Mobile and Sensor Systems

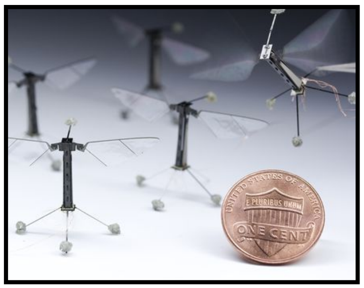
Mobile Robots for Robotic Sensor Networks

Dr. Amanda Prorok

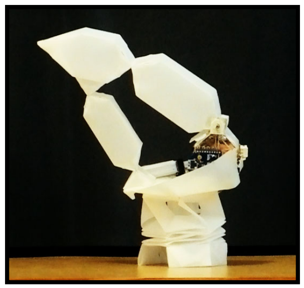
UNIVERSITY OF CAMBRIDGE

Autonomous Robots

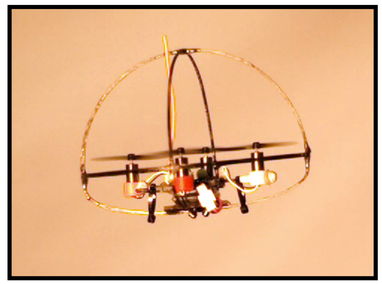
- What is a robot?



microrobots
[Wood, Harvard]

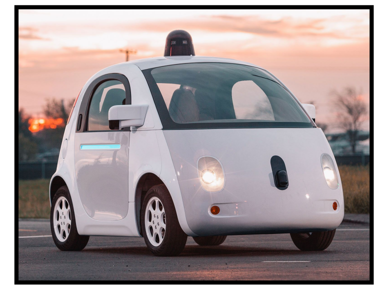


self-foldable / self-actuated
[Sung and Rus; MIT]



lightweight aerial robots
[Kumar et al.; UPenn]

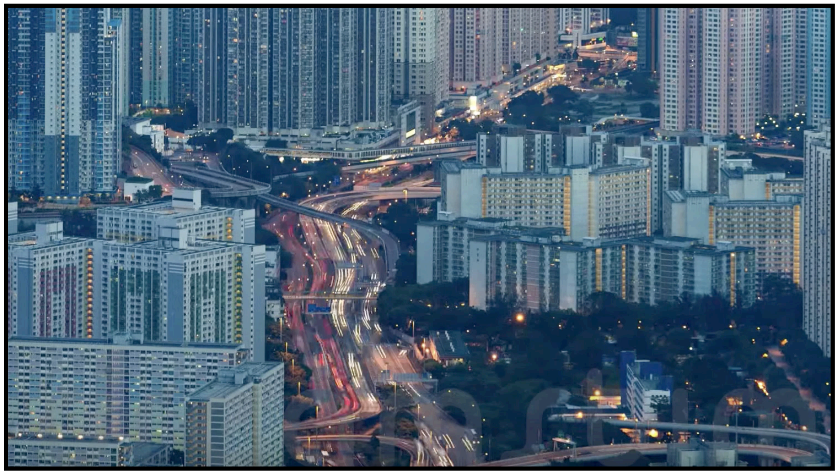
consumer-grade drones



autonomous vehicles
[Google]

- Challenges:
  - How to model and perceive the world?
  - How to process information and exert control?
  - How to reason and plan in the face of uncertainty?

Robots and Mobile Systems



smart infrastructure / mobility-on-demand

connected vehicles / automated highways

drone swarms / surveillance

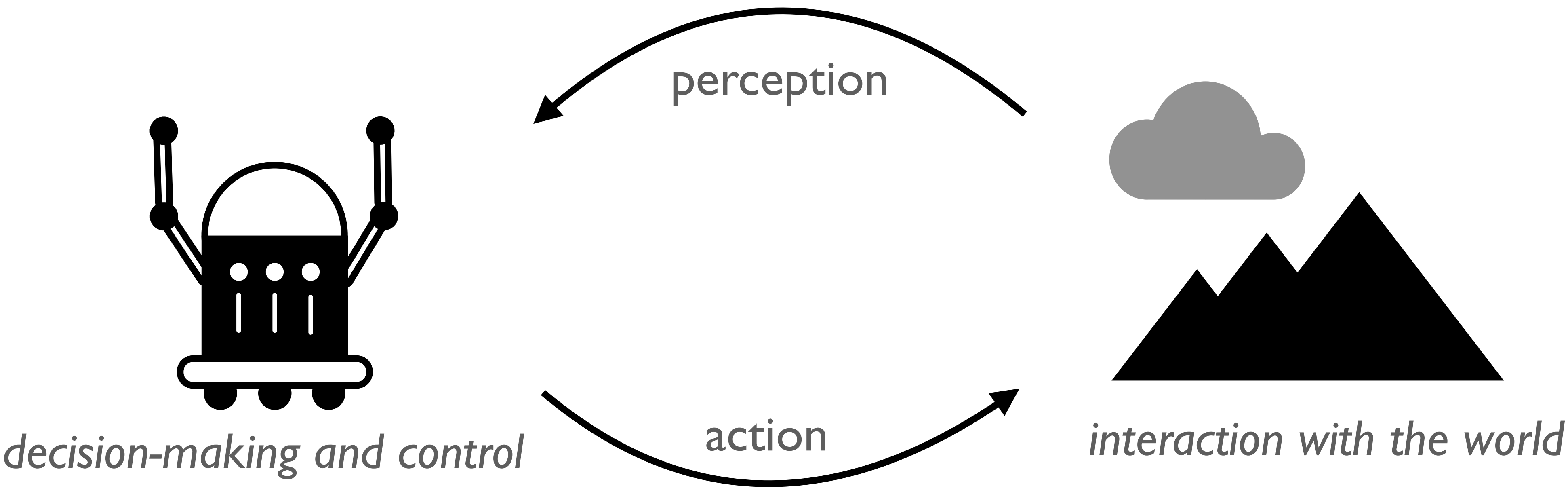
truck platoons / long-haul transport

In this Lecture

- Introduction to mobile robots
- Methods to create a **robotic sensor network**
  1. How to deploy multiple robots to cover an area?
     - *Area tessellation*
     - *Coverage control*
     - *Lloyds algorithm*
  2. How to use multiple robots for pose estimation?
     - *Collaborative particle filter*
  3. How to move a robot?
     - *Basic principles of kinematics*

What is a Robot?

- Basic building block of autonomy: perception-action loop



Three main variants:

1. Reactive (e.g., nonlinear transform of sensor readings)
2. Reactive + memory (eg., filter, state variables)
3. Deliberative (e.g., planning)

Sensors for Robots

- Proprioceptive vs. exteroceptive
  - **Proprioceptive:** *"body"* sensors, e.g., motor speed, battery voltage, joint angle
  - **Exteroceptive:** *"environment"* sensors, e.g., distance measurement, light intensity
- Passive vs. active
  - **Passive:** *"measure ambient energy"*, e.g., temperature probes, cameras, microphones
  - **Active:** *"emit energy, and measure the environmental reaction"*, e.g., infrared proximity sensors, ultrasound sensors

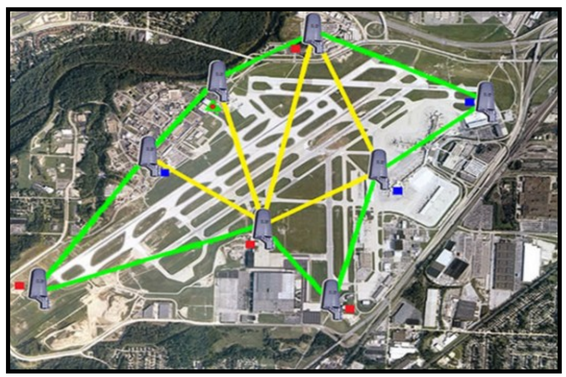
Sensor and Actuators

- Actuators
  - For different purposes: e.g., locomotion, control of a body part, heating, sound emission.
  - Examples of electrical-to-mechanical actuators: DC motors, stepper motors, servos, loudspeakers.
- Uncertainty and disturbances
  - Causes for **actuation noise**:
    e.g., wheel slip, slack in mechanism, "kidnapping"
  - Causes for **sensor noise**:
    e.g., environmental factors, cheap circuitry

Multi-Robot Systems

- Terms used: robot swarms / robot teams / robot networks
- Why?
  - Distributed nature of many problems
  - Overall performance greater than sum of individual efforts
  - Redundancy
- Numerous commercial, civil, military applications

search & rescue



surveillance / monitoring

product pickup / delivery

UNIVERSITY OF CAMBRIDGE

Taxonomy of Multi-Robot Systems

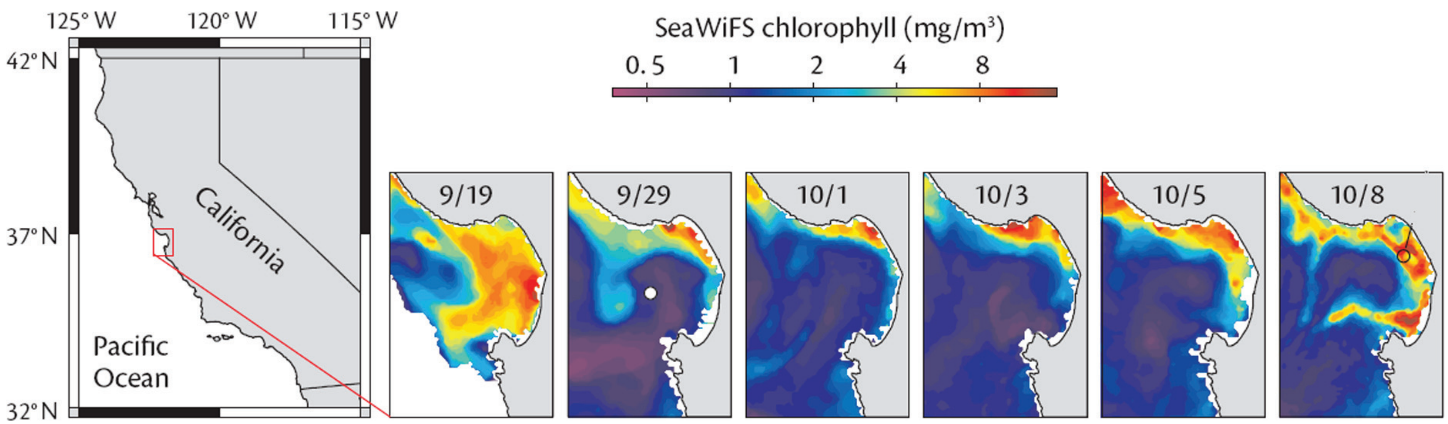
- Architecture: centralized vs. decentralized
  - **Centralized:** one control/estimation unit communicates with all robots to issue commands; requires synchronized, reliable communication channels; single-point failures
  - **Decentralized:** scalable, robust to failure; often asynchronous; sub-optimal performance (w.r.t centralized)
- Communication: explicit vs. implicit
  - **Implicit:** observable states; information exchanged through observation
  - **Explicit:** unobservable states; need to be communicated explicitly
- Heterogeneity: homogenenous vs. heterogeneous
  - Robot teams can leverage inter-robot complementarities

Decentralization

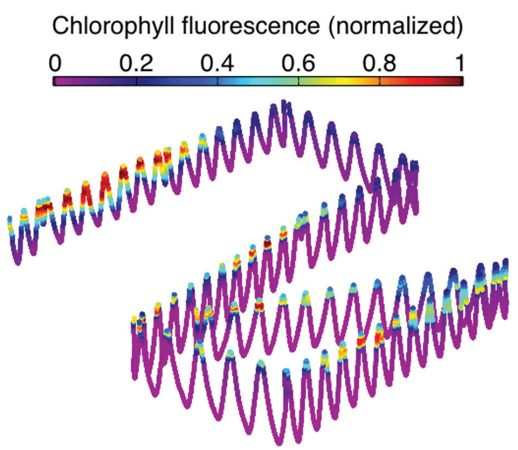
- Goal: Achieve similar (or same) performance as would be achievable with an ideal, centralized system.
- Challenges:
  - Communication: delays and overhead
  - Input: asynchronous; with rumor propagation
  - Sub-optimality with respect to the centralized solution
- Advantages:
  - No single-point failure
  - Can converge to optimum as time progresses
  - 'Any-comm' algorithms exist (with graceful degradation)
  - 'Any-time' algorithms exist (that guarantee gradual improvements)

Robotic Sensor Networks

A key application of multi-robot systems: robotic sensor networks.
Three examples:



1. Coordinated sampling of dynamic oceanographic features with underwater vehicles [Das et al., 2012]:

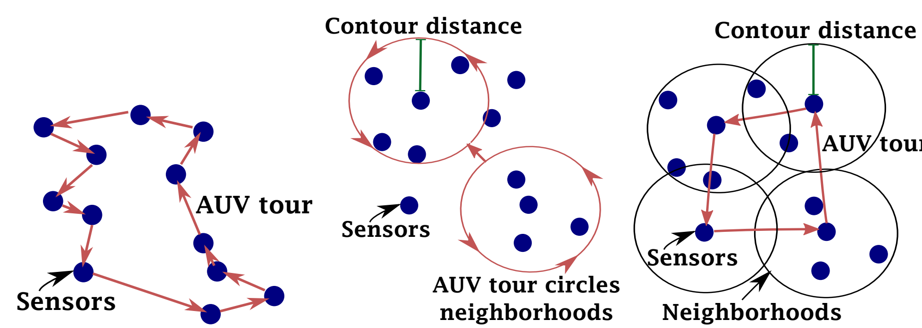


Robotic Sensor Networks


(a)
(b)

2. Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem; [Howard et al., 2002]

3. Underwater Data Collection Using Robotic Sensor Networks; [Hollinger et al., 2011]



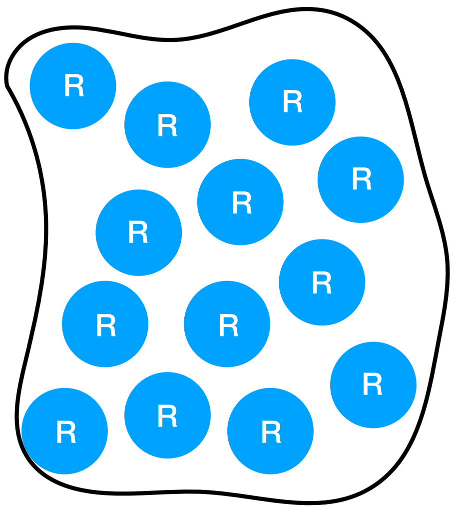
How to obtain coverage of an area?

Coverage

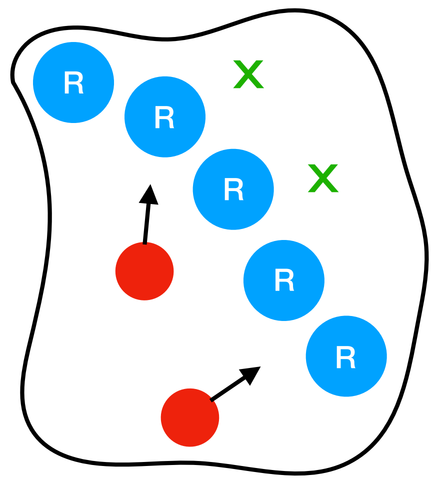
- Coverage classes:
  - **Blanket**: Deploy sensors, e.g. carried by networked robots, in a *static arrangement* to cover an area.
  - **Barrier**: Deploy sensors in a *static arrangement* that minimizes the probability of undetected penetration through the barrier.
  - **Sweep**: *Move a group* of sensors across a coverage area to achieve a balance between maximizing the number of detections per time and minimizing the number of missed detections per area.

[D.W. Gage, 1992]

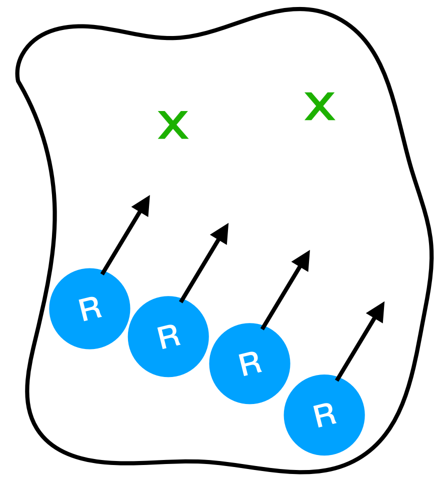
Coverage Classes



Blanket



Barrier



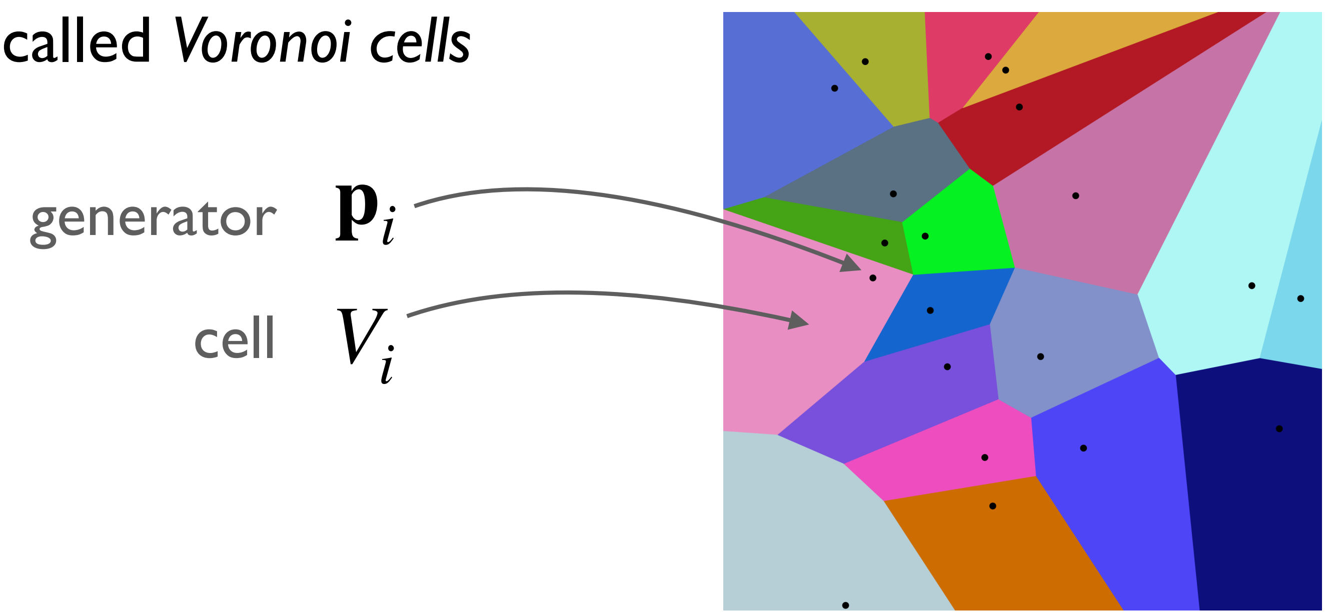
Sweep

Coverage Applications

Application	Coverage Class
Target search & rescue	Sweep
Reconaissance	Sweep
Sentry duty	Barrier
Communications relay	Blanket
Maintenance / inspection	Blanket

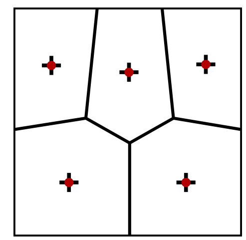
Tessellation

- Voronoi diagram:
  - Partitioning of a plane into regions based on distances to points in a specific subset of the plane.
  - A set of points (called seeds, sites, or generators) is specified beforehand, and for each seed there is a corresponding region consisting of all points *closer to that seed than to any other*.
  - Regions called *Voronoi cells*



Voronoi Coverage

- A widely studied class of solutions to coverage use Voronoi tessellations that optimize the configuration of n robots
- Assumption: One robot (generator) per Voronoi cell
- Optimization objective: minimize the average distance between robots and all points in their respective cells.
- Centroidal Voronoi Tessellation (CVT):



generator position coincides with cell centroids

Density function $\phi(\mathbf{x})$ describes importance of different areas in space

Mass of a cell: $$M_{V_i} = \int_{V_i} \phi(\mathbf{x})\ d\mathbf{x}$$

Centroid of a cell: $$\mathbf{c}_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} \mathbf{x}\ \phi(\mathbf{x})\ d\mathbf{x}$$

Centroidal Voronoi Tessellation

- CVTs minimize this cost function (using Euclidean distance):

$$H(\mathbf{P}) = \sum_{i=1}^{n} H(\mathbf{p}_i) = \frac{1}{2} \sum_{i=1}^{n} \int_{V_i} \|\mathbf{p}_i - \mathbf{x}\|_2^2 \; \phi(\mathbf{x}) \; d\mathbf{x}$$

position of robot i

- A Voronoi tessellation becomes a CVT when all generators coincide with the cell centroids.

$$\frac{\partial H(\mathbf{p}_i)}{\partial \mathbf{p}_i} = -\, M_{V_i}(\mathbf{c}_{V_i} - \mathbf{p}_i) = 0$$

Coverage Control

$$\frac{\partial H(\mathbf{p}_i)}{\partial \mathbf{p}_i} = -M_{V_i}(\mathbf{c}_{V_i} - \mathbf{p}_i) = 0$$

- Control strategy for 1st order dynamics:

$$u_i = \dot{\mathbf{p}}_i = k(\mathbf{c}_{V_i} - \mathbf{p}_i)$$

What kind of controller is this?

How to compute centroid positions?

How to compute robot positions in a MRS?

Robot control

Lloyds algorithm

Collaborative localization

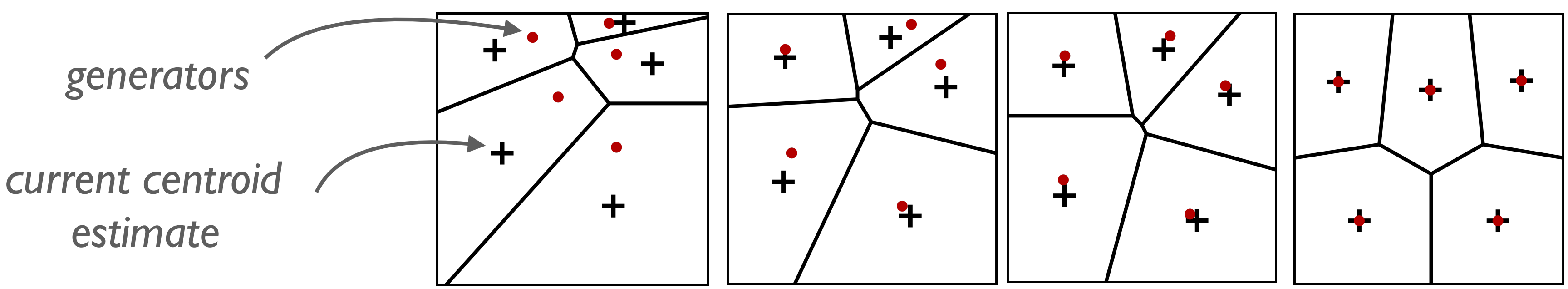
$$u_i = \dot{\mathbf{p}}_i = k(\mathbf{c}_{V_i} - \mathbf{p}_i)$$

How to compute centroid positions?

Lloyds algorithm

Lloyd's Algorithm

- Lloyd's algorithm:
  - Deterministic way of **constructing CVTs**.
  - Iterates over 3 steps:
    1. Construct the Voronoi partition for the generators
    2. Compute the centroids of these regions
    3. Move generators to centroids and start over.



- Convergence of Lloyd's algorithm:
  - A set of points in a given environment converges under the Lloyd algorithm to a centroidal Voronoi configuration. (proof exists)

* image credit: Wikipedia

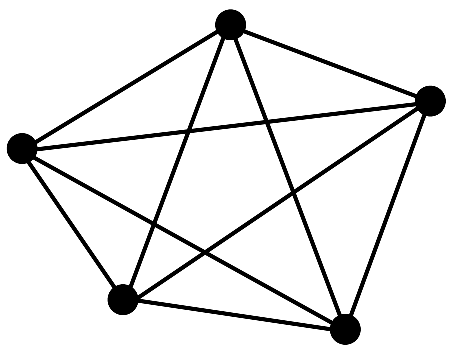
$$u_i = \dot{\mathbf{p}}_i = k(\mathbf{c}_{V_i} - \mathbf{p}_i)$$

How to compute robot positions in a MRS?

Collaborative localization

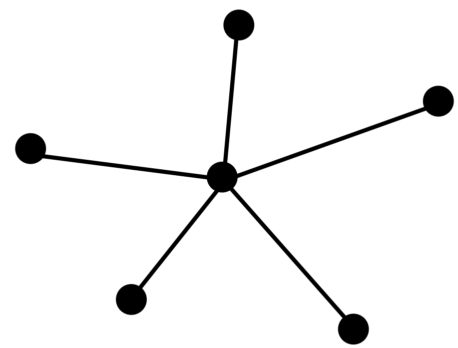
Collaborative Multi-Robot Systems

Communication Topologies:



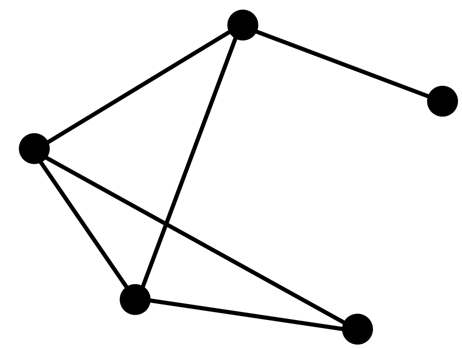
fully connected

centralized / decentralized coordination



star topology

centralized / decentralized coordination



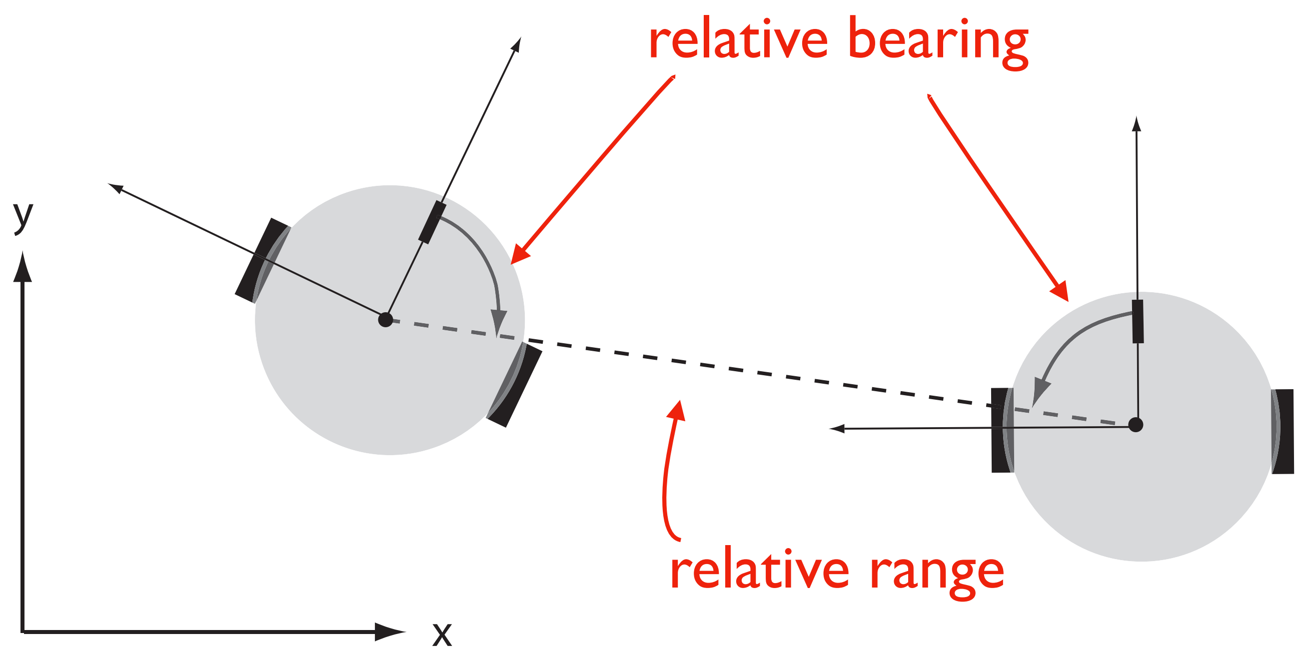
random mesh

decentralized coordination

Distributed Estimation

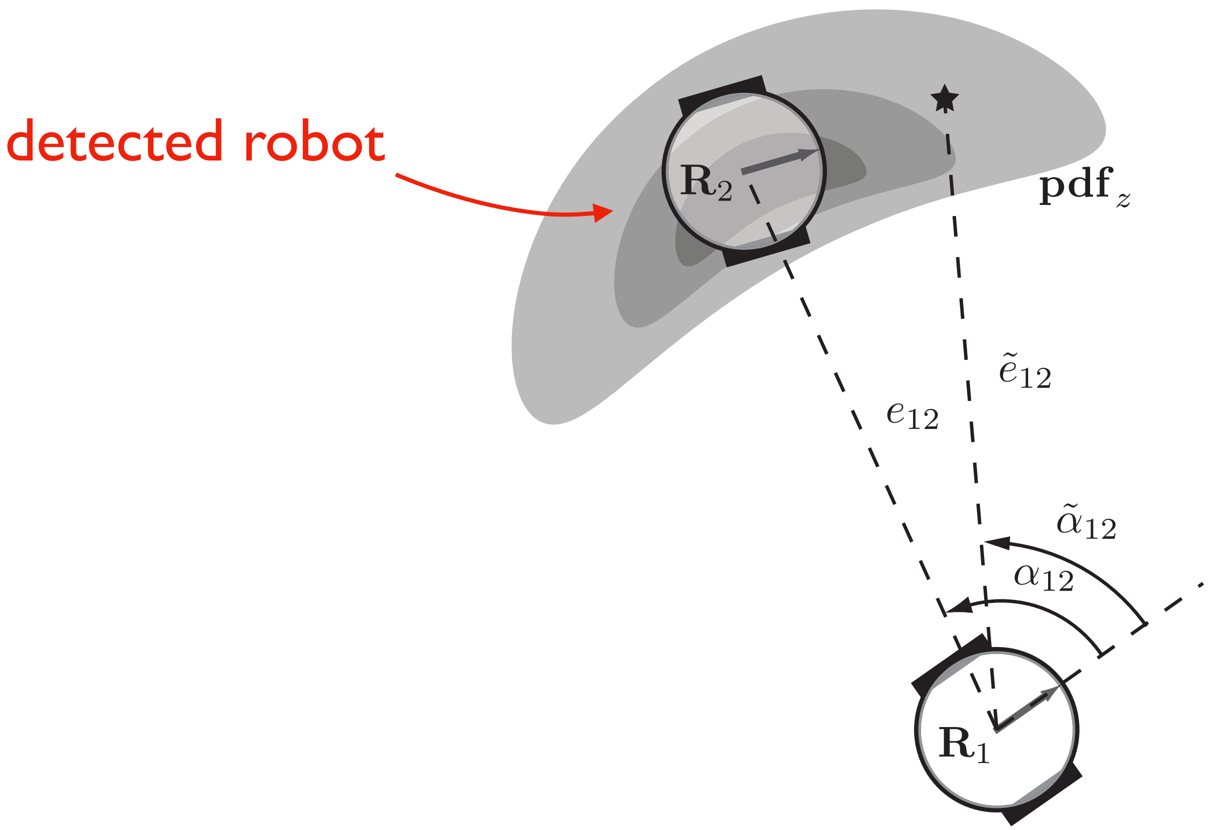
- Goal: Estimate a local or global variable in distributed manner
- Filters can be distributed
  - Examples: Kalman filter, particle filter
  - Method: fuse relative observations of other robots
  - Correct implementation considers relative observations as dependent measurements; the whole history of measurements needs to be tracked (to avoid rumor propagation)!
- Other mechanisms:
  - Opportunistic mechanisms
  - Consensus (agreement mechanism)

Collaborative Localization



- Collaborative localization uses relative inter-robot observations
- Robots communicate their position estimate
- Fuse relative observation by transforming position into local frame

Collaborative Localization

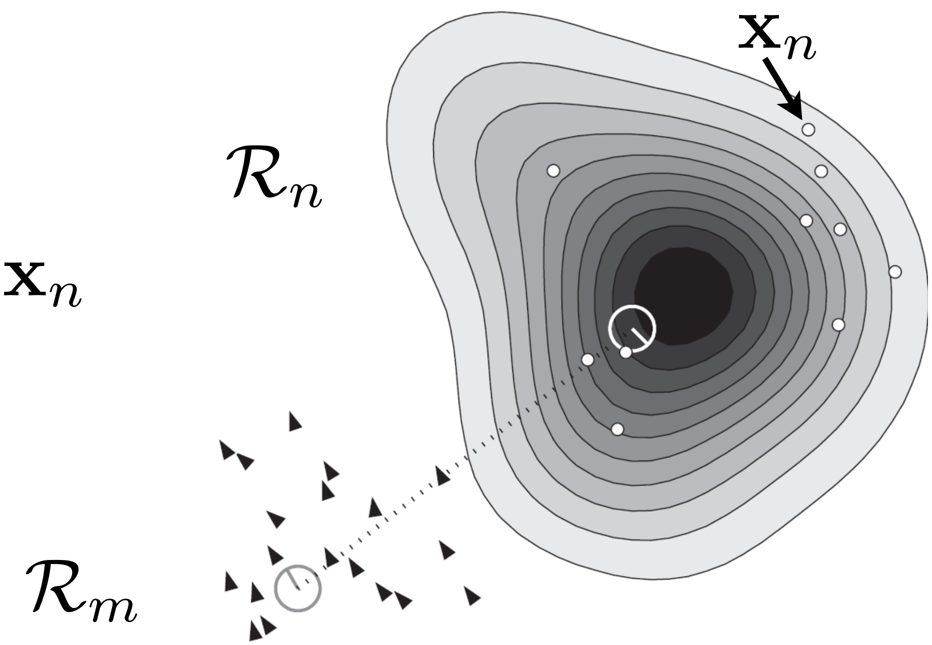


- This example considers a particle filter (Kalman filter also possible)
- Detected robot weights its particles using belief of detecting robot
- Particles re-sampled according to new weights (standard filter)

Range & Bearing Model

$r_{mn}^{[i]}$: range with center $\mathbf{x}_m^{[i]}$ to $\mathbf{x}_n$

$\theta_{mn}^{[i]}$: bearing from $\mathbf{x}_m^{[i]}$ with respect to $\mathbf{x}_n$



detection data

$$d_{mn} = \langle r_{mn}, \theta_{mn}, X_m \rangle$$

$$p(\mathbf{x}_n | d_{mn}) = \eta \cdot \sum_{\left\langle \mathbf{x}_m^{[i]}, w_m^{[i]} \right\rangle \in X_m} \Phi\left(\left[\begin{array}{c} r_{mn}^{[i]} \\ \theta_{mn}^{[i]} \end{array} \right]; \left[\begin{array}{c} r_{mn} \\ \theta_{mn} \end{array} \right], \Sigma \right) \cdot w_m^{[i]}$$

Collaborative Localization Algorithm

Algorithm 1 $\texttt{MultiRob\_Recip\_MCL}(X_{n,t-1}, u_{n,t}, z_{n,t}, D_{n,t})$

$\bar{X}_{n,t} = X_{n,t} = \emptyset$
for $i = 1$ to M **do**
  $\mathbf{x}_{n,t}^{[i]} \leftarrow \texttt{Motion\_Model}(u_{n,t}, \mathbf{x}_{n,t-1}^{[i]})$
  $w_{n,t}^{[i]} \leftarrow \texttt{Measurement\_Model}(\mathbf{x}_{n,t}^{[i]})$
  $w_{n,t}^{[i]} \leftarrow \texttt{Detection\_Model}(D_{n,t}, \mathbf{x}_{n,t}^{[i]}, w_{n,t}^{[i]})$
  $\bar{X}_{n,t} \leftarrow \bar{X}_{n,t} + \left\langle \mathbf{x}_{n,t}^{[i]}, w_{n,t}^{[i]} \right\rangle$
end for
for $i = 1$ to M **do**
  $r \sim \mathcal{U}(0, 1)$
  if $r \leq (1 - \alpha)$ **then**
    $\mathbf{x}_{n,t}^{[i]} \leftarrow \texttt{Sampling}(\bar{X}_{n,t})$
  else
    $\mathbf{x}_{n,t}^{[i]} \leftarrow \texttt{Reciprocal\_Sampling}(D_{n,t}, \bar{X}_{n,t})$
  end if
  $X_{n,t} \leftarrow X_{n,t} + \left\langle \mathbf{x}_{n,t}^{[i]}, w_{n,t}^{[i]} \right\rangle$
end for
return $X_{n,t}$

[Prorok et al., 2011]

Collaborative Localization

UNIVERSITY OF CAMBRIDGE

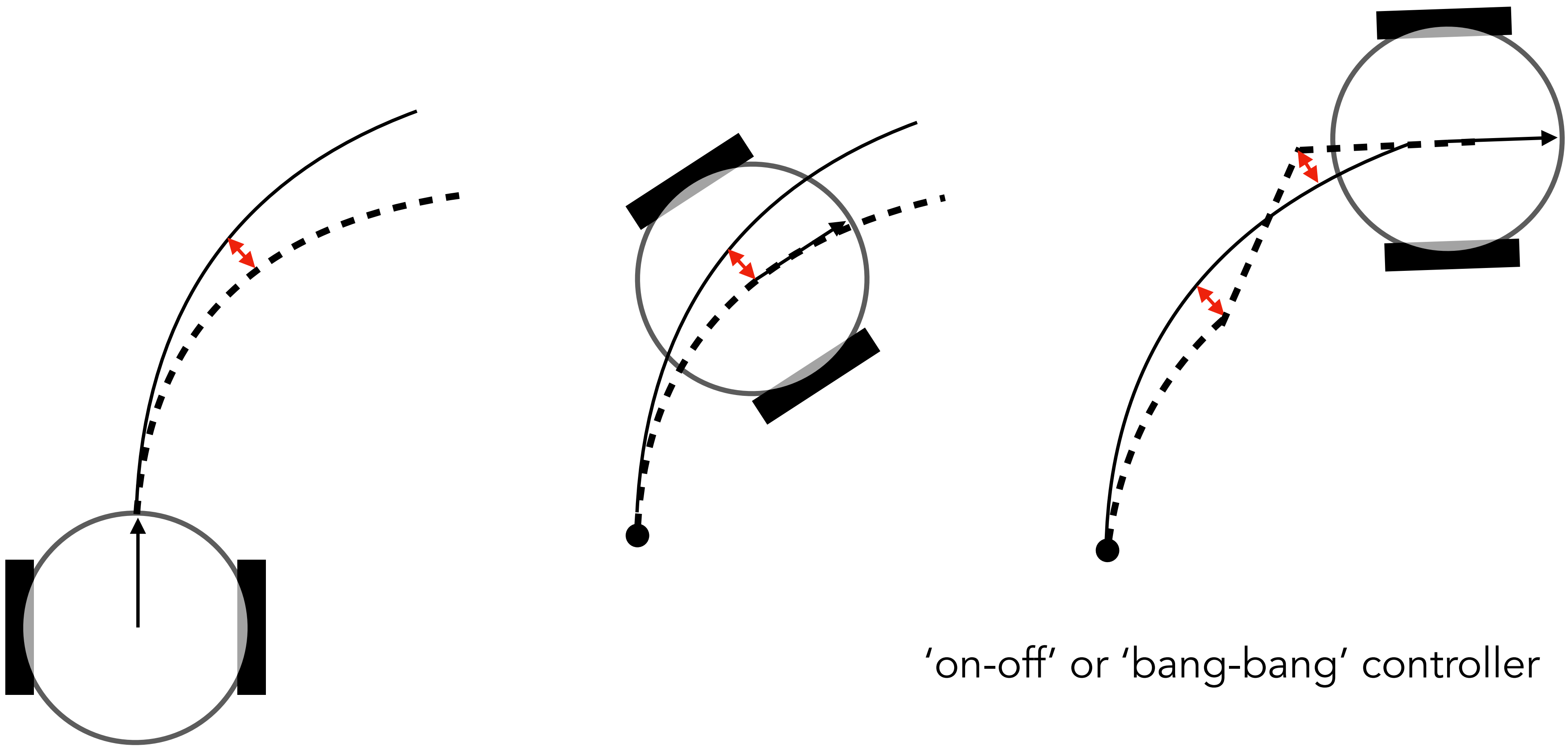
$$u_i = \dot{\mathbf{p}}_i = k(\mathbf{c}_{V_i} - \mathbf{p}_i)$$

What kind of controller is this?

Robot control

Control

- Goal: reach desired position / follow desired trajectory
- Example: trajectory tracking
- Assumption: robot receives **feedback** on distance to desired trajectory.



'on-off' or 'bang-bang' controller

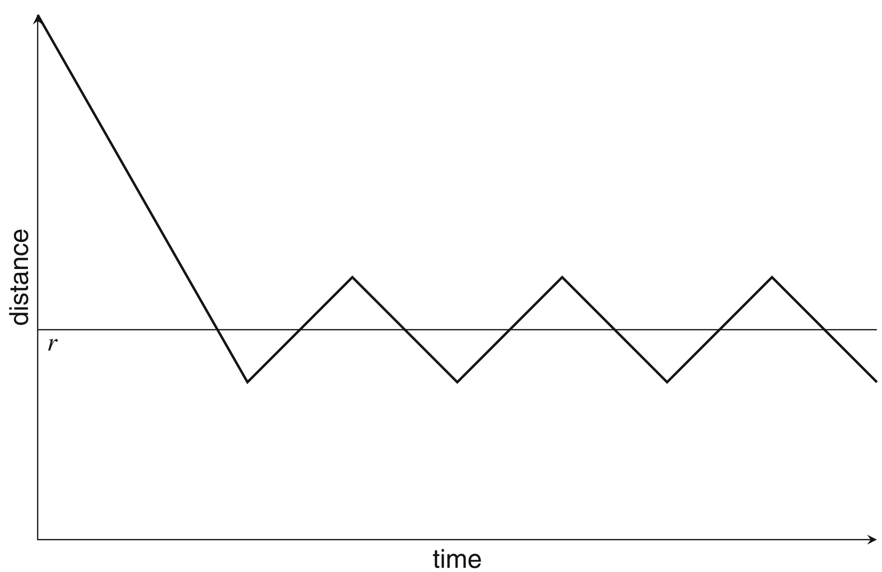
Control

A Simple Closed-Loop Controller:

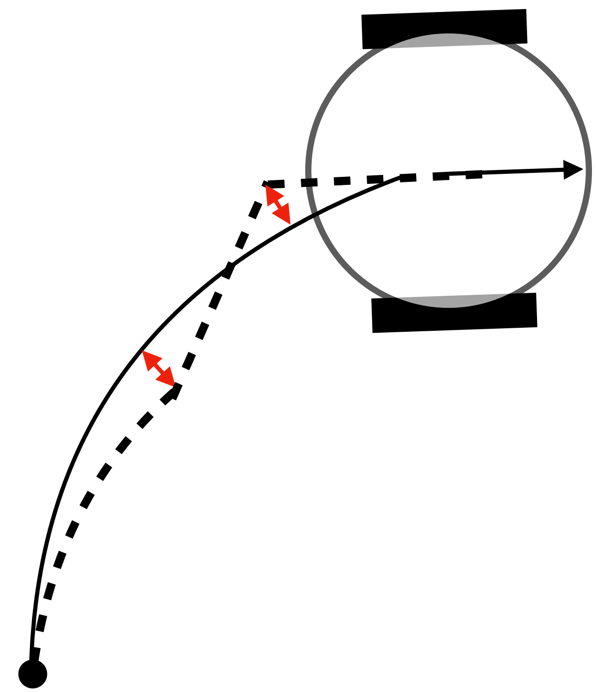
```
Algorithm: Bang-Bang Controller

forever do:
     error ← reference − measured // Distance
     if error < 0                   // Too far left
      left-motor-power ← 100
      right-motor-power ← -100
     if error > 0                   // Too far right
      left-motor-power ← −100
      right-motor-power ← 100
     if error = 0                   // Just right
      left-motor-power ← 100
      right-motor-power ← 100
```

Bang-Bang Controller

- Example: trajectory tracking
- Assumption: robot receives feedback on distance to desired trajectory.



zig-zag behavior: we can do better!

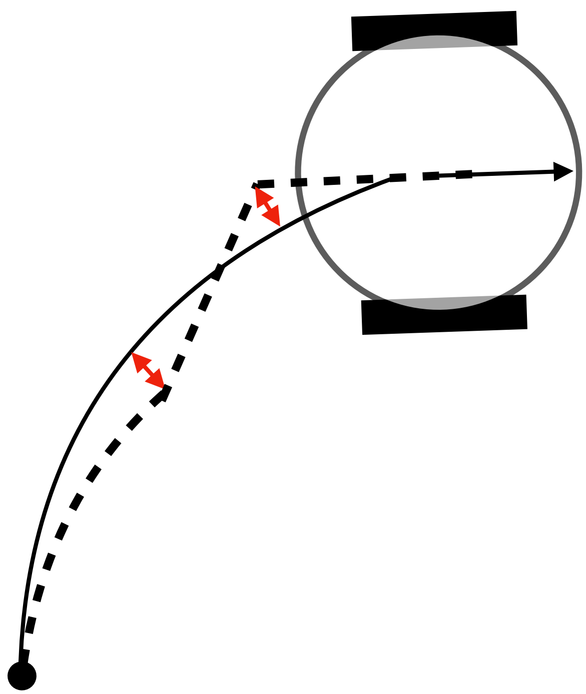


'on-off' or 'bang-bang' controller

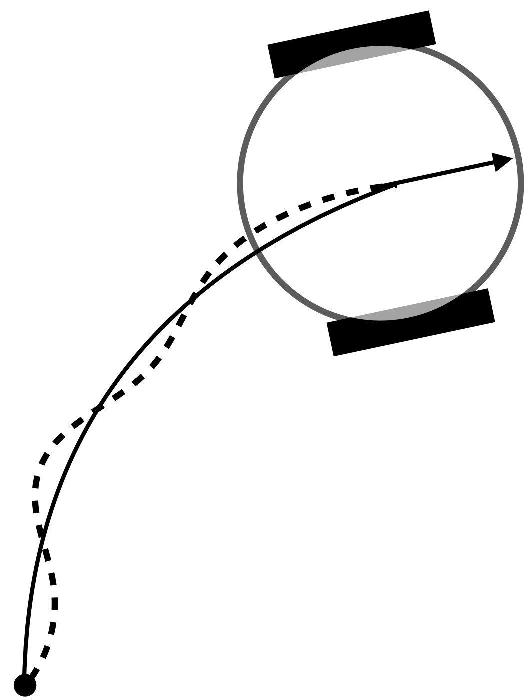
* image credit: Elements of Robotics

Proportional Control (P-Control)

- Example: trajectory tracking
- Assumption: robot receives feedback on distance to line.
- Robot computes **error,** and **adjusts** control as a function of error



previous slide: oscillatory behavior



adjustment is proportional to error!

error = distance-to-trajectory

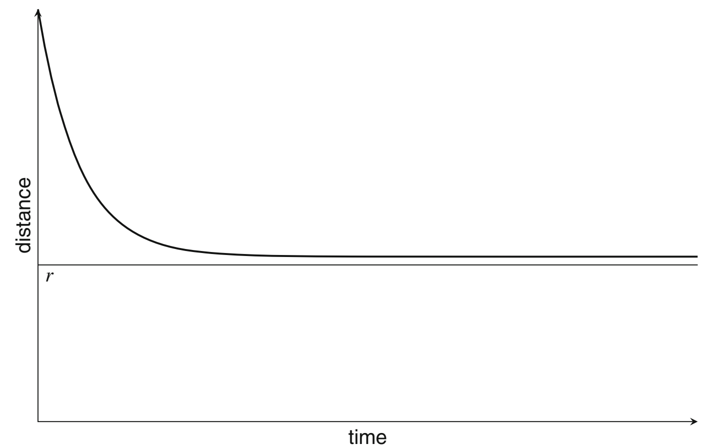
turning-control = K * error

Proportional Control (P-Control)

```
Algorithm: P-Controller

forever do:
     error ← reference – measured  // Distance
     power ← gain * error          // Control value
     left-motor-power ← power_left
     right-motor-power ← power_right
```

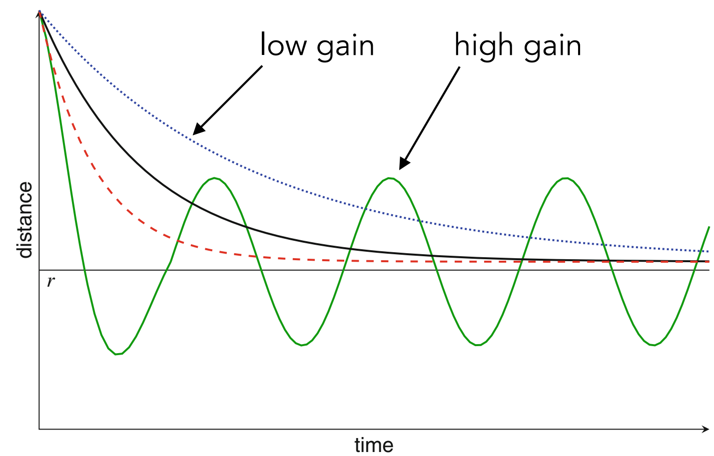
Proportional Control (P-Control)

- Behavior of P-control:
  - Adapt control proportionally to your perceived error to set-point.
  - $u(t) = \kappa_p e(t)$



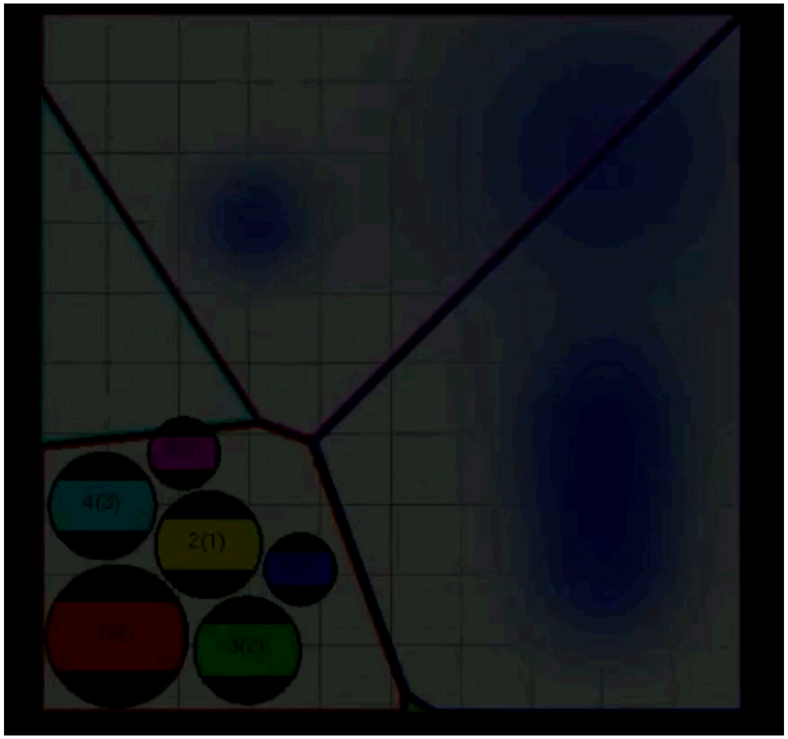
- Why is the target distance not reached?
  - Methods to overcome this:
    PID control (advanced)

- Behavior for varying gain values
- High gains not desirable! We call this an *unstable* controller.



* image credit: Elements of Robotics

UNIVERSITY OF CAMBRIDGE



$$u_i = \dot{\mathbf{p}}_i = k(\mathbf{c}_{V_i} - \mathbf{p}_i)$$

What kind of controller is this?

Robot control

How to compute centroid positions?

Lloyds algorithm

How to compute robot positions in a MRS?

Collaborative localization

UNIVERSITY OF CAMBRIDGE

* movie credit: Omur Arlsan; 2018

Further Reading

Fundamental concepts:

- Elements of Robotics, F Mondada et al., 2018
- Autonomous Mobile Robots, R Siegwart et al., 2004

State of the art:

- Springer Handbook of Robotics — library has a copy!
- The grand challenges of Science Robotics, *Science*, Yang et al. 2018

Further reading:

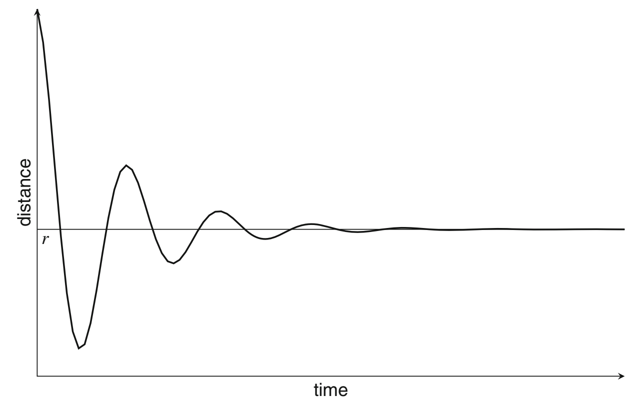
- Probabilistic Robotics, S Thrun et al, 2005
- Springer Handbook of Robotics, B Siciliano et al., 2008

PID Control (Advanced)

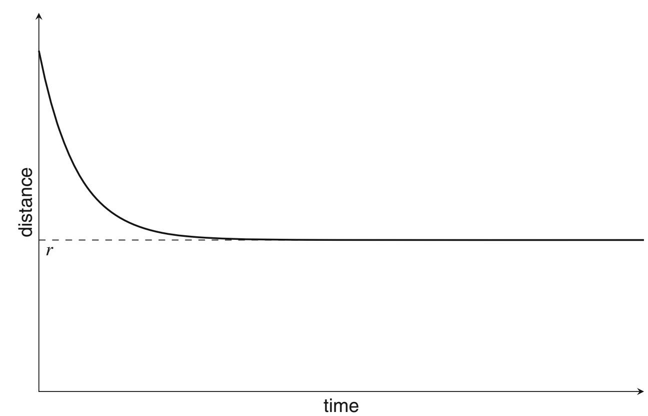
- PI-controller:
  - takes into account **accumulated error** over time

$$u(t) = \kappa_p e(t) + \kappa_i \int_0^t e(\tau)\, d\tau$$

  - E.g., in presence of friction, error will be integrated causing higher motor setting to overcome remaining delta.



- PID-controller:
  - take into account **future error** by computing rate of change of error.
  - acts as a *'dampener'* on control effort.



* image credit: Elements of Robotics