Lab 3 - Hints

Low-level drive functionality

- In this lab, we ask you to use only the `drive_wheels` command for navigation
- Do not use `drive_straight`, `GoToPose`, `GoToObject` or any other high-level commands

drive_wheels(*l_wheel_speed*, *r_wheel_speed*, *l_wheel_acc=None*, *r_wheel_acc=None*, *duration=None*)

Tell **Cozmo** to move his wheels / treads at a given speed, and optionally stop them after a given duration.

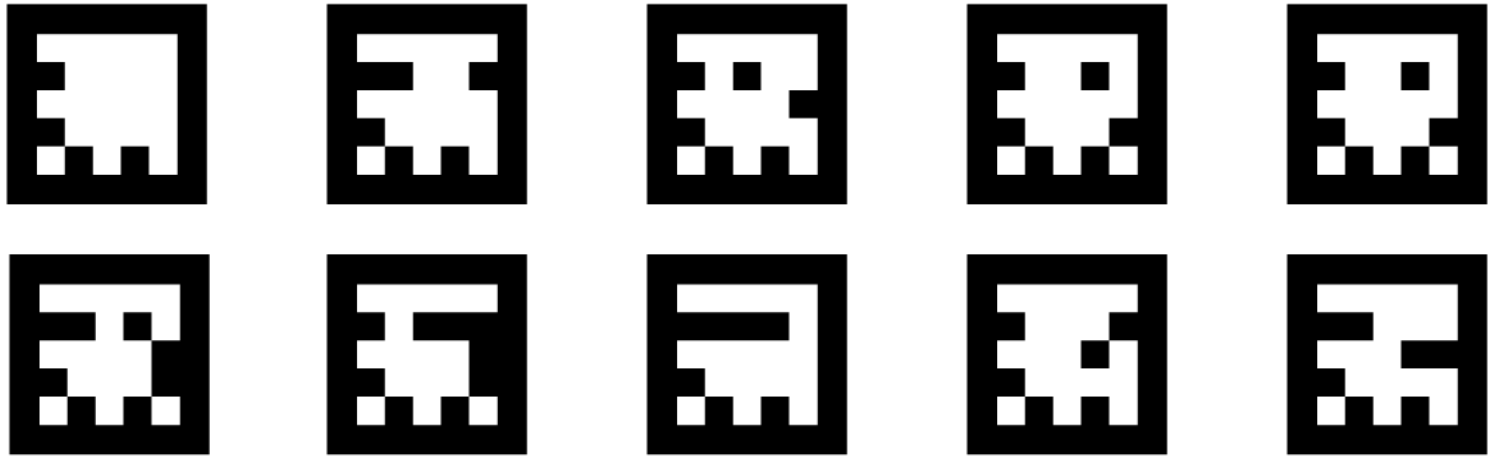
If duration is `None` then this is equivalent to the non-async `drive_wheel_motors()` method.

Parameters:

- **l_wheel_speed** (*float*) – Speed of the left tread (in millimeters per second).
- **r_wheel_speed** (*float*) – Speed of the right tread (in millimeters per second).
- **l_wheel_acc** (*float*) – Acceleration of left tread (in millimeters per second squared). `None` value defaults this to the same as l_wheel_speed.
- **r_wheel_acc** (*float*) – Acceleration of right tread (in millimeters per second squared). `None` value defaults this to the same as r_wheel_speed.
- **duration** (*float*) – Time for the **robot** to drive. Will call `stop_all_motors()` after this duration has passed.

AR Markers

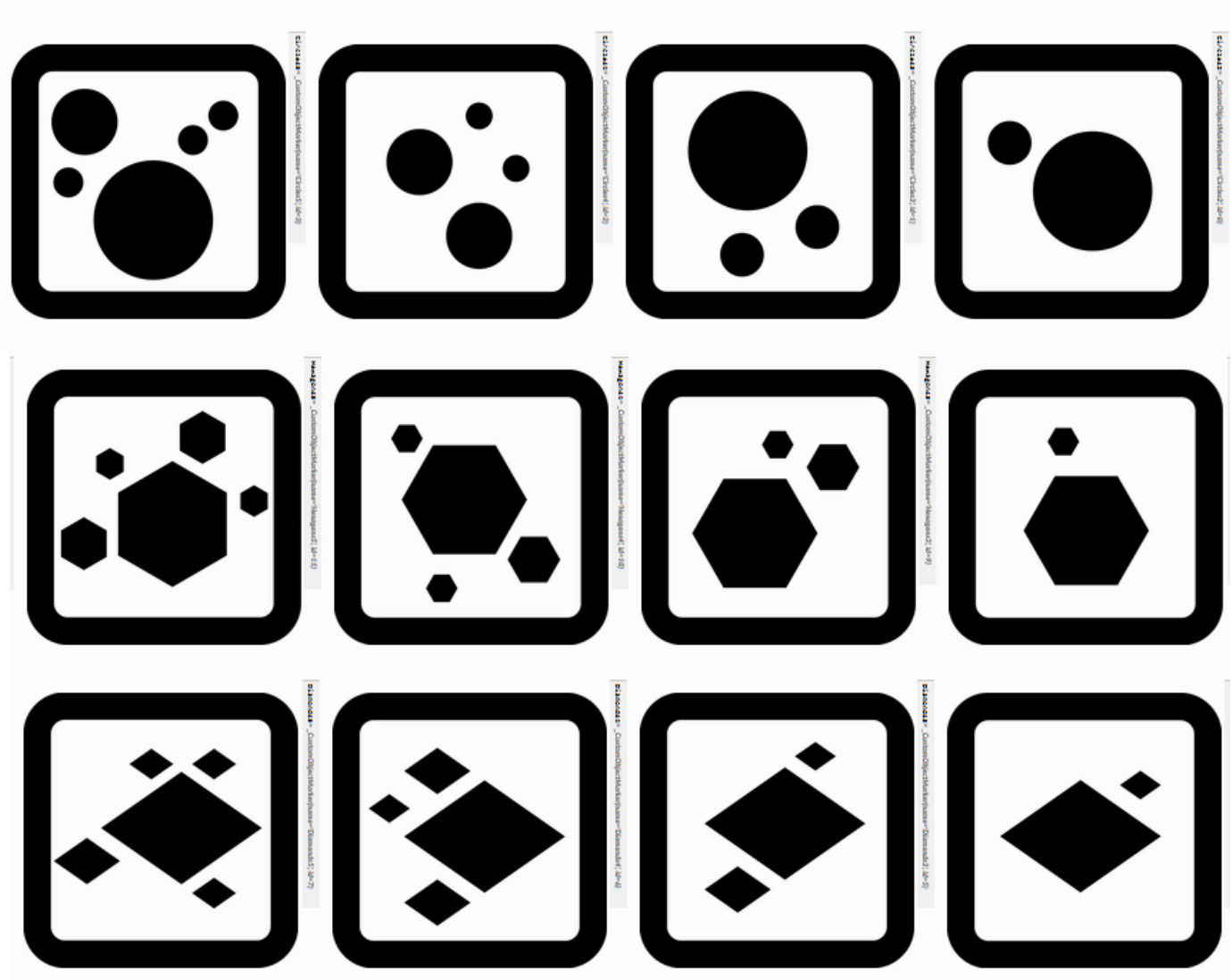
- AR markers are used in robotics for fast and reliable object detection
- Instead of color markers, in this lab, you will use AR markers.



AR Markers on Cozmo cubes

- Each cube has its own unique AR marker
- Object detection using AR markers is much more robust to change in light condition and to noise.

Customized AR Markers on Cozmo cubes



- From Cozmo's SDK
- Each one is unique
- They could be printed in different scales depending on the application

AR Markers for Lab 3

As mentioned in instruction file, to generate a customized marker, use one of the following options:

- Pre-defined custom markers

http://cozmosdk.anki.com/docs/generated/cozmo.objects.html#cozmo.objects.CustomObject

- AR markers

https://pypi.org/project/ar-markers/ (not available anymore)

- Opencv ArUco

1. https://www.pyimagesearch.com/2020/12/14/generating-aruco-markers-with-opencv-and-python/
2. https://www.pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-and-python/
3. https://www.pyimagesearch.com/2020/12/28/determining-aruco-marker-type-with-opencv-and-python/
4. https://mecaruco2.readthedocs.io/en/latest/notebooks_rst/Aruco/aruco_basics.html

- You will need two different markers for this lab.

Coordinate Transformation

- The pose (position + orientation) of a cube (and the robot) are reported in a fixed coordinate system (i.e., robot is not always (0,0))
- You need to perform coordinate transformation to find the relative pose of the cube w.r.t the robot
- Use this information for navigating towards the cube

Finite State Machine

- You can use an existing python library

There exist several implementations and tutorials:
https://pypi.org/project/python-statemachine/

Install it using: *pip install python-statemachine*

There are others that we do not recommend:
https://pypi.org/project/fsmpy/
https://pythonspot.com/python-finite-state-machine/
https://www.python-course.eu/finite_state_machine.php
https://python-3-patterns-idioms-test.readthedocs.io/en/latest/StateMachine.html