**DIGITAL SIGNAL PROCESSING**

Laporan ini merupakan bagian dari Assignment EL5226 – Jaringan Informasi Sosial ITB2015. Source code dan laporan juga tersedia di link github : <https://github.com/soedomoto/ITB2015/tree/EL5226/Assignment 2 - DSP>

**1. AUDIO FINGERPRINTING AND RECOGNITION**

Merekam suara dan mencocokkan dengan potongan suara yang tersedia untuk identifikasi

Audio fingerprinting merupakan suatu teknik identifikasi suara dengan cara memecahnya menjadi beberapa sampel. Sebuah file audio dikonversi menjadi grafik sinus, kemudian sampel diperoleh dengan cara pencarian titik puncaknya. Algoritma pencarian titik puncak ini dikenal dengan FFT (<https://en.wikipedia.org/wiki/Fast_Fourier_transform>). Misalnya sebuah file .mp3 stereo (2 channel) dengan durasi 3 menit akan memiliki sampel :

3 min \* 60 sec \* 44100 samples per sec \* 2 channels = 15,876,000 samples

Kemudian sebanyak 15,876,000 samples tersebut akan disimpan ke dalam sebuah database. Begitu seterusnya untuk setiap file audio yang akan diidentifikasi. Kemudian ketika terdapat sebuah potongan audio, misalnya 10 detik audio .mp3 ingin diidentifikasi siapakah penyanyi dari potongan audio tersebut, maka potongan audio tersebut akan di-sampling, kemudian setiap sampling akan dicocokkan pada database fingerprint. Metode seperti ini seperti yang diterapkan pada beberapa aplikasi pencarian lagu populer seperti Midomi (<http://www.midomi.com/>).

Salah satu tools atau implementasi audio fingerprinting and recognition adalah Dejavu (<https://github.com/worldveil/dejavu>). Dejavu merupakan aplikasi berbasis python yang bekerja dengan algoritma FFT. Dejavu bekerja dengan memanfaatkan database MySQL, akan tetapi untuk meningkatkan portabilitas, sebuah kelas python yang mengimplementasikan interface dejavu.database.Database dibuat agar mendukung database SQLite3 ([dejavu\_sqlite.py](https://github.com/soedomoto/ITB2015/blob/EL5226/Assignment%202%20-%20DSP/Audio%20Fingerprint%20and%20Recognition%20Python/dejavu_sqlite.py)). Berikut penjelasan singkatnya :

**System Dependencies :**

Aplikasi dejavu merupakan aplikasi yang ditulis dengan bahasa Python, sehingga sebenarnya dejavu bersifat platform independent. Tapi dalam kasus ini, dejavu akan dijalankan pada platform berbasis Linux Ubuntu 14.04 Trusty Tahr. Dejavu memerlukan beberapa dependency terhadap aplikasi tertentu :

1. ffmpeg, untuk konversi audio ke .wav format,

2. portaudio, untuk membaca file audio, termasuk membaca audio lewat microphone,

3. audacity, untuk modifikasi file audio (join, splice, resample)

**Python Dependencies :**

1. numpy, untuk membaca signal audio,

2. pydub, python wrapper ffmpeg,

3. matplotlib, untuk menggambarkan spectrogram dan plotting,

4. scipy, untuk pencarian titik puncak dengan algoritma FFT,

5. pyaudio, python wrapper untuk portaudio

**Penjelasan Program :**

Download [Dejavu](https://github.com/worldveil/dejavu), atau modifikasinya dengan menghilangkan dependecy terhadap MySQL disini [Modifikasi Dejavu](https://github.com/soedomoto/ITB2015/tree/EL5226/Assignment%202%20-%20DSP/Audio%20Fingerprint%20and%20Recognition%20Python/dejavu). Dejavu yang telah dimodifikasi memiliki dependency terhadap [dejavu\_sqlite.py](https://github.com/soedomoto/ITB2015/blob/EL5226/Assignment%202%20-%20DSP/Audio%20Fingerprint%20and%20Recognition%20Python/dejavu_sqlite.py). Struktur database yang digunakan dalam dejavu-sqlite adalah :

CREATE TABLE songs (

song\_id integer primary key autoincrement,

song\_name varchar(255) not null,

fingerprinted tinyint default 0,

file\_sha1 binary(10) not null

);

CREATE TABLE fingerprints (

hash binary(10) not null,

song\_id unsigned mediumint not null,

offset unsigned int not null,

UNIQUE(song\_id, offset, hash)

);

Import library yang diperlukan :

from dejavu import Dejavu

from dejavu.recognize import FileRecognizer, MicrophoneRecognizer

import dejavu\_sqlite

Buat konfigurasi database dengan menggunakan sqlite :

config = {

"database\_type": "sqlite",

"database": {

"db": "dejavu-test.db"

}

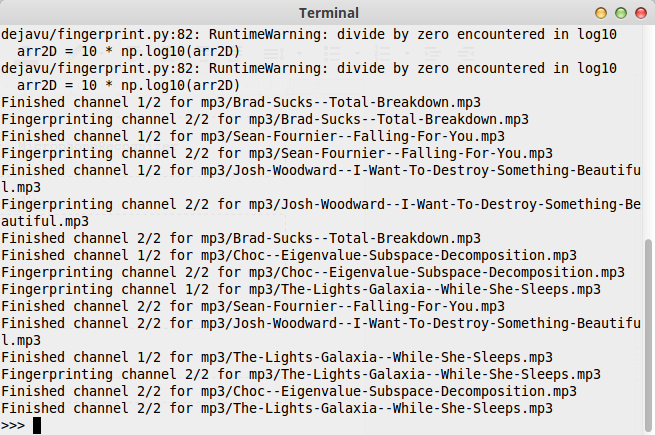
}

Instansiasi kelas Dejavu :

djv = Dejavu(config)

Fingerprint semua file .mp3 dalam sebuah directory mp3 (File audio tersedia di[sini](https://github.com/soedomoto/ITB2015/tree/EL5226/Assignment%202%20-%20DSP/Audio%20Fingerprint%20and%20Recognition%20Python/mp3)) :

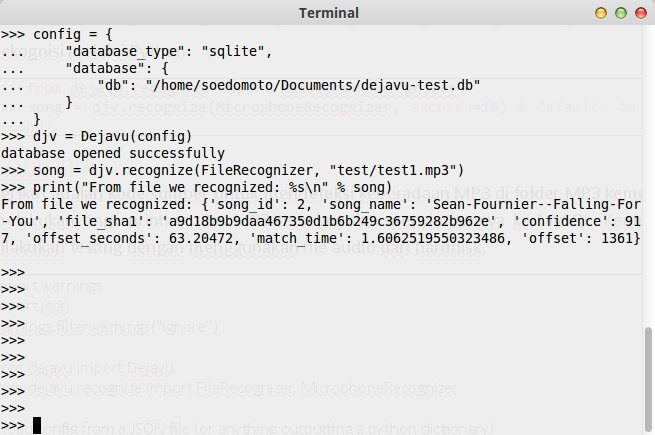
djv.fingerprint\_directory("mp3", [".mp3"])



Deteksi potongan lagu (minimal 5 detik potongan lagu), dan kenali penyanyi-nya (File potongan lagu untuk testing tersedia di[sini](https://github.com/soedomoto/ITB2015/tree/EL5226/Assignment%202%20-%20DSP/Audio%20Fingerprint%20and%20Recognition%20Python/test)) :

song = djv.recognize(FileRecognizer, "test/test1.mp3")

print "From file we recognized: %s\n" % song



Deteksi potongan lagu melalui microphone, dan kenali penyanyi-nya :

secs = 5

song = djv.recognize(MicrophoneRecognizer, seconds=secs)

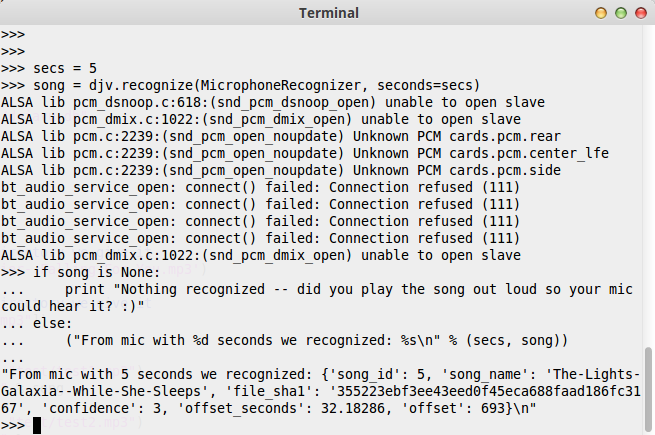
if song is None:

print "Nothing recognized -- did you play the song out loud so your mic could hear it? :)"

else:

print "From mic with %d seconds we recognized: %s\n" % (secs, song)





**2. FACE MOVEMENT RECOGNITION**

Merekam video muka dan mengamati pergerakannya (kanan, kiri)

Face recognition adalah suatu proses mengidentifikasi satu atau lebih wajah dalam sebuah gambar atau video dengan menggunakan sebuah algoritma. Tools yang biasa digunakan untuk mengenali wajah adalah Matlab dan OpenCV. Dalam kasus ini akan digunakan OpenCV dan Python yang dijalankan pada platform berbasis Linux Ubuntu 14.04 Trusty Tahr.

**OpenCV**

[OpenCV](http://opencv.org/) (Open Source Computer Vision) adalah sebuah library yang pertama kali dikembangkan oleh Intel untuk pengolahan citra secara real time. OpenCV bersifat platform independent, dan dapat dijalankan di berbagai platform popular seperti Windows, Linux, FreeBSD, MacOS, dan Android. OpenCV hanyalah sebuah library, dan memerlukan programming interface agar dapat berjalan. Programming interface yang didukung oleh OpenCV adalah C, C++, Java, dan Python.

OpenCV Installation

Secara umum, instalasi OpenCV di ubuntu cukup mudah, karena ubuntu telah menyediakan repository opencv ([detail](https://help.ubuntu.com/community/OpenCV)). Command berikut akan otomatis menginstall opencv berserta seluruh dependency-nya :

sudo apt-get -qq install libopencv-dev build-essential checkinstall cmake pkg-config yasm libjpeg-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev python-dev python-numpy libtbb-dev libqt4-dev libgtk2.0-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils ffmpeg cmake qt5-default checkinstall

Akan tetapi, repository ubuntu tidak menyediakan versi terbaru dari OpenCV. Versi terakhir yang disupport oleh ubuntu adalah OpenCV versi 2.4. Sementara itu untuk menginstall OpenCV versi terbaru, OpenCV 3.1, diperlukan untuk mem-build dari source. Berikut cara instalasi-nya :

Install dependency :

sudo apt-get install build-essential

sudo apt-get install cmake git libgtk2.0-dev pkg-config libavcodec-dev libavformat-dev libswscale-dev

sudo apt-get install python-dev python-numpy libtbb2 libtbb-dev libjpeg-dev libpng-dev libtiff-dev libjasper-dev libdc1394-22-dev

Install OpenCV :

git clone <https://github.com/Itseez/opencv.git>

cd opencv

mkdir build

cd build

cmake -D CMAKE\_BUILD\_TYPE=Release -D CMAKE\_INSTALL\_PREFIX=/usr/local ..

make -j7 # runs 7 jobs in parallel

sudo make install

**Python Application**

Python merupakan salah satu Programming Language yang didukung oleh oleh OpenCV. Untuk melakukan face recognition dengan OpenCV dalam bahasa python, diperlukan dependency yang dapat diperoleh pada OpenCV example :

1. common.py, dapat diunduh di[sini](https://github.com/soedomoto/ITB2015/blob/EL5226/Assignment%202%20-%20DSP/Face%20Movement%20Python%20OpenCV/video.py),
2. video.py, dapat diunduh di[sini](https://github.com/soedomoto/ITB2015/blob/EL5226/Assignment%202%20-%20DSP/Face%20Movement%20Python%20OpenCV/video.py)

Penjelasan Program

Pada kasus ini, akan digunakan webcam dan file .mp4 sebagai media memperoleh video. Video yang ditangkap dari webcam atau dibaca dari file .mp4 kemudian akan dibaca frame by frame, sehingga yang sesungguhnya dideteksi adalah image. Berikut penjelasanya programmnya (Full code dapat diunduh di[sini](https://github.com/soedomoto/ITB2015/blob/EL5226/Assignment%202%20-%20DSP/Face%20Movement%20Python%20OpenCV/face_movement_detection.py)) :

Import library yang diperlukan :

import cv2

# local modules

from video import create\_capture

from common import clock, draw\_str

Buat fungsi-fungsi yang diperlukan :

# Detect cascade (pattern) from image

def detect(img, cascade):

rects = cascade.detectMultiScale(img, scaleFactor=1.3, minNeighbors=4, minSize=(30, 30),

flags=cv2.CASCADE\_SCALE\_IMAGE)

if len(rects) == 0:

return []

rects[:,2:] += rects[:,:2]

return rects

# Draw rectangle from detected object

def draw\_rects(img, rects, color):

for x1, y1, x2, y2 in rects:

cv2.rectangle(img, (x1, y1), (x2, y2), color, 2)

Definisikan video yang ingin dideteksi pada argument, jika tidak ada argument yang diberikan, maka gunakan webcam sebagai sumber :

# Try to get 1st argument as a source for video-file

args, video\_src = getopt.getopt(sys.argv[1:], '', ['cascade=', 'nested-cascade='])

try:

video\_src = video\_src[0]

except:

# If no video-file provide, then use webcam as video source

video\_src = 0

args = dict(args)

Definiskan algoritma yang ingin dipakai (algoritma ditulis pada sebuah file XML) :

# Default : use haarcascade algoritm to detect face and eyes

face\_fn = args.get('--cascade', "data/haarcascades/haarcascade\_frontalface\_alt.xml")

left\_eye\_fn = args.get('--nested-cascade', "data/haarcascades/haarcascade\_lefteye\_2splits.xml")

right\_eye\_fn = args.get('--nested-cascade', "data/haarcascades/haarcascade\_righteye\_2splits.xml")

# Define the classifier for each algorithm

face = cv2.CascadeClassifier(face\_fn)

left\_eye = cv2.CascadeClassifier(left\_eye\_fn)

right\_eye = cv2.CascadeClassifier(right\_eye\_fn)

Capture video dari sumber yang telah didefinisikan (video-file atau webcam) :

# Capture video from source (either webcam or video-file)

cam = create\_capture(video\_src)

Code berikut harus diletakkan dalam iterasi yang tidak pernah berhenti (kecuali diinterupsi) dengan menggunakan while :

while True:

# Next code put after this comment

Baca video sumber per frame yang menghasilkan image, dan konversi menjadi grayscale :

# Capture each frame as image

ret, vis = cam.read()

# Convert image to gray

gray = cv2.cvtColor(vis, cv2.COLOR\_BGR2GRAY)

gray = cv2.equalizeHist(gray)

Deteksi setiap wajah yang terdapat dalam image, kemudian gambarkan boundary wajah sebagai sebuah kotak berwarna hijau. Tuliskan juga label “Face” pada kotak yang terbentuk :

# Detect face from image

rects = detect(gray, face)

# Draw face rectangle in image with green border

draw\_rects(vis, rects, (0, 255, 0))

# Read face rectangle coordinates

for face\_x1, face\_y1, face\_x2, face\_y2 in rects:

# Draw face label

draw\_str(vis, (face\_x1, face\_y1), 'Face')

Crop image sesuai kotak boundary wajah, sehingga hanya akan diperoleh image wajah :

# Get face image

face\_gray = gray[face\_y1:face\_y2, face\_x1:face\_x2]

vis\_face = vis[face\_y1:face\_y2, face\_x1:face\_x2]

Pada image wajah, deteksi mata kiri, kemudian gambarkan boundary mata kiri sebagai sebuah kotak berwarna biru. Tuliskan juga label “Left eye” pada kotak yang terbentuk. Selain itu, hitung titik tengah pada boundary mata kiri (center of left-eye) :

center\_of\_leye = None

# detect left eye coordinates in face image

left\_eye\_rect = detect(face\_gray, left\_eye)

# Draw left eye rectangle in face boundary with blue border

draw\_rects(vis\_face, left\_eye\_rect, (255, 0, 0))

# Read left eye rectangle coordinates

for leye\_x1, leye\_y1, leye\_x2, leye\_y2 in left\_eye\_rect:

# Draw left eye label

draw\_str(vis\_face, (leye\_x1, leye\_y1), 'Left eye')

# Calculate the center of left eye

center\_of\_leye = leye\_x1 + ((leye\_x2 - leye\_x1) / 2)

Masih pada image wajah, deteksi mata kanan, kemudian gambarkan boundary mata kanan sebagai sebuah kota berwarna merah. Tuliskan juga label “Right eye” pada kotak yang terbentuk. Selain itu, hitung juga titik tengah pada boundary mata kanan (center of right-eye) :

center\_of\_reye = None

# detect right eye coordinates in face image

right\_eye\_rect = detect(face\_gray, right\_eye)

# Draw right eye rectangle in face boundary with red border

draw\_rects(vis\_face, right\_eye\_rect, (0, 0, 255))

# Read right eye rectangle coordinates

for reye\_x1, reye\_y1, reye\_x2, reye\_y2 in right\_eye\_rect:

# Draw right eye label

draw\_str(vis\_face, (reye\_x1, reye\_y1), 'Right eye')

# Calculate the center of right eye

center\_of\_reye = reye\_x1 + ((reye\_x2 - reye\_x1) / 2)

Berdasarkan jarak boundary kiri terhadap center-of-left-eye (left-eye-border), bandingkan dengan jarak boundary kanan terhadap center-of-right-eye (right-eye-border). Jika left-eye-border lebih kecil dari right-eye-border maka dapat disimpulkan wajah sedah menghadap ke kiri, dan sebaliknya. Tuliskan label “Look left” dan “Look right” berdasarkan kesimpulan yang didapat :

# Label of status will be placed in the bottom of face rectangle

label\_pos = (face\_x1, face\_y2)

# If both left eye and right eye are not detected

if center\_of\_leye is None and center\_of\_reye is None:

draw\_str(vis, label\_pos, 'Eyes not detected')

# If only left eye is not detected -> Look left

if center\_of\_leye is None:

draw\_str(vis, label\_pos, 'Look left')

# If only right eye is not detected -> Look right

elif center\_of\_reye is None:

draw\_str(vis, label\_pos, 'Look right')

# If both are detected

else:

if abs((center\_of\_leye - face\_x1) - (face\_x2 - center\_of\_reye)) < 100:

draw\_str(vis, label\_pos, 'Look forward')

# if right eye is more center than left

elif (center\_of\_leye - face\_x1) > (face\_x2 - center\_of\_reye):

draw\_str(vis, label\_pos, 'Look left')

# If left eye is more center than right

else:

draw\_str(vis, label\_pos, 'Look right')

Terakhir, tampilkan setiap image (frame) :

# Show image

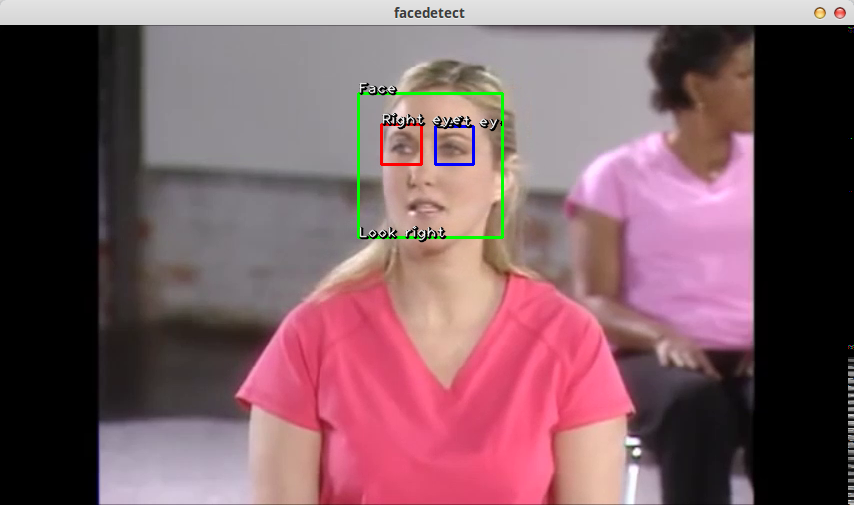
cv2.imshow('facedetect', vis)

**Uji Coba**

Menggunakan file video .mp4. File untuk ujicoba dapat diunduh di[sini](https://github.com/soedomoto/ITB2015/blob/EL5226/Assignment%202%20-%20DSP/Face%20Movement%20Python%20OpenCV/data/face_video.mp4) :

python face\_movement\_detection.py data/face\_video.mp4





Menggunakan webcam :

python face\_movement\_detection.py

