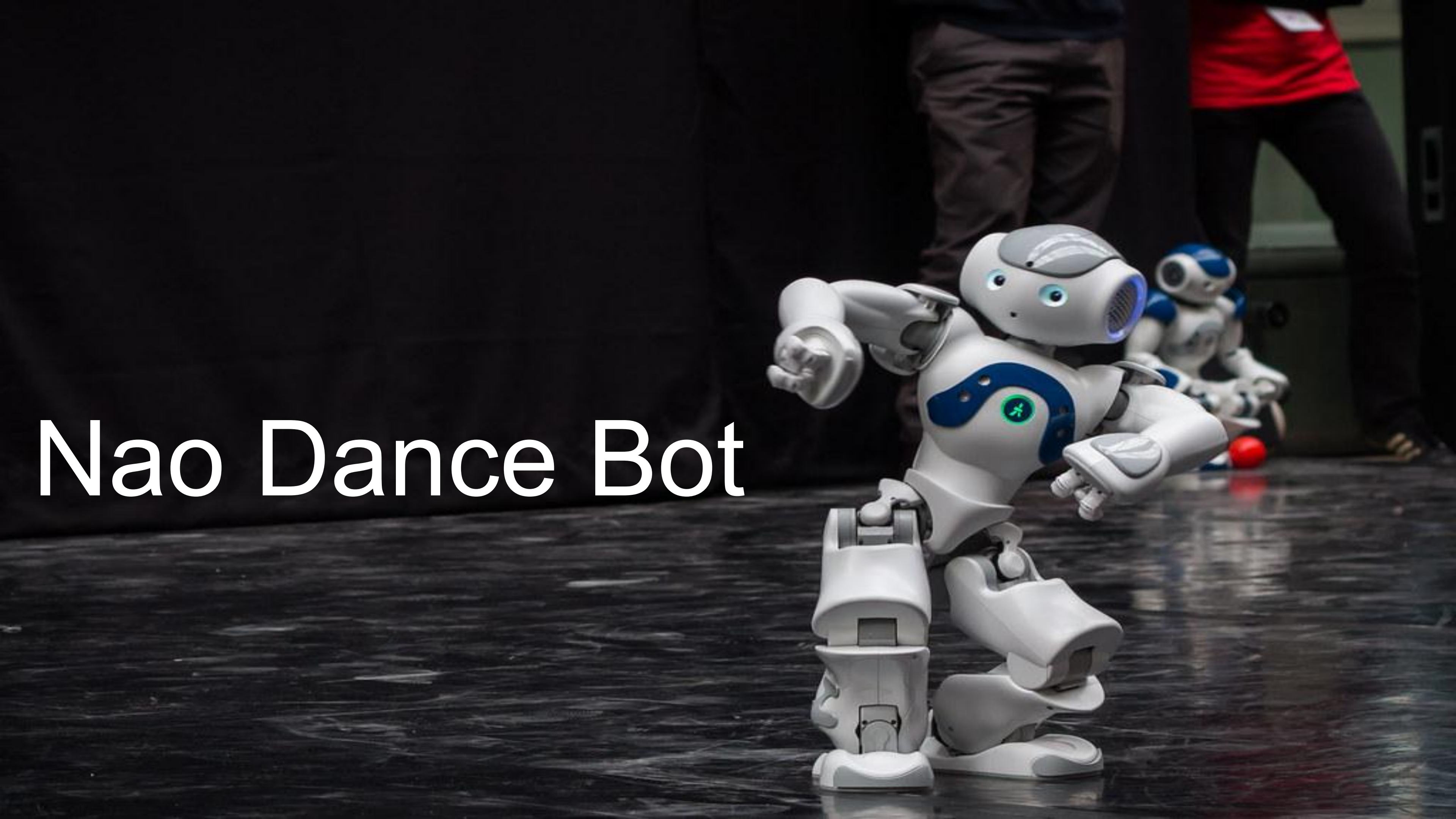


Nao Dance Bot

Nao can distinguish songs and perform the corresponding dance if it hears a song played in the room

Steps

1. Nao records surrounding sounds and sends recording to host
2. Recording is compared to songs from our database
   - ➔ Get matching song
3. Load corresponding dance based on pre-defined keyframes
4. Nao performs the dance

Step 1 - Recording

- Approach: Capture bytes recorded by Nao’s microphone using ALAudioDevice’s subscribe function

```
audio = naoqi.ALProxy("ALAudioDevice", self.strNaoIp, self.naoPort)
audio.subscribe(self.getName())
```

- We then save those bytes in a raw output file and convert the raw bytes into a .wav file

```
with open("./out.raw", "rb") as inp_f:
    data = inp_f.read()
    out_f = wave.open("./out.wav", "wb")
    out_f.writeframesraw(data)
```

Step 2 - Song Comparison (Python 2 to 3 interface)

- Tested own implementation of sound-recognition and multiple libraries
  - ➔ Settled on library called “abracadabra” - a sound recogniser written in Python
- Challenge: Abracadabra uses python 3, but Nao uses python 2
- Solution: Run python 3 script from python 2 using a custom interface

```
# Command to run the Python 3 script
cmd = [self.python_path, self.script_path, function_name] + list(args)

# Run the command and capture the output
output = subprocess.check_output(cmd)

# Decode the output from bytes to string
result = output.decode('utf-8').strip()
```

Step 2 - Song Comparison

- Python 2 call to Python 3 runner

```
def recognize_from_file(self):
    ...
    runner = python3_runner.PythonRunner(python_path, script_path)
    result = runner.run_script('recognise', 'out.wav').splitlines()
    result = {
        'song_name': result[0].encode('ascii', 'replace'),
        'score': result[1].encode('ascii', 'replace')
    }
    return result
```

```
song_name = self.recognize_from_file()
```

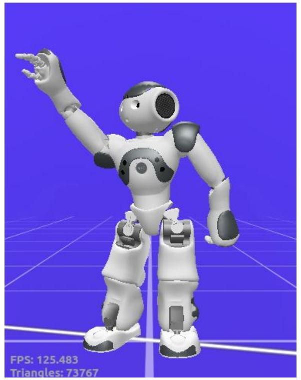
- Abracadabra song recognition inside Python 3 script

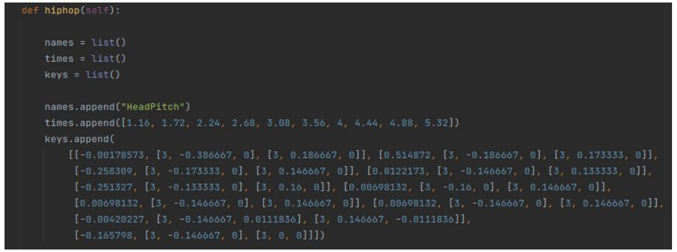
```
def recognise(...):
    print(abracadabra.recognise.recognise_song(filepath)[-1])
```

Step 3 - Dance Room

- Define keyframes using Choreographe software
- Export keyframes into python code
- Load correct pre-defined keyframes based on recognized song

```
def load_dance(self, song_name):
  ...
if song_name == "Macarena":
   speed = 0.4
names, times, keys = self.macarena()
elif (song_name == "YMCA"):
   names, times, keys = self.ymca()
  .....
for i in range(len(times)):
   for j in range(len(times[i])):
       times[i][j] = times[i][j] * (1 / speed)
return names, times, keys
```





```
def hiphop(self):

    names = list()
    times = list()
    keys = list()

    names.append("HeadPitch")
    times.append([1.16, 1.72, 2.24, 2.68, 3.08, 3.56, 4, 4.44, 4.88, 5.32])
    keys.append(
        [[-0.00178573, [3, -0.386667, 0], [3, 0.186667, 0]], [0.514872, [3, -0.186667, 0], [3, 0.173333, 0]],
         [-0.258309, [3, -0.173333, 0], [3, 0.146667, 0]], [0.0122173, [3, -0.146667, 0], [3, 0.133333, 0]],
         [-0.251327, [3, -0.133333, 0], [3, 0.16, 0]], [0.00698132, [3, -0.16, 0], [3, 0.146667, 0]],
         [0.00698132, [3, -0.146667, 0], [3, 0.146667, 0]], [0.00698132, [3, -0.146667, 0], [3, 0.146667, 0]],
         [-0.00420227, [3, -0.146667, 0.0111836], [3, 0.146667, -0.0111836]],
         [-0.165798, [3, -0.146667, 0], [3, 0, 0]]])
```

Step 4: Dance Performance

- Use of ALMotion module
- Call angleInterpolationBezier function with keyframes from Dance Room

```
def perform_dance_from_keyframes (self, names, times, keys):
    self.motion.wakeUp()
    self.motion.moveInit()
    self.motion.setStiffnesses("Body", 0.5)
    self.motion.angleInterpolationBezier(names, times, keys)
    self.posture.goToPosture("StandInit", 0.5)
```

Bringing it all together

```python
def dance_loop(self):
    while True:
        self.danceRoom.posture.goToPosture("StandInit", 0.5)
        self.start_listening()
        time.sleep(self.seconds)
        self.stop_listening()

def stop_listening(self):
    ...
    song_info = self.recognize_from_file()
    self.dance(song_info)

def dance(self, song_info):
    self.tts.say("Song recognized, dancing to " + song_info.get('song_name'))
    time.sleep(1)
    if song_info.get('score') < 3:
        self.tts.say("Not recognizing the song, please try again")
        time.sleep(1)
    else: self.danceRoom.perform_dance(song_info.get('song_name'))
```

Challenges along the way

- First tried using ALAudioRecorder instead of reading bytes from ALAudioDevice but recordings were only saved on Nao locally
- Tried many different approaches for song recognition until we found the “abracadabra” library which worked reliably, but it is a python 3 library
  - ➔ call python 3 script from python 2 using cmd output parsing
- Limited testing capabilities due to many groups testing on one robot
- Dances created in Choreographe behave differently in simulation than on real robot because robot can’t fall over in simulation
  - ➔ Add ability to adjust speed to each dance individually for fine-tuning
- Robot would recognize “random” song if no song was playing
  - ➔ Adjust abracadabra library to return score and set cutoff below minimum score