COURSE Review 2
MOBILE ROBOTICS
CENG 5437-01, CENG 4391-02 SPRING 2022

Wheeled Robots and Differential Drive Steering Basic Kinematics and Math Basic
Turtlebot and Gazebo
Python program – Move Turtlebot.
Physics of Wheeled Robots

SENSORS Classes of Sensors Characterization of Sensors Types of Errors – Statistical and Random

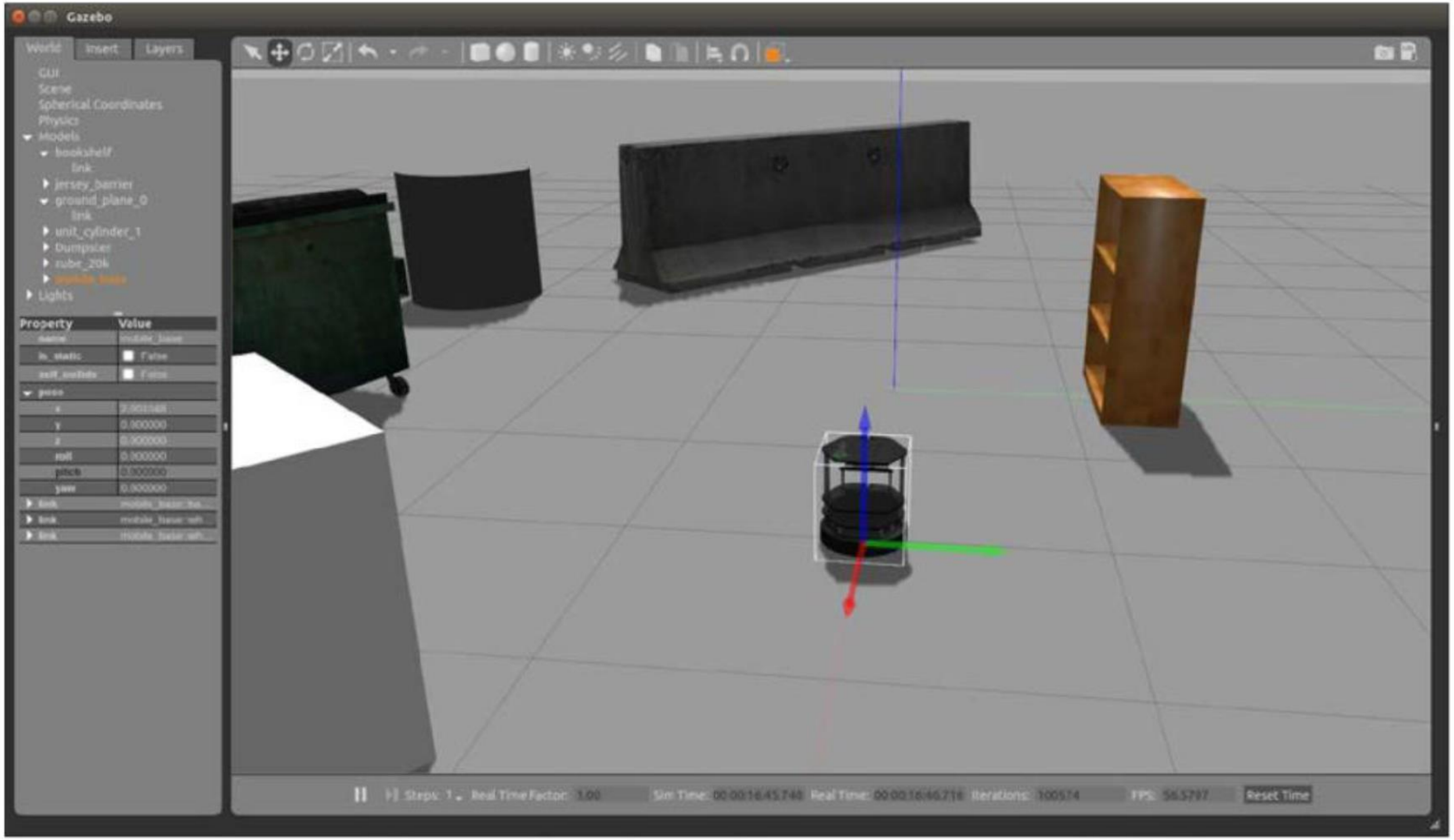
Specific Sensors Position- Absolute and Relative Range

Connect Sensors to the Computer – A2D converters, etc.

PG 84 Text Pose of a Turtlebot Robot and its Topics (Sensors, etc.)

DIFFERENTIAL DRIVE ROBOT

Turtlebot 2



TurtleBot simulated with axes shown

```
$ rosservice call gazebo/get_model_state '{model_name: mobile_base}'
```

The output of the preceding command is similar to the following if TurtleBot is at the origin:

```
pose:
 position:
  x: 0.00161336508139
  y: 0.0091790167961
  z: -0.00113098620237
 orientation:
  x: -5.20108036968e-05
  y: -0.00399736084462
  z: -0.0191615228716
  w: 0.999808408868
 twist:
  linear:
   x: 9.00012388429e-06
   y: 6.54279879125e-05
   z: -1.4365465304e-05
  angular:
   x: -0.000449167550145
   y: 0.000197996689198
   z: -0.000470014447946
success: True
status_message: GetModelState: got properties
```

With ROS commands, you can move the TurtleBot, as we did with the turtle in Turtlesim in *Chapter 1, Getting Started with ROS*. First, find the topic that will control the `mobile_base` link since that is the name given in Gazebo's left panel:

```
$ rostopic list | grep mobile_base
```

The output is as follows:

```
/mobile_base/commands/motor_power
/mobile_base/commands/reset_odometry
/mobile_base/commands/velocity
/mobile_base/events/bumper
/mobile_base/events/cliff
/mobile_base/sensors/bumper_pointcloud
/mobile_base/sensors/core
/mobile_base/sensors/imu_data
/mobile_base_nodelet_manager/bond
```

To cause the robot to turn in a circle requires some forward velocity and angular velocity, which the following command shows:

```
$ rostopic pub -r 10 /mobile_base/commands/velocity \geometry_msgs/Twist
'{linear: {x: 0.2}, angular: {x: 0, y: 0, z: 1.0}}'
```

The linear speed is 0.2 meters/second and the rotation is 1.0 radian (about 57 degrees) per second.

To view the messages sent, type the following command in a separate terminal window:

```
$ rostopic echo /mobile_base/commands/velocity
```

DIRECT MESSAGE ON NETWORK TO TURTLEBOT

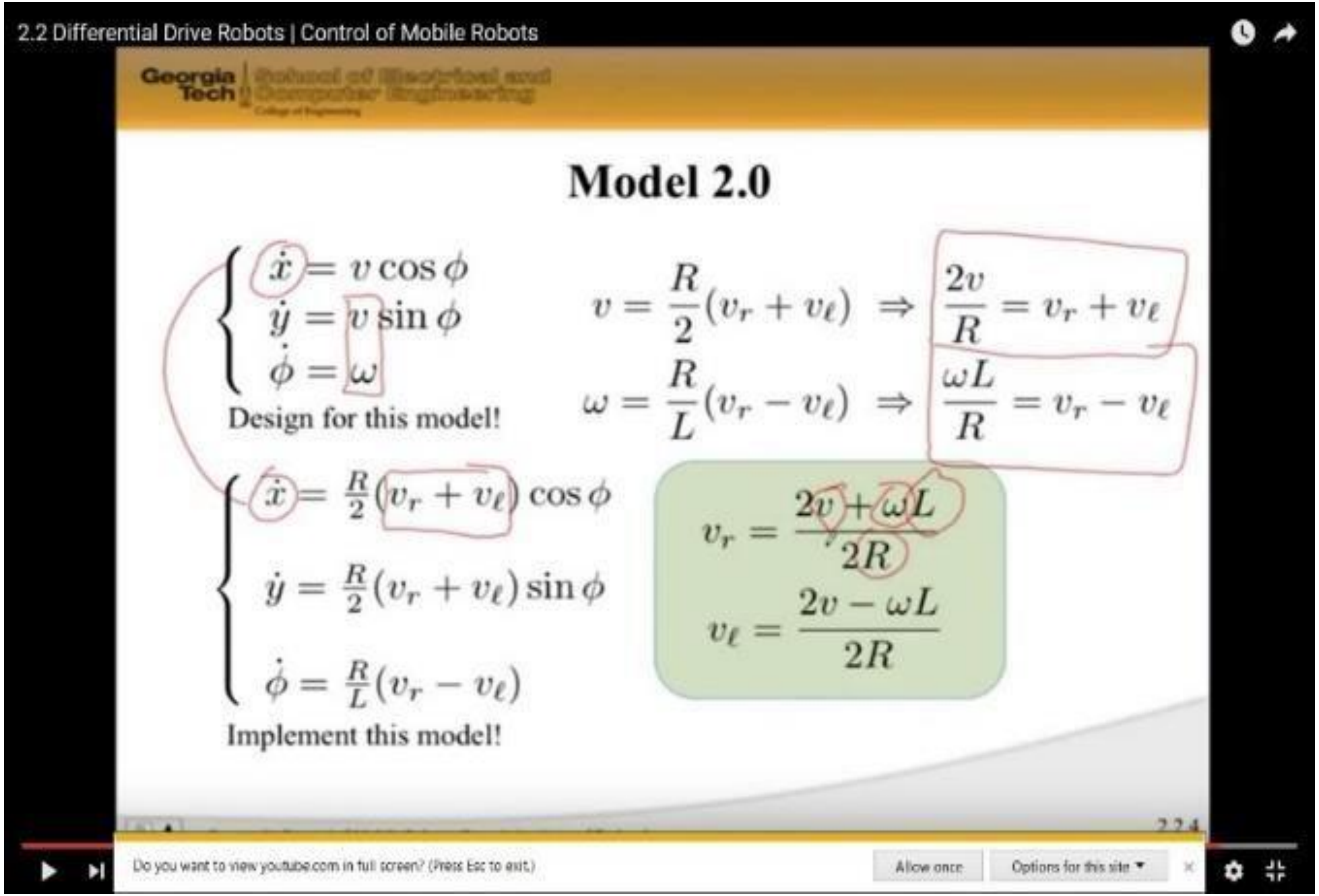
```
#!/usr/bin/env python
# Execute as a python script
# Set linear and angular values of TurtleBot's speed and turning.
import rospy      # Needed to create a ROS node
from geometry_msgs.msg import Twist    # Message that moves base

classControlTurtleBot():
  def __init__(self):
    # ControlTurtleBot is the name of the node sent to the #master
    rospy.init_node('ControlTurtleBot', anonymous=False)

    # Message to screen
    rospy.loginfo("Press CTRL+c to stop TurtleBot")

    # Keys CNTL + c will stop script
    rospy.on_shutdown(self.shutdown)

    # Publisher will send Twist message on topic
    # cmd_vel_mux/input/navi

    self.cmd_vel = rospy.Publisher('cmd_vel_mux/input/navi',
    Twist, queue_size=10)
```

RATE

```
        # TurtleBot will receive the message 10 times per second.
        rate = rospy.Rate(10);
        # 10 Hz is fine as long as the processing does not exceed
        #    1/10 second.

        # Twist is geometry_msgs for linear and angular velocity
        move_cmd = Twist()
        # Linear speed in x in meters/second is + (forward) or -
        # (backwards)
        move_cmd.linear.x = 0.3
        # Modify this value to change speed
        # Turn at 0 radians/s
        move_cmd.angular.z = 0
        # Modify this value to cause rotation rad/s

        # Loop and TurtleBot will move until you type CNTL+c
        while not rospy.is_shutdown():
            # publish Twist values to TurtleBot node /cmd_vel_mux
            self.cmd_vel.publish(move_cmd)
            # wait for 0.1 seconds (10 HZ) and publish again
            rate.sleep()


    def shutdown(self):
        # You can stop turtlebot by publishing an empty Twist
        # message
        rospy.loginfo("Stopping TurtleBot")

        self.cmd_vel.publish(Twist())
        # Give TurtleBot time to stop
        rospy.sleep(1)

if __name__ == '__main__':
    try:
        ControlTurtleBot()
    except:
        rospy.loginfo("End of the trip for TurtleBot")
```

x, ω

CONTROL – WHEEL VELOCITY, ROTATION



MAGNUS

https://www.youtube.com/watch?app=desktop&v=aE7RQNhwnPQ&sns=em

Sensors

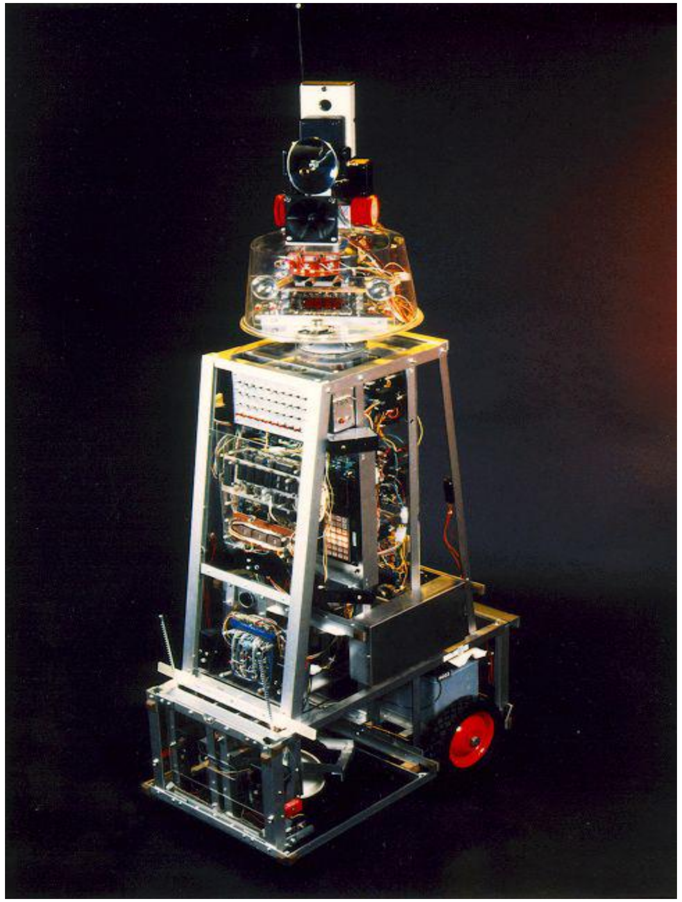
- **Proprioceptive Sensors**

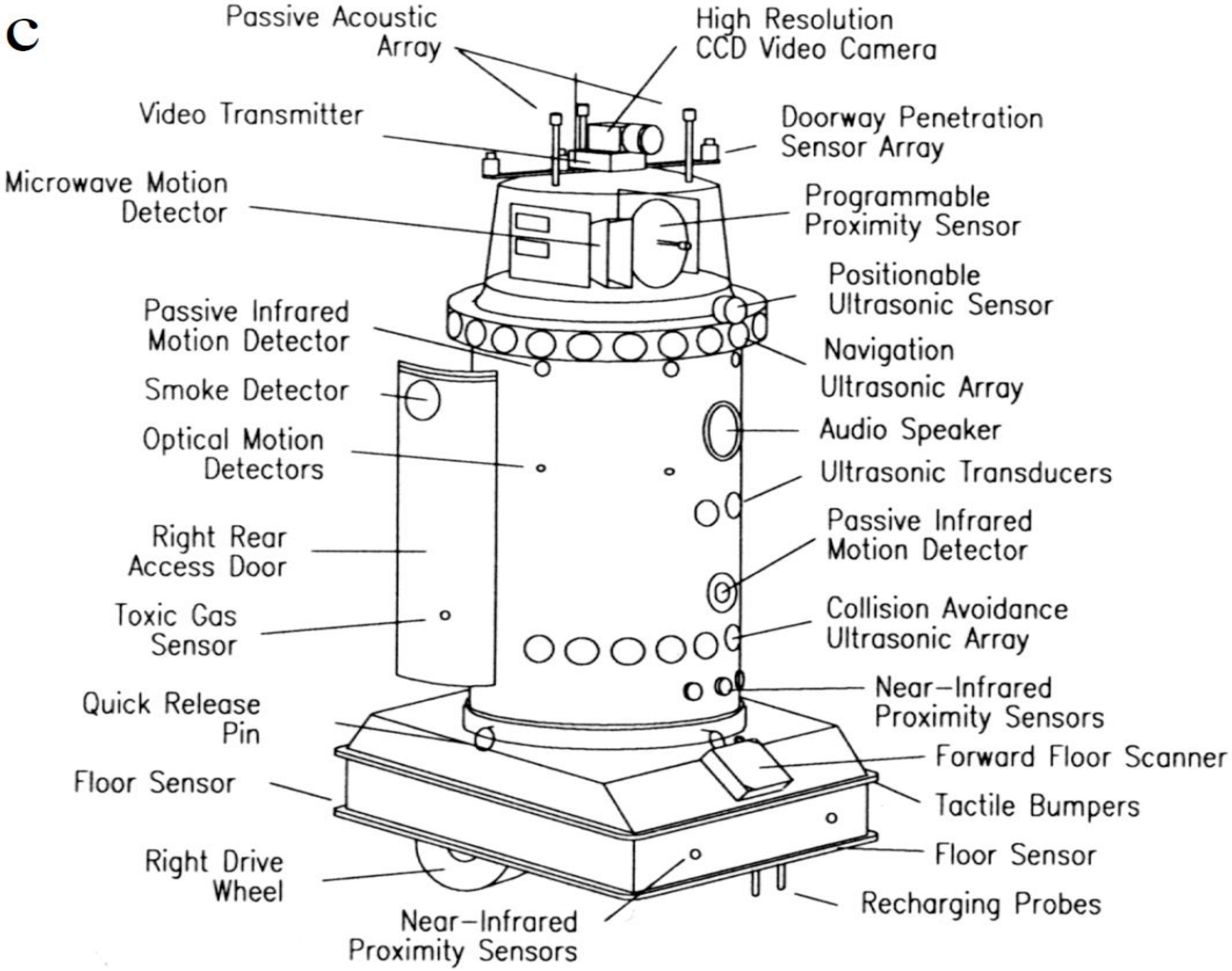
  (monitor state of robot)
  - IMU (accels & gyros)
  - Wheel encoders
  - Doppler radar …

- **Exteroceptive Sensors**

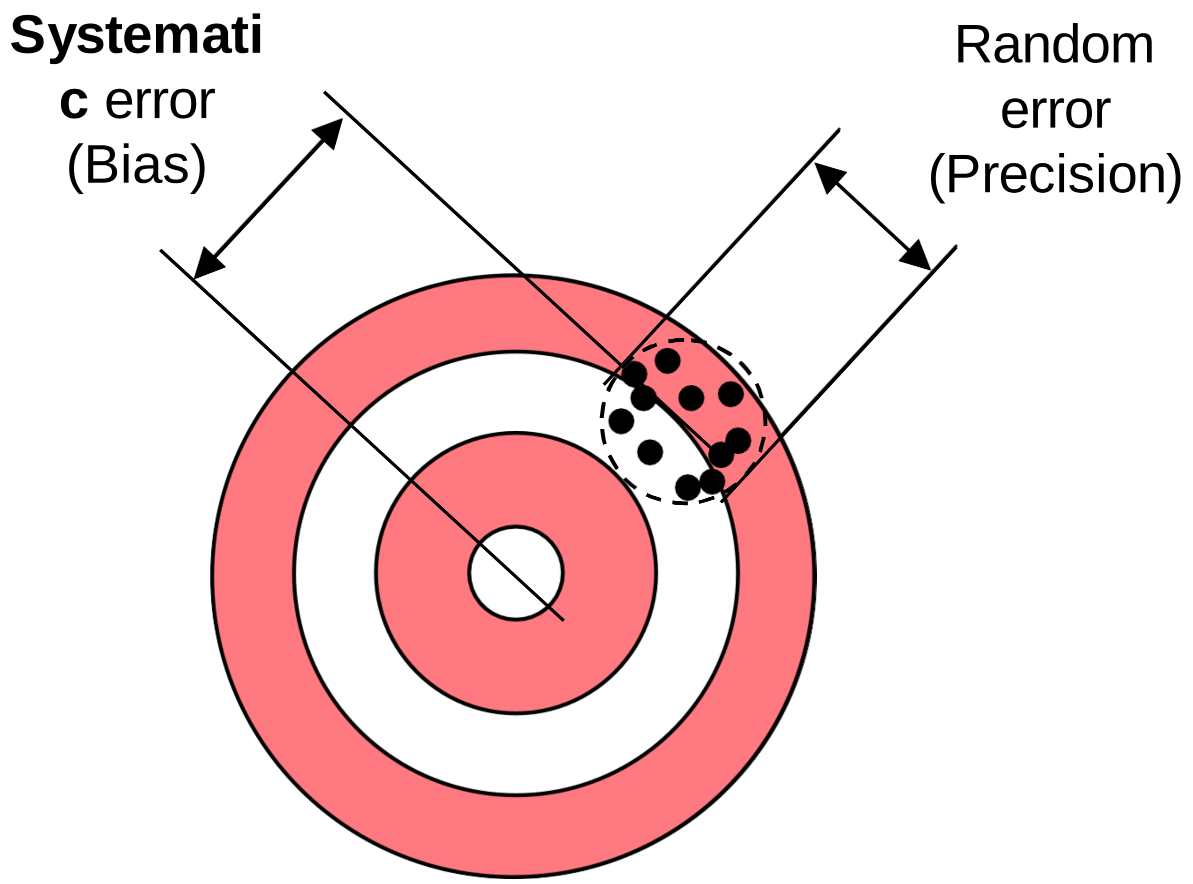
  (monitor environment)
  - Cameras (single, stereo, omni, FL
  - Laser scanner
  - MW radar
  - Sonar
  - Tactile…

1980-82 – ROBART-I Sentry Robot – H. R. Everett





Example: systematic and random errors



Accuracy and errors

- **Systematic errors**
  - Result from a variety of factors
    - Interfering or modifying variables (i.e., temperature)
    - Drift (i.e., changes in chemical structure or mechanical stresses)
    - The measurement process changes the measurand (i.e., loading errors)
    - The transmission process changes the signal (i.e., attenuation)
    - Human observers (i.e., parallax errors)
  - **Systematic errors can be corrected with COMPENSATION methods (i.e., feedback, filtering)**
- **Random errors**
  - Also called NOISE: a signal that carries no information
  - True random errors (white noise) follow a Gaussian distribution
  - Sources of randomness:
    - Repeatability of the measurand itself (i.e., height of a rough surface)
    - Environmental noise (i.e., background noise picked by a microphone)
    - Transmission noise (i.e., 60Hz hum)
  - Signal to noise ratio (SNR) should be >>1
    - With knowledge of the signal characteristics it may be possible to interpret a signal with a low SNR (i.e., understanding speech in a loud environment)

1. Measurement and Correction of Systematic Odometry Errors in Mobile Robots

By **Johann Borenstein and Liqiang Feng**

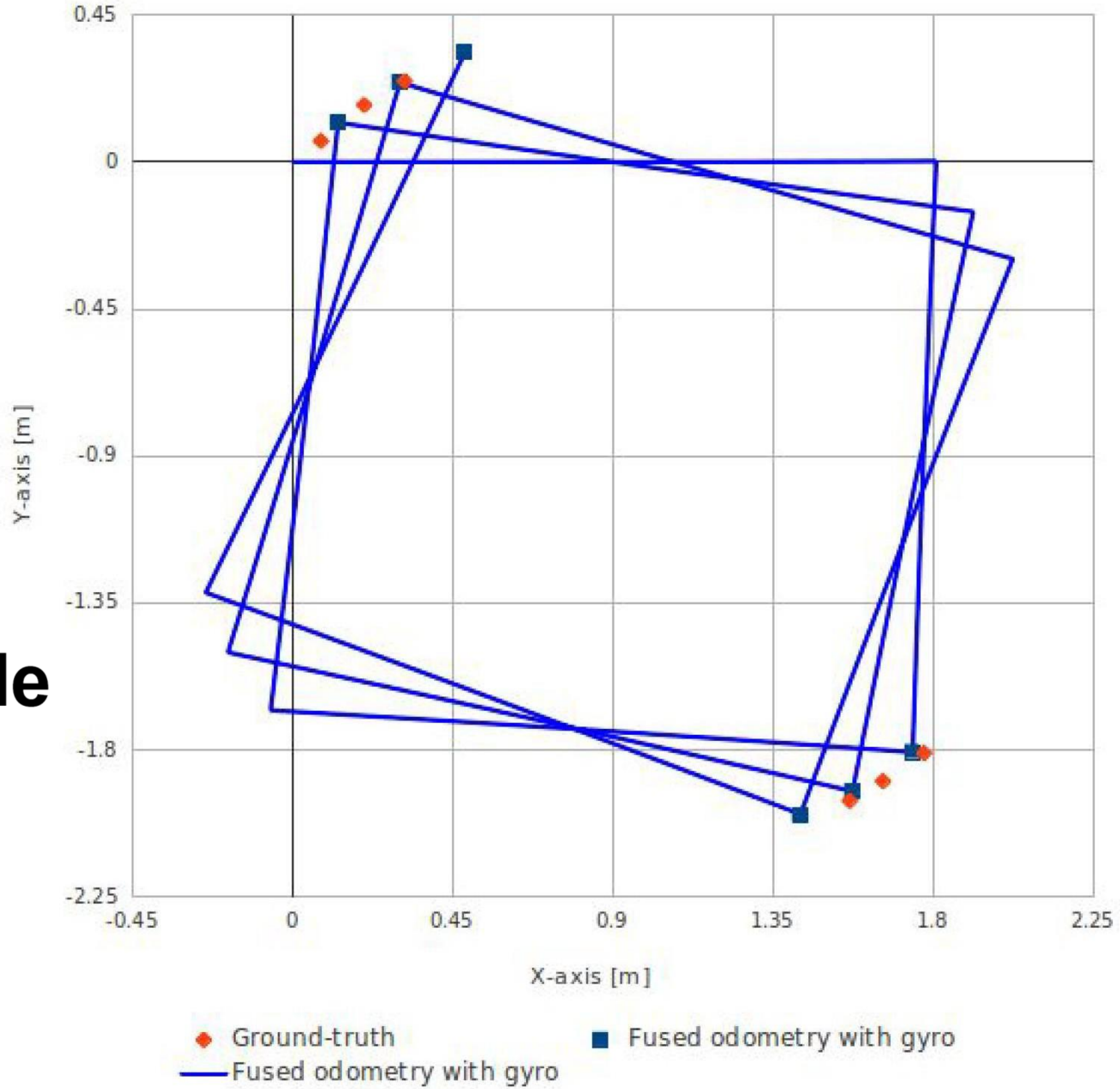
(It would be difficult to get more information on Odometry)

http://www-personal.umich.edu/~johannb/Papers/paper58.pdf

THERE IS HOPE TO ELIMINATE (REDUCE) THE SYSTEMATIC ERRORS BY CALIBRATION!

REDUCING RANDOM ERRORS FROM THE SENSORS IS ANOTHER STORY!
(TO BE TOLD LATER IN COURSE)

ACTUAL PATH ?



Kobuki User Guide

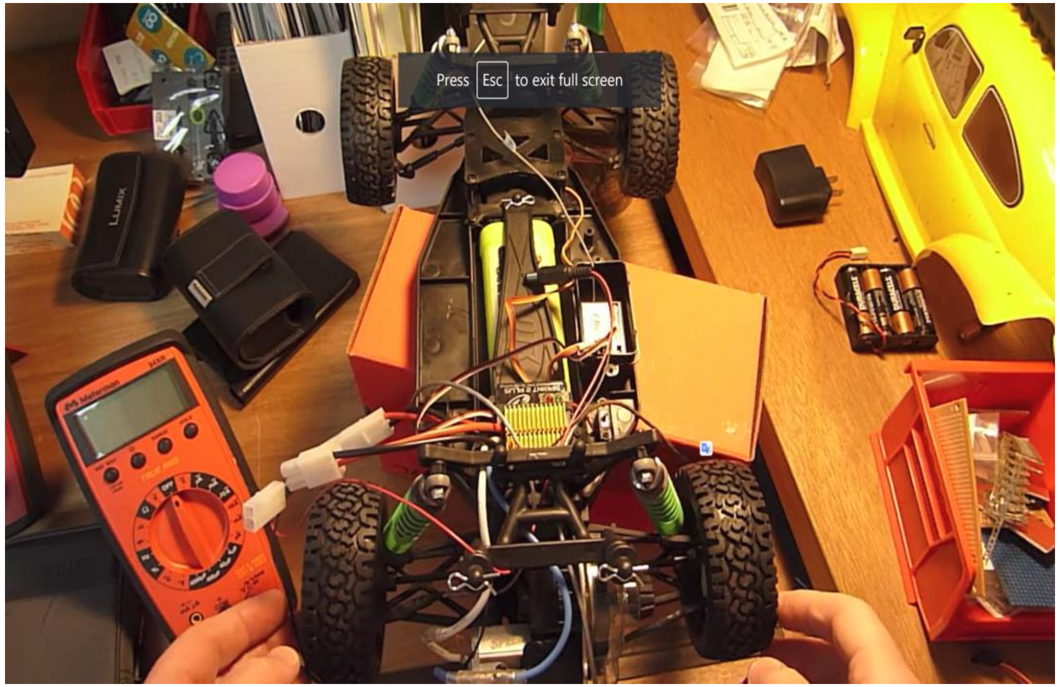
This graph shows the position error of fused odometry with gyro, when robot moves along a square path. Robot moved with 0.1 m/s on the line segment and rotated with 30 deg/s on the corner.

CAN WE IMPROVE IT??

Robot Odometry Calibration

9,586 views Jun 2, 2013 6:57

https://www.youtube.com/watch?v=qsdiIZncgqo



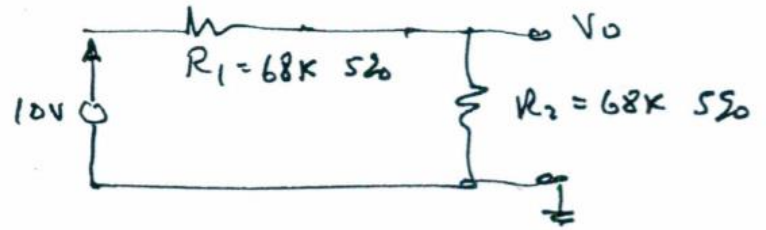
- **Resolution:** Resolution is the minimum step size within the range of measurement of the sensor. In a wire-wound potentiometer, it will be equal to the resistance of one turn of the wire. In a digital device with n bits, the resolution will be

$$\text{Resolution} = \text{Full Range} / 2^n$$

For example, an absolute encoder with 4 bits can report positions up to $2^4 = 16$ different levels. Thus, its resolution is $360/16 = 22.5°$.

Accuracy: Accuracy is defined as how close the output of the sensor is to the expected value. If for a given input, the output is expected to be a certain value, the accuracy is related to how close the sensor's output is to this value.

(1) Tolerance



$$V_{OUT} = V_{IN} \frac{R_2}{R_1 + R_2}$$

(1) If $R_1 = R_2 = 68K$ $V_o = 5.00$ volts

(2) Let $R_1 = 68K(.95) = 64.6K$

$R_2 = 68K(1.05) = 71.4K$

Thus if $R_2 = 68K \pm 5\%$, low voltage out

$$V_o = 10 \times \frac{64.6}{64.6 + 71.4} = 4.75 \text{ volts}$$

If $R_2 = 71.4$; $R_1 = 64.6$, high voltage out

$$V_o = 10 \times \frac{71.4}{R_1 + R_2} = 5.25 \text{ volts}$$

YOU COULD MEASURE EACH RESISTOR AND SELECT THE BEST –
OR DESIGN A FEEDBACK SCHEME TO REDUCE THE EFFECT OF THE VARIABILITY.

https://electronics.stackexchange.com/questions/98357/is-the-error-in-a-5-resistor-consistent-across-measurements

> What I'm really saying is that if a given resistor is "off" by 3.5% I don't really care... as long as it's **always** off by the same 3.5%. But if from one measurement (voltage? current?) to another it might be +2% one time and -3% another time, then I need to get higher quality components ?

The answer to your questions is mostly covered in the data sheets. A 5% tolerance resistor will also have a specification for temperature drift, "load life" (drift with time under certain environmental conditions) and so on. It's possible to make a 1% resistor that is just as crappy as a 5% resistor in stability, it's just trimmed closer to begin with (and at a certain temperature). Calibration can reduce the initial inaccuracy, but it won't reduce the other kinds of drift. The drift will determine whether you can make a 0.1% circuit with 1% resistors or a 0.5% circuit with 5% resistors.

Temperature Coefficient	$1\ \Omega \le R \le 10\ \Omega$	±200 ppm/°C
	$10\ \Omega < R \le 10\ M\Omega$	±100 ppm/°C
	$10\ M\Omega < R \le 22\ M\Omega$	±200 ppm/°C

Why Punching a Robot Is a Bad Idea (Go for the Sensors Instead) !

https://www.theatlantic.com/video/index/257060/why-punching-a-robot-is-a-bad-idea-go-for-the-sensors-instead/

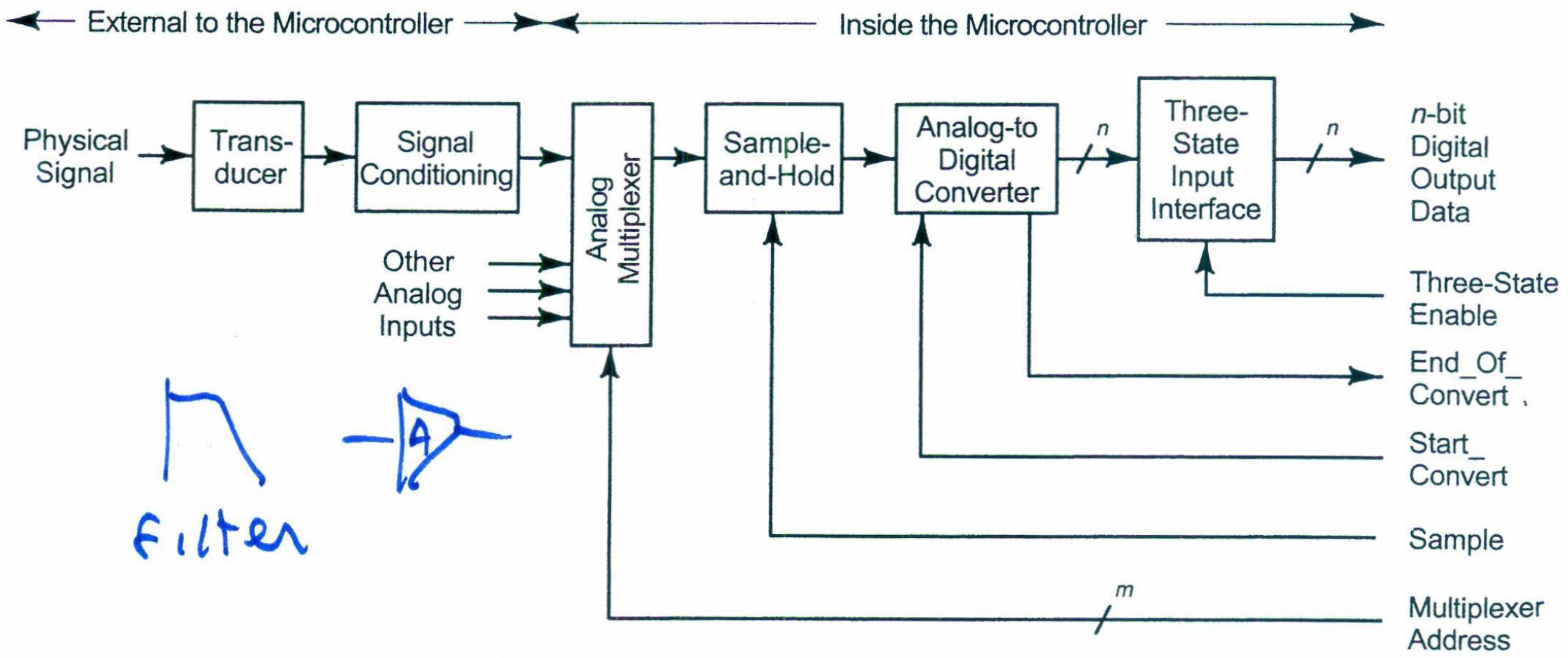


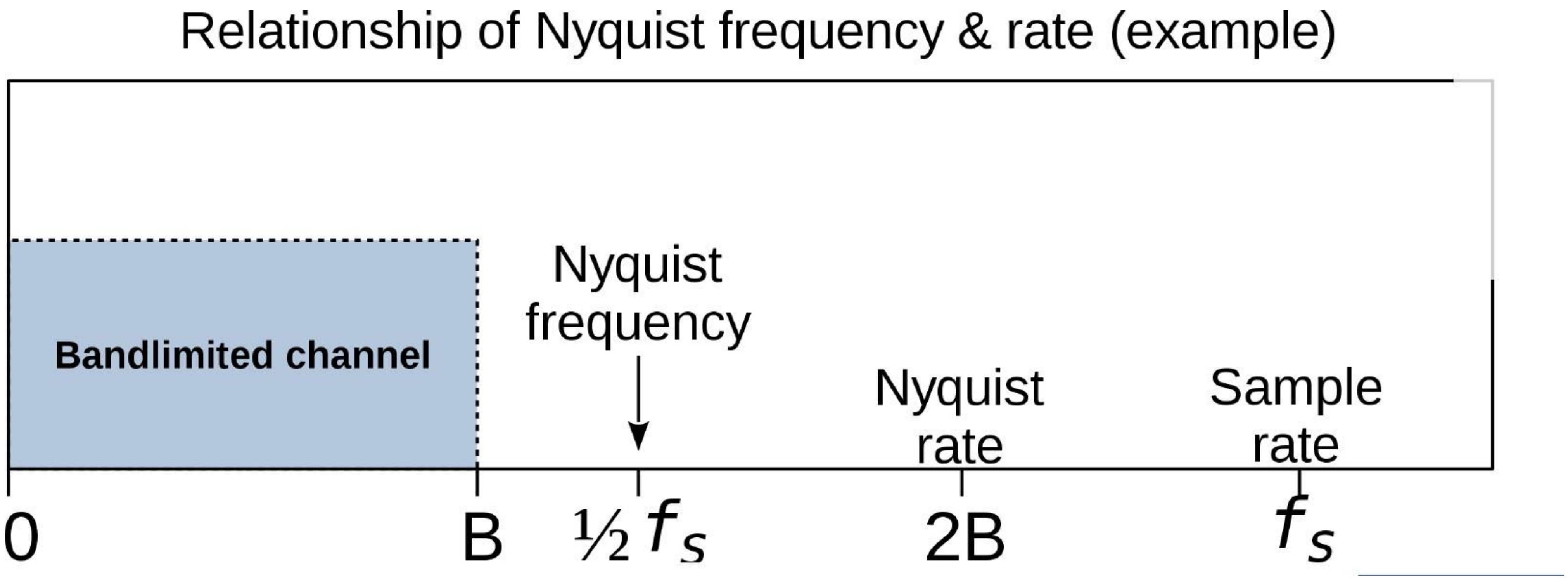
Figure 13-1 Data acquisition system.

SAMPLING THEOREM THE BIG DEAL!!

- HOW OFTEN DO WE NEED TO SAMPLE?
  - DEPENDS on FREQUENCY of SINUSOID
  - ANSWERED by SHANNON/NYQUIST Theorem
  - ALSO DEPENDS on "**RECONSTRUCTION**"

Shannon Sampling Theorem

A continuous-time signal $x(t)$ with frequencies no higher than $f_{\max}$ can be reconstructed exactly from its samples $x[n] = x(nT_s)$, if the samples are taken at a rate $f_s = 1/T_s$ that is greater than $2f_{\max}$.



Basic Sampling at 2x Highest Frequency in Band (B)

Nyquist Limit

The absolute limiting resolution of a sensor is determined by its Nyquist limit. This is defined as being one half of the sampling frequency, a.k.a **the number of pixels/mm** (Equation 3). For example, the Sony ICX285 is a monochrome CCD sensor with a horizontal active area of 9mm containing 1392 horizontal pixels each 6.45µm in size. This represents a horizontal sampling frequency of 155 pixels/mm (1392 pixels / 9mm = 1mm / 0.00645 mm/pixel = 155).

SPATIAL SAMPLING

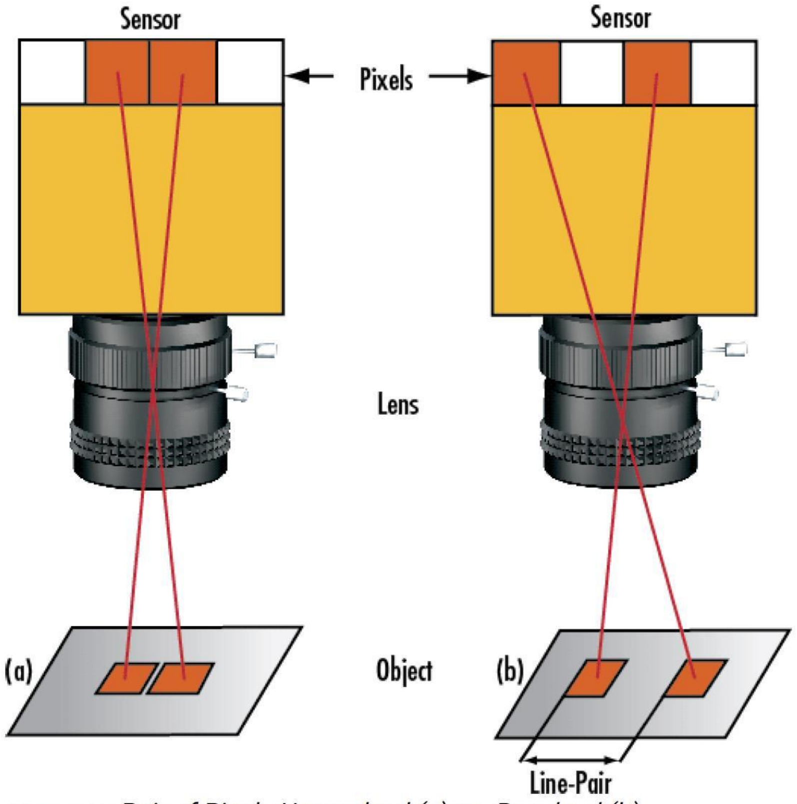


Figure 2: *Pair of Pixels Unresolved (a) vs. Resolved (b)*

Lp = LinePairs

https://www.edmundoptics.com/knowledge-center/application-notes/imaging/camera-resolution-for-improved-imaging-system-performance/

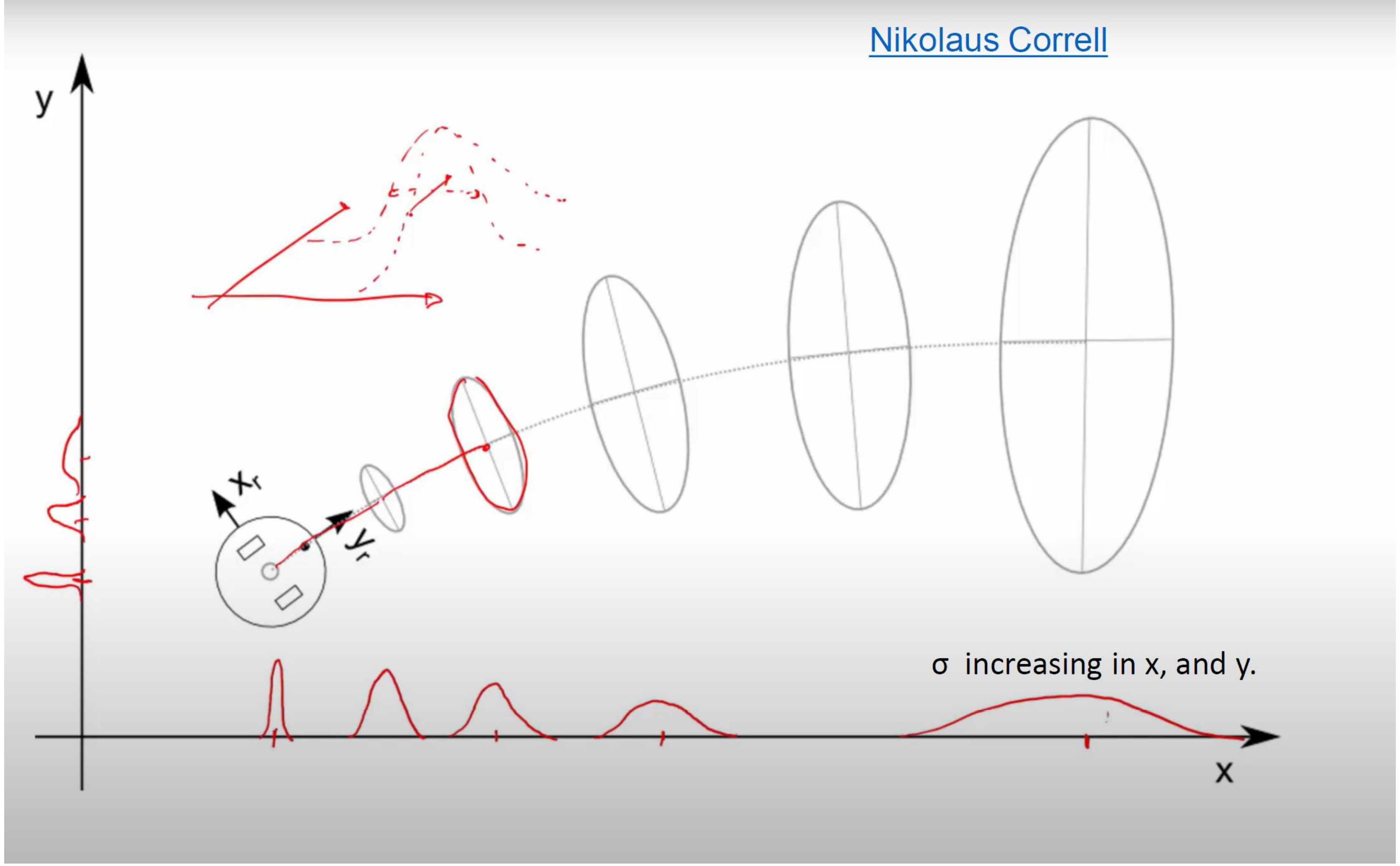
Video Aliasing

Why car wheels rotate backwards in movies 4:25

https://www.youtube.com/watch?v=SFbINinFsxk&feature=youtu.be

INCORRECT SAMPLING LEADS TO “FUNNY THINGS” IN VIDEOS ALSO.

Navigation Errors



IF YOU DO NOT CALIBRATE CAREFULLY!

IF YOU DO NOT UNDERSTAND RANDOM ERRORS IN ROBOT NAVIGATION

That’s All Folks

Tolerance Errors Resistors

Capacitance and RC circuits

Data Acquisition Resolution, Accuracy, Dynamic Range

Temperature Sensor to Voltage to Digital

Smoothing – filtering – MATLAB

Least Squares Curve Fitting