Aibo Greeters

Gabe Mulley
Matt Wilson
Elizabeth DeLouise
Trevor Houston
Goal

To create an ergonomic and useful robotic greeter, capable of interacting with the user in a relatively natural manner and providing them with information and assistance.
Desired Functionality

- Movement/Navigation
  - Pods, Waypoints
- Vision
  - Pods, Wall
- Speech
  - Speaker-independent keyword detection
- Ergonomics
  - User menu, Sounds, Actions
State Machine Outline
Navigation

- Three main steps went into creating the Navigation system
  - Design
  - Implementation
  - Transferring it into useful information for FIFO
Navigation design

- System needs a map that is easily accessed and will produce a representation of the path that will be traversed.

- System needed to traverse around corners and through slim spaces, so there may be many changes in direction when moving from point to point.

- System needed to be able to recalibrate itself in order to account for error.
Navigation design

- Two parts make up the core navigation system
  - A software map of safe places and paths that are traversable
  - A physical object recognized by the FIFO that has known coordinates
Navigation design

- Cartesian
- Polar
- Coordinate
- Directed Graph
- Waypoint
- Search
- Text File
  That loads the content for the safe paths through the world

World
Navigation implementation

- Cartesian, Polar, and Coordinate classes
  - Allows easy interface with different data types
  - Gives full functionality of any number class: adding, dividing, subtracting, multiplying
Waypoint class

- The nodes in the graph
- Have absolute position relative to axis in the world
- Can have a name which allows searching to and from that waypoint
Navigation implementation

- DirectedGraph class
  - Stores waypoints and a list of connections each waypoint has
  - Allows the search algorithm to find shortest path as well as minimizes storage
  - Contains DirectedGraphNode class that has no ability to be accessed by other classes
  - Templated class allows for storage of any data type
Navigation implementation

- **Search Class**
  - Presently, uses Depth First Search to search through the nodes
    - Returns the first path that it finds to a node
  - Will use Dijkstra’s Algorithm
    - Returns the shortest path
    - Allows for multiple path possibilities
Navigation implementation

- World class
  - Combines all the different elements into an application that is strictly used for FIFO in order to navigate through the world
  - Loads a text file and stores all the data in the map
  - Allows for the ability to step through the path that FIFO will travel
Navigation implementation

- Text file
  - .world extension
  - All of the waypoints have properties which can be set
  - To search to a waypoint, you set the name property
  - Connections are made based on the order that the other waypoints are declared
#Template for walking, save as .world

#set up number of waypoints

COUNT: 6

#waypoint 0
WAY: LOCATION: 0 0 CARTESIAN NAME: Bathroom CONNECTIONS: 3

#waypoint 1
WAY: LOCATION: 10 0 CARTESIAN NAME: Elevator CONNECTIONS: 3 4

#waypoint 2
WAY: LOCATION: 15 10 CARTESIAN NAME: Candy CONNECTIONS: 5

#waypoint 3
WAY: LOCATION: 5 0 CARTESIAN CONNECTIONS: 0 1

#waypoint 4
WAY: LOCATION: 15 0 CARTESIAN CONNECTIONS: 5 1

#waypoint 5
WAY: LOCATION: 15 5 CARTESIAN CONNECTIONS: 2 4
Transferring Movement to AI BO

- After a path is searched for, the points are put into Tekkotsu’s waypointEngine class.
- As soon as FIFO reaches the destined waypoint:
  - FIFO searches for a pod which will be at a fixed location.
  - This will allow him to re-orient himself in real space.
TREVOR Movement
Scan for Pod

- **Goal:**
  - Pan head from right to left
  - Vision object (pod) causes transition to next state
  - Can set values for the neck joint but…
  - How to make it pan continuously?
Scan for Pod

- `head_mc->setJointValue(…)`
  - Only uses last value set
  - Can’t do a motion sequence

- `state->outputs[…]`
  - Check actual position of head
  - Wait until head is at a certain position?
  - Doesn’t update in while loop

- **Small Motion Sequence “pan”**
  - `pan->setOutputCmd(…)`
  - Uses pre-set panning time
  - Executes all at once
Normalize with Wall

- Goal:
  - WalkToPod is vision-based, not coordinate-based
  - Need a way to re-calibrate dog’s orientation
  - ∴ Using IR sensor, find angle in relation to the wall and use it to turn
WallTestBehavior
- Calculates angle to/of Walls 0, 1, 2
- Line in slope-intercept form
- Relationship between slope and dog’s angle?
- Extra unnecessary calculations
Normalize with Wall

- NormalizeWallNode
  - Pan head (using Small Motion Sequence)
  - IR sensor calculates distances to wall
  - Record distances and corresponding head offsets
  - Calculate shortest distance (perpendicular) and corresponding head offset
  - Turn based on head offset
Vision: Goals

- Person Detection – when should it start trying to interact?
- Pod Detection – see them and “home” in on them
- Introduce as little error as possible
- Not mission critical, but would be nice
Vision : Status

- Currently able to see pods and move towards them
- Infrastructure in place for different colored pods, allowing the choice of a very uncommon color
- Pods enable dead reckoning motion between waypoints, prevent error from accumulating
Vision: Problems

- An attempt was made at introducing multi-color patterned pods to help false positives.
- This approach requires very careful color segmentation which there is no good way of doing.
- Too much error was introduced, making that approach almost useless.
- A simplification could be explored (fewer colors, simpler pattern, distance restriction).
Speech: Goals

- Be able to recognize a small set of isolated keywords
- Be relatively speaker-independent
- Setup an infrastructure that enables the dog to react to detected keywords
Speech : Pattern Recognition

Source: Rabiner, 71
Speech

Parameter Measurement

- Uniform filter bank implemented with short-time fast Fourier transform
- Restrict detection to vowels and easily recognized phonemes
Speech

Reference Pattern Generation and Comparison

- Reference patterns will be generated from the training set
- Comparison done with time shifting to account for different speech speeds
SpeechRecognizer runs in the background, does keyword detection and throws speechEGI D events when it detects words, encoding that word in the event so that the state transitions can tell whether to fire or not.

SpeechTrans is a state transition that fires when a keyword is detected allowing change of state based on keyword utterance.
Speech : Status

- Event generation is done (implemented with clap detection instead of word detection)
- FFT is done, a FFT can be run on each microphone frame or a sequence of frames
- Still have to do the hard part – parameter measurement, reference pattern generation and comparison
- Training set collection still needs to be done
Speech: Problems

- Noise – the microphone on the Aibo is of poor quality
- No WAV file saving functionality in Tekkotsu, makes debugging difficult
- A very complicated problem without a simple solution (no way to cheat)
- Will attempt to model in Matlab first, to make sure it will work
- Contingency plan?