Seven Big Ideas in Robotics, and How To Teach Them

David S. Touretzky
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
dst@cs.cmu.edu

ABSTRACT

Robotics is widely recognized as an interdisciplinary mixture of engineering and computer science, but the latter component is not well represented at many undergraduate institutions. The sophisticated technologies that underlie perception, planning, and control mechanisms in modern robots need to be made accessible to more computer science undergraduates.

Following the curriculum design principles of Wiggins and McTighe (*Understanding by Design*, 2^{nd} Ed.), I present seven big ideas in robotics that can fit together in a one semester undergraduate course. Each is introduced with an essential question, such as *"How do robots see the world?"* The answers expose students to deep concepts in computer science in a context where they can be immediately demonstrated. Hands-on labs using the Tekkotsu open source software framework and robots costing under $1,000 facilitate mastery of these important ideas. Courses based on parts of an early version of this curriculum are being offered at Carnegie Mellon and several other universities.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education, Curriculum; I.2.9 [**Robotics**]: Commercial robots and applications

General Terms

Algorithms, Design

Keywords

computer vision, path planning, kinematics, Tekkotsu

1. INTRODUCTION

Mobile robots provide a compelling context for introducing students to a wide range of computing topics, including machine perception, heuristic search, software engineering, and geometry and linear algebra [13]. But many computer science departments still aren't teaching serious robotics at the undergraduate level. Those that do offer a robotics elective are all too often teaching a high school curriculum where students assemble primitive robots from parts kits and program simple-minded reactive controllers. This paper is a call to action: we *can* make sophisticated robot technologies accessible to undergraduates. Doing so will help them develop into better computer scientists.

In this paper I present seven "big ideas" in robotics that can fit together in a one semester course for junior or senior level computer science majors. Following the *Understanding by Design* methodology of Wiggins and McTighe [17], each big idea is introduced with an essential question, such as *"How do robots see the world?"* The answers to these questions involve deep computer science concepts, some of which can only be fully mastered in graduate classes. But exposing undergraduates to these ideas – at an appropriate level – can broaden their understanding of computer science and encourage them to learn more. All of these ideas are implemented in the Tekkotsu software framework [11, 12] and can be demonstrated on commercially available, fully-assembled robots costing less than $1,000.

Courses based on parts of an early version of this curriculum are being offered at Carnegie Mellon and several other universities, including members of the ARTSI Alliance, a consortium of historically black colleges and universities and major research universities that is working to become a national resource for robotics education [2]. Course materials are available on the web at wiki.Tekkotsu.org.

2. THE TEKKOTSU "CREW"

The guiding principle advocated here is that students should be exposed to the richness of robotics algorithms and representations from the beginning, which precludes their implementing everything themselves from scratch. Instead they should be provided with a comprehensive software framework on which to build. The Tekkotsu framework includes a collection of interacting software modules known as the "Crew" [15] that students use to explore the big ideas to be described below. The Crew presently has four components:

- The **Lookout** manages the robot's sensor package and controls the "head" if one is present. Typically the head is a pan/tilt with a video camera.
- The **MapBuilder** [14] is responsible for vision, and for maintaining egocentric (body-centered) and allocentric (world-centered) maps of the environment.

SIGCSE'12, February 29–March 3, 2012, Raleigh, North Carolina, USA.

- The **Pilot** [16] handles localization and navigation.
- The **Grasper** [9] is responsible for manipulation, for those robots that include an arm.

Tekkotsu programmers issue requests to the Crew to achieve desired effects. The Crew modules are responsible for determining how to satisfy these requests, which may require invocation of complex algorithms such as path planners or inverse kinematics solvers. These algorithms are also directly accessible to students, and can be explicitly invoked when students want to explore them in detail. But under normal circumstances students work at a higher level of abstraction and leave these details to the Crew.

3. SEVEN QUESTIONS AND THE SEVEN BIG IDEAS

3.1 How Do Robots Know What To Do?

Big idea: Autonomous robot behaviors are **mechanisms** constructed from carefully designed algorithms and representations.

Underlying technology: event-based architectures and hierarchically structured, parallel state machines.

Learning goal: students will be able to think algorithmically about robot behavior.

Specific skills: students will be able to decompose a behavior into a series of states and transitions, draw state machine diagrams by hand, translate between graphical and textual state machine notations, make appropriate use of fork and join operations, visualize the execution of a state machine using Tekkotsu's Storyboard tool, and locate and describe errors in state machine definitions.

3.2 How Do Robots See The World?

Big idea: Robots use sophisticated but imperfect **computer vision algorithms** to deduce real world object representations from arrays of pixels.

Underlying technologies: basic vision algorithms (e.g., Hough transforms), AprilTags [10], SIFT (Scale-Invariant Feature Transform) [7], face detection, and more.

Learning goal: students will be familiar with both the capabilities and limitations of state of the art machine perception algorithms.

Specific skills: students will be able to use the MapBuilder to demonstrate the function of several built-in vision algoirithms, and use these components effectively in robotics applications.

3.3 How Do Robots Know Where They Are?

Big idea: Robots estimate their position and orientation in the world using a combination of **odometry**, **visual landmarks**, and other types of sensory information.

Underlying technology: particle filters.

Learning goal: students will understand the uses and limitations of odometry and visual landmarks, the basic principles of particle filters, and how particle filters are used for localization.

Specific skills: students will be able to demonstrate effective robot navigation behavior by arranging landmarks appropriately in the environment and invoking the Pilot's localization mechanism as needed to determine their robot's position.

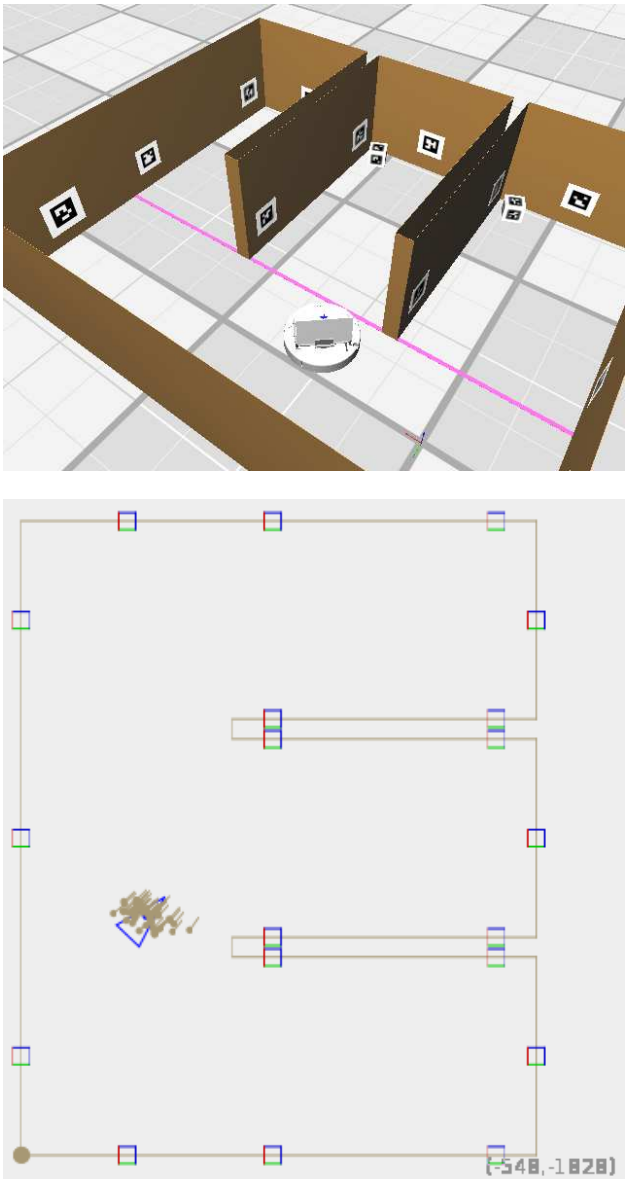

Figure 1: Top: Robot wandering in an E-shaped maze with AprilTag landmarks. Bottom: robot's estimated position and heading shown by a blue triangle; particle cloud shows each particle's position and heading value.

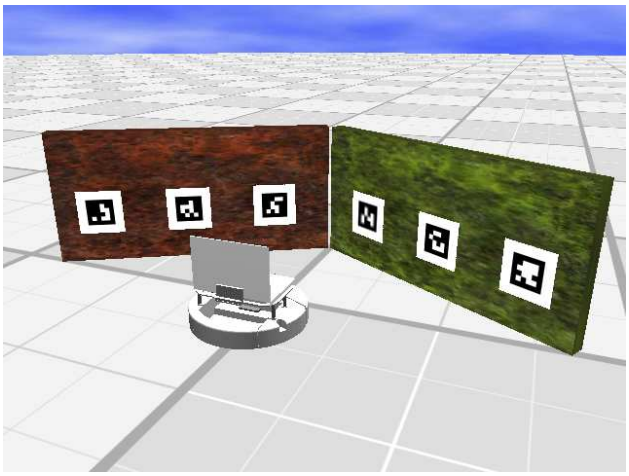

Figure 2: Starting position of the robot relative to a vee-shaped barrier in the Mirage simulator. The real-world version of the barrier uses two sheets of posterboard at a 120° angle.

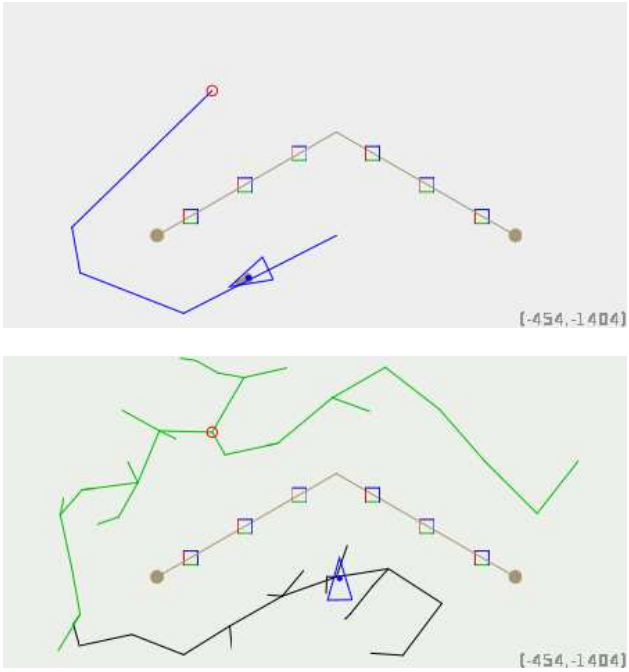

Figure 3: Top panel: world map with barrier and AprilTags; the path planned by the Pilot is shown in blue, and the goal location marked with a red circle. The robot is following the path. Bottom panel: how the path planner works. Once search trees from the start location (black) and goal location (green) meet in the RRT-Connect algorithm, the path can be extracted and then smoothed, yielding the blue line in the top panel.

3.4 How Do Robots Know Where To Go?

Big idea: Robots navigate through the world using a **path planner** to search for routes around obstacles and an **execution monitor** to ensure that the robot stays on the path.

Underlying technology: RRTs (Rapidly-exploring Random Trees) [6].

Learning goal: students will understand the concept of a stochastic search algorithm, how the RRT-Connect algorithm is used for path planning, and how planning can fail.

Specific skills: students will be able to invoke one of the Pilot's path planners to plan and execute paths through environments containing obstacles.

3.5 How Do Robots Control Their Bodies?

Big idea: Robots describe their bodies as **kinematic trees** and use **kinematics solvers** to translate between joint angles and body coordinates.

Underlying technology: kinematic description files, forward and inverse kinematics solvers.

Learning goal: students will understand kinematic descriptions expressed as trees of joints and links, and how kinematics solvers use this representation when translating between world coordinates and joint coordinates.

Specific skills: students will be able to construct and visualize kinematic descriptions using Tekkotsu's DH Wizard tool, and invoke kinematics solvers via the Lookout or Grasper to produce desired robot motions.

3.6 What Can We Do When A Robot Becomes Too Complex For One Person To Fully Understand It?

Big idea: Robots are complex software systems that employ standard **abstraction** and **software engineering** techniques to **manage complexity**.

Underlying technologies: tools such as modular design, coding standards, doxygen (automatically generated documentation), standard libraries, etc.

Specific skills: students will be able to search and navigate through the extensive online Tekkotsu documentation (over 3500 web pages), make appropriate use of abstraction mechanisms (e.g., write classes that inherit from predefined node types, or create appropriate enumerated types for state machine signaling), produce code that conforms to Tekkotsu coding conventions, and give a picture of the modular organization of Tekkotsu in terms of directories and namespaces.

3.7 How Do We Calculate the Quantities Needed To Make A Robot Function?

Big idea: Geometry, trigonometry, and linear algebra are the **mathematical underpinnings** of much of robotics.

Underlying technology: Tekkotsu's angular arithmetic classes and **fmat** matrix/vector manipulation package.

Learning goal: students will understand frames of reference, angular and vector arithmetic, homogeneous coordinates, coordinate transformation matrices, and how these concepts are applied in robotics.

Specific skills: students will be able to solve simple kinematics problems using Tekkotsu's built-in kinematics solvers and matrix/vector classes.

4. HOW TO TEACH THESE IDEAS

The first stage of the *Understanding by Design* process is to define the learning goals, big ideas, and essential questions,

as listed above. The second stage, which will not be covered here, is to plan for assessment by deciding what constitutes acceptable evidence that students have met the learning goals. The third stage is to choose specific instructional activities that will lead students to achieve these learning goals. For most of the big ideas, a seven-step framework similar to the WHERETO schema of Wiggins and McTighe [17] or the framework of Bransford et al. [3] can be used to guide instruction. Here is how this framework would be applied to the third big idea in our list, localization:

- **Motivate:** Introduce an essential question and big idea: How do robots know where they are?
- **Demonstrate:** Show the technology students will learn about, e.g., set out some landmarks and demonstrate the robot using them to keep track of its position as it moves. How did it do that?
- **Explain:** Provide a high level explanation of how the technology works, e.g., describe particle filters and how they're used by the Pilot for localization.
- **Visualize:** Give students a way to see the algorithm in action, e.g., show them how to display the particles the Pilot is using, and observe how the particles change as the robot moves (Figure 1).
- **Experiment:** Guide students to play with the technology to test its accuracy and uncover its limitations. For example, how does the particle filter respond if you move one of the landmarks? How does it cope with ambiguous (visually identical) landmarks?
- **Apply:** Show students how to apply the technology in programs they write themselves. Example: show them how to ask the Pilot to use specific landmarks for localization, and how to obtain the robot's position estimate from the particle filter.
- **Review:** Ask students to summarize what they've learned about particle filters and localization, and to demonstrate mastery by answering quiz questions and solving small problems.

Algorithm visualization (step 4) is the most technically demanding of the seven steps, since it requires software enhancements, but most of this work has already been done for Tekkotsu. Figure 1 shows the particle filter's state made visible using a tool called the SketchGUI [14]. The robot is visualized using the Mirage simulator. AprilTags [10] on the walls of the maze serve as landmarks.

Tekkotsu includes an extensive collection of visualization tools, such as the Storyboard, which generates and displays a state machine execution trace, and the DH Wizard, which displays the kinematic structure of a robot. To take one more example: when the Pilot is navigating from the robot's starting position (Figure 2) to a goal location on the other side of a vee-shaped barrier, the planned path is automatically displayed as a blue line (Figure 3, top); also note the tight particle cloud. The bottom panel shows the search tree constructed by the path planner using the RRT-Connect algorithm [6]. Figure 2 was generated by the Mirage simulator, and Figure 3 by the SketchGUI tool. Students can pose their own path planning problems and see the results visualized automatically using this tool.

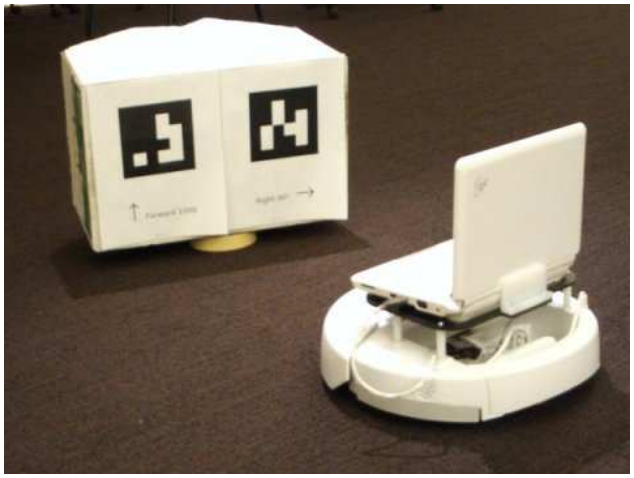

Figure 4: A challenge problem using pairs of AprilTags to guide the robot.

Guided experimentation (step 5) is also difficult to do well, and we expect to explore many variations on this activity over the next few years. For example, we recently developed an activity called Particle Filter Bingo in which students use Tekkotsu to participate in a competitive simulation of robot localization. It's fun, but we don't know yet whether this activity achieves its intended effect of reinforcing student understanding of how particles are evaluated by a particle filter.

5. WHAT CAN STUDENTS DO?

Figure 4 shows part of a challenge task from the 2011 ARTSI Student Robotics Competition. Students had to program their robot to read and follow instructions to make its way around a complex course they had not seen before. The instructions consisted of pairs of AprilTags describing a distance to travel and a direction to turn to reach the next instruction pair in the sequence. The task combines vision, navigation, and state machine control. Some student teams who completed this task had taken or were presently enrolled in a Tekkotsu-based robotics elective, but other teams learned in less formal settings, using the material on the wiki.

A high school student who learned Tekkotsu programming in a summer enrichment program was able to program a Chiara hexapod robot to approach an electronic piano keyboard, visually locate the keys, and play Ode to Joy with its right front leg (Figure 5). This work combined computer vision algorithms with kinematics calculations. The student's video explaining how his software worked received an award at the 2011 Association for the Advancement of Artificial Intelligence conference [5].

Other student projects have included chasing and swatting a ball in a maze, and playing tic-tac-toe using the Grasper to calculate arm trajectories for moving the pieces. A masters student programmed a robot to play chess on a real chessboard [4].

6. BARRIERS TO IMPLEMENTATION

The Tekkotsu software and curriculum materials are freely available on the web, so the cost of adoption is limited to the

Figure 5: Chiara hexapod robot programmed by a high school student to play Ode to Joy.

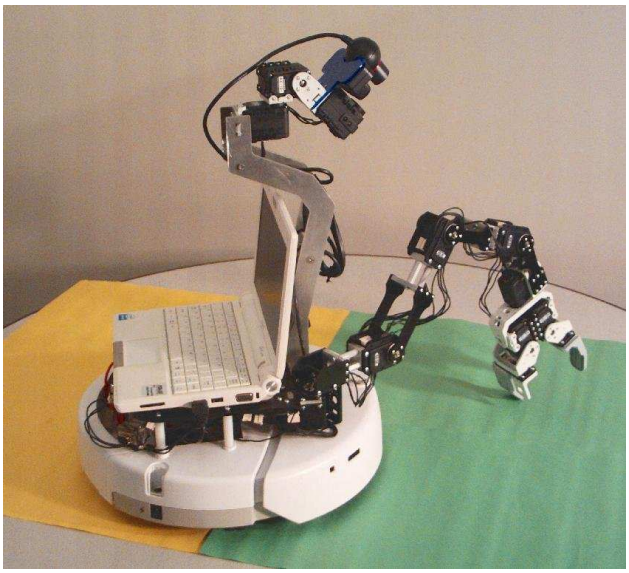

Figure 6: Calliope prototype with pan/tilt camera and 5 degree-of-freedom arm. A version with a Microsoft Kinect in place of the webcam is under development.

hardware cost of the robots. One of the biggest barriers to improving undergraduate robotics courses since the demise of the Sony AIBO has been the lack of capable but affordable robots. The iRobot Create mobile base provides excellent price/performance but is not a complete solution due to its lack of sensors and on-board computing power. Fortunately, several good options are now available that place an Ubuntu Linux netbook atop a Create. The Calliope from RoPro Design costs $900 for the basic model and is available from RobotShop.com with Tekkotsu pre-installed. Versions with a pan/tilt and an arm (Figure 6) are planned for 2012. The Turtlebot from Willow Garage, and the Bilibot, which originated from a group at MIT, both sell for roughly $1200 and include a Microsoft Kinect as the camera. Both come with ROS, the Robot Operating System from Willow Garage [18]. The Bilibot also includes a 2 degree of freedom arm. All three designs are open source.

Although these platforms are admittedly more expensive than LEGO or VEX kits, fewer are needed, because they are interchangeable and can thus easily be shared. Four robots are recommended for a class of 8 to 12 students.

We've recently begun supplying students with bootable 8GB flash drives that have Ubuntu, Tekkotsu, and the Mirage simulator pre-installed. This allows students to use their personal laptops for robotics assignments without altering their hard drive in any way.

While some work can be done in simulation, students will still need ample laboratory time to interact with physical robots. And the robots need space to run around in. This is perhaps the hardest barrier to overcome, as space is at a premium at many schools. The ideal arrangement is a dedicated lab with keycard access so students can have unrestricted access to the robots. Some schools instead opt to have designated periods each day for the laboratory to be open with a faculty member or graduate student present.

A final issue is lack of a textbook specific to this curriculum. At present, instructors are using the labs and tutorials on the Tekkotsu wiki as their primary source, with supplementary material of their own choosing. A popular supplementary textbook has been Mataric's *Robotics Primer* [8].

The curriculum design presented here could potentially be used with other software frameworks that provide particle filters, path planners, and visualization tools, such as ROS. Tekkotsu's design philosophy differs from ROS [12], e.g., Tekkotsu provides tightly-integrated components and uses a shared memory model for simplicity, while ROS emphasizes modularity (at the cost of increased complexity) and uses a networking model for scalability. Both seek to provide services to robot programmers based on state of the art algorithms.

7. DISCUSSION

Robotics courses using the Tekkotsu framework have been taught for six years at Carnegie Mellon, and for several years at ARTSI Alliance-affiliated schools including Norfolk State University, Florida A&M University, Hampton University, Winston-Salem State University, Jackson State University, and the University of the District of Columbia. Tekkotsu and the associated curriculum materials have both evolved substantially over this time. The curriculum described here, which is the new target for both Carnegie Mellon and the ARTSI Alliance, is partially implemented now and will be completed over the next few years.

7.1 What Do Students Gain?

Besides learning about robotics, students gain several other important things in a course of this type:

Appreciation for mathematics. We require CS majors to take a lot of math, but almost none of it is used in their CS courses. For many, especially those who have not taken computer graphics, robotics may be their only opportunity to apply trigonometry and linear algebra to real computing problems. For some students, robotics will actually be their introduction to linear algebra.

Mastery of advanced programming concepts. Many students report that learning Tekkotsu has made them better C++ programmers by helping them understand features such as templates, multiple inheritance, operator overloading, functors, and namespaces. These topics are often introduced superficially in undergraduate courses, using only toy examples. Since they are used extensively in Tekkotsu, stu-

dents can see how they are actually employed in "industrial strength" software.

Software engineering skills. Robotics provides a compelling context for teaching software engineering because students are being asked to delve into and extend a software system far more complex than anything they could expect to write from scratch themselves. Baltes and Anderson, describing a mixed reality infrastructure for robotics instruction, have likewise observed that working on robotics assignments gave students useful software engineering experience [1].

7.2 Common Misconceptions

Certain misconceptions about robotics must be corrected for CS education to move forward [13]:

Robotics is not about building robots. Not for computer scientists, anyway. We don't expect our students to build their own laptops. Why should they be trying to build robots, when the robots they ought to be using are more complex than any laptop? Leave the construction to industry. CS students' time is best spent learning to program the most capable robots we can provide them with.

Robotics is not just a vehicle for teaching CS1, and robot programming involves more than writing simple reactive controllers. As computer science faculty become more familiar with the rich intellectual content of robotics and the tools for teaching this material to undergraduates, they will be less likely to trivialize the subject.

7.3 More Big Ideas

Some additional big ideas that could be taught to undergraduates include:

- **Human-robot interaction:** *How should robots behave around people?*
- **Multi-robot coordination:** *How can robots work together?*
- **Task-level planning:** *How can robots formulate plans for solving complex problems?*

These topics would be difficult to fit into the one semester course outlined in this paper, but would be good choices for the second half of a two-semester robotics sequence.

8. ACKNOWLEDGMENTS

This research was supported in part by National Science Fondation awards DUE-0717705 and CNS-1042322. I am grateful to Sharon Carver at Carnegie Mellon for introducing me to the *Understanding by Design* methodology and for several very helpful discussions. I would also like to thank the faculty and students of the ARTSI Alliance for their insights and feedback regarding the Tekkotsu framework and curriculum.

9. REFERENCES

[1] J. Baltes and J. E. Anderson. Leveraging mixed reality infrastruture for robotics and applied AI instruction. In *Proceedings of EAAI-10: The First Symposium on Educational Advances in Artificial Intelligence*, Menlo Park, CA, 2010. AAAI Press.

[2] C. Boonthum-Denecke, D. S. Touretzky, E. J. Jones, T. Humphries, and R. Caldwell. The ARTSI Alliance: Using robotics and AI to recruit African-Americans to computer science research. In *Proceedings of FLAIRS-24*. AAAI Press, 2011.

[3] J. D. Bransford, A. L. Brown, and R. R. Cocking. *How people learn: Brain, Mind, Experience, and School*. National Academy Press, Washington, DC, 2000.

[4] J. Coens. Taking Tekkotsu out of the plane. Master's thesis, Carnegie Mellon University, Computer Science Department, 2010. Available at http://reports-archive.adm.cs.cmu.edu/anon/2010/CMU-CS-10-139.pdf.

[5] A. Iyengar. Chiara robot plays the piano, 2011. Video available at aivideo.org or `http://www.youtube.com/watch?v=-e8zmGypBDg`.

[6] J. J. Kuffner and S. M. LaValle. RRT-connect: an efficient approach to single-query path planning. In *ICRA'2000*, 2000.

[7] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV 2*, pages 1150–1157, 1999.

[8] M. J. Mataric. *The Robotics Primer (Intelligent Robotics and Autonomous Agents)*. The MIT Press, Cambridge, MA, 2007.

[9] G. V. Nickens, E. J. Tira-Thompson, T. Humphries, and D. S. Touretzky. An inexpensive hand-eye system for undergraduate robotics instruction. In *SIGCSE 2009*, pages 423–427, 2009.

[10] E. Olson. AprilTag: A robust and flexible multi-purpose fiducial system. Technical report, University of Michigan April Laboratory, 2010. Available at http://april.eecs.umich.edu/papers/details.php?name=olson2010tags.

[11] Tekkotsu robotics development framework, 2011. http://Tekkotsu.org.

[12] E. J. Tira-Thompson and D. S. Touretzky. The Tekkots robotics development environment. In *Proceedings of ICRA-2011*, Shanghai, China, 2011.

[13] D. S. Touretzky. Preparing computer science students for the robotics revolution. *Communications of the ACM*, 53(8):27–29, August 2010.

[14] D. S. Touretzky, N. S. Halelamien, E. J. Tira-Thompson, J. J. Wales, and K. Usui. Dual-coding representations for robot vision in Tekkotsu. *Autonomous Robots*, 22(4):425–435, 2007.

[15] D. S. Touretzky and E. J. Tira-Thompson. The Tekkotsu "crew": Teaching robot programming at a higher level. In *Proceedings of EAAI-10: The First Symposium on Educational Advances in Artificial Intelligence*, Menlo Park, CA, 2010. AAAI Press.

[16] O. Watson and D. S. Touretzky. Navigating with the Tekkotsu Pilot. In *Proceedings of FLAIRS-24*. AAAI Press, 2011.

[17] G. Wiggins and J. McTighe. *Understanding by Design*. Pearson Education, Upper Saddle River, NJ, expanded 2nd edition, 2005.

[18] Willow Garage. ROS: Robot Operating System, 2011. http://ros.org.