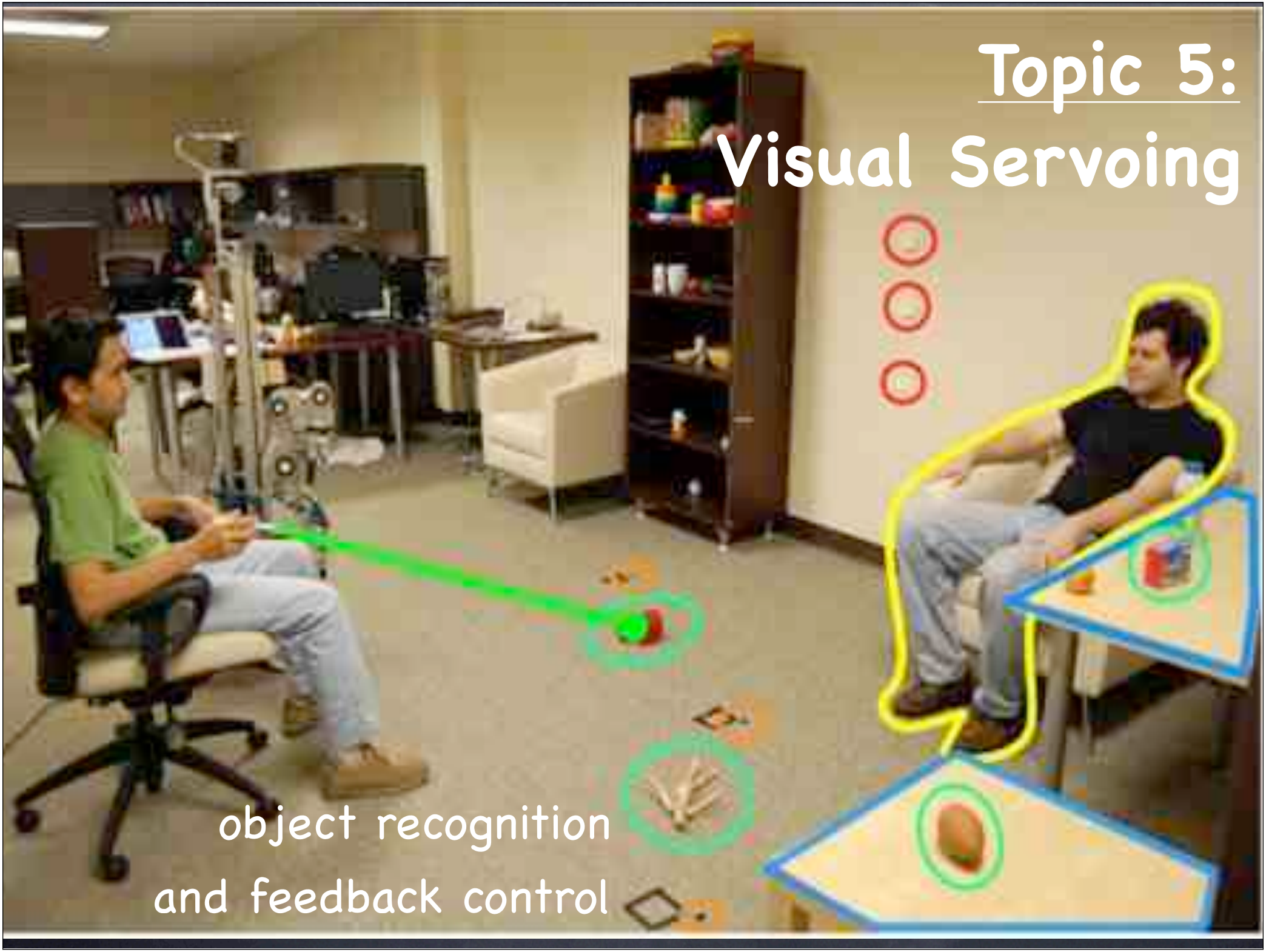
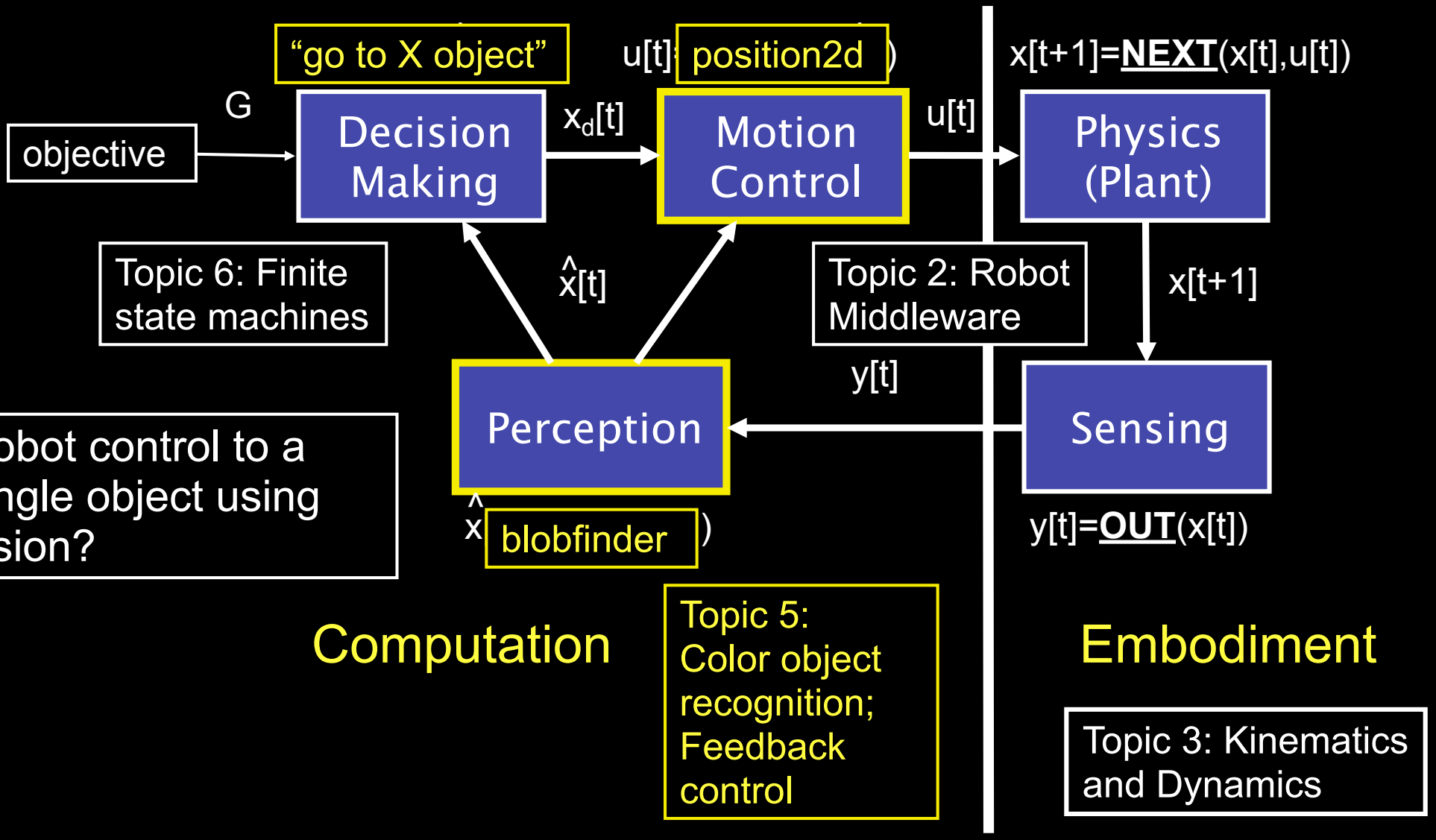

Topic 5:
Visual Servoing
object recognition
and feedback control

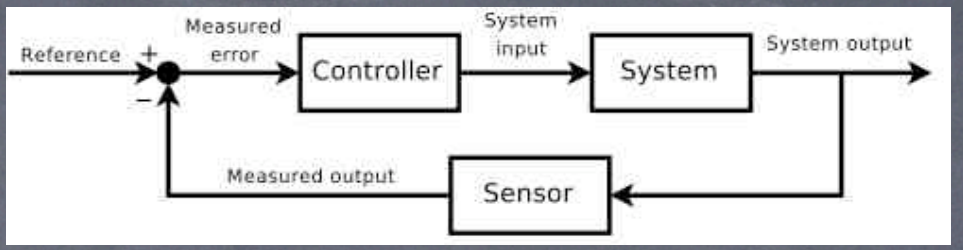
robot control loop

- someone please sketch on the board

The Robot Control Loop

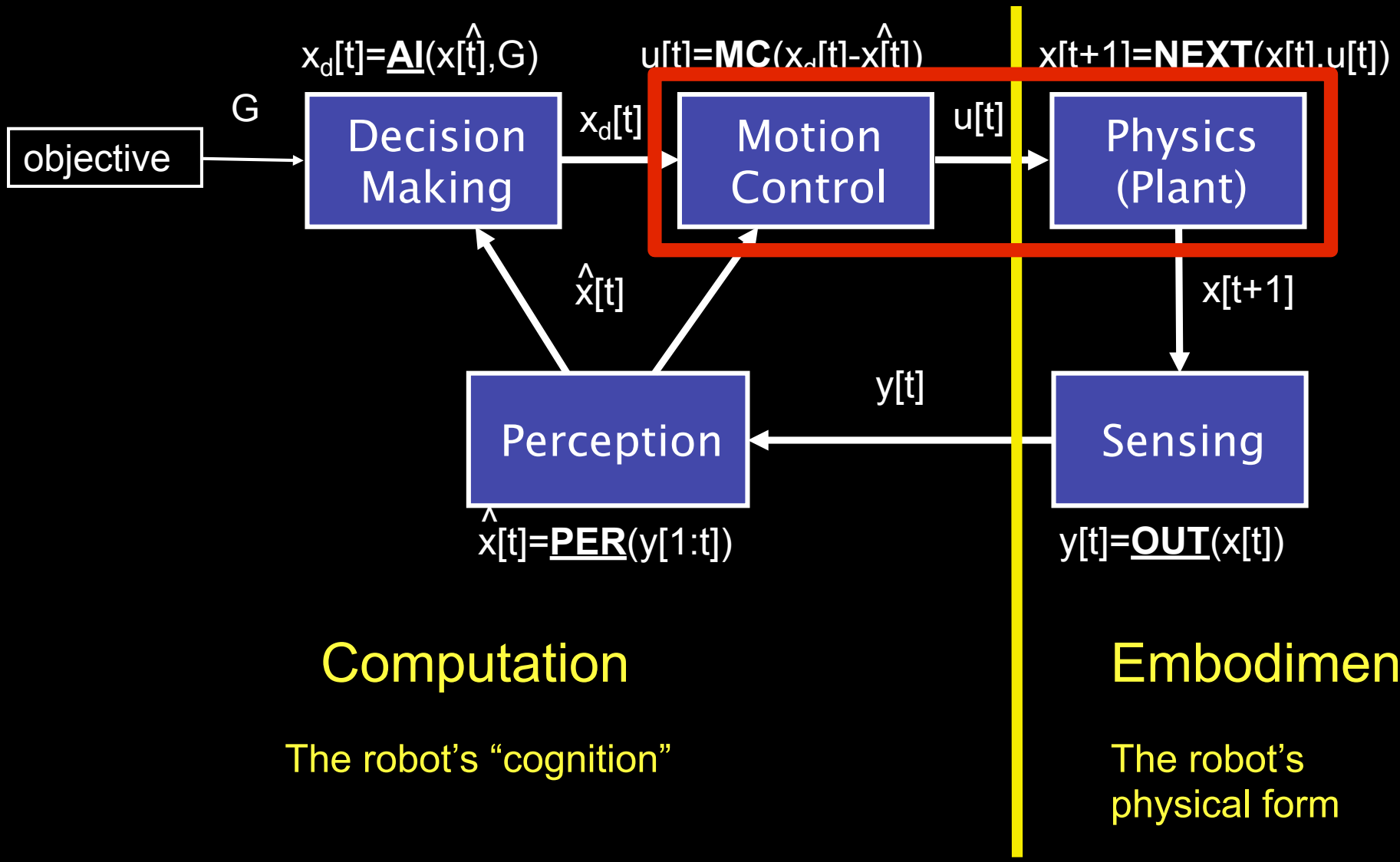


Control theory

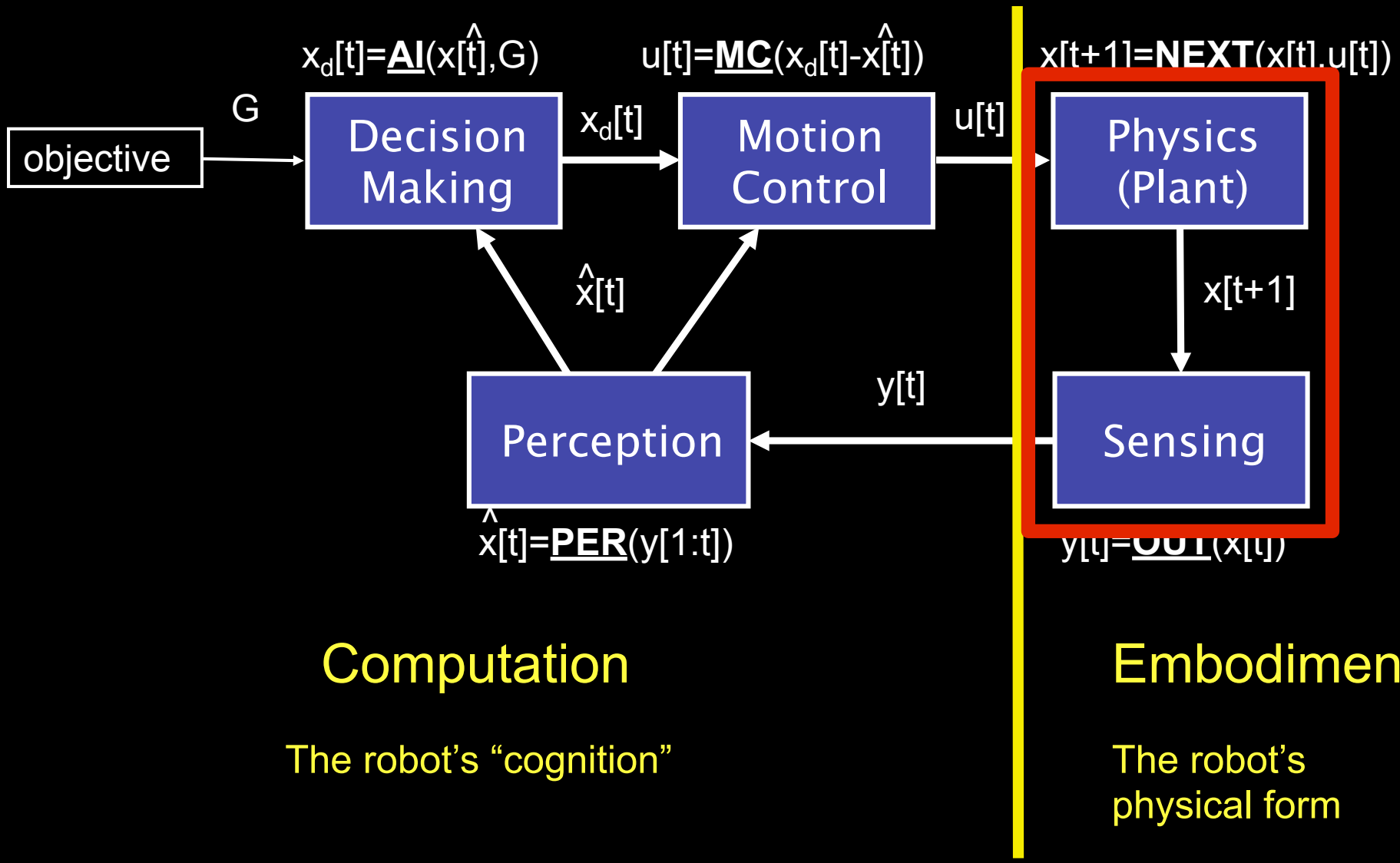


- Control theory: study behavior of dynamical systems
- Forms of control systems:
  - Open-loop control: without system feedback
  - Closed-loop control: adjusts to sensor observations
    - Feedback control: minimize sensed error
  - Open and closed loop
    - Feedback-feedforward: predict and correct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

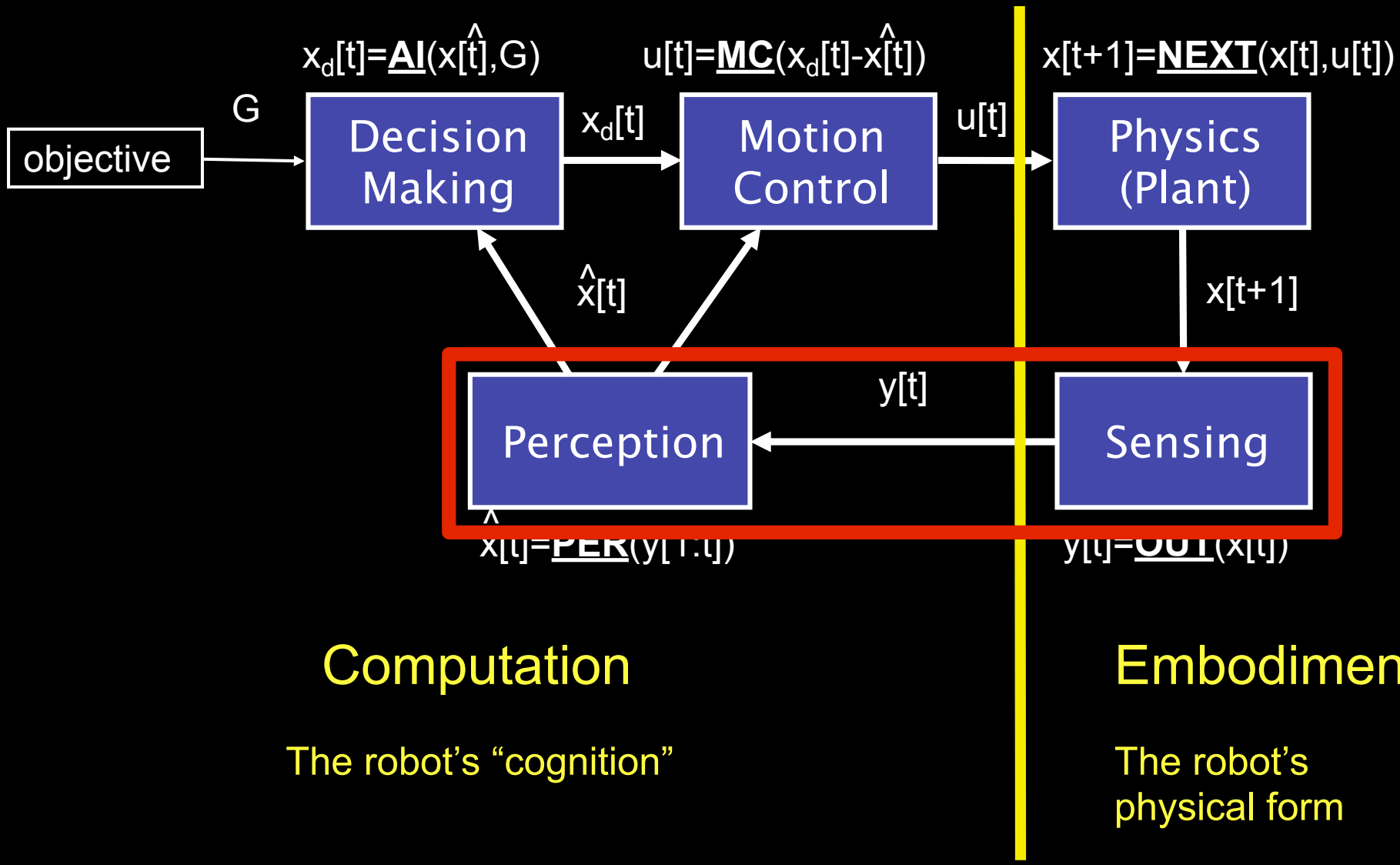
Open-loop control: traditional



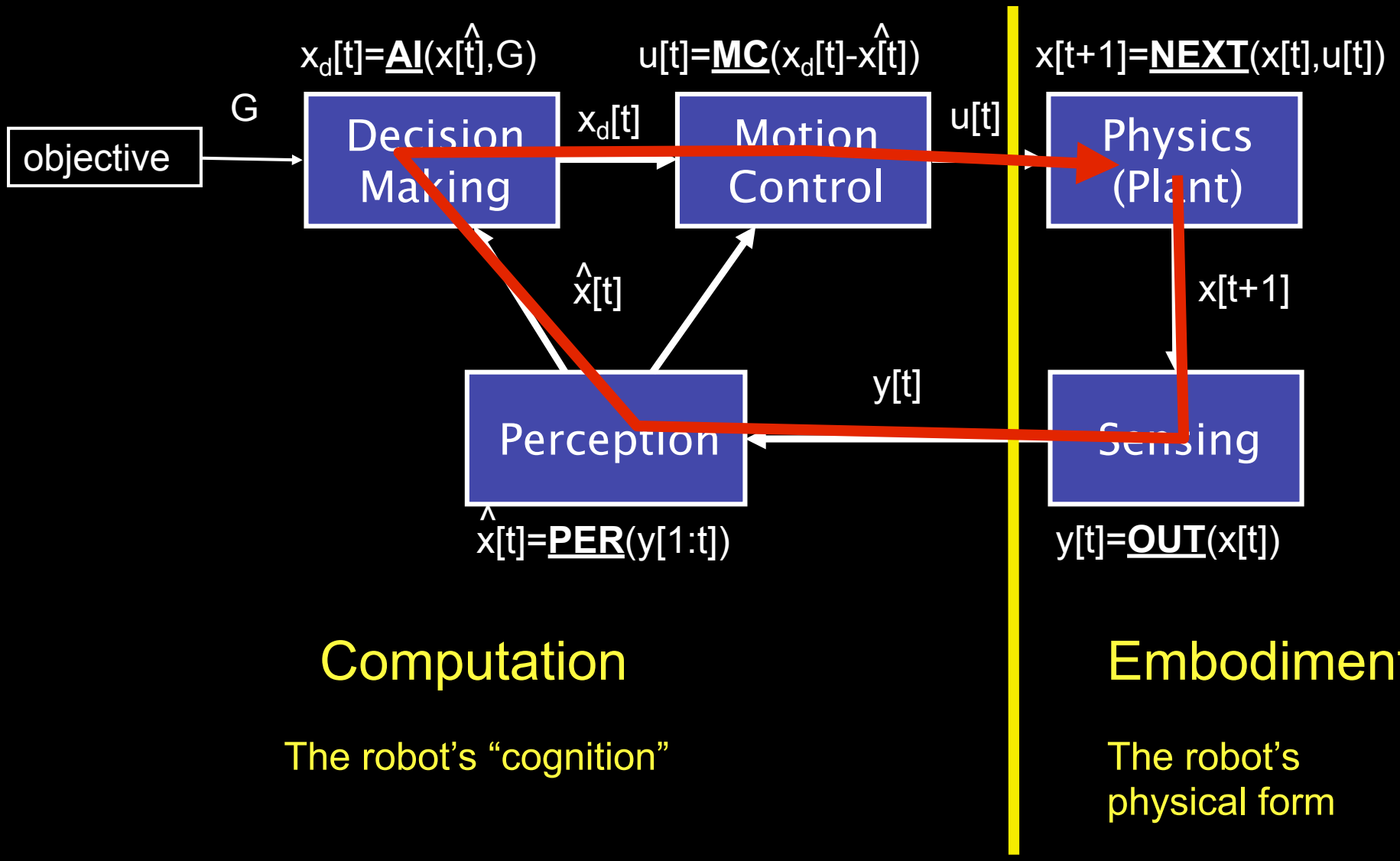
Open-loop: graphics



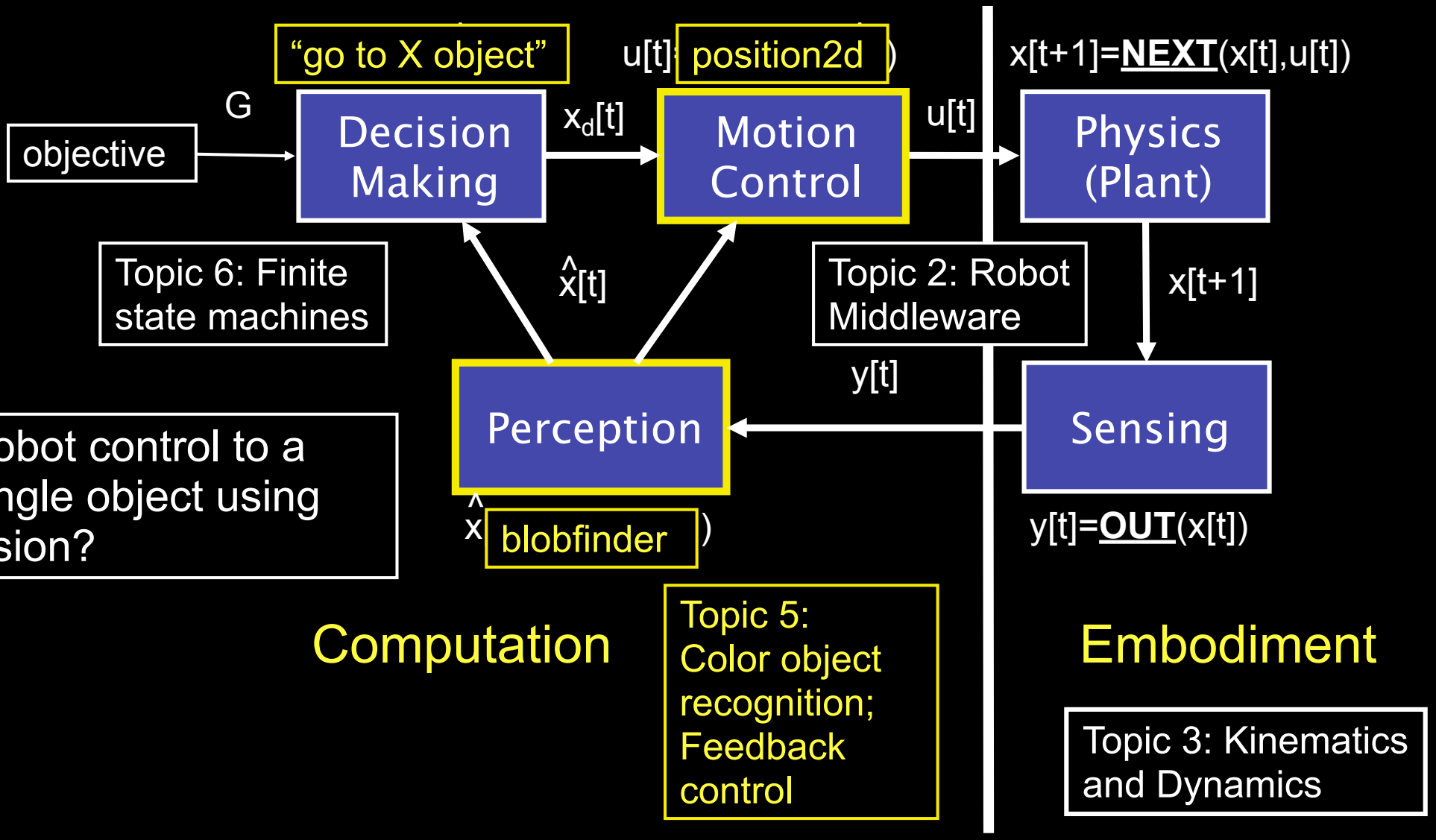
Open-loop: vision



Closed loop: robotics



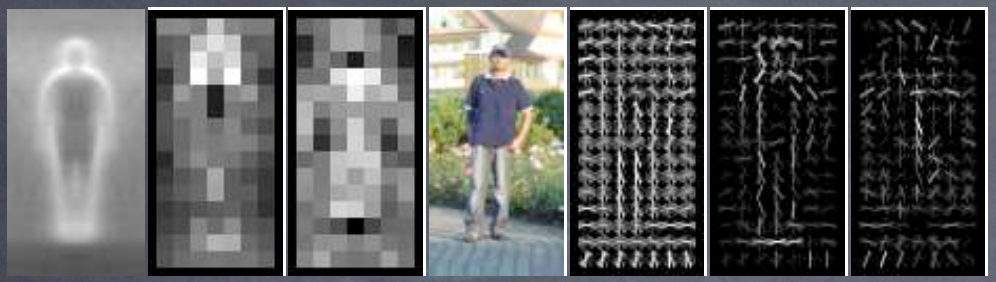
The Robot Control Loop



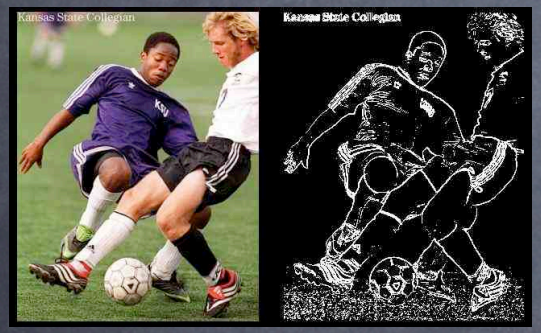
Object recognition

- 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

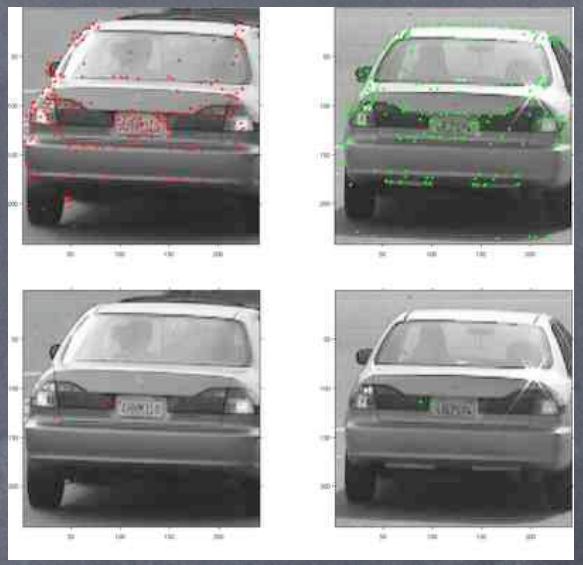
Appearance features



HoG: histograms of oriented gradients

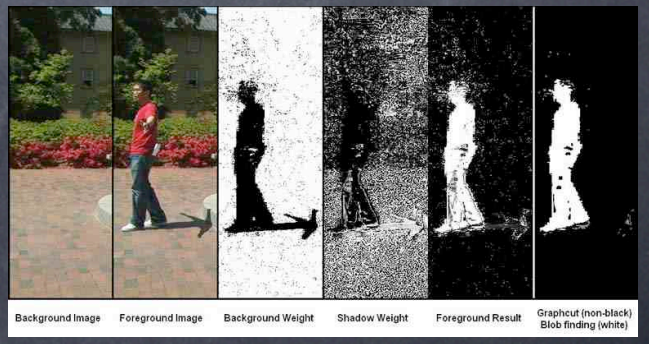


edge detection



SIFT: corners, hierarchy

template matching



silhouette detection:
"green screen"

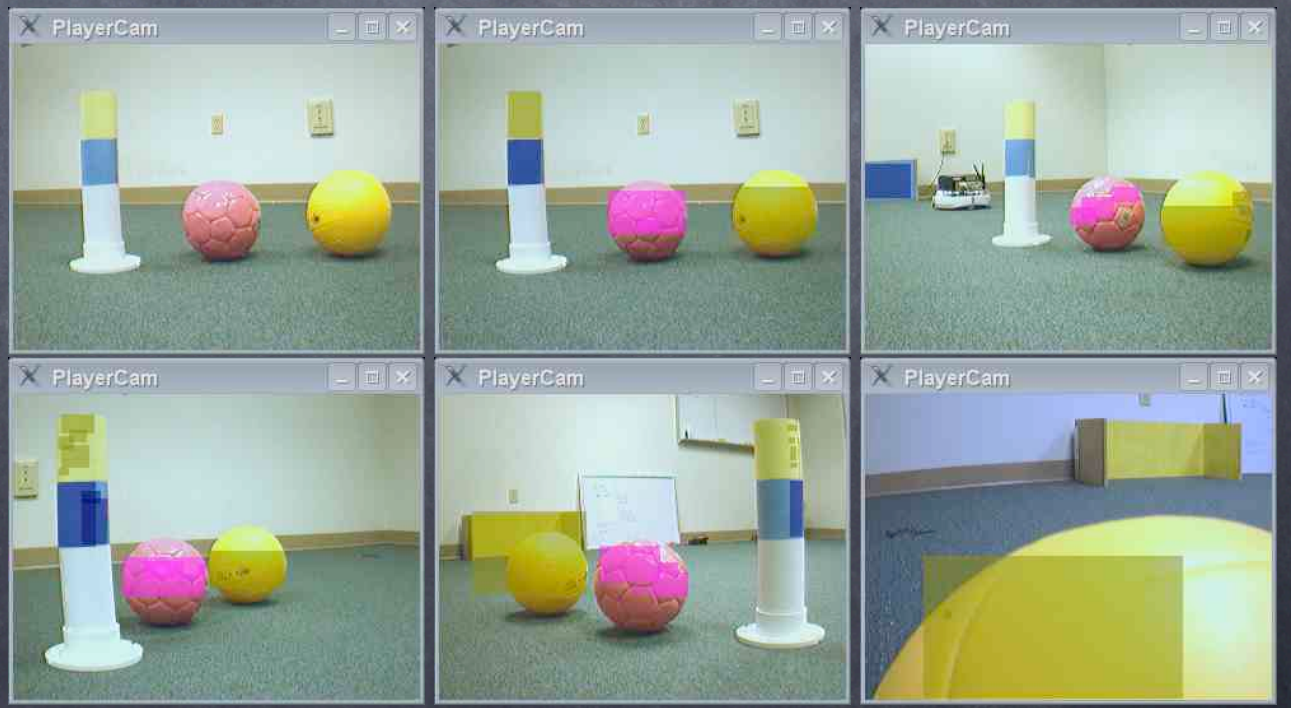
blobfinder

video4l2

-> camerauvc

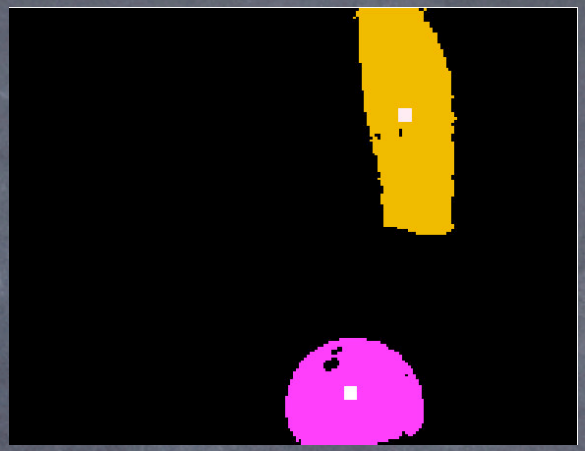
-> blobfinder

```
typedef struct player_blobfinder_blob
{
  uint32_t id;
  uint32_t color;
  uint32_t area;
  uint32_t x;
  uint32_t y;
  uint32_t left;
  uint32_t right;
  uint32_t top;
  uint32_t bottom;
  float range;
} player_blobfinder_blob_t;

typedef struct player_blobfinder_data
{
  uint32_t width;
  uint32_t height;
  uint32_t blobs_count;
  player_blobfinder_blob_t blobs[PLAYER_BLOBFINDER_MAX_BLOBS];
} player_blobfinder_data_t;
```



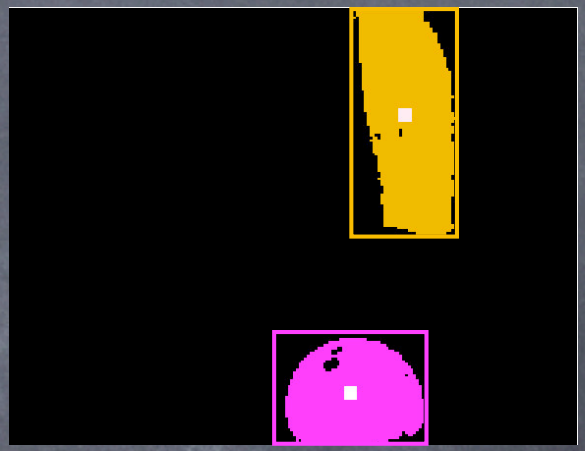
Color blobfinding

Input image



Color threshold

How?

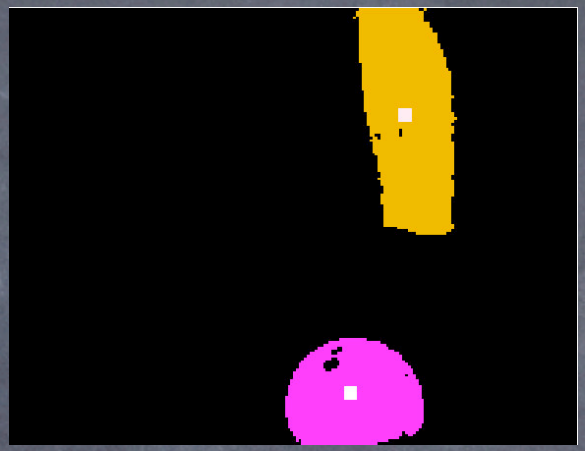


Group regions

How?

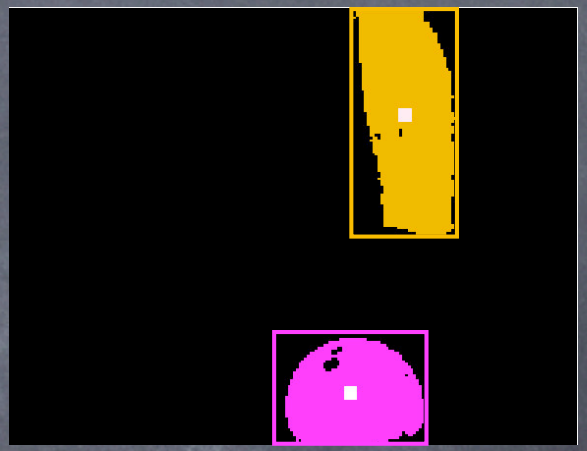
Color blobfinding

Input image



Color threshold

Color calibration



Group regions

Connected Components

Color calibration

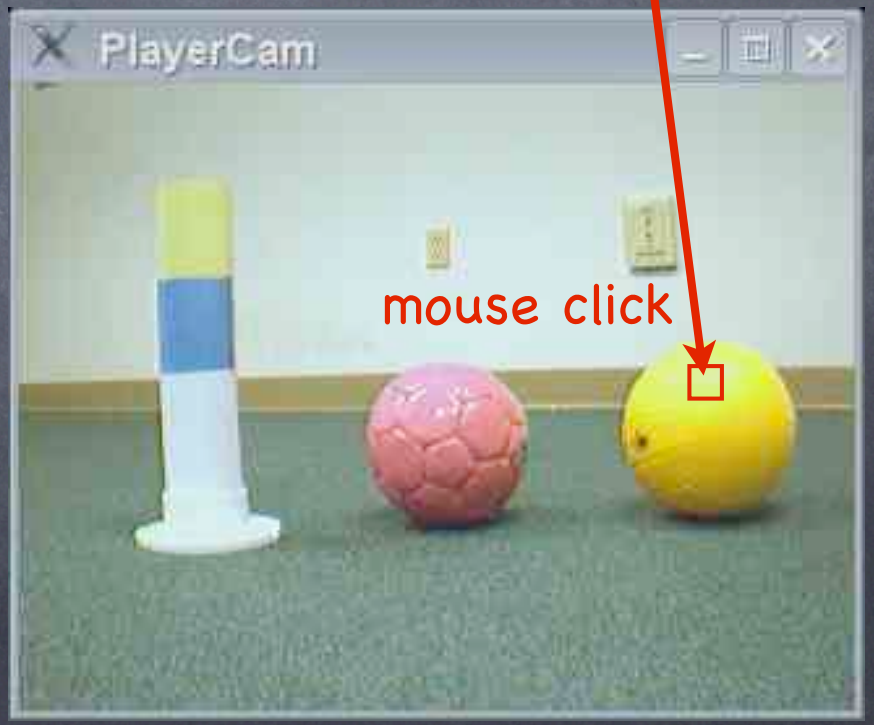
```
[Colors]
(255,  0,  0) 0.000000 10 Red
(  0,255,  0) 0.000000 10 Green
(  0,  0,255) 0.000000 10 Blue

[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)

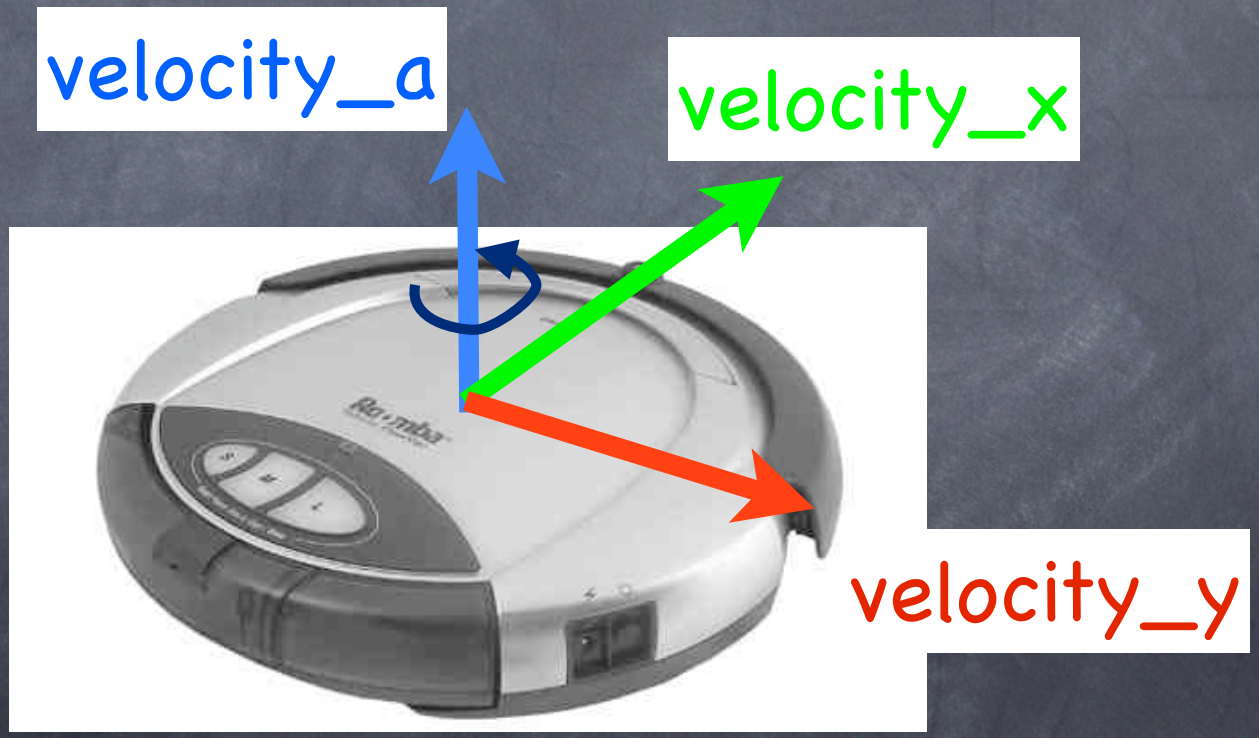
playercam_yuv output

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



position2D

playerc_position2d_set_cmd_vel(vx, vy, va)

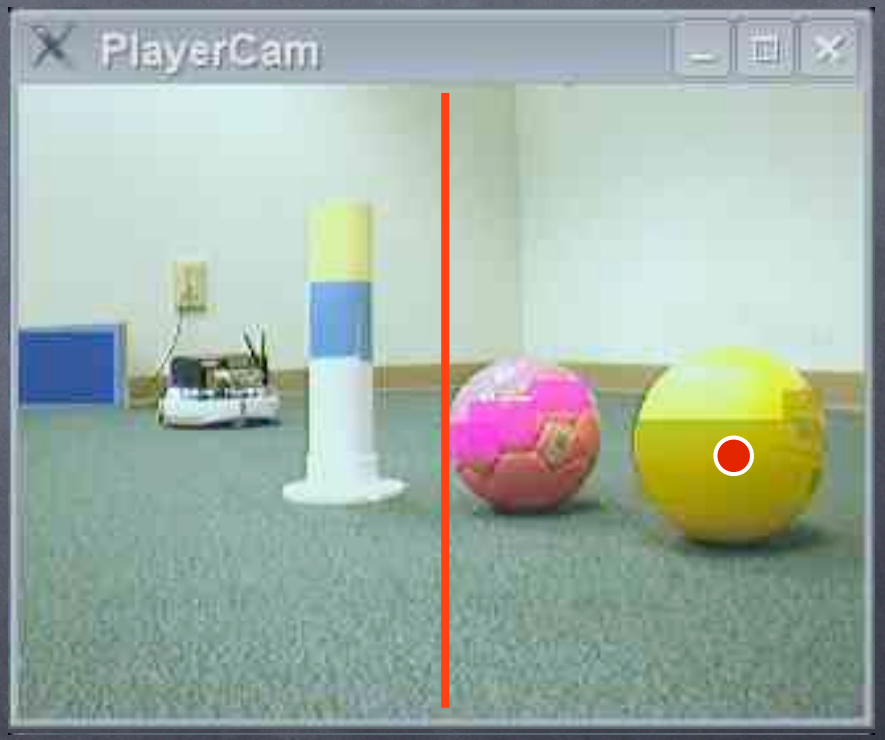


Move to the ball?

- how can our robot move to the ball?

One approach

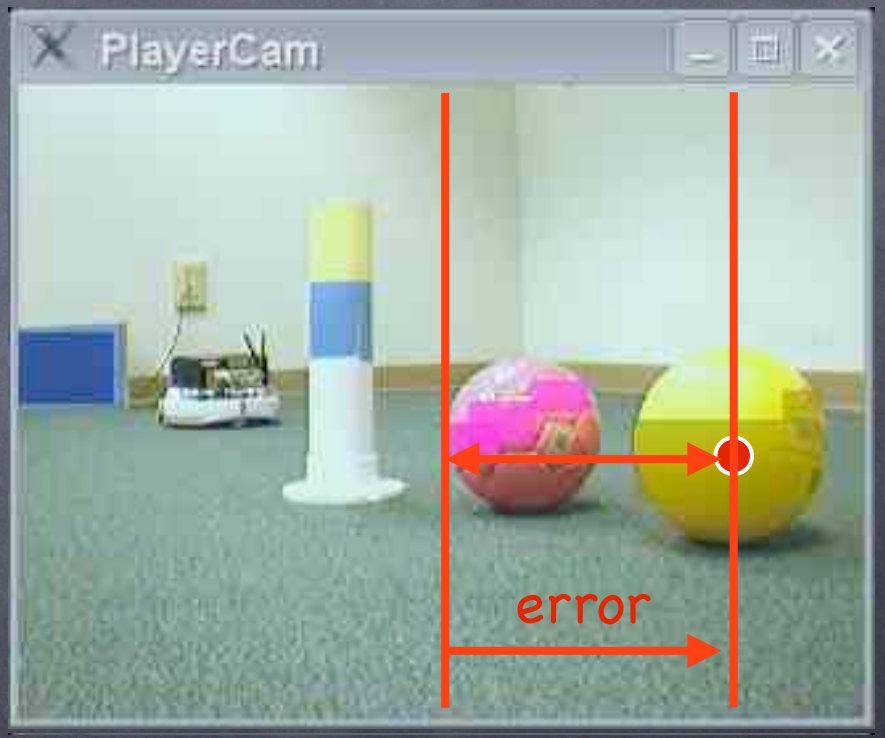
- Center ball in image
- Once centered, drive forward
- Can we state in terms of vx and va?



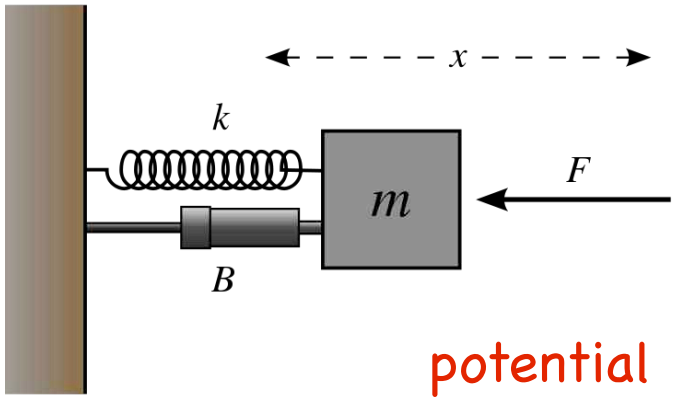
P Servo

va = -k * error

simple form of feedback control



spring mass model

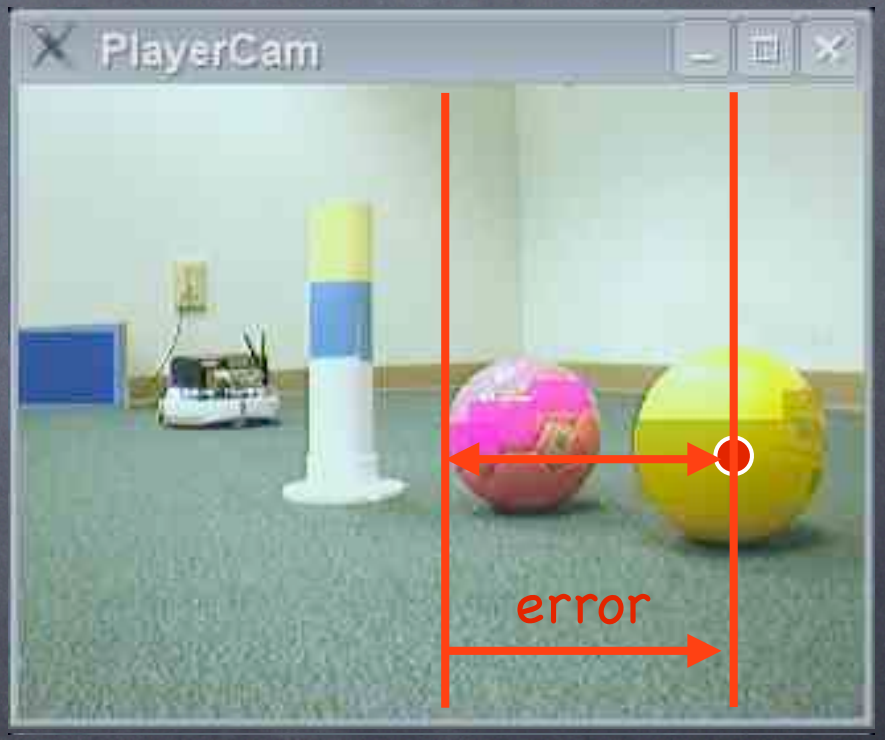


potential problems?

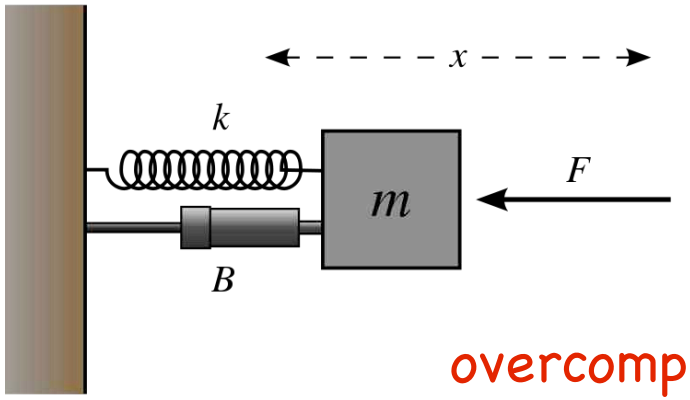
P Servo

va = -k * error

gain selection
ball out-of-view
false positives



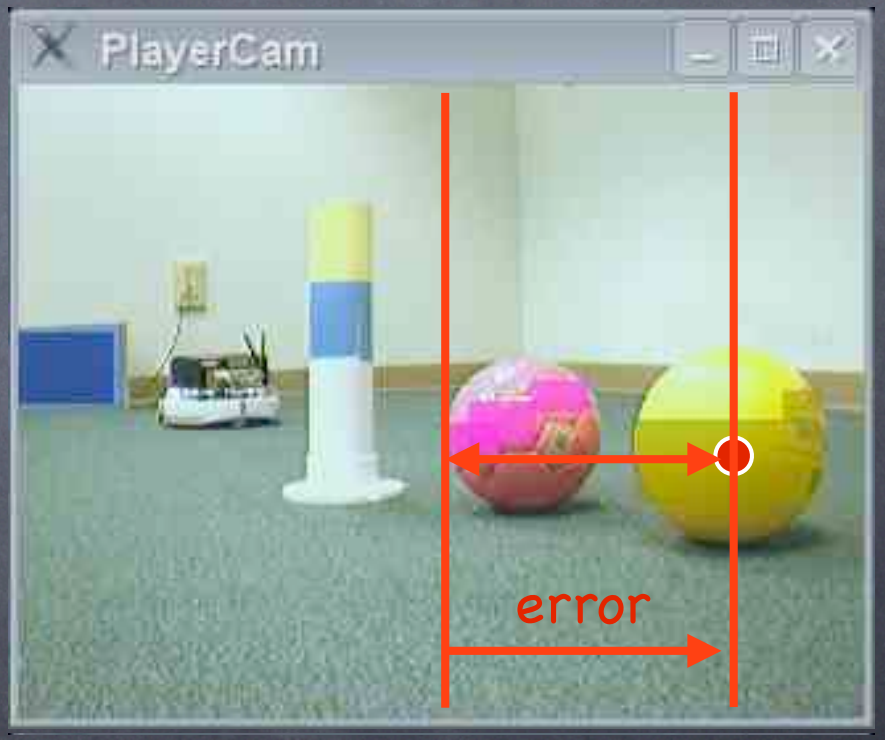
spring mass model



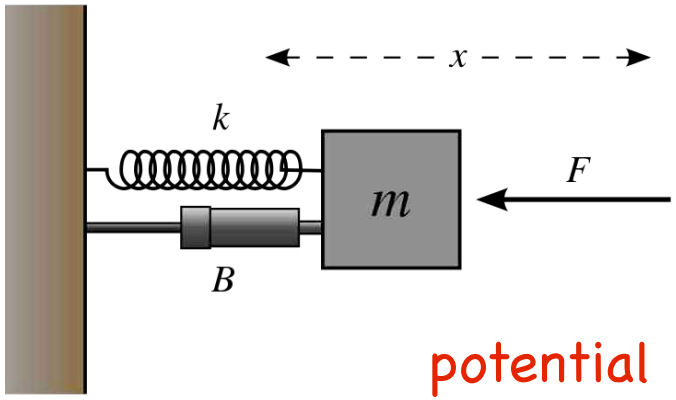
overcompensation

PD Servo

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



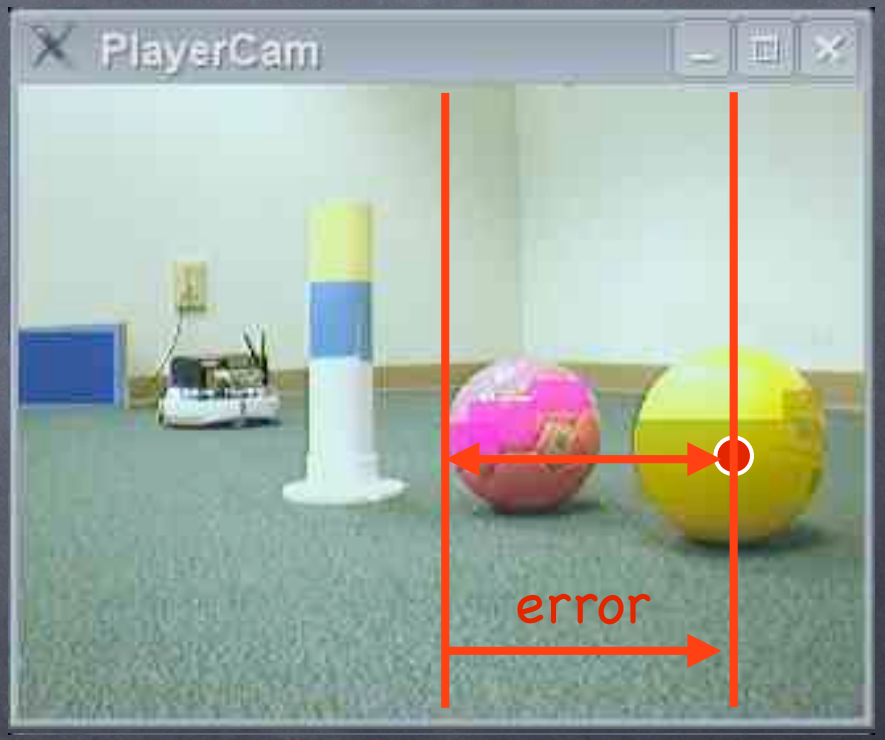
spring-damper model



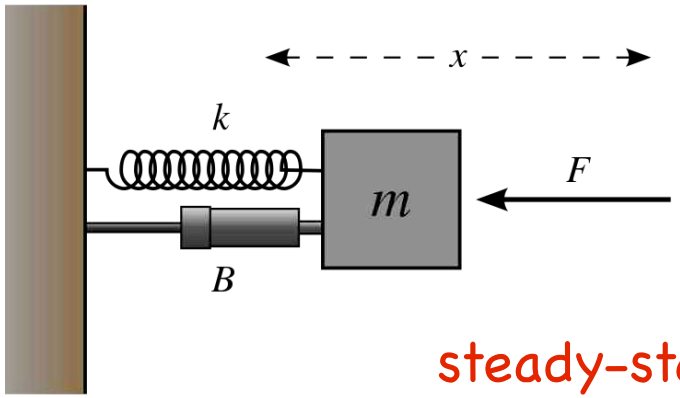
potential problems?

PD Servo

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



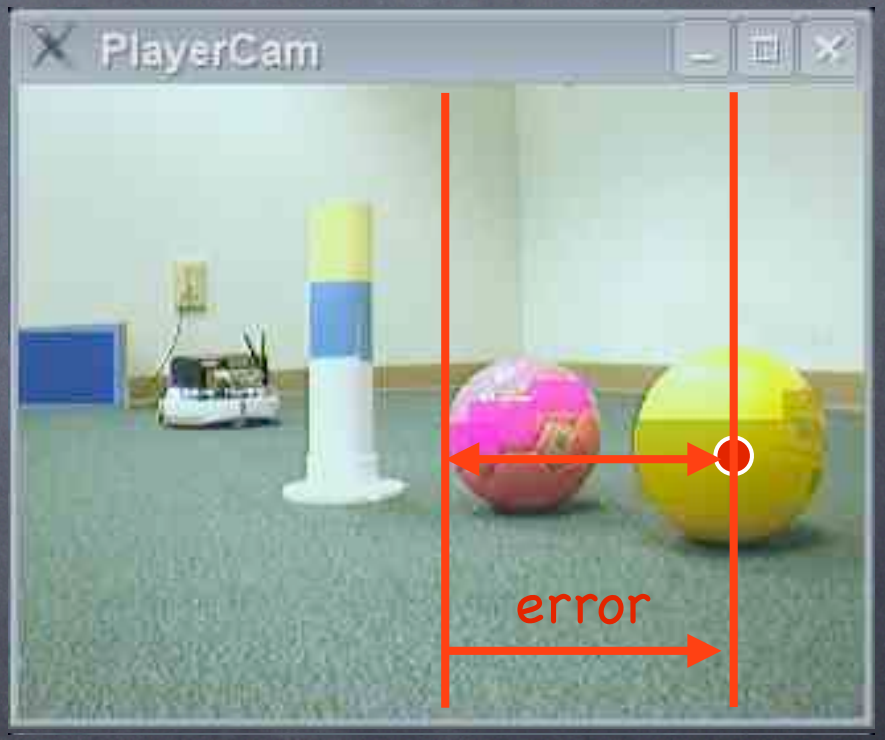
spring-damper model



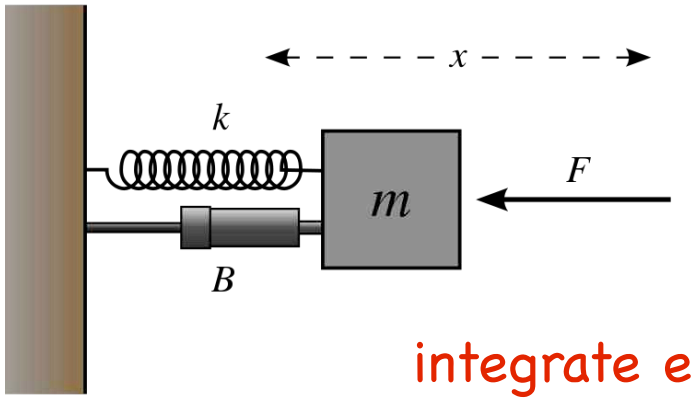
steady-state error

PID Servo

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



spring-damper model



integrate error over time

matlab example

- Proportional-derivative-integral servo (PID)
- /course/cs148/pub/pd_control_1Dservo.m
- 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
```