RCAP CoSpace Autonomous Driving (Useful Functions)

A POSTERIORI

Play · Experience · Learn

Competition Timeline

- 22 May (Team Description and Video)
  - Submit Team Description Paper & Video
  - Template will be provided by email
- 23 to 26 May (Warm-up)
  - Warm up exercises (...not graded)
  - Helps you familiarize yourself with competition procedure
- 29 May (Preliminary games) (Saturday)
  - Given a fixed time to solve challenge map
  - Do from home
  - Details to be sent via email

Competition Timeline

- 31 May (Announcement of Finalist)
  - Notified via email
- Finalists: 3 Jun (Video submission)
  - Another video. This time describing the game strategy
- Finalists: 5 - 9 Jun (Interview)
  - Interview via Zoom
- Finalists: 10 Jun (Announcement of selected students)
- Finalist: 12 Jun (Grand Finals)

What we should have now...

- State Machine Pattern
  - Helps us handle challenge in a structured manner
- Actions
  - Line follower, stop, turn
- End conditions
  - Wait for Duration, isPurple, isOrange

What Else?

- Actions
  - move_steering(steering, speed)
    - Separates “steering” (curvature of turn) and “speed”
    - Allows you to change speed without changing how much the robot turns
    - You may already have the algorithm (...it’s in the proportional line follower)
  - gyro_follow(angle, speed)
    - Used when not following line
    - Situational; May be useful for shortcuts
  - turn_to_angle(angle)
    - Turns fast at start, then slow down when close to angle

What Else?

- End conditions
  - isColor(red, green, blue, tolerance)
    - Used by isPurple(), isOrange(), etc
  - isAngle(angle, tolerance)
    - Used by isNorth(), isEast(), etc
  - isLeftJunction(), isRightJunction(), isTJunction()
    - Detect junctions without colors

What Else?

- Depending on your map, these may or may not be useful

- Possible benefits...
  - Program faster (...less trial and error)
  - Take short cuts
  - Move faster

Gyro Follower

- Very similar to line follower

Line Follower

- Look at line position
- Decide whether to...
  - Turn left
  - Turn right
  - Go straight

Gyro Follower

- Look at gyro angle
- Decide whether to...
  - Turn left
  - Turn right
  - Go straight

- Main difference; We always keep the line in the center (value = 0), but for gyro, the value depends on the direction we are following

Gyro Follower

- Uses the same algorithms as line follower (eg. 2 States, 3 States, Proportional)

```
void gyro_follow(angle, speed)
{
  if (RotationZ > angle) {
    // Turn Right
    WheelLeft = 20;
    WheelRight = -20;
  } else {
    // Turn Left
    WheelLeft = -20;
    WheelRight = 20;
  }
}
```

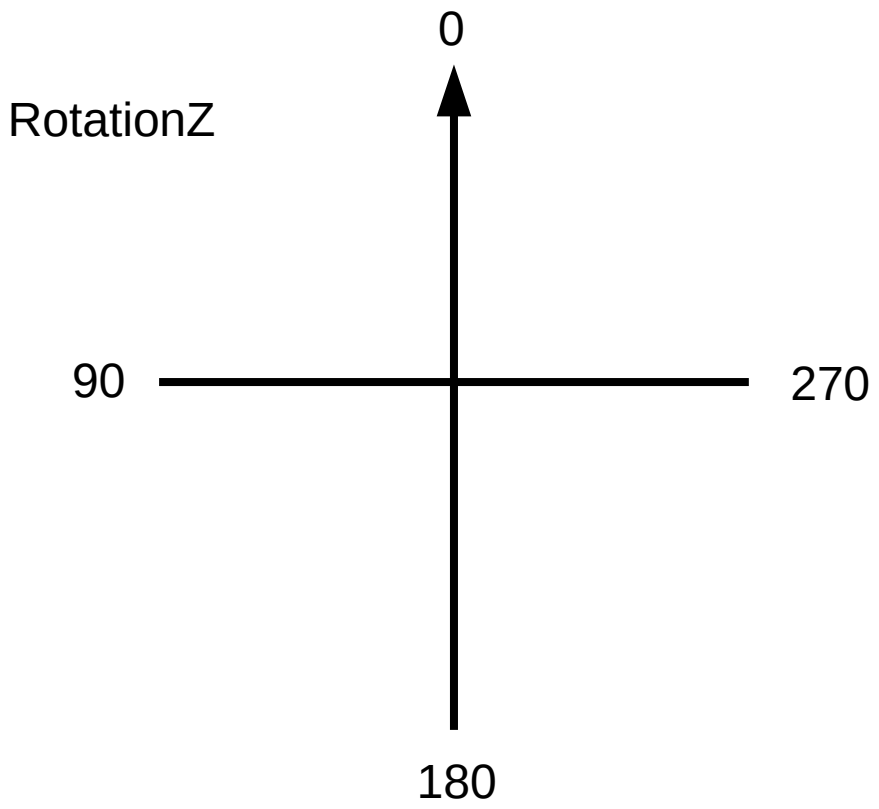
2 States

```
void gyro_follow(angle, speed)
{
  if (RotationZ > angle + 1) {
    // Turn Right
    WheelLeft = 20;
    WheelRight = -20;
  } else if (RotationZ > angle - 1) {
    // Go straight
    WheelLeft = 20;
    WheelRight = 20;
  } else {
    // Turn Left
    WheelLeft = -20;
    WheelRight = 20;
  }
}
```

3 States

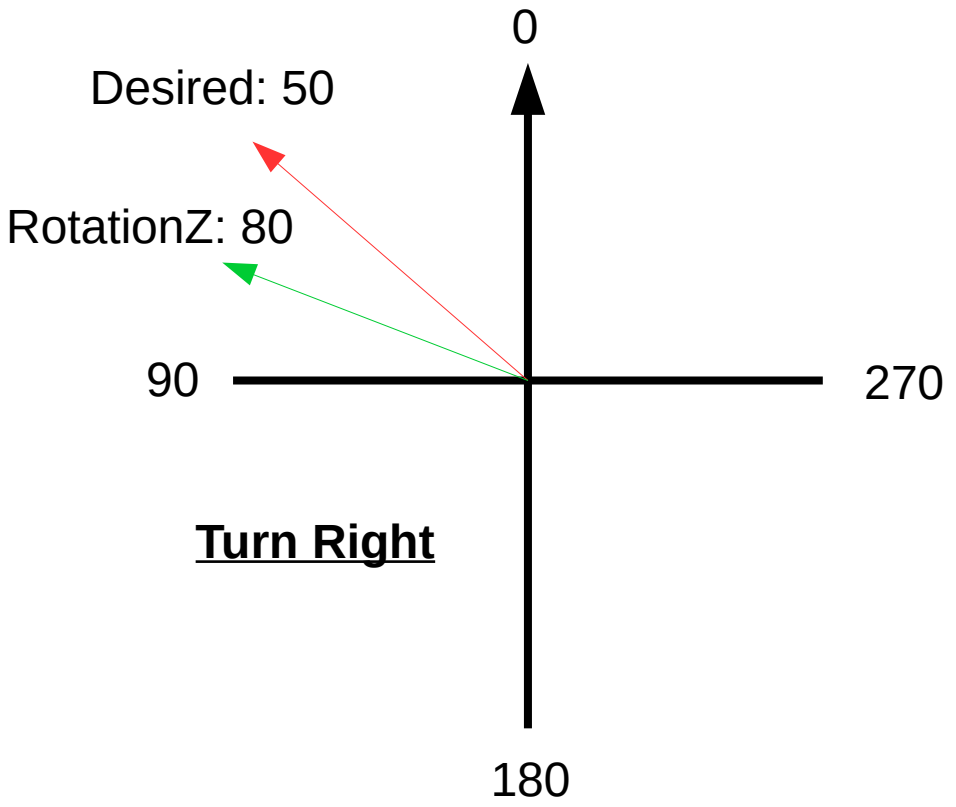
Problem 1

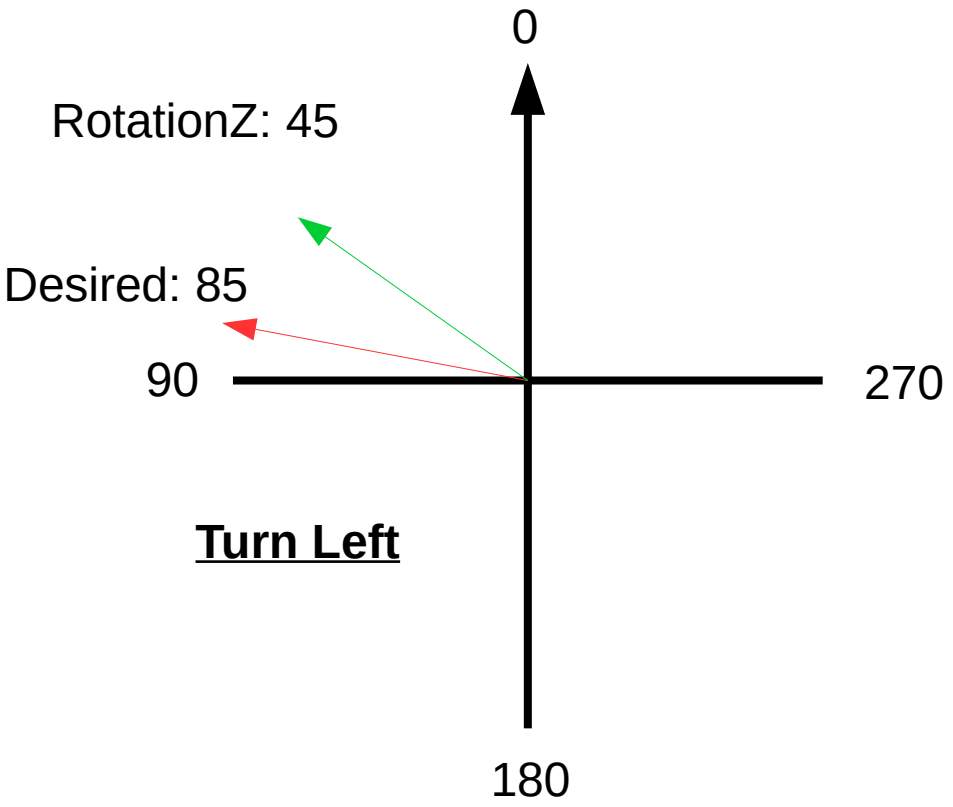
The angles in Cospace are messed up...



Solution 1

- Not a big deal
- Just be aware of the directions...
  - If RotationZ > direction you want: Turn Right
  - If RotationZ < direction you want: Turn Left





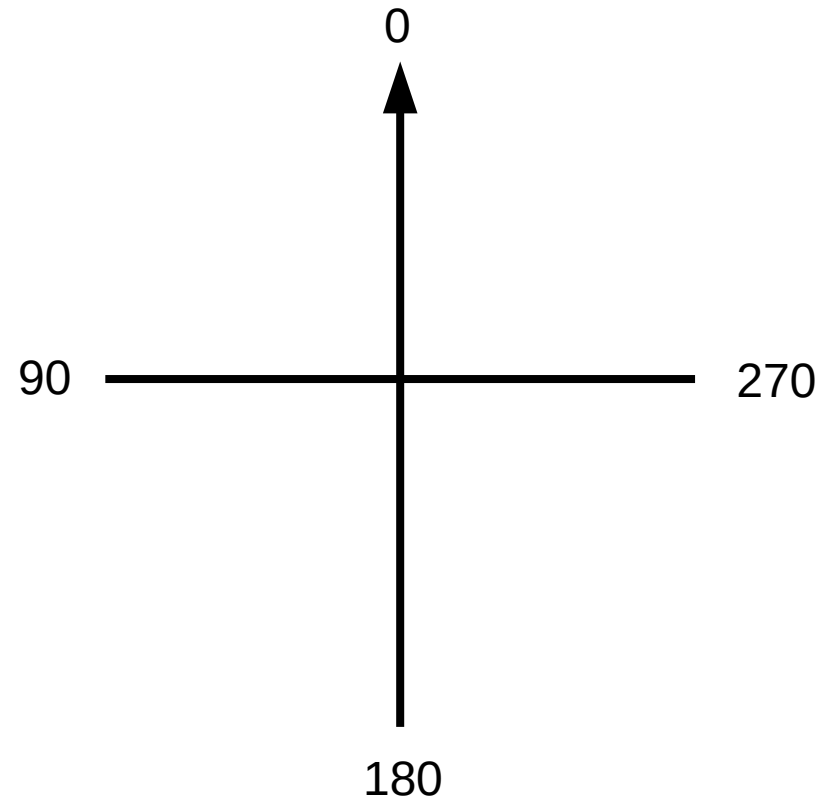
Problem 2

- Angles jumps from 0 to 359 when you turn right
- Consider...
  - Angle you want to follow: 0 degrees
  - If robot is at 5 degrees, it should turn right
  - If robot is at 355 degrees, it should turn left
  - ...but 5 and 355 are both larger than 0!

Solution 2

- Modify RotationZ so that angles on the left are always larger, and angles on the right are always smaller

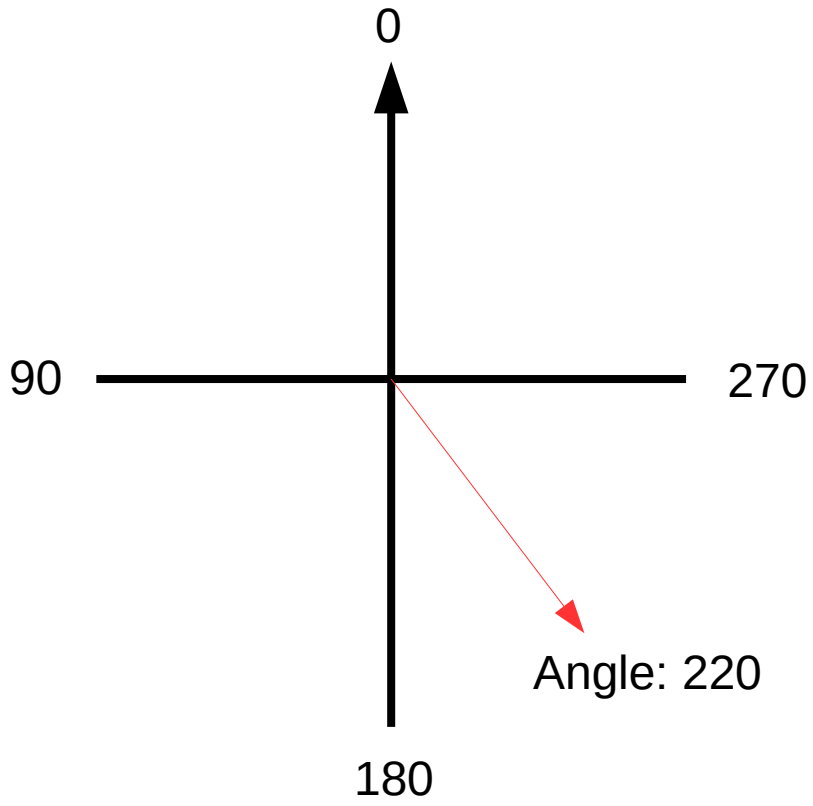
```
if (angle > 180) {
  if (RotationZ < (angle - 180)) {
    RotationZ += 360;
  }
} else {
  if (RotationZ > (angle + 180)) {
    RotationZ -= 360;
  }
}
```



Solution 2

- If desired angle is > 180...

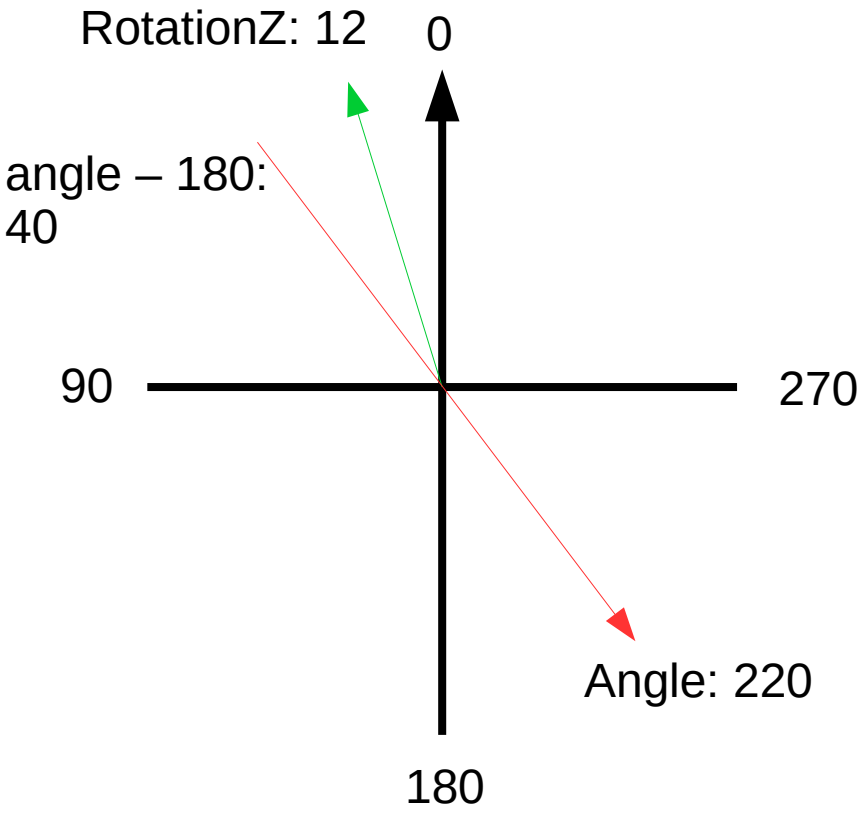
```
if (angle > 180) {
  if (RotationZ < (angle - 180)) {
    RotationZ += 360;
  }
} else {
  if (RotationZ > (angle + 180)) {
    RotationZ -= 360;
  }
}
```



Solution 2

- ...and RotationZ < (angle – 180)...

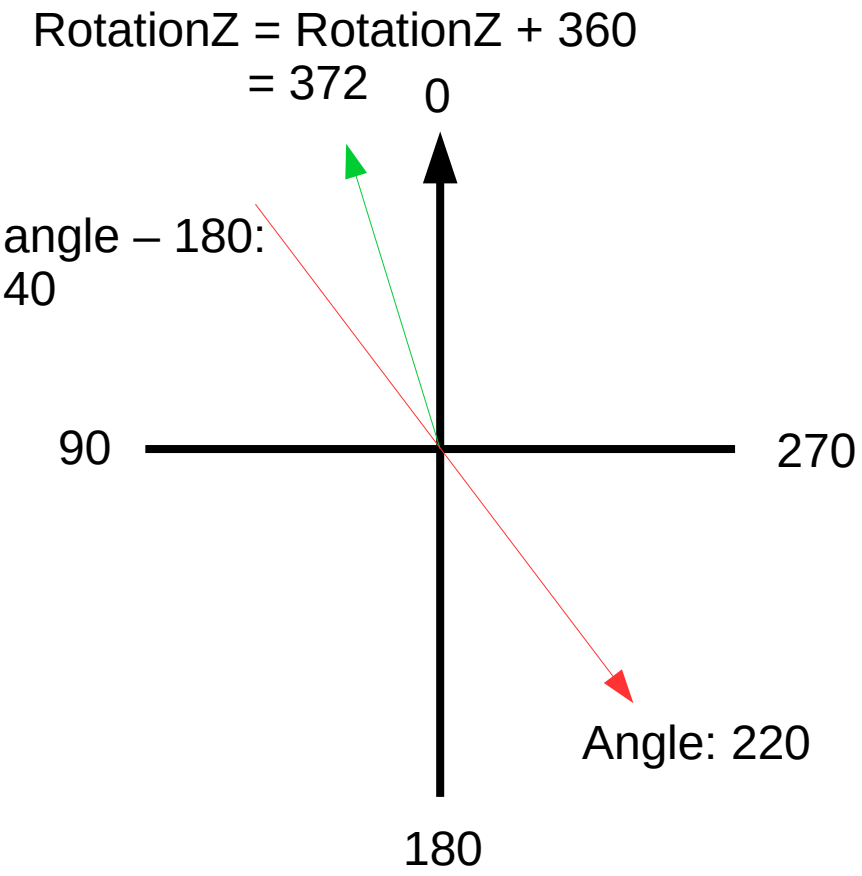
```
if (angle > 180) {
  if (RotationZ < (angle - 180)) {
    RotationZ += 360;
  }
} else {
  if (RotationZ > (angle + 180)) {
    RotationZ -= 360;
  }
}
```



Solution 2

- ...then RotationZ += 360

```
if (angle > 180) {
  if (RotationZ < (angle - 180)) {
    RotationZ += 360;
  }
} else {
  if (RotationZ > (angle + 180)) {
    RotationZ -= 360;
  }
}
```



2 States Gyro Follower

```
void gyro_follow(angle, speed)
{
  if (angle > 180) {
    if (RotationZ < (angle - 180)) {
      RotationZ += 360;
    }
  } else {
    if (RotationZ > (angle + 180)) {
      RotationZ -= 360;
    }
  }

  if (RotationZ > angle) {
    // Turn Right
    WheelLeft = 20;
    WheelRight = -20;
  } else {
    // Turn Left
    WheelLeft = -20;
    WheelRight = 20;
  }
}
```

Modify RotationZ if necessary

2 States Gyro Follower
(Same as before)

Proportional Gyro Follower

```
void gyro_follow(angle, speed)
{
  int error;

  if (angle > 180) {
    if (RotationZ < (angle - 180)) {
      RotationZ += 360;
    }
  } else {
    if (RotationZ > (angle + 180)) {
      RotationZ -= 360;
    }
  }

  error = RotationZ - angle;

  move_steering(0.1 * error, speed);
}
```

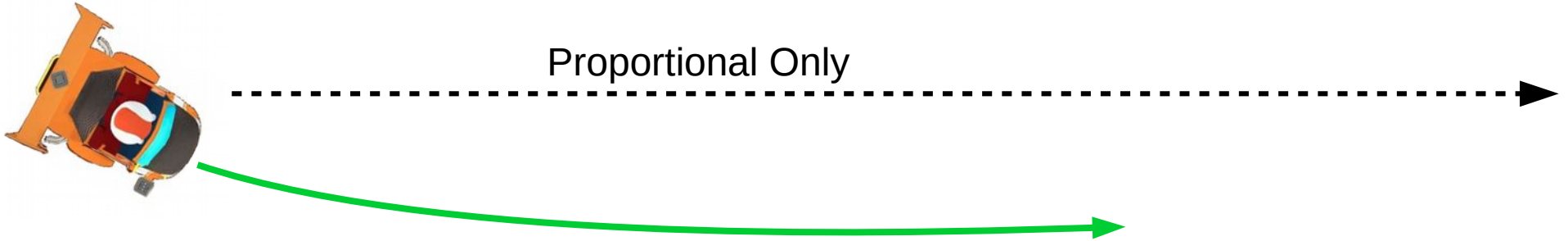
Modify RotationZ if necessary

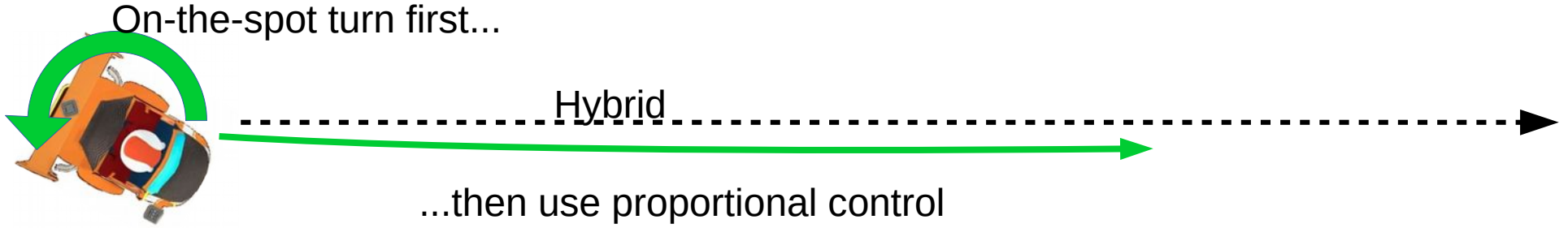
Proportional Gyro Follower
(...you'll need to tune the gain)

Gain (0.1)

Possible Enhancements

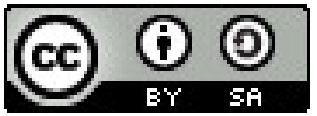
- Hybrid approach
  - Do an on-the-spot turn if the error is large
  - Use proportional control if the error is small





Copyright

- Created by A Posteriori LLP
- Visit http://aposteriori.com.sg/ for more tips and tutorials
-



A POSTERIORI

Play · Experience · Learn