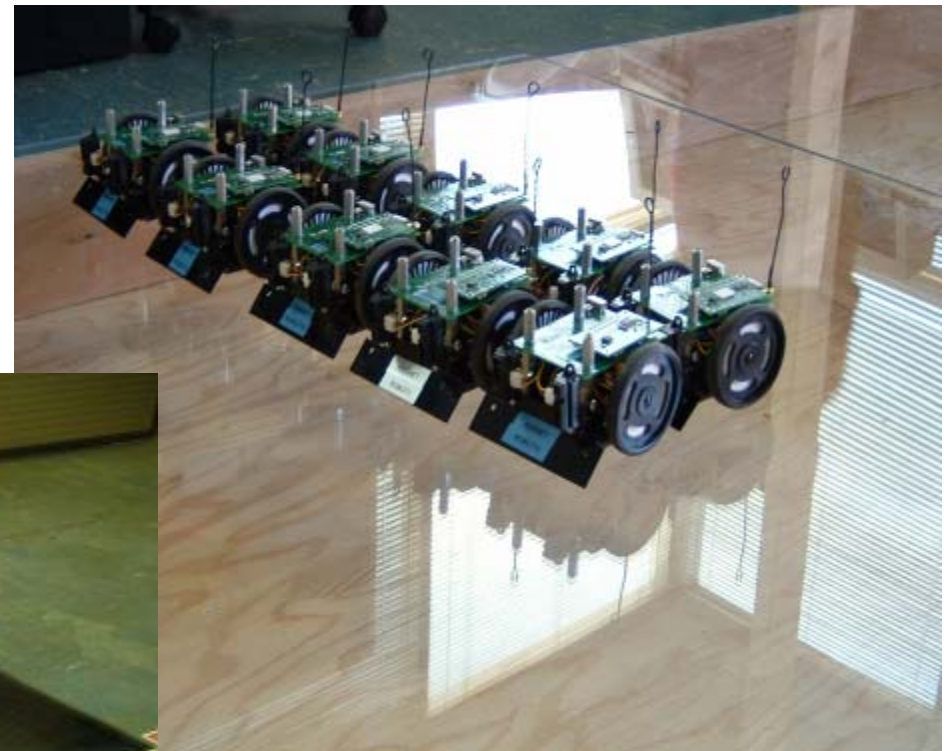


Mote-Based Distributed Robots



COLORADO SCHOOL OF MINES

Prototype plume tracking testbed



Example: Consensus with a Leader

(movies thanks to USU CSOIS, Wei Ren and YangQuan Chen)



COLORADO SCHOOL OF MINES

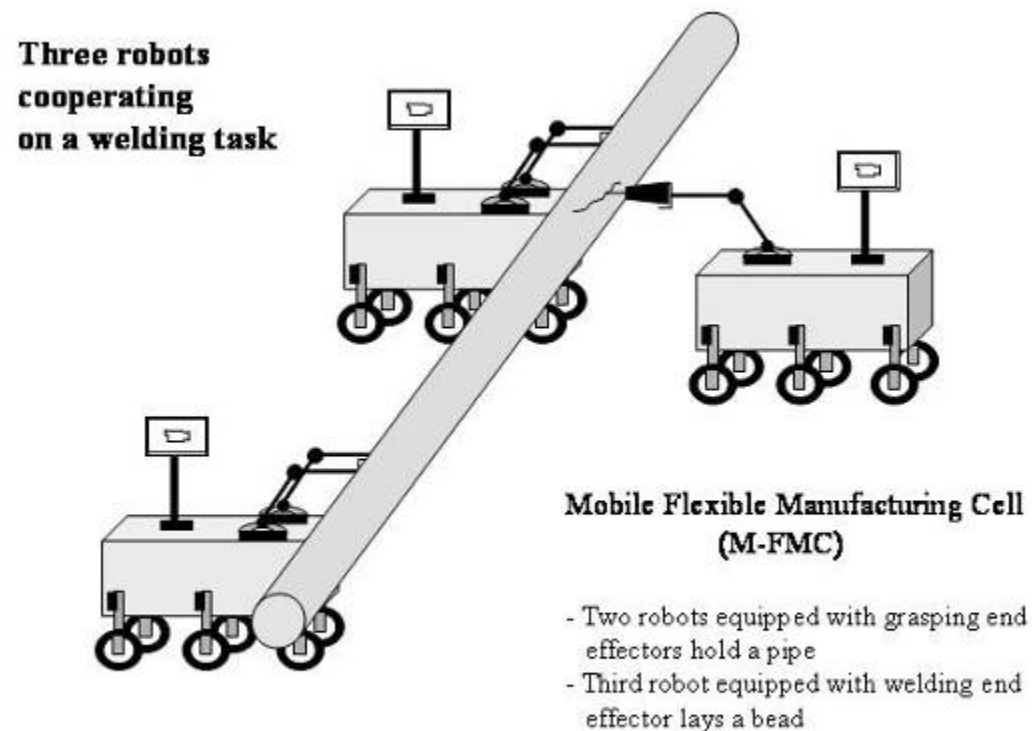




Cooperative Autonomy...

COLORADO SCHOOL OF MINES

- Exercise left to the reader!



Workshop SC841 Unmanned Systems 101

Future Directions in Unmanned Systems Multiple Agent Environments

Presenter: **Nicholas S. Flann**, Utah State University, Autonomous Solutions

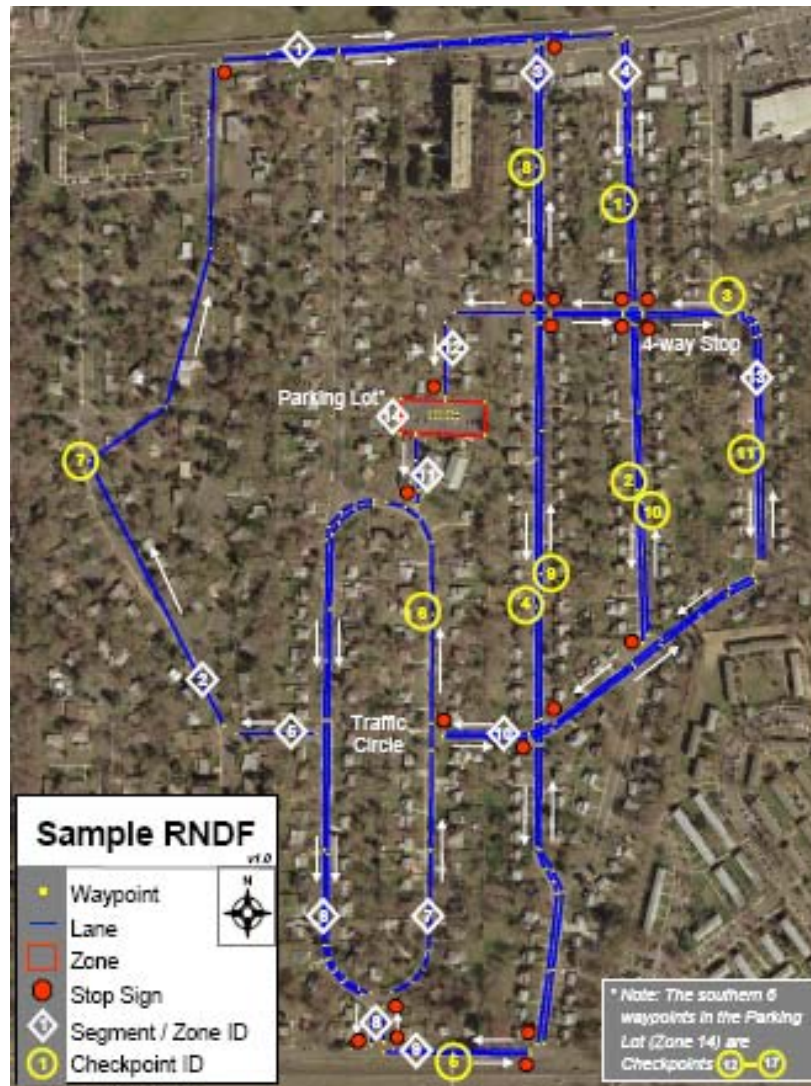
SPIE 2007 Security & Defense Symposium
Orlando Florida

9 April 2007

Multiple Agent Environments

- DARPA Urban Grand Challenge
 - Multiple unmanned systems in same environment
 - Cooperative and competitive
 - Urban driving
 - Rules of the road conventions
 - Hidden intentions of other agents
 - Competitive agents
 - Complex dynamic environment
 - Manned/Unmanned mix
- Simplifications
 - No pedestrians
 - Limited scenarios
- Goal Unmanned transportation system using existing infrastructure

Example UGC Map



- Example map file
- Routes not drivable
- Stop intersections shown
- Mission waypoints
- Free travel zones
- Two-way roads

Challenges

- Accurate situational awareness
 - Trajectory of dynamic vehicle
 - Intentionality
 - Goals of other agents
 - Behaviors of other agents
 - Environment
 - Road signs, lanes
 - Rules of the road (stop sign precedence)
- Robustness
 - Unknown scenarios
 - Complex agent interactions with intentionality

Enabling Technology/Methodology

- High fidelity sensor systems
 - Stereo vision (Sarnoff)
 - Vehicle object shape/recognition
 - Trajectory + state signals etc. recognition
- Modeling
 - Physical movements
 - Intentionality of other agents
- Simulation & Learning for Robustness
 - Scenarios organized by complexity
 - Problem: passing, 2 way, 3 way, 4 way stops etc.
 - Number of other vehicles
 - Behavior generation of other agents
 - Learning
 - Reinforcement learning methods
 - Generalization, automated feature extraction

Workshop Schedule

Time	Topic	Presenter
8:30-8:45	Introductions and Course Overview	Moore
8:45-10:00	Unmanned Systems: Components and Architectures	Moore
10:00-10:30	Break	
10:30-12:30	Control Algorithms for Unmanned Systems	Berkemeier
12:30-1:30	Lunch	
1:30-3:30	Intelligent Behavior Generation	Flann
3:30-4:00	Break	
4:00-5:15	Future Directions in Unmanned Systems	All
5:15-5:30	Wrap-up	Moore

Workshop Summary

- This course was intended to give a basic introduction to the technical aspects of unmanned systems.
- The course emphasized:
 - Systems engineering perspective on the conceptual design and integration of the components of an unmanned system for a specific application.
 - Specify the components needed to implement an unmanned system
 - Task-appropriate architectures to integrate and coordinate the algorithms in an unmanned system.
 - Servo-level and path-tracking controllers, based on actuator modeling and system-level kinematics.
 - Navigation, motion planning, and intelligent behavior generation problems using high-level planning techniques based on searching.
- Concepts in the course were illustrated by design examples and their implementation using Matlab™ and Mobius™.
- Course content is generally applicable to all types of unmanned systems, but especially autonomous unmanned ground systems.