Navigating with the Pilot

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

Carnegie Mellon
Spring 2007

How Does AIBO Walk?

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

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:

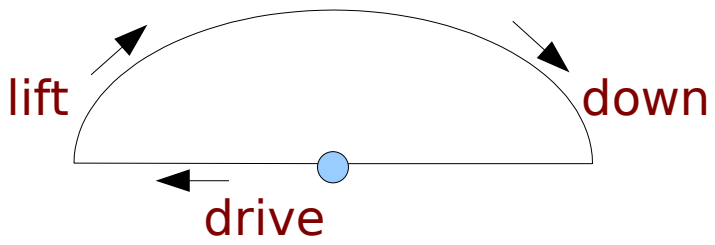
- Height of body (1)
- Angle of body (1)
- Hop amplitude (1)
- Sway amplitude (1)
- Walk period (1)

Modified fom 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 dog were pacing in place (traveling with zero velocity).

Path of the leg during one walk cycle



From Sonia Chernova's lecture notes

Leg Lift and Leg Plant

- Left 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

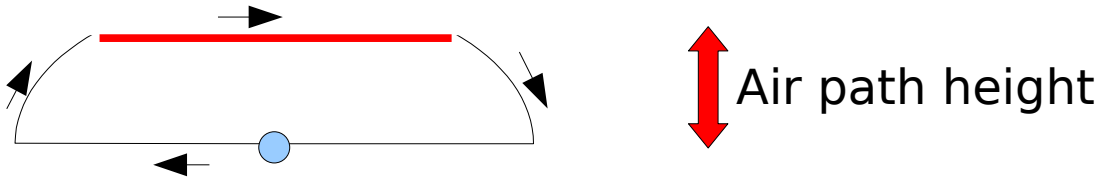
Walk Period

- The walk period (msec) specifies the time of one walk cycle.
- Note that this is independent of speed.
- To walk faster, the dog 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

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.



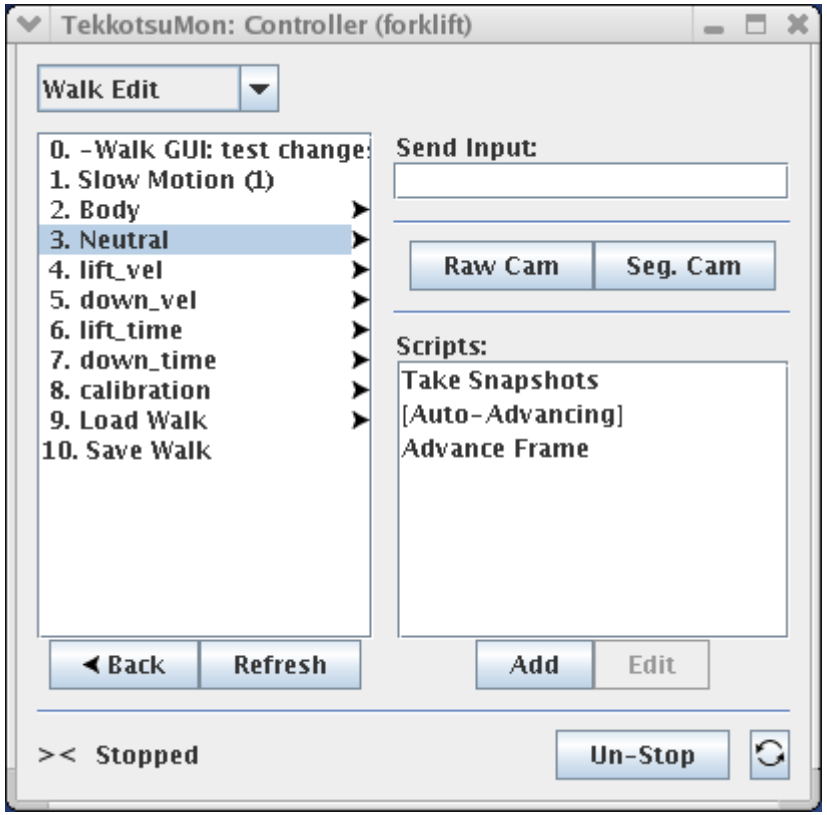
From Sonia Chernova's lecture notes

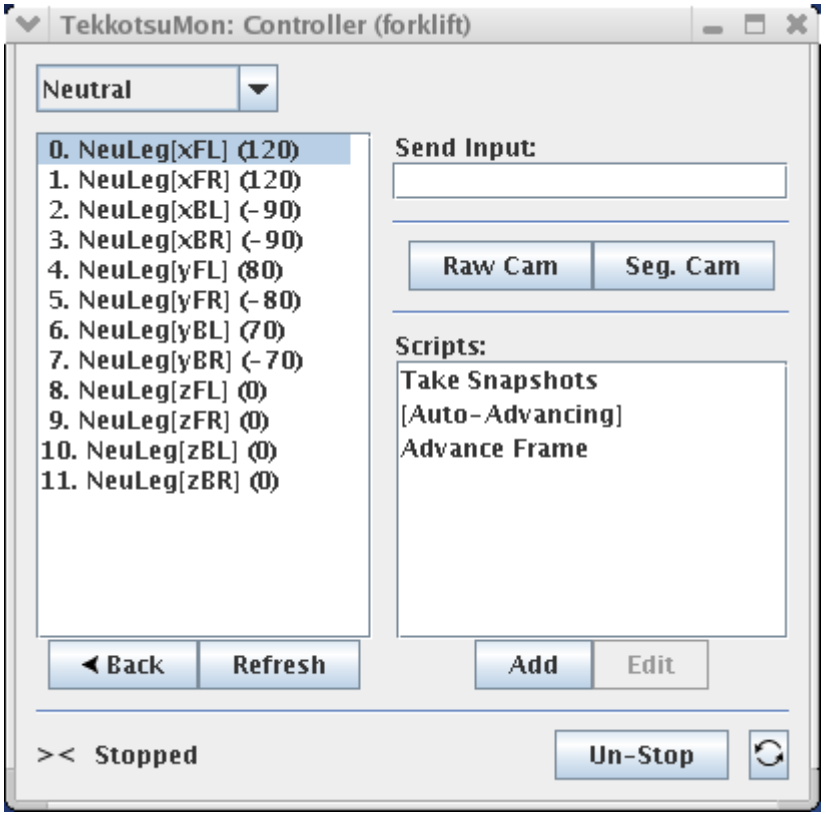
Walk Parameter Optimization

- 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 > Walk Edit
- Values are stored in a walk parameter file
  - Default parameter file is walk.prm

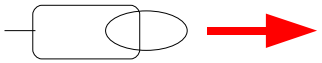




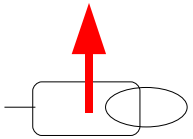
WalkMC

- WalkMC is a motion command that uses the CMPack walk engine to calculate leg trajectories.

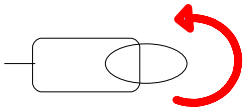
- Walking is controlled by four parameters:
  - x velocity (forward motion)



  - y velocity (lateral motion: strafing)



  - angular velocity (rotation)



  - number of steps (-1 = walk forever)

WalkNode

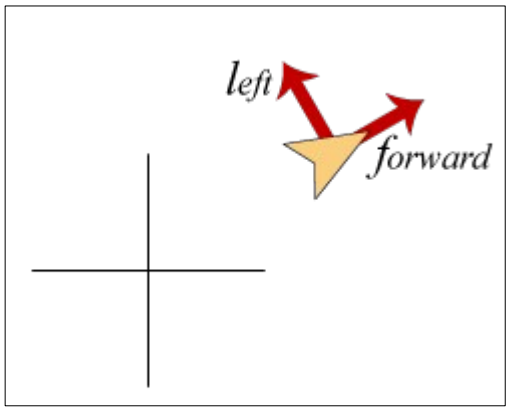
- Subclass of StateNode
- Activates a WalkMC on DoStart()
- Deactivates it on DoStop()
- Posts a status event when walk completes
- Provides functions to set (x,y,a) velocities, # steps, etc.

Walk Calibration

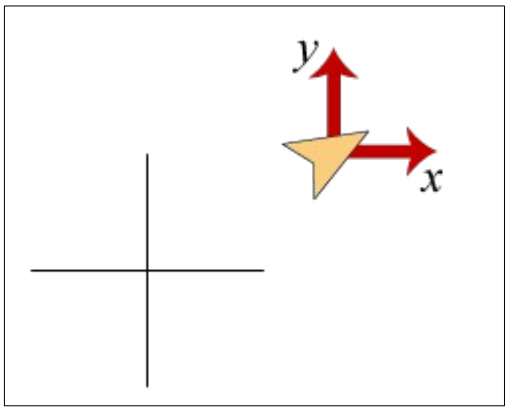
- The surface the dog is walking on will affect its motion.
- Robot peculiarities (e.g., one leg stronger than another) will also affect motion.
- Tekkotsu provides a walk calibration tool to compensate for these effects.
- Collect sample trajectories:
  - forward/strafe
  - forward/rotate
  - strafe/rotate
  - backward/rotate, backward/strafe
  - rotate
- Create walk calibration matrix; store in .prm file.

Waypoint Engine

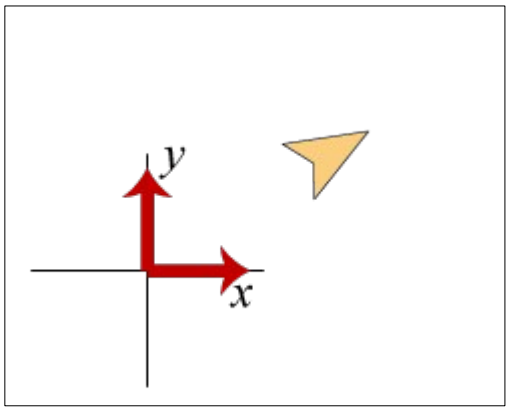
- Takes the dog through a path defined by a series of waypoints.
- Each waypoint specifies a position (x,y) and orientation.
- Three waypoint types:



Egocentric

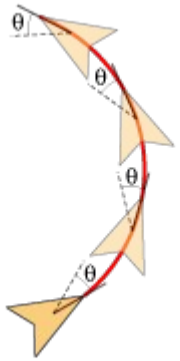


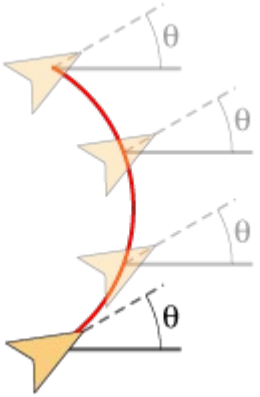
Offset



Absolute

Specifying Orientation





`angleIsRelative == true`

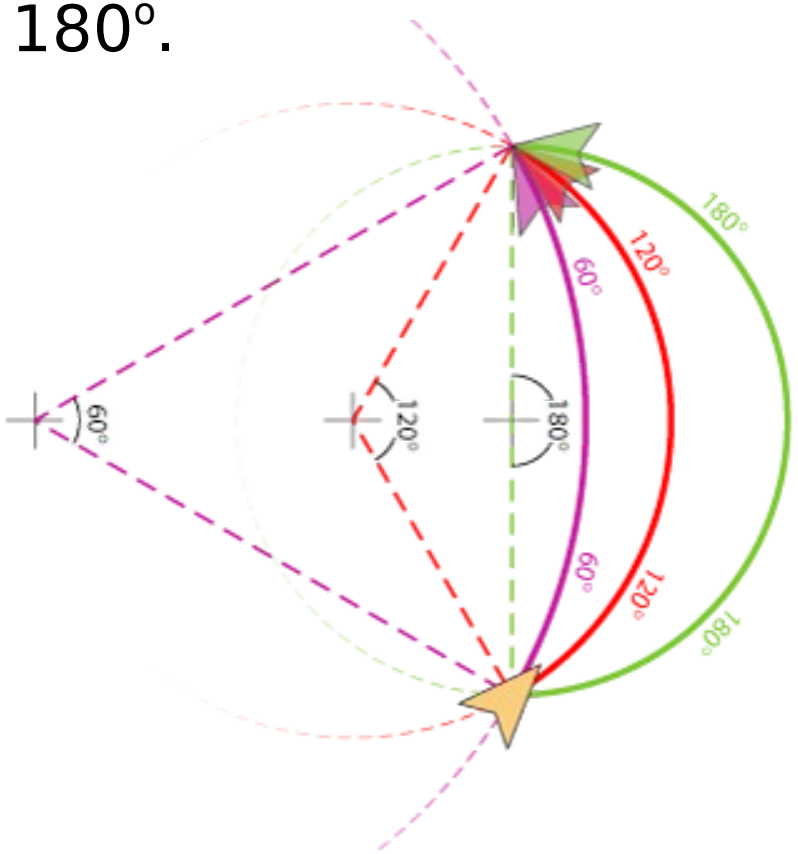
The angle is relative to the path, so an angle of 0 means the robot's head will **follow** the direction of travel.

`angleIsRelative == false`

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

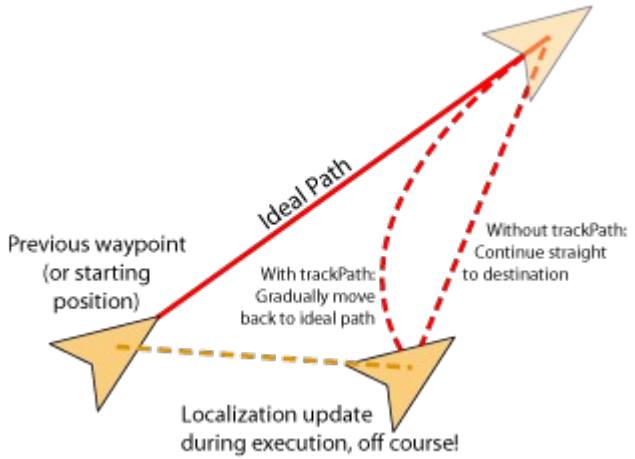
Arcing Trajectories

- 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 > 180º.



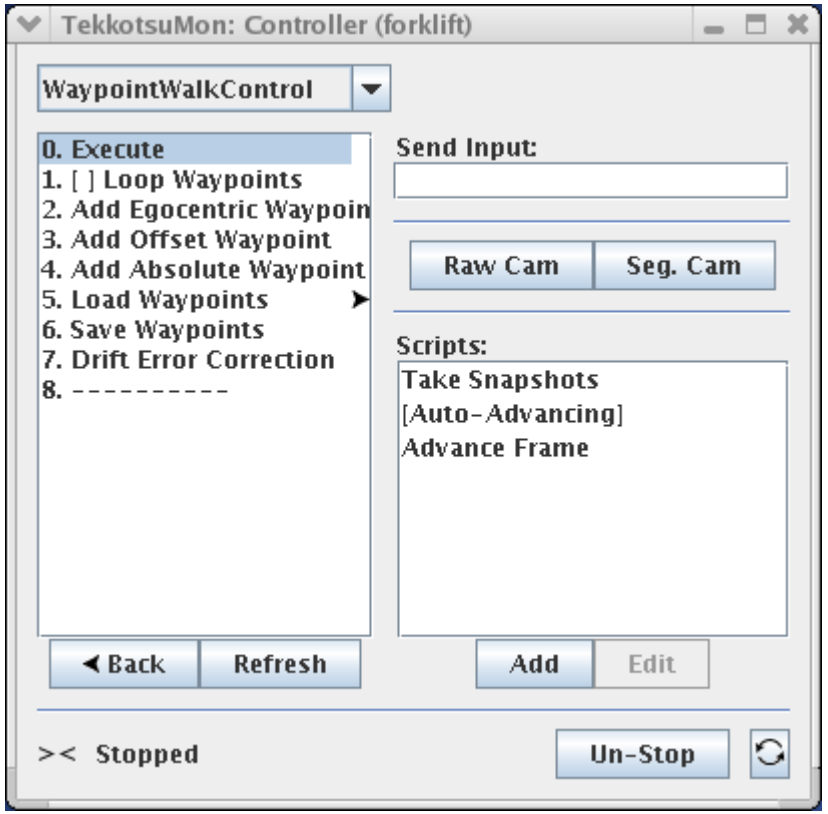
Track Path (Error Correction)

- 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.



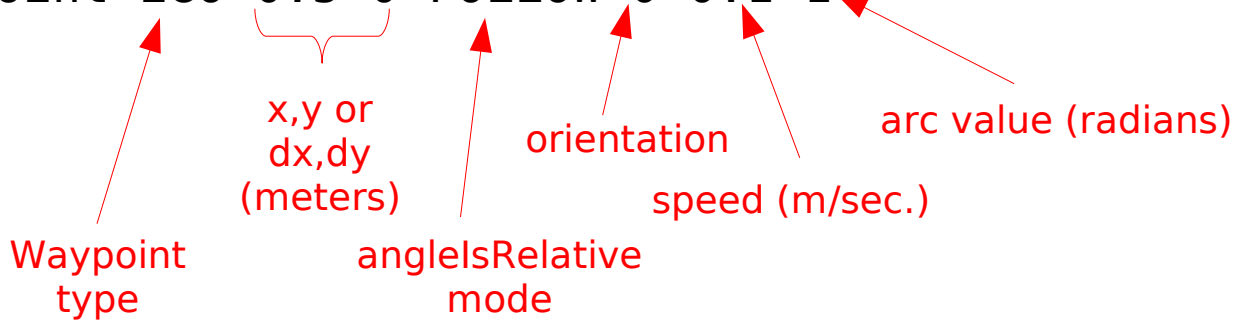
Waypoint Walk Editor

- Root Control > File Access > WaypointWalk Control
- Allows interactive creation, execution of waypoint file.



Sample 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
```



WaypointWalk

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

The Pilot

- Higher level approach to locomotion.
- Specify effect to achieve, rather than mechanism:
  - Go to an object.
  - Maintain a bearing or distance relative to an object.
- Specify policies to use:
  - Cliff detection (use chest IR)
  - Obstacle avoidance (turn off to knock down soda cans)
  - Localization procedure
- Experimental code; changing rapidly.

Example: Walk to Object

- Use Lookout to find and then track an object.
- Use pilot to walk toward the object Lookout is tracking.

```
virtual void DoStart() {
  VisualRoutinesBehavior::DoStart();

  lmSIDs.push_back(ProjectInterface::visPinkBallSID);

  erouter->addListener(this, EventBase::lookoutEGID);

  lookout.executeRequest(
     LookoutRequest::Find(findID,lmSIDs,false,true,ptVec));

}
```

Invoking the Pilot

```
virtual void processEvent(const EventBase& e) {
  if (e.getGeneratorID() != EventBase::lookoutEGID)
    return;

  switch (e.getSourceID()) {

  case findID:
    if (e.getTypeID() == EventBase::statusETID) { //lm found, track it
      dY = lookout.lm_location.coordY();
      dT = lookout.lm_orientation;
      cout << "dY,dT: " << dY << "," << dT << endl;

      pilot.executeRequest(PilotRequest::TranslateX(0,50,dY,dT));

      lookout.removeRequest(findID);
      lookout.executeRequest(
        LookoutRequest::Track(trackID, lmSIDs, false, false));
      }
      break;
```

Invoking the Pilot

```
case trackID:
  if (e.getTypeID() == EventBase::statusETID){ // lm updated
    dY = lookout.lm_location.coordY();
    dT = lookout.lm_orientation;
    cout << "dY,dT: " << dY << "," << dT << endl;
  }

  else if (e.getTypeID() == EventBase::deactivateETID){
    // lm lost, find it
    lookout.removeRequest(trackID);
    lookout.executeRequest(
      LookoutRequest::Find(findID, lmSIDs, false, true,ptVec));
    }
    break;

default:
    cout << "WalkToObjectBehavior::processEvent: Unknown Event\n";
  };
}
```

Manipulation by Walking

- Course project by Ethan Tira-Thompson

  http://ethan.tira-thompson.com/stuff/16-741/project.html

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