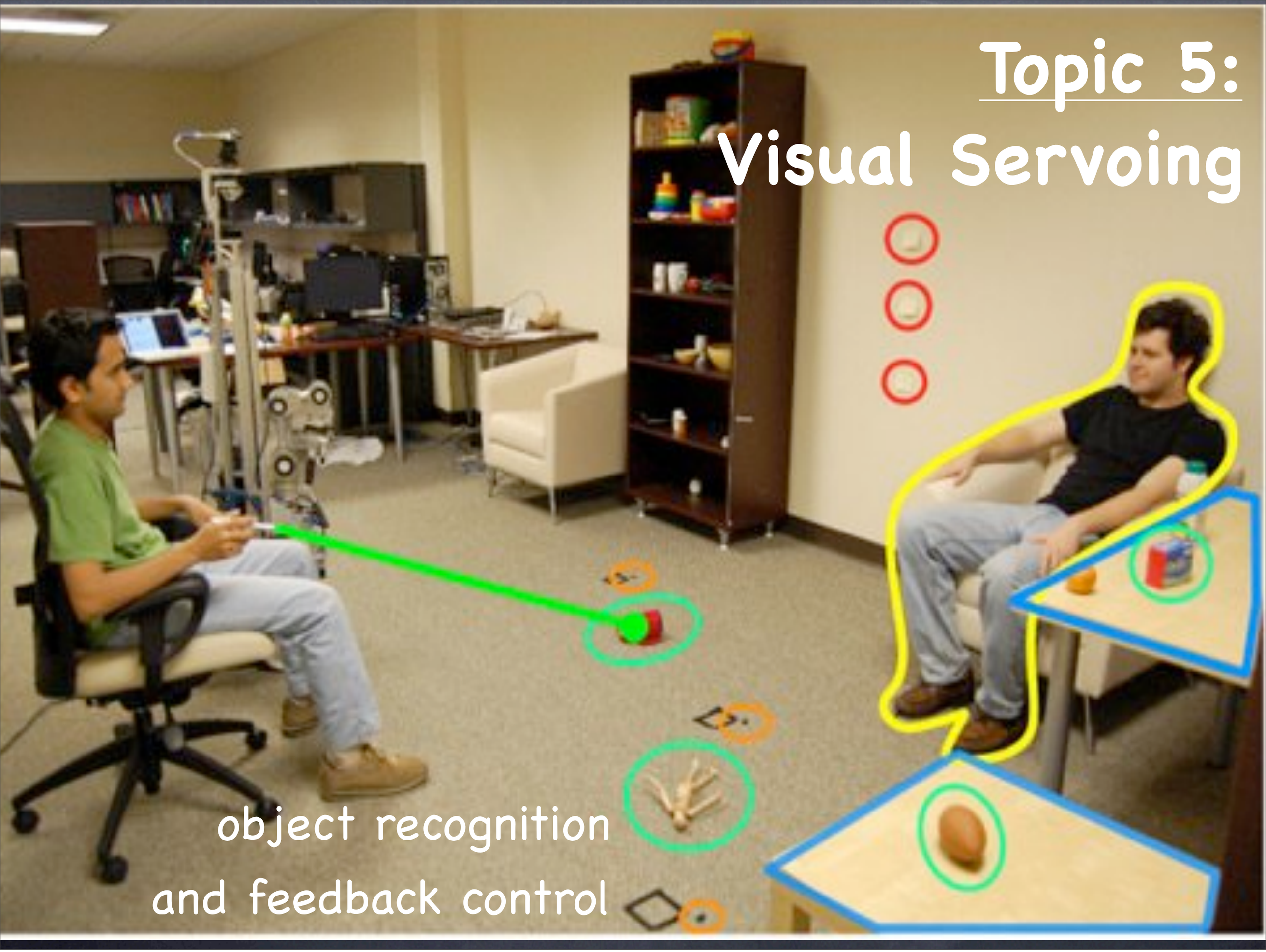
Topic 5:

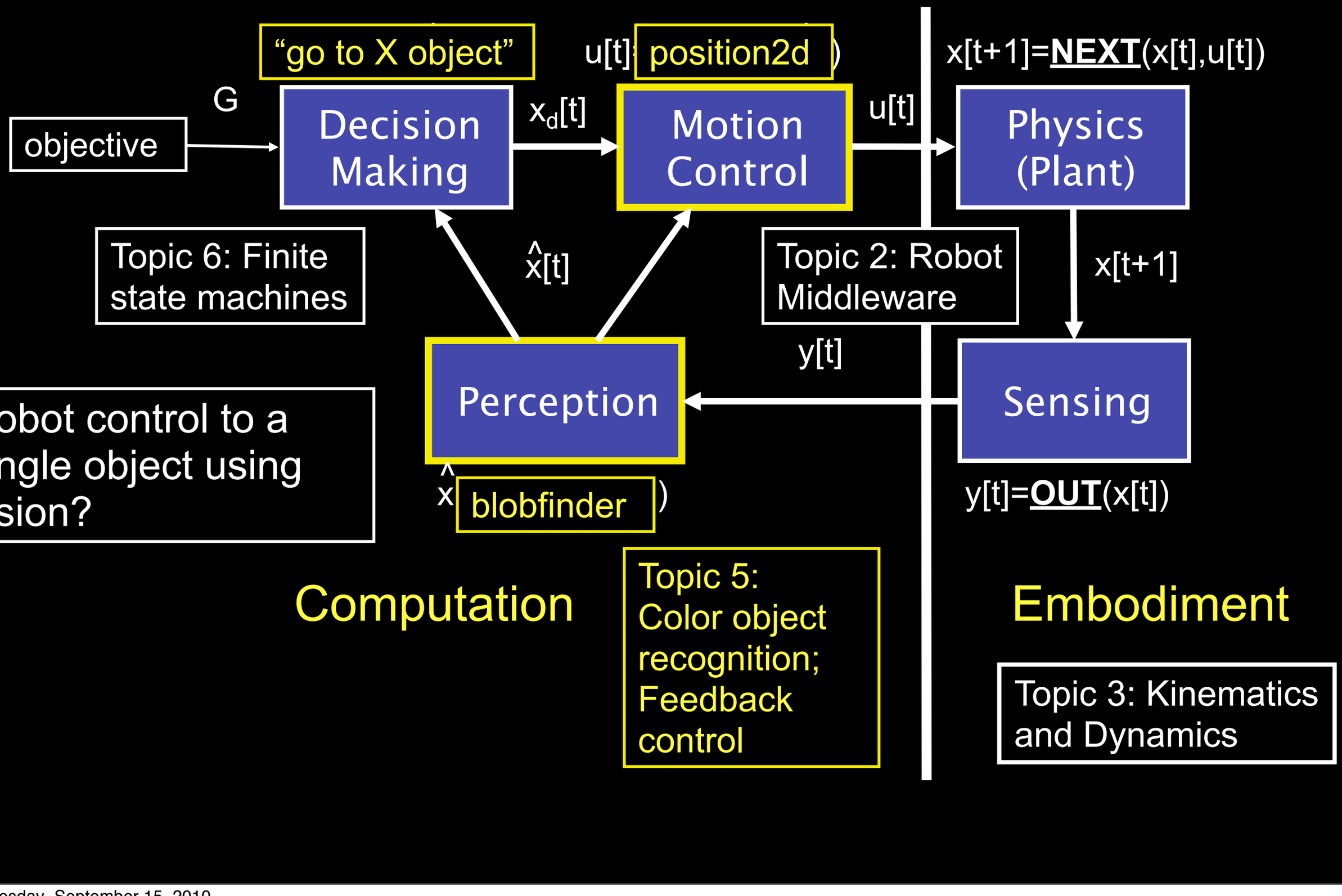
Visual Servoing



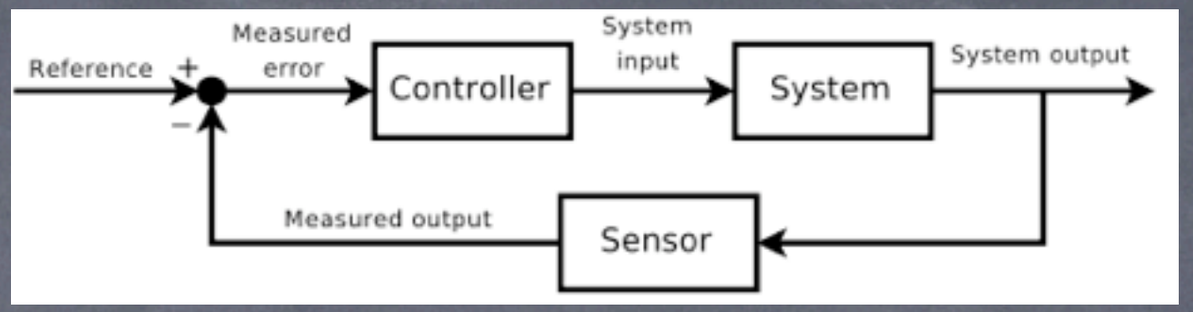
robot control loop

- someone please sketch on the board

The Robot Control Loop

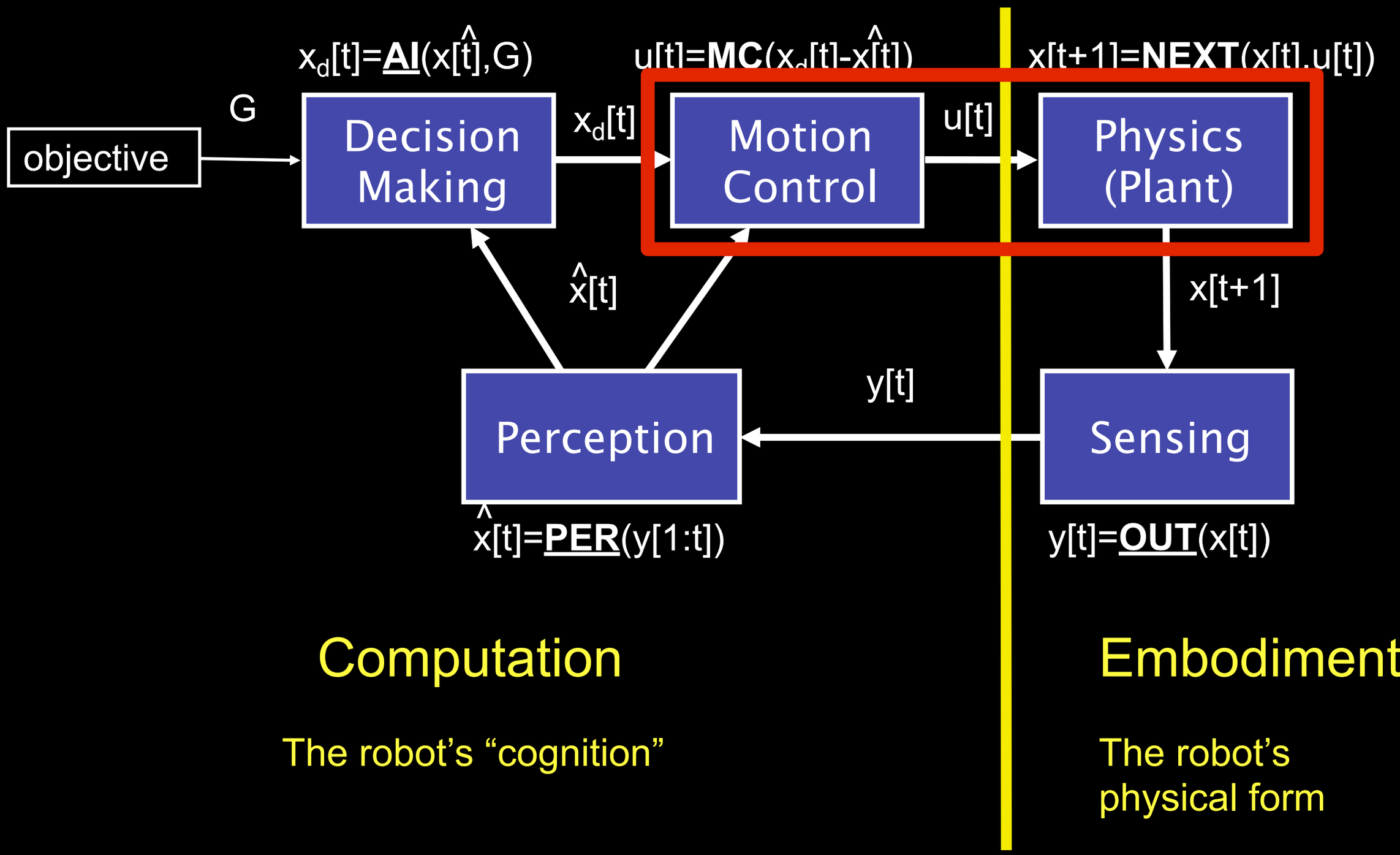


Control theory

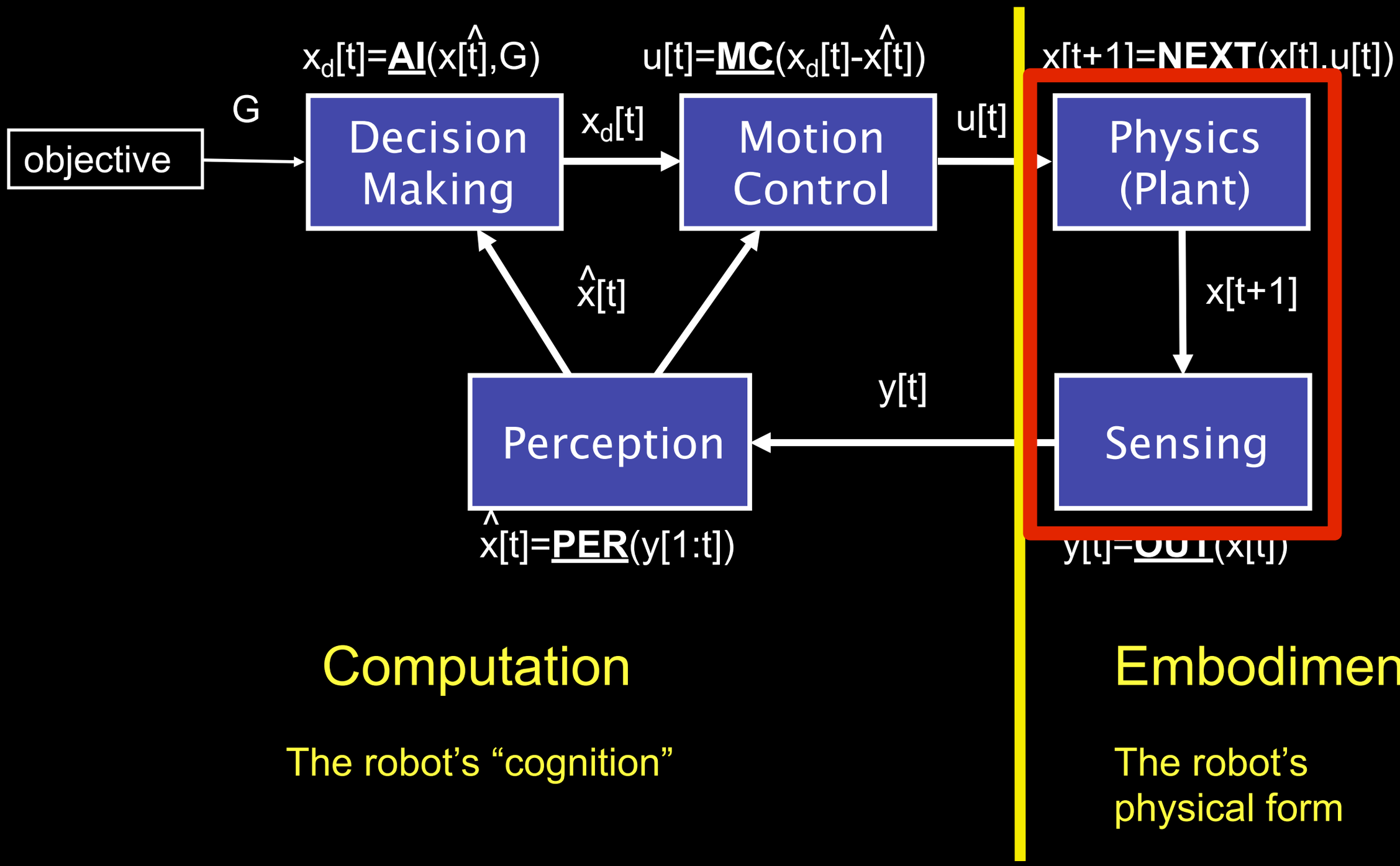


- Control theory: study behavior of dynamical systems
- Forms of control systems:
  - Open-loop control: without system feedback
  - Closed-loop control: adjusts to sensor observations
    - Feedback control: minimize sensed error
  - Open and closed loop
    - Feedback-feedforward: predict and correct
- Important properties: stability, robustness, controllability

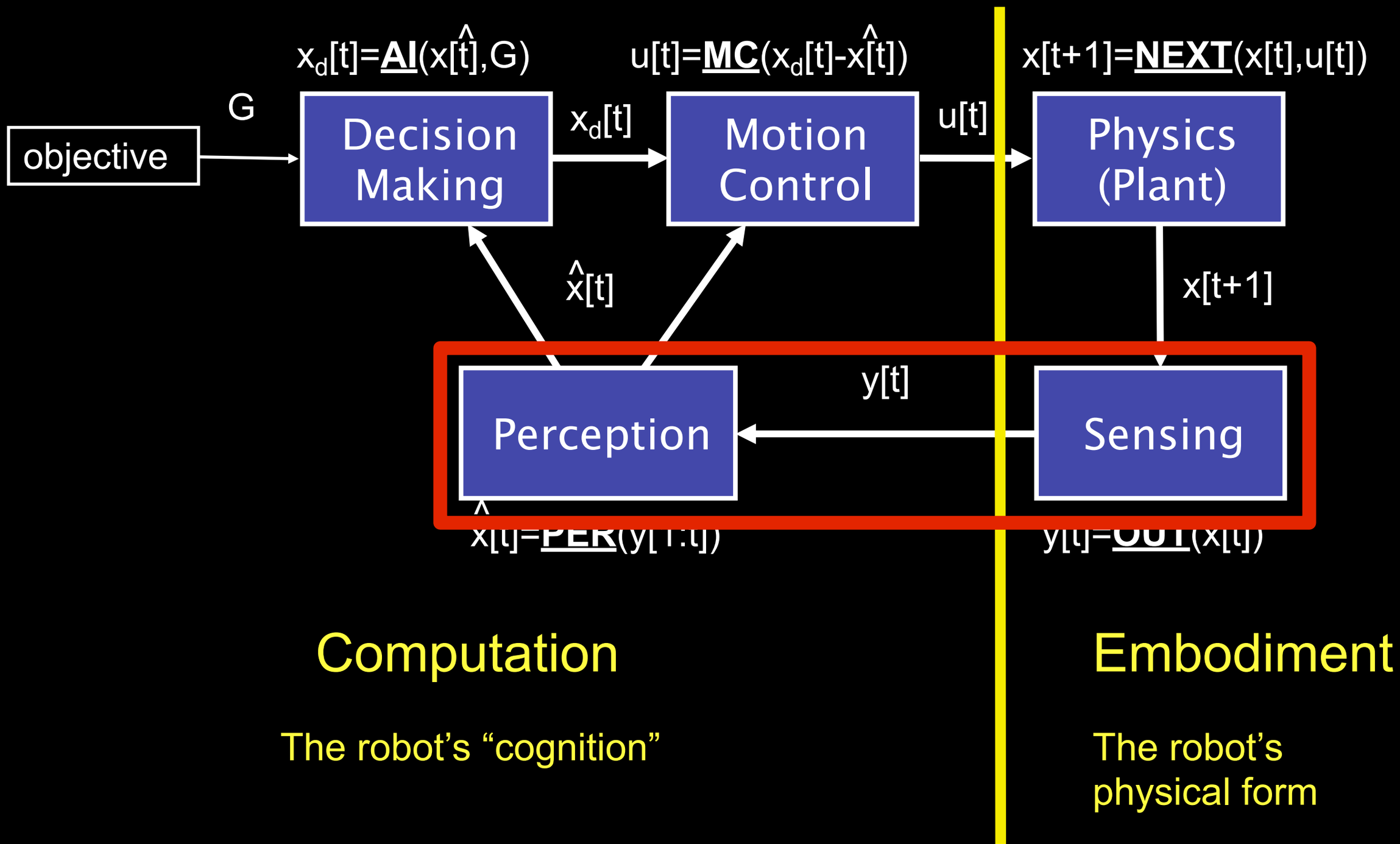
Open-loop control: traditional



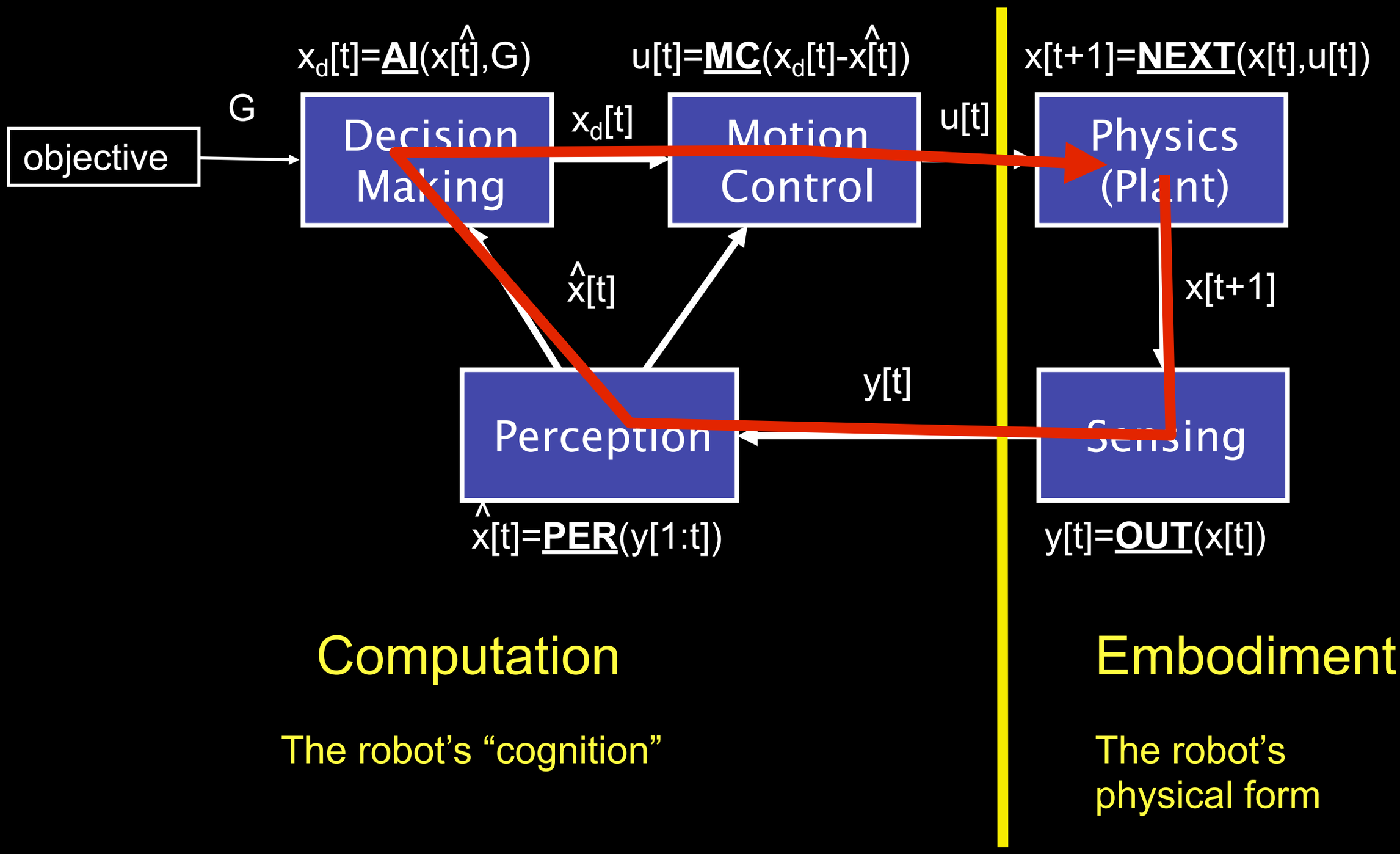
Open-loop: graphics



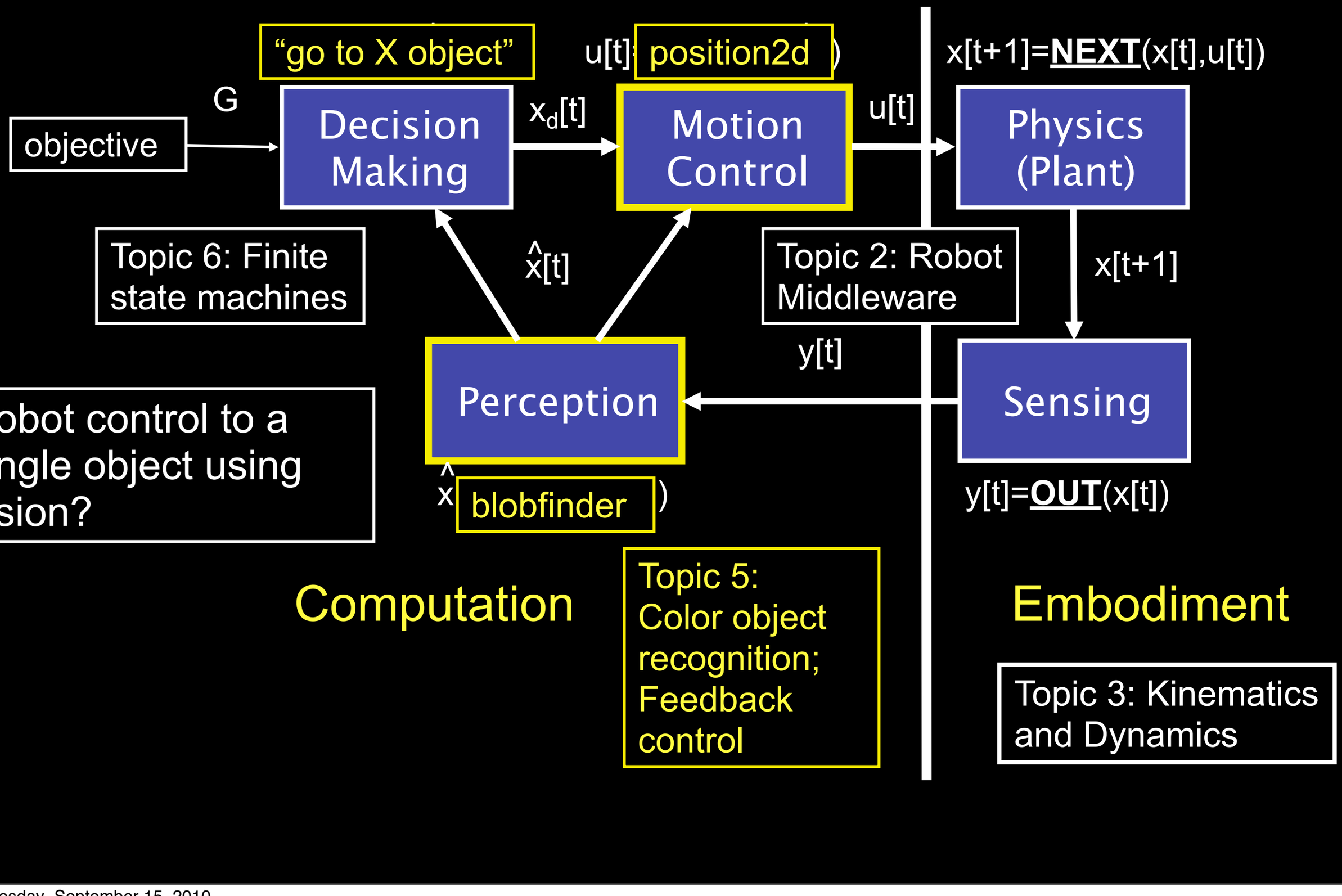
Open-loop: vision



Closed loop: robotics



The Robot Control Loop



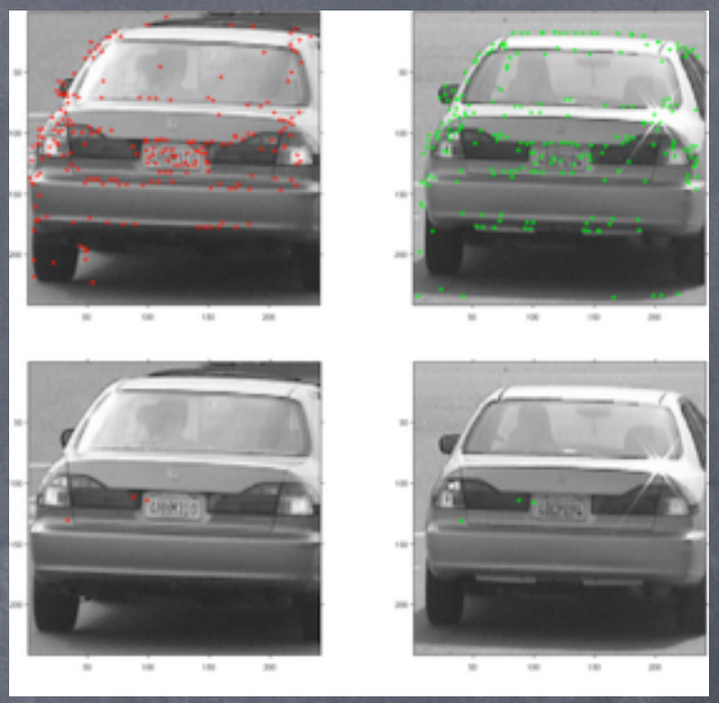
Object recognition

- Training: build database of object appearance
  - Collect representative images of objects
  - Store and extract features from these images
  - Features describe object appearance
- Given a new image: search over image for object
  - Find database entries with features matching those in the image

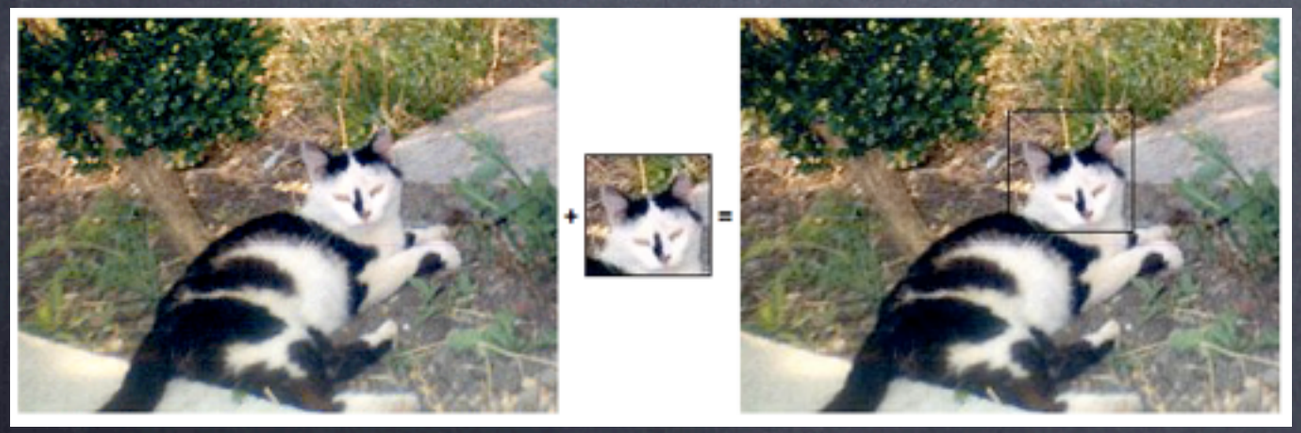
Appearance features

HoG: histograms of oriented gradients

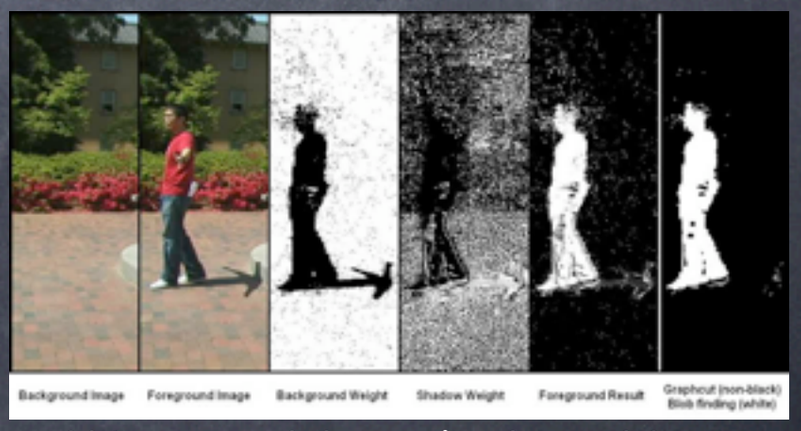
edge detection



SIFT: corners, hierarchy



template matching

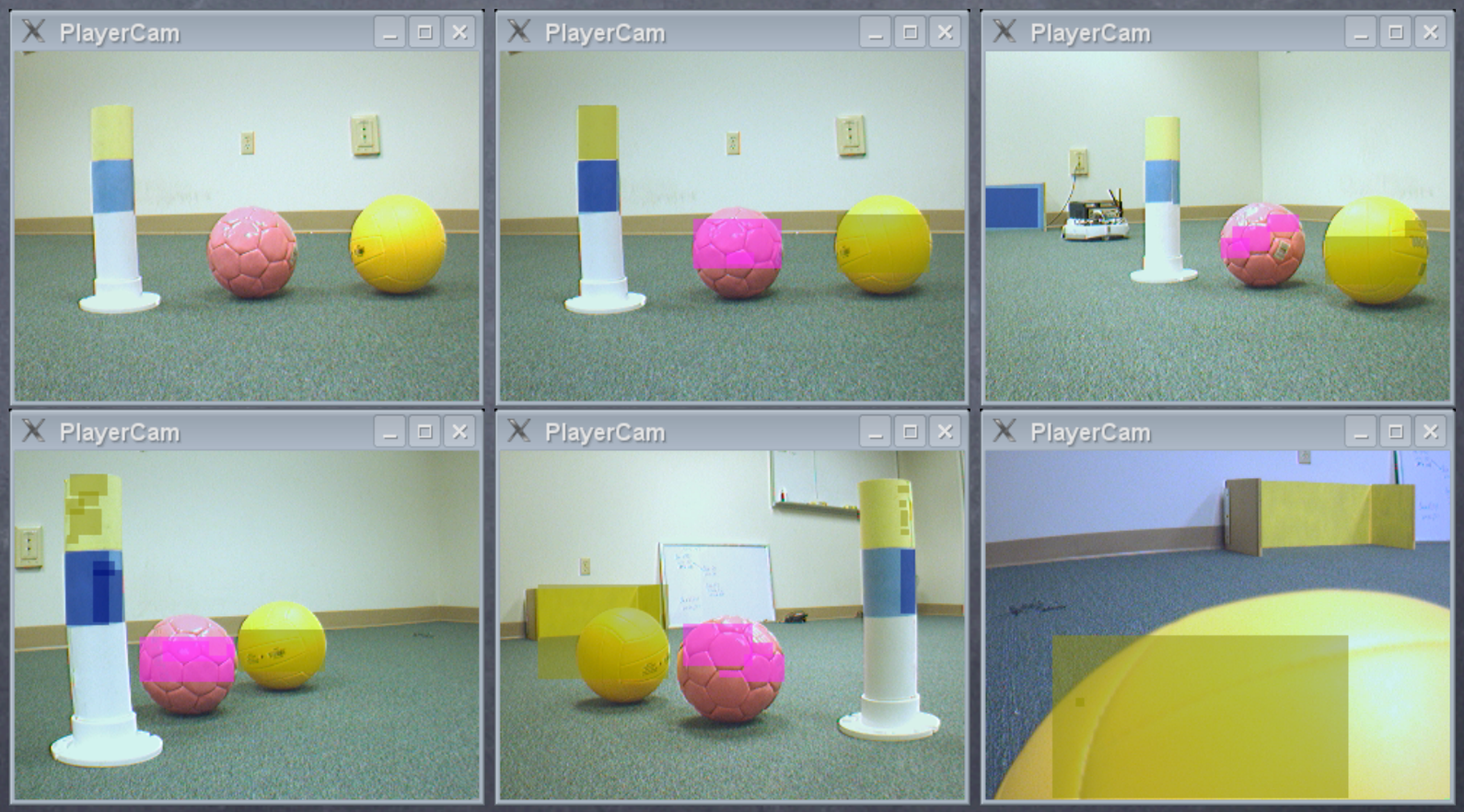


silhouette detection:
"green screen"

Color blobfinder

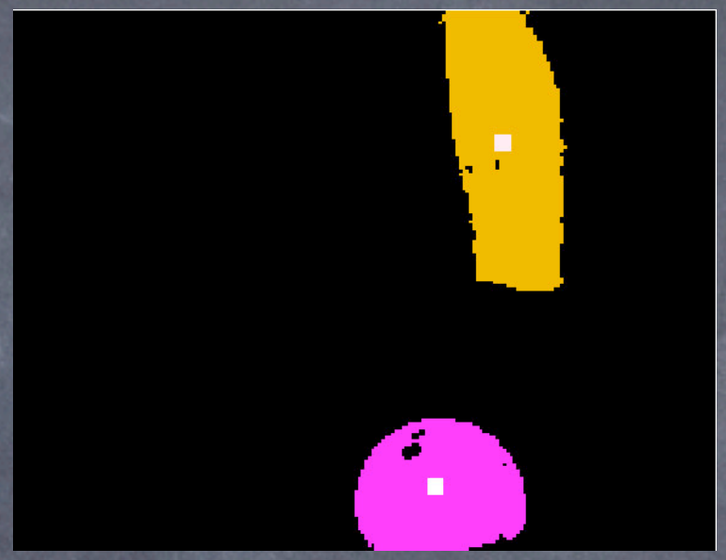
Camera node:
gscam (Gstreamer)

Blobfinding node:
cmvision (cmvision)



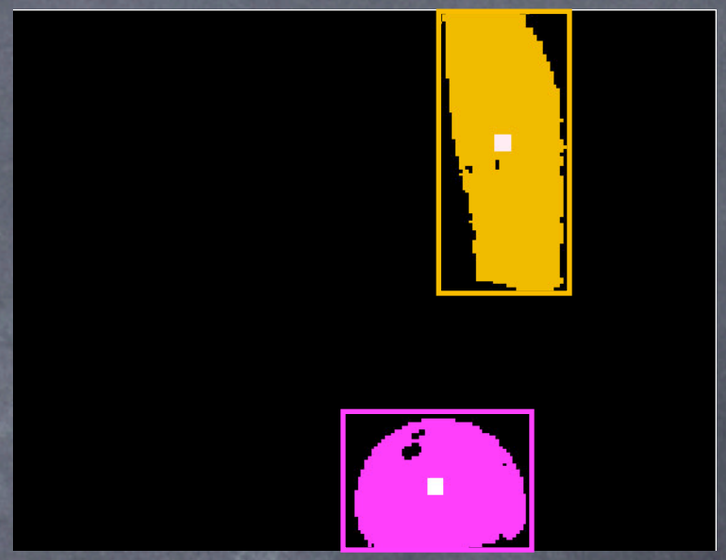
Color blobfinding

Input image



Color threshold

How?

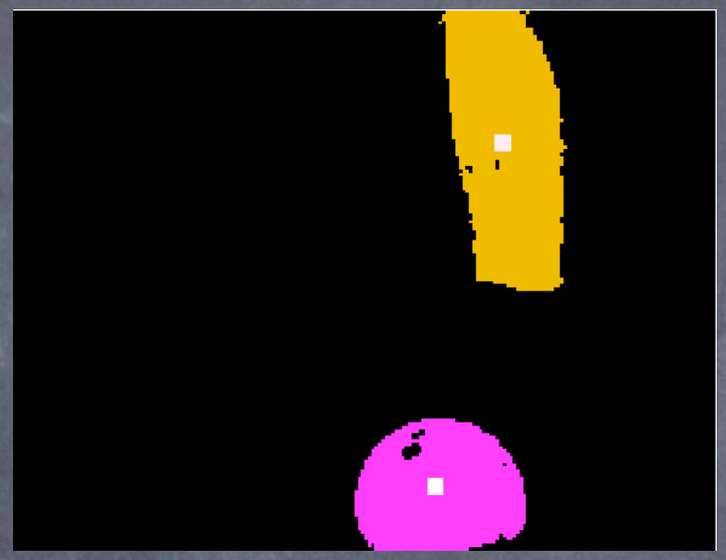


Group regions

How?

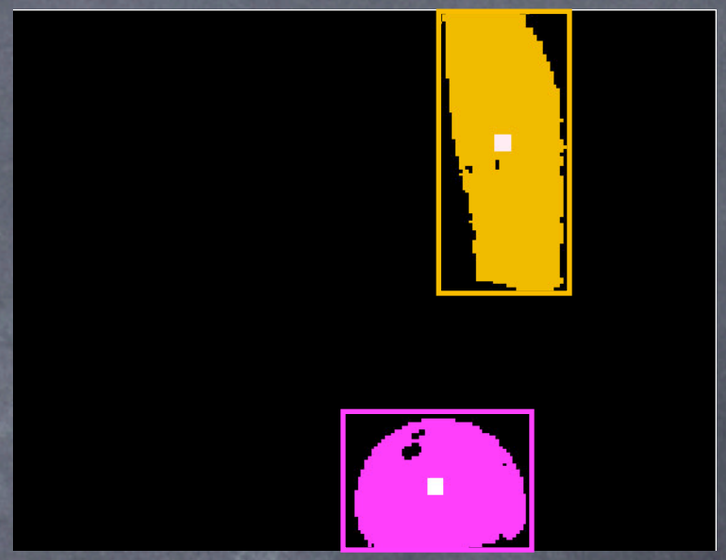
Color blobfinding

Input image



Color threshold

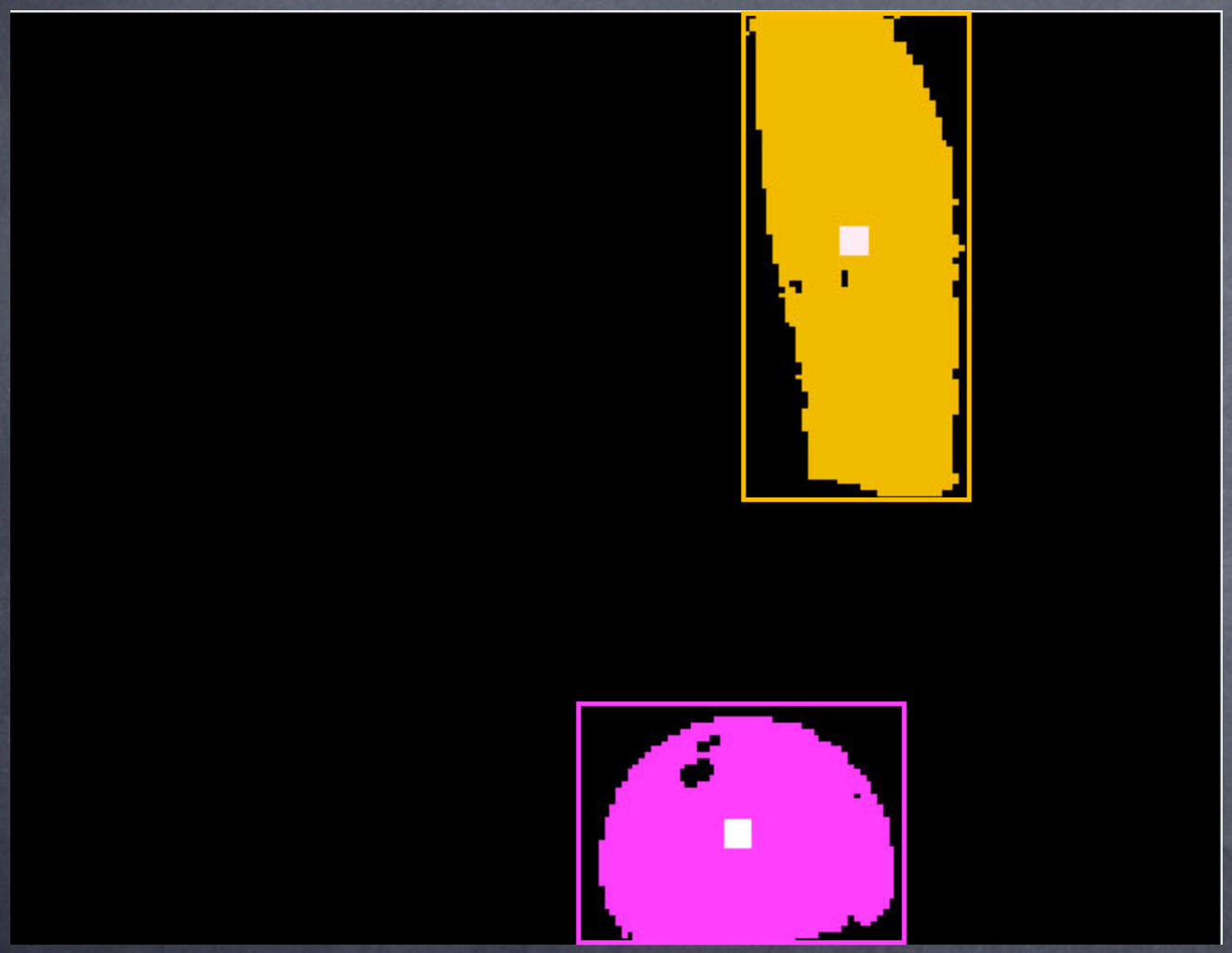
Color calibration



Group regions

Connected Components

Color blobfinding



Blobs message:

Header header
  uint32 seq
  time stamp
  string frame_id
uint32 image_width
uint32 image_height
uint32 blob_count
Blob[] blobs
  uint32 red
  uint32 green
  uint32 blue
  uint32 area
  uint32 x
  uint32 y
  uint32 left
  uint32 right
  uint32 top
  uint32 bottom

Color calibration

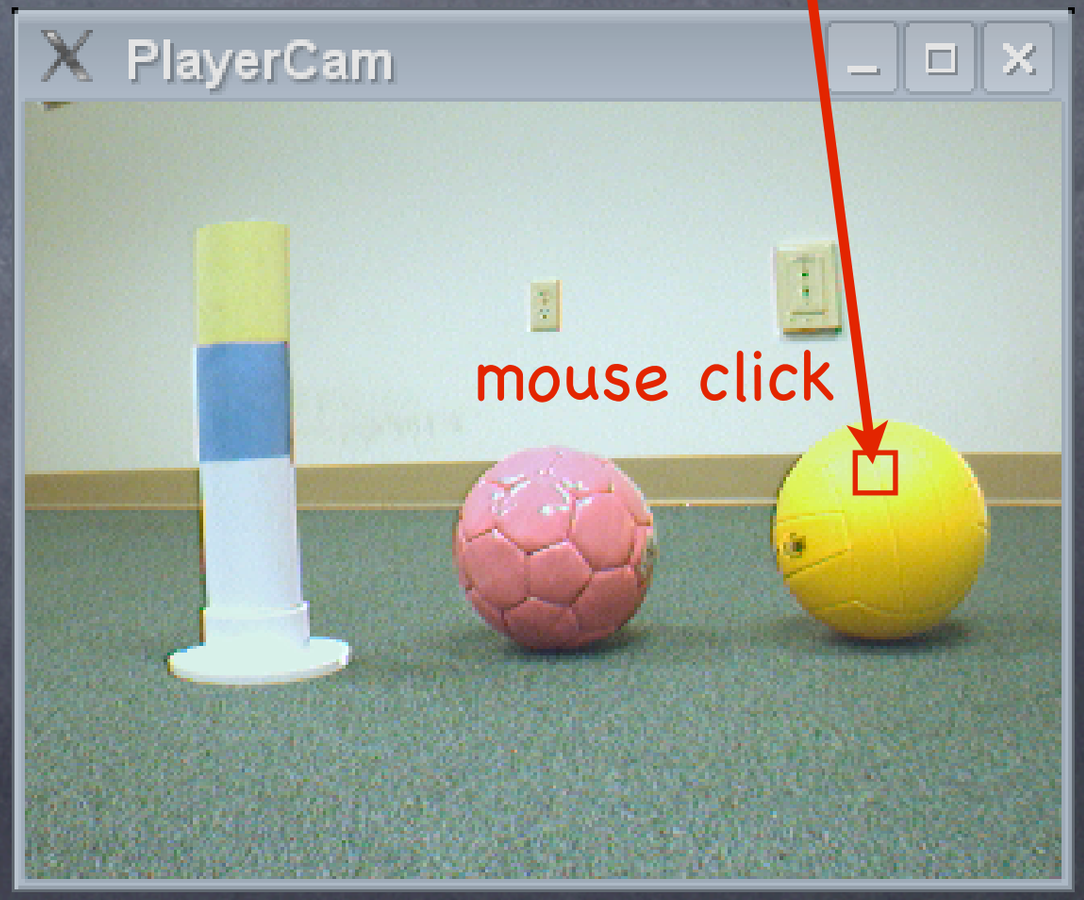
```
[Colors]
(255,  0,  0) 0.000000 10 Red
(  0,255,  0) 0.000000 10 Green
(  0,  0,255) 0.000000 10 Blue

[Thresholds]
( 25:164, 80:120,150:240)
( 20:220, 50:120, 40:115)
( 15:190,145:255, 40:120)
```

blobcolors.txt example

(thresholds in YUV color space)

playercam_yuv output

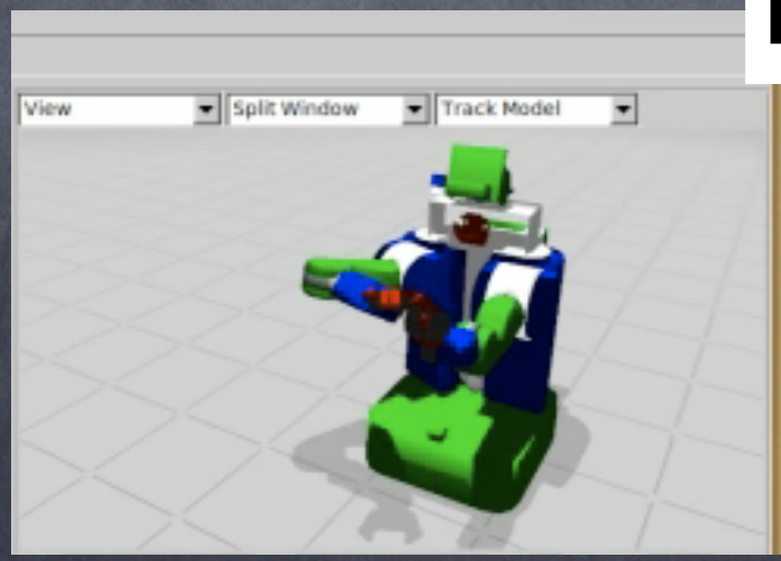
```
[51, 145] = RGB [0 140 0] : : YUV [82 81 69]
```



AR tag recognition

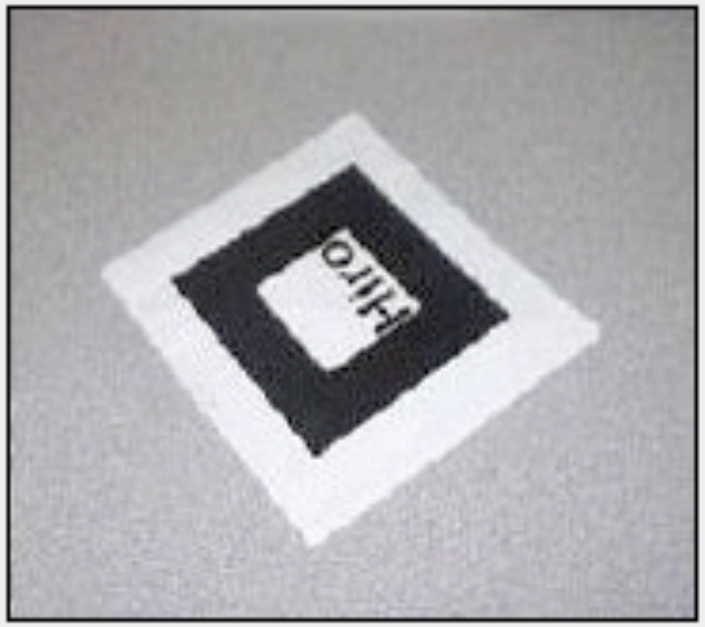
Camera node:
gscam (Gstreamer)

AR tag node:
ar_recog (ARToolkit)

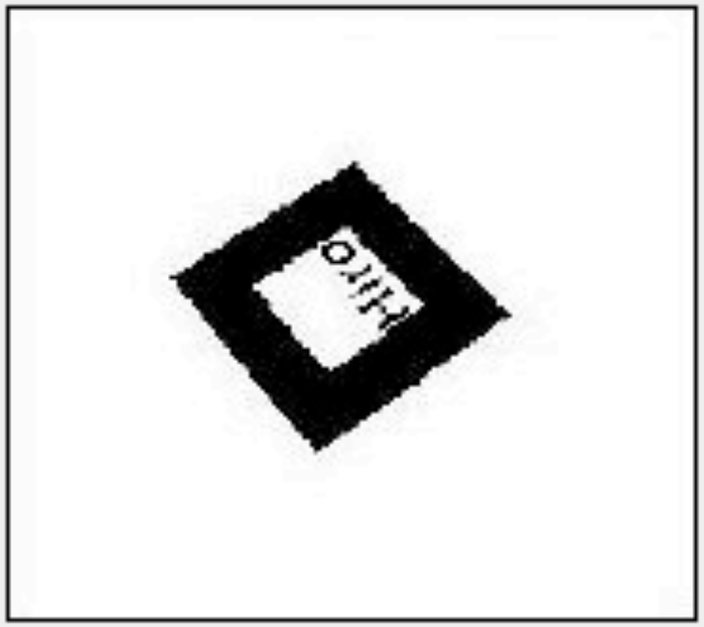


ar_recog PR2 teleop: http://www.flickr.com/photos/brownrobotics/4584683111/

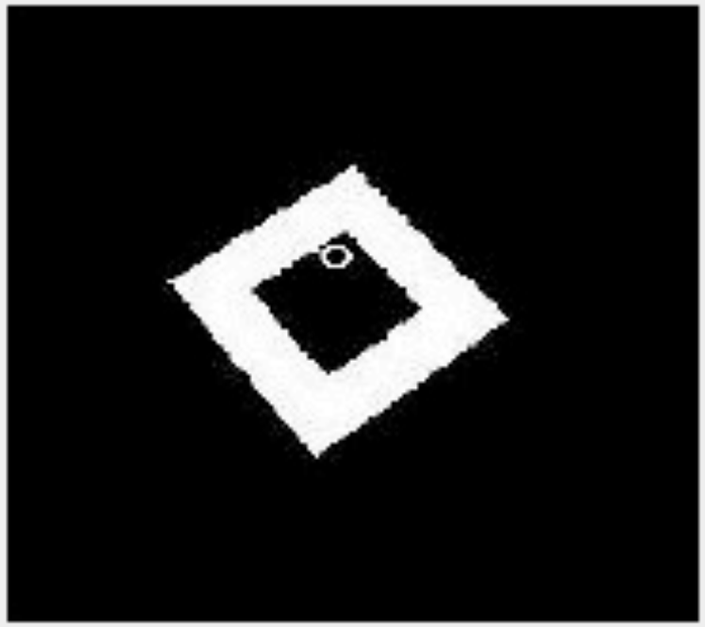
AR tag recognition



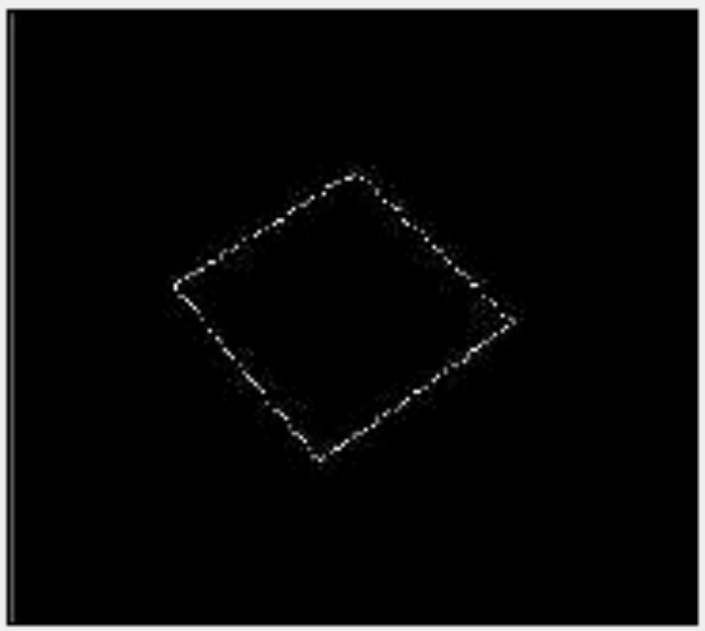
a. Original image



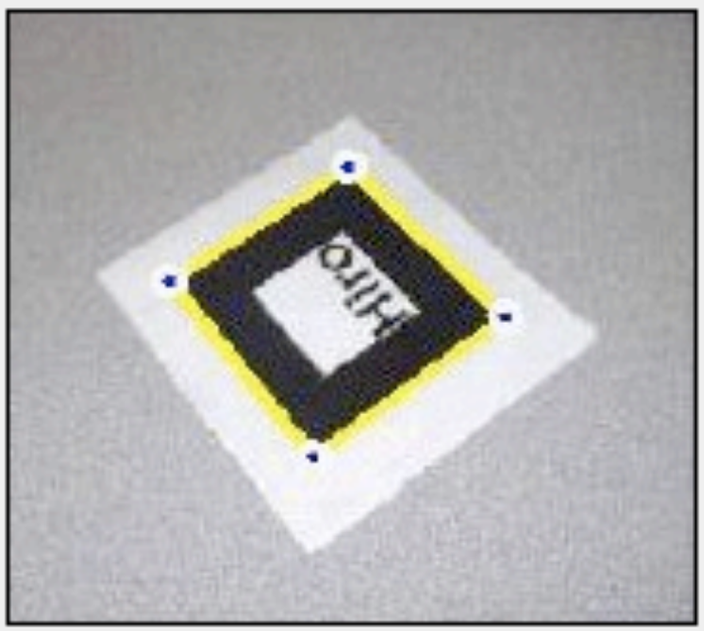
b. Thresholded image



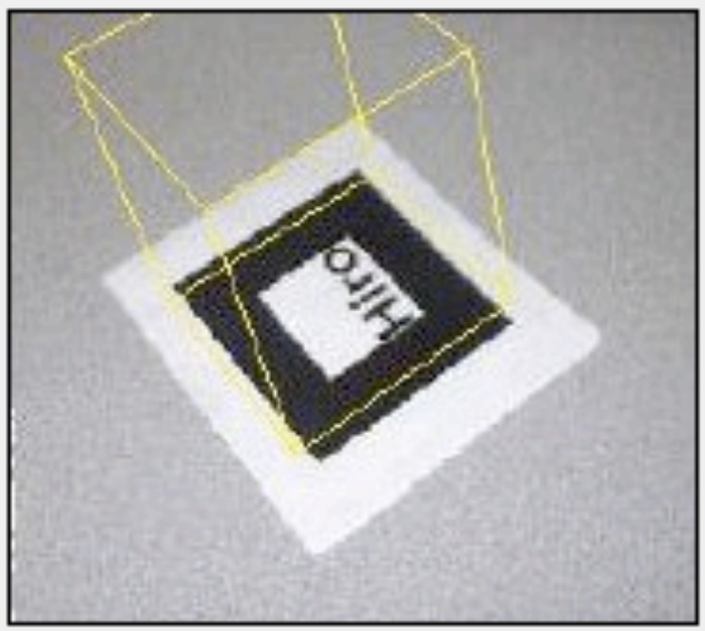
c. Connected components



d. Contours

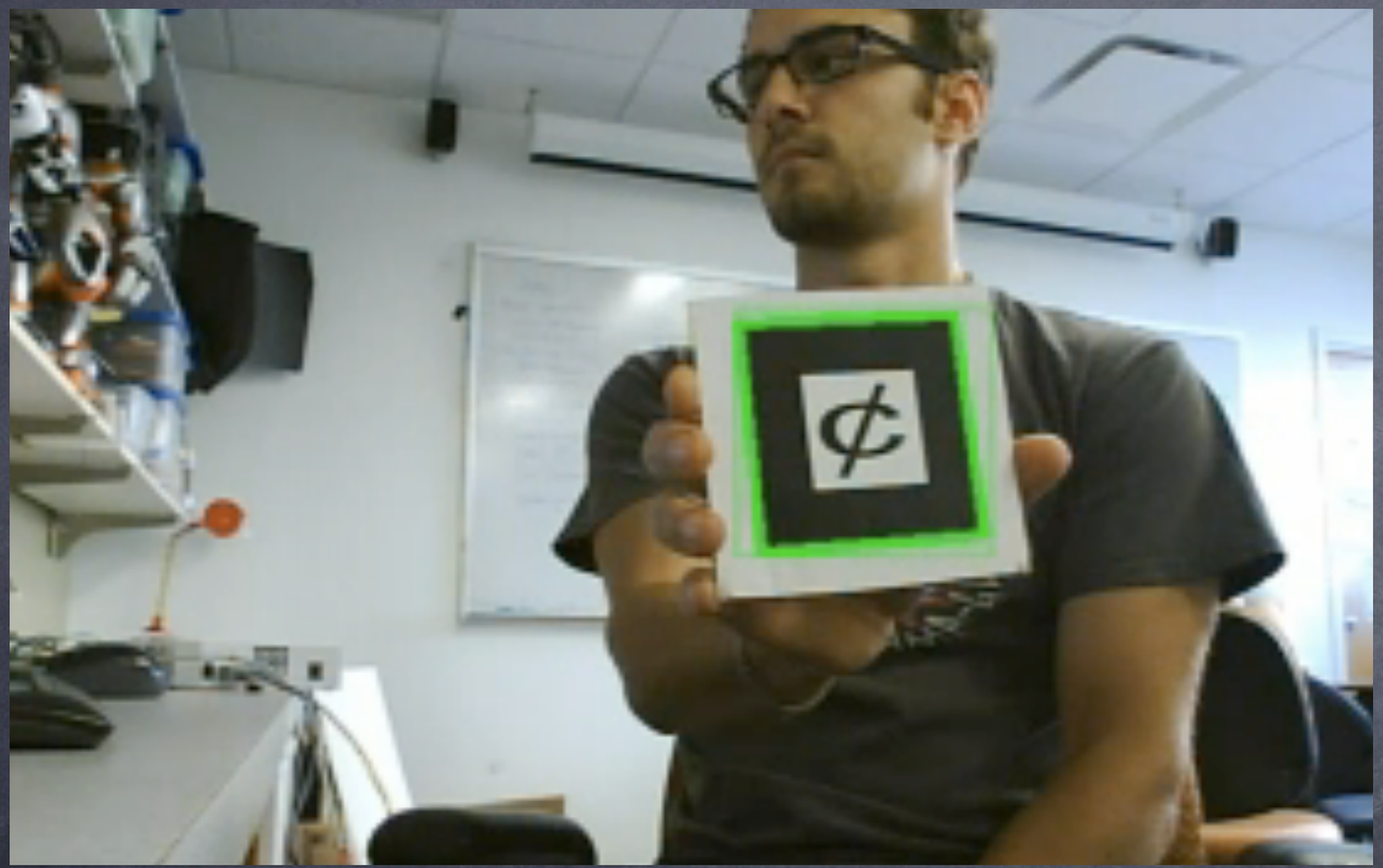


e. Extracted marker edges and corners



f. Fitted square

AR tag recognition



Tags message:

Header header
uint32 image_width
uint32 image_height
float64 angle_of_view
uint32 tag_count
Tag[] tags
  uint32 id
  float64 cf
  uint32 x
  uint32 y
  uint32 diameter
  uint32 distance
  float64 xRot
  float64 yRot
  float64 zRot
  float64[8] cwCorners

For this class, we will treat tags like color blobs

cmd_vel

cmd_vel node:
irobot_create_2_1 (Open Interface)

velocity_z

velocity_x

velocity_y

Twist message:

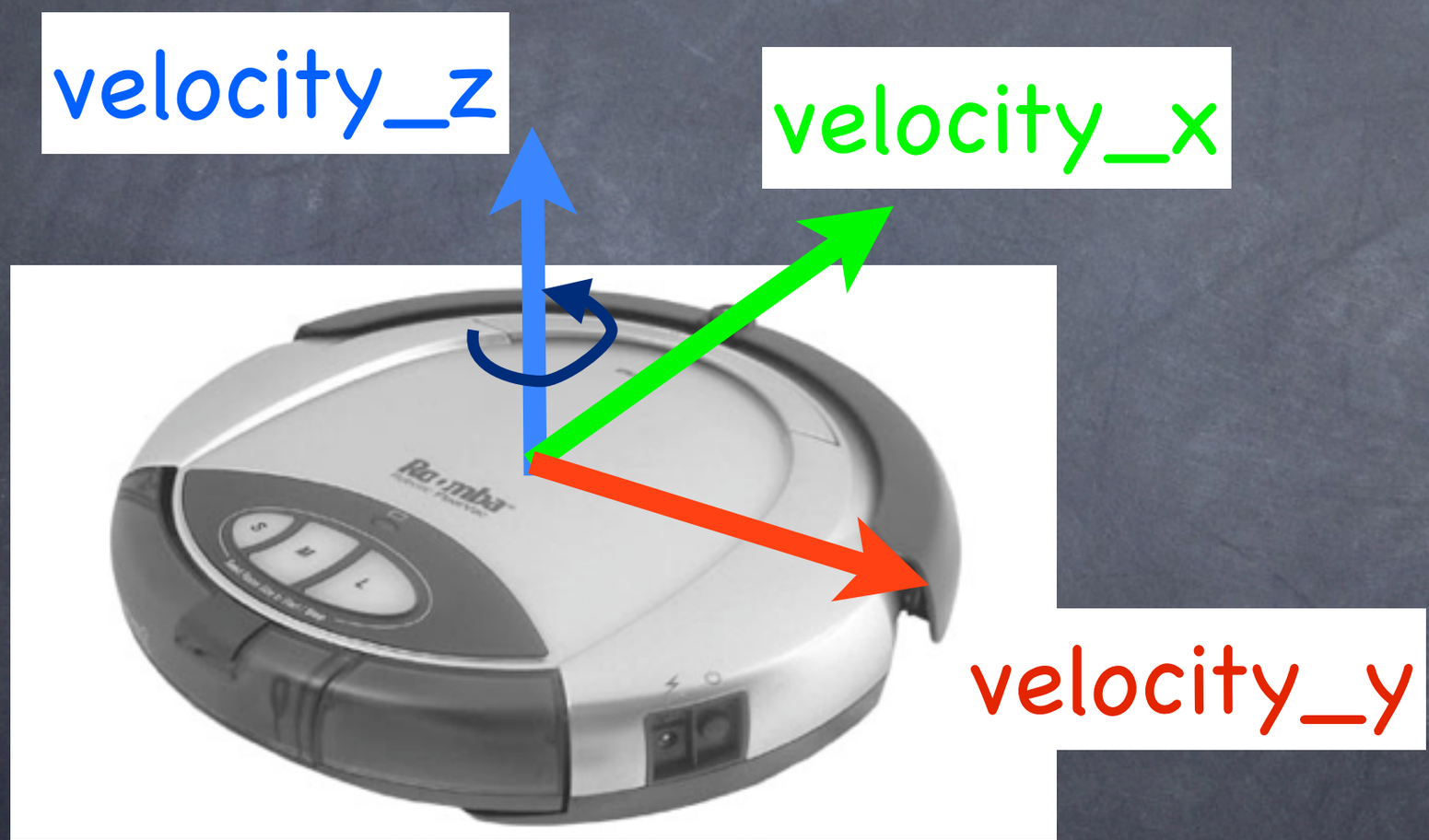
Vector3 linear
- float64 x
- float64 y
- float64 z

Vector3 angular
- float64 x
- float64 y
- float64 z

can we use all
of these variables?

cmd_vel

cmd_vel node:
irobot_create_2_1 (Open Interface)



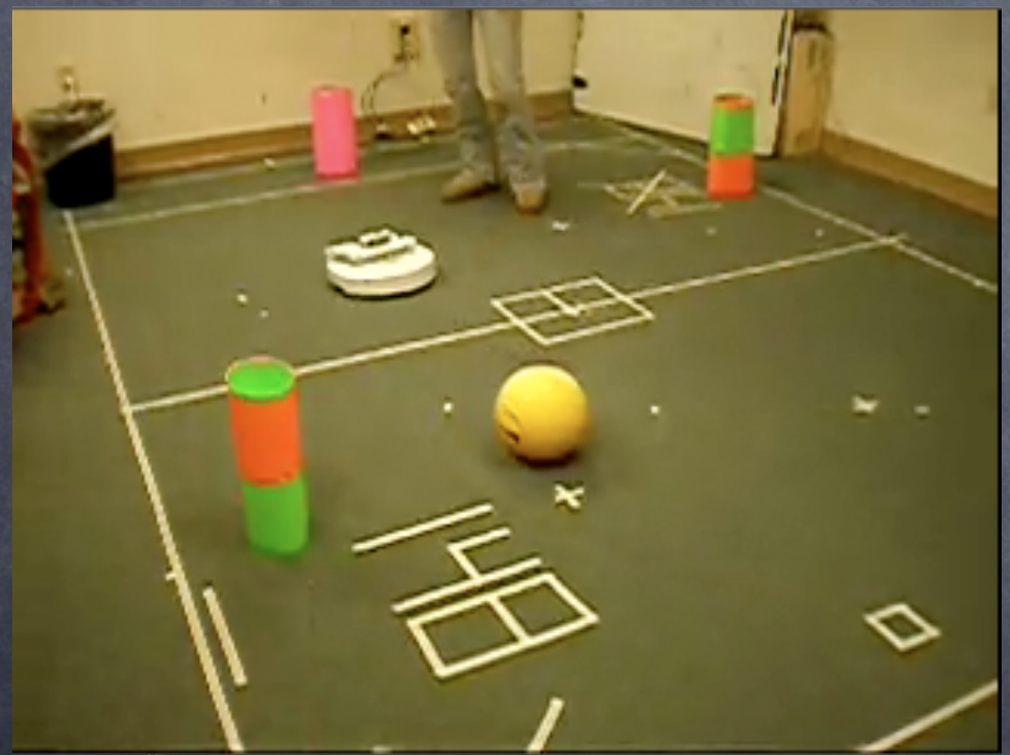
Twist message:

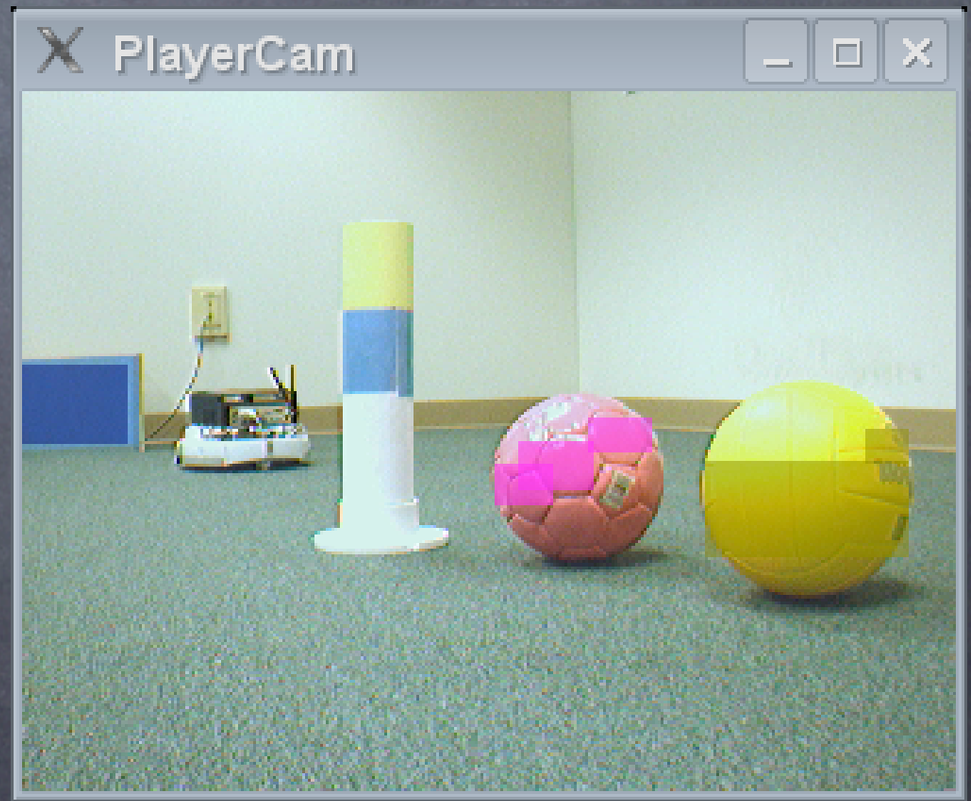
Vector3 linear
- float64 x drive: (vx)
- float64 y
- float64 z

Vector3 angular
- float64 x
- float64 y
- float64 z turning: (va)

Move to the ball?

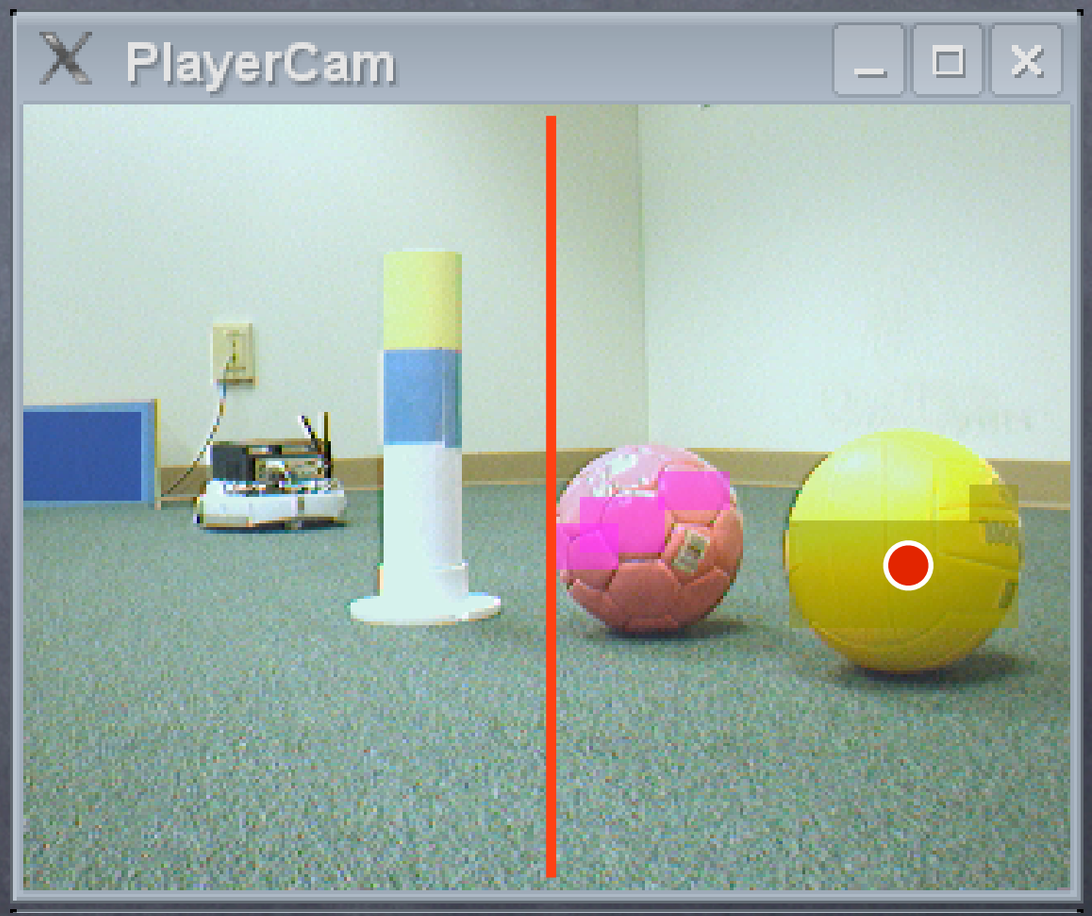
- how can our robot move to the ball?





One approach

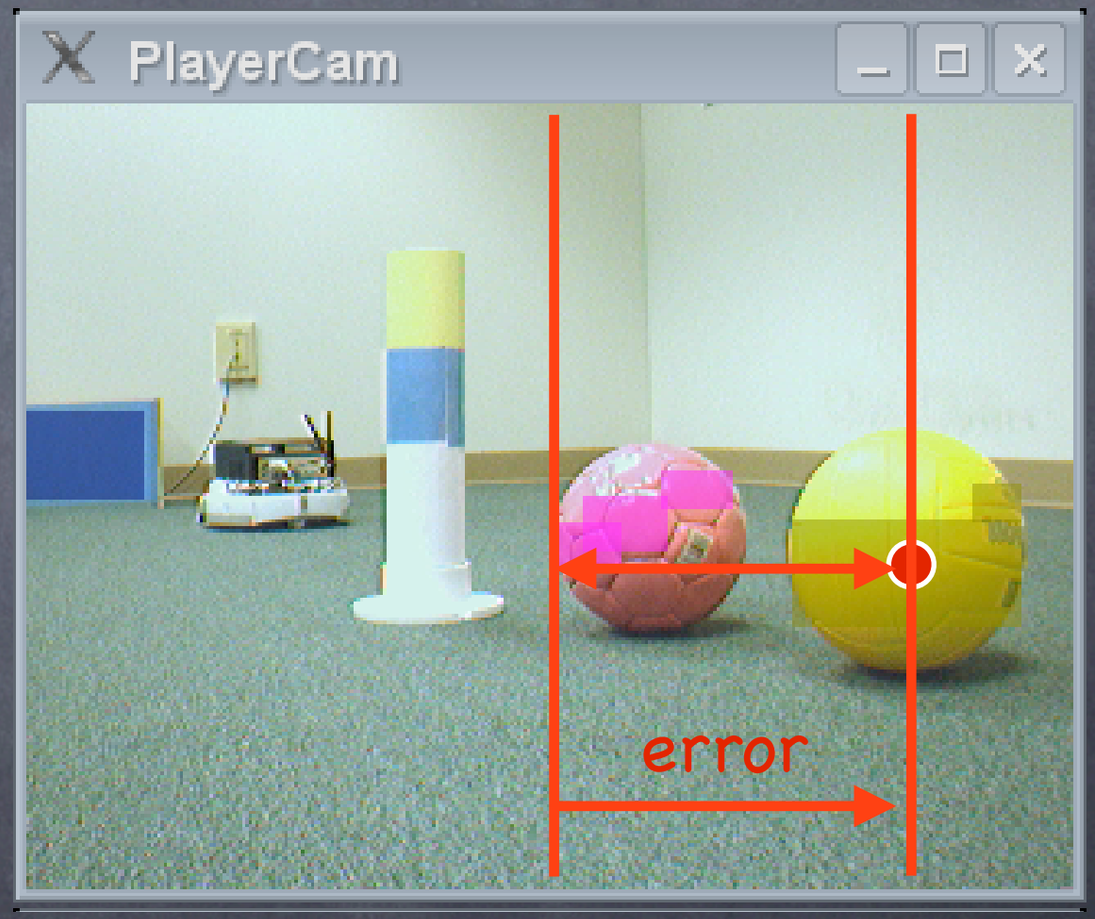
- Center ball in image
- Once centered, drive forward
- Can we state in terms of vx and va?



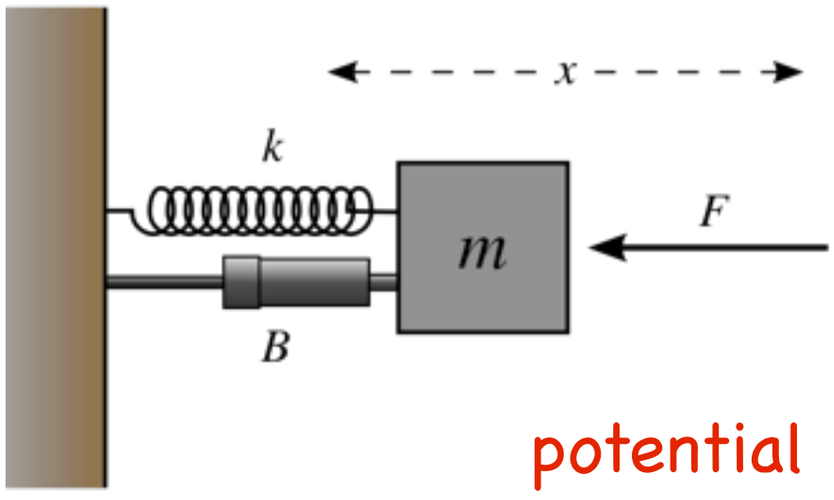
P Servo

va = -k * error

simple form of feedback control



spring mass model

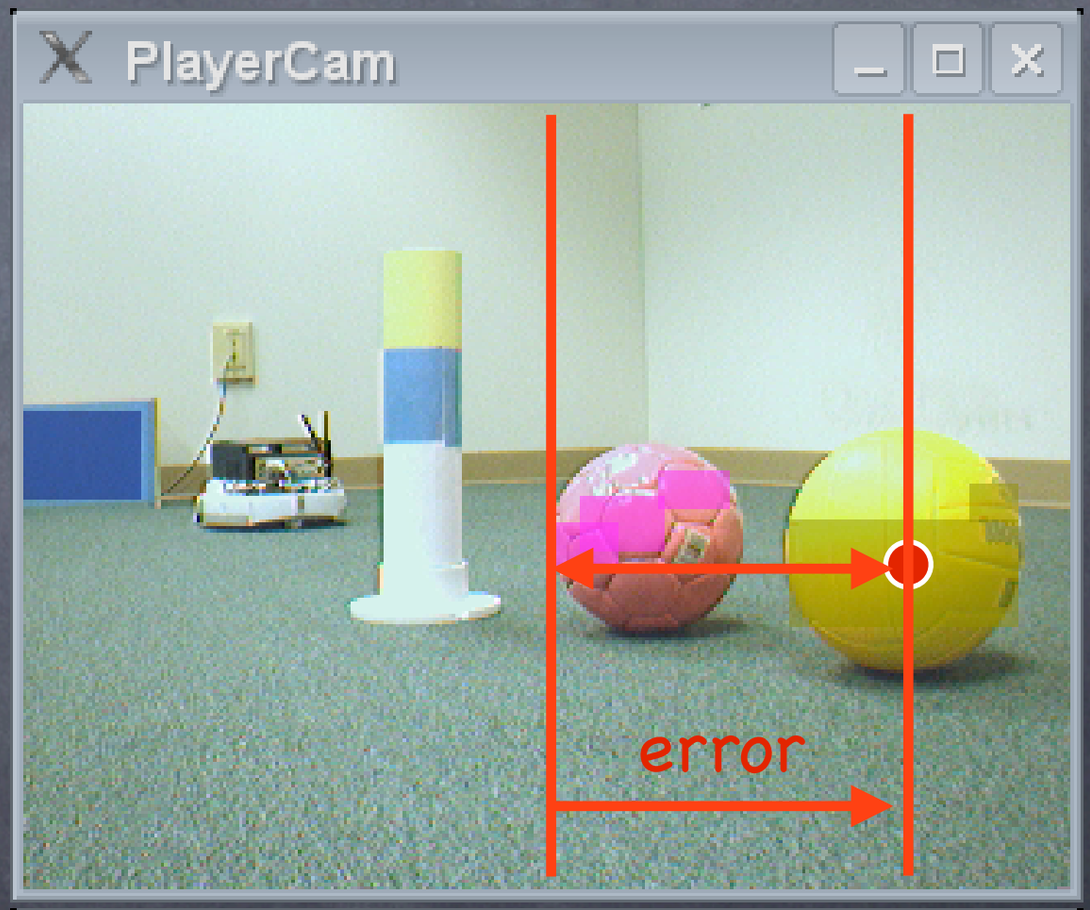


potential problems?

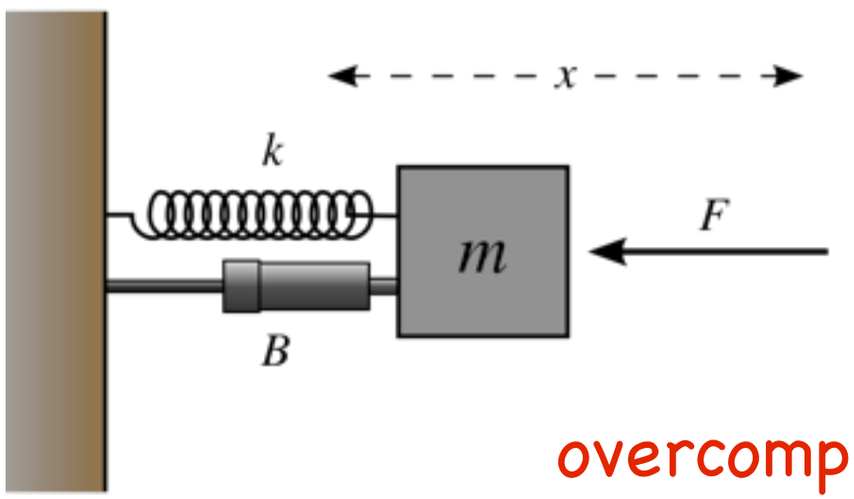
P Servo

va = -k * error

gain selection
ball out-of-view
false positives



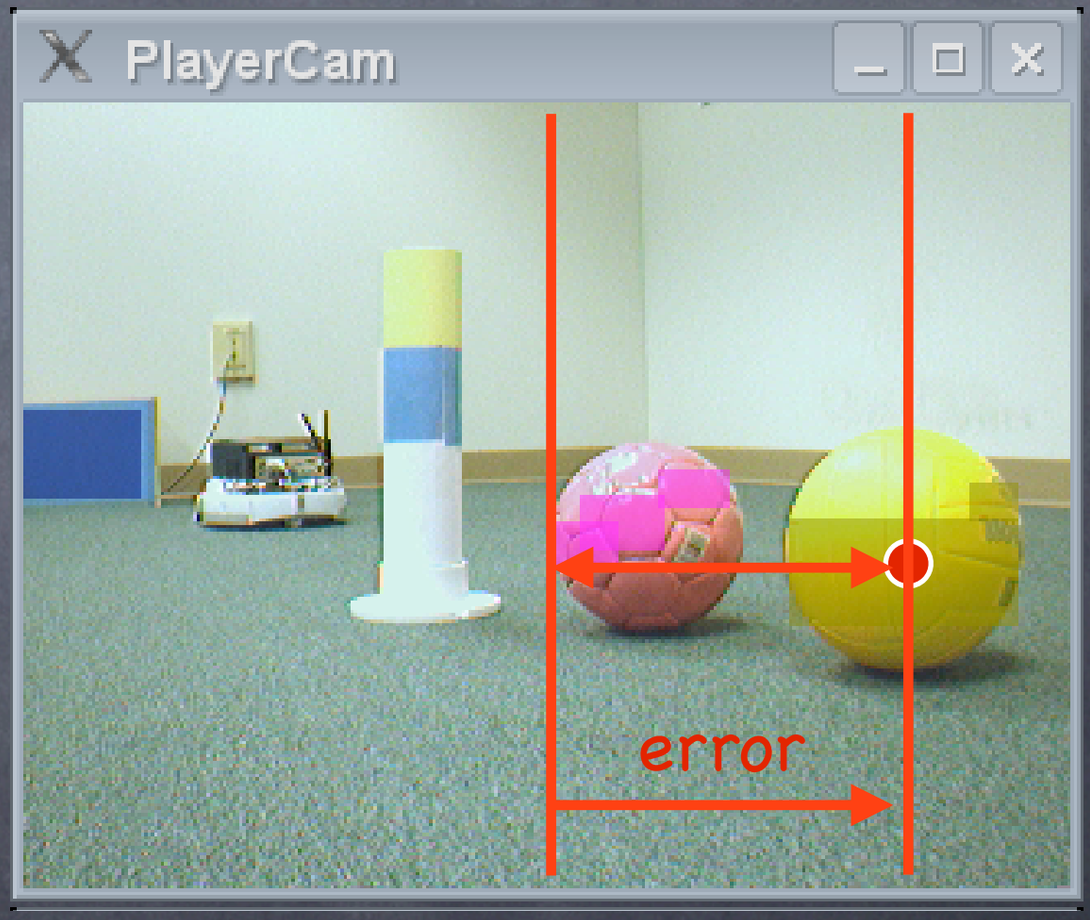
spring mass model



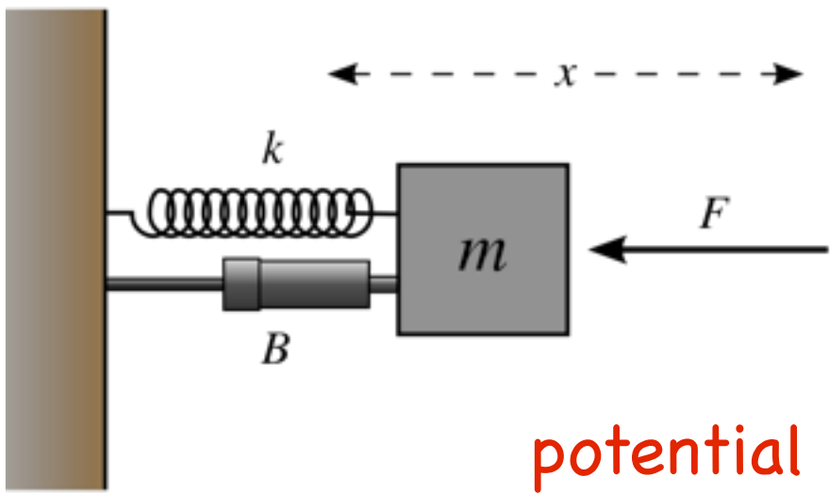
overcompensation

PD Servo

$$va = -k * error + -k_d * error\_derivative$$



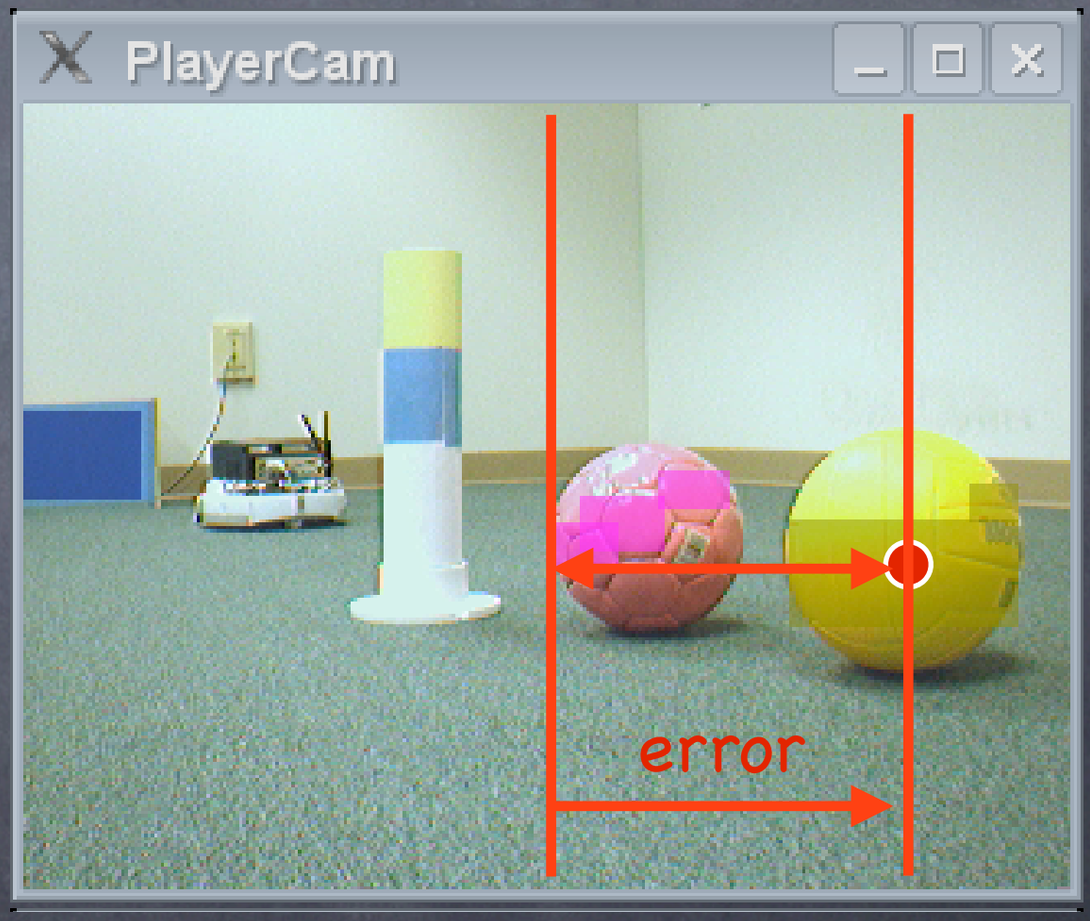
spring-damper model



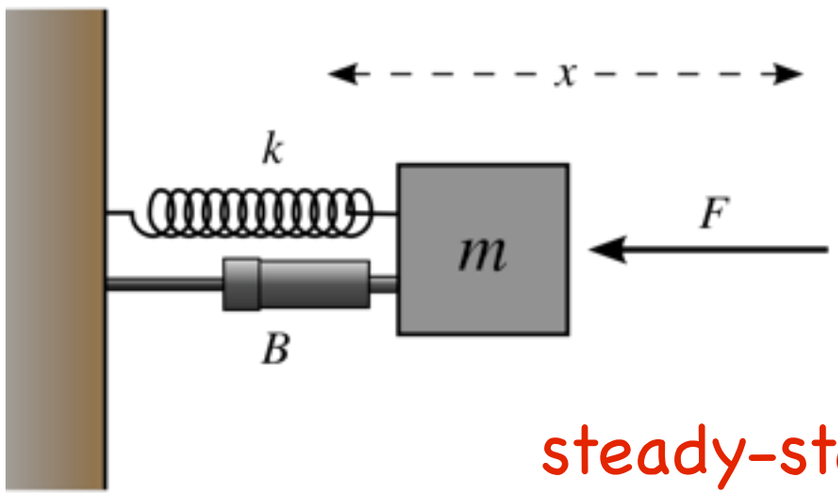
potential problems?

PD Servo

$$va = -k * error + -k_d * error\_derivative$$



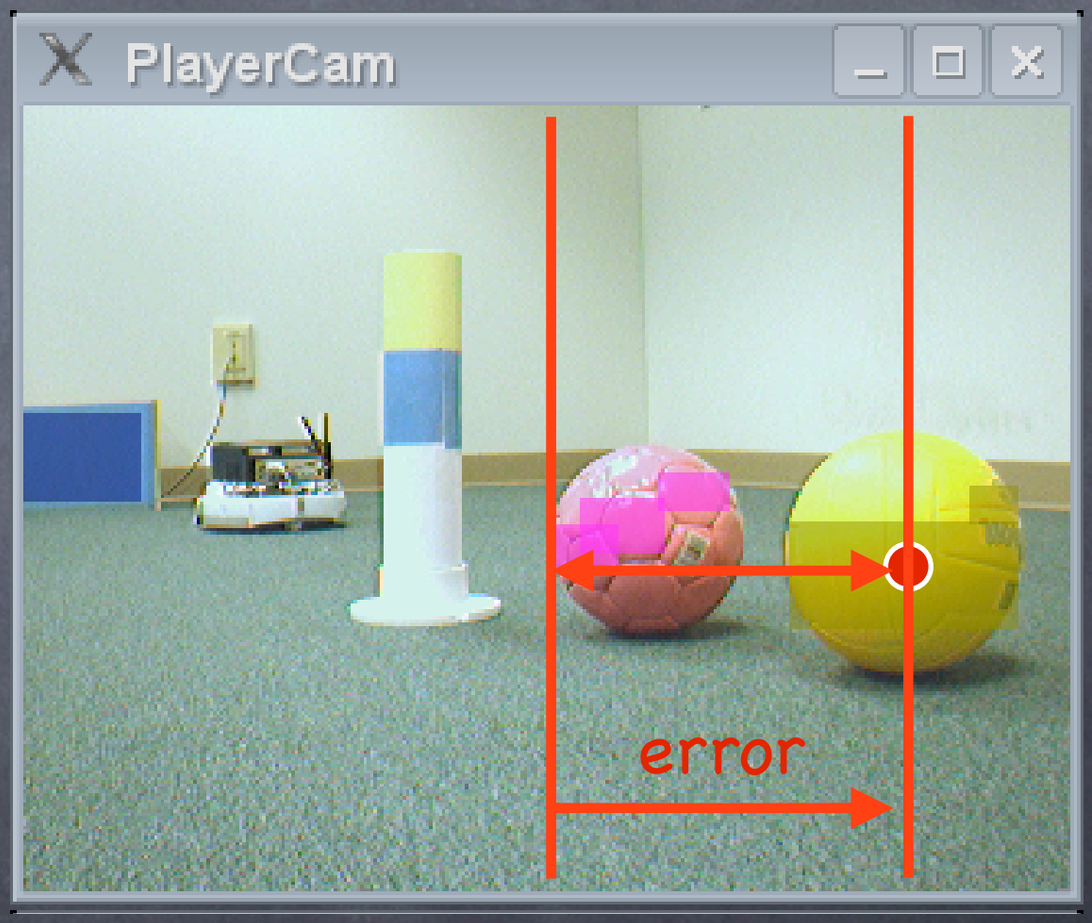
spring-damper model



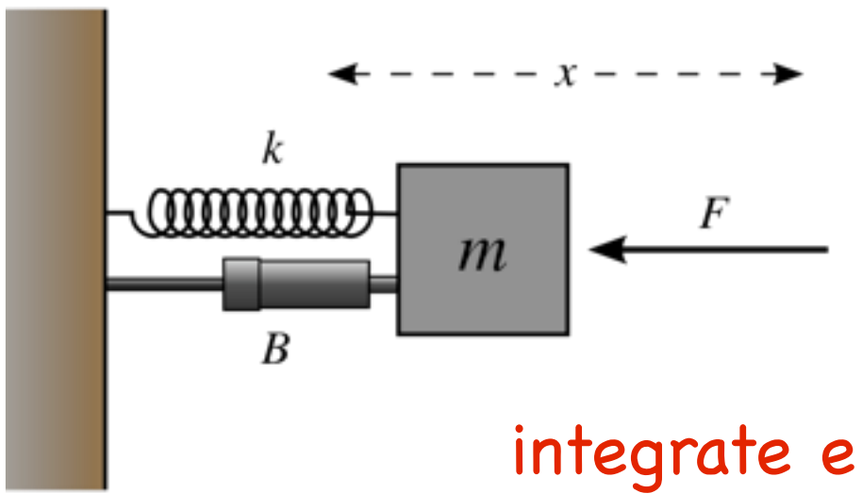
steady-state error

PID Servo

$$va = -k * error + -k_d * error\_derivative + k_i \Sigma_{t..t-\Delta}\{error\}$$



spring-damper model



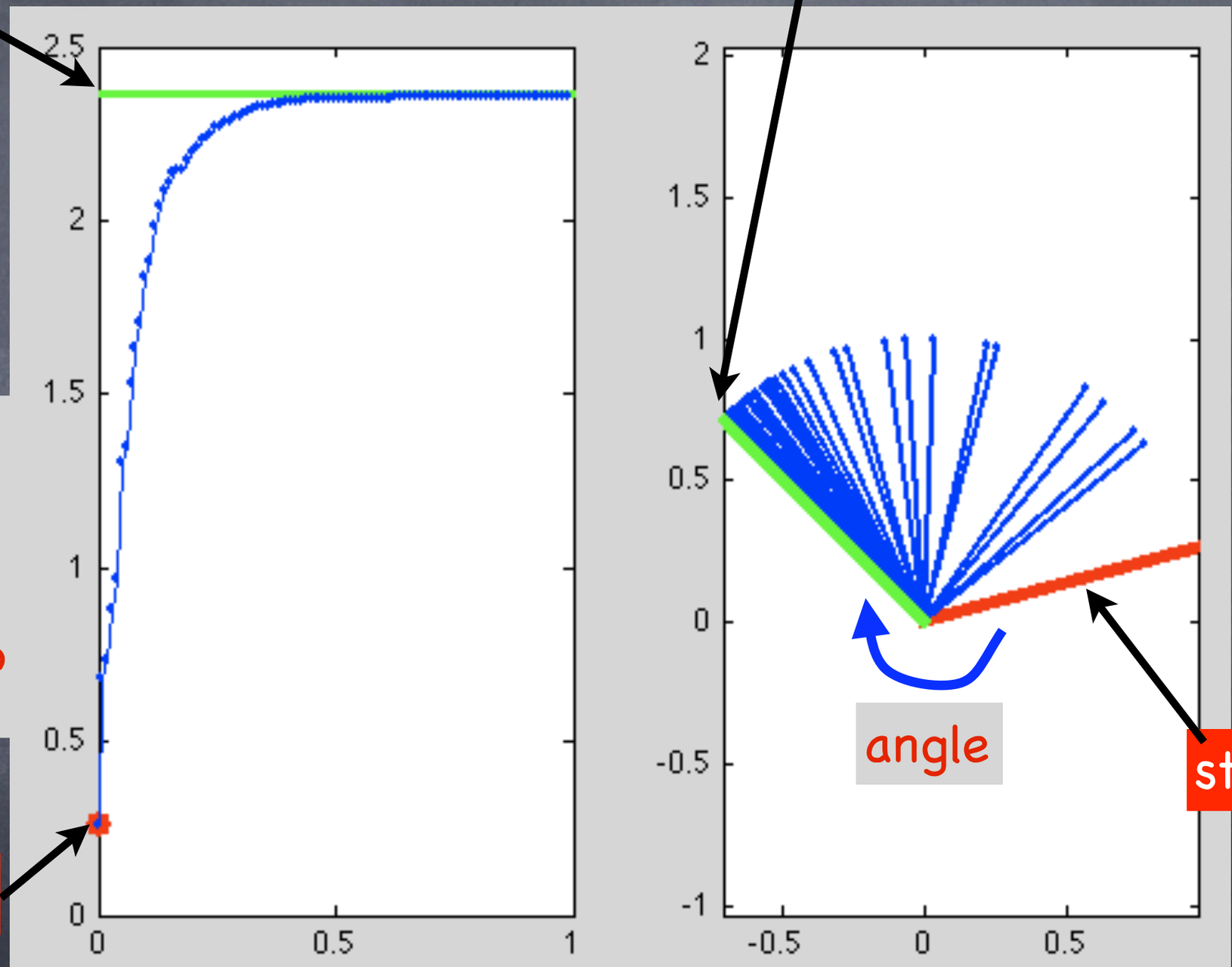
integrate error over time

1 DOF matlab example

- Proportional-derivative-integral servo (PID)
- http://www.cs.brown.edu/courses/cs148/pub/pd_control_1Dservo.m
- 4 gain settings, hacky dynamics
- PID pseudo code from Wikipedia

```
previous_error = 0
integral = 0
start:
  error = setpoint - actual_position
  integral = integral + (error*dt)
  derivative = (error - previous_error)/dt
  output = (Kp*error) + (Ki*integral) + (Kd*derivative)
  previous_error = error
  wait(dt)
  goto start
```

desired angle

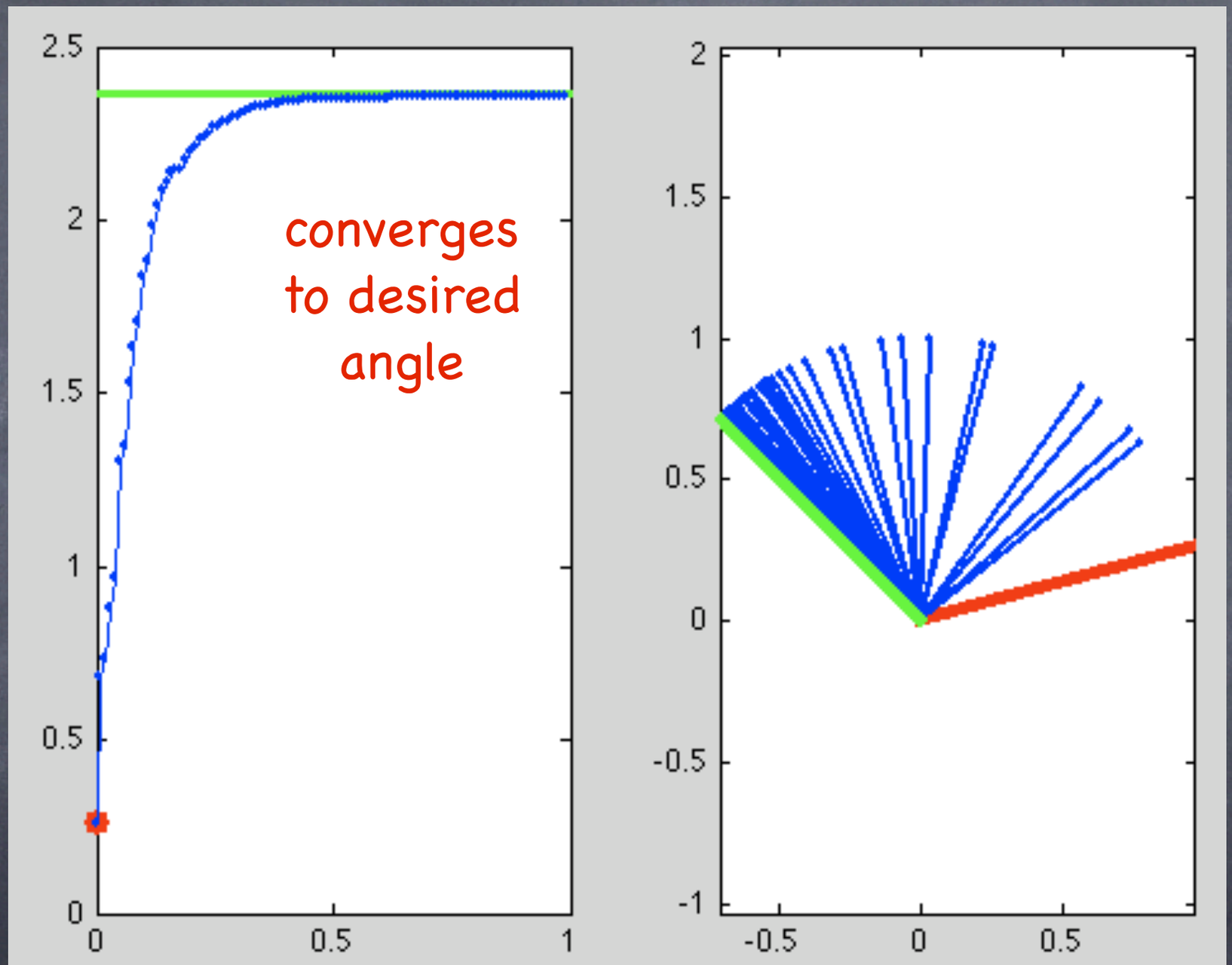


desired angle

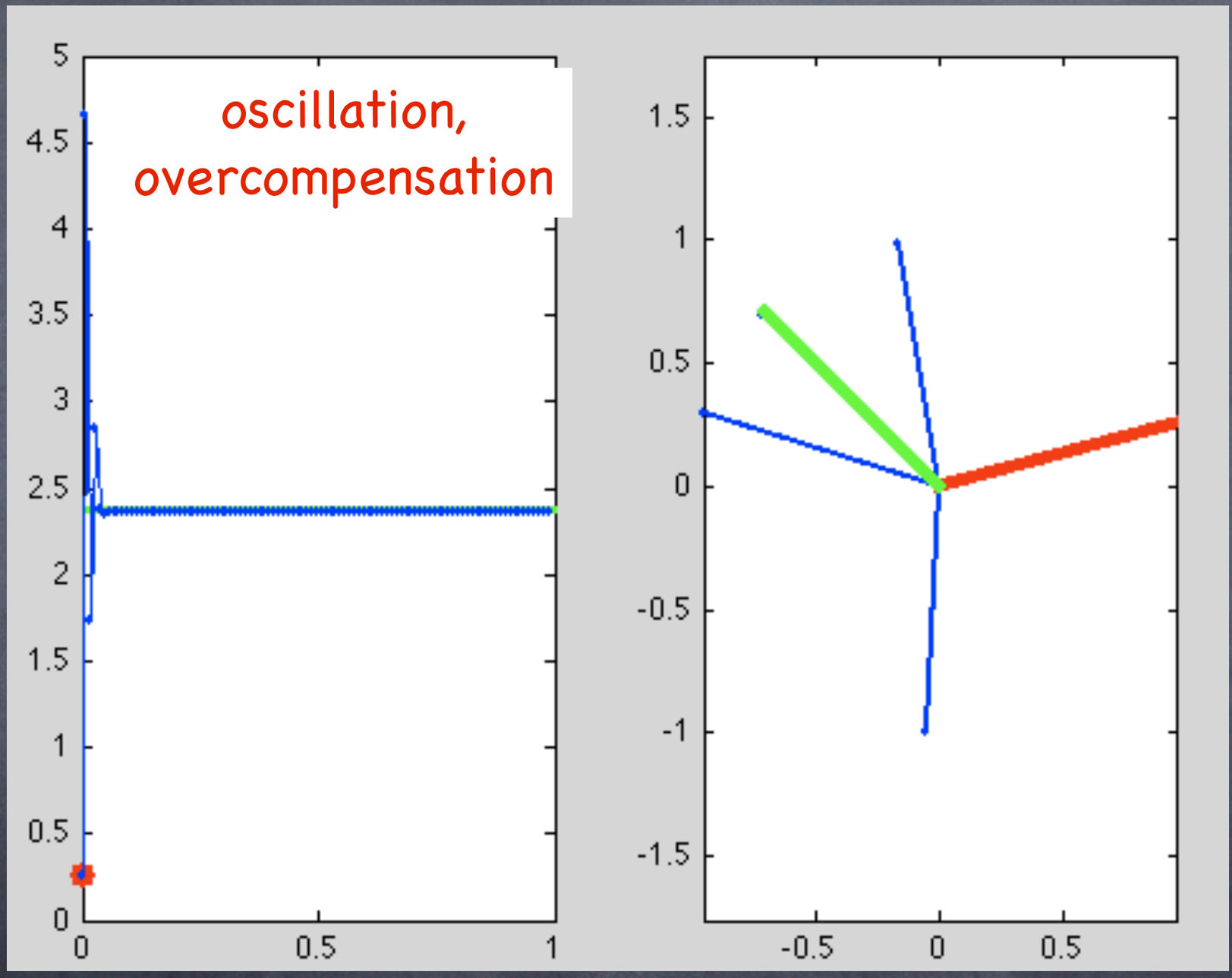
start angle

start angle

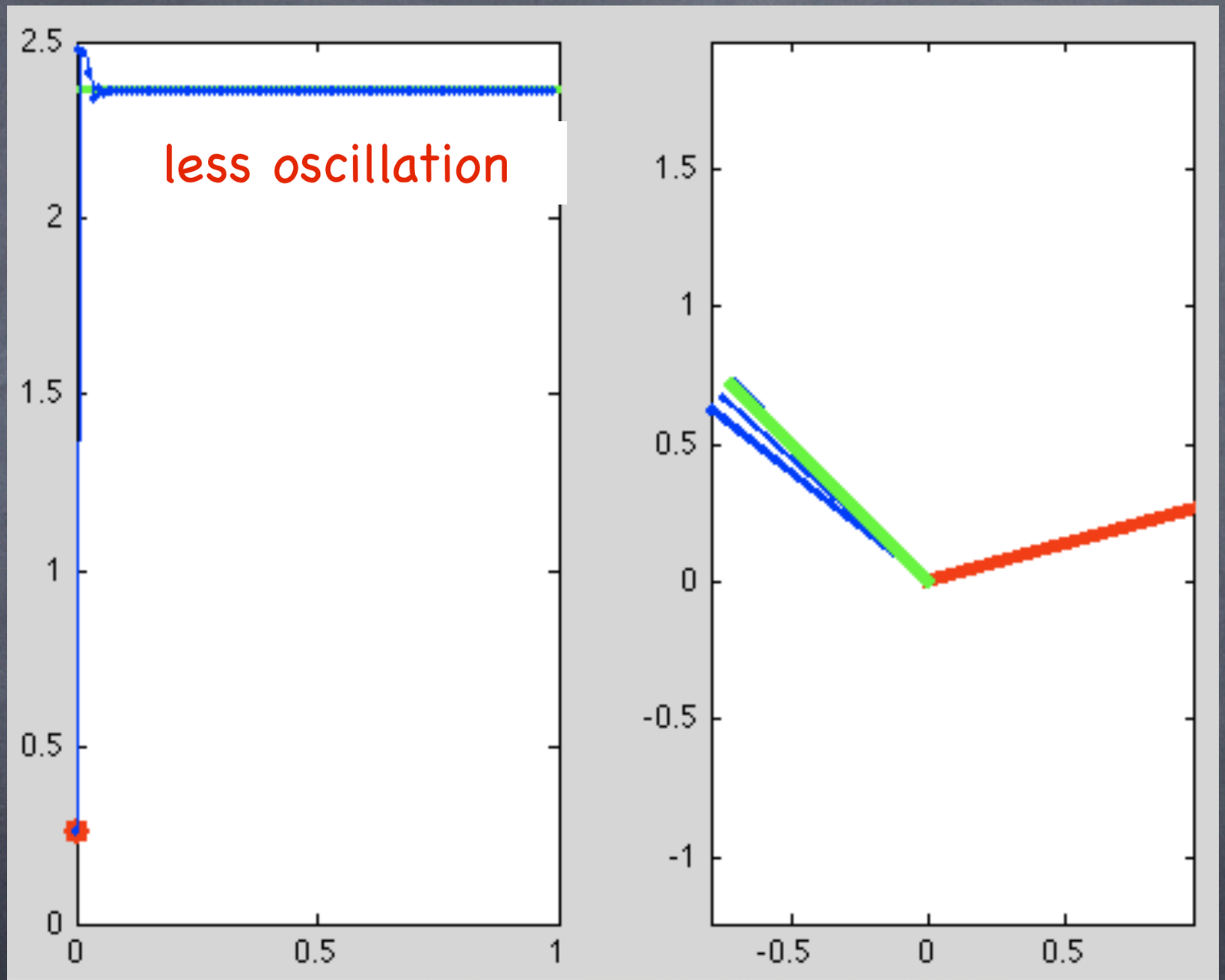
"lightly stiff spring"



"stiffer spring, slightly damped"



"stiffer spring, more damping"



"overstiff spring"

diverges, unstable