Carnegie Mellon University

FPGA-AMP

Final Presentation

Matt Ngaw, Yufei Shi, Chris Stange
Team C5

Background

Motion planning

- Critical step in the robotics pipeline
- Guides motion of the robot
- Rapidly-exploring Random Tree (RRT) finds collision-free trajectories from a start position to a goal position
- Run A* on the set of collision-free trajectories in order to find optimal route

Problem

- For complex, latency sensitive robots, RRT is too slow on CPUs and too power-inefficient on GPUs

Solution

- Use FPGAs to accelerate motion planning while also consuming less power

https://en.m.wikipedia.org/wiki/File:Rapidly-exploring_Random_Tree_%28RRT%29_500x373.gif

Carnegie Mellon University

Quantitative Design Requirements

Relative to the reference implementation on 10th Generation Intel Core i7

95% accuracy

- Similar but not exact same motion plan (RRT is a non-deterministic algorithm)

10x speedup (latency)

- Time elapsed while servicing motion planning query
- Prior academic research achieved 1000x speedup
  - However, the comparison is questionable
- 10x is a modest speedup that justifies the addition of an FPGA

70% less power

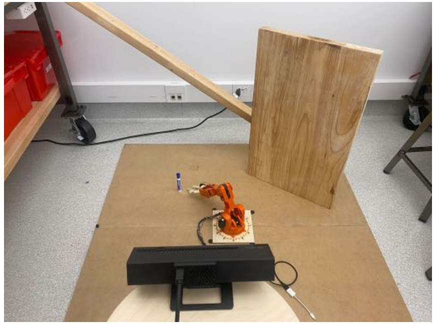
- 10th Generation Intel Core i7 consumes 105 W, FPGA use 30-40W

98% less energy

- Assuming the speedup and power requirements hold

Carnegie Mellon University

Solution Approach & System Specification



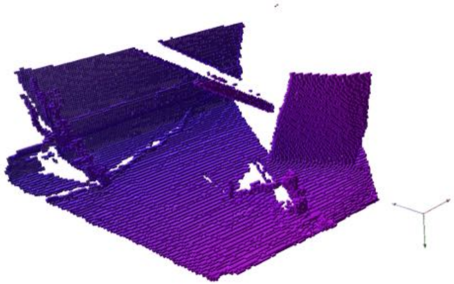
https://github.com/code-iai/iai_kinect2

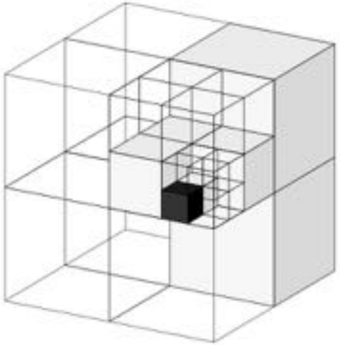
https://octomap.github.io/

http://www.arminhornung.de/Research/pub/hornung13auro.pdf

libfreenect2 +
iai_kinect2

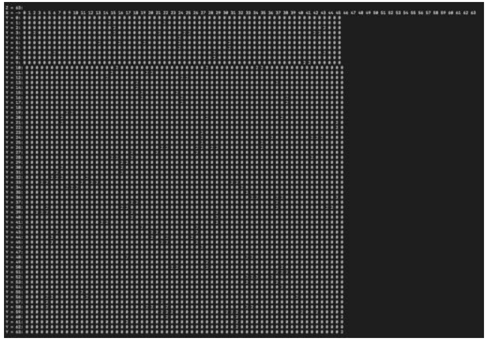
octomap





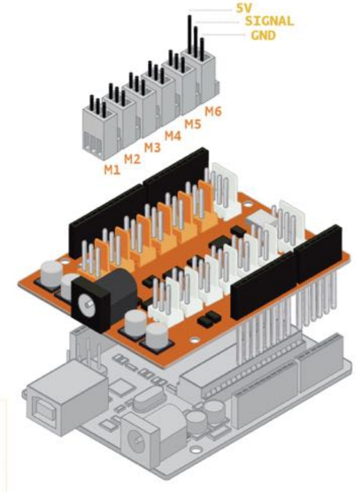
voxels

Dense Matrix



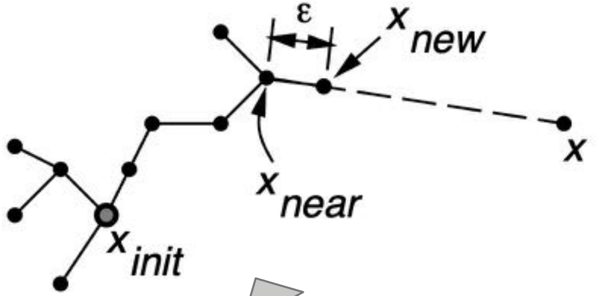
Carnegie Mellon University

Solution Approach & System Specification



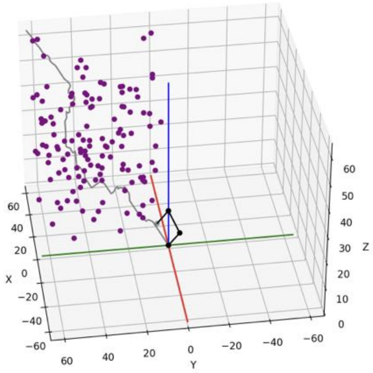
MOTOR “1“ BASE
MOTOR “2“ SHOULDER
MOTOR “3” ELBOW
MOTOR “4” VERTICAL WRIST
MOTOR ”5” ROTATORY WRIST
MOTOR “6” GRIPPER

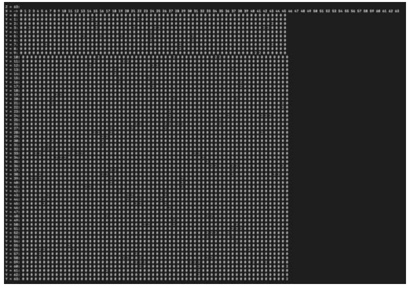
dynamics



RRT & A*

motion plan (kinematics)





http://www.arminhornung.de/Research/pub/hornung13auro.pdf

http://msl.cs.uiuc.edu/~lavalle/papers/LavKuf01.pdf

https://arxiv.org/pdf/1911.04676.pdf

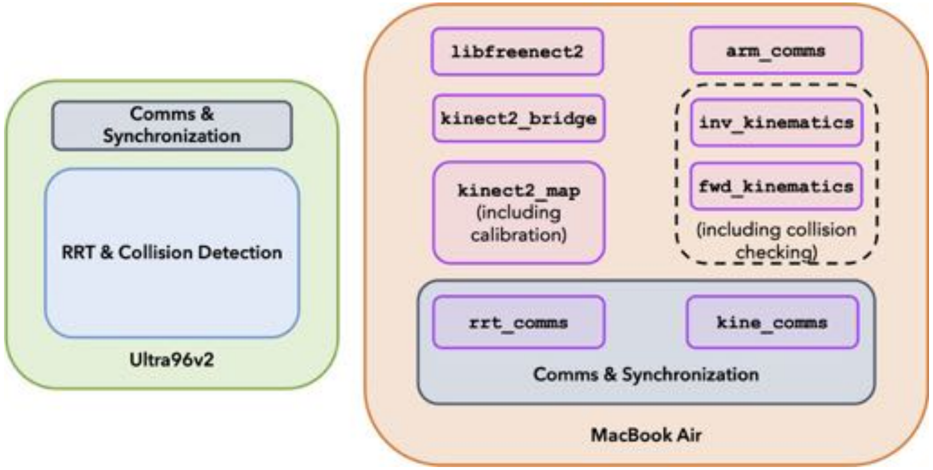
https://cdn.robotshop.com/media/A/Ard/RB-Ard-81/pdf/arduino-braccio-robotic-arm-quick-start-guide.pdf

Carnegie Mellon University

Block Diagram

System Implementation

- x86 MacBook Air running Ubuntu Linux is necessary due to its ability to run ROS
- Used to run A* and acts as a central control and synchronization hub
- Control and synchronization are implemented via ROS nodes
- Ultra96v2 is used as an offload accelerator and implements RRT
- Shell scripts on the laptop and communicate with the kernel's host program on the Ultra96v2 via a local network
- Communication with the robotic arm is done via UART



Carnegie Mellon University

Block Diagram

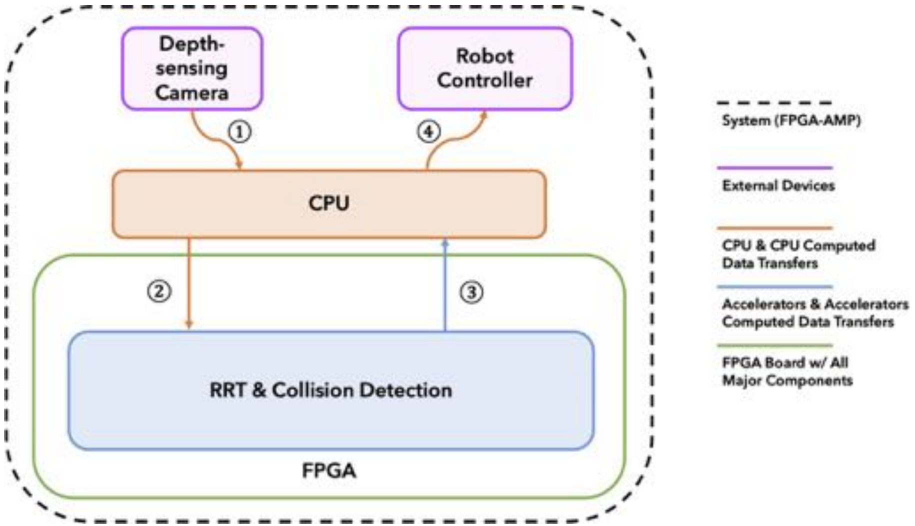
System Functionality

1) Depth-sensing camera feeds the raw 3D data it captures to the CPU

2) The CPU processes the raw 3D data, maps it into a grid of voxels, and serializes all voxels into a bit-stream, which is then stored in the block RAM on the FPGA board.

3) FPGA runs RRT and by searching collision data stored as a dense matrix in block RAM. Tree generated by RRT is transmitted back to CPU.

4) CPU runs A* and kinematics. Sends final commands to robotics controller.



Complete Solution

Demo:

Pick and place

- Field of obstacles
- Targets will be placed in hard to reach spots
- Simulate real world environments (i.e car factory)

What's left:

Perception

- Open source library using Xbox One Kinect depth sensor
- Final calibration based on the test environment

FPGA

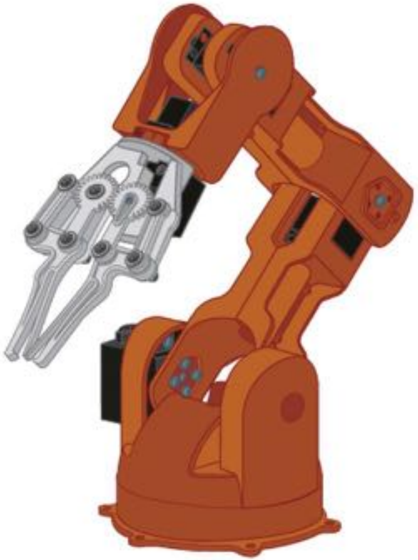
- Reimplementation of RRT in HLS to run on FPGA is complete
- Fine tuning in order to optimize power and performance

Robotic Arm Control

- Implemented custom forward kinematics solver, analytic inverse kinematics solver, and graphic simulation environment
- Calibration and synchronization with the perception system

Communication

- The Ultra96 creates its own local network, used for data transfer between laptop and Ultra96
- Implemented a synchronization protocol via a polling shell script
- Ensures Ultra96 always operates on most recent perception data
- Need to integrate laptop communication shell script into perception ROS node



https://cdn.robotshop.com/media/A/Ard/RB-Ard-81/pdf/arduino-braccio-robotic-arm-quick-start-guide.pdf

Carnegie Mellon University

Testing, Verification, and Metrics

Requirements	Testing	Metrics
Generate Collision Free Paths	Check that the path generated avoids collisions	Avoid collisions on >95% of scenes tested
Generate Optimal Paths	Ensure the delta between the optimal path and the reference solution is similar to the delta between the optimal path and FPGA-AMP	Similar deltas on >95% of scenes tested
Low Latency	FPGA-AMP generates paths significantly faster than the reference solution	10x speedup vs. reference solution
Power Efficient	FPGA-AMP generates paths while being significantly more power efficient than reference solution	70% decrease in power consumption vs. reference solution
Energy Efficient	FPGA-AMP generates paths while being significantly more energy efficient than reference solution	98% decrease in energy consumption per path vs. reference solution

Note: All requirements will be tested in both simulation and with real hardware.

Carnegie Mellon University

Testing, Verification, and Metrics

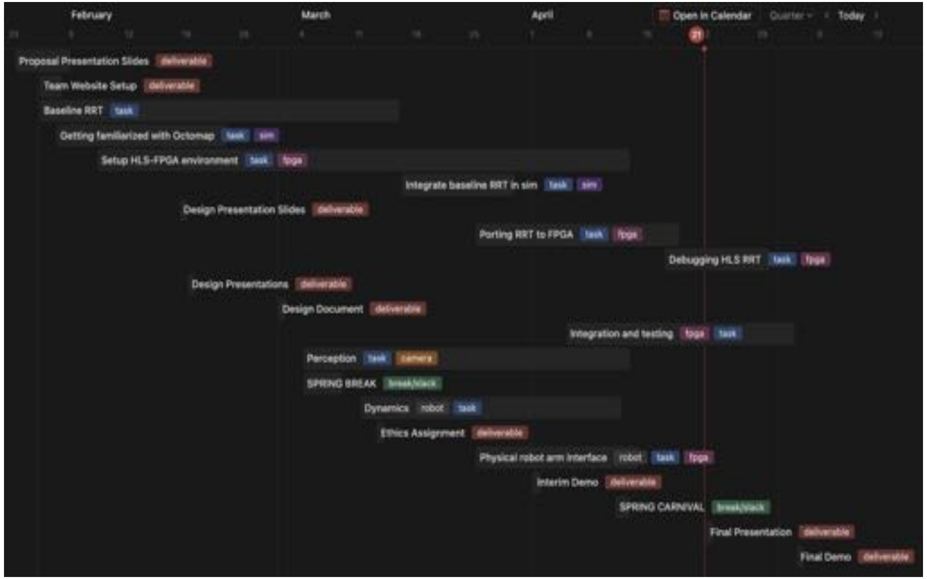
Perception	RRT	A*	Kinematics	Communication
>= 95% mapped voxels should have a resolution of 0.01m. Tested by matching the dimensions of the mapped objects with their real life dimensions.	RRT should generate a collision-free tree that connects a given pair of (start,end). Tested by visually examining the output tree and comparing the CPU and FPGA RRT outputs.	A* should generate a collision-free path that connects given pair of (start,end). Tested by plotting path in the simulation environment and achieving convergence >= 95% of runs	Angles and points generated by forward and inverse kinematics must match. Calibration with the perception system is tested by matching robots position with voxelized objects.	Laptop should be able to communicate with FPGA in an asynchronous and race-free fashion without losing any data. Tested by comparing the data transmission logs on CPU and FPGA.

Carnegie Mellon University

Schedule

Original Schedule

Adjusted Schedule



Carnegie Mellon University

Lessons Learned

- Start early and don't be afraid to pivot
- Better risk management and mitigation
- Computer Architecture project → Robotics project
    - Be flexible and willing to learn new things