Rajesh kumar - 11805020 – K18JF

Augmented reality application **saltwashar** final report

SaltwashAR is a Python Augmented Reality application.

SaltwashAR uses OpenCV computer vision to detect a 2D marker in a webcam, and OpenGL graphics library to render a 3D robot upon the marker.

We can interact with the robots:

- ask a robot to search the web, translate phrases, help us gamble or practice acting, or provide a world weather report
- let the robot learn a card game, classify iris flowers and audio, give a slideshow, be a talking calculator or a mixing desk
- a robot can watch TV, detect shapes, respond to our hand gestures, read printed words or just be happy!

Contents

- Optical Character Recognition
- Happy Colour
- Play Your Cards Right
- Hand Gesture
- Browser

Application files

- calibration
- images
- markers
- scripts
- scripts/features

Further information

Software requirements

Here are core dependencies, each with their recommended version number:

- Python 2.7.11
- PyOpenGL 3.1.1b1
- OpenCV 2.4.9
- NumPy 1.10.1
- PIL 1.1.7

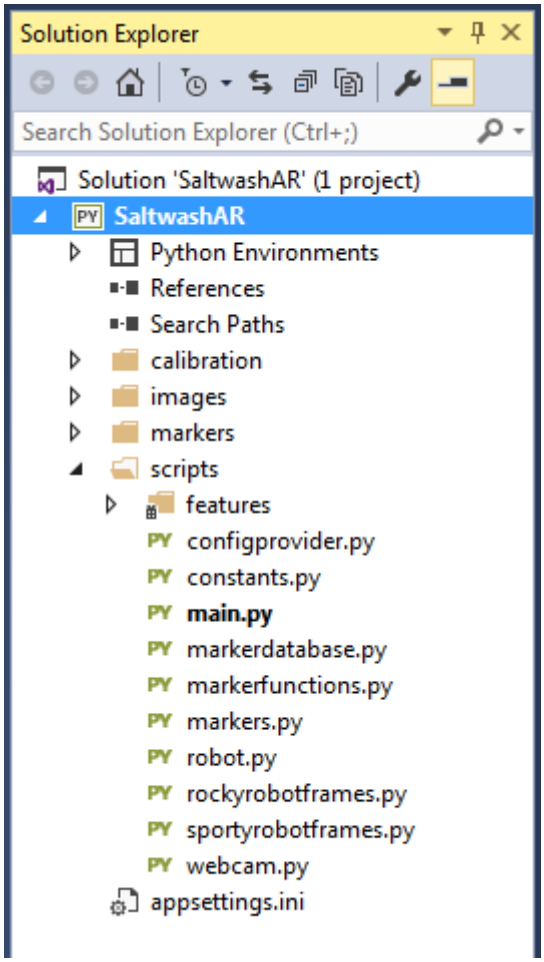
Here are the feature dependencies (optional), each with their recommended version number:

- BeautifulSoup 4.4.1
- SpeechRecognition 3.1.3
- pyttsx 1.1
- PyBrain 0.3.3
- PyTesser 0.0.1
- Pygame 1.9.2a0
- sklearn 0.17
- scipy 0.16.0

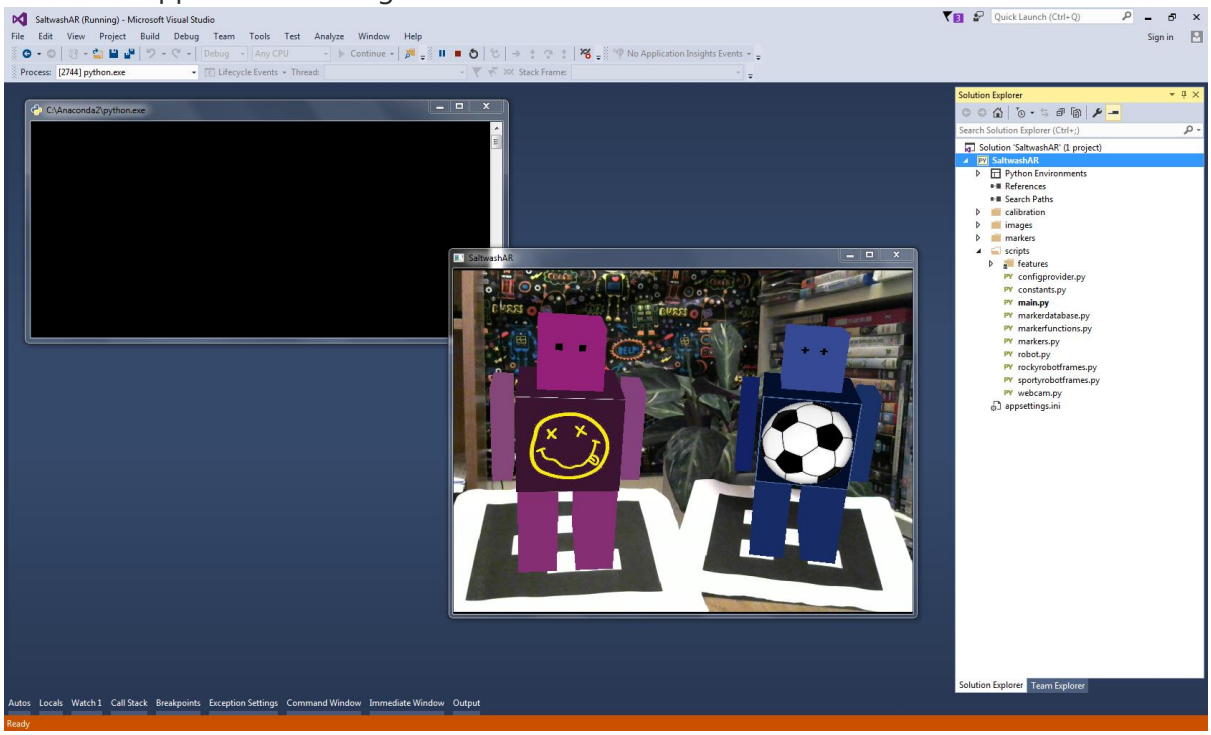
Setting up the application

I am running the application on my Windows 10 64-bit PC, using Python Tools for Visual Studio with the Anaconda (Python 2.7 Windows 64-bit) interpreter.

Here's the folder setup, with the Visual Studio project file:

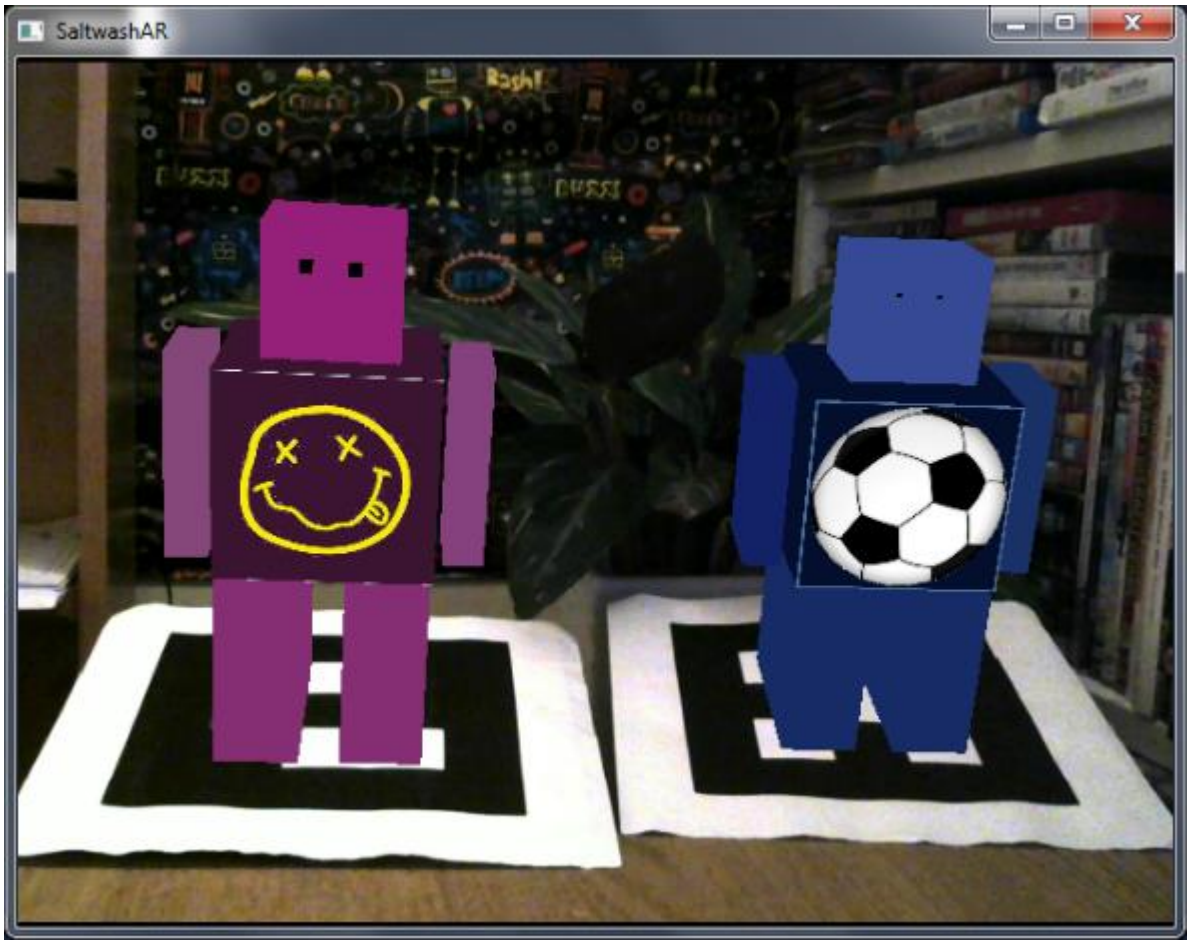


Here's the application running in Visual Studio



Out of the box

Out of the box, SaltwashAR lets us render 3D robots on 2D markers:



Simply open the SaltwashAR markers folder and print out the two marker images onto paper.

Next, attached a webcam to your PC and start the SaltwashAR application.

Hold the 2D markers in front of the webcam and watch the robots appear!

Features

Features let us interact with the robots.

Note: a robot needs to be facing the webcam for us to interact with it.

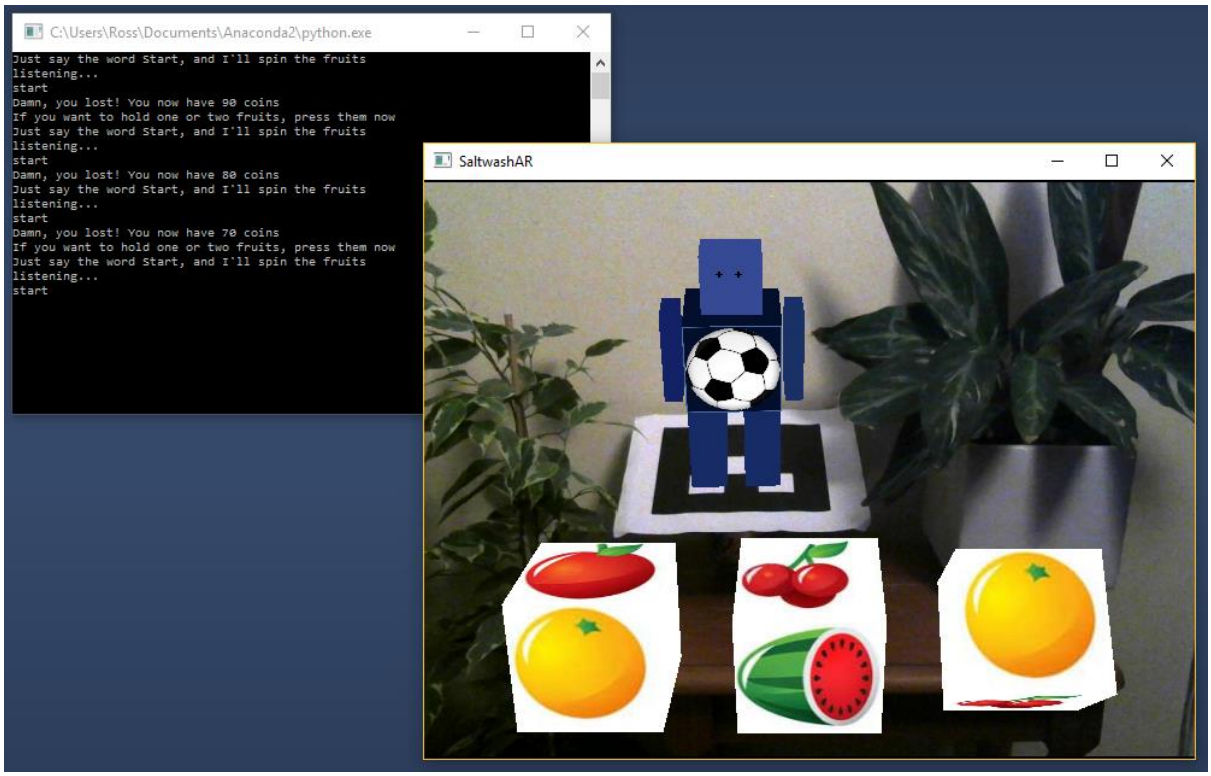
Fruit Machine feature

Install the Fruit Machine dependencies:

- SpeechRecognition 3.1.3
- pyttsx 1.1

Set FruitMachine=True in the SaltwashAR appsettings.ini file.

Sporty Robot will now help you play the fruit machine:



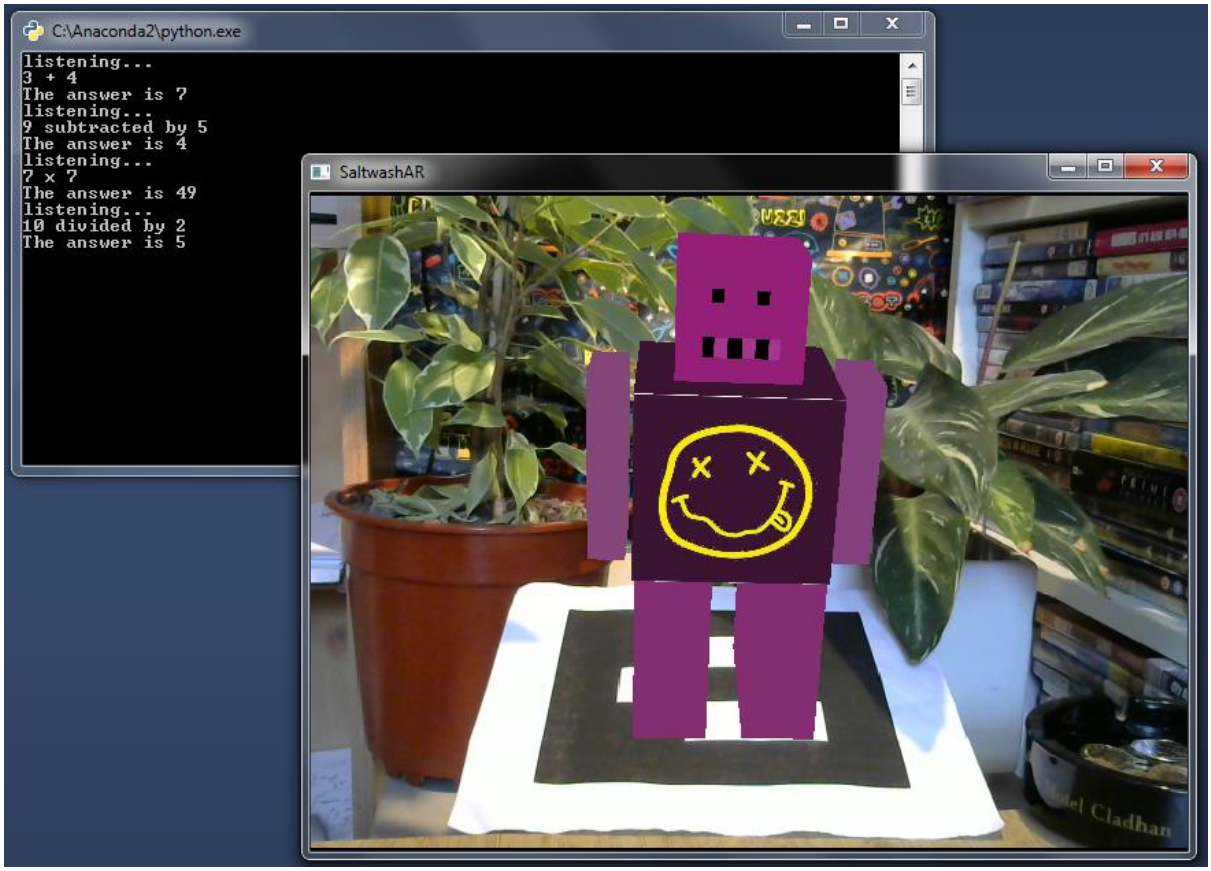
Calculator feature

Install the Calculator dependencies:

- SpeechRecognition 3.1.3
- pyttsx 1.1

Set Calculator=True in the SaltwashAR appsettings.ini file.

The robot is now a talking calculator:



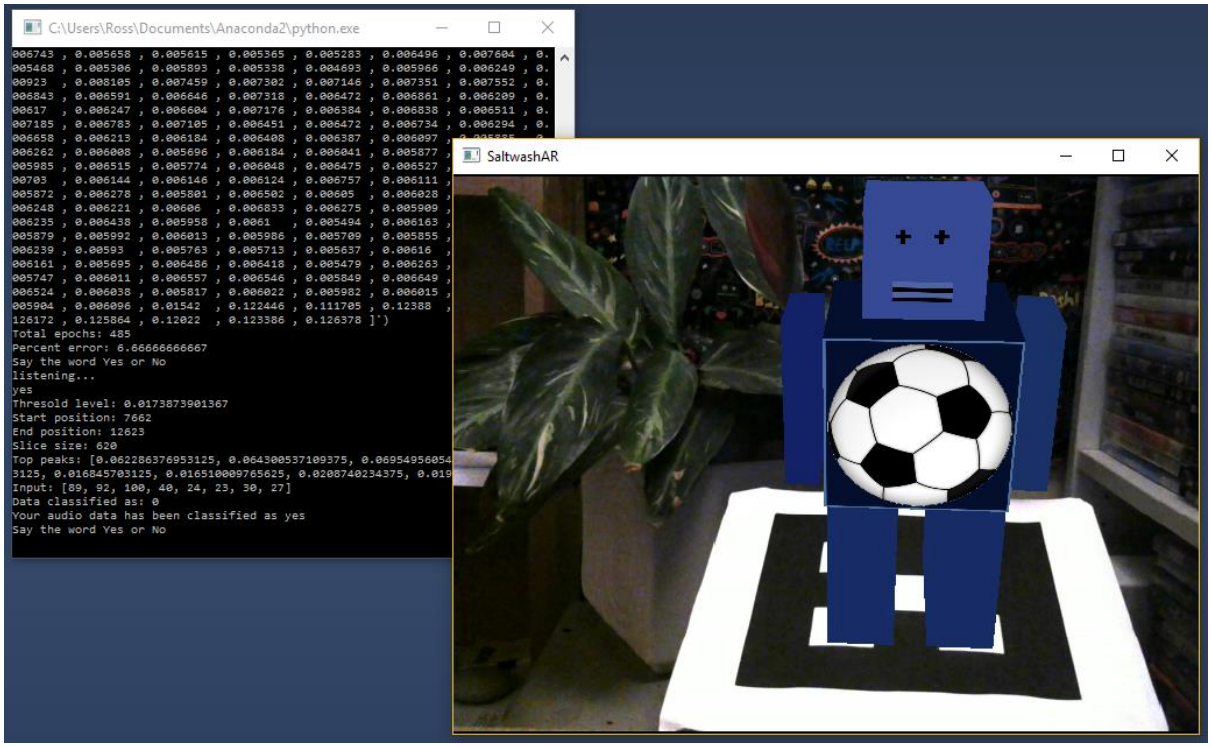
Audio Classifier feature

Install the Audio Classifier dependencies:

- SpeechRecognition 3.1.3
- pyttsx 1.1
- PyBrain 0.3.3
- scipy 0.16.0

Set AudioClassifier=True in the SaltwashAR appsettings.ini file.

The robot will now use a neural network to classify a spoken word as Yes or No:



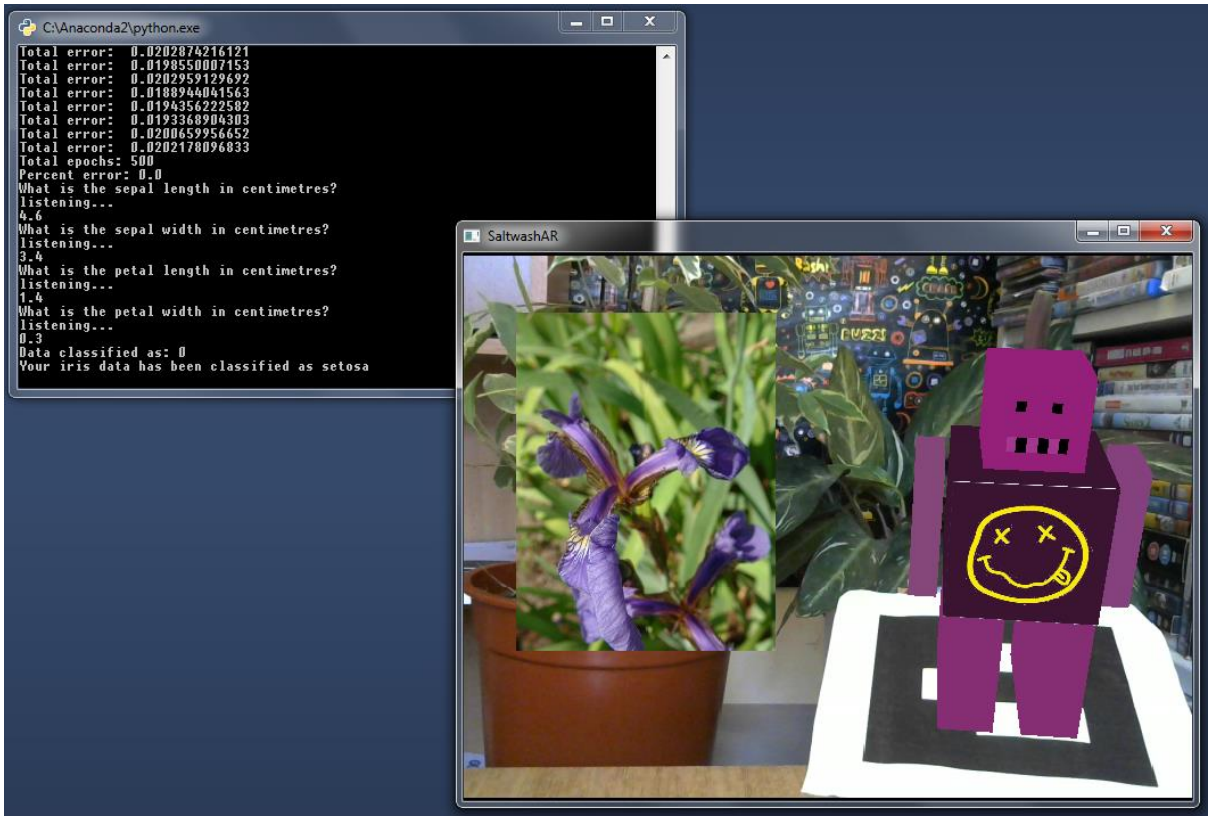
Iris Classifier feature

Install the Iris Classifier dependencies:

- SpeechRecognition 3.1.3
- pyttsx 1.1
- PyBrain 0.3.3
- sklearn 0.17

Set IrisClassifier=True in the SaltwashAR appsettings.ini file.

The robot will now use a neural network to classify an Iris flower into one of three species:



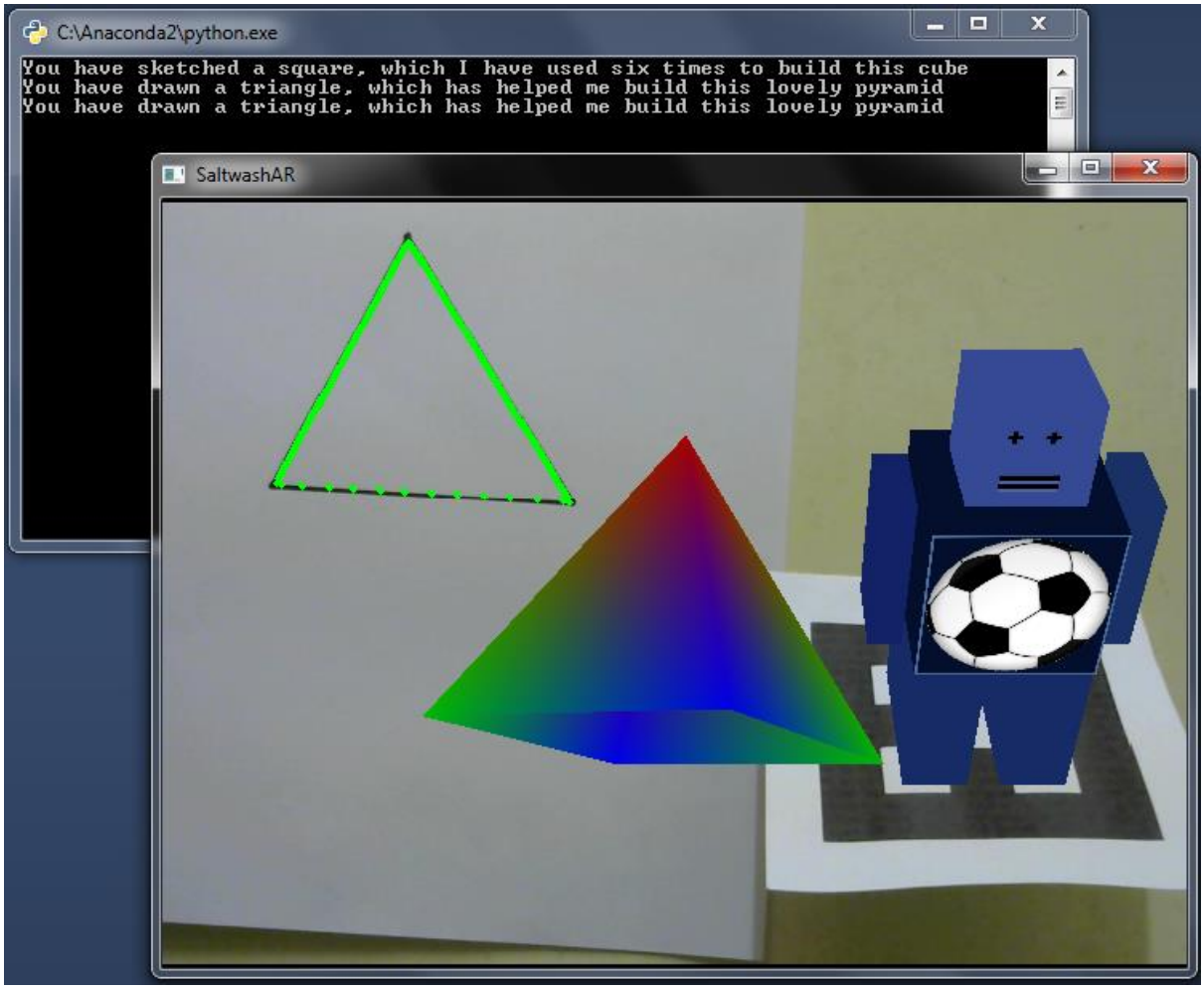
Shapes feature

Install the Shapes dependencies:

- pyttsx 1.1

Set Shapes=True in the SaltwashAR appsettings.ini file.

The robot will now detect shapes:



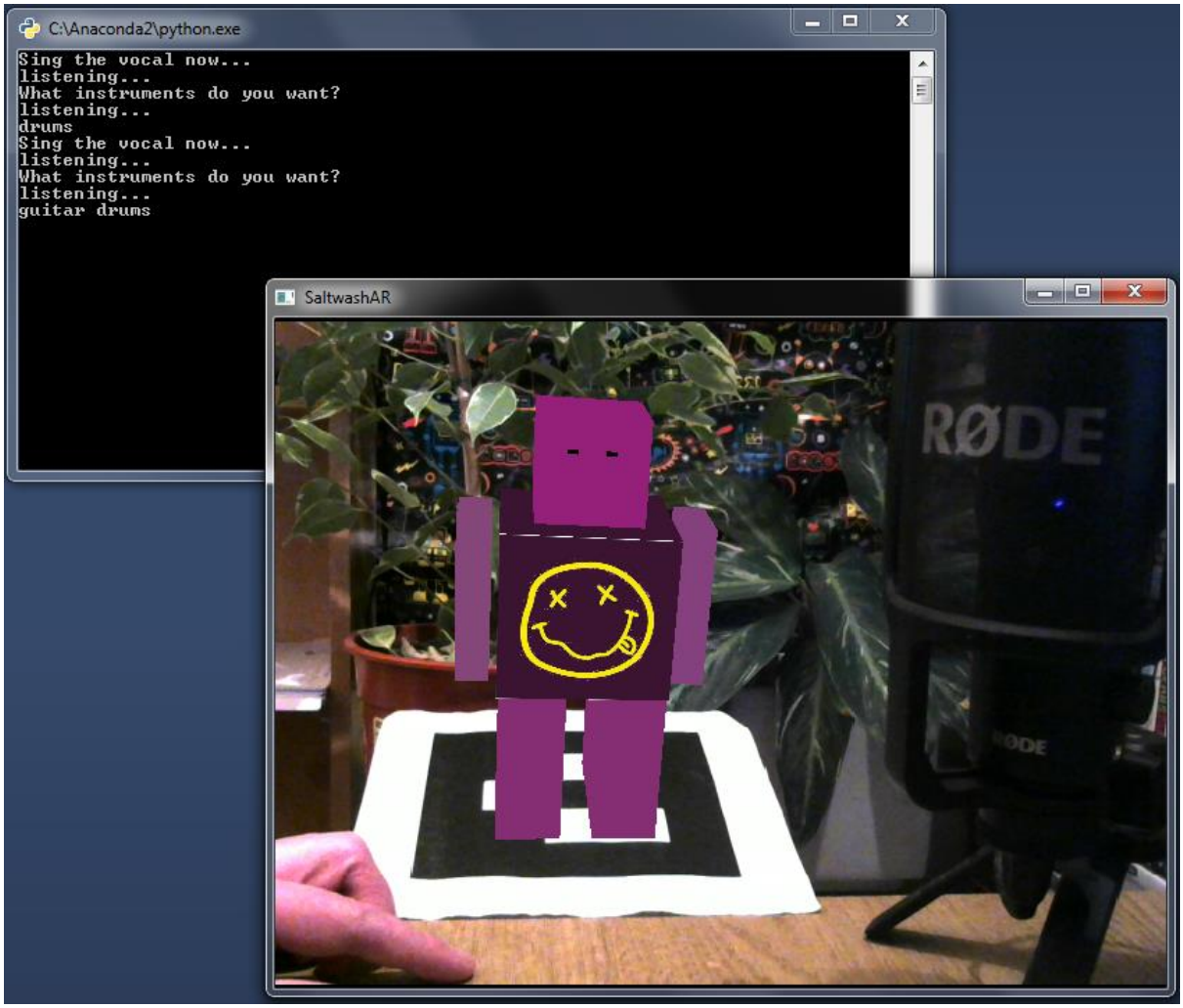
Mixing Desk feature

Install the Mixing Desk dependencies:

- SpeechRecognition 3.1.3
- pyttsx 1.1
- Pygame 1.9.2a0

Set MixingDesk=True in the SaltwashAR appsettings.ini file.

Rocky Robot is now a mixing desk:



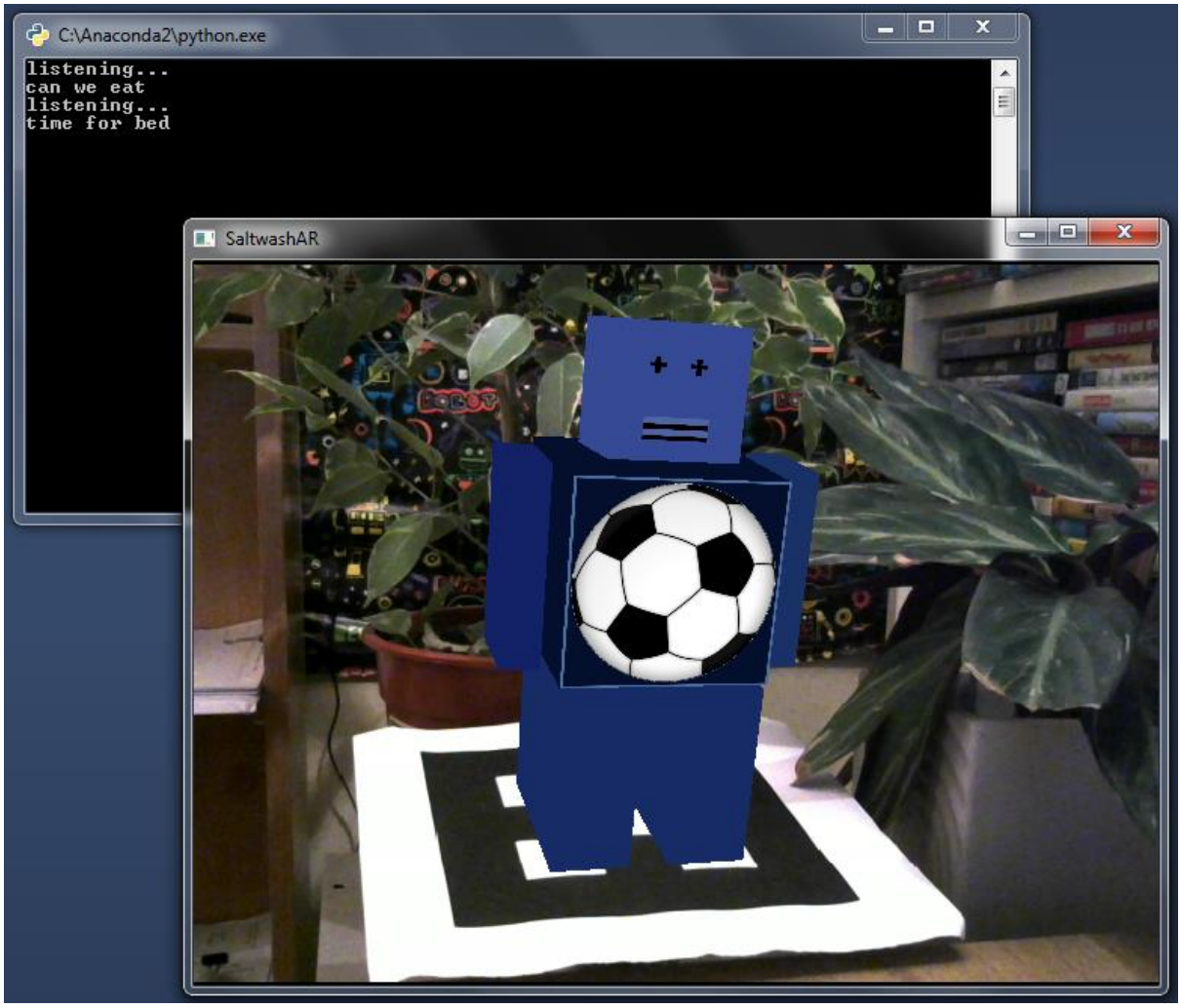
Phrase Translation feature

Install the Phrase Translation dependencies:

- SpeechRecognition 3.1.3
- Pygame 1.9.2a0

Set PhraseTranslation=True in the SaltwashAR appsettings.ini file.

Sporty Robot can now translate your phrases:



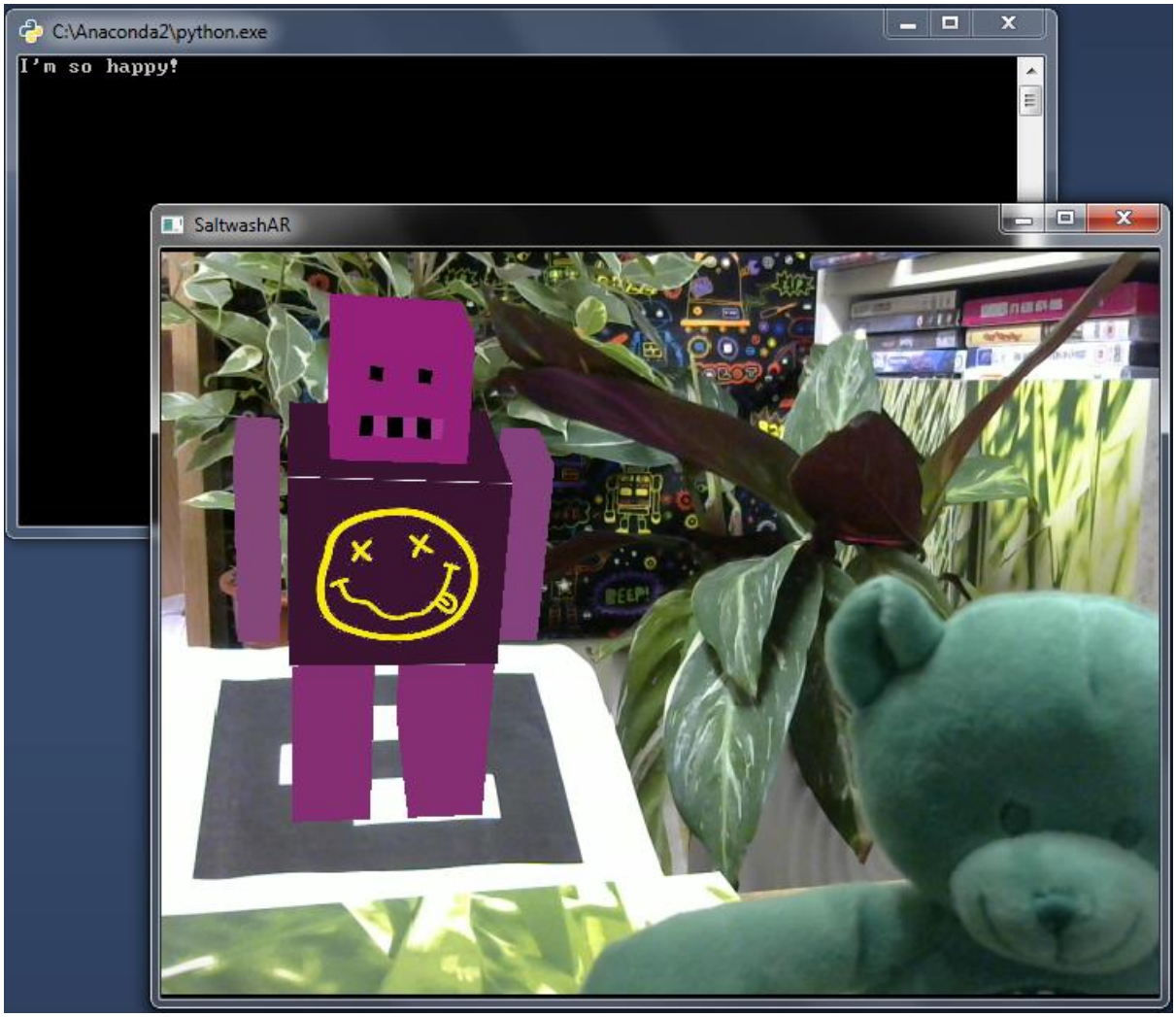
Happy Colour feature

Install the Happy Colour dependencies:

- pyttsx 1.1

Set HappyColour=True in the SaltwashAR appsettings.ini file.

You can now use colour to help Rocky Robot be happy:



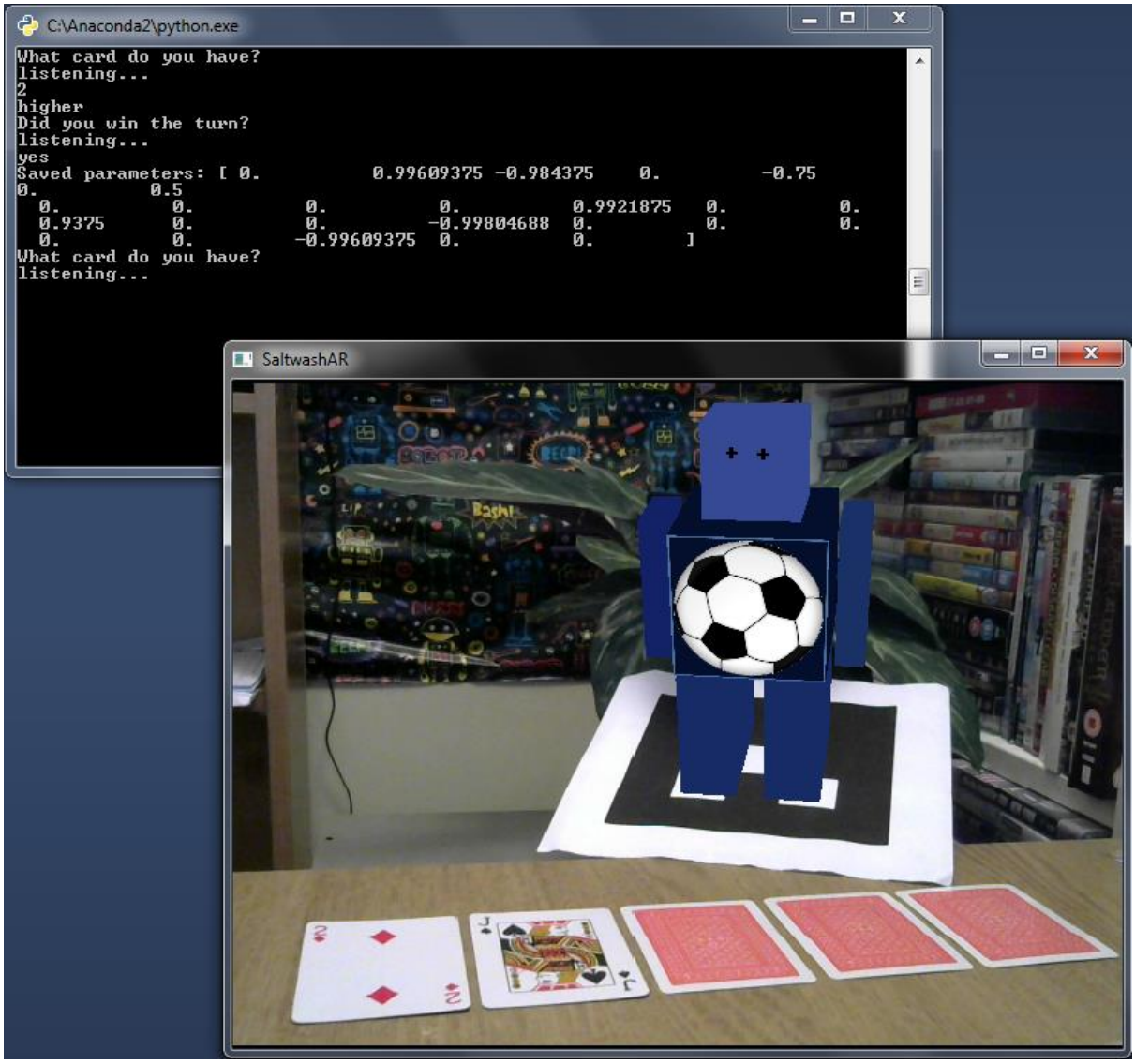
Play Your Cards Right feature

Install the Play Your Cards Right dependencies:

- SpeechRecognition 3.1.3
- pyttsx 1.1
- PyBrain 0.3.3

Set PlayYourCardsRight=True in the SaltwashAR appsettings.ini file.

You can now help Sporty Robot to learn a card game:



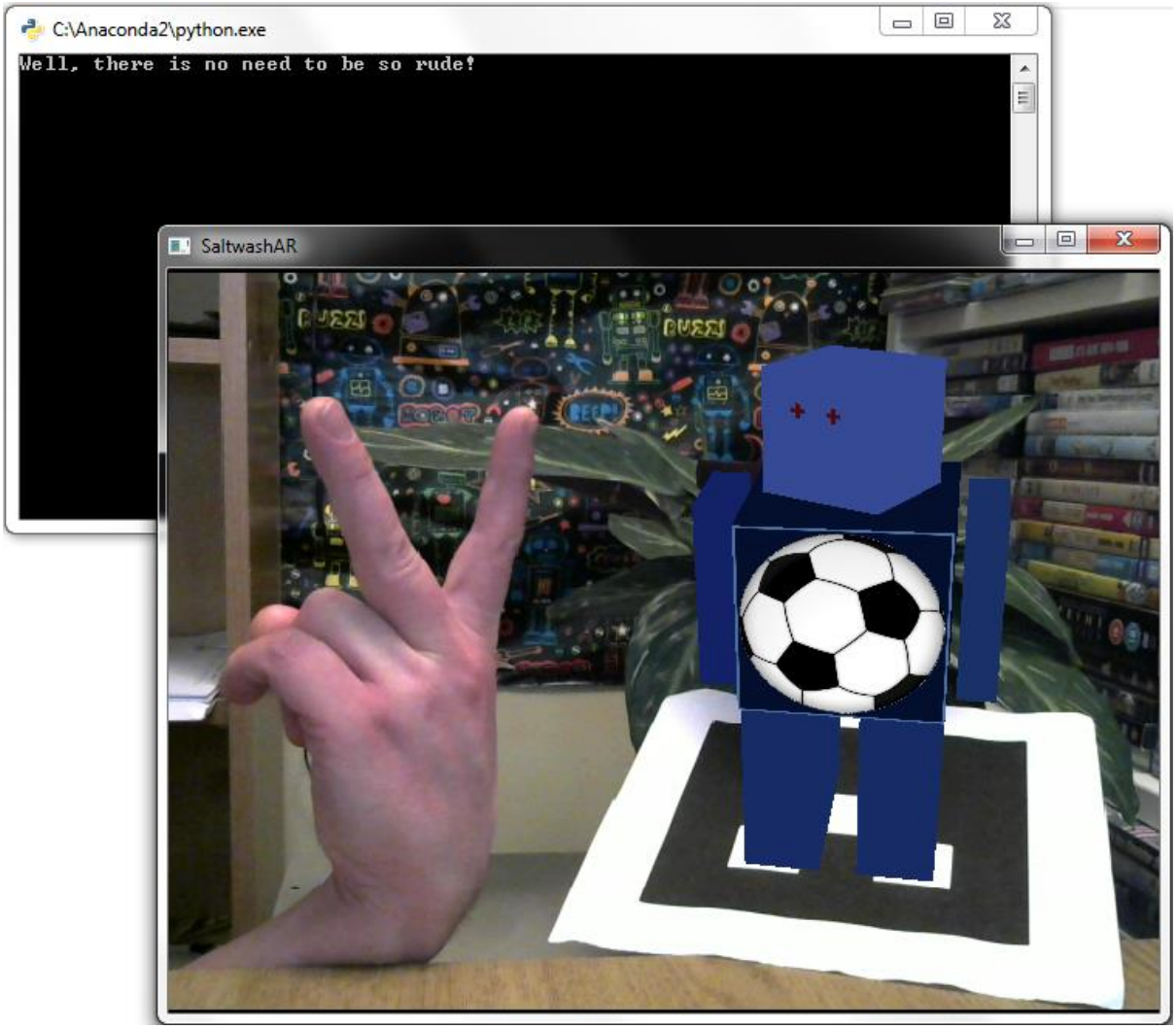
Hand Gesture feature

Install the Hand Gesture dependencies:

- pyttsx 1.1

Set HandGesture=True in the SaltwashAR appsettings.ini file.

The robot will now respond to Okay and Vicky hand gestures



Application files

calibration

The webcam_calibration_ouput.npz file is specific to my webcam - check out my post OpenCV Camera Calibration and Pose Estimation using Python on how to calibrate your own camera.

Note: main.py has the OpenGL command gluPerspective(33.7, 1.3, 0.1, 100.0) - check out the comments on my post Augmented Reality with 3D object rotation on how to calculate the field of view angle and aspect ratio specific to your camera.

images

The image files are used for the chest and face of each robot.

markers

The two image files are our 2D markers. Print them out onto paper and hold them in front of your webcam - and watch the robots appear!

scripts

configprovider.py - supplies configuration from the appsettings.ini file to the application. If Animation=True then our robots are animated (default is True). If Acting=True then the robots can help us practice acting (default is False). And so on.

constants.py - application-wide constants.

markerdatabase.py - holds records for our 2D markers, plus a method to match to the pattern detected in our webcam.

markerfunctions.py - basic functions to help detect our 2D markers.

markers.py - the core stages of 2D marker detection.

main.py - the main program, which orchestrates detection of 2D markers, rendering of robots and interaction with robots.

robot.py - represents a robot, including all frames of animation.

rockyrobotframes.py - auto-generated code for rocky robot animation

sportyrobotframes.py - auto-generated code for sporty robot animation

webcam.py - runs its process in a thread so as not to block the main application. Provides frames from our webcam.

scripts/features

features.py - orchestrates requests from the main application to the features, and responses from the features to the main application.

acting/acting.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, to help you practice acting.

audioclassifier/audioclassifier.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, and classifies audio data with a neural network.

audioclassifier/neuraldata.py - functions to create, save and load an audio data set.

audioclassifier/neuralnetwork.py - functions to build and use a PyBrain Neural Network.

base/emotion.py - base class functionality for emotion.

base/feature.py - base class functionality for threading.

base/speaking.py - base class functionality for Text To Speech.

browser/browser.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, and Beautiful Soup to load web content.

browser/searchdatabase.py - holds records for our online content, plus a method to match to the search category and term.

calculator/calculator.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, and calculates a mathematical expression.

fruitmachine/detection.py - handles OpenCV motion detection.

fruitmachine/fruitmachine.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, and lets you play a fruit machine.

fruitmachine/reels.py - controls and draws the fruit machine reels.

handgesture/handgesture.py - runs its process in a thread so as not to block the main application. Handles OpenCV Haar Feature-based Cascade Classifiers for 'Okay' and 'Vicky' hand gestures, and Text To Speech for the robot's response.

happycolour/happycolour.py - runs its process in a thread so as not to block the main application. Handles OpenCV colour detection, and Text To Speech for the robot's response.

irisclassifier/irisclassifier.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, and classifies Iris data with a neural network.

irisclassifier/neuralnetwork.py - functions to build and use a PyBrain Neural Network.

mixingdesk/mixingdesk.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, and Pygame to mix guitar and drums with vocals.

opticalcharacterrecognition/opticalcharacterrecognition.py - runs its process in a thread so as not to block the main application. Handles PyTesser Optical Character Recognition, and Text To Speech for the robot's response.

phrasetranslation/phrasetranslation.py - runs its process in a thread so as not to block the main application. Handles Speech To Text and Pygame to let the robot translate phrases.

playyourcardsright/gameenvironment.py - PyBrain Reinforcement Learning Environment implementation.

playyourcardsright/gameinteraction.py - provides Text To Speech and Speech To Text communication with the robot.

playyourcardsright/gametable.py - PyBrain Reinforcement Learning ActionValueTable implementation, allowing parameters to be loaded and saved.

playyourcardsright/gametask.py - PyBrain Reinforcement Learning Task implementation.

playyourcardsright/playyourcardsright.py - runs its process in a thread so as not to block the main application. Handles Text To Speech and Speech To Text communication with the robot, and PyBrain Reinforcement Learning to let the robot master a card game.

shapes/shapes.py - runs its process in a thread so as not to block the main application. Handles OpenCV shape detection, and Text To Speech in a secondary thread for the robot's response.

shapes/shapesfunctions.py - functions to render an OpenGL pyramid and cube.

slideshow/slideshow.py - runs its process in a thread so as not to block the main application. Handles Text To Speech in a secondary thread, to let the robot talk you through a slideshow.

speechtotext/speechtotext.py - converts Speech To Text (configured for Google Speech Recognition).

television/television.py - runs its process in a thread so as not to block the main application. Handles OpenCV object detection of 2D television marker, to let the robot watch television.

television/televisionfunctions.py - basic functions to help detect our 2D television marker.

texttospeech/texttospeech.py - converts Text To Speech.