## Vision: GoPiGo Robot's Camera

Learning Objectives:

- Understand how the camera module and vision APIs work.
- Learn how to take a picture using the camera module.
- Learn how to locate a picture file on the Raspberry Pi using the VNC.
- Understand how to classify a picture using an image recognition API (Inception-v3).
- Write and edit snippets of Python code to take pictures, classify objects and play sounds.

### Background

TensorFlow™ is a library used to recognize an object in an image.

Additional Note: TensorFlow™ is an (open source) library used to quickly and easily build a neural network. A neural network is used to recognize patterns in images, and is modelled in the way the human brain works, with neurons connected to each other by synapses. Inception is the model included with Tensorflow, which is a Convolutional Neural Network (CNN) that allows the computer to recognize objects in an image. Inception was trained to identify 1000 classes.
Tensorflow (https://www.tensorflow.org/)
(CNN) called inception-v3 (https://www.tensorflow.org/tutorials/image_recognition

### Workflow

1. Call init_eyes() for initialization purposes. This is done once per program run.
2. Call snap() to take a picture with the camera (takes 1~2 seconds). This takes a picture and writes the image data to the file `tmp.jpeg` in `/tmp` directory i.e., `/tmp/tmp.jpeg` is created or overwritten.
3. Call see() to classify the image `/tmp/tmp.jpeg` (takes approximately 7 seconds). This method returns two things, a `label` and a `score`. The `score` (between 0 and 1) is the accuracy probability of the identification of the object in the `label`. The label is an array with a single string that represents the object that was identified e.g. ['zebra']. The string can be accessed as `label[0]`. Some labels are comma separated lists, for example, `'mobile phone, cellular phone, cell'`. Simple functions like `split` and `strip` can be used to

handle these cases.

**Note:**
Ensure your object is about 6 inches away from the camera, centered and has a white sheet of paper as a background to get the most accurate results.

**Example Code**

```
from UWIGoPiGo import *
init_eyes()
snap()
label, score = see() #For instance label[0] is ‘mobile phone, cellular phone, cell’
suggestions = [x.strip() for x in label[0].split(',')]
#suggestions contains the list of identified objects. We can then iterate through the suggestions list to check for a specific object
for guess in suggestions:
      if guess == "cell":
            #Do something
      else:
#cell not found
```

**Note - To locate your image file on the VNC:**
Go to the file manager. Look for the “tmp” folder. All pictures taken are stored here. Only the last image snapped is kept due to space constraints.

**Example 1:** Show and Tell – Your robot is curious about the world.

In this example, your bot will ask the user for a picture of an object in the real world. Your bot will then snap a photo of the picture. Your bot will then classify the image and tell you what object you have presented, as well as, the probability of its guess. Your bot has a list of objects that will be used for randomly selecting one object.

```
from UWIGoPiGo import *
import time
import os
import random
from __future__ import print_function

#A list is made containing objects to be drawn
lists = ["a pen", "a cellphone", "a computer", "a computer mouse", "some eyeglasses"]

#Randomly choose an item from the array “lists”
rand = random.choice(lists)

#espeak is a library that allows the robot to talk
#-ven+f4 -p100 -s130 is used to change the intonation, pitch and rate of the voice
os.system("espeak -ven+f4 -p100 -s130 'Show me %s'" % rand)
print('Show me a %s' % rand)

snap() #take a picture of the object
```

```
result, score = see() #classify the image, ‘result’ contains the image
identified and ‘score’ the probability of the identification

print(result, score)
os.system("espeak -ven+f4 -p100 -s130 'OH I think it’s %s ' " % result )
print('OH it is a %s ' % result )
```

**Example 2: Space Disaster**

In this example, we are going to build a game, which is described as follows:

The player is lost at space in a damaged spaceship. Due to the severe malfunctions, the ship’s computer can no longer listen to the user’s commands. However, the ship's computer can still see them. The ship's manual has a collection of predefined objects which correspond to different repair instructions. The players must now present these objects to the camera before the timer ends in order to fix the multiple errors and prevent their ship from blowing up.

```
from UWIGoPiGo import *
import time
import os
import random
import pygame
from __future__ import print_function

# Here, PyGame is used to allow us to play .wav sound files. It is initiated
first for responsiveness
pygame.mixer.init()
explosion = pygame.mixer.Sound("/tmp/explosion.wav")

#We need to have a list that contains the mapping of an object to the repair
instruction. For example, if the player points the camera to a cabinet, then the
computer will run the “Hull Breach” instruction.
lists = [   {'watch':'Hull Breach'},
            {'ballpoint pen':'Life Support'},
            {'computer':'Central Processing Unit'},
            {'bottle':'Water Purifier'},
            {'glasses':'Visual Cortex'},
            {'cellphone ':'Power Systems'}
        ]
# Additional details: Dictionaries are objects in Python that have keys and
values. The key, such as ‘drill,’ can be a number or string and will correspond
to a value, ‘Hull Breach.’ They are written with curly braces. Here, an array is
made of dictionaries by using square braces.

correct = 0 #initially, the player has no correct drawings
```

```
random.shuffle(lists) # Shuffles the array to make every run randomized

# Takes the first 3 elements of the array and places it into the new array run
run = lists[:3]

# We now print the problems that require fixing in the ship (the first 3 stored
in run)
print ("Warning. Systems Critical.")
for i in run:
      print(i.values())
# Here we ask the player to print the key of the associated instruction
for obj in run:
os.system("espeak -ven+f4 -a200 -p100 -s130 'Show me a %s You have    10
        seconds'" % obj.keys())
      print('Show me a ‘ + obj.keys())

      # Here we give the player 10 seconds to obtain the picture and we add some
      # audio as well
      num = 10 # Amount of time allotted for the player to draw the key
      for i in range(num):
            os.system("espeak -ven+f4 -a200 -p100 -s130 '%d'" % (num-i))
            print(num-i)

    # Here we snap the picture, classify it, and respond with audio and print
    snap()
    result, score = see()
    print(result, score)
    os.system("espeak -ven+f4 -a200 -p100 -s130 'You showed me a %s ' " % result
)
    print('You showed me a %s ' % result)

# If the result from see()  is the same as the required object, then the
    # score is incremented
    if obj.keys() in result:
        correct+=1
        print("Initiating repairs on %s" % obj.values())
        os.system("espeak -ven+f4 -a200 -p100 -s130 'Initiating repairs on %s' "
        % obj.values() )

# Otherwise the computer says that it is invalid and the required repair wasn’t
    made. No points are gained.
    else:
        print("Cannot proceed with repairs.")
        os.system("espeak -ven+f4 -a200 -p100 -s130 'Cannot proceed with
        repairs.' ")
        sleep(1)
```

```
# Here we determine whether the player has won. The 3 objects have been
displayed and classified. We let the player win if one of more drawings were
correct. Again, audio and print are used.
if correct >=1:
    os.system("espeak -ven+f4 -a200 -p100 -s130 'You are saved! ' ")
    print('You are saved!')
else:
    explosion.play()
    time.sleep(4)
    explosion.stop()
```