MRover Robotic Arm

Carolyn Busch, Anthony Liang, Karthik Urs
University of Michigan

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF MICHIGAN

MROVER
MICHIGAN MARS ROVER TEAM

Background: MRover + URC

Michigan Mars Rover (MRover) [1]

- A multidisciplinary, student-run robots team housed in the Wilson Student Team Project Center
- We build a new off-road robot every year for an international competition (University Rover Challenge)
- ~90 students across 10+ departments

University Rover Challenge [2]

- An international competition at the Mars Desert Research Station in the Southern Utah desert
- Simulation of an astronaut-assist robot on Mars
- Four parts: Autonomous Terrain Traversal, Extreme Retrieval and Delivery, Equipment Servicing, Science/Sample Analysis
- 36 Teams from N. America, Central America, Europe, Asia, Africa, and Australia

Problem and Solution

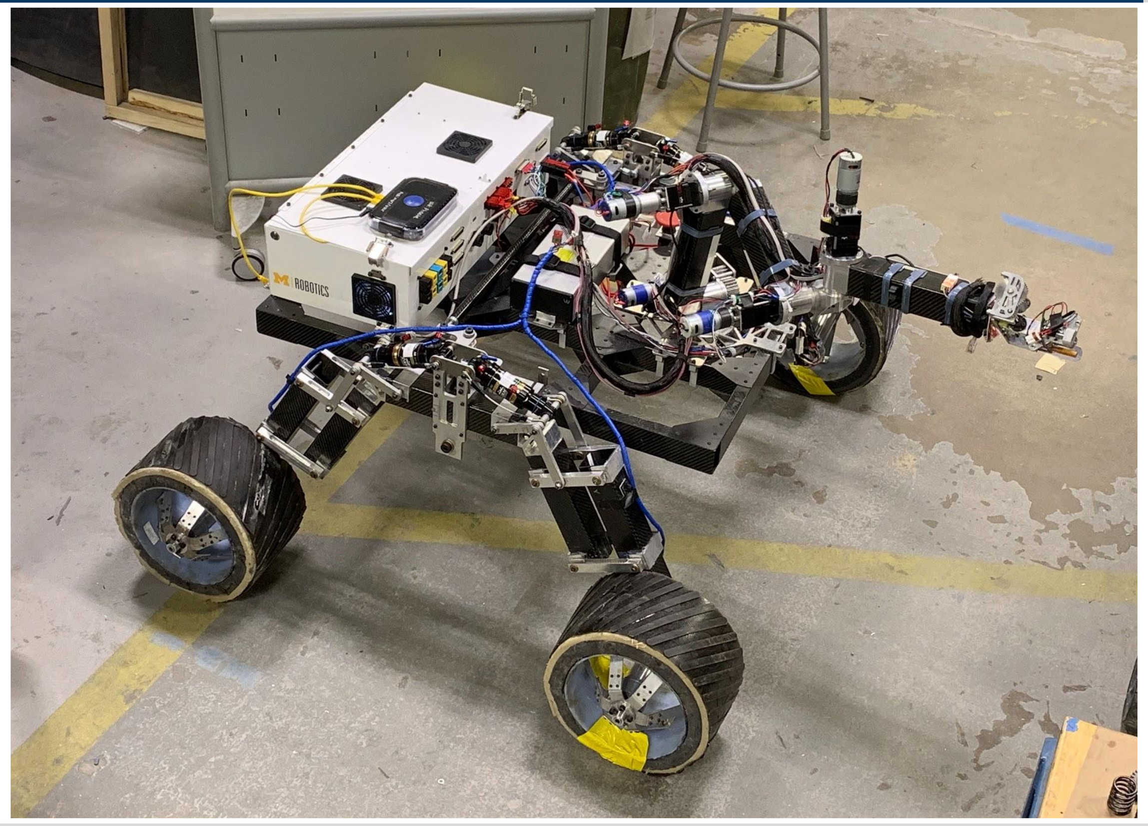
Problem:

- The competition requires the rover to perform high-dexterity operations in short times.
- The rover must interact with instrument panels and o lander to "service" it.
- Teleoperation is allowed, but is slow and requires high operator skill for the six DOF arm.
- How to go faster (earn more points) and make operation easier?

Solution:

- Inverse kinematic solver for the bottom five degrees of freedom.
- Motion planning with self- and world-collision avoidance.
- Perception and 3D mapping/reconstruction to find world obstacles.

Fully Assembled Rover w/ Arm



Methods and Architecture

Hardware

- MRover Arm: Six (revolute) degree of freedom
- nVidia Jetson TX2 (onboard)
- Stereolabs Zed camera (RGB-depth)
- Basestation computer

Forward Kinematics

- Simplified Denavit-Hartenberg [3] transformation method: Retain parallel frames at zero positions instead of link-local orientation

Inverse Kinematics

- Jacobian method [4] for lowest five degrees of freedom.
- Geometric approach to target: Each iteration will take a step some constant fraction of the length left to target.
- Retry with random seed configuration to leave local minimas and unsafe configurations.

Self- and World-Collision

- Modeled arm as a set of capsules and spheres for efficient but high-fidelity collision model.
- Model point cloud of world as a set of spheres.

Motion Planning

- RRT-connect [5] in configuration space (5-D) with self-collision, world-collision, and joint limit constraints.

Configuration space spline fit (cubic polynomial).

Motion Control

- Pure-pursuit [6] along spline plan -- velocity and acceleration constrained
- PID with direction-depended feed control scheme

User Interface/Simulator

- Browser-based GUI built upon KinEval [7]
- Kinematic simulation mode for development
- World-space target selection
- IK solver visualization
- Path preview
- Teleoperation mode -- joint command and end effector cartesian control

Perception -- In Progress

- Mounted Zed Camera running real time Yolo Detector [8]
- Perceive and identify objects and obstacles
- Communicates with KinEval to display pointcloud and object segmentation

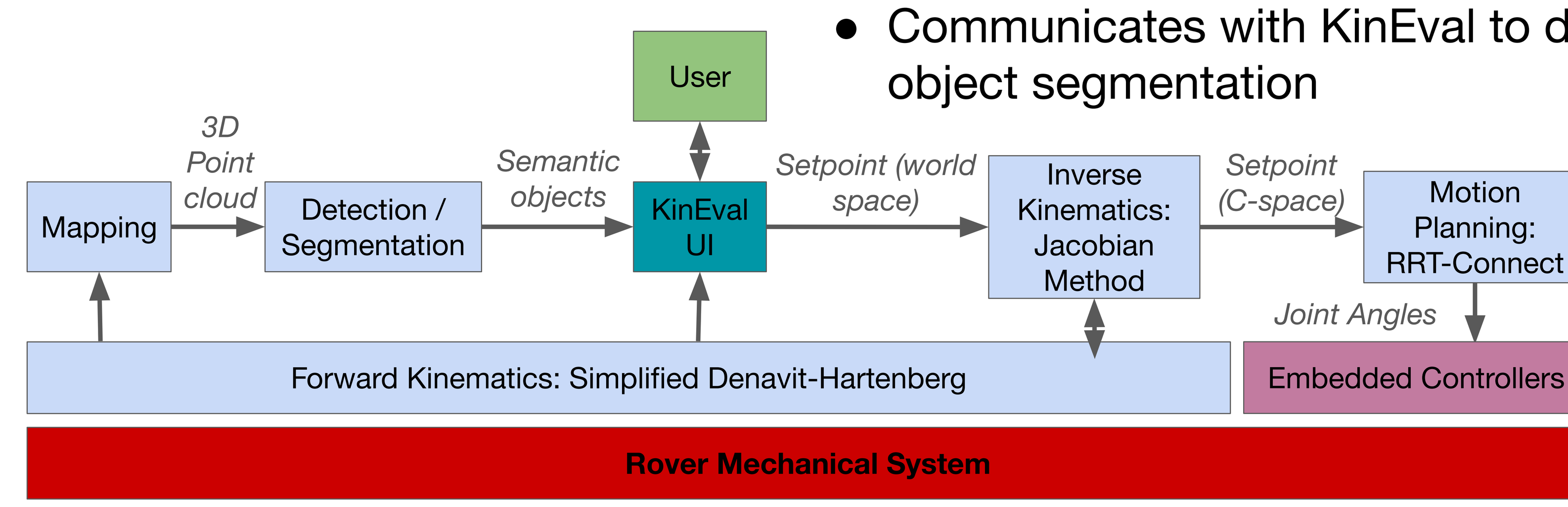


Figure 1: Simplified architecture diagram of implementation. Arrows represent data sent over Lightweight Communications Marshalling (LCM).

Functional Evaluation and Discussion

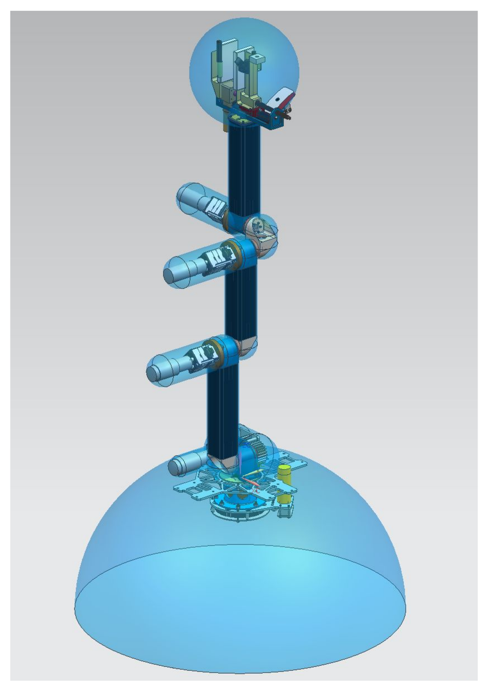


Figure 2: Collision model of arm (capsules and spheres) on overlaid on arm design.

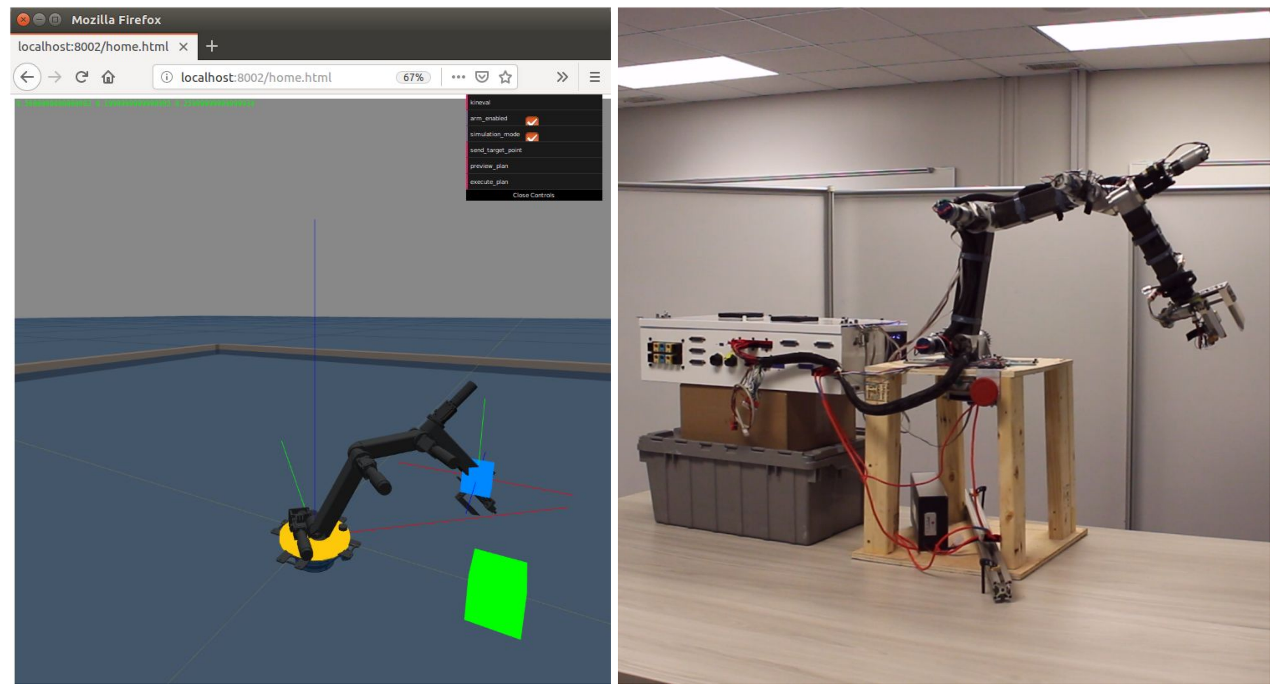


Figure 3: KinEval user interface tracking real-time position of arm (left); arm in a typical working configuration with temporary electrical enclosure setup (right).

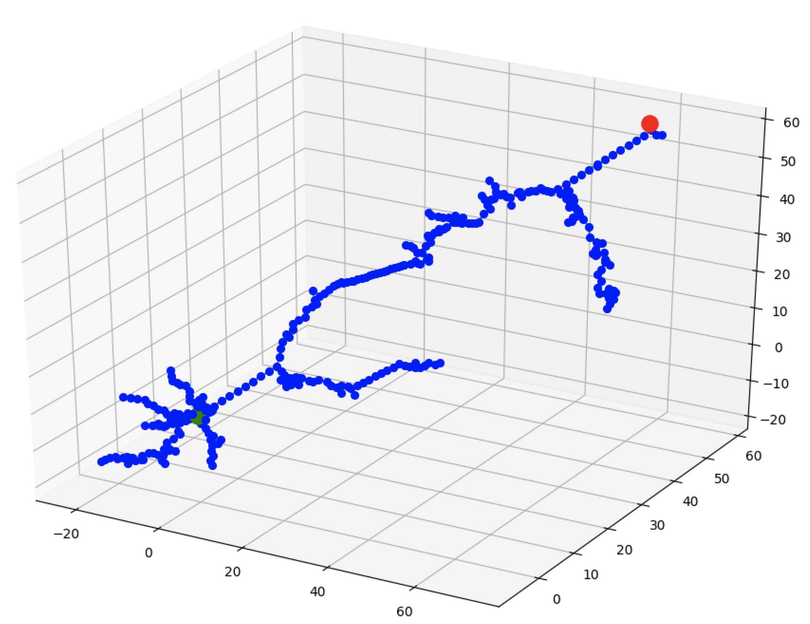


Figure 4: Plot of RRT-connect exploration in configuration space -- three dimensions (lowest three joints) shown.

Successes

- Easy to use UI
- Useful safety features like confirmations and previews
- Minimal delay for UI model update (Figure 3)
- High fidelity self collision model (Figure 2)
- Well-tuned motion control
- High accuracy in most configurations

Improvement Areas

- Slow world collision avoidance with point clouds
- Inefficient/suboptimal path planning around obstacles with regards to path smoothness and computation time
- Low accuracy in certain configurations (high extension)
- Poor mechanical backlash
- Poor cable management

Challenges

Integration

Software:

- Using LCM with JavaScript: Created a bridge server to translate messages into data sent over a socket

Hardware:

- Very fragile encoder wiring -- susceptible to dropping signal. New cable standard needs to be selected.

Inverse Kinematics

- Initially used Cyclic Coordinate Descent (CCD) -- too slow and with minimum angle enforcement, motivating switch to Jacobian.

Motion Control

- Embedded controllers (on motor drivers) had simple PID implementation -- not suitable for the arm in full working space
- Attempted writing a custom controller, but encoder refresh rate was too slow
- Found more control options for embedded controller that reached performance goals

Perception

- Installation of PCL library and integration with ZED cameras

Future Investigation

- UI features
- Data logging and playback
- Predictive feedforward controls for gravity compensation
- World-space path command
- Backlash compensation
- Rover chassis self-collision avoidance
- Process templates for common actions (turning a knob, etc.)
- Grasping pose detection

References

[1] Michigan Mars Rover: mrover.org
[2] University Rover Challenge: urc.marssociety.org/
[3] Forward Kinematics: The Denavit-Hartenberg Convention: https://users.cs.duke.edu/~brd/Teaching/Bio/asmb/current/Papers/chap3-forward-kinematics.pdf
[4] CMU 15-464: Technical Animation Lecture 6: http://www.cs.cmu.edu/~15464-s13/lectures/lecture6/IK.pdf
[5] Kuffner, J. J., & LaValle, S. M. (2000). RRT-Connect: An Efficient Approach to Single-Query Path Planning: https://www.cs.cmu.edu/afs/cs/academic/class/15494-s14/readings/kuffner_icra2000.pdf
[6] Pure Pursuit Controller: https://www.mathworks.com/help/robotics/ug/pure-pursuit-controller.html
[7] KinEval: https://github.com/autorob/kineval-stencil
[8] Yolo Detector: https://pjreddie.com/darknet/yolo/