Navigating with the Pilot

15-494 Cognitive Robotics
David S. Touretzky & Ethan Tira-Thompson

Carnegie Mellon
Spring 2011
How Does the Robot Walk?

- Multiple walk engines incorporated into Tekkotsu:
  - CMPack '02 AIBO walk engine from Veloso et al. (CMU), with modifications by Ethan Tira-Thompson
  - UPennalizers AIBO walk engine from Lee et al. (U. Penn)
  - XWalk engine by Ethan Tira-Thompson for the Chiara

- Basic idea is the same:
  - Cyclic pattern of leg motions
  - Parameters control leg trajectory, body angle, etc.
  - Many different gaits are possible by varying phases of the legs
  - “Open loop” control: no force feedback
  - Can't adapt to rough terrain
  - Can move quickly, but not very accurately
ERS-7 Legs

<table>
<thead>
<tr>
<th></th>
<th>Δx</th>
<th>Δy</th>
<th>Δz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. - shoulder</td>
<td>65</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2. - elevator</td>
<td>0</td>
<td>0</td>
<td>62.5</td>
</tr>
<tr>
<td>3. - knee</td>
<td>69.5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>f4. - ball</td>
<td>69.987</td>
<td>-4.993</td>
<td>4.7</td>
</tr>
<tr>
<td>h4. - ball</td>
<td>67.681</td>
<td>-18.503</td>
<td>4.7</td>
</tr>
</tbody>
</table>

Diameter of ball of foot is 23.433 mm
Each link offset is relative to previous link

The shins shown in this diagram appear to be slightly distorted compared to a real robot.
Corresponding measurements have been taken from actual models.
Modified CMPack Walk Engine

46 Leg Parameters:

- Neutral kinematic position (3x4)
- Lift velocity (3x4)
- Lift time (1x4)
- Down velocity (3x4)
- Down time (1x4)
- Sag distance (1)
- Differential drive (1)

5 Body Parameters:

- Height of body (1)
- Angle of body (1)
- Hop amplitude (1)
- Sway amplitude (1)
- Walk period (1)

Modified from Sonia Chernova's lecture notes
Neutral Kinematic Position

- Position \((x,y,z)\) of the leg on the ground at some fixed point during the walk cycle.
- Where the legs would hit the ground if the robot were pacing in place (traveling with zero velocity).

From Sonia Chernova's lecture notes
Leg Lift and Leg Plant

- Lift velocity vector (mm/sec) determines how leg is lifted off the ground.
- Down velocity vector (mm/sec) determines how leg is placed back on the ground.
- Lift time and down time (1 value each per leg) control the order of leg motions.
  - Expressed as a percentage of time through the walk cycle that the leg is raised and lowered.
  - Governs which legs move together and which move at opposite times: pace vs. trot vs. gallop.

From Sonia Chernova's lecture notes
Body Angle/Height; Hop & Sway

- Body angle (radians) relative to the ground, measured at the origin of the motion coordinate frame.
  - Controls whether the robot is pitched up or down.

- Body height (mm) relative to the ground, measured at the origin of the motion coordinate frame.

- Hop and sway amplitudes (mm) constrain the body's vertical and horizontal oscillations during walking. (Usually set to 0.)

From Sonia Chernova's lecture notes
Walk Period

- The walk period (msec) specifies the time of one walk cycle.

- Note that this is independent of speed.

- To walk faster, the AIBO takes larger steps; it does not change the period of the walk cycle the way a person would do.

From Sonia Chernova's lecture notes

- Chiara walks are statically stable, and period does vary with speed.
New CMPack Parameter: Front & Back Leg Height Limits

- Height of the air path of the front and back legs.
- Upper bound: may not be reached, depending on other leg motion parameters.

From Sonia Chernova's lecture notes
Walk Parameter Optimization

- Many RoboCup groups use machine learning techniques to optimize walk parameters.

- CMPack uses a genetic algorithm.

- Candidates are evaluated by having the robot walk and measuring the results.

- CMPack got 20% speedup over previous hand-tuned gaits.
Tekkotsu Walk Editor

- Root Control > File Access > XWalk Edit
- Values are stored in a walk parameter file
  - Default parameter file is walk.plist
Chiara Gaits

• One leg at a time (default).
  – Requires the least power.
  – Slow: 6 beats/cycle.

• Two legs at a time.
  – Intermediate speed and power.
  – 3 beats/cycle.

• Three legs at a time: tripod gait.
  – Fastest gait that is still statically stable.
  – Requires lots of power.
  – 2 beats/cycle.
A Five-Legged Gait

• Sherene Campbell of Florida A&M University got the Chiara to walk on five legs so it could use its right front leg as a pincer. (See video on YouTube.)
XWalkMC

- XWalkMC is a motion command that uses the Chiara walk engine to calculate leg trajectories.

- Walking is controlled by three parameters:
  - x velocity (forward motion)
  - y velocity (lateral motion: strafing)
  - angular velocity (rotation)
WalkNode

- Subclass of StateNode
- Activates a WalkMC on start()
- Deactivates it on stop()
- Provides functions to set (x,y,a) velocities
- WalkNode($, xdisp, ydisp, adisp, time, WalkNode::DISP)
  - Displacements in mm and radians; time in sec.
  - Use a time of 0 to request maximum velocity.
- WalkNode($, xvel, yvel, avel, time, WalkNode::VEL)
  - Velocities xvel, yvel in mm/sec; avel in rad/sec; time in sec
Waypoint Engine

• Takes the robot through a path defined by a series of waypoints.
• Each waypoint specifies a position \((x,y)\) and orientation.
• Three waypoint types:

  - **Egocentric**
    - “Three steps forward”

  - **Offset**
    - “Three steps north”

  - **Absolute**
    - “To (30,12)”
Controlling Body Orientation

\( \text{angleIsRelative} == \text{true} \)

The angle is relative to the path, so an angle of 0 means the robot's body will **follow** the direction of travel.

\( \text{angleIsRelative} == \text{false} \)

The angle is relative to the world coordinate system, so the body will **hold** a constant heading while walking.
Arcing Trajectories

- Paths can be either straight lines or arcs.
- Arc parameter (in radians, not degrees) corresponds to the angle of the circle which is swept.
- Don't use values > 180°.
Track Path (Error Correction)

- `setCurPos()` function can be used to correct position if you have a localization module.
- When `trackPath` flag is true, the robot will attempt to return to its planned path after a perturbation.
- When false, it just goes straight to the destination.
Waypoint Walk Editor

- Root Control > File Access > WaypointWalk Control
- Allows interactive creation, execution of waypoint file.
# WyP
#add_{point|arc} {ego|off|abs} x_val y_val {hold|follow} angle_val
# speed_val arc_val
max_turn_speed 0.65
track_path 0
add_point EGO 0.3 0 FOLLOW 0 0.1 0
add_point EGO 0.5 0 FOLLOW 0 0.1 1
#END

Waypoint type
x, y or dx, dy (meters)
orientation
speed (m/sec.)
arc value (radians)
WaypointWalk

- WaypointWalk is a motion command.
- Can load waypoints from a waypoint file, or construct them dynamically with function calls.
- Uses a XWalkMC to do the actual walking.
- XWalkMC will post status events indicating the progress of the walk.
Manipulation by Walking

- Course project by Ethan Tira-Thompson

- Inspired by Matt Mason's “mobipulator” project.
The Pilot

• Higher level approach to locomotion.

• Specify effect to achieve, rather than mechanism:
  – Walk a certain distance.
  – Go to an object.

• Specify policies to use:
  – Cliff detection (IR sensor)
  – Obstacle avoidance (turn off to knock down soda cans)
  – Localization procedure

• Experimental code; changing rapidly.
Pilot Request Types

- **walk** – essentially a WalkMC request
- **waypointWalk** – provides Waypoint walk functionality
- **setVelocity** – set speed and go forever
- **localize** – look for landmarks and invoke the particle filter
- **goToShape** – path plan and travel to the location of a shape on the world map

- *More functions are planned...*
Trivial Pilot Example

$nodeclass MyPilotDemo : VisualRoutinesStateNode {
    $nodeclass Goer : PilotNode($, PilotTypes::walk) : doStart {
        pilotreq.dx = 500;   // forward half a meter
    }

    $setupmachine{
        Goer =PILOT=> SpeechNode($,"I have arrived")
    }
}

REGISTER_BEHAVIOR(MyPilotDemo);
Collision Detection

$nodeclass PilotLab3 : VisualRoutinesStateNode {

$nodeclass Forward500 : PilotNode($, PilotTypes::walk) : doStart {
  pilotreq.dx = 500;
  pilotreq.forwardSpeed = 100;  // speed 100 millimeters/second
}

$nodeclass Backup : PilotNode($, PilotTypes::walk) : doStart {
  pilotreq.dx = -100;         // negative displacement means back up
  pilotreq.forwardSpeed = 30; // speeds are always non-negative
}

$setupmachine {
  forward: Forward500
  forward =PILOT=> SpeechNode($,"done")
  forward =PILOT(collisionDetected)=>
    SpeechNode($,"Ouch! I hit something.") =C=> Backup
}
}
Path Planning

```c
$nodeclass DoIt : PilotNode($, PilotTypes::goToShape) : doStart {
    NEW_SHAPE(avoidMe, EllipseData,
             new EllipseData(worldShS, Point(250, 500, 0, allocentric),
                             80, 50, 0.5));
    avoidMe->setColor(rgb(255,0,0));

    NEW_SHAPE(destination, PointData,
              new PointData(worldShS, Point(700, 700, 0, allocentric)));
    destination->setObstacle(false);
    pilotreq.targetShape = destination;
}
```