# PSYCO 457

***Week 8: The Subsumption Architecture***

**Brooks' Philosophy**
**Introducing The Subsumption Architecture**
**Levels Of Control In AntiSLAM**

## Preliminary Discussion

- **Seeking comments or questions concerning the main themes of readings to this point in "From Bricks To Brains" and in "Embodied Cognition"**

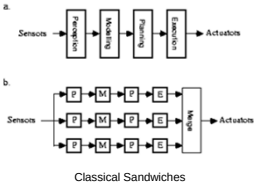

## Behavior-Based Robotics

- **No explicit knowledge representation**
  - **"*The world is its own best model*".**
- **Distributed control**
- **Sense-act cycle**
  - **Minimize sense-think-act processing!**
- **Higher-order competences are built upon existing lower-order competences**
  - **Principle of modular design**

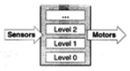
Rodney Brooks on how brains work

## Layers Of Behaviors

- **The *subsumption architecture* is an explicit reaction against the classical sandwich**
- **Subsumption architecture is built in layers**
- **Each layer has sensors and motors**
- **Lower layers provide low level competences**
- **Higher layers depend on lower layers, but might modify them**



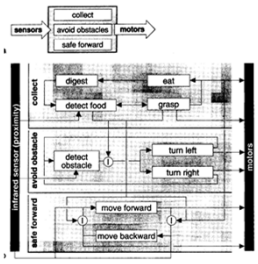

Classical Sandwiches




Subsumption Architecture

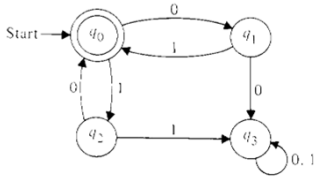
## Modular Design And Control

- **Within a layer in the subsumption architecture, capabilities are implemented as modules**
- **Modules in higher layers might modify (inhibit) activity of modules in lower layers**




## Control Properties

- **Within the subsumption architecture, layers are usually implemented as augmented finite state machines**
- **Layers operate in parallel**
- **Layers operate asynchronously**



## AntiSLAM

- **Our interest is a robot that uses the subsumption architecture for navigating**
- **We also study its behavior in a classic navigational setting, the reorientation task**

## SLAM: The Classical Navigator

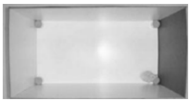
- **Typical accounts of navigation exploit disembodied sense-think-act processing**
- **Gallistel (1990, p. 121) notes "orienting towards points in the environment by virtue of the position the point occupies in the larger environmental framework is the rule rather than the exception and, thus, cognitive maps are ubiquitous."**
- **Similar accounts for robots, such as SLAM (simultaneous localization and mapping), are common**
- **"Low level robots may function quite adequately in their environment using simple reactive behaviors and random exploration, but more advanced capabilities require some type of mapping and navigation system" (Milford, 2008, p. 10).**

Randy Gallistel

Michael Milford

## The Reorientation Task

- **One aspect of navigation is studied by using the reorientation task**
  - **Find a reinforced location in an arena**
  - **Use geometric information (shape)**
  - **Use local information (wall color, landmarks)**
  - **Later, reorient one's self to the goal location when placed in new arena**
- **How is this accomplished? What cues are used? What happens when geometric and local cues conflict?**
- **One approach to answering these questions is very classical in nature**



## Rotational Error

- **Geometric cues on their own are ambiguous**
- **Rotational error: go to A and C mostly and equally, and rarely go to B or D**
- **Rotational error is often viewed as evidence of geometric cues being processed by a geometric module**

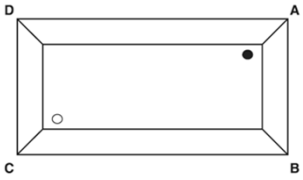
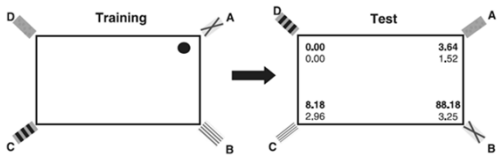



**Figure 1** *Schematic representation of the geometrical information which is available in a rectangular-shaped environment. The target (filled dot) stands in the same geometric relation to the shape of the environment as its rotational equivalent (open dot). Metric information (i.e. distinction between a short and a long wall) together with sense (i.e. distinction between left and right) suffices to distinguish between locations A-C and locations B-D, but not to distinguish between A and C (or between B and D).*

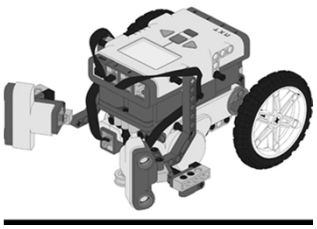
## Affine Transformation

- **When landmarks are rotated, this is equivalent to an affine transformation of the entire shape of the arena. How will the agent reorient – will it use the landmarks alone?**



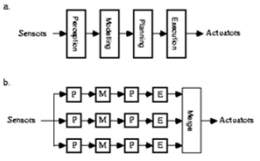

## antiSLAM's Behavior

- **This short video clip on YouTube provides an example of this robot's behavior both in the "real world" and in the restricted world of the reorientation arena**

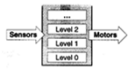


## AntiSLAM's Layers

- **The *subsumption architecture* is an explicit reaction against the classical sandwich, and was pioneered by Rodney Brooks**
- **antiSLAM is programmed using a particular instance of the subsumption architecture**




Classical Sandwiches




Subsumption Architecture

## antiSlam Level 0

- **"We start by building a complete robot control system which achieves level 0 competence. It is debugged thoroughly. We never alter that system" (Brooks, 1999, p.10)**
- **Our choice for Tortoise Level 0 is *drive***

```
/*=====Level 0: Drive==================================
Feed the distance from each ultrasonic sensor to a motor.
The robot is wired contralaterally, and thus avoids all walls
equally. As a result, when it reaches a corner, it slows down
and ends up stopping in the corner "for free".
*/

task DriveRight(){
  while(true){
    OnFwd(RightMotor, RightSpeed);
  }
}

task DriveLeft(){
  while(true){
    OnFwd(LeftMotor, LeftSpeed);
  }
}
```

## Level 1: Escape Corners

- **AntiSlam Level 1 is *escape***

```
/*=====Level 1: Escape=================================
If the motors move less than a stated threshold over a delay period,
the robot's sensors are temporarily overriden (zero sensitivity) as it
spins around. It ends up pointing approx. 45 degrees from the corner
when normal operation resumes.
*/

int Threshold, Delay;

task Retreat(){
  long RotCount;//Tracks motor rotation.
  while(true){
    RotCount = MotorRotationCount(LeftMotor) + MotorRotationCount(RightMotor);
    Wait(Delay);
    if((MotorRotationCount(LeftMotor)+MotorRotationCount(RightMotor)-RotCount)
        < Threshold){ //If, after Delay, the motors haven't moved enough:
      PlayTone(440, 500); //Beep to indicate feature flipping
      Sensitivity = 0; Reverse = 35; //Disable sensors, enable spin term
      Wait(4000);   //Time to spin in milliseconds
      Reverse = 0; Sensitivity = 1; //Return to default settings
      ResetRotationCount(LeftMotor); ResetRotationCount(RightMotor);
      Wait(500);
    }
  }
}
```

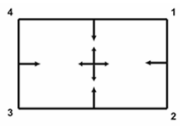
## Level 2: Follow Walls

- **AntiSlam Level 2 is *Follow Walls***

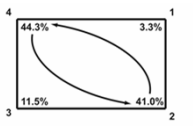
```
/*=====Level 2: Follow=================================
Introduce a bias in the robot's movement. This bias varies depending on its
preferred side (which sensor it reads), and results in the robot turning toward
a wall. The bias overcomes its natural wall aversion (level 0), causing it to
follow a wall more closely on one side.
*/

//Return the value of the sensor nearest the wall.
int Nearest(bool hand){ //Note: "nearest" is defined by which handedness the robot's using.
True = left.

  if (hand) return SensorUS(LeftEar);
  else return SensorUS(RightEar);
}

bool preferred;

task Seek(){
  int bias;
  while(true){
    bias = Nearest(preferred) * (-1) + 40; //Linear function: Wall dist. -> bias
    if (bias < 5) bias = 5; //Constraint: No zero or negative biases
    //Assign the bias to the correct motor and unbias the other one.
    if (preferred) {RightBias = bias; LeftBias = 0;}
    else {LeftBias = bias; RightBias = 0;}
  }
}
```

## antiSLAM's Rotational Error

- **Using Levels 0 through 2 alone, antiSLAM will generate rotational error, without using cognitive maps, and without relying on associative cues**




Starting States for antiSLAM




Locations of turnarounds

## Level 3: Move To Light

- **AntiSlam Level 3 is *light attraction***
- **Light sensors affect motors to attract robot to light, while interacting with other levels**
- **Nolfi's robots were not sensitive to features**
- **Now a lit corner can be described as the "place with the correct landmark"**

```
/*=====Level 3: Feature ================================================
Enables and reads the light sensors (eyes) as a percentage based on "Vision"
(a sensitivity term), such that more light = more speed. Since the connection is contralateral, this results in the
     robot turning toward sources of light.
However, level -1 weighs this visual sense with the earlier ultrasonic sense,
allowing both terms to influence the robot's final behavior. */
int Vision; //The strength of the light sensors in percent.
task See(){
//Sets the strength of the robot's visual response to a scaled percentage.
  while(true){
    LVis = Sensor(LeftEye)*Vision/100;
    RVis = Sensor(RightEye)*Vision/100;
  }
}
```

## Level -1: Integration

- **Using bricolage, we need to find some way of integrating competing signals into an integrated sense-act link to motors**
- **We typically use a 'Level -1' to do this; in other systems this would just be 'wiring'**

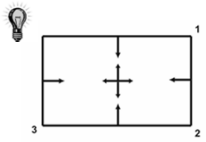
```
/*=====Level -1": Integration=================================================
Each of the terms (Sensitivity, Reverse, LeftBias, RightBias) is part of a later
level's connection to the motors. See the main task to see their defaults.
On its own, this task does nothing. However, it will function at every level
without modification.
*/

int Sensitivity,Reverse, LeftSpeed,RightSpeed, LeftBias,RightBias, LVis,RVis;
int Hearing;

task Drive(){
  while(true){
    //"Hearing/255" converts from responsive raw ultrasonic to % motor speed.
    RightSpeed = ((SensorUS(RightEar)*(Hearing-LeftBias)/255)+RVis)
                 * Sensitivity+Reverse;
    LeftSpeed = ((SensorUS(LeftEar)*(Hearing-RightBias)/255)+LVis)
                * Sensitivity-Reverse;
```

## Main Task

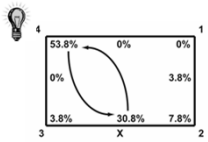
```
//=====Main Task==============================================================

task main(){
  //Set up ultrasonic sensors and speed calculation weights.
  SetSensorLowspeed(LeftEar);
  SetSensorLowspeed(RightEar);
  SetSensorMode(LeftEar, SENSOR_MODE_RAW);
  SetSensorMode(RightEar, SENSOR_MODE_RAW);
  Sensitivity = 1; //Level 0 connection: Ultrasonic sensitivity. Default 1.
  Reverse = 0; //Level 1 connection: Lets robot spin and escape. Default 0.
  LeftBias = 0; //Level 2 connection: Causes robot to prefer left turns. Def. 0.
  RightBias = 0;//Level 2 connection: As above, but prefers right turns. Def. 0.
  LVis = 0; //Impact of the left eye on movement. Zero at this level.
  RVis = 0; //Impact of the right eye on movement. Zero at this level.
  Hearing = 100; //Strength of ultrasonic sense. (Overridden at level 3.)
  start Drive; //Starts mapping the motor speeds to the collective input.
  //Level 0.
  start DriveRight; //Turn the right motor on.
  start DriveLeft;  //Turn the left motor on.
  //Level 1. (Delete below this line for a level 0 robot.)
  Threshold = 360; //Combined motor movement to be considered 'on'. Default 360.
  Delay = 5000; //How long the robot needs to have been stopped. Default 5000ms.
  start Retreat; //Allow the robot to escape corners.
  //Level 2. (Delete below this line for a level 1 robot.)
  preferred = true; //True for left-handed (right-following), false otherwise.
  bias = 40; //Fixed value for handedness bias. Default 40.
  start Seek; //Follow the wall on your preferred side.
  //Level 3. (Delete below this line for a level 2 robot.)
  //Set up eyes.
  SetSensorType(LeftEye, SENSOR_TYPE_LIGHT_INACTIVE);
  SetSensorMode(LeftEye, SENSOR_MODE_PERCENT);
  SetSensorType(RightEye, SENSOR_TYPE_LIGHT_INACTIVE);
  SetSensorMode(RightEye, SENSOR_MODE_PERCENT);
  Hearing = 40;// % of ultrasonic sense that feeds to the motors. Default 40.
  Vision = 60;// % of light sense that feeds to the motors. Default 60.
  start See;
}
```

## Light Cooperates With Geometry

- **We can test antiSLAM when the light cue cooperates with geometric information**
- **Using all 3 levels, antiSLAM prefers the lit corner**
- **Note how its trajectory is altered compared to the previous study**




Starting States for antiSLAM




Locations of turnarounds

## Light Competes With Geometry

- **When cues are in conflict, antiSLAM generates animal-like behavior that reflects combined influences of local and geometric features**
- **It prefers the light, but also generates rotational error**
- **It also generates very complex trajectories – data not typically reported in animal studies**
- **Note that all of this was obtained "for free" by building a robot that would follow walls, escape corners, and be attracted to light**
- **<u>Might navigation be scaffolded exploration?</u>**