

EXPLORE ROBOTICS – CISC 1003

CISC1003 – UNIT C

LOCOMOTION

Locomotion



Topics

- Modes of Locomotion
- Algorithm
- Multitasking

Locomotion





Locomotion

- Locomotion = locus (place) + motion
- Locomotion refers to the way a body moves
 - from place to place.
- A fundamental function of humans, animals
 - Acquired through training
 - Requiring significant “brain power”
- It's generally the first challenge for a robot
- Many modes of locomotion exist



Modes of Locomotion

- Legs:
 - Walking, crawling, climbing, jumping, hopping etc.
- Wheels:
 - Rolling
- Arms:
 - Swinging, crawling, climbing, lifting
- Wings:
 - Flying
- Flippers:
 - Swimming



Modes of Locomotion

- Most common, legged vs. Wheeled
- Benefits and challenges:
 - Wheeled:
 - Most efficient use of power, low DOFs.
 - Legged:
 - Large DOFs, challenge of stability.



Two Kinds of Stability

- **Static stability:** robots maintain upright without constant active control
 - Are humans statically stable?
 - We as humans are not statically stable!
 - Fall if fainting, etc.



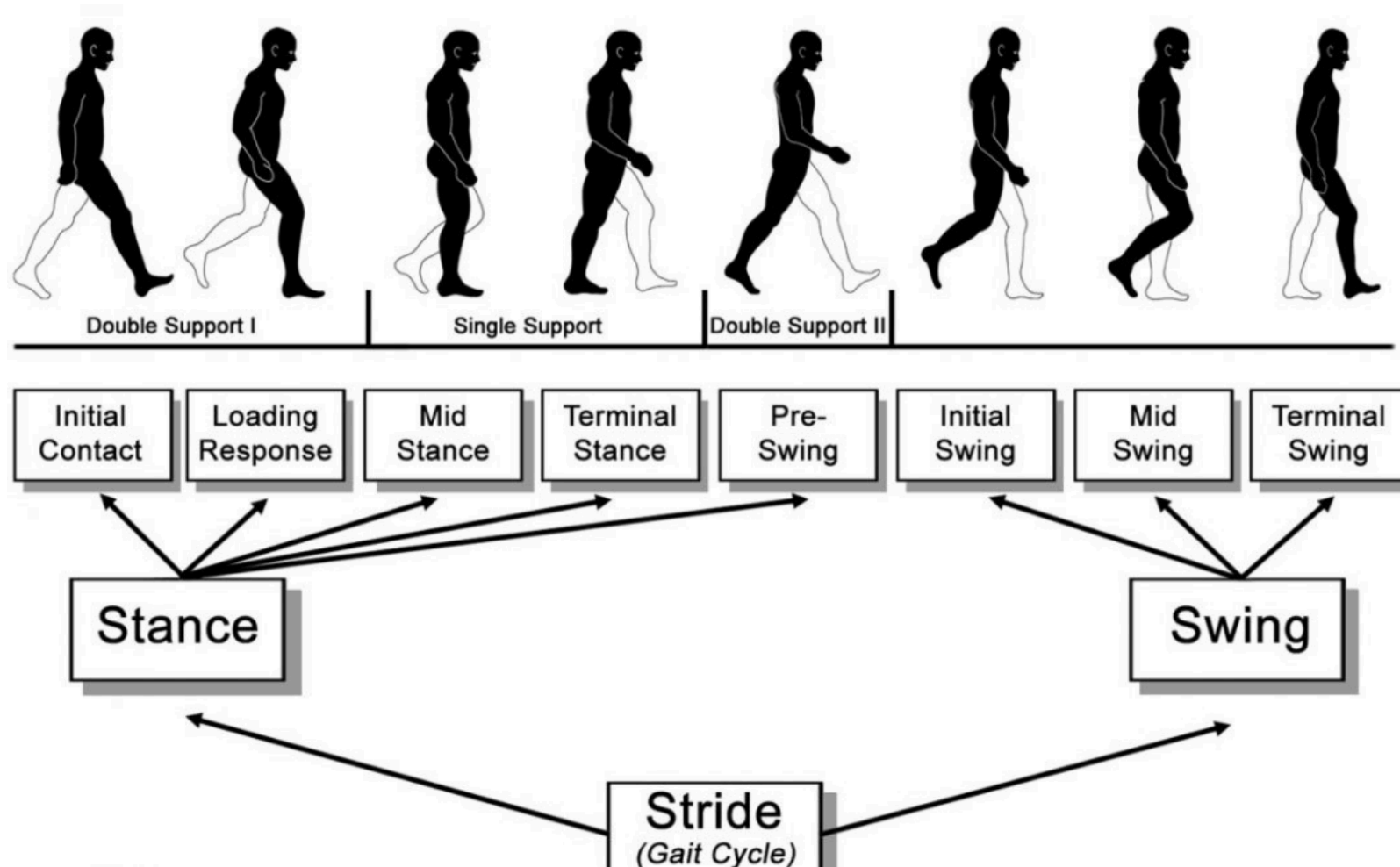
Two Kinds of Stability

- **Static stability:** robots maintain upright without constant active control
 - Maintained when center of gravity (COG) is above a certain horizontal region
 - Region called **support polygon**
 - horizontal region over which the center of mass must lie to achieve static stability
 - Statically stable walking is slow, energy inefficient



Two Kinds of Stability

- **Dynamic stability:** robots must actively balance or move to maintain stability
 - Two legged walking
 - alternates between swing and stance phase



- <https://www.protokinetics.com/2018/11/28/understanding-phases-of-the-gait-cycle/>



Two Kinds of Stability

- A statically stable robot can use dynamically stable walking to better use energy
 - tradeoff between **stability/speed**.

Gaits



- The way a robot moves by using a particular pattern of footfall
- Depending on the number of legs and choice of gait

Example of Robot Gaits

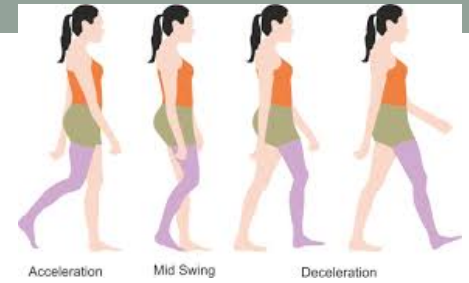
- 2 legged:
 - alternating swing and stance phases.
- 4 legged:
 - Diagonal walking: the feet on opposite sides move forward in sequence

Robot Gaits Examples



- 6 legged: alternating tripod gait vs. ripple gait.
 - Tripod gait: three legs move at a time
 - while the other three remain stationary
 - <https://www.youtube.com/watch?v=nRtJu4qrqn0>
 - Ripple gait: two legs from opposite sides shift each time
 - https://www.youtube.com/watch?v=3_Qk5svpUc0

Gaits



- Consideration for desirable robot gaits
 - Stability, speed, energy
 - Robustness, simplicity

Wheels and Steering



- Wheels are the choice of locomotion in robotics
 - Advantages of wheels:
 - Highly efficient
 - Simple to control
- Most wheeled robots are not holonomic

Wheels and Steering



- ***Motion planning*** = following a specific trajectory
- ***Navigation*** = moving from one place to another
- Which is more complex?
- Motion planning

Wheels and Steering



- Differential drive(steering):
 - Wheels are driven independently by separate motors => easier control.

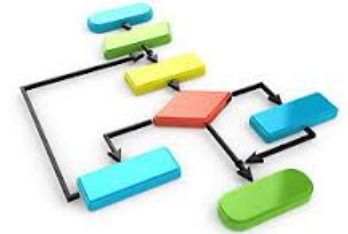
Go Beyond Locomotion - Dancing Automaton



- One or more robots come together
 - With music, dressed in costume
 - Moving in creative harmony.
- Need to develop an **algorithm**.
- Robot will be **multitasking**
 - allowing the program to perform more than one computer task at a time

https://www.youtube.com/watch?v=Fg0AGH_TaiQ

Algorithm



ComputerHope.com

- A step-by-step sequence of instructions for carrying out some task.
- Examples of algorithms outside of computing:
 - Cooking recipes
 - Dance steps
 - Proofs (mathematical or logical)
 - Solutions to mathematical problems
- Often, there is more than one way to solve a problem.

Algorithms -Solving problems

- In computing, algorithms are synonymous with problem solving.
- *How To Solve It* [George Polya, 1945]
 - Understand the problem
 - Devise a plan
 - Carry out your plan
 - Examine the solution

Algorithms –Polya[1945]

- Understand the problem:
 - Understand all the words, goal
 - Create a picture or a diagram to help solve
 - Is there enough information to solve the problem?
- Devise a plan
 - Choose a strategy: guess and check, eliminate possibilities, etc.

Algorithms –Polya[1945]

- Carry out your plan
 - Write the program, run the system
- Examine the solution
 - Look back, did you solve the problem?

Algorithms - features

- Speed (number of steps)
- Memory (size of work space)
- Complexity (can others understand it?)
- Parallelism (can you do more than one step at once?)

CASE STUDY – *BOIDS* ALGORITHM BY CRAIG REYNOLDS

Algorithm - *Boids* by Craig Reynolds

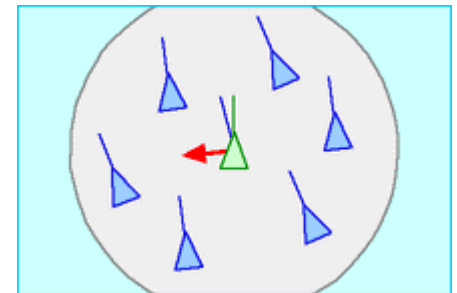
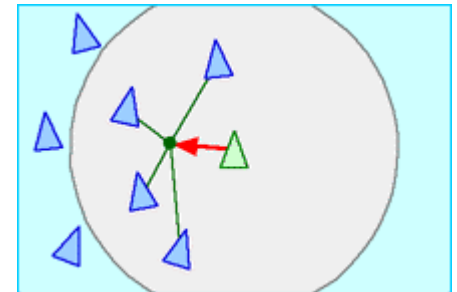
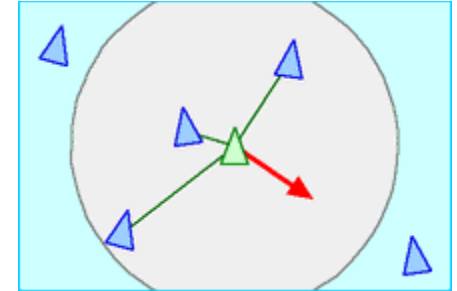
- Algorithmic for coordinated animal motion
 - Models steering behaviors
 - for animated flocking creatures.
 - Allowed individual elements to navigate their digital environments in a “life-like” manner
 - with strategies for different actions:
 - seeking, fleeing, wandering, arriving, pursuing, evading, path following, obstacle avoiding, etc.

Algorithm - *Boids* by Craig Reynolds (cont.)

- System has multiple characters
 - each steering according to simple locally-based rules,
- Surprising levels of complexity emerge
 - the most famous example being Reynolds' "boids" model for "flocking"/"swarming" behavior.

Algorithm - *Boids* by Craig Reynolds (cont.)

- Simple steering behaviors:
 - Separation:
 - avoid crowding neighbors
 - Alignment:
 - steer towards average heading of neighbors
 - Cohesion:
 - steer towards average position of neighbors



Algorithm - *Boids* by Craig Reynolds (cont.)

- An animated short featuring the boids model called **Stanley and Stella in: Breaking the Ice** was created
 - [Boids](#) video



INTRODUCTION TO PROGRAMMING

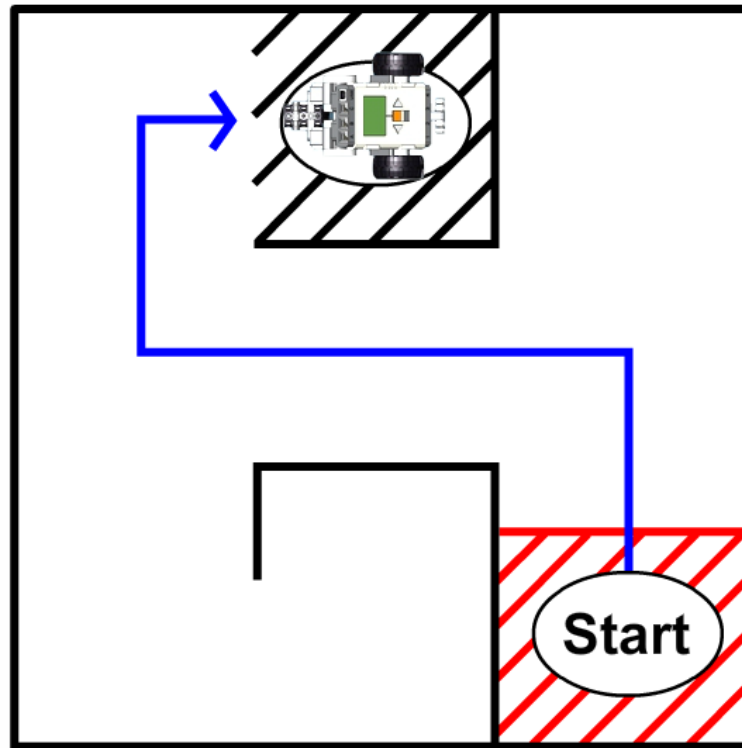
Planning and Behavior, Pseudocode

Planning and Behavior

- What is the problem?
 - Identify the behavior you need

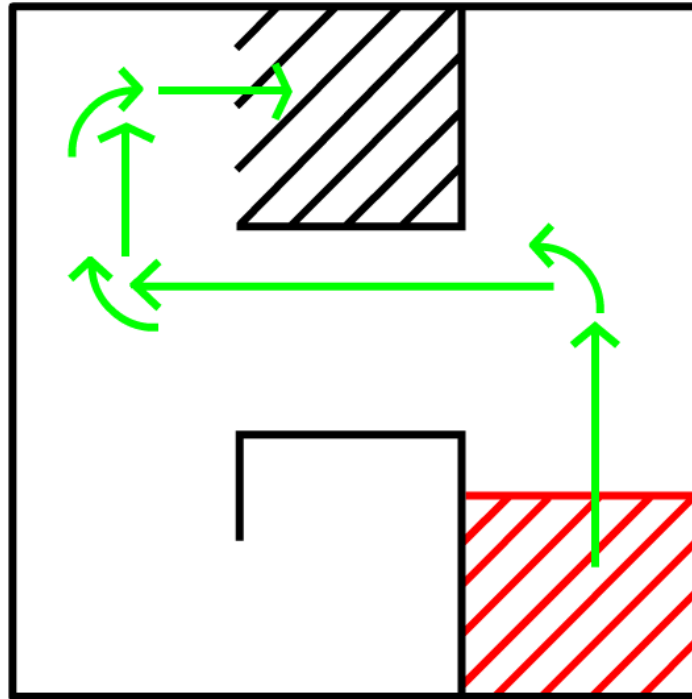
Planning and behavior

- Example: follow the path



Planning and behavior

- Break the main path into smaller paths:

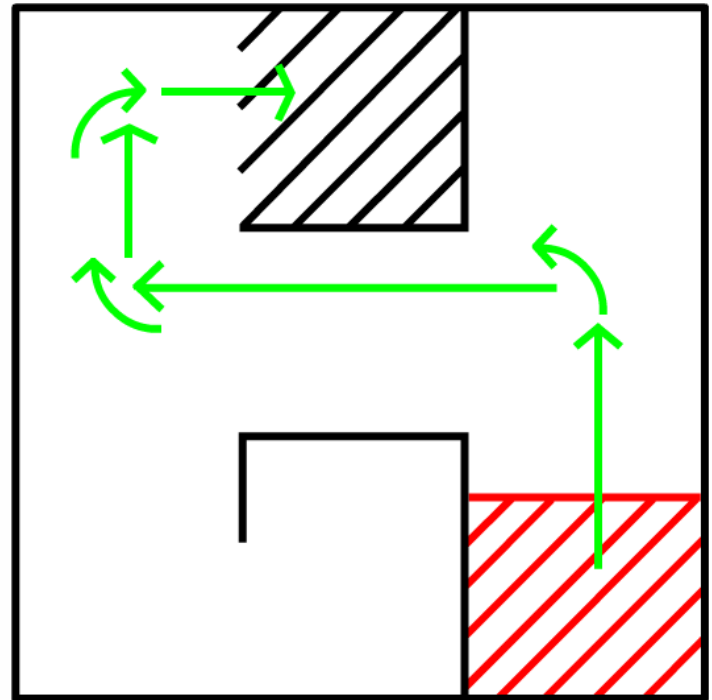


Planning and behavior

- Each of the smaller paths is called a behavior
- Write down the sequence of behaviors that is needed

Planning and behavior

- Follow the path:
 - Move forward
 - Turn left
 - Move forward
 - Turn right
 - Move forward
 - Turn right
 - Move forward

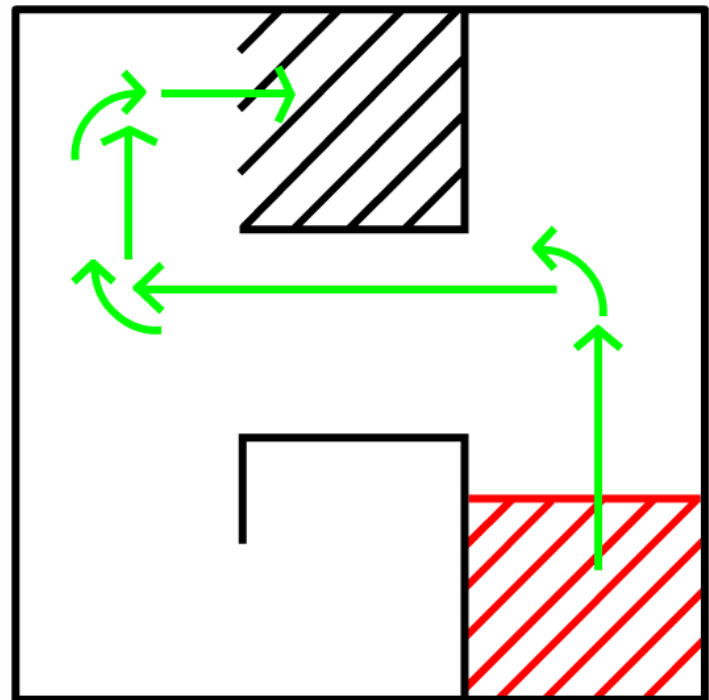


Planning and behavior

- Can we break these into smaller tasks?

Planning and behavior

- Follow the path:
 - Move forward
 - Left motor forward
 - Right motor forward
 - Wait 2 seconds
 - Turn left
 - Left motor reverse
 - Right motor forward
 - Wait 1 second
- Etc...



Pseudocode

- As we increase the level of details, we will reach commands we can express directly
 - in programming language
- This is the plan the robot needs to follow
- The steps are written in English
 - So can be understood by the human programmer
- This is called *Pseudocode*

Pseudocode Example

- Goal: prepare PB&J sandwich

Pseudocode Example

- Take exactly two pieces of bread.
- Take one piece of bread that is not covered with peanut butter on any side
- use a knife to spread peanut butter on one side
- Take a second piece of bread that is not covered with jelly on any side
- use a knife to spread jelly on one side

Pseudocode Example

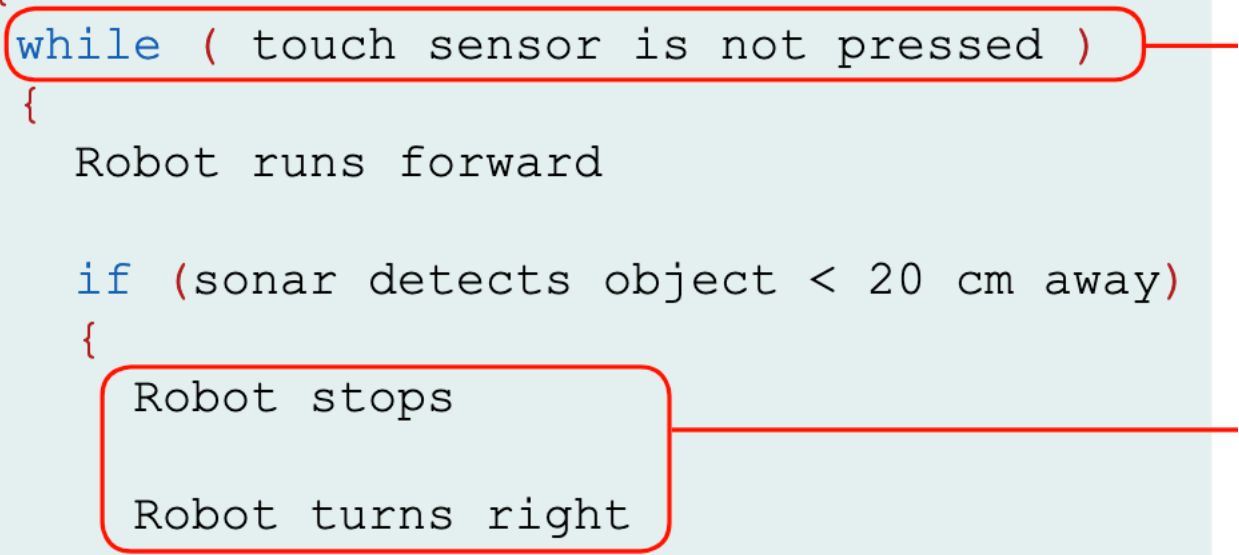
- Place the jelly side of the second piece of bread against the peanut butter side of the first piece of bread.
- Place the combined pieces of bread on plate

Pseudocode Example

- Robot runs forward. If an obstacle is detected within 20 cm distance, robot stops and turns right, and then continues moving forward. Robots deactivates when the touch sensor is pressed.

Pseudocode Example

```
task main()  
{  
  while ( touch sensor is not pressed )  
  {  
    Robot runs forward  
  
    if (sonar detects object < 20 cm away)  
    {  
      Robot stops  
      Robot turns right  
    }  
  }  
}
```



Pseudocode Example

- Can we improve this/suggest alternative solution?
 - Sense before moving?
 - Making sure we don't advance into a wall before sensor result is received.

Pseudocode Example

- task main()
- {
 - While (touch sensor is not pressed)
 - {
 - If (sonar detects object < 20 cm away)
 - {
 - Robot Stops;
 - Robot turns right;
 - }
 - Robot moves forward
 - }
 - }



LAB

- Let's start working with virtual robots!