Brian Thomas
Robots for Education (Chad Jenkins)
Brown University
Spring 2011

Images courtesy of www.tekkotsu.org,
www.robotshop.com
Core aims/motivations

- Handle routine low-level robot tasks
- Let developers focus on high-level programming
- Originally developed for AIBO but now supports a larger number of platforms.
Approach taken by Tekkotsu

- **Framework** for robot software development
- **Libraries** for routine tasks
- Made at **CMU**; licensed under the **LGPL**
"Performance and low overhead are important design considerations."

- Tekkotsu website
- Object-oriented
- Event-passing
- Want to expose both high-level and low-level controls
- => uses C++
Services Provided

- Visual processing
- Localization
- Forward/Inverse kinematics
- Real-time motion control
- Teleoperation
Supported Robots

- Aibo
- iRobot Create
- Chiara
- HandEye
- Lynxmotion Arms
- Qwerk
Supported Hardware

- Cameras (using video4linux)
- Lynxmotion SSC-32 servo controller
- Lynxmotion pan/tilt controller
- Bioloid actuators
Supported Algorithms

- State Machines (with a GUI viewer, Storyboard)
- Kinematics
- Dual coding (high-level computer vision routines)
- CMVision (color segmentation, blob detection)
- MapBuilder (2D)
- Particle Filtering
- Motion Modelling (dead reckoning)
- Tone/pitch detection
- Random number generator
Tekkotsu uses lots of 3\textsuperscript{rd} party code

- NEWMAT (matrix operations), libjpeg, libpng, libxml2, and zlib
- CMVision package by Jim Bruce for color segmentation and region grouping
- Aibo walk engine from Manuela Veloso’s 2002 RoboSoccer entry, CMPack’02
Where is Tekkotsu used?

- Past classes (Spring 2007 and previous)
  - **Carnegie Mellon University** (David Touretzky - Cognitive Robotics)
  - **University of Alberta** (Michael Bowling - CMPUT412: Experimental Mobile Robotics)
  - **SUNY Albany** (Prof. Tomek Strzalkowski - Robotics Seminar, Spring 2005)
  - **University of Pittsburgh** (Prof. Donald Chiarulli - CS 1567: Programming and System Design using a Mobile Robot)
  - **Lehigh University** (John Spletzer - CSE398/498, Spring 2005)
- And, apparently, some current classes as well.
Where is Tekkotsu used?

- Lots of **research institutions** have used (and maybe still use) it...
  - Bar-Ilan University - Israel
  - Carnegie Mellon University Tekkotsu Lab - U.S.
  - City University of Hong Kong - Hong Kong
  - Dutch ARchitecture Project for Aibos (DARPA) - Netherlands
  - Instituto Superior Técnico - Instituto de Sistemas e Robótica - Portugal
  - Lawrence Technological University - U.S.
  - Lehigh University - U.S.
  - Lund University - Sweden
  - National University of Singapore – Singapore
  - Spelman College - U.S.
  - SUNY Albany - U.S.
  - Università degli Studi di Messina - Italy
  - University of Alberta (2)- Canada
  - University of Applied Sciences Gießen-Friedberg - Germany
  - University of Edinburgh - Scotland
  - University of Iowa - U.S.
  - University of Minnesota - U.S.
  - University of New Orleans Robotics - U.S.
  - University of Pittsburgh - U.S.
  - Uppsala University - Sweden
...after the talk, due to wireless configuration.

- AIBOs communicate with a host computer through a common router.
#include "Shared/RobotInfo.h"
#endif TGT_HAS_HEAD

#include "StareAtBallBehavior.h"
#include "Events/EventRouter.h"
#include "Events/VisionObjectEvent.h"
#include "Shared/WorldState.h"
#include "Motion/HeadPointerMC.h"
#include "Motion/MMAccessors.h"
#include "Shared/ProjectInterface.h"
#include "Shared/ERS7Info.h"
#include "IPC/SharedObject.h"

// double registration, run on its own or in combination with SimpleChaseBallBehavior
REGISTER_BEHAVIOR_MENU(StareAtBallBehavior, DEFAULT_TELE/menu);
REGISTER_BEHAVIOR_MENU_OPT(StareAtBallBehavior, "Background Behaviors", BEH_NONEXCLUSIVE);

// Converts degrees to radians
inline double DtoR(double deg) { return (deg/180.0*M_PI); }

void StareAtBallBehavior::doStart() {
    BehaviorBase::doStart();
    headpointer_id = motman->addPersistentMotion(SharedObject<HeadPointerMC>();
    erouter->addEventListener(this, EventBase::visObjEIGID, ProjectInterface::visPinkBallSID);
}

void StareAtBallBehavior::doStop() {
    erouter->removeEventListener(this);
    motman->removeMotion(headpointer_id);
    BehaviorBase::doStop();
}

// this could be cleaned up event-wise (only use a timer when out of view)
void StareAtBallBehavior::doEvent() {
    float horiz=0, vert=0;
    if(event->getGeneratorID()==EventBase::visObjEIGID &amp; event->getTypeID()==EventBase::statusETID) {
        const VisionObjectEvent &amp; obj = static_cast&lt;const VisionObjectEvent &gt;(*event);
        vert = obj.getCenterY();
    }

    // for portability, look to see if the host hardware has a head pan &amp; tilt joints
    const unsigned int panIdx = capabilities.findOutputOffset(ERS7Info::outputNames[ERS7Info::HeadOffset+ERS7Info::PanOffset]);
    const unsigned int tiltIdx = capabilities.findOutputOffset(ERS7Info::outputNames[ERS7Info::HeadOffset+ERS7Info::TiltOffset]);
    if((panIdx===10) || (tiltIdx===10))
        return; // guess not...

    //cout &lt;&lt; horiz &lt;&lt; " " &lt;&lt; vert &lt;&lt; endl;

    // Very simple visual servoing control -- move the head a small distance in the direction of the target
    // This is *proportional* control, because we move the head proportionally further when the error (horiz and vert) is larger
    // so it homes in on the ball (here p=12, dist to move is e^r/2)
    // http://en.wikipedia.org/wiki/Proportional_control
    float tilt state = outputs[tiltIdx] - vert*CameraVertFOV/6;
    float pan = state - outputs[panIdx] - horiz*CameraHorizFOV/6;

    // now request access to the headpointer we added in doStart and set the joint angles
    MMAccessors&lt;HeadPointerMC&gt; headpointer(headpointer_id);
    #ifdef TGT_IS_AIBO
    if(RobotName == ERS7Info::TargetName) {
        // on an ers-7, we want to set the nod joint to look up (maximum value), since tilt can only look down
        headpointer.setJointAngles(tilt, pan, outputRanges[HeadOffset+HeadOffset][MaxRange]);
    } else {
        // on other models (we'll just assume ers-2xx), center the roll joint
    }
    #endif
// this could be cleaned up event-wise (only use a timer when out of view)

void StareAtBallBehavior::doEvent() {
    float horiz=0, vert=0;
    if(event->getGeneratorID()==EventBase::visObjEGID & event->getTypeID()==EventBase::statusETID) {
        VisionObjectEvent& objet=static_cast<const VisionObjectEvent&>(*event);
        horiz=objet.getCenterX();
        vert=objet.getCenterY();
    }

    // for portability, look to see if the host hardware has a head pan & tilt joints
    const unsigned int panIdx = capabilities.findOutputOffset(ERS7Info::outputNames[ERS7Info::HeadOffset+ERS7Info::PanOffset]);
    const unsigned int tiltIdx = capabilities.findOutputOffset(ERS7Info::outputNames[ERS7Info::HeadOffset+ERS7Info::TiltOffset]);
    if(panIdx==-1U || tiltIdx==-1U) {
        return; // guess not...
    }

    // cout << horiz << ' ' << vert << endl;

    // Very simple visual servoing control -- move the head a small distance in the direction of the target
    // This is "proportional" control, because we move the head proportionally further when the error (horiz and vert) is larger
    // so it homes in on the ball (here p=12, dist to move is err*FOV/2)
    // http://en.wikipedia.org/wiki/Proportional_control
    float tilt=state->outputs[tiltIdx].vert*CameraVertFOV/6;
    float pan=state->outputs[panIdx].horiz*CameraHorizFOV/6;

    // now request access to the headpointer we added in doStart and set the joint angles
    MNAccessor<HeadPointerMC> headpointer(headpointer_id);
    #ifdef TGT_IS_AIBO
    if(RobotName == ERS7Info::TargetName) {
        // on an ers-7, we want to set the nod joint to look up (maximum value), since tilt can only look down
        headpointer->setJoints(tilt,pan,outputRanges[HeadOffset+NodOffset][MaxRange]);
    } else {
        // on other models (we'll just assume ers-2xx), center the roll joint
        headpointer->setJoints(tilt,pan,0);
    }
    #endif

    /* really should do a kinematic solution with lookInDirection, but that assumes
    * user has done a .kin file for this robot. Let's just keep it simple and try to
    * set the joints directly */
    if(NumHeadJoints>2) {
        tilt/=2; // we're going to replicate the tilt parameter in the next call, so divide by 2
        headpointer->setJoints(tilt,pan,tilt);
    }
}
#include ...

/// Converts degrees to radians
inline double DtoR(double deg) { return (deg/180.0*M_PI); }

void StareAtBallBehavior::doStart() {
    BehaviorBase::doStart();
    ...
}

void StareAtBallBehavior::doStop() {
    ...
    BehaviorBase::doStop();
}

// this could be cleaned up event-wise (only use a timer when out of view)
void StareAtBallBehavior::doEvent() {
    ...
    // for portability, look to see if the host hardware has a head pan & tilt joints
    if(...) // not
        return; // guess not...
    ...
    // pan and tilt speeds by proportional servoing

#ifdef TGT_IS_AIBO
    if(RobotName == ERS7Info::TargetName) {
        // on an ers-7, we want to set the nod joint to look up (maximum value), since tilt can only look down
        ...
    } else {
        // on other models (we'll just assume ers-2xx), center the roll joint
        ...
    }
#else
    /* really should do a kinematic solution with lookInDirection, but that assumes
     * user has done a .kin file for this robot. Let's just keep it simple and try to
     * set the joints directly */
    ...
#endif