Oklahoma Drone workshop

Dr Amy McGovern
School of Computer Science and School of Meteorology
University of Oklahoma

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Drone art

Winter Olympics

https://www.youtube.com/watch?v=fCd6P7Ya160

Time Magazine

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

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Afternoon agenda

- User input
- Conditionals
- Looping
- Sensors
- *Vision*

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

December Challenge Day Tasks

Regular challenges (red means we worked on them today)

1. Take off, fly a short distance, land
2. Take off, fly to a precise location
3. Take off, fly to the hospital, land, take off again, turn, fly to the second hospital, land.
4. Take off, fly a specific geometric pattern in the air in the horizontal plane, land.
5. Take off, fly a specific geometric pattern in the air in the vertical plane, land.
6. Take off, fly a n-sided polygon in the horizontal plane, land.
7. Take off, fly a n-sided polygon in the vertical plane, land.
8. Take off, fly a specific user specified distance, land at the location (or within a radius).

Advanced challenges

1. Take off, fly until you see a specific vision marker, land.
2. Take off, fly to a specific marker, use the marker to take a specific action, land when you see the landing marker.
3. Take off, Find the hidden object (using markers), fly back home.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

User Input: Part 1

- Programs are more interesting with input! We can use the input to change the behavior of our programs.
- Start a new python file in your editor and type

```
name = input("What is your name?")
```

- `input()` is a python function that prints the text you give it and waits for you to type a response AND hit return/enter

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

User Input: Part 2

- Now add the 2nd line to your program

```
name = input("What is your name?")
print("Nice to meet you %s" % name)
```

- Run the program. What does it do?
- `print()` prints everything inside the parentheses to the screen
- `"%s" % name`
  - Transforms the variable *name* into a string that can be printed

Conditionals Part 1

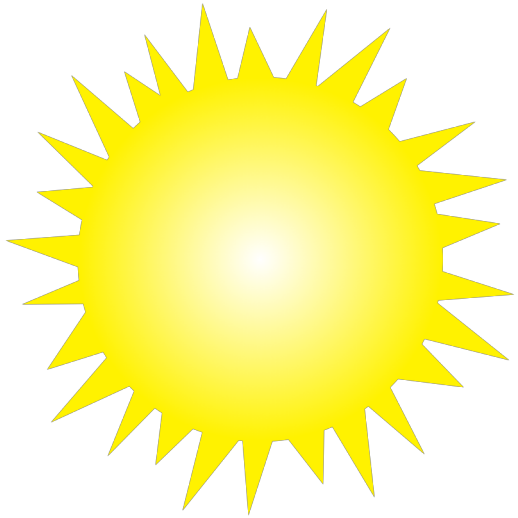
If

Then

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Conditionals Part 2

If 

Then 

Otherwise 

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Conditionals in Python

```
shape = input("Should we fly a circle or a square?")
if (shape == "circle"):
    print("Let's fly in a circle!")
elif (shape == "square"):
    print("Let's fly in a square")
else:
    print("Shape %s is unknown.  Help me learn how to fly it!" % shape)
```

Notice the use of colons here.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Your turn!

Ask the user if they want a circle or a square flown.
Make sure your program prints out their choice.

For the December challenge, have your program actually fly the shape that the user specified!

For today, we will move on towards using sensors.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Looping: While loop

- While teeth are not clean
  - Brush teeth

- While we have not reached the target
  - Fly forward

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Quick While Loop Practice

- Syntax in python is:

```
count = 1
while (count <= 10):
    # do something
    print("Count is %d" % count)
```

Notice the use of colon here.

- Your quick task:
  - Use a while loop to have your drone fly forward for 3 seconds total but in small movements. Do NOT just do fly_direct with a duration of 3!
  - Max duration is 0.1
  - **Useful code**

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Looping: `for` loop

- Can we loop a specific number of times?
- You can use while but `for` loops are designed for this
- Python syntax

```
for side in range(1,4):
    # fly forward
    # turn 60 degrees
```

- One key thing to remember for python: numbers are inclusive on the lower end but exclusive on the higher end
- range(1,4) gives you numbers 1, 2, and 3 but NOT 4!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Looping: `for` loop

Example `for` loop:

```
for side in range(1,4):
    print("Flying side %d" % side)
```

- How can you draw a triangle using a drone?
- For side = 1 to 3
  - Fly forward
  - Turn 60 degrees

```
for side in range(1,4):
    # fly forward
    # turn 60 degrees
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Your turn!

Use a for loop to have your drone repeat a sequence of actions 3 times:

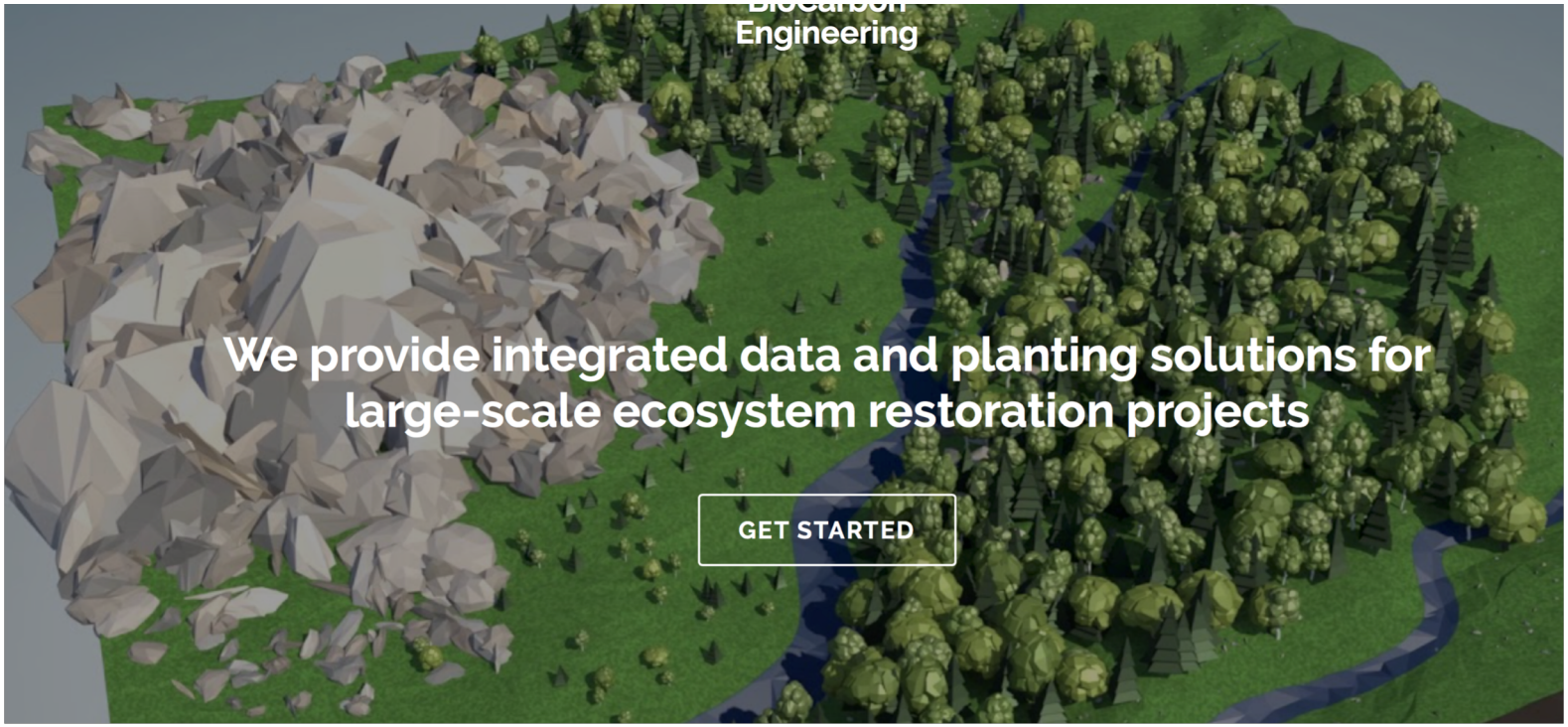
fly forward

turn

```
for side in range(1,4):
    # fly forward
    # turn 60 degrees
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA



Planting New Trees

https://www.youtube.com/watch?v=JcJ7vLwtSIM

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Sensors

- Mambo sensors
  - Flying state
  - Battery
  - Speed in x, y, and z (m/s)
    - Timestamp is in milliseconds!
  - Altitude (this is REALLY noisy)
  - Estimated orientation
  - All sensors updated at 2Hz

- Bebop sensors
  - Flying state
  - Battery
  - Altitude
  - Pitch, Roll, Yaw
  - GPS location (**only useful outside**)
  - Speed in x, y, z
    - Timestamp is in milliseconds!
  - All sensors updated at 10 Hz
  - Many more but beyond the scope of this workshop

Using the Sensors: Flying state

- Flying state

  `mambo.sensors.flying_state`

  `bebop.sensors.flying_state`

- Possible flying states
  - “motor_ramping”
  - “landed”
  - “takingoff”
  - “hovering”
  - “flying”
  - “landing”
  - “emergency”
  - “init”

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Using the Sensors: Speed

- Speed in the forward direction:

  ```
  mambo.sensors.speed_x
  bebop.sensors.sensors_dict["SpeedChanged_SpeedX"]
  ```

- Units of meters/second
- Average speed since the last update in speed
- Ignore y and z speeds for today

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Using the Sensors: Timestep of the updates

- Timestep

  `mambo.sensors.speed_ts`

  `Bebop does not broadcast this data.  It runs approximately every 100 ms`

- Measured in **milliseconds**
- Converting milliseconds to seconds:
  - timestep_in_seconds = speed_ts / 1000

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Using the Sensors: Battery

- Battery

  `mambo.sensors.battery`

  `bebop.sensors.battery`
- Initialized to 100%
- Becomes accurate after a few seconds
- Do NOT believe the initial 100%! Only believe it after a few seconds
- If your mambo eyes are blinking red, your batteries need to be charged!
- If your bebop won't take off, your batteries need to be charged!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Estimating Distance

- If I drive in my car at 55 mph for 1 hour, how far do I go?
- $Distance = rate * time$
- $Total\ distance = sum\ over\ all\ times\ (rate\ at\ time\ t * time\ spent\ at\ that\ speed)$

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Using Sensors: Tying this all together

<u>Activity:</u> Fly the drone a user specified distance

Step 1: Ask the user how far you should fly (User Input)

Step 2: Create your mambo object and connect to it **after** asking the user

Step 3: Hover until the sensors are ready (Check function sheet for help)

Step 4: Estimate the distance flown using the sensors!

- Distance flown = distance flown already + new distance flown
- New distance flown = speed over the last x seconds * x seconds
- Mambo: this will be more accurate for lower speeds

Step 5: Fly until the distance meets or exceeds the distance the user specifies (While loop)

Step 6: Land and disconnect

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Step 1: User input

- How do we ask the user how far we want to go?
  - answer = input(“How far do you want to fly?”)
- How do you save the result as a float and not a string (the default)
  - distance = float(answer)

Step 2: Create your mambo object and connect to it

- You can re-use your code from your earlier projects
- Create the mambo object, set the parameters for maximum speed, tell it to use flat_trim, and take off

```
from pyparrot.Minidrone import Mambo

# create the drone object and connect
drone = Mambo(use_wifi=True)
drone.connect(num_retries=3)

# maximum speeds - DO NOT CHANGE
drone.set_max_tilt(10)
drone.set_max_vertical_speed(1)

drone.flat_trim()

# takeoff
drone.safe_takeoff(5)
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Step 3: Hover until the speed sensor is sending data

- The speed sensor does not send data right away
- Write a small (2 lines!) while loop that hovers as long as mambo.sensors.speed_ts is 0

Check function sheet for help!

while (condition):

check sensor

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Step 4: Initialize your variables

- `timestep` – timestep from the last time you sensed speed
  - Mambo: approximately every 500 ms (but not always exactly equal)
  - Bebop: approximately every 100 ms
  - You need to track this for your calculation!

- `estimatedDistance` - Estimated distance you have traveled so far
  - This will be updated each time you get a new sensor reading
  - Initialize it to 0

Step 5: Fly until you exceed the user's distance

- While loop
  - What is the condition?
- Inside the loop:
  - Fly a SHORT amount (less than 0.5 seconds)
  - Sleep briefly (check function sheet for help)
  - Check the sensor values
    - Calculate how much time has passed
      (mambo.sensors.speed_ts)
      (~100 ms in bebop)
  - Compute how far you have moved
    - Distance = rate * time
    - mambo.sensors.speed_x
    - bebop.sensors.sensors_dict["SpeedChanged_SpeedX"]
    - **Remember time is in milliseconds!**

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Step 6: Ending the loop

- You flew the distance!
- Land
- Disconnect

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Code we typed in the workshop for sensors

```
from pyparrot.Minidrone import Mambo

# ask the user how far to fly
answer = input("How far do you want to fly in meters? ")
distance = float(answer)
print(distance)

# create the drone object and connect
drone = Mambo(use_wifi=True)
drone.connect(num_retries=3)

# maximum speeds - DO NOT CHANGE
drone.set_max_tilt(10)
drone.set_max_vertical_speed(1)

drone.flat_trim()

# takeoff
drone.safe_takeoff(5)

# hover while speed sensor is NOT working
while (drone.sensors.speed_ts == 0):
    drone.smart_sleep(0.1)

total_distance = 0
last_timestep = drone.sensors.speed_ts

while (total_distance <= distance):
    # do something flying related
    drone.fly_direct(roll=0, pitch=10,yaw=0,vertical_movement=0, duration=0.1)

    # how much time has elapsed?
    # convert from milliseconds to seconds
    elapsed_time = (drone.sensors.speed_ts - last_timestep)/1000.0

    # distance = rate times time
    total_distance = total_distance + drone.sensors.speed_x * elapsed_time

    # update last time step
    last_timestep = drone.sensors.speed_ts

drone.safe_land(2)
drone.disconnect()
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Your turn!

Use loops and sensors and user input to have your drone fly forward a specific number of meters that your user wants

Be safe and don't hit the walls at full speed!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

December Challenge Day Tasks

Regular challenges (red means we worked on them today)

1. Take off, fly a short distance, land
2. Take off, fly to a precise location
3. Take off, fly to the hospital, land, take off again, turn, fly to the second hospital, land.
4. Take off, fly a specific geometric pattern in the air in the horizontal plane, land.
5. Take off, fly a specific geometric pattern in the air in the vertical plane, land.
6. Take off, fly a n-sided polygon in the horizontal plane, land.
7. Take off, fly a n-sided polygon in the vertical plane, land.
8. Take off, fly a specific user specified distance, land at the location (or within a radius).

Advanced challenges

1. Take off, fly until you see a specific vision marker, land.
2. Take off, fly to a specific marker, use the marker to take a specific action, land when you see the landing marker.
3. Take off, Find the hidden object (using markers), fly back home.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Thank you! See you December 10 for the KIPR Drone Challenge Day!!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Vision slides to be used for advanced groups only

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Vision software

- This is NOT on your USB drive
- VLC
  - https://www.videolan.org/vlc/index.html
- Opencv
  - `conda install opencv`

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Saving the Swimmers

- https://www.youtube.com/watch?v=OTM_dsGZZpI

https://www.bbc.com/news/world-australia-42731112

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Drone Vision

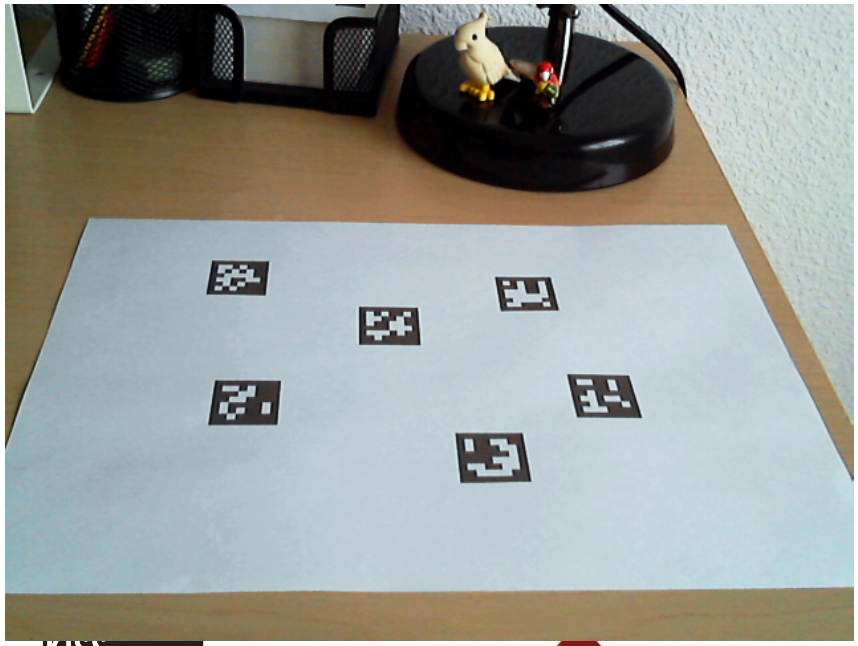
- Vision is the best sensor for both drones!
- FPV camera
  - Vision runs at 30 frames per second
  - Approximately ½ second delay on high-end laptops
  - Up to 1 second on slower laptops
  - Mambo: forward facing
  - Bebop: camera can be rotated (up and down)
- Mambo has a ground facing camera
  - Photos can be snapped and downloaded with about 0.1-0.3 second delay
  - Resolution is poor

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Processing Images

- Image processing is a very difficult task in general
- Opencv provides a lot of built in image processing tools
- We will use the aruco marker library today



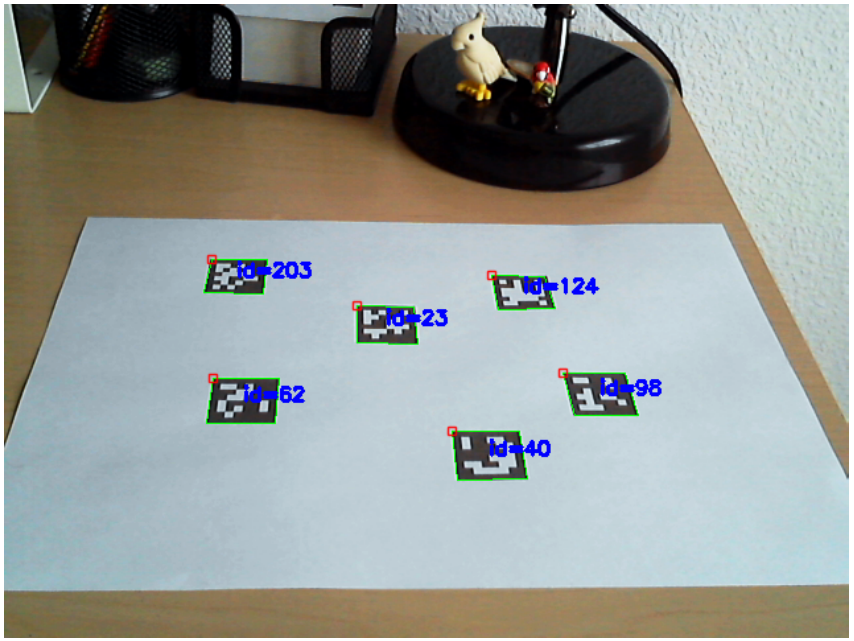


Image processing

- I have provided initial image processing code to you to simplify the process
- You are welcome to use this code for your team and to use it in the December challenge as well

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Starting code

```
def run_vision_code(droneVision, args):
    """
    Code that is run when the user presses "run" on the vision GUI

    :param droneVision: the drone vision object
    :param args: other arguments (first one is drone)
    :return:
    """

    # get the drone object out of the user arguments
    drone = args[0]

    # takeoff
    drone.safe_takeoff(5)
    drone.smart_sleep(3)

    ##### OK 2018 workshop - write the code for what you want your drone to to!
    # this is where you tell your drone to take off, fly, act according to the markers, land, etc



    # cleanup the vision and the connection in this function at the end of your other code
    drone.safe_land(5)
    droneVision.close_video()

    # disconnect nicely so we don't need a reboot
    drone.disconnect()
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Aruco Marker Magic

- Detecting Aruco markers requires initialization

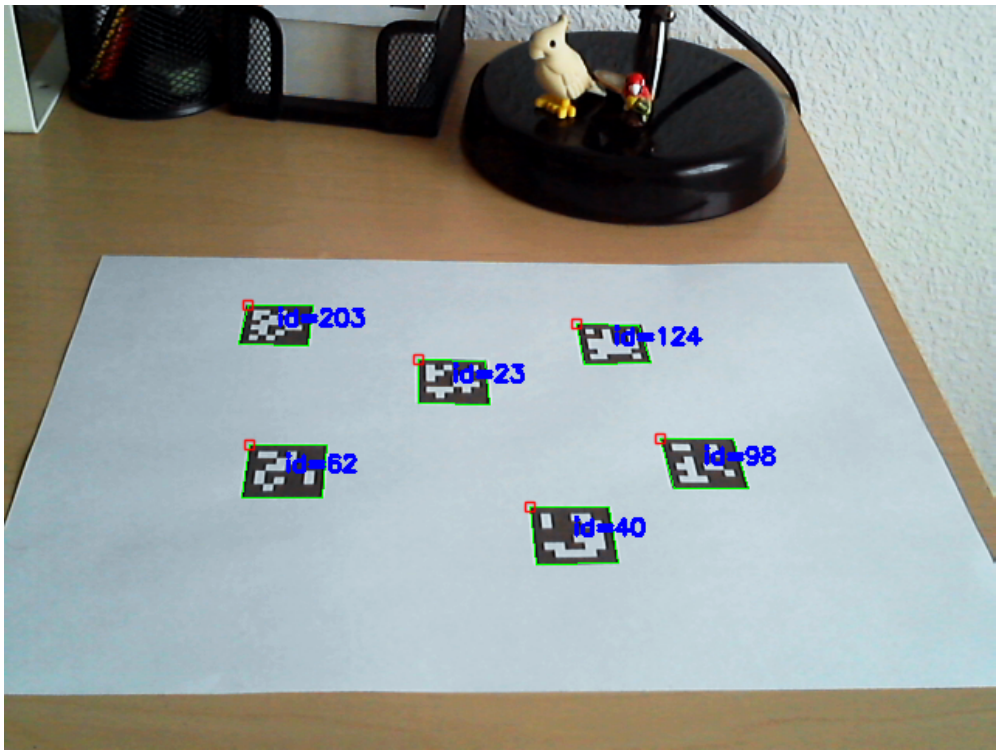
```
# parameters needed to make aruco vision work
aruco_dict = aruco.Dictionary_get(aruco.DICT_6X6_250)
aruco_parameters = aruco.DetectorParameters_create()
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Finding Aruco Markers

```
# find markers
corners, ids, rejectedImgPoints = aruco.detectMarkers(img, aruco_dict, parameters=aruco_parameters)
```



KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Aruco Markers

```python
class ArucoMarker:
    """
    Computes and holds useful information about the aruco markers
    all in a single object.  Keeps the id, the centers, the corners, and the size.
    """
    def __init__(self, corners, id):
        self.corners = corners
        self.id = id

        # calculate the center of the box
        centers = np.mean(self.corners, axis=0)
        self.center = (centers[0], centers[1])

        # compute the size of the box
        maxes = np.max(self.corners, axis=0)
        mins = np.min(self.corners, axis=0)
        self.size = (maxes[0] - mins[0]) * (maxes[1] - mins[1])
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Creating Markers

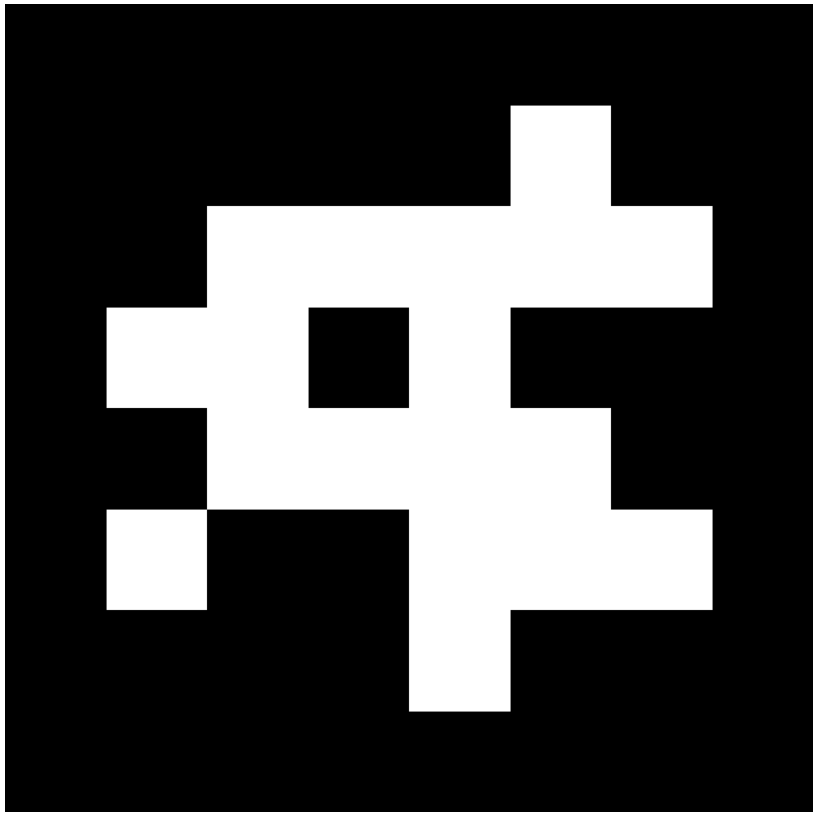
```
def create_aruco_markers(corner_list, ids):
    """
    Makes the aruco_marker list from the data returned from aruco.detectMarkers
    :param corner_list: list of all the corners.  Size is nx4x2 where n is the number of markers seen
    :param ids: ids for each marker
    :return: list of aruco_marker objects
    """

    if (ids is None):
        return list()

    markers = list()

    for counter, id in enumerate(ids):
        my_id = id[0]
        new_marker = ArucoMarker(np.squeeze(corner_list[counter]), my_id)
        markers.append(new_marker)

    return markers
```

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Your Turn: First Vision Task

- Fill in the blank (where it says "OK Workshop put code here")
- Have your drone hover until it sees Marker 16
- Marker 16 means flip
  - Make sure you send the flip command and then a sleep command for at least 2 seconds
- Land after you have flipped once
  - Repeated flips will cause a CRASH



KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

December Challenge Day Tasks

Regular challenges (red means we worked on them today)

1. Take off, fly a short distance, land
2. Take off, fly to a precise location
3. Take off, fly to the hospital, land, take off again, turn, fly to the second hospital, land.
4. Take off, fly a specific geometric pattern in the air in the horizontal plane, land.
5. Take off, fly a specific geometric pattern in the air in the vertical plane, land.
6. Take off, fly a n-sided polygon in the horizontal plane, land.
7. Take off, fly a n-sided polygon in the vertical plane, land.
8. Take off, fly a specific user specified distance, land at the location (or within a radius).

Advanced challenges

1. Take off, fly until you see a specific vision marker, land.
2. Take off, fly to a specific marker, use the marker to take a specific action, land when you see the landing marker.
3. Take off, Find the hidden object (using markers), fly back home.

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA

Thank you! See you December 10 for the KIPR Drone Challenge Day!!

KISS INSTITUTE FOR PRACTICAL ROBOTICS

The UNIVERSITY of OKLAHOMA