Sound Communication: Lab II

RealTime Machine Learning Modeling (II):
Image to Sound

Sergio Giraldo
Universitat Pompeu Fabra

2017-2018

Introduction

The objective of this lab is to follow the framework presented in Lab 1. In this case, we will implement a communication system which receives as an input features obtained form face expressions recognition, and maps it into sound. You will be provided with a simple implementation, and your task is to build a system on top of it.

1 Background Theory Concepts

1.1 Image processing

Images have always been close to humans along history. Ancient civilizations used images as written communication. In the Middle Ages, images in the temples were used to transmit the Holy Script to the ordinary people. Renaissance represents a turning point in the history of images. Perspective appears in pictures, light and shadow achieve new meaning, and the basis of composition in photography, which will appear several centuries later, are established. Nowadays, imaging has a great impact, more than ever before, not only in art (photography) or in entertainment (movies), but also in science and engineering. Imaging has become a way of visualizing the physical world, from electron microscopy or astronomy to medical imaging, where the behavior of the body to different physical phenomena is represented as an image. For example, X-rays or Computed Tomography images show the capacity of body tissues of absorbing radiation, while echographic images show how ultrasounds are reflected by different parts of the body, and diffusion tensor images show the diffusivity of water molecules in a structured tissue, such as the brain or the muscles.

Digital image processing is a subcategory of the digital signal processing research domain. It consist of computer algorithms to developed perform image processing on digital images. It aims at solving several typical problems within the signal processing domain. For instance, in the acquisition process, images can be altered by the instrumentation used and they must then be processed to improve their quality. Similarly the images received through a communication channel are usually noisy.

1.2 Digital Image encoding

An image can be formally defined as a function (see Figure 1):

$$(x, y) : \omega \rightarrow \Re^2 \tag{1}$$

where $\omega \subset \Re^2$ is the domain where the image is defined, usually a rectangle. If the image is represented on a screen, $(x, y) \in \omega$ can be considered as representing the pixels positions, while the scalar values $u(x, y)$ are the grey levels (light intensity) at each pixel. In the mathematical representation black is $u(x, y) = 0$ and white $u(x, y) = 1$. Each intensity or grey level between black and white will be a value within the interval $[0, 1]$.

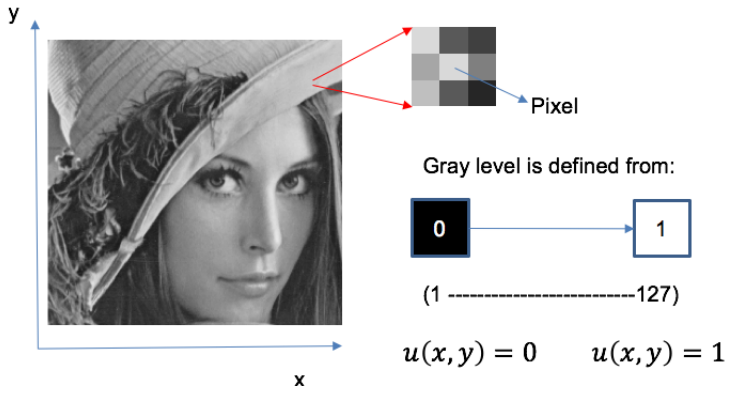

Figure 1: Spectrogram of an audio wave form

In a colour image, $u(x, y)$ is usually an $\Re^3$ vector instead of a scalar. It is made of the levels of R red, G green and B blue. There are other 3D representation systems of colour, such as com HSL (Hue, Saturation, Lightness).

1.3 Object recognition in digital images

In the field of computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat in different view points, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view. This task is still a challenge for computer vision systems. Many approaches to the task have been implemented over multiple decades:

In general methodologies for object recognition can be grouped into appearance based methods, and feature based methods. The former are based on example images or templates where similar patterns in terms of lighting, color and shape are searched. The later use feature extracted form both the objects and the images, and aim to find similar matches among them. Machine learning methods such as Knn are used to find the aforementioned similarities, using pixel based features.

2 Practical Work

In this section you will develop a system for automatic classification of face gestures in real time, to later map the predicted class to sounds.

2.1 materials

- Computer with sound card and built-in microphone.
- FaceOsc: a real-time face gesture tracker. `https://github.com/kylemcdonald/ofxFaceTracker/releases`
- Wekinator. You can download wekinator from: `www.wekinator.org`
- Pure Data: A software for graphical computation. You can download from `https://puredata.info/`. You are encouraged to use *Processing or other programing language that facilitates the visualization of data.*

NOTE: Optinal material for camera based recognition approaches that work within the wekinartor framework can be found at: `http://www.wekinator.org/examples/#VideoWebcam`

2.2 Framework

The framework of the expected system is depicted in Figure **??**.

2.2.1 Input

The input of the system will be face recognition descriptors. The FaceOSC framework can send though the OSC port several features, including:

1. Pose
    - center position: /pose/position
    - scale: /pose/scale
    - orientation (which direction you're facing): /pose/orientation
2. Gestures
    - mouth width: /gesture/mouth/width
    - mouth height: /gesture/mouth/height
    - left eyebrow height: /gesture/eyebrow/left
    - right eyebrow height: /gesture/eyebrow/right
    - left eye openness: /gesture/eye/left
    - right eye openness: /gesture/eye/right
    - jaw openness: /gesture/jaw
    - nostril flate: /gesture/nostrils
3. Raw
    - raw points (66 xy-pairs, total 132): /raw

The faceOSC sends OSC data to localhost, port 8338. If you want to change this, see the section about editing the settings.xml file at the faceOSC web site. You can toggle "pose" and "gesture" whether to send the pose and gestural stats over OSC, respectively. "raw" toggles whether to send the raw points of the face mesh (66 XY-points = 132 values). Choose at the feature extractor as many features as you consider. Be aware of how many total inputs is the feature extractor sending.

2.3 Wekinator

Open Wekinator and start a new project. Choose the port number for the input as 8338.

Put the number of inputs wekinator is receiving from the audio feature extractor.

Choose the number of outputs (i.e. models) you want wekinator to train and output. Also set the output port of wekinator at 7005.

Train the models according to the expected output of your system until Wekinator is able to classify different sounds.

2.4 output

Wekinator sends the output of each model trough OSC. At pure data you will be provided with a patch which is able to listen to this port and output the received values to boxes. After, you should map these received classification probabilities values (i.e. the value of the slider) to tiger a sound. For instance, at the provided patch a threshold of 0.5 is set for the output of model one. The "select" box checks its input against the constant value (which is defined by the creation argument). If they match, the first outlet gives "bang", otherwise the input is simply sent through to the second outlet. The "bang" signal is used to tiger the sound. Details of the sound patch will be provided at the lab session.

The aim is then to map different face gestures to different sounds. Also you can consider other extensions within this same concept by using other input devices (or camera inputs). You can also implement audio effects to change its parameters on real time. However, in any case the output should be always sound.

Submission

You have to deliver all the code and/or patches created for the practical work (i.e. wekinator files, and puredata files). You also have to submit as well a written report explaining the framework of your work, inputs, outputs models used. All the files should be submitted in a zip file though the Aula Global.

You will have to present and demonstrate your system at the next class.