Autonomous agents
Lecture 4, 20160128

Simulation of autonomous robots

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today’s learning goals

- After this lecture you should be able to
  - Explain the concept of event timing in simulations
  - Describe how noise is added to sensor readings in robot simulations
  - Describe, and derive equations for, ray readings in ray-based simulated sensors
  - Describe how ray readings are interpreted and used in different sensors
  - Define the equations for integrating the dynamic equations for robot motion

Robot simulations: Motivation

- Simulation are important for …
  - …testing before construction, to avoid catastrophic failures.
  - …testing before construction, to save money (and time).
  - …optimization – can be fast in simulation, but slow and difficult (components that break, constant monitoring required etc.) in real robots.
- However, any results obtained in simulation must be tested in real robots – no simulation is perfect.

Simulation flow

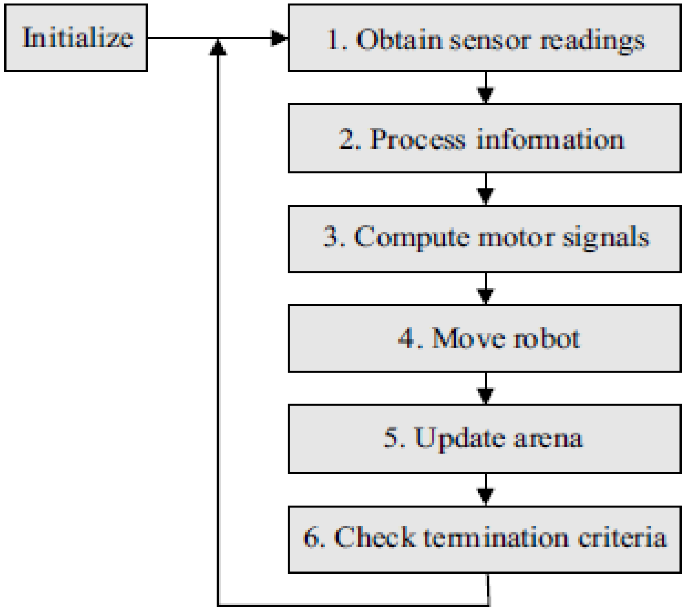


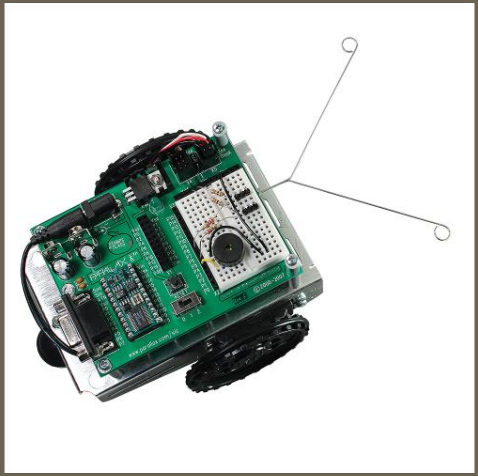
Figure 3.1: *The flow of a single-robot simulation. Steps 1 through 6 are carried out in each time step of the simulation.*

Event timing

- Simulation results should at least be *possible* to transfer to a real robot.
- Types of events (in a simulation)
  1. Events that are slow in simulation, but fast in a real robot
     - Example: Collision-checking (see the following slides).
  2. Events that are fast in simulation, but slow in a real robot
     - (Some types) of sensing.

Event timing example: Collision-checking

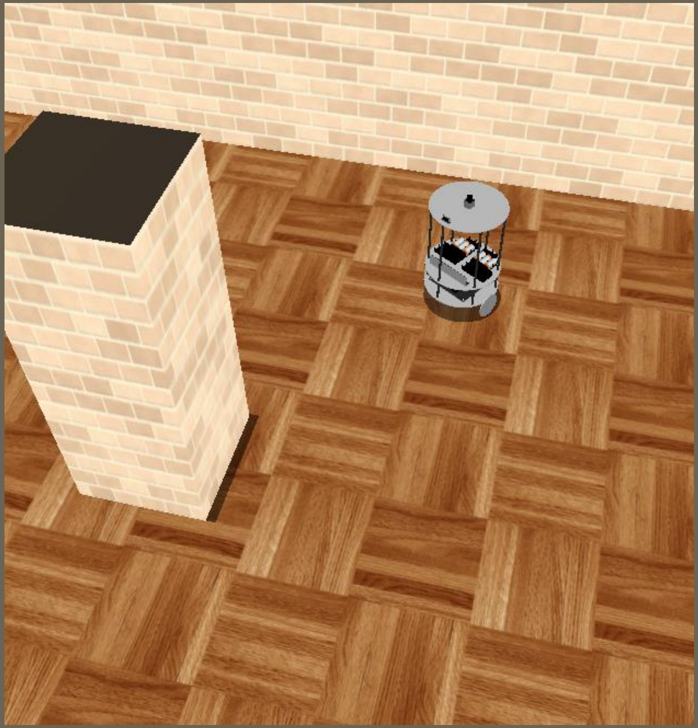
- In a real robot, this type of sensor can be read off fast (single bit of information from a collision sensor!)
- Example: Boe-Bot with whiskers:



Event timing example: Collision-checking

- In simulation, collision-checking can be quite complex, if the robot (or the obstacles) has a complicated geometry.
- The process is generally speeded up by the use of (i) a grid and (ii) a simplified collision geometry (see the following slides)
- Still, the process can be rather time-consuming.

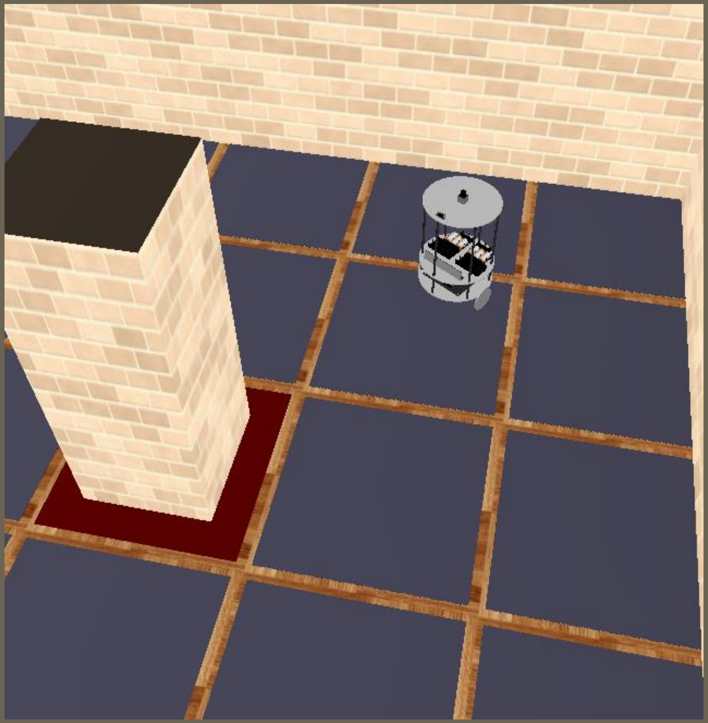
Event timing example: Collision-checking



Event timing example: Collision-checking

The simulator checks for collisions between the robot and any obstacles partially covering the red cells.

Event timing example: Collision-checking



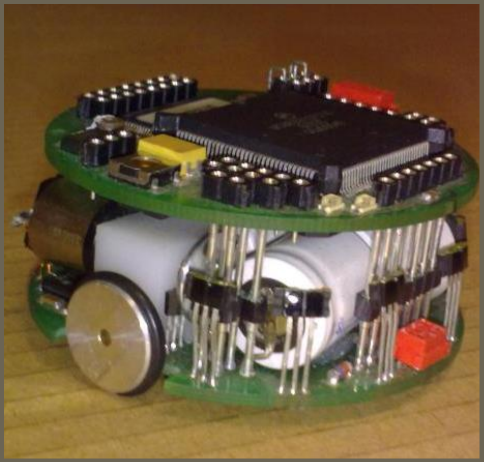
In this example, collisions with the column (in the red cell) need not be checked …

Event timing example: Collision-checking

Simplified collision geometry. In this case, around 16 planes instead of hundreds of irregular objects.

Event timing example: IR sensors

- In simulation, reading a simple IR sensor can be quite fast (as explained further below).
- In a robot, it might take more time, for example due to limits on data transfer.
- Specific example: A Khepera robot, with 8 IR sensors that are read sequentially.
- Each sensor reading takes 2.5 ms => 8 x 2.5 = 20 ms = 0.02 s in total.



Event timing example: IR sensors

- Of course, another kind of robot may be able to read this particular sensor type much faster.
- However, the point still remains: Some sensors do take quite a bit of time to read (e.g. LRFs), for example due to hardware limitations.
- In the simulator, one can introduce a **readability state**, only reading the sensor at some (not all) time steps.

Timing diagram

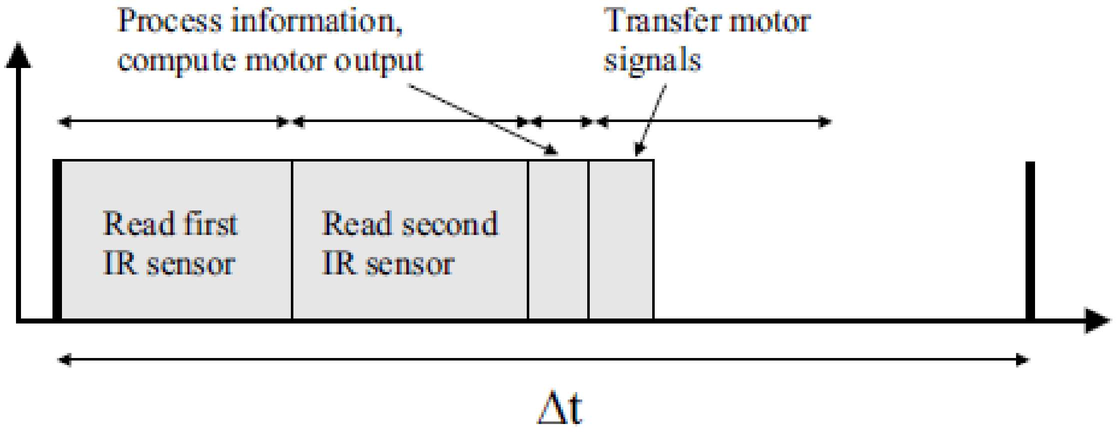


Figure 3.2: *A timing diagram. The boxes indicate the time required to complete the corresponding event in hardware, i.e. a real robot. In order for the simulation to be realistic, the time step Δt used in the simulation must be longer than the total duration (in hardware) of all events taking place within a time step.*

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today's learning goals

- After this lecture you should be able to
  - Explain the concept of event timing in simulations ✓
  - Describe how noise is added to sensor readings in robot simulations
  - Describe, and derive equations for, ray readings in ray-based simulated sensors
  - Describe how ray readings are interpreted and used in different sensors
  - Define the equations for integrating the dynamic equations for robot motion

Noise

- Real sensors are always noisy (and can only be updated with limited frequency).
- Moreover, even two supposedly identical sensors will not give identical readings all the time.
- Two ways of adding noise
  1. Using a model (e.g. Gaussian)

$$\hat{S} = SN(1,\sigma)$$

  2. Measuring a real sensor and storing the results in a lookup table (possible for simple sensors).

Noise

- Method 2 (example):
  - IR sensor, range 0.10 - 0.50 m. Measure k samples at 0.10 m, 0.15 m, ... 0.50 m.
  - In the simulation, if the distance to an obstacles is, say, 0.23 m, pick one sample ($\hat{s}_{20}$) from 0.20 m and one ($\hat{s}_{25}$) from 0.25 m, and interpolate: $\hat{S} = \hat{s}_{20} + \frac{0.23-0.20}{0.25-0.20}(\hat{s}_{25} - \hat{s}_{20})$
- Normally, Method 1 is used, however.

Today's learning goals

- After this lecture you should be able to
  - Explain the concept of event timing in simulations ✓
  - Describe how noise is added to sensor readings in robot simulations ✓
  - Describe, and derive equations for, ray readings in ray-based simulated sensors
  - Describe how ray readings are interpreted and used in different sensors
  - Define the equations for integrating the dynamic equations for robot motion

Simulated sensors

- Important: A simulated robot must only be given information that would be obtained (in a real robot) from the available sensors.
- For example, a simulated robot may have access to (noisy) odometric readings, but not its exact position!
- Many simulated sensors (involving light or sound) are **ray-based**.

Ray-based simulated sensors

- These sensors use ray tracing:
  - Send out a ray (a line) from the sensor.
  - Check for intersection between the ray and objects in the arena.
- In ARSim, objects are built from (2D) lines.
- Here, consider only the 2D case:
  - All surfaces have a vertical extension
  - All rays are parallel to the ground.
- Even in more complex simulators (e.g. GPRSim), some sensors are effectively two-dimensional.

Ray-based simulated sensors

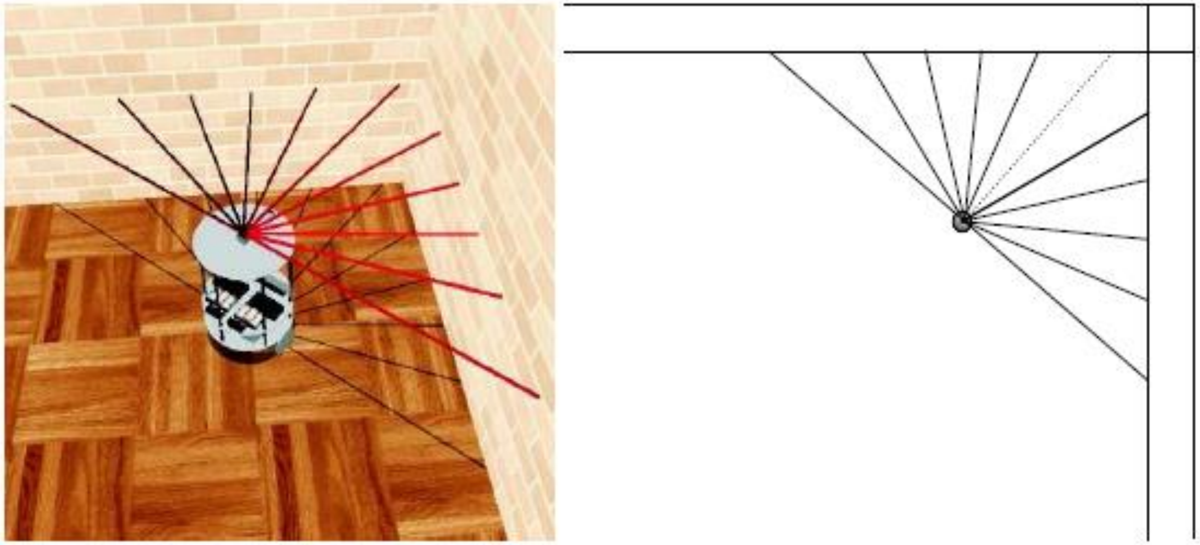
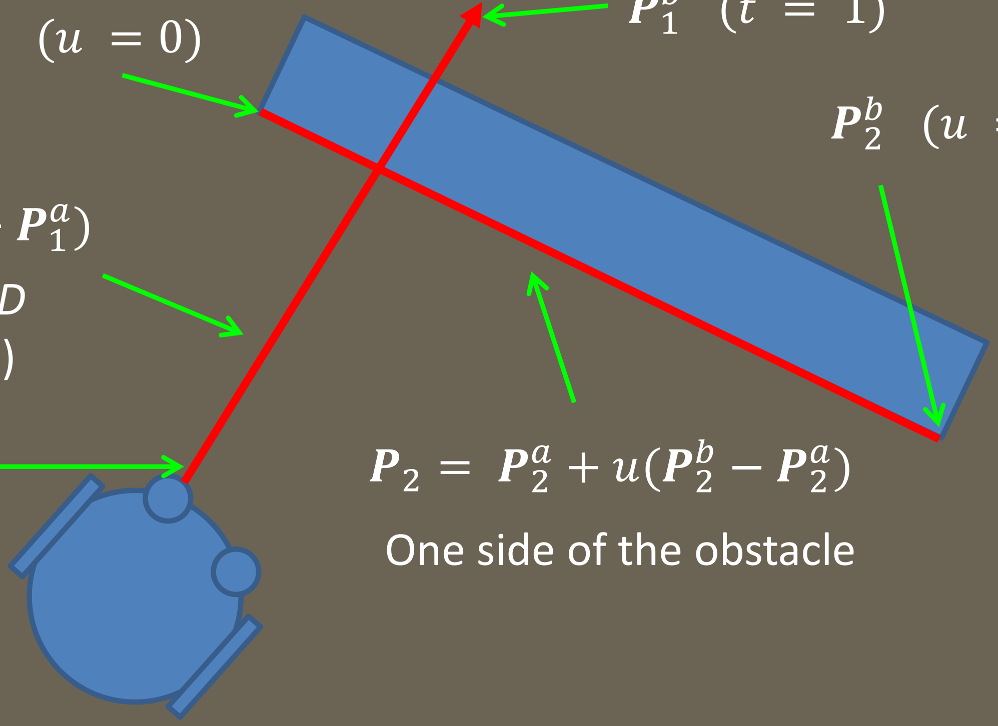


Figure 3.3: *Left panel: A screenshot from GPRSim, showing an LRF taking a reading in an arena containing only walls. Right panel: A two-dimensional representation of the sensor reading. The dotted ray points in the forward direction of the robot which, in this case, coincides with the forward direction of the LRF.*

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Obtaining ray readings



$P_2^a \quad (u = 0)$

$P_1^b \quad (t = 1)$

$P_2^b \quad (u = 1)$

$$P_1 = P_1^a + t(P_1^b - P_1^a)$$

Sensor ray: Length *D*
(= the sensor range)

$P_1^a \quad (t = 0)$

$$P_2 = P_2^a + u(P_2^b - P_2^a)$$

One side of the obstacle

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Obtaining ray readings

- Intersection checking is carried out by solving the equation $\boldsymbol{P}_1 = \boldsymbol{P}_2$.
- The expressions for t and u become

$$t = \frac{(x_2^{\mathrm{b}} - x_2^{\mathrm{a}})(y_1^{\mathrm{a}} - y_2^{\mathrm{a}}) - (y_2^{\mathrm{b}} - y_2^{\mathrm{a}})(x_1^{\mathrm{a}} - x_2^{\mathrm{a}})}{(y_2^{\mathrm{b}} - y_2^{\mathrm{a}})(x_1^{\mathrm{b}} - x_1^{\mathrm{a}}) - (x_2^{\mathrm{b}} - x_2^{\mathrm{a}})(y_1^{\mathrm{b}} - y_1^{\mathrm{a}})}$$

$$u = \frac{(x_1^{\mathrm{b}} - x_1^{\mathrm{a}})(y_1^{\mathrm{a}} - y_2^{\mathrm{a}}) - (y_1^{\mathrm{b}} - y_1^{\mathrm{a}})(x_1^{\mathrm{a}} - x_2^{\mathrm{a}})}{(y_2^{\mathrm{b}} - y_2^{\mathrm{a}})(x_1^{\mathrm{b}} - x_1^{\mathrm{a}}) - (x_2^{\mathrm{b}} - x_2^{\mathrm{a}})(y_1^{\mathrm{b}} - y_1^{\mathrm{a}})}$$

- An intersection occurs if $t \in [0,1], u \in [0,1]$.

Obtaining ray readings

- Thus, to determine a ray reading:
  1. Find the starting point $\boldsymbol{P}_1^a$
  2. Find the end point $\boldsymbol{P}_1^b = (x_a + D\cos\beta_i\,, y_a + D\sin\beta_i)$, where β_i is the direction of the ray.
  3. Determine the intersections with all relevant arena object lines.
  4. The ray reading d_i is the shortest distance (= the distance to the nearest obstacle) thus obtained.

Obtaining ray readings

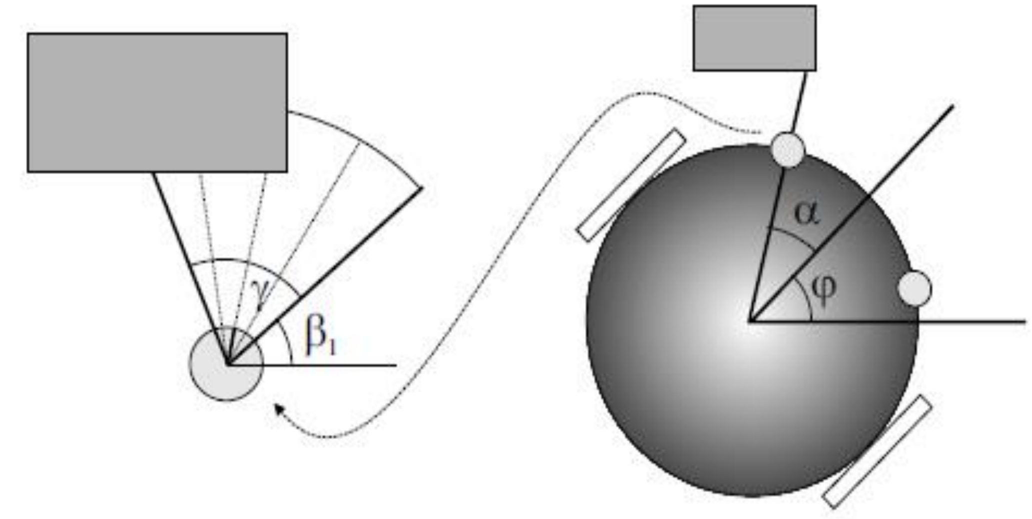


Figure 3.4: *The right panel shows a robot equipped with two IR sensors, and the left panel shows a blow-up of the left sensor. In this case, the number of rays (N) was equal to 5. The leftmost and rightmost rays, which also indicate the opening angle γ of the IR sensor are shown as solid lines, whereas the three intermediate rays are shown as dotted lines.*

Mattias Wahde, PhD, Professor, Chalmers University of Technology
e-mail: mattias.wahde@chalmers.se, http://www.me.chalmers.se/~mwahde

Today’s learning goals

- After this lecture you should be able to
  - Explain the concept of event timing in simulations ✓
  - Describe how noise is added to sensor readings in robot simulations ✓
  - Describe, and derive equations for, ray readings in ray-based simulated sensors ✓
  - Describe how ray readings are interpreted and used in different sensors
  - Define the equations for integrating the dynamic equations for robot motion

Using ray readings

- IR sensors:
  - These sensors give a fuzzy reading.
  - The sensor rays *have no physical counterpart*. They are only used as a computational tool to obtain the reading S.

$$S = \frac{1}{N}\sum_{i=1}^{N} \rho_i$$

  - where ...

$$\rho_i = \min\left(\left(\frac{c_1}{d_i^2} + c_2\right)\cos\kappa_i\,, 1\right)$$

Using ray readings

- Sonar sensors:
  - Again the sensor rays are only used as a computational tool to obtain the reading S.
  - Set $S = \min_i d_i$, with probability p (close to 1), and $S = D \equiv D_{\max}$ (no reading) with probability $1 - p$.
  - If $S < D_{\min}$, set $S = D_{\min}$.

Using ray readings

- Laser range finders (LRFs):
  - Give a vector-valued reading.
  - Here, the rays do correspond to the actual laser beams.
  - If $d_i > D$, set the ray reading to -1 (to indicate that no reading was obtained from that particular ray).

Today's learning goals

- After this lecture you should be able to
  - Explain the concept of event timing in simulations ✓
  - Describe how noise is added to sensor readings in robot simulations ✓
  - Describe, and derive equations for, ray readings in ray-based simulated sensors ✓
  - Describe how ray readings are interpreted and used in different sensors ✓
  - Define the equations for integrating the dynamic equations for robot motion

Robot motion (in simulation)

- To move the robot forward, use Eqs. (2.31) and (2.32) to obtain $\dot{V}$ and $\ddot{\varphi}$.
- The new values of the speed and angular speed are then obtained as

$$V' = V + \dot{V}\Delta t$$
$$\dot{\varphi}' = \dot{\varphi} + \ddot{\varphi}\Delta t$$

- The new heading is then obtained as

$$\varphi' = \varphi + \dot{\varphi}'\Delta t$$

Robot motion (in simulation)

- The components of the robot’s velocity can then be computed as...

$$V'_x = V' \cos\varphi$$
$$V'_y = V' \sin\varphi$$

- ...so that the new positions can be obtained

$$X' = X + V'_x \Delta t$$
$$Y' = Y + V'_y \Delta t$$

Today's learning goals

- After this lecture you should be able to
  - Explain the concept of event timing in simulations ✓
  - Describe how noise is added to sensor readings in robot simulations ✓
  - Describe, and derive equations for, ray readings in ray-based simulated sensors ✓
  - Describe how ray readings are interpreted and used in different sensors ✓
  - Define the equations for integrating the dynamic equations for robot motion ✓

ARSim