Extending Tekkotsu to New Platforms for Cognitive Robotics

Ethan J. Tira-Thompson
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Glenn V. Nickens *
Dept. of Computer & Info. Science
University of the District of Columbia
Washington, DC 20008

David S. Touretzky
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Tekkotsu is an open source application development framework for mobile robots that promotes a high level approach or robot programming which we call "cognitive robotics". Originally developed for the Sony AIBO, Tekkotsu now supports a variety of platforms under the Linux and Mac OS X operating systems. We present the first version of a new educational robotics platform, Regis, designed specifically for teaching cognitive robotics.

Introduction

The use of inexpensive platforms has profoundly limited introductory robotics courses. In many instances, students spend considerable time on robot construction. This is required in Lego Mindstorms-based courses, but also occurs in other courses where students assemble small wheeled carts with single-board microprocessor controllers. While robot building is a good way to learn mechanical engineering concepts, it consumes valuable time that would be better spent on topics connected to robotic *intelligence*. Furthermore, the types of robots beginning students can build are crude compared to the platforms they would want to use for intelligent perception and manipulation. We advocate a different strategy for introducing computer science students to robotics.

Tekkotsu[1] is an open source application development framework for mobile robots that promotes a high level approach to robot programming which we call "cognitive robotics". Cognitive robotics draws inspiration from – and makes explicit reference to – ideas in cognitive science.[2] It encourages students to focus on problems in perception and manipulation rather than merely reacting to raw sensor values by turning motors on and off.

Tekkotsu was originally developed for the Sony AIBO robot dog, which at the time was the only reasonably-priced educational platform powerful enough to support this approach. Since the AIBO is no longer offered for sale, and comparably-priced alternatives are not yet available, we have reengineered Tekkotsu to support a variety of other platforms, including the Qwerkbot+ (Nourbakhsh et al., 2007a; Nourbakhsh et al., 2007b) and Lynx Motion Servo Erector Set arm (Lynx Motion, 2007). We have also begun exploring our own designs that can be assembled from off-the-shelf components and better meet the needs of the cognitive robotics curriculum. Our first such design, Regis, is presented here.

An undergraduate course in Cognitive Robotics using Tekkotsu on the AIBO has been taught twice at Carnegie Mellon, and will be taught for a third time in January 2008. The lecture notes, labs, and homework assignments are freely available via the Tekkotsu.org web site, as is the Tekkotsu software itself. Courses and directed research projects utilizing some of this material have also been offered at Spelman College, the University of the District of Columbia, Hampton University, and Florida A&M University under a grant from the National Science Foundation's Broadening Participation in Computing Program (Williams et al., 2007).

Cognitive Robotics

The central thesis of cognitive robotics is that ideas in cognitive science about perception and manipulation can inspire the design of robot primitives and give us a language for talking about them. Some examples include visual routines (Ullman, 1984), dual-coding theory (Paivio, 1986), affordances (Gibson, 1979), and motor schemas (Schmidt, 1975). Because perception and manipulation remain unsolved problems, we cannot expect these theoretical proposals to be directly translatable into running code. The relationship is more subtle, and engineering is also a prominent consideration because our solutions must work on real robots. So the cognitive robotics philosophy is really a two-part argument:

1. Beginning robot programmers are better served by providing them with high-level primitives for perception and manipulation, allowing them to implement more interesting behaviors than would be possible with lower-level primitives.

*Current address: Computer Science Department, Norfolk State University, Norfolk, VA 23504.

[1] The name Tekkotsu means "skeleton" or "framework" (literally "iron bones") in Japanese.

[2] The term "cognitive robotics" is used in a variety of ways by other research groups.

2. Robotics students should have some appreciation for how cognitive science theories of perception and manipulation can inform the design of these primitives.

To give a specific example: Paivio's dual-coding theory of mental representations posits complementary imagistic and verbal representations with extensive referential connections between them. Tekkotsu's vision system follows this approach, offering iconic (pixel-based) and symbolic (geometry-based) representations of the robot's visual world (Touretzky et al., 2007). The iconic representations can be manipulated by a set of composable operators inspired by Ullman's notion of visual routines. They include operations such as connected components labeling, seed fill, neighbor sum, and convex hull. Extraction and rendering operators convert between iconic and symbolic representations. In one exercise in the Cognitive Robotics course, students use a mixture of these primitives, including line and ellipse extraction, line rendering, and region intersection, to parse an image of a tic-tac-toe board as seen through the robot's camera.

Tekkotsu Features

Most of Tekkotsu's features have been described elsewhere (Tira-Thompson, 2004; Touretzky et al., 2007), so we offer only a brief summary here:

- Implemented in C++, making extensive use of modern language features such as templates, multiple inheritance, and operator overloading.
- Separate threads for high level behavioral control (Main) and low-level realtime control (Motion) which communicate via shared memory (Figure 1).
- Event-based, message passing architecture for communication among Tekkotsu components. (Tekkotsu provides its own specialized event router.)
- "Transparency" of operation, via a suite of GUI tools for robot monitoring and teleoperation. The tools are implemented in Java for portability, and run on a PC, communicating with the robot via wireless Ethernet.
- Hierarchical state machine formalism for defining complex behaviors, with fork/join operators for parallel execution.
- Pipelined low-level vision system, including color image segmentation and blob detection using CMVision (Bruce et al., 2000).
- Automated map building, using the dual-coding vision system.
- Localization, using a particle filter.
- Forward and inverse kinematics solvers.
- Inter-robot communication via an extension to the Tekkotsu message passing formalism.
- Human-robot interaction primitives using a remote display controlled by the robot.
- Simulator for debugging code on a PC.

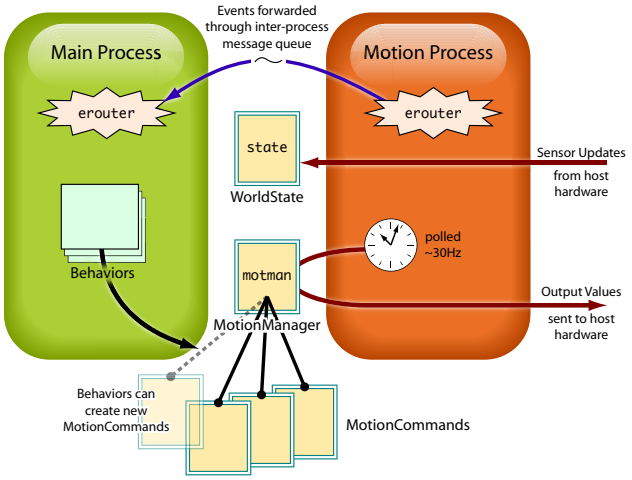

Figure 1: The Main thread is responsible for high-level deliberative behavior, while the Motion thread implements low-level realtime control, receiving sensor updates and controlling the robot's effectors at around 30 Hz. The two threads communicate via two types of shared memory structures: WorldState contains robot state information, and MotionCommands describe motions the robot is to perform. Behaviors running in Main can generate MotionCommands which are then executed by the motion manager running in Motion.

- Released as open source, free software under the Gnu Lesser General Public License (LGPL). The code is available at Tekkotsu.org.

Hardware Abstraction Layer

The imminent release of Tekkotsu 4.0 includes a new Hardware Abstraction Layer which allows behaviors to run on a wide variety of devices and architectures.

Each hardware device interface is a subclass of the DeviceDriver class (Figure 2). Drivers can receive motion instructions from Tekkotsu, and return sensor or image data (Figure 3. Drivers are provided for:

- SSC-32 servo controller from Lynx Motion
- Telepresence Robotics Kit (TeRK) interface for Charmed Labs' Qwerk board
- Previously recorded images and sensor values to be loaded from disk (for debugging and simulation)
- Live video from either local cameras or network streams

Where possible, communication between a device and Tekkotsu is abstracted by one of a variety of CommPort subclasses for interfaces such as file system devices, network sockets, serial ports, and binary executables (communicating via pipes). This allows us to separate the transport mechanism from the device protocol, increasing flexibility and reusability. For example, the SSC-32 servo controller can be sent commands either by writing them to a serial port on the local host, or by piping the commands over a net-

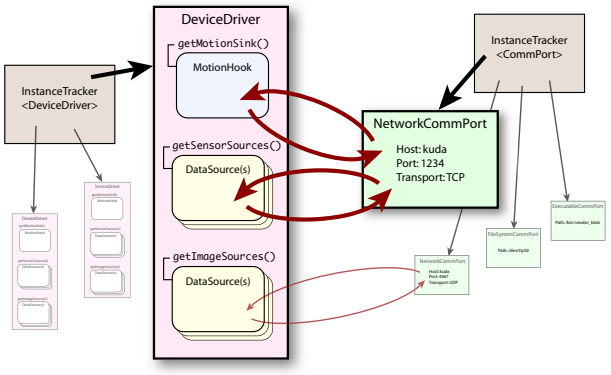

Figure 2: Device drivers and comm ports in Tekkotsu's hardware abstraction layer. The device driver knows how to format data for a servo controller board or image stream; the comm port knows how to transport the data over a socket, filesystem interface, serial port, or Unix pipe.

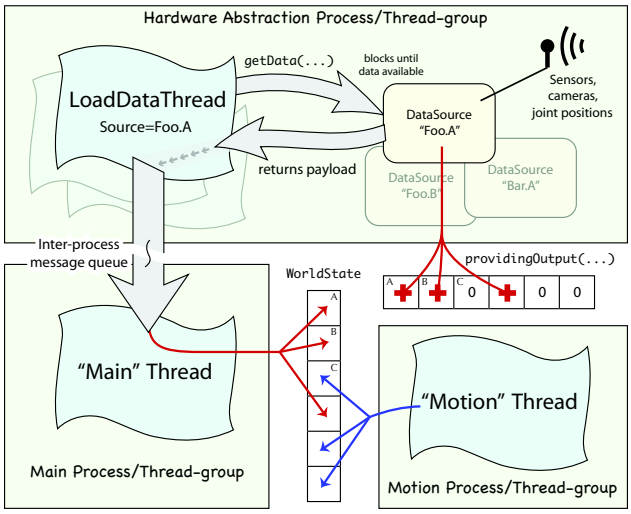

Figure 3: The data source Foo.A in the upper right is receiving sensor data, and has indicated it is providing feedback for some of the outputs (effector values). An example would be commanding a joint to move to a certain position; the feedback would be the actual position reported by the encoder. The payload from data source Foo.A is passed to the Main thread, which inserts the values into WorldState. Meanwhile, the Motion thread supplies values for other outputs; these will be commanded values, since the actual values aren't available (e.g., there may not be encoders on those joints.)

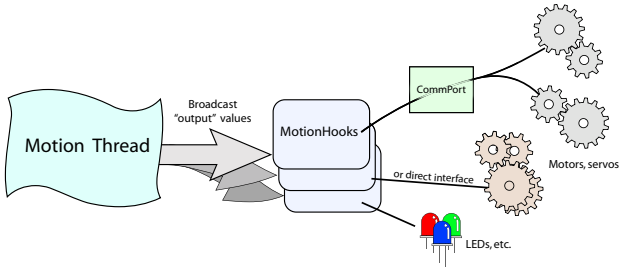

Figure 4: The Motion thread interprets MotionCommands and generates a stream of desired effector values at around 30 Hz. These values are then sent to the effectors, either directly if the device is local, or via a CommPort for remote devices.

work connection to a remote device (such as a Gumstix) that forwards them to the SSC-32.

In this way, a Tekkotsu process can control a variety of local and/or remote devices as desired by the user (Figure 4). The configuration of CommPorts and DeviceDrivers can be read from a file at launch, or dynamically reconfigured from the Tekkotsu command prompt. Users can switch between running on-board for reduced latency, off-board for maximum computational power, or even combinations of both for distributed control.

The Design of Regis

Regis is a prototype educational robot developed as an interim replacement for the AIBO, specifically for cognitive robotics (Figure 6). The principal design criteria for Regis are listed below:

Tabletop-friendly. The robot should be small enough to wander around comfortably on a tabletop. While larger platforms can hold more computers, sensors, batteries, etc., and are better suited for human-scale navigation tasks such as tour guides or office deliveries, they are impractical in a classroom setting where multiple robots will be in use at once. Regis is somewhat larger than an AIBO but can work on a tabletop.

Adequate computing power. The latest model AIBO, the ERS-7, used a 576 MHz RISC processor with 64MB of RAM. Regis uses a 600 MHz Gumstix verdex with 128 MB of RAM. The Gumstix runs Linux and uses the Gnu toolchain, so users can program in full C++.

Off the shelf components. Regis is constructed almost entirely from commercially available robotics hobbyist parts. It uses Hitec HS-645MG and HSR-5990TG servos, a Lynx Motion rover base, and two Lynx Motion servo erector set arms. The only locally fabricated parts are some acrylic plate extensions to the rover body to accommodate the geared servo in the arm shoulder; the webcam mount, constructed from two plastic bottle caps; and a low-profile serial port connector, necessary because of limited space inside the body.

Economical. The parts for Regis totaled around $1700, which is less than the $2K retail cost of an AIBO. Several

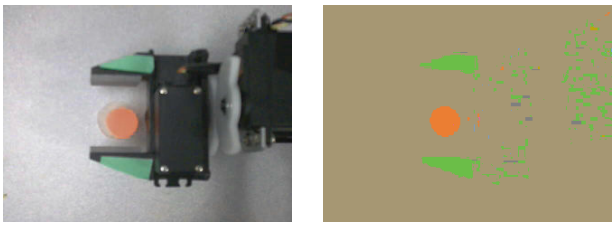

Figure 5: View of the gripper from the robot's webcam with the goose neck positioned overhead, looking down. Left: raw JPEG image; right: color segmented image from Tekkotsu vision pipeline.

types of legged robots proposed as AIBO replacements to the RoboCup federation in the summer of 2007 had costs estimated at around $4K, so Regis is competitive as a lower cost alternative.

Regis also has some unique features specific to its intended use in cognitive robotics instruction:

"Goose neck" webcam. Most robot designers put the camera in the wrong place. They mount it so that the robot can avoid obstacles, but cannot see much of its own body. This is a serious problem if the robot is expected to manipulate objects. Robots without grippers are still capable of manipulation by using their bodies to push against an object, but visual feedback is necessary if an object is to be positioned precisely. If the robot cannot see the point of contact between its body and the object, obtaining the necessary feedback becomes more difficult. For fine manipulation with a gripper the problem is much more acute.

Regis solves this problem by mounting the camera at the end of a long 4-dof arm called the "goose neck". The arm can rotate at the base; the remaining three degrees of freedom (shoulder, elbow, wrist) lie in the plane. The goose neck is long enough that it can lean forward and look down on the gripper (Figure 5). It can also point straight up and get a "big picture" view of the robot's workspace, or turn to the side and observe the robot's wheels.

"Crab arm" manipulator. A common design for a simple robot arm, such as the Rhino, or the Lynx Motion family of arms, is a three-link planar configuration mounted on a rotating base. The plane of the arm is perpendicular to the workspace, as this affords the greatest workspace area and allows the arm to reach over one object to get to another. However, this approach makes visual servoing difficult because the arm often obstructs the camera's view.

Regis's 6-dof arm uses a different design. It is called a crab arm because it extends from the front of the robot, not the top, and in crab mode it lies in the plane of the workspace. Because hobby servos have at most 180° of motion, the crab arm has a limited reach, but this can be increase by rotating the shoulder by 180° to flip the arm over. It is also possible to take the arm out of the plane of the workspace by rotating the shoulder by 90°, as shown in the bottom right image of Figure 6. However, in this vertical configuration the arm can travel forward/backward and up/down, but not left/right, so we would have to rely on the wheels to rotate the arm in the plane of the workspace.

The principal drawback of the crab arm design is that it places a fair amount of weight forward of the front wheels. Regis' batteries are mounted as far aft as possible, to act as a counterweight. The arm also extends the overall length of the robot by a considerable amount.

Operating in the plane of the workspace increases the chance of collisions with other objects. We will have to see how much of a problem this turns out to be in practice.

Future Work

In the coming year we expect to refine the design of our cognitive robotics platform by building either a second version of the Regis prototype or a hexapod walker with gripper. In either case, we will continue to use a goose neck webcam so the robot can view its own body.

We are also working on developing manipulation primitives that use visual servoing to grasp or push objects. This will require coordinated control of the goose neck and crab arm. One way to address this problem is to develop a set of stereotyped motion schemas with a small number of parameters each. For example, to grasp an object we may want to bring the goose neck in close in order to increase the resolution of the image. Once the object is firmly in hand, we will want to bring the goose neck to a more vertical position to increase the camera's field of view, and also to get it out of the way of the arm should we decide to switch from crab arm mode to vertical mode.

Once we are satisfied with our design, we plan to follow the example of Nourbakhsh et al. and publish a recipe for constructing this robot. This is why we have tried to restrict ourselves to off-the-shelf parts.

Tekkotsu's new hardware abstraction layer makes it easy to extend support to additional platforms, and we are in the process of adding support for the iRobot Create. The same Qwerk controller board that runs the Qwerkbot has been used to control a Create, but we may also develop a Gumstix-based solution so that Tekkotsu can run on-board the robot instead of teleoperating it.

Finally, we are continuing to add primitives to Tekkotsu's vision system. Currently we're working on adding a SIFT-like object recognition facility which can be integrated with the existing map building code.

Acknowledgments

This research was supported by the National Science Foundation's Broadening Participation in Computing program through award number 0540521 to DST.

References

Bruce, J., Balch, T., and Veloso, M. 2000. fast and inexpensive color image segmentation for interactive robots. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '00)*, volume 3, 2061–2066.

Gibson, J. J. 1979. *The Ecological Approach to Visual Perception.* Boston: Houghton Mifflin.

Figure 6: The Regis prototype posing with the second author. Note the variety of configurations of the "goose neck" webcam and "crab arm" manipulator with gripper.

Lynx Motion, Inc. 2007. Rover and robotic arm descriptions available at www.lynxmotion.com.

Nourbakhsh, I, Hamner, E., Lauwers, T., DiSalvo, C., and Bernstein, D. 2007. TeRK: A flexible tool for science and technology education. In *Proceedings of AAAI Spring Symposium on Robots and Robot Venues: Resources for AI Education*, Stanford, California, March 26–28, 2007.

Nourbakhsh, I., et al. 2007. Telepresence robotics kit: Qwerkbot+. Robot recipe available online at www.terk.ri.cmu.edu/recipes/qwerkbot+-recipe.php

Paivio, A. *Mental Representations: A Dual-Coding Approach.* New York: Oxford University Press.

Schmidt, R. A. 1975. A schema theory of discrete motor skill learning. *Psychological Review* 82:225–260.

Tira-Thompson, E. J. 2004. Tekkotsu: a rapid development framework for robotics. MS thesis, Robotic Institute, Carnegie Mellon University, May 2004. Available online at tekkotsu.org in the Bibliography section.

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

Ullman, S. 1984. Visual routines. *Cognition* 18:97–159.

Williams, A., Touretzky, D. S., Tira-Thompson, E. J., Manning, L., Boonthum, C., and Allen, C. S. 2007. Introducing an experimental cognitive robotics curriculum at historically black colleges. Manuscript under review.