Introduction to Programing and Robotics

By Joaquin Pockels and Ramon Cardona

Robotics

Robotics Frameworks:

- CLARAty (NASA + CMU + JPL + UM)
- Robot Operating System (ROS)
- Microsoft Robotics Developer Studio (MRDS)
- Tekkotsu

What is Tekkotsu?

- Tekkotsu means 'iron bones' in Japanese
- Tekkotsu is an open source, free software project that builds on several third party libraries
- Services Tekkotsu provides include visual processing, localization, forward and inverse kinematics solvers, remote monitoring and teleoperation tools, and realtime motion control.

Developers

Carnegie Mellon University

- David S. Touretzky
- Ethan Tira-Thompson

Sony? And AIBO

Why Tekkotsu?

- A framework for constructing robotics applications.
- This software package aims to give you a structure on which to build, handling routine tasks so that you can focus on higher level programming.
- Free and opensource.

Why Tekkotsu?

- Basics: behaviors, events, motion commands, state machines
- Hardware platforms
- Dual-coding vision system
  - Graphical Interface: ControllerGUI, SketchGUI
- Kinematics and manipulation
- Locomotion and navigation

Why Tekkotsu?

--Low level robotics

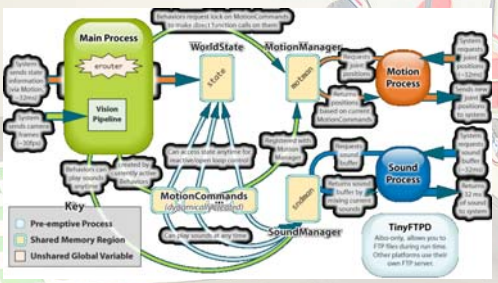
- Handles all the low-level details of running a robot
- Handles real-time programming duties
- Support for multiple hardware platforms

Why Tekkotsu?

--High level robotics

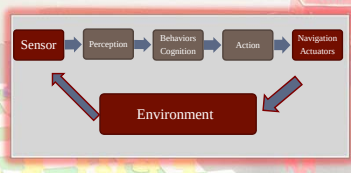
- Color image segmentation
- Dual-coding (iconic and symbolic) vision system
- Forward and inverse kinematics solvers
- Path planning
- Localization
- Navigation planner
- Text-to-speech
- etc...

How does Tekkotsu works?



How Tekkotsu works?

- In reality is very simple…



How Tekkotsu works?

- The Crew

The crew presently consists of:

- The Lookout
- The MapBuilder
- The Pilot
- The Grasper

How Run Tekkotsu?

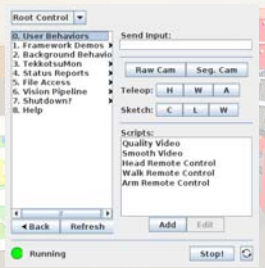
- Ubuntu 10.04 + (Linux)
- Terminal knowledge
- Tekkotsu and Tools:

After installation:

- Tekkotsu (project directory):
  ./tekkotsu-[robot name] [flags] [.plist file]
- ControllerGUI (tools/mon directory):
  ControllerGUI [host]
- Mirage (tools/mirage):
  Mirage [.mirage file](tools/mirage/worlds)

How Run Tekkotsu?

ControllerGUI:



How Run Tekkotsu?

Basic Unix commands:

- **cd [directoryname]:** go to that directory (folder)
- **cd .. :** go up a directory
- **cd ~ :** go to your home folder
- **cd / :** go to root
- **ls :** list files + folders in current directory
- **ps :** list current process
- **kill -9 [process #] :** stop a process
- **exit :** close terminal window

Useful hotkeys:
Crtl + c = kill current process
Crtl + Alt + t = open new terminal windows/tab
Crtl + Shift + c/p = terminal copy/paste

First things first…

Program Structure

- Statements to establish the start of the program
- Variable declaration
- Program statements (blocks of code such as functions)

Language	Example program

"C"	`#include <stdio.h>`
	`void main() {`
	`  printf("Hello World");`
	`}`

C++	`#include <iostream>`
	`int main()`
	`{`
	`cout << "Hello World";`
	`return 0;`
	`}`

Pascal	`program helloworld (output);`
	`begin`
	`  writeln('Hello World');`
	`end.`

Oracle PL/SQL	`CREATE OR REPLACE PROCEDURE helloworld AS`
	`BEGIN`
	`  DBMS_OUTPUT.PUT_LINE('Hello World');`
	`END;`

Java	`class helloworld`
	`{`
	`  public static void main (String args [])`
	`  {`
	`  System.out.println ("Hello World");`
	`  }`
	`}`

Perl	`#!/usr/local/bin/perl -w` `print "Hello World";`
Basic	`print "Hello World"`

Variable Declaration

Variables are place holders for data a program might use or manipulate. Variables are given names so that we can assign values to them and refer to them later to read the values. Variables typically store values of a given type. Types generally include:

- Integer - to store integer or "whole" numbers
- Real - to store real or fractional numbers (also called float to indicate a floating point number)
- Character - A single character such as a letter of the alphabet or punctuation.
- String - A collection of characters

Variable Declaration

```
#include <stdio.h>
void main() {
   int   age;
   float salary;
   char  middle_initial;

   age = 21;
   salary = 29521.12;
   middle_initial = "K";

   printf("I am %d years old ", age);
   printf("I make %f8.2 per year " salary);
   printf("My middle initial is %c ", middle_initial);
}
```

Boolean Algebra

Boolean logic is called a Two Valued Logic because an expression may only take on one of two values: True or False. An expression is some collection of logical operands and logical operators that are combined together.

Operators

The Boolean operators

Language	AND	OR	NOT
Mathematics	$\wedge$	$\vee$	$\neg$
"C" or C++	&&	||	!
SQL	AND	OR	NOT
Pascal	AND	OR	NOT
Perl	&	||	!
Java	&	||	!
Basic	AND	OR	NOT

Operators

Comparison Operators

Language	Equality	Greater than	Less than	Inequality
Mathematics	=	>	<	≠
"C" or C++	==	>	<	!=
Pascal	=	>	<	<>
SQL	=	>	<	<>
Perl	== or eq	>	<	!= or ne
Java	==	>	<	!=
Basic	=	>	<	!=

Operators

Comparison Operators

Question	Expression
Does Alice make more than $35,000 per year?	`Alice_Salary > 35000`
Did the NY Giants defeat the Dallas Cowboys?	`Giants_Points > Cowboys_Points`
Did anyone get a perfect score on the test?	`TestScore = 100`

Conditional Statements (IF..THEN..ELSE)

Programming Language	Conditional Code
"C", C++, Java, Perl	`if (condition) { statements } else { else_statements }`
Pascal	`if (condition) then begin statements end;`
Oracle PL/SQL	`if (condition) then statements else statements end if;`
Basic	`If condition Then statements Else statements End If`

Conditional Statements (IF..THEN..ELSE)

```
#include <stdio.h>
main() {
   int  department;
   department = 5;
   if (department == 5) {
      printf("Employees should be paid more than 25,000");
   } else {
     printf("Employees should be paid exactly 40,000");
   }
}
```

Finite-state Machines… What are they?

Mathematical model used to design computer programs and digital logic circuits.

The machine is in only one state at a time; the state it is in at any given time is called the current **state or node**. It can change from one state to another when initiated by a triggering event or condition, this is called a **transition**.

Each node describes an action the robot should take and are fired through the use of transitions.

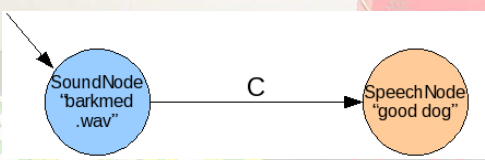
Finite-state Machines… What are they?

- Convenient control mechanism for writing applications.
- Both nodes and transitions are implemented as Behaviors.
- A few dozen built-in node and transition types
- Users can subclass these to define their own variants.

Finite-state Machines… What are they?

- StateNode is the base class:
  - SoundNode, SpeechNode
  - VisualRoutinesStateNode, MapBuilderNode
- StateNode is a child of BehaviorBase

Finite-state Machines… What are they?

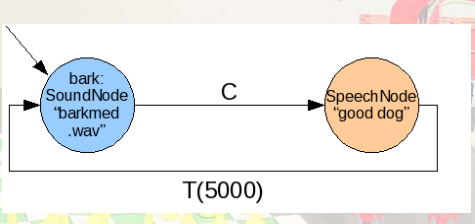


State Machine Shorthand Notation

```
bark: SoundNode("barkmed.wav")
speak: SpeechNode("good dog")

bark =C=> speak
```

Finite-state Machines… What are they?



State Machine Shorthand Notation

```
bark: SoundNode("barkmed.wav")
speak: SpeechNode("good dog")

bark =C=> speak =T(5000)=>bark
```

Types of Nodes

StateNode	No action. Parent for all other node classes.
SoundNode	Plays a sound, then signals completion.
SpeechNode	Speaks some text, then signals completion.
LedNode	Flashes the robot's LEDs.
PilotNode	Makes the robot move.
MapBuilderNode	Makes the robot look for things.

Types of Transitions

CompletionTrans	=C=>	Fires when the source node's action is complete.
TimeoutTrans	=T(n)=>	Fires after *n* milliseconds have elapsed.
NullTrans	=N=>	Fires immediately.

Types of Transitions

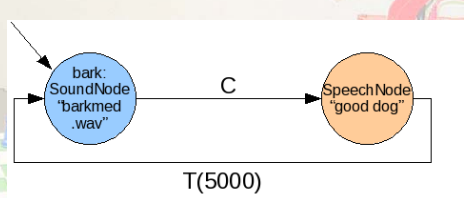
```
bark =T(3000)=> speak
```

```
bark =N=> speak
```

```
bark: SoundNode("barkmed.wav")
wait: StateNode
speak: SpeechNode("good dog")

bark =C=> wait
wait =T(3000)=> speak
```

Looping



The Crew

- **Map Builder** – constructs cameracentric, egocentric, or allocentric maps of the world.
- **Lookout** – moves the head: interest point detection, visual search, rangefinder scanning, object tracking, etc.
- **Pilot** – moves the body: walk; navigate to a point in space; localize (particle filter).
- **Grasper** – manipulate objects using an arm with gripper.

Activities

- **Teleoperation**: Use the ControllerGUI to drive the robot around; use the RawCam viewer to see the world from the robot's point of view.
- **Sound effects**: In the ControllerGUI, go to File Access > Play Sound and let students try the different sound files.
- **Talking robot**: In the ControllerGUI, go to Framework Demos > Interaction > Text to Speech and let students experiment with making the robot talk. After starting the demo, type "msg Hi there" in the Tekkotsu console to speak those words.

Activities

- **Obstacle course**: Set up an obstacle course or maze using cardboard boxes, empty soda cans, and similar objects, and have students use the Walk Control to navigate the robot through the obstacle course. For a more realistic (and harder) task: make them rely on the RawCam display instead of looking directly at the robot.
- **Robot soccer**: Use the Walk Control to try to push a ball into a goal. (You can make a goal by turning a cardboard box on its side.) If you have multiple robots, you can have an attacker and a defender.

Activities

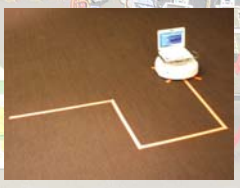
- **Color perception**: how does the robot see the world?
- Use the SegCam viewer to explore how the robot classifies colors. What effect does changing the room lighting have?
- Try this demo: Framework Demos > Vision > SeeShapes; show the robot some lines and ellipses.

Activities

- **Building and navigating virtual worlds**: Show students how to use the Mirage simulator and navigate through a world using the Mirage keyboard commands.
- Show students how to drive a robot around in a Mirage virtual world. Explore the different worlds.
- Introduce students to the Mirage world builder tools and show them how to make their own worlds.

Activities

- **State machine projects:** these projects can be done using only the built-in node classes, so students don't have to write any C++ code.
- Make the robot follows a sequence of visual/non visual instructions.



Activities

- Solve organization problems such as pushing cylinders into a pit.

Activities

- Pick up a small container and drop it somewhere else

Useful Links

Tekkotsu overall information:

- Tekkotsu web page: www.tekkotsu.org
- Tekkotsu Wiki: http://wiki.tekkotsu.org/index.php/Main_Page
- Tekkotsu installation : http://wiki.tekkotsu.org/index.php/Install_instructions
- Tekkotsu tutorials: http://wiki.tekkotsu.org/index.php/Tutorials