CSCE574 – Robotics
Spring 2014 – Review Sheet for Final Exam

Test date: Thursday, May 1, 12:30–3:00pm

This review sheet is intended as a guide to help you prepare for the test.

- ✓ The test will span the entire course, with a focus on material covered since the midterm.
- ✓ You will have 150 minutes to complete the exam, but it will be designed to be completed in 120 minutes.
- ✓ No notes nor gadgets are permitted.

You should expect:

- Multiple choice questions.
- Questions that ask you to write a sentence or two of English.
- Questions that ask you to draw simple diagrams.
- Questions that ask you to fill in missing parts of ROS C++ code.

You should not expect:

- True/false questions.
- Questions requiring entire paragraphs of English writing.

Here's an incomplete list of places I will look for questions as I write the test:

- "Vocabulary words" from the notes (usually in boldface).
- The figures in the notes. "This picture was meant to illustrate ...."
- The places in the notes where specific details were left for you to write in.
- The homework assignments.
- The projects and corresponding notes.

1. Introduction

1. What is a robot? Defining characteristics. Mont Fuji illustration.
2. Why is robotics hard?
3. Why can't we focus strictly on computing issues? Why do we need to understand both theoretical and practical issues?
4. Seven basic problem types. What does each problem type ask the robot to do?

2. ROS

1. What is ROS? Why is software like ROS necessary? Distributed computation. Code reuse. Seamless simulations.
2. Know the meanings and relationships between various ROS objects: Packages, nodes, the master, launch files (including the difference between rosrun and roslaunch), topics, publishers, subscribers, message types.
3. What ROS concepts are used for one-way many-to-many communication? Two-way one-to-one communication?
4. Understand the output from rqt_graph.
5. Be able to use the output of rosmsg show to write C++ code, including nested message types and vectors.
6. Understand the three example programs from Chapter 3 of the textbook: ROS_INFO_STREAM, ros::init, Publisher and Subscriber objects, subscription callbacks, ros::Rate, ros::spin, and ros::spinOnce. Topic names vs. message types. Specific details about drawing with turtlesim or tracking moving people are *not* important.
7. Basic launch file usage. Be able to convert between roslaunch node elements and the equivalent rosrun command lines.

3. Bug algorithms

1. Basic model. How does the robot move? What can it sense? Be able to tell if the Bug algorithms are applicable in a given situation.
2. Why doesn't the naive optimistic algorithm work?
3. Online vs. offline algorithms. Planning and execution interleaved vs. forming a complete plan before moving. Be able to apply these concepts to all of the navigation algorithms we've learned.
4. Hit points and Leave points
5. Bug1. Be able to execute the algorithm. Be able to state and explain an upper bound on path length.
6. Bug2. Be able to execute the algorithm. Be able to state and explain an upper bound on path length.
7. Comparison between Bug1 and Bug2. Be able to explain when Bug1 is better. Be able to explain when Bug2 is better.
8. TangentBug: Basic idea. Differences from Bug 1 and Bug 2. Be able to explain how the behavior changes as a function of the sensing range.

4. Potential fields

1. When are potential fields applicable? Intuitive explanation of how potential fields work.
2. Domain and range of the potential function. Robot follows the negative gradient of the potential function.
3. Definition of typical potential function as sum of attractive and repulsive forces. Details for attractive and repulsive components.
4. Notation: x, $U(x)$, ζ, η, Q_i^*, d_i
5. Local minima. What are they? Why do they cause problems for potential fields? Possible solutions.

5. Visibility graphs

1. Applicability: Under what conditions can the visibility graph method be applied?

2. Definition: What are the nodes? What are the edges? Be able to draw the visibility graph for a given set of obstacles.

3. How do we know that the shortest path must be along the visibility graph?

4. Difference between visibility graph and reduced visibility graph. Separating edges and supporting edges. Be able to identify edges that are in the visibility graph but not in the reduced visibility graph.

5. Testing colinearity. Testing clockwise vs. counterclockwise. Know that these are simple, constant time computations, followed comparing to zero. Understand what clockwise and counterclockwise mean. No need to memorize the formula. What happens if the points are reordered?

6. Testing for segment intersections. Use CW tests. Which four CW tests are needed? Why? Be able to write the formula and explain each part.

7. Testing for segment polygon intersections. Check whether the segment crosses the polygon boundary, then check whether one of its endpoints is within the polygon. Be able to explain how to check whether a point is inside or outside of a polygon.

8. Degeneracies: Definition. Two choices for how to deal with them. Be able to identify degeneracies. Example: Are three colinear points a degeneracy? Two colinear points? Four co-circular points? Three co-circular points?

9. Numerical robustness: Why is it a much bigger issue for geometric algorithms than other kinds of algorithms? What can go wrong?

6. Cell decompositions

1. Definition. What makes a good cell decomposition? In what sense should the cells be "simple"? Simplicial complex definition.

2. Trapezoidal decompositions. Definition. What kinds of cells do we get? Adjacency graph. Use for path planning. How to generate a path, given a sequence of cells. Sweep line algorithm: What are the invariants? What data structures does the algorithm maintain? What are the events? How are those data structures updated? Be able to draw the trapezoidal decomposition and its adjacency graph for a given set of obstacles.

3. Boustrophedon decomposition. Difference from trapezoidal decomp. Application to coverage problems. What kinds of cells do we get? Definition of monotone polygon. Be able to draw the boustrophedon decomposition and its adjacency graph for a given set of obstacles.

7. Pursuit and evasion

1. Problem definition. What can the pursuer see? What can the evaders do? What's the goal? Does the fact that the evaders can move arbitrarily quickly make the problem harder or easier?

2. Gaps: What is a gap? Clear and contaminated labels. Initial labels? Goal labels? How can gaps change? Appear, disappear, split, merge. How do the labels change for each of these events?

3. Algorithm: Cell decomposition: What kinds of cells do we get? What is a conservative region? Information graph. Where do we start in the information graph? What's the goal in this graph? How to reconstruct a path in the environment from a path in the graph. Recontaminations.

8. Geometric localization

1. What is localization? Types of localization problems. Active versus passive. Global versus local.
2. Geometric localization problem. It's active and global. What information does the robot have access to? No sensor noise. Visibility polygons. Be able to explain the importance of the robot's compass. What's the robot's goal?
3. Hypothesis generation. General idea, purpose. intuition of algorithm. How many hypotheses can there be? What does the set of possible states look like after the robot takes its first sensor reading?
4. Decision trees. What information is stored at internal nodes? Leaves? Edges? How to "execute" the strategy described by a decision tree.
5. Hypothesis elimination. How to compute visibility cell decomposition. Why is this decomposition useful? How and why to "overlay" multiple copies of the visibility cell decomposition. How to generate a decision tree (possibly many levels) from these overlays.

9. Motion planning

1. What is a configuration space? Why is it important?
2. Example C-spaces: Translating disk in the plane, translating and rotating polygon in the plane, robot arm with revolute joints. Be able to determine the configuration space for similar kinds of systems.
3. Common C-spaces: $SO(2)$, $SE(2)$, $SO(3)$, $SE(3)$
4. How many dimensions does a given C-space have?
5. Obstacle configurations. Relationships between configuration space, obstacle space, and free space.
6. Collision detection. Checks for points or for short segments in C-space.
7. Motion planning problems. What are the inputs? What are the outputs? Hardness of combinatorial approaches.
8. Basic idea of sampling-based algorithms: Avoid representing the obstacle boundaries explicitly.
9. Probabilistic roadmaps (PRM).
10. Rapidly-exploring random trees (RRT). Voronoi bias.
11. What does "completeness" mean? Is PRM complete? Is RRT complete? Probabilistic completeness: What does it mean? What algorithms have this property?
12. When is motion planning hard?
13. Comparison between PRM and RRT. Single query vs. multiple query.

10. The Kalman Filter

1. Notation for modeling motion (X, x, U, u, Θ, θ, f) sensing (Y, y, Ψ, ψ, h).
2. What is a Linear-Gaussian system? Linear transitions, linear sensing, Gaussian noise. Know the notation: A, B, C, G, H, ${}_\theta$, Σ_ψ. Why are LG systems special?
3. Kalman filter: What are its inputs? What are its outputs?
4. Kalman filter updates. No need to memorize the update equations, but do know how to use them.
5. What's the difference between the Kalman filter and the EKF?

11. Histogram and particle filters

1. Why do we need more algorithms beyond the KF and EKF?
2. In the figure titled "Basic Idea of Probabilistic Localization," why does the final distribution have 5 peaks? Why do these peaks have different shapes?
3. Transition models and observation models.
4. Histogram filter: How does it work? What tradeoffs occur from the choice of cell size?
5. Particle filter: Represent the distribution indirectly by a set of samples. Where do the weights come from? How does the resampling step work? Why is resampling important?

12. SLAM

1. Definition. What's the goal? Why is SLAM harder than either localization or mapping alone?
2. Mapping without worrying about localization. How is the map represented? Understand the role of each part of the update equation for occupancy grid probabilities. Observation models. Prior probabilities.
3. SLAM algorithm. Treat as a huge localization problem in a much bigger state space. Disadvantage to this viewpoint. What is Rao-Blackwellization? What kinds of particles does the algorithm maintain? How are the particles updated? How does this algorithm use the "mapping without worrying about localization" algorithm?

14. Final thoughts

1. Fear of technological unemployment
2. Centralized control: Multi-robot motion planning; joint C-space
3. Diagnostics
4. Security
5. (Lack of) robustness in centralized control
6. Need for usable human interfaces