Initialization

EECS 398-001
Intro. to Autonomous Robotics

ME/EECS 567
Robot Modeling and Control

Fall 2017

autorob.github.io
Agenda

• So, where is my robot?
• Roadmap for autonomous robotics
• Course administrative overview
• Assignment 1 (Path Planning) assigned today
  • JavaScript/HTML5 and git covered this and next lecture
• Action items: what I need from you now
Paralyzed by a stroke, Henry Evans uses a telepresence robot to take the stage — and show how new robotics, tweaked and personalized for people like him, can help in daily life. He shows
So, where is my robot?
Mobile Manipulation Robots
Willow Garage PR2

Fetch

Cost

$400K

$100K

Time

2009

2015

UM EECS 398/567 - autorob.github.io
Goal-directed autonomy

Pick-and-Place
(“Put that there”)

Teleoperation
(“Remote Control”)

Willow Garage PR2

Fetch

NASA Robonaut

Your robot

Cost

$1.5M

$400K

$100K

$40K

Time

2001

2009

2015

2020

Your robot

NASA Robonaut

Willow Garage PR2

Fetch

Your robot

2001

2009

2015

2020
NASA Robonaut 2 on the International Space Station
Wearable wireless motion capture for robot teleoperation

[Miller, Jenkins, et al., 2004, “Motion Capture from Inertial Sensing for Untethered Humanoid Teleoperation”]
**“Pick-and-Place” Autonomy**

NASA Robonaut

Willow Garage PR2

Fetch

Cost

- $1.5M
- $400K
- $100K
- $40K

Time

- 2001
- 2009
- 2015
- 2020

Your robot
Autonomous Robotics in 3 words

**Sense.**
Perceive a model of the current world state.

**Plan.**
Search over actions towards a goal state.

**Act.**
Execute actions through forces at robot’s motors.
Color + Depth Camera

Laser Rangefinder
ROB 550 BotLab / EECS 467 Escape Challenge
Michigan Next Generation Vehicle (Olson, Eustice, et al.)
Michigan Next Generation Vehicle (Olson et al.)
Color+Depth Camera

Laser Rangefinder
Object Manipulation

Willow Garage
Point Cloud Processing (briefly)

- For every point:
  - compute nearest neighbors
  - compute principal components in neighborhood $\text{eig}(\text{cov}(\text{nbhd}(:,1:3)))$
  - surface normal is eigenvector for smallest eigenvalue
  - Cluster points based on direction similarity of normal
Object Manipulation
Goal-directed Autonomy

“Pick-and-Place” Autonomy

NASA Robonaut

Willow Garage PR2

Fetch

Teleoperation

Cost

$1.5M

$400K

$100K

$40K

2001

2009

2015

2020

Time

Your robot

Cost

Time
Goal-directed Manipulation

Initial Scene → Goal Scene

“put small objects in bin”
“put small objects in bin”
Agenda

• So, where is my robot?

• Roadmap for autonomous robotics

• **Course administrative overview**

• Assignment 1 (Path Planning) assigned today
  
  • JavaScript/HTML5 and git covered this and next lecture
Course Staff

• Instructor: Chad Jenkins (ocj)
  • Office hours (Beyster 3644)
    • Monday 3-5pm, Tuesday 2-4pm
  • GSI: Zhen Zeng (zengzhen)
    • Office hours (Beyster 1637 A)
      • Wednesday 3-5pm, Thursday 3-5pm
Administrivia

• Meeting time/place
  • MW 1:30-3pm, Dow 2150

• Website
  • http://autorob.github.io/

• Discussion channel
  • https://autorob.slack.com/
#general

This is the very beginning of the #general channel, which you created today. The purpose of this channel is: *This channel is for team-wide communication and announcements. All team members are in this channel.* (edit).

+ Add an app or custom integration   & Invite people to autorob

Today

- ocj 10:14 AM
  - joined #general

- ocj 10:18 AM
  - Welcome to the AutoRob course, Introduction to Autonomous Robotics.
You are correct to be concerned about the relative weighting of dimensions for vectors in configuration space. Each dimension may need to be weighted based on the properties of that degree of freedom, such as its units, extents, scaling, etc. Scaling each dimension $s_i$ by a weight $w_i$ converts your computation of distance between two configurations $q$ and $q'$ to this form: $d(q,q') = \sqrt{\sum w_i (q_i - q'_i)^2}$
Course Structure

• Autonomous robot modeling and control

• Objective: Give you the computational skills to model and control any mobile manipulator

• Project-focused class

• 7 individual projects: from single joint control up to articulated motion planning

• Computing-friendly introduction to robotics: projects in JavaScript
Course Textbook

- Robot Modeling and Control (Spong, Hutchinson, Vidyasagar)
- Alternative: Robotics, Vision, and Control (Corke)
- In-depth coverage of concepts and math contained in textbooks
- Additional handouts and links will appear on the course website
Optional reading
Projects

• Projects implemented in JavaScript/HTML5 using KinEval stencil
  • Projects submitted and tracked through git (gitlab|github|bitbucket)
  • Instructor (ohseejay|ocj) needs read/commit access

• 7 projects
  • 6 Programming, 1 Written/Oral

• Grading: projects are broken down into features that are “checked”
  • points are earned through successful implementation of features
Projects

• **Path Planning** A-star search in 2D world

• **Pendularm** physical simulation and PID control of 1 DoF robot

• **Forward Kinematics** convert robot configuration to 3D space

• **Dance Contest** control of robot joints to do a dance

• **Inverse Kinematics** control gripper of a robot to reach a point in 3D

• **Motion Planning** collision-free planning over robot configurations

• **Best Use of Robotics** what will you do with all of this power?
Grading Summary

EECS 398-004: Introduction to Autonomous Robotics

- 7 projects (12 points each)
- 5 quizzes (4 points each)

A: 93+ points
B: 83+ points
C: 73+ points

EECS 598-009: Robot Kinematics and Dynamics

- 7 projects (18 points each)
- 5 quizzes (4 points each)
- Advanced features (4 points)

A: 135+ points
B: 120+ points
C: 105+ points
github.com/ohseejay

bitbucket.org/ohseejay
Collaboration Policy

• All work submitted must be your own
• No code can be communicated, including verbally
  • Explicit use of external sources must be clearly cited
  • Repositories must be private for proper compliance
• Free flow of discussion and ideas is encouraged
Late Policy

• Projects submitted after deadline may not be graded (zero credit)

• If a late submission is allowed, it can receive at most
  • 80% credit if pushed within 2 weeks of the deadline
  • 60% credit if pushed within 4 weeks of the deadline
  • 50% credit if pushed anytime before final grading
Regrading policy

- Projects features are graded with:
  - “CHECK” (sufficiently completed)
  - “DUE” (insufficiently completed)
  - “PENDING” (not due yet)

- A project feature can be regraded for partial credit for at most
  - 80% credit if pushed within 2 weeks of the last returned grading
  - 60% credit if pushed anytime before final grading
Projects for EECS398 Intro to Autonomous Robotics at the University of Michigan

- `71 commits`
- `7 branches`
- `0 releases`
- `3 contributors`

**sdnt Updated motion planning to disallow rotations about x-z axis**

- `js`
  - Fall 2016 release
- `kineval`
  - Disallow rotations about x-z axis. Fixed return val. Can now trace path
  - 8 months ago
- `project_pathplan`
  - 2d rrt-connect complete using drawHighlightedPath. Had to change give...
  - 8 months ago
- `project_pendularm`
  - pendulum seems complete
  - 11 months ago
- `robots`
  - Fall 2016 release
  - a year ago
- `tutorial_heapsort`
  - Added heap increase key function to heap.js to help with A* algorithm
  - 10 months ago
- `tutorial_js`
  - Playing with Javascript samples
  - a year ago
- `worlds`
  - Deleted all source_*.js files and added honor.txt
  - 10 months ago
- `.gitignore`
  - Implemented and tested matrix multiply and matrix transpose
  - 9 months ago
- `README.md`
  - Fall 2016 release
  - a year ago
- `grading.txt`
  - added grading for Final grading
  - 9 months ago
- `home.html`
  - Fall 2016 release
  - a year ago
53: student name
EECS 398-004 F16
Assignment 1 feature 1 (4.00/4): PathPlan_Heap: CHECK
commend: good work
Assignment 1 feature 2 (8.00/8): PathPlan_AStar: CHECK
commend: good work
Assignment 2 feature 3 (4.00/4): Pendulum_Euler: CHECK
error: Euler integrator is incorrect
commend: why is this integrator stable?
error: integration of dynamics is incorrect
commend: double check computation of pendulum acceleration
regrade: servo converges after several oscillations, borderline control performance
Assignment 2 feature 4 (4.00/4): Pendulum_VelocityVerlet: CHECK
error: integration of dynamics is incorrect
commend: double check computation of pendulum acceleration
regrade: working
Assignment 2 feature 5 (4.00/4): Pendulum_PID: CHECK
error: integration of dynamics is incorrect
commend: modulo correction is not needed
Assignment 3 feature 6 (2.00/2): FK_MatrixRoutines: CHECK
KinEval code stencil
KinEval code stencil

- Code stencil for AutoRob projects in 3D
- Uses threejs 3D rendering library and WebGL
- URDF-like robot description
- Usable, but not perfect, camera and UI controls
- AABB collision detection provided for planning
Let's start with base navigation

How to get from Location A to Location B?
Project 1: 2D Path Planning

• A-star algorithm for search in a given 2D world

• Implement in JavaScript/HTML5

• Heap data structure for priority queue

• 598: DFS, BFS, Greedy

• Submit through your git repository
### Dijkstra shortest path algorithm

all nodes ← \{dist_{start} ← \infty, parent_{start} ← \text{none}, visited_{start} ← \text{false}\}

start_node ← \{dist_{start} ← 0, parent_{start} ← \text{none}, visited_{start} ← \text{true}\}

visit_queue ← start_node

while visit_queue != empty & & current_node != goal

\begin{itemize}
  \item cur_node ← \text{min\_distance}(visit\_queue)
  \item \text{visited}_{\text{cur\_node}} ← \text{true}
  \item \text{for each}nbr \text{in not\_visited(adjacent(cur\_node))}
  \begin{itemize}
    \item enqueue(nbr to visit\_queue)
    \item \text{if} dist_{nbr} > dist_{\text{cur\_node}} + \text{distance}(nbr,cur\_node)
      \begin{itemize}
        \item parent_{nbr} ← current\_node
        \item dist_{nbr} ← dist_{\text{cur\_node}} + \text{distance}(nbr,cur\_node)
      \end{itemize}
  \end{itemize}
\end{itemize}

end if

end for loop

end while loop

output ← parent, distance
### A-star shortest path algorithm

all nodes ← \{\text{dist}_{\text{start}} \leftarrow \infty, \text{parent}_{\text{start}} \leftarrow \text{none}, \text{visited}_{\text{start}} \leftarrow \text{false}\}  \\
\text{start\_node} ← \{\text{dist}_{\text{start}} \leftarrow 0, \text{parent}_{\text{start}} \leftarrow \text{none}, \text{visited}_{\text{start}} \leftarrow \text{true}\}  \\
\text{visit\_queue} ← \text{start\_node}  \\

\text{while } (\text{visit\_queue} \neq \text{empty}) \&\& \text{current\_node} \neq \text{goal}  \\
\quad \text{dequeue: cur\_node} ← f\text{\_score(visit\_queue)}  \\
\quad \text{visited}_{\text{cur\_node}} ← \text{true}  \\
\quad \text{for each} \ \text{nbr} \ \text{in} \ \text{not\_visited(adjacent(cur\_node))}  \\
\quad \quad \text{enqueue:} \ \text{nbr} \ \text{to} \ \text{visit\_queue}  \\
\quad \quad \text{if} \ \text{dist}_{\text{nbr}} > \text{dist}_{\text{cur\_node}} + \text{distance}\text{(nbr,cur\_node)}  \\
\quad \quad \quad \text{parent}_{\text{nbr}} ← \text{current\_node}  \\
\quad \quad \quad \text{dist}_{\text{nbr}} ← \text{dist}_{\text{cur\_node}} + \text{distance}\text{(nbr,cur\_node)}  \\
\quad \quad \quad \text{f\_score} ← \text{distance}_{\text{nbr}} + \text{line\_distance}_{\text{nbr,goal}}  \\
\quad \quad \text{end if}  \\
\quad \text{end for loop}  \\
\text{end while loop}  \\
\text{output} ← \text{parent, distance}
Heaps

A heap is a tree-based data structure satisfying the heap property:
every element is greater (or less) than its children

max heap

min heap
Heap operations: Insert

1) add new element to end of tree

2) swap with parent

3) until heaped, do (2)
Heap operations: Extract

1) extract root element

2) put last element at root

3) swap with smaller child

4) until heaped, do (3)
git basics

- Create a git repository from gitlab, github, or bitbucket website
- Install git on your machine
  - OSX: https://code.google.com/p/git-osx-installer/
git basics

• create a local copy of a repository: `git clone <repo url>`
• add files to a repository: `git add <file listing>`
• commit changes to local repository: `git commit -a -m "<msg>"`
• push local changes to a remote repository: `git push`
• pull remote changes to a local repository: `git fetch` or `git pull`
• create a code branch in a repository: `git branch <branch name>`
• checkout a code branch from a repository: `git branch <branch name>`
• merge branches in a repository:
  - `git checkout <branch name>`
  - `git merge <other branch name>`
Highly recommended tutorial

http://learngitbranching.js.org/
What I need from you

• Accept invitation to the course discussion channel
  • https://autorob.slack.com

• Over the discussion channel, send me:
  • informal introduction confirming your name, email, and enrollment
  • pointer to your git repo for the course

• Get started on Assignment 1 (Path Planning)
  • Clone kineval-stencil-fall16 repository (will be released tonight)
  • Study examples in tutorial_js subdirectory
  • Complete tutorial_heapsort (noted with “STENCIL” in files)
  • Complete project_pathplan (concepts covered next week)