8-Beyond-Text-sunny

June 3, 2020

1 Week 8 - Beyond Text

This week, we "trascend" text to explore analysis of sound and visual content. Trillions of digital audio, image, and video files have been generated by cell phones and distributed sensors, preserved and shared through social medial, the web, private and government administrations. In this notebook, we read in and visualize audio and image files, process them to extract relevant features and measurement, then begin to explore how to analyze and extract information from them through the same approaches to supervised and unsupervised learning we have performed thoughout the quarter with text.

For this notebook we will use the following packages:

[1]:
```
#Special module written for this class
#This provides access to data and to helper functions from previous weeks
#Make sure you update it before starting this notebook
import lucem_illud_2020 #pip install -U git+git://github.com/
 ↪Computational-Content-Analysis-2020/lucem_illud_2020.git

#All these packages need to be installed from pip
import scipy #For frequency analysis
import scipy.fftpack
import nltk #the Natural Language Toolkit
import requests #For downloading our datasets
import numpy as np #for arrays
import pandas #gives us DataFrames
import matplotlib.pyplot as plt #For graphics
import seaborn #Makes the graphics look nicer
import IPython #To show stuff

#Image handling install as Pillow
import PIL
import PIL.ImageOps

#install as scikit-image, this does the image manupulation
import skimage
import skimage.feature
import skimage.segmentation
import skimage.filters
```

```
import skimage.color
import skimage.graph
import skimage.future.graph

#these three do audio handling
import pydub #Requires ffmpeg to be installed https://www.ffmpeg.org/download.
 ↪html; on a mac "brew install ffmpeg"
import speech_recognition #install as speechrecognition
import soundfile #Install as pysoundfile

#This 'magic' command makes the plots work better
#in the notebook, don't use it outside of a notebook.
#Also you can ignore the warning it may generate.
%matplotlib inline

import os
import os.path
import csv
import re
```

2 Audio analysis

First we will consider media that predates written language…sound and spoken language. Audio (and video) files come in two major categories, lossy or lossless. Lossless files save all information the microphone recorded. Lossy files, by contrast, drop sections humans are unlikely to notice. Recorded frequencies for both types are then typically compressed, which introduces further loss. To work with audio files, we want a format that is preferably lossless or minimally compressed. We will work with `wav` files here. Note that `mp3` is not acceptable. If you do not have `wav` files, we can use python to convert to `wav`.

You might need to install `ffmpeg` and `ffprobe`.

[6]:
```
root = '/Users/sunny/Documents/CCA Spring 2020/Content-Analysis-2020-master/
 ↪data/audio_samples/'
```

[7]:
```
samplePath = '/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/SBC060.mp3'
transcriptPath = '/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/SBC060.trn'

IPython.display.Audio(samplePath)
```

[7]:
```
<IPython.lib.display.Audio object>
```

[8]:
```
# We are using a different package to convert than the in the rest of the code
def convertToWAV(sourceFile, outputFile, overwrite = False):
    if os.path.isfile(outputFile) and not overwrite:
```

```
        print("{} exists already".format(outputFile))
        return
    #Naive format extraction
    sourceFormat = sourceFile.split('.')[-1]
    sound = pydub.AudioSegment.from_file(sourceFile, format=sourceFormat)
    sound.export(outputFile, format="wav")
    print("{} created".format(outputFile))
wavPath = 'sample.wav'
convertToWAV(samplePath, wavPath)
```

```
sample.wav created
```

Now that we have created our `wav` file, notice that it is much larger than the source `mp3`. We can load it with `soundfile` and work with it as a numpy data array.

[9]:
```
soundArr, soundSampleRate = soundfile.read(wavPath)
soundArr.shape
```

[9]:
```
(65705472, 2)
```

This is the raw data as a column array, which contains two channels (Left and Right) of the recording device. Some files, of course, will have more columns (from more microphones). The array comprises a series of numbers that measure the location of the speaker membrane (0=resting location). By quickly and rhythmically changing the location a note can be achieved. The larger the variation from the center, the louder the sound; the faster the oscillations, the higher the pitch. (The center of the oscillations does not have to be 0).

[10]:
```
soundSampleRate
```

[10]:
```
44100
```

The other piece of information we get is the sample rate. This tells us how many measurements made per second, which allows us to know how long the entire recording is:

[11]:
```
numS = soundArr.shape[0] // soundSampleRate
print("The sample is {} seconds long".format(numS))
print("Or {:.2f} minutes".format(numS / 60))
```

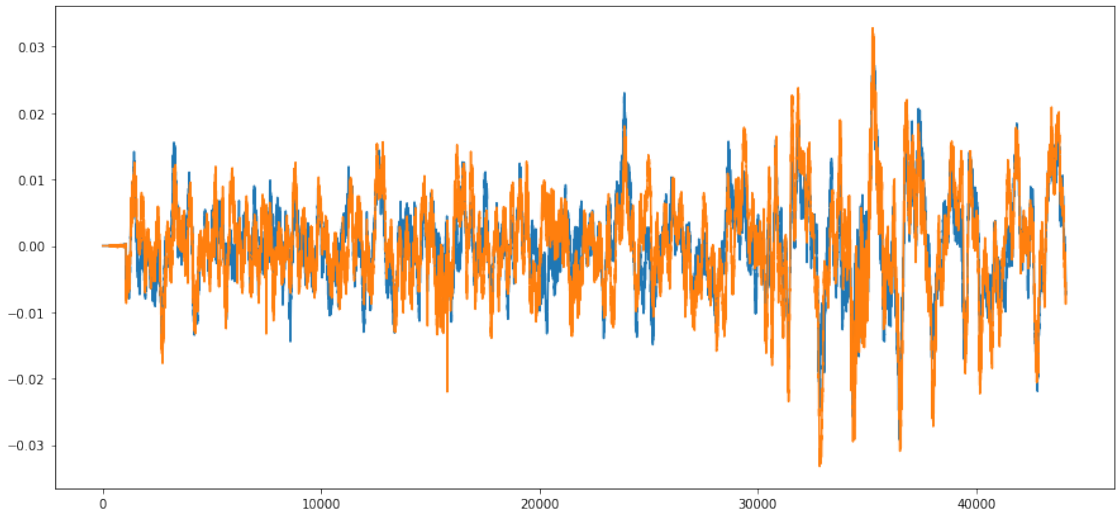
```
The sample is 1489 seconds long
Or 24.82 minutes
```

The final critical parameter of sound digitization is quantisation, which consists in assigning a value to each sample according to its amplitude. These values are attributed according to a bit scale. A quantisation of 8 bit will assign amplitude values along a scale of $2^8 = 256$ states around 0. Most recording systems use a $2^{16} = 65536$ bit system. Quantisation is a rounding process, where high bit quantisation produces values close to reality with values rounded to a high number of significant digits, and low bit quantisation produces values further from reality with values rounded a low number of significants digits. Low quantisation can lead to impaired quality signal. This figure

illustrates how digital sounds is a discrete process along the amplitude scale: a 3 bit, $2^3 = 8$, quantization (gray bars) gives a rough approximation of the sin wave (red line).

Let's look at the first second of the recording:

[12]:
```
fig, ax = plt.subplots(figsize = (15, 7))
ax.plot(soundArr[:soundSampleRate])
plt.show()
```



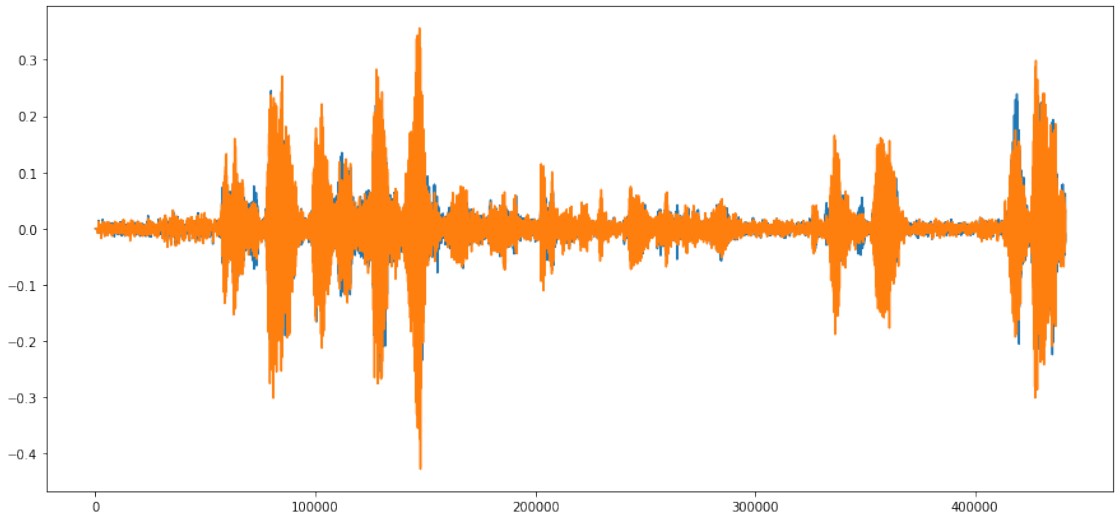
We get 2 (Left and Right) nearly "flat" (or equally wavy) lines. This means that there is very little noise at this part of the recording. What variation exists is due to compression or interference and represents the slight hiss you sometimes hear in low quality recordings.

Let's expand our scope and look at the first 10 seconds:

[13]:
```
soundArr.shape
```

[13]:
```
(65705472, 2)
```

[14]:
```
fig, ax = plt.subplots(figsize = (15, 7))
ax.plot(soundArr[:soundSampleRate * 10])
plt.show()
```



Now we can see definite spikes, where each represents a word or discrete sound.

To see what the different parts correspond to, we can use a transcript. Because we got this file from the Santa Barbara Corpus of Spoken American English, we just need to load the metadata, which includes a transcription.

[16]:
```
def loadTranscript(targetFile):
    #Regex because the transcripts aren't consistent enough to use csv
    regex = re.compile(r"(\d+\.\d+)\s(\d+\.\d+)\s(.+:)?\s+(.*)")
    dfDict = {
        'time_start' : [],
        'time_end' : [],
        'speaker' : [],
        'text' : [],
    }
    with open(targetFile, encoding='latin-1') as f:
        for line in f:
            r = re.match(regex, line)
            dfDict['time_start'].append(float(r.group(1)))
            dfDict['time_end'].append(float(r.group(2)))
            if r.group(3) is None:
                dfDict['speaker'].append(dfDict['speaker'][-1])
            else:
                dfDict['speaker'].append(r.group(3))
            dfDict['text'].append(r.group(4))
    return pandas.DataFrame(dfDict)

transcriptDF = loadTranscript(transcriptPath)
transcriptDF[:10]
```

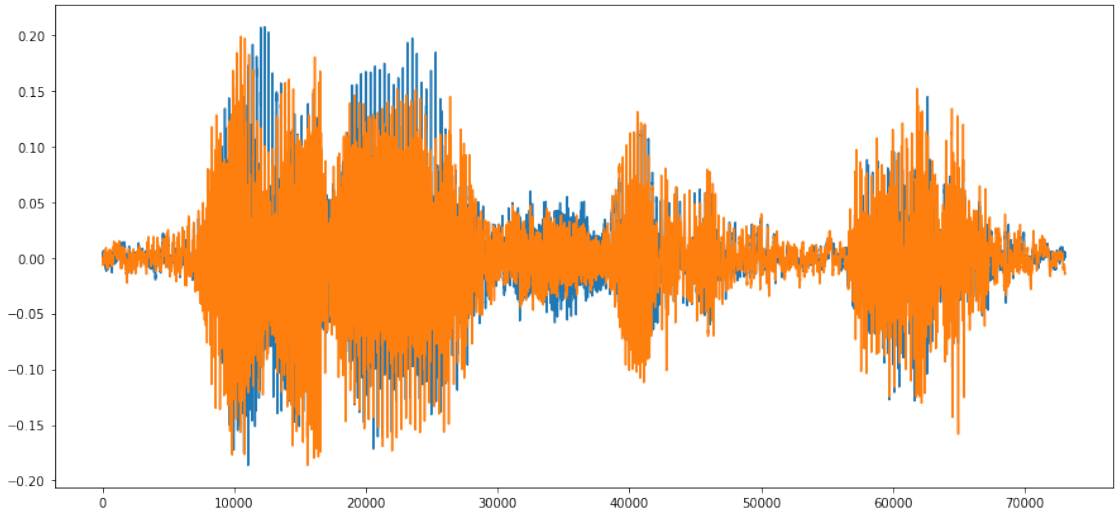
[16]:
```
   time_start  time_end speaker  \
0       0.000     2.572   ALAN:
1       2.572     3.820   ALAN:
2       3.820     6.645   ALAN:
3       6.645     8.378   ALAN:
4       8.378    10.178   ALAN:
5      10.178    10.678   ALAN:
6      10.678    12.335   ALAN:
7      12.335    14.727   ALAN:
8      14.727    17.012   ALAN:
9      17.012    18.761    JON:

                                                 text
0                  … (H) I got a story to tell you,
1                            it's a shaggy dog story,
2  but as long as we re talking about that record…
3                       (H)= I ll tell you the story,
4                                   (H)= oh gosh,
5                                        uh (Hx),
6                        Rae and I and Sue and Buddy,
7                                    … took a trip,
8                                 … to Mexico City,
9                                        … (SNIFF)
```

Now let's look at a few sub-sections. First, to make things easier, we will convert the seconds markers to sample indices:

[17]:
```
#Need to be ints for indexing, luckily being off by a couple indices doesn't␣
↪matter
transcriptDF['index_start'] = (transcriptDF['time_start'] * soundSampleRate).
↪astype('int')
transcriptDF['index_end'] = (transcriptDF['time_end'] * soundSampleRate).
↪astype('int')
```

Lets see what `'Rae and I and Sue and Buddy,'` looks like, which is the seventh row:

[18]:
```
fig, ax = plt.subplots(figsize = (15, 7))
subSample1 = soundArr[transcriptDF['index_start'][6]:␣
↪transcriptDF['index_end'][6]]
ax.plot(subSample1)
plt.show()
```



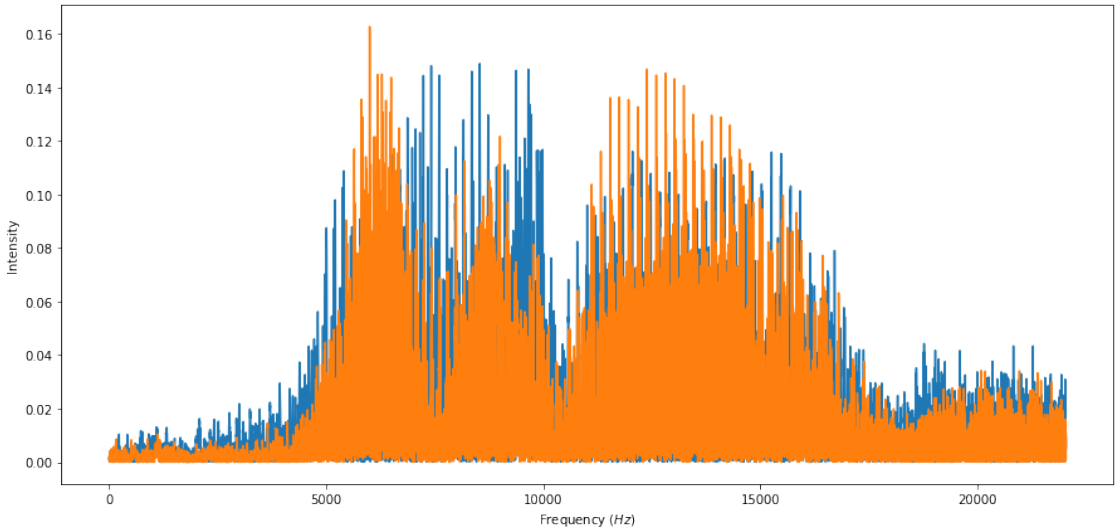
And now let's hear what that sounds like:

[20]:
```
soundfile.write('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/sample1.wav', subSample1,␣
 ↪soundSampleRate)
IPython.display.Audio('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/sample1.wav')
```

[20]:
```
<IPython.lib.display.Audio object>
```

In order to see sounds in the frequency space, we can take the Fourier transform. This is a reversible mathematical transform named after the French mathematician Joseph Fourier (1768-1830) . The transform decomposes a time series into a sum of finite series of sine or cosine functions.

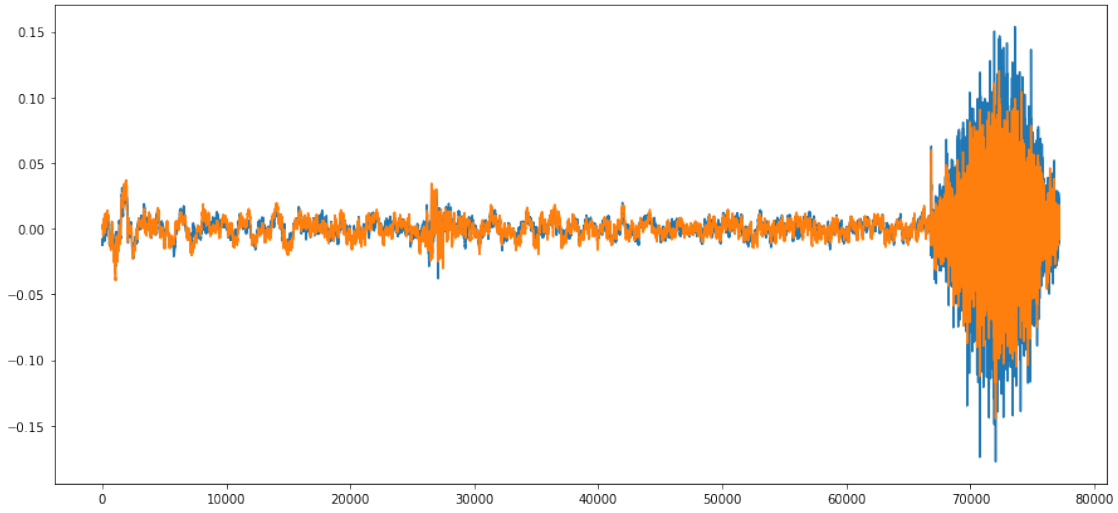
[21]:
```
sample1FFT = scipy.fftpack.ifft(subSample1)
N = len(sample1FFT)
freq = scipy.fftpack.fftfreq(N, d = 1 / soundSampleRate)
fig, ax = plt.subplots(figsize = (15, 7))
ax.set_xlabel('Frequency ($Hz$)')
ax.set_ylabel('Intensity')
ax.plot(freq[:N//2], abs(sample1FFT)[:N//2]) #Only want positive frequencies
plt.show()
```



This shows that there are two frequencies to the 'Rae and I and Sue and Buddy' snippet: a higher pitched 'Rae and I…Sue…Buddy' (~14000 *Hz*) and the final two 'and's (one at ~6000 *Hz* and the second at ~8000 *Hz*).

What does a sniff look like?

[22]:
```
fig, ax = plt.subplots(figsize = (15, 7))
subSample2 = soundArr[transcriptDF['index_start'][9]:␣
 ↪transcriptDF['index_end'][9]]
ax.plot(subSample2)
plt.show()
```



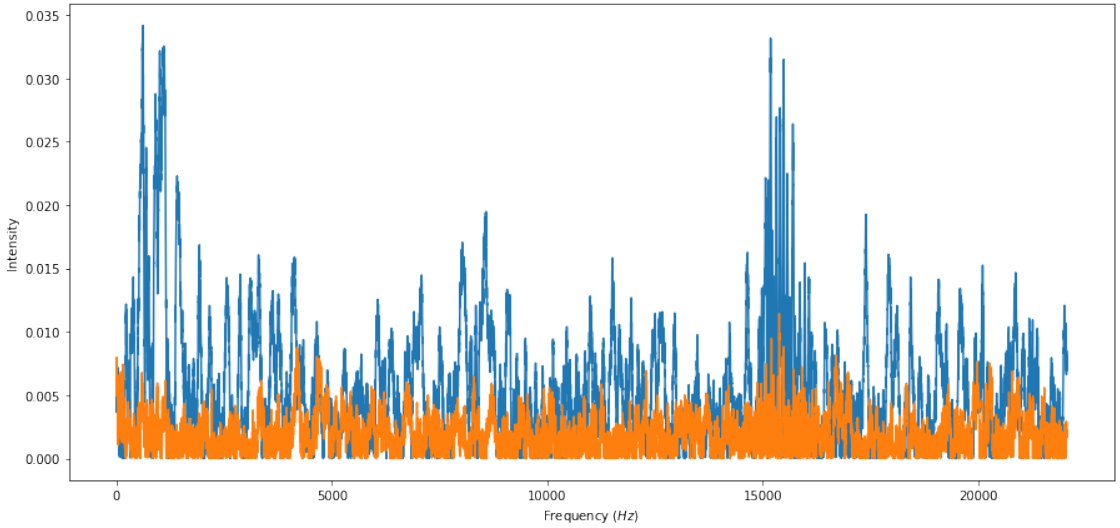
Very different from speech. And now let's see what that sounds like:

[24]:
```
soundfile.write('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/sample2.wav', subSample2,␣
 ↪soundSampleRate)
IPython.display.Audio('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/sample2.wav')
```

[24]:
```
<IPython.lib.display.Audio object>
```

and in frequency space:

[25]:
```
sample2FFT = scipy.fftpack.ifft(subSample2)
N = len(sample2FFT)
freq = scipy.fftpack.fftfreq(N, d = 1 / soundSampleRate)
fig, ax = plt.subplots(figsize = (15, 7))
ax.plot(freq[:N//2], abs(sample2FFT)[:N//2]) #Only want positive frequencies
ax.set_xlabel('Frequency ($Hz$)')
ax.set_ylabel('Intensity')
plt.show()
```

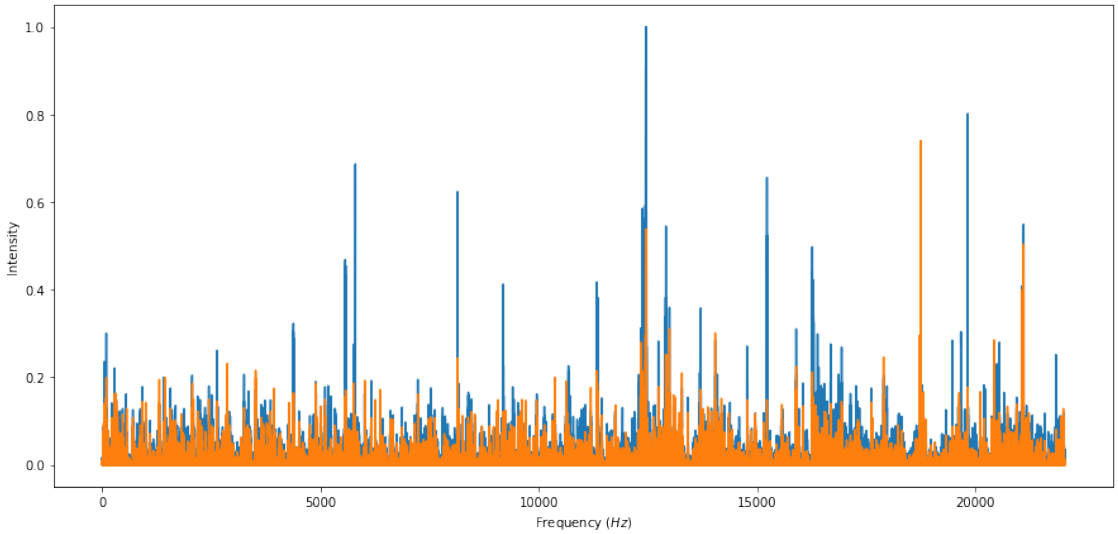


Notice how there is not a dominant frequency for the sniff as there was for the noun phrase earlier. This means that the sniff activated noise all across the frequency spectrum.

We can also investigate dominant frequencies for the entire record:

[26]:
```
#This takes a while
fig, ax = plt.subplots(figsize = (15, 7))
fullFFT = scipy.fftpack.ifft(soundArr)
N = len(fullFFT)
freq = scipy.fftpack.fftfreq(N, d = 1 / soundSampleRate)
ax.plot(freq[:N//2], abs(fullFFT)[:N//2]) #Only want positive frequencies
```

```
ax.set_xlabel('Frequency ($Hz$)')
ax.set_ylabel('Intensity')
plt.show()
```



[27]:
```
freq[len(freq) // 2 -10: len(freq) // 2 + 5]
```

[27]:
```
array([ 22049.99328823,  22049.99395941,  22049.99463058,  22049.99530176,
        22049.99597294,  22049.99664412,  22049.99731529,  22049.99798647,
        22049.99865765,  22049.99932882, -22050.        , -22049.99932882,
       -22049.99865765, -22049.99798647, -22049.99731529])
```

Here we capture each person's frequencies across their entire collection of statements:

[28]:
```
def maxfreq(sample, topN = 10):
    sampleFFT = scipy.fftpack.ifft(sample)
    N = len(sample)
    freqs = scipy.fftpack.fftfreq(N, d = 1 / soundSampleRate)
    tops =  np.argpartition(abs(sampleFFT[:, 0]), -topN)[-topN:]

    return np.mean(tops)

freqs = []
for i, row in transcriptDF.iterrows():
    freqs.append(maxfreq(soundArr[row['index_start']: row['index_end']]))

transcriptDF['frequency FFT'] = freqs
```

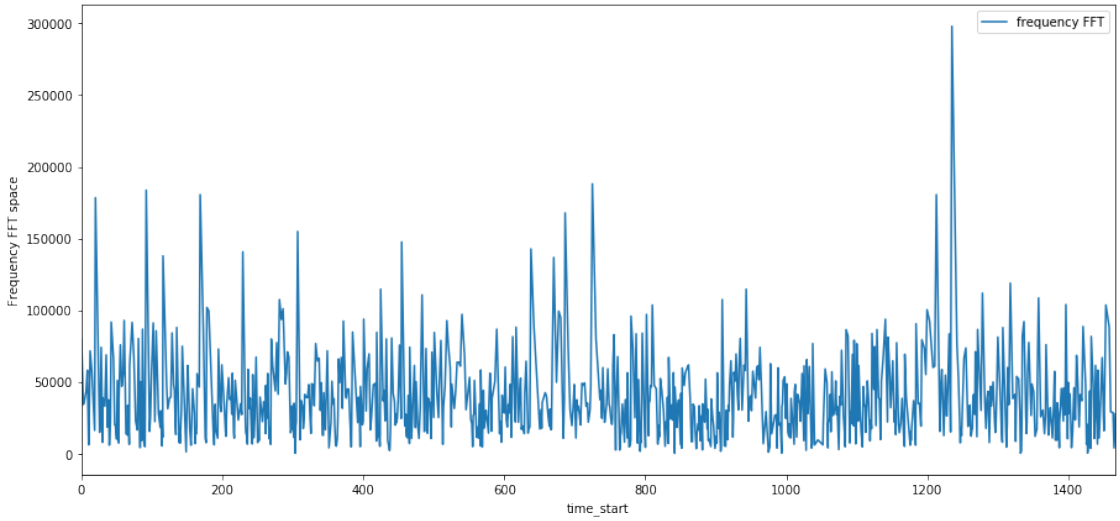
[56]:
```
transcriptDF
```

[56]:
```
      time_start  time_end speaker  \
0          0.000     2.572   ALAN:
1          2.572     3.820   ALAN:
2          3.820     6.645   ALAN:
3          6.645     8.378   ALAN:
4          8.378    10.178   ALAN:
…            …       …     …
1008    1483.042  1484.999    JON:
1009    1484.999  1486.347    JON:
1010    1486.347  1488.519    JON:
1011    1488.519  1489.281    JON:
1012    1489.281  1489.862    JON:

                                                  text  index_start  \
0                    … (H) I got a story to tell you,            0
1                              it's a shaggy dog story,       113425
2     but as long as we re talking about that record…       168462
3                          (H)= I ll tell you the story,       293044
4                                      (H)= oh gosh,       369469
…                                                  …          …
1008                          … oh further than that,     65402152
1009                            maybe nineteen-seventy.     65488455
1010                                 … (H) .. in % --     65547902
1011                                        in Houston,     65643687
1012                                                …     65677292

      index_end  frequency FFT
0        113425        79742.7
1        168462        34108.1
2        293044        35699.1
3        369469        43339.9
4        448849        58084.1
…          …             …
1008   65488455        67882.5
1009   65547902         4696.0
1010   65643687        69707.7
1011   65677292        15633.9
1012   65702914        21196.0

[1013 rows x 7 columns]
```

Alan's speech exhibits the following frequencies:

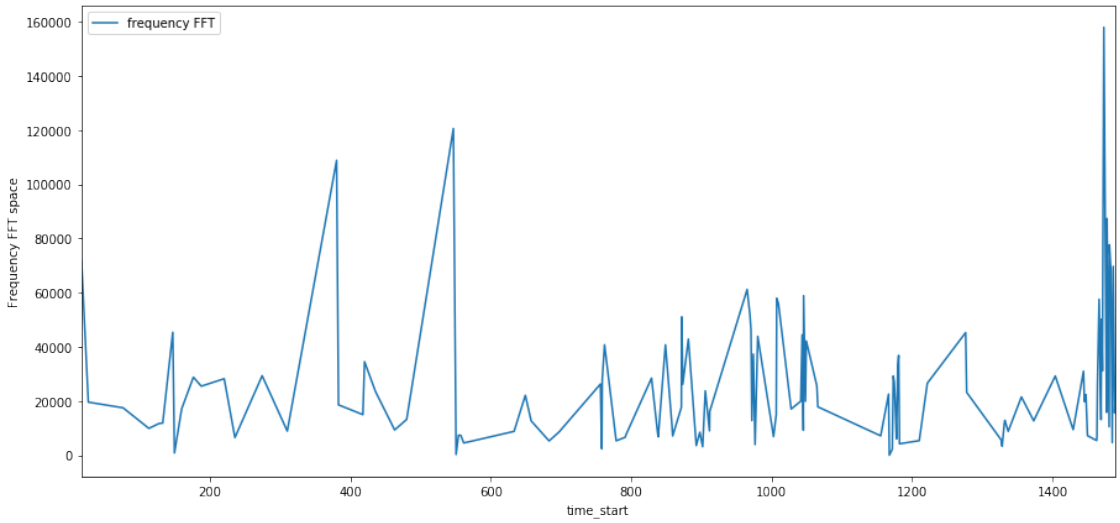
[29]:
```
fig, ax = plt.subplots(figsize = (15, 7))
transcriptDF[transcriptDF['speaker'] == 'ALAN:'].plot( 'time_start', 'frequency␣
 ↪FFT', ax = ax)
ax.set_ylabel("Frequency FFT space")
```

```
plt.show()
```



...while Jon's voice is **much** lower:

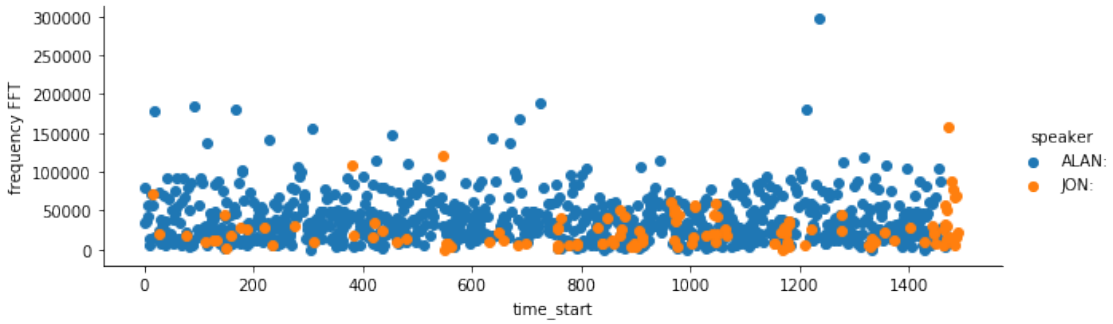
[30]:
```
fig, ax = plt.subplots(figsize = (15, 7))
transcriptDF[transcriptDF['speaker'] == 'JON:'].plot( 'time_start', 'frequency␣
↪FFT', ax = ax)
ax.set_ylabel("Frequency FFT space")
plt.show()
```



Or we can look at them togther

[31]:
```
fg = seaborn.FacetGrid(data=transcriptDF, hue='speaker', aspect = 3)
fg.map(plt.scatter, 'time_start', 'frequency FFT').add_legend()
```

[31]:
```
<seaborn.axisgrid.FacetGrid at 0x1472cf4d0>
```



2.1 *Exercise 1*

Construct cells immediately below this that read in 10 audio files (e.g., produced on your smartphone recorder?) from at least two different speakers, which include sentences of different types (e.g., question, statement, exclamation). At least two of these should include recordings of the two speakers talking to each other (e.g., a simple question/answer). Contrast the frequency distributions of the words spoken within speaker. What speaker's voice has a higher and which has lower frequency? What words are spoken at the highest and lowest frequencies? What parts-of-speech tend to be high or low? How do different types of sentences vary in their frequency differently? When people are speaking to each other, how do their frequencies change? Whose changes more?

[15]:
```
root_hw = '/Users/sunny/Documents/CCA Spring 2020/Content-Analysis-2020-master/
 ↪data/audio_samples_hw'
```

SBC010 Letter of Concerns A business conversation recorded in New Mexico. Brad and Phil are board members of a local arts society. Phil wants to talk business, while Brad keeps trying to leave to pick up his wife who's waiting for him at a bookstore.

[16]:
```
samplePath_hw = '/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples_hw/SBC010.mp3'
transcriptPath_hw = '/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples_hw/SBC010.trn'

IPython.display.Audio(samplePath_hw)
```

[16]:
```
<IPython.lib.display.Audio object>
```

[17]:
```
# We are using a different package to convert than the in the rest of the code
def convertToWAV(sourceFile, outputFile, overwrite = False):
    if os.path.isfile(outputFile) and not overwrite:
```

```
        print("{} exists already".format(outputFile))
        return
    #Naive format extraction
    sourceFormat = sourceFile.split('.')[-1]
    sound = pydub.AudioSegment.from_file(sourceFile, format=sourceFormat)
    sound.export(outputFile, format="wav")
    print("{} created".format(outputFile))

wavPath_hw = 'sample_hw.wav'
convertToWAV(samplePath_hw, wavPath_hw)
```

```
sample_hw.wav exists already
```

[18]:
```
soundArr, soundSampleRate = soundfile.read(wavPath_hw)
soundArr.shape
```

[18]:
```
(41489280, 2)
```

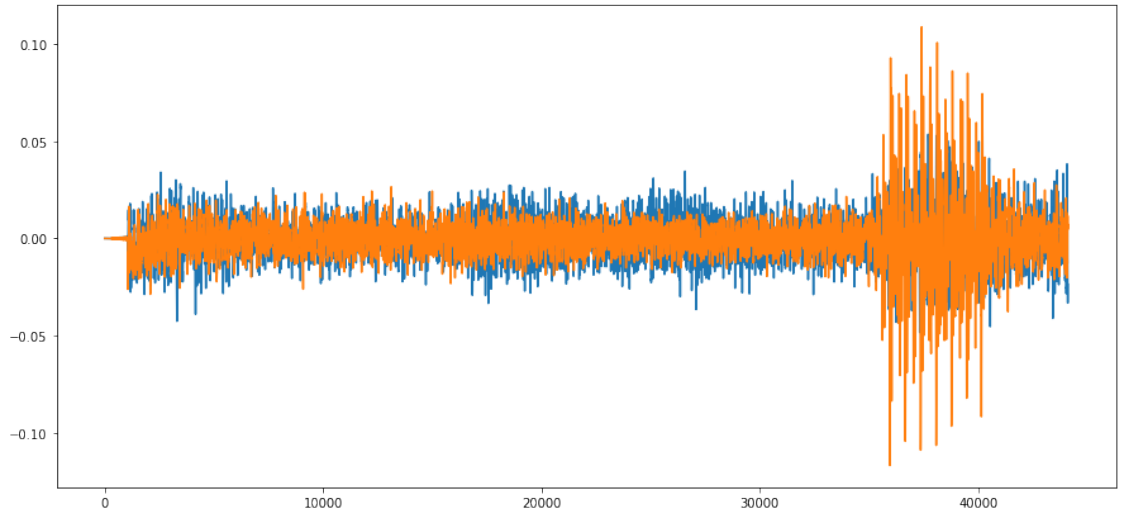
[19]:
```
soundSampleRate
```

[19]:
```
44100
```

[20]:
```
numS = soundArr.shape[0] // soundSampleRate
print("The sample is {} seconds long".format(numS))
print("Or {:.2f} minutes".format(numS / 60))
```

```
The sample is 940 seconds long
Or 15.67 minutes
```

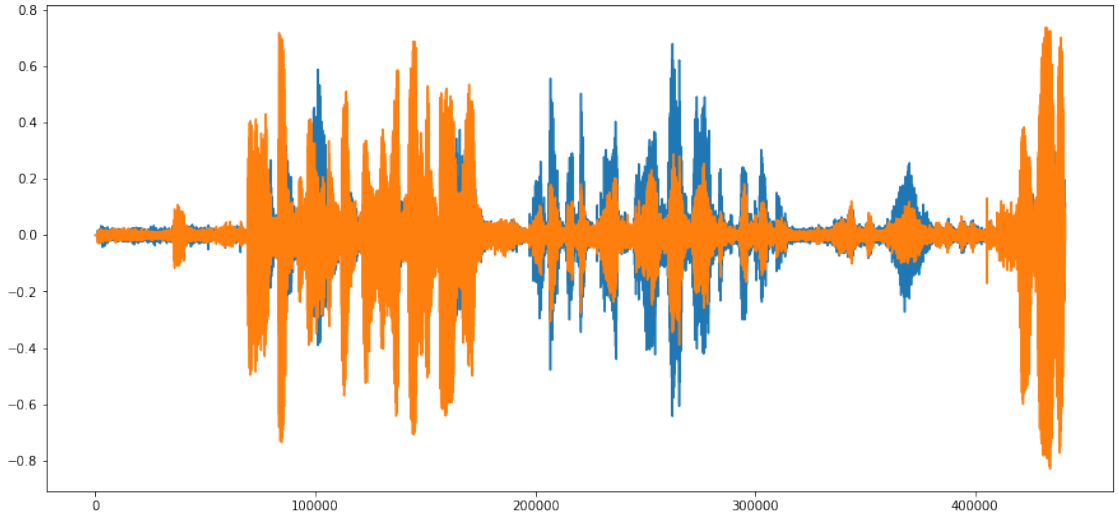
[21]:
```
fig, ax = plt.subplots(figsize = (15, 7))
ax.plot(soundArr[:soundSampleRate])
plt.show()
```



[22]:
```
soundArr.shape
```

[22]:
```
(41489280, 2)
```

[23]:
```
fig, ax = plt.subplots(figsize = (15, 7))
ax.plot(soundArr[:soundSampleRate * 10])
plt.show()
```



[25]:
```
def loadTranscript(targetFile):
    #Regex because the transcripts aren't consistent enough to use csv
    regex = re.compile(r"(\d+\.\d+)\s(\d+\.\d+)\s(.+:)?\s+(.*)")
    dfDict = {
        'time_start' : [],
        'time_end' : [],
        'speaker' : [],
        'text' : [],
    }
    with open(targetFile, encoding='latin-1') as f:
        for line in f:
            r = re.match(regex, line)
            dfDict['time_start'].append(float(r.group(1)))
            dfDict['time_end'].append(float(r.group(2)))
            if r.group(3) is None:
                dfDict['speaker'].append(dfDict['speaker'][-1])
            else:
                dfDict['speaker'].append(r.group(3))
            dfDict['text'].append(r.group(4))
```

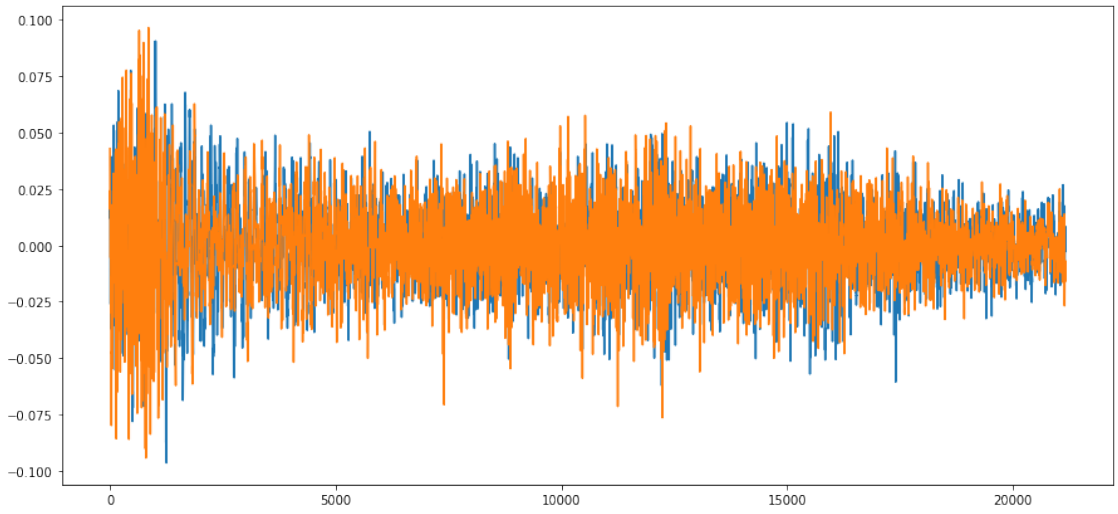
```
    return pandas.DataFrame(dfDict)

transcriptDF = loadTranscript(transcriptPath_hw)
transcriptDF[:10]
```

[25]:

	time_start	time_end	speaker	text
0	0.00	0.80	PHIL:	… But,
1	0.80	1.85	PHIL:	.. (H) <X But X> anyways,
2	1.85	2.89	PHIL:	back to the [.. the] first thing,
3	2.17	2.37	BRAD:	[Okay].
4	2.89	3.95	PHIL:	what we were talking a[2bout was2],
5	3.65	4.05	BRAD:	[2Yeah2],
6	3.95	4.43	PHIL:	(H)=
7	4.43	5.40	BRAD:	I've gotta pick up [!Pat,
8	5.06	5.45	PHIL:	[(Hx)=]
9	5.40	5.65	BRAD:	.. I 1-] --

[73]:
```
#Need to be ints for indexing, luckily being off by a couple indices doesn't␣
 ↪matter
transcriptDF['index_start'] = (transcriptDF['time_start'] * soundSampleRate).
 ↪astype('int')
transcriptDF['index_end'] = (transcriptDF['time_end'] * soundSampleRate).
 ↪astype('int')
```

[74]:
```
fig, ax = plt.subplots(figsize = (15, 7))
subSample1 = soundArr[transcriptDF['index_start'][6]:␣
 ↪transcriptDF['index_end'][6]]
ax.plot(subSample1)
plt.show()
```



[ ]:
```
samplePath_hw = '/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/trump_obama.wav'
#transcriptPath_hw = '/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples_hw/SBC010.trn'

IPython.display.Audio(samplePath_hw)
```

2.2 Speech-to-Text

We can also do speech recognition on audio, but this requires a complex machine learning system. Luckily there are many online services to do this. We have a function that uses Google's API. There are two API's: one is free but limited; the other is commercial and you can provide the function `speechRec` with a file containing the API keys, using `jsonFile=` if you wish. For more about this look here or the `speech_recognition` docs.

[2]:
```
#Using another library so we need to use files again
def speechRec(targetFile, language = "en-US", raw = False, jsonFile = ''):
    r = speech_recognition.Recognizer()
    if not os.path.isfile(jsonFile):
        jsonString = None
    else:
        with open(jsonFile) as f:
            jsonString = f.read()
    with speech_recognition.AudioFile(targetFile) as source:
        audio = r.record(source)
    try:
        if jsonString is None:
            print("Sending data to Google Speech Recognition")
            dat =  r.recognize_google(audio)
        else:
            print("Sending data to Google Cloud Speech")
            dat =  r.recognize_google_cloud(audio, credentials_json=jsonString)
    except speech_recognition.UnknownValueError:
        print("Google could not understand audio")
    except speech_recognition.RequestError as e:
        print("Could not request results from Google service; {0}".format(e))
    else:
        print("Success")
        return dat
```

The example above is of too low quality so we will be using another file `data/audio_samples/english.wav`

[3]:
```
import wave

# open up a wave
```

```
wf = wave.open('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/english.wav', 'rb')
swidth = wf.getsampwidth()
RATE = wf.getframerate()
```

[4]:
```
IPython.display.Audio('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples/english.wav', rate=RATE)
```

[4]:
```
<IPython.lib.display.Audio object>
```

[5]:
```
speechRec('/Users/sunny/Documents/CCA Spring 2020/Content-Analysis-2020-master/
 ↪data/audio_samples/english.wav')
```

```
Sending data to Google Speech Recognition
Success
```

[5]:
```
"what if somebody decides to break it be careful that you keep adequate coverage
but look for places to save money baby it's taking longer to get things squared
away than the bankers expected during the Y for once company may win her tax
hated retirement income based on the two naked bone when the title of this type
of song is in question or waxing or gassing needed work on a flat surface and
smooth out system uses a single self-contained unit the old shop at it still
holds a good mechanic is usually a bad boss or higher in later years"
```

2.3 *Exercise 2*

Construct cells immediately below this that use the 10 audio files from at least two different speakers read in previously, attempt to automatically extract the words from Google, and calculate the word-error rate, as descibed in Chapter 9 from *Jurafsky & Martin*, page 334. How well does it do? Under what circumstances does it perform poorly?

[6]:
```
# open up a wave
wf_hw = wave.open('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/audio_samples_hw/SBC012.wav', 'rb')
swidth_hw = wf_hw.getsampwidth()
RATE = wf_hw.getframerate()
```

[61]:
```
import os
import fnmatch
import pandas as pd
from jiwer import wer
```

[53]:
```
recog_list
```

[53]:
```
["I'm a big ham radio just got on that stage and I'm do it I don't really think
myself that seriously not anymore there was a time when the other days where
I'll music was raped pump isn't very serious when people start",
```

```
 None,
 'and what is it like amazing crown',
 'space right now',
 None,
 'like figuring out what my',
 "a three-year-old boys went to fly fish job the guys from Shoreline only hard
one but said I think I'll do them last y'all question whoever gave me the best
thing to her on her yes for Stovall sit in your opinion what's the fastest thing
in the world no more set of thought he should have thought it ain't",
 "I wanted to ask you how much wtt has meant to you over the years and how much
you enjoyed participating in this kind of even I've been part of this event ever
since I was a little girl ever since I won major titles or big tournaments it
was really the event that gave me the experience that I needed at that age I
would travel from City to City competing every single night and I remember
leaving Wilson tennis I went on to win a a smaller WTA event and it was really
the beginning of my career and so yeah I'm very grateful for the opportunities
to still be able to come out on the court I'm sorry I'm not playing this evening
but for a good reason but yeah this is this tournament I played on world team
tennis on this court so great memories to be back here I think being played here
against each other I'm telling you a couple minutes ago I still very much loved
the game but every athlete whether you're a female or male as an athlete you
have to say I'm finished at one point and for me that came this week figure it
out for the next match some of the best part of this game",
 "oh hey you must be here for the job nope. Tell me a little bit about yourself
of course that's why I'm here astrology is real zodiac sign is a Leo",
 "hi there I'm Trevor Noah and I want to interview you right here on the set of
The Daily Show seriously it's all a part of an amazing initiative to support a
program called education changemakers and it's going to be a lot of fun I'll put
you up in a fancy hotel in New York City and I'll even fly you out here I mean
not literally I don't have a pilot's license like a real pilot with a real
mustache will fly you out here and then you and me will hang out and you can
even bring a friend with you and if you don't to do that you can bring me with
you",
 "and now there's a new form of cyber matchmaking College networking website is
this perhaps the next big thing in Hartford and Stillman co-creator of lively
and universities West Match.com and then in Watertown Massachusetts Mark
Zuckerberg free creator of Harvard the facebook.com and I guess we'll start with
you Dan at what compelled you to do this"]
```

[54]:
```
aud_path = "/Users/sunny/Documents/CCA Spring 2020/Content-Analysis-2020-master/
 ↪data/transcript/wav_files/"
recog_list = []
for file_name in os.listdir(aud_path):
    print(file_name)
    if fnmatch.fnmatch(file_name, '*.wav'):
        rec = speechRec(aud_path + file_name)
        print(rec)
```

```
        recog_list.append(rec)
```

```
Freddie_Mercury_Interview.wav
Sending data to Google Speech Recognition
Success
I'm a big ham radio just got on that stage and I'm do it I don't really think
myself that seriously not anymore there was a time when the other days where
I'll music was raped pump isn't very serious when people start
What BERT is Not.wav
Sending data to Google Speech Recognition
Could not request results from Google service; recognition connection failed:
[Errno 32] Broken pipe
None
Freya_Zywoo_interview.wav
Sending data to Google Speech Recognition
Success
and what is it like amazing crown
Ted.wav
Sending data to Google Speech Recognition
Success
space right now
The_Devil_Wears_Prada.wav
Sending data to Google Speech Recognition
Google could not understand audio
None
Selena_Gomez_Interview.wav
Sending data to Google Speech Recognition
Success
like figuring out what my
Hilarious Job Interview.wav
Sending data to Google Speech Recognition
Success
a three-year-old boys went to fly fish job the guys from Shoreline only hard one
but said I think I'll do them last y'all question whoever gave me the best thing
to her on her yes for Stovall sit in your opinion what's the fastest thing in
the world no more set of thought he should have thought it ain't
sharapova_interview.wav
Sending data to Google Speech Recognition
Success
I wanted to ask you how much wtt has meant to you over the years and how much
you enjoyed participating in this kind of even I've been part of this event ever
since I was a little girl ever since I won major titles or big tournaments it
was really the event that gave me the experience that I needed at that age I
would travel from City to City competing every single night and I remember
leaving Wilson tennis I went on to win a a smaller WTA event and it was really
the beginning of my career and so yeah I'm very grateful for the opportunities
to still be able to come out on the court I'm sorry I'm not playing this evening
```

```
but for a good reason but yeah this is this tournament I played on world team
tennis on this court so great memories to be back here I think being played here
against each other I'm telling you a couple minutes ago I still very much loved
the game but every athlete whether you're a female or male as an athlete you
have to say I'm finished at one point and for me that came this week figure it
out for the next match some of the best part of this game
the interview.wav
Sending data to Google Speech Recognition
Success
oh hey you must be here for the job nope. Tell me a little bit about yourself of
course that's why I'm here astrology is real zodiac sign is a Leo
trevor_noah.wav
Sending data to Google Speech Recognition
Success
hi there I'm Trevor Noah and I want to interview you right here on the set of
The Daily Show seriously it's all a part of an amazing initiative to support a
program called education changemakers and it's going to be a lot of fun I'll put
you up in a fancy hotel in New York City and I'll even fly you out here I mean
not literally I don't have a pilot's license like a real pilot with a real
mustache will fly you out here and then you and me will hang out and you can
even bring a friend with you and if you don't to do that you can bring me with
you
Mark_Zuckerberg_Interview.wav
Sending data to Google Speech Recognition
Success
and now there's a new form of cyber matchmaking College networking website is
this perhaps the next big thing in Hartford and Stillman co-creator of lively
and universities West Match.com and then in Watertown Massachusetts Mark
Zuckerberg free creator of Harvard the facebook.com and I guess we'll start with
you Dan at what compelled you to do this
```

[37]:
```
aud_path = "/Users/sunny/Documents/CCA Spring 2020/Content-Analysis-2020-master/
 ↪data/transcript/wav_files/"
recog_list = []
for file_name in os.listdir(aud_path):
    print(file_name)
    if fnmatch.fnmatch(file_name, '*.wav'):
        rec = speechRec(aud_path + file_name)
        print(rec)
        recog_list.append(rec)
```

```
Freddie_Mercury_Interview.wav
Sending data to Google Speech Recognition
Success
What BERT is Not.wav
Sending data to Google Speech Recognition
Could not request results from Google service; recognition connection failed:
[Errno 32] Broken pipe
```

```
Freya_Zywoo_interview.wav
Sending data to Google Speech Recognition
Success
Ted.wav
Sending data to Google Speech Recognition
Success
The_Devil_Wears_Prada.wav
Sending data to Google Speech Recognition
Google could not understand audio
Selena_Gomez_Interview.wav
Sending data to Google Speech Recognition
Success
Hilarious Job Interview.wav
Sending data to Google Speech Recognition
Success
sharapova_interview.wav
Sending data to Google Speech Recognition
Success
the interview.wav
Sending data to Google Speech Recognition
Success
trevor_noah.wav
Sending data to Google Speech Recognition
Success
Mark_Zuckerberg_Interview.wav
Sending data to Google Speech Recognition
Success
```

[55]:
```
IPython.display.Audio("/Users/sunny/Documents/CCA Spring 2020/
↪Content-Analysis-2020-master/data/transcript/wav_files/sharapova_interview.
↪wav")
```

[55]:
```
<IPython.lib.display.Audio object>
```

[62]:
```
truth = pd.read_csv('/Users/sunny/Documents/CCA Spring 2020/
↪Content-Analysis-2020-master/data/transcript/transcript/sharapova_interview.
↪csv')
```

[66]:
```
truth_si = ''
for text in truth['text']:
    truth_si = truth_si + ' ' + text
truth_si
```

[66]:
```
" they wanted to ask you how much wtt has meant to you over the years and how
much you enjoy participating in these kind of evens I've been part of this event
ever since I was a little girl ever since I wann major titles or big tournaments
it was really the event that gave me the experience that I needed at that age I
```

would travel from city to city competing every single night and I remember leaving world TeamTennis I went on to win a smaller WTA event and it was really the beginning of my career and so yeah I'm very grateful for the opportunities to still be able to come out on the court I'm sorry I'm not playing this evening but for a good reason but yeah this is an incredible event I played this tournament I've played World Team Tennis on this court so great memories to be back here I think we even played here against each other [Applause] if you step away from the game what are you most excited about just to give my body a little break it's been a long 28 years of my life so many reflections in this past week after the announcement took a long time to write that essay it's never I guess it's it's never an easy time right to step away from something that you love but it's brought me incredible moments throughout my life tough moments as well and to still think the toughest thing as I was just telling you a couple minutes ago is that I still very much love the game but every athlete whether you're a female or male as an athlete you have to say I'm finished at one point and for me that came this week yeah it comes for everyone what is the thing that you're gonna miss the most about the game competing I loved working with my team I was so fortunate to work with some of the best people around the world from coaches to physical therapists to hitting partners I really had phenomenal people on on my team that I could call my friend and I can call them right now and have a great conversation I'm incredibly proud of that and competing there's nothing like it no matter what I'll choose to do in my future that moment of victories and match points and and losing and then having to figure it out for the next match it's one of the best part of this game yeah well we all gonna miss you big time and thanks like for everything you've done thank you"

[71]:
```
hypothesis = recog_list[7]
hypothesis
```

[71]: "I wanted to ask you how much wtt has meant to you over the years and how much you enjoyed participating in this kind of even I've been part of this event ever since I was a little girl ever since I won major titles or big tournaments it was really the event that gave me the experience that I needed at that age I would travel from City to City competing every single night and I remember leaving Wilson tennis I went on to win a a smaller WTA event and it was really the beginning of my career and so yeah I'm very grateful for the opportunities to still be able to come out on the court I'm sorry I'm not playing this evening but for a good reason but yeah this is this tournament I played on world team tennis on this court so great memories to be back here I think being played here against each other I'm telling you a couple minutes ago I still very much loved the game but every athlete whether you're a female or male as an athlete you have to say I'm finished at one point and for me that came this week figure it out for the next match some of the best part of this game"

[72]:
```
# use a package to calculate word-error rate
#pip install jiwer
wer(truth_si, hypothesis)
```

[72]: 0.5563063063063063

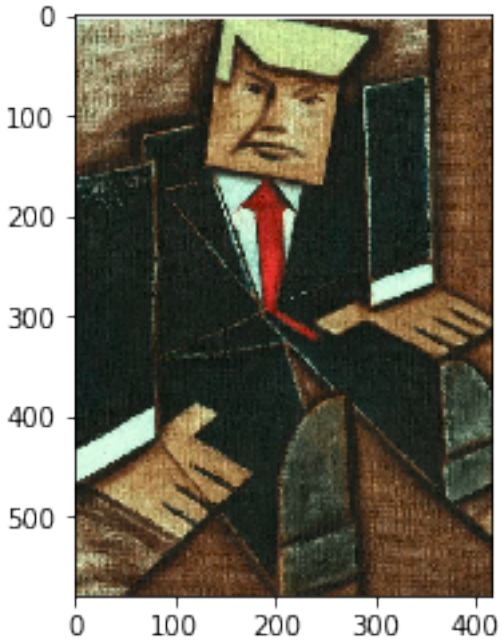
3 Image analysis

[47]:
```
import os
import fnmatch
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

[59]:
```
im_path = "/Users/sunny/Documents/CCA Spring 2020/Content-Analysis-2020-master/
 ↪data/"
original_im = []
for file_name in os.listdir(im_path):
    if fnmatch.fnmatch(file_name, '*.jpg'):
        print(file_name)
        image = PIL.Image.open(im_path + file_name)
        original_im.append(image)
```

```
donald-trump-abstract_paintings.jpg
trump_realism.jpg
trump_outdoor.jpg
trump.jpg
```

[57]:
```
trump_abstract = plt.imshow(mpimg.imread(im_path+␣
 ↪'donald-trump-abstract_paintings.jpg'));
```



[58]:
```
trump_realism = plt.imshow(mpimg.imread(im_path+ 'trump_realism.jpg'));
```

[ ]:
```
```

[ ]:
```
```

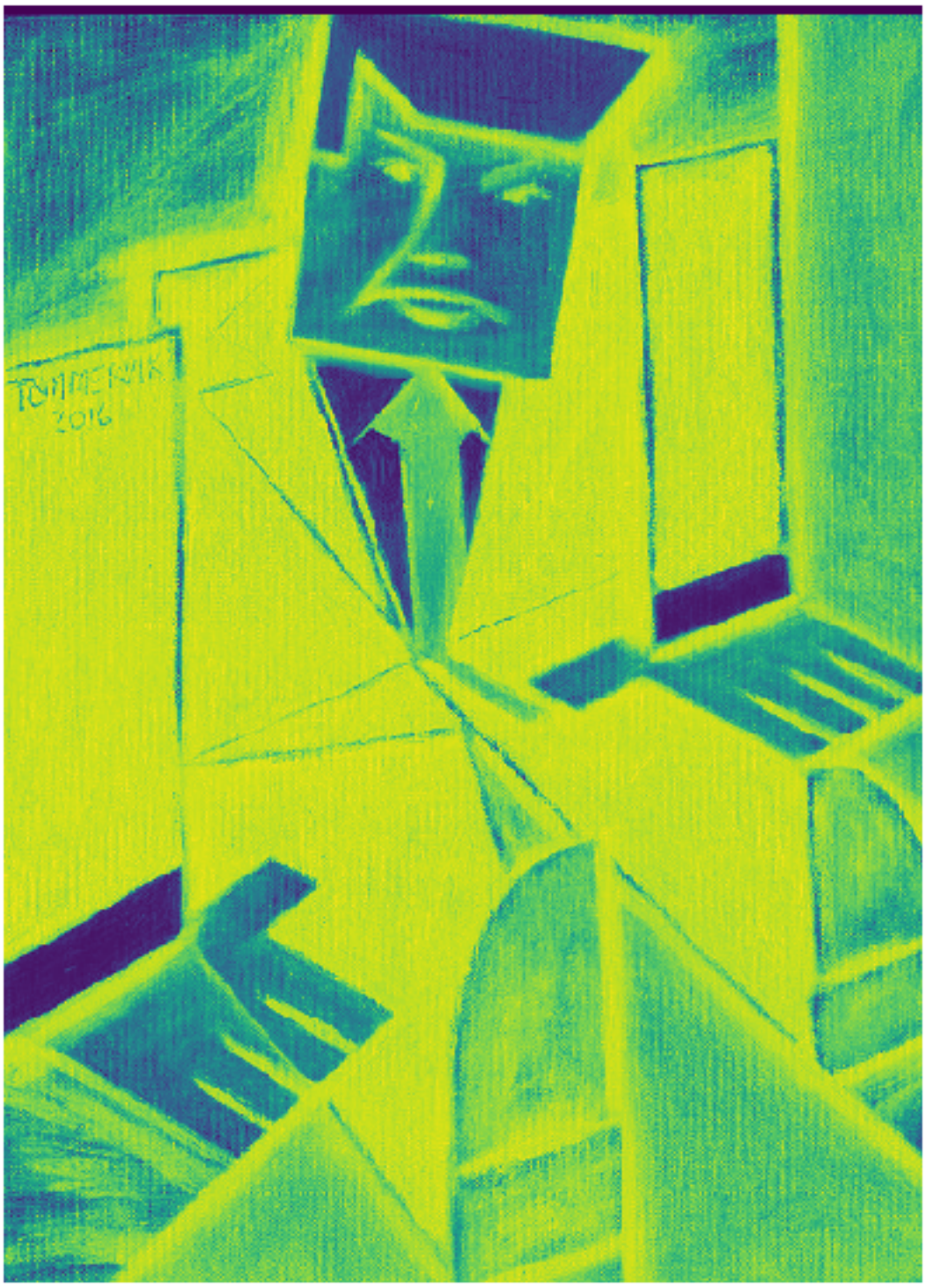
[45]:
```
gray_im = []
gray_im_arr = []
for image in original_im:
    image_gray = PIL.ImageOps.invert(image.convert('L'))
    image_grayArr = np.asarray(image_gray)
    print(image_grayArr.shape)
    gray_im.append(image_gray)
    gray_im_arr.append(image_grayArr)
gray_im
```

```
(580, 417)
(744, 1240)
(532, 800)
(478, 624)
(3300, 2550)
```

[45]:
```
[<PIL.Image.Image image mode=L size=417x580 at 0x13632FF10>,
 <PIL.Image.Image image mode=L size=1240x744 at 0x13632F8D0>,
 <PIL.Image.Image image mode=L size=800x532 at 0x13632FED0>,
```

```
 <PIL.Image.Image image mode=L size=624x478 at 0x136E60090>,
 <PIL.Image.Image image mode=L size=2550x3300 at 0x136E60ED0>]
```

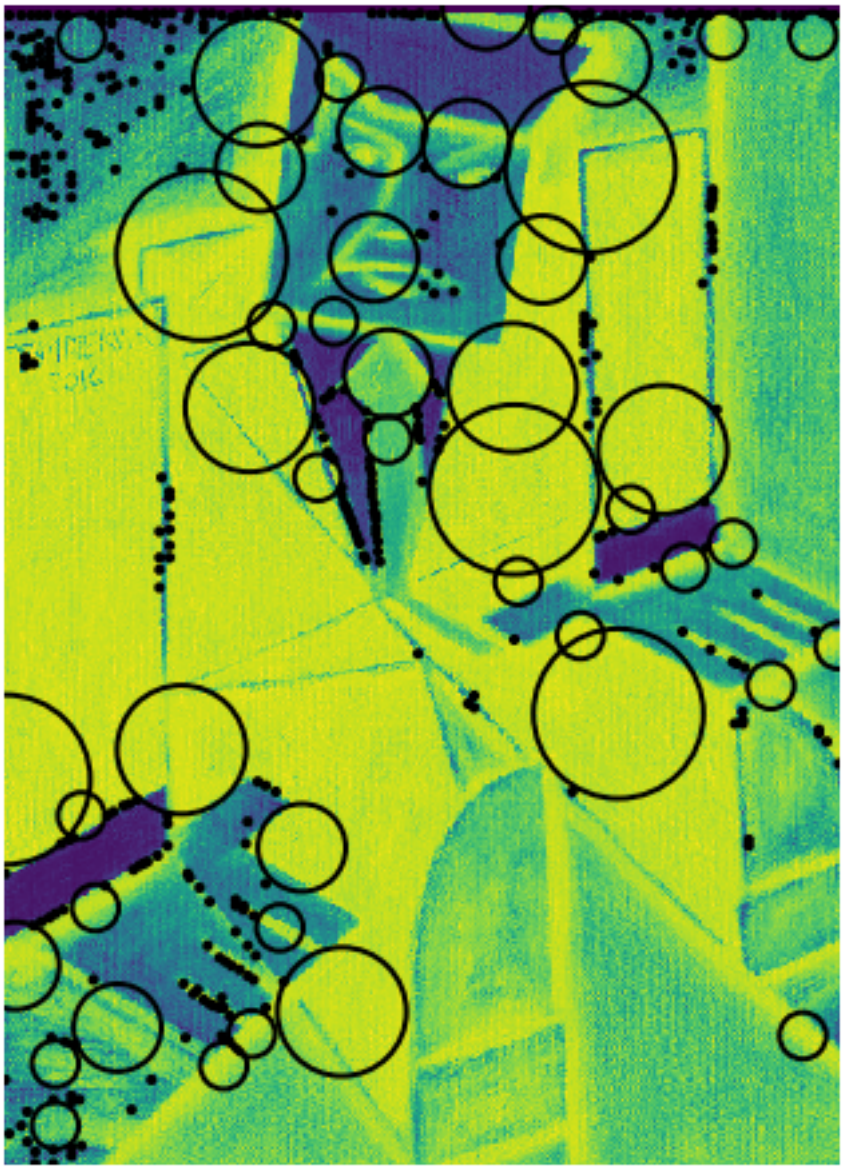
[34]:
```
# Check
fig, ax = plt.subplots(figsize = (10, 10))
ax.imshow(gray_im_arr[0]) #No third dimension
ax.axis('off')
plt.show()
```



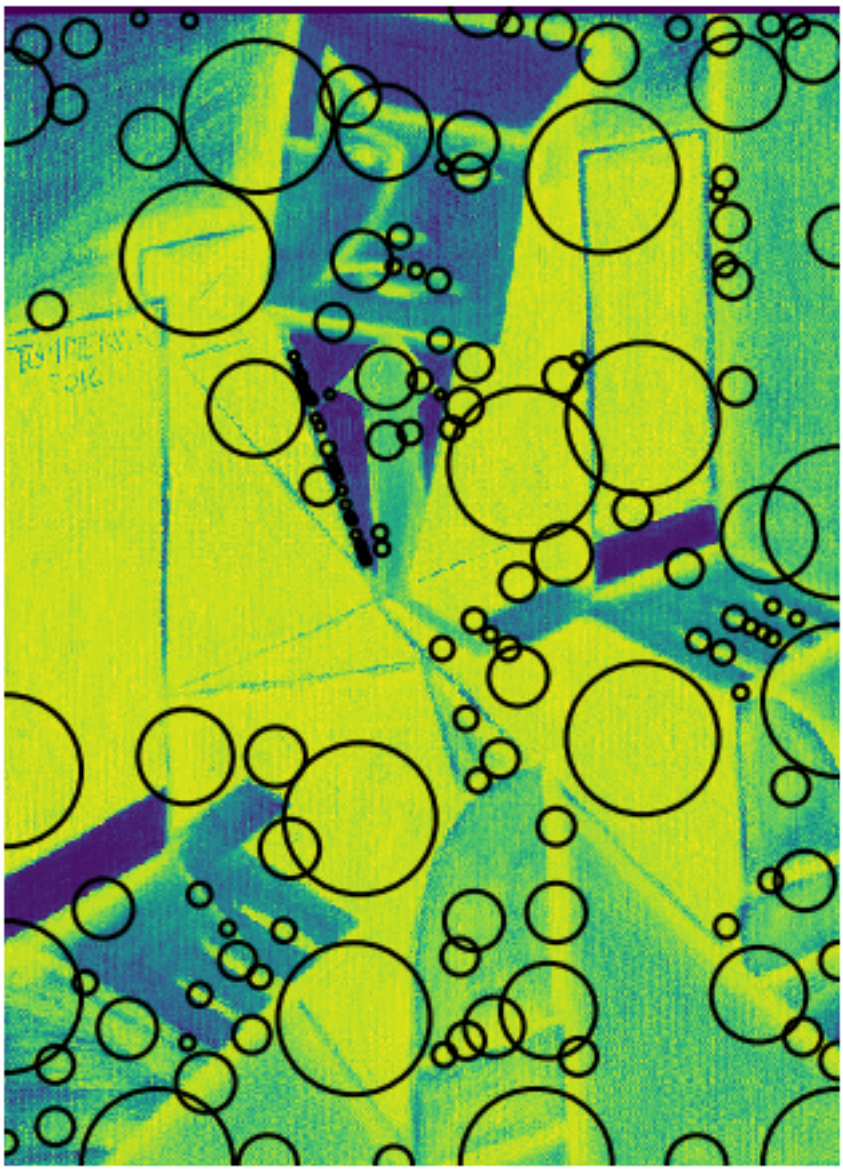
[37]:
```
for i, image in enumerate(gray_im_arr[:2]):
    for j in range(3):
        if j == 0:
```

```
            print('laplacian of gaussian')
            ax = log_plot(image)
            plt.show()
        elif j == 1:
            print('difference of gaussian')
            ax = dog_plot(image)
            plt.show()
        elif j == 2:
            print('determinant of hessian')
            ax = doh_plot(image)
            plt.show()
```

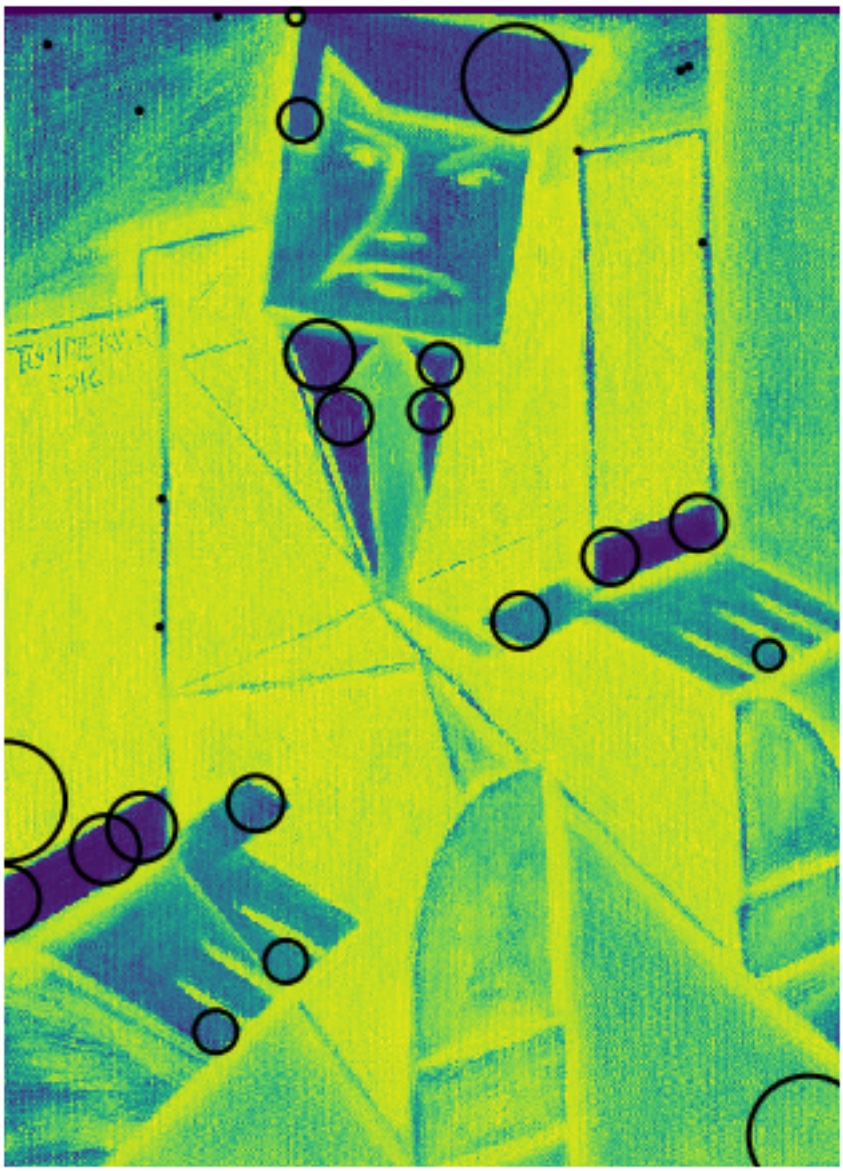
```
laplacian of gaussian
```



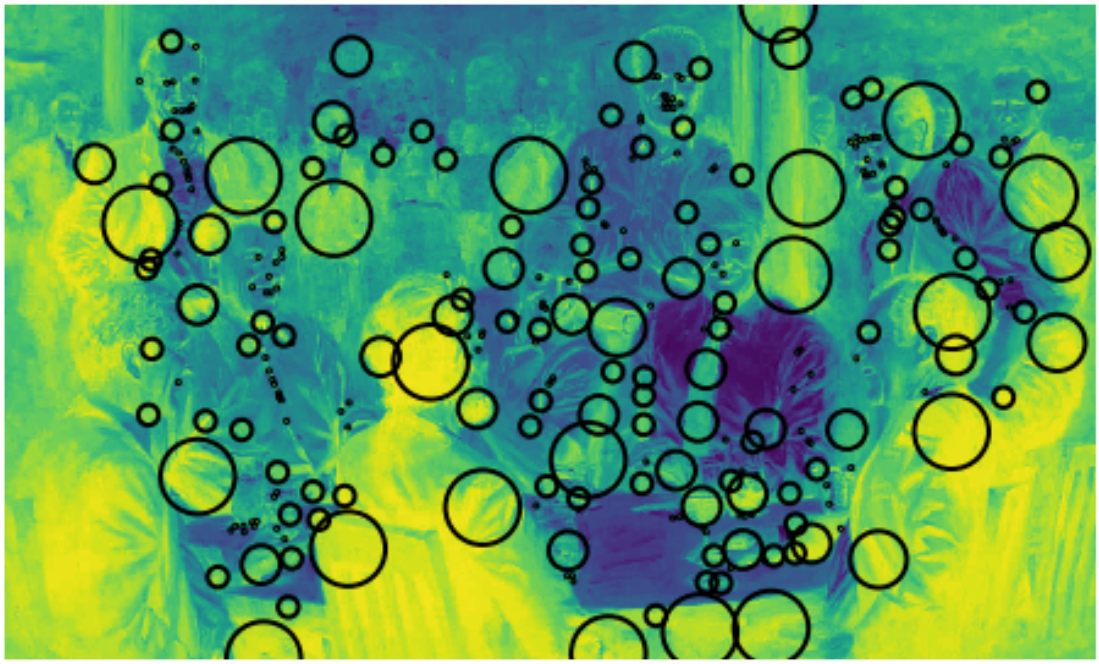
difference of gaussian



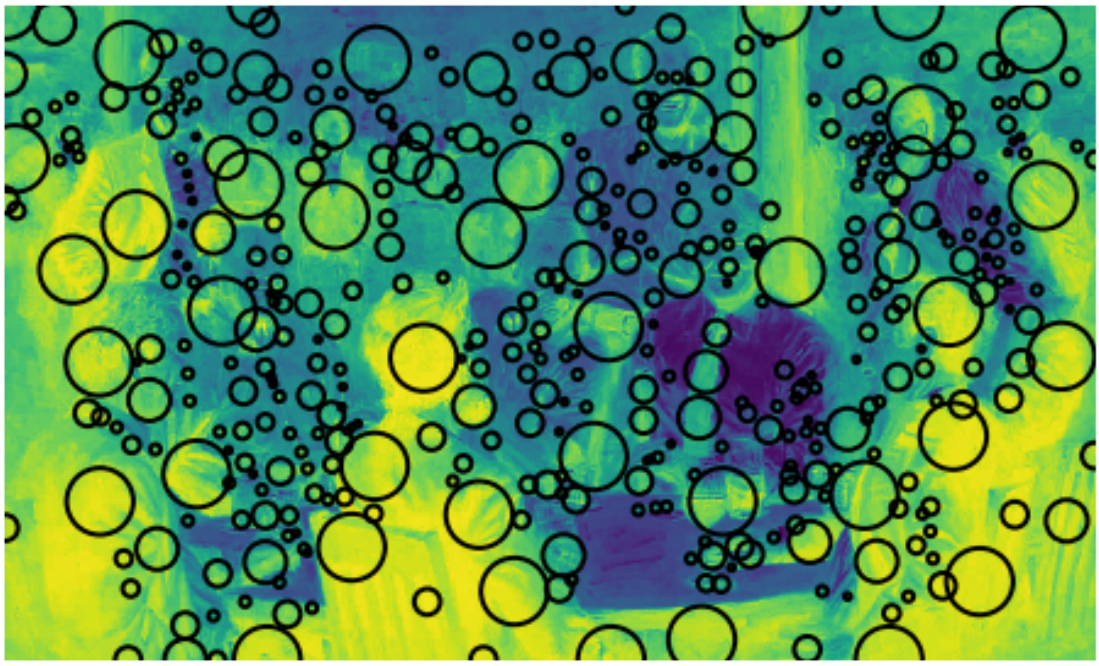
determinant of hessian



laplacian of gaussian



difference of gaussian



determinant of hessian

[20]:
```
image = PIL.Image.open('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/trump.jpg')
image_outdoor = PIL.Image.open('/Users/sunny/Documents/CCA Spring 2020/
 ↪Content-Analysis-2020-master/data/trump_outdoor.jpeg')
```

[3]:
```
image
```

[3]:

[21]:
```
image_outdoor
```

[21]:

[4]:
```
imageArr = np.asarray(image)
imageArr.shape
```

[4]:
```
(478, 624, 3)
```

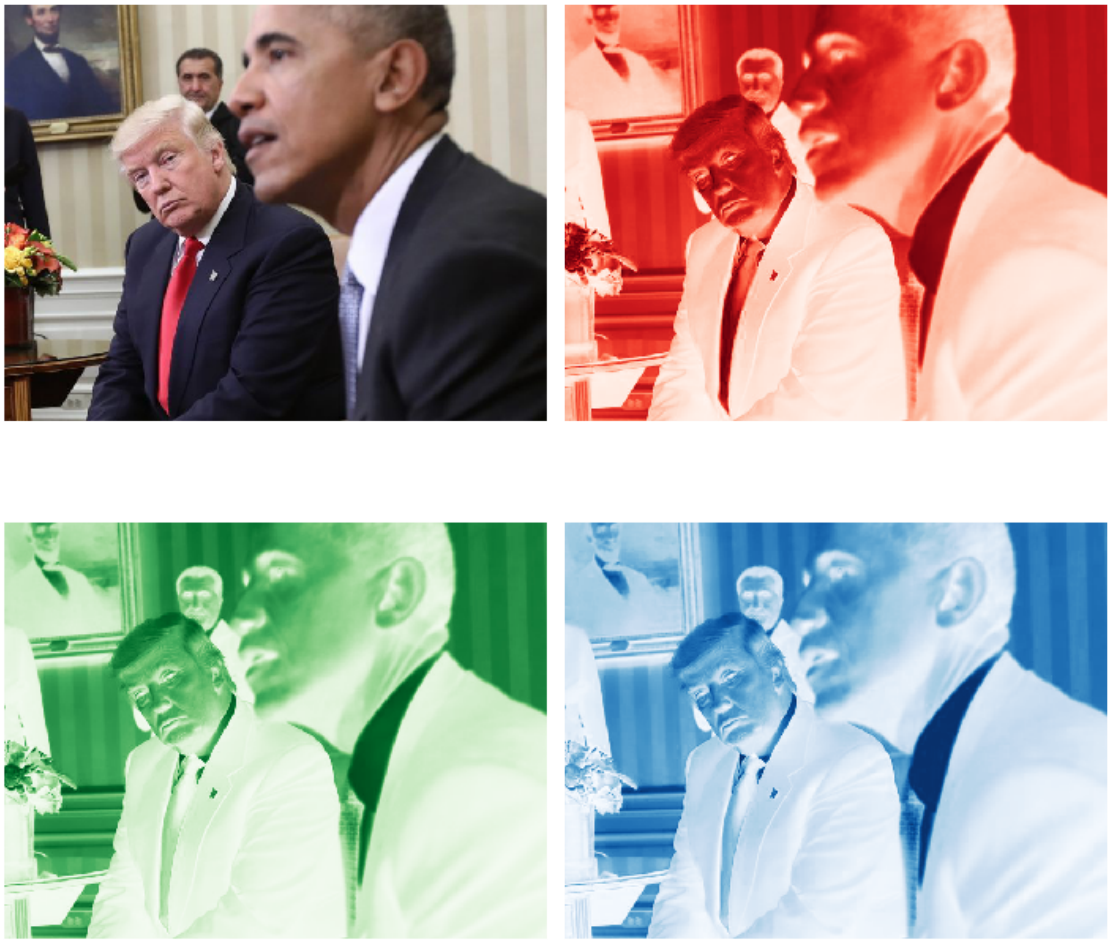
[22]:
```
imageArr_o = np.asarray(image_outdoor)
imageArr_o.shape
```

[22]:
```
(532, 800, 3)
```

The image we have loaded is a raster image, meaning it is a grid of pixels. Each pixel contains 1-4 numbers giving the amounts of color contained in it. In this case, we can see it has 3 values per pixel, these are RGB or Red, Green and Blue values. If we want to see just the red, we can look at just that array:

[5]:
```
fig, axes = plt.subplots(figsize = (10, 10), ncols=2, nrows=2)
axeIter = iter(axes.flatten())
colours = ["Reds", "Greens", "Blues"]
ax = next(axeIter)
ax.imshow(imageArr)
ax.axis('off')

for i in range(3):
    ax = next(axeIter)
```

```
    ax.imshow(imageArr[:,:,i], cmap=colours[i]) #The order is R G B, so 2 is␣
 ↪the Blue
    ax.axis('off')
plt.tight_layout()
plt.show()
```



[23]:
```
fig, axes = plt.subplots(figsize = (10, 10), ncols=2, nrows=2)
axeIter = iter(axes.flatten())
colours = ["Reds", "Greens", "Blues"]
ax = next(axeIter)
ax.imshow(imageArr_o)
ax.axis('off')

for i in range(3):
    ax = next(axeIter)
    ax.imshow(imageArr_o[:,:,i], cmap=colours[i]) #The order is R G B, so 2 is␣
 ↪the Blue
```

```
    ax.axis('off')
plt.tight_layout()
plt.show()
```

A grayscale image is defined by its pixel intensities (and a color image can be defined by its red, green, blue pixel intensities). show the grayscale of the image

[7]:
```
image_gray = PIL.ImageOps.invert(image.convert('L'))
image_grayArr = np.asarray(image_gray)
image_grayArr.shape
```

[7]: (478, 624)

[24]:
```
image_gray_o = PIL.ImageOps.invert(image_outdoor.convert('L'))
image_grayArr_o = np.asarray(image_gray_o)
image_grayArr_o.shape
```

[24]: (532, 800)

[8]:
```
imgRatio = imageArr.shape[0] / imageArr.shape[1]
fig, ax = plt.subplots(figsize = (15, 15))
ax.imshow(image_grayArr) #No third dimension
ax.axis('off')
plt.show()
```

[25]:
```
imgRatio_o = imageArr_o.shape[0] / imageArr_o.shape[1]
fig, ax = plt.subplots(figsize = (15, 15))
ax.imshow(image_grayArr_o) #No third dimension
ax.axis('off')
plt.show()
```

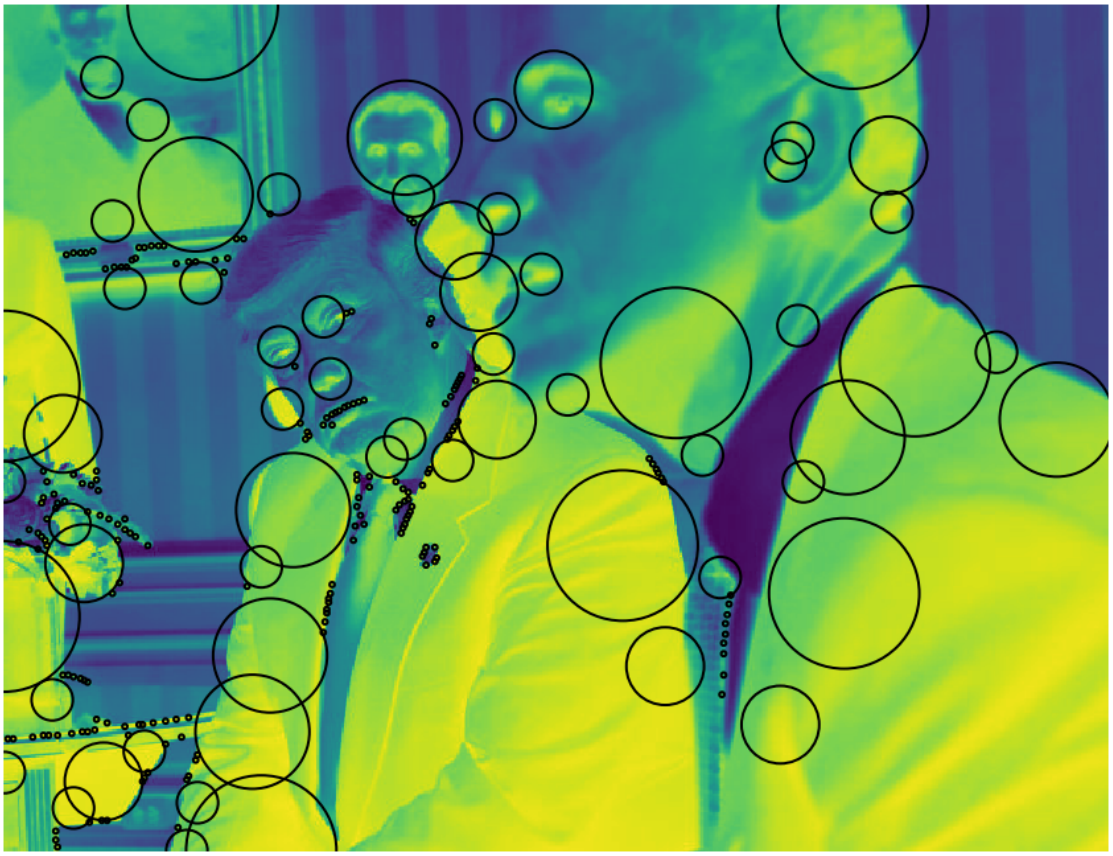
The blob detector computes the Laplacian of Gaussian (LoG) images with successively increasing standard deviation and stacks them up in a cube. Blobs are local maximas within this cube. Detecting larger blobs is slower because of larger kernel sizes during convolution. Bright blobs on dark backgrounds are detected.

[31]:
```
def log_plot(image_grayArr):
    # laplacian of gaussian
    blobs_log = skimage.feature.blob_log(image_grayArr, max_sigma=30,␣
 ↪num_sigma=5, threshold=.1)
    blobs_log[:, 2] = blobs_log[:, 2] * np.sqrt(2) #Radi
    fig, ax = plt.subplots(figsize = (8, 8))
    ax.axis('off')

    plt.imshow(image_grayArr, interpolation='nearest')
    for blob in blobs_log:
        y, x, r = blob
        c = plt.Circle((x, y), r, linewidth=2, fill=False)
        ax.add_patch(c)
    return ax

def dog_plot(image_grayArr):
    # difference of gaussian
```

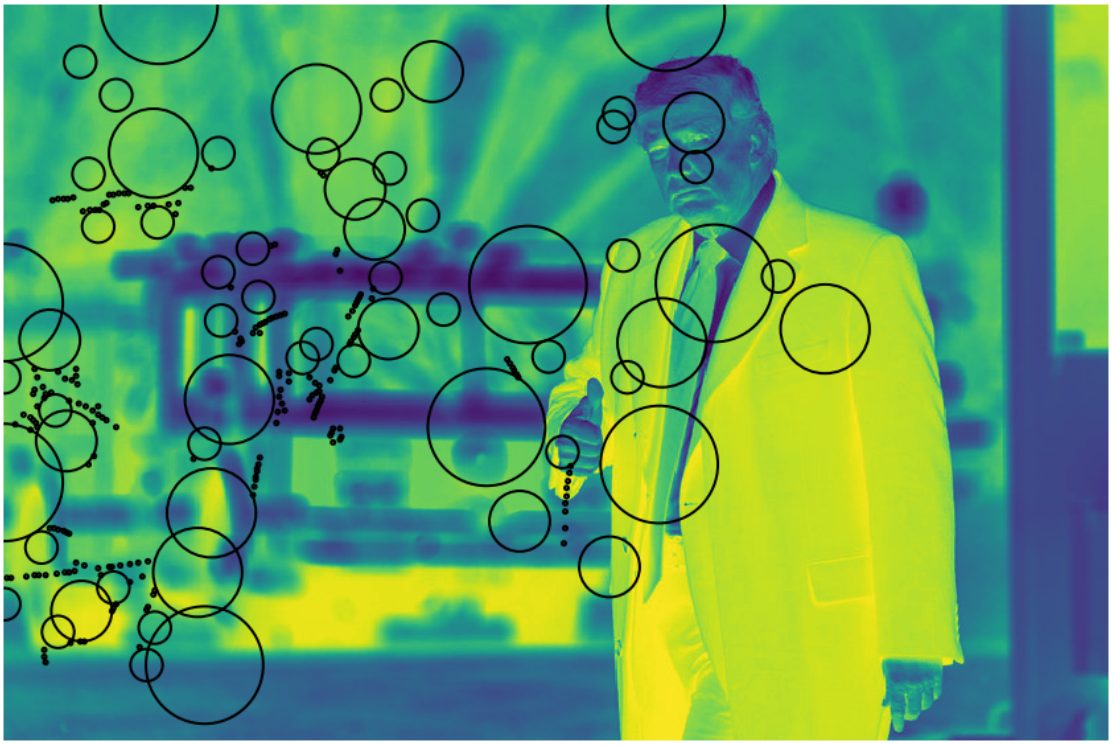
```
    blobs_dog = skimage.feature.blob_dog(image_grayArr, max_sigma=30,␣
 ↪threshold=.1)
    blobs_dog[:, 2] = blobs_dog[:, 2] * np.sqrt(2)
    fig, ax = plt.subplots(figsize = (8, 8))
    ax.axis('off')

    plt.imshow(image_grayArr, interpolation='nearest')
    for blob in blobs_dog:
        y, x, r = blob
        c = plt.Circle((x, y), r, linewidth=2, fill=False)
        ax.add_patch(c)
    return ax

def doh_plot(image_grayArr):
    # determinant of hessian
    blobs_doh = skimage.feature.blob_doh(image_grayArr, max_sigma=30,␣
 ↪threshold=.01)
    fig, ax = plt.subplots(figsize = (8, 8))
    ax.axis('off')

    plt.imshow(image_grayArr, interpolation='nearest')
    for blob in blobs_doh:
        y, x, r = blob
        c = plt.Circle((x, y), r, linewidth=2, fill=False)
        ax.add_patch(c)
    return ax
```

[ ]:

[9]:
```
blobs_log = skimage.feature.blob_log(image_grayArr, max_sigma=30, num_sigma=5,␣
 ↪threshold=.1)
blobs_log[:, 2] = blobs_log[:, 2] * np.sqrt(2) #Radi
fig, ax = plt.subplots(figsize = (15, 15))
ax.axis('off')

plt.imshow(image_grayArr, interpolation='nearest')
for blob in blobs_log:
    y, x, r = blob
    c = plt.Circle((x, y), r, linewidth=2, fill=False)
    ax.add_patch(c)
```

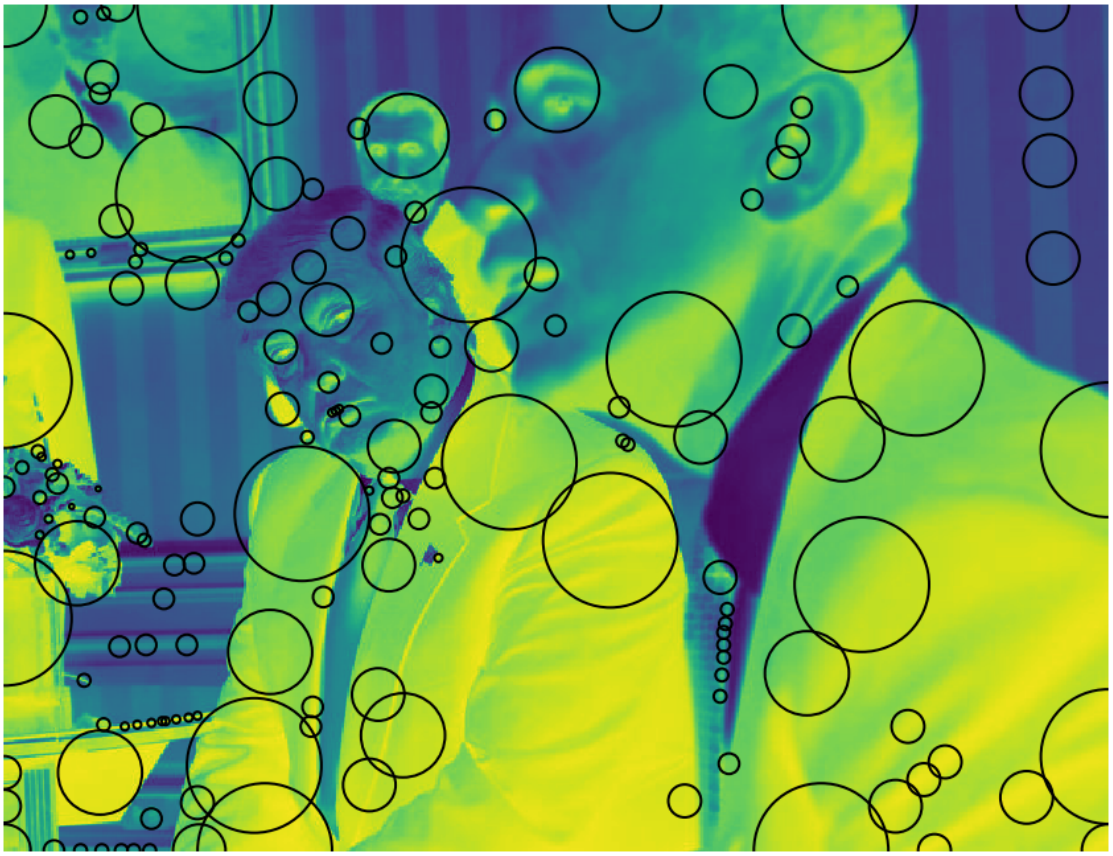


[28]:
```
blobs_log_o = skimage.feature.blob_log(image_grayArr_o, max_sigma=30,␣
 ↪num_sigma=5, threshold=.1)
blobs_log_o[:, 2] = blobs_log_o[:, 2] * np.sqrt(2) #Radi
fig, ax = plt.subplots(figsize = (15, 15))
ax.axis('off')

plt.imshow(image_grayArr_o, interpolation='nearest')
for blob in blobs_log:
    y, x, r = blob
    c = plt.Circle((x, y), r, linewidth=2, fill=False)
    ax.add_patch(c)
```

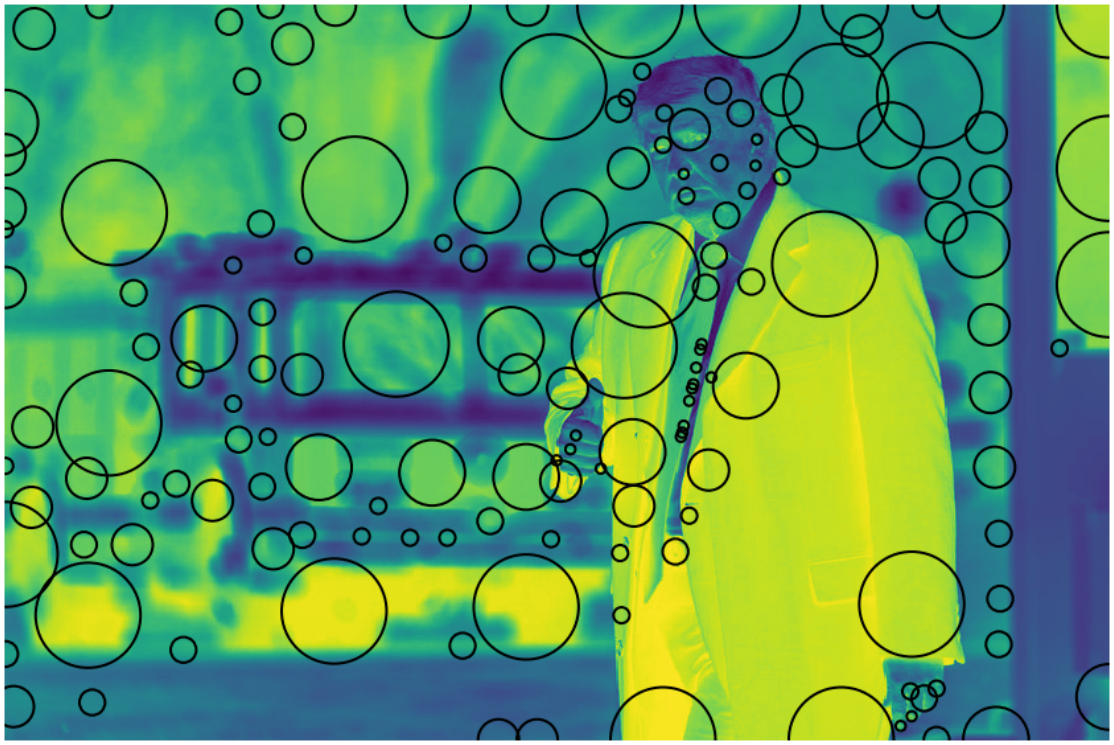


Second, we look at Difference of Gaussian (DoG), a much faster approximation of the LoG approach in which an image is blurred with increasing standard deviations and the difference between two successively blurred images are stacked up in a cube.

[10]:
```
blobs_dog = skimage.feature.blob_dog(image_grayArr, max_sigma=30, threshold=.1)
blobs_dog[:, 2] = blobs_dog[:, 2] * np.sqrt(2)
fig, ax = plt.subplots(figsize = (15, 15))
ax.axis('off')

plt.imshow(image_grayArr, interpolation='nearest')
for blob in blobs_dog:
    y, x, r = blob
    c = plt.Circle((x, y), r, linewidth=2, fill=False)
    ax.add_patch(c)
```



[29]:
```
blobs_dog_o = skimage.feature.blob_dog(image_grayArr_o, max_sigma=30,␣
 ↪threshold=.1)
blobs_dog_o[:, 2] = blobs_dog_o[:, 2] * np.sqrt(2)
fig, ax = plt.subplots(figsize = (15, 15))
ax.axis('off')

plt.imshow(image_grayArr_o, interpolation='nearest')
for blob in blobs_dog_o:
    y, x, r = blob
    c = plt.Circle((x, y), r, linewidth=2, fill=False)
    ax.add_patch(c)
```



Finally, we consider the Determinant of Hessian (DoH) approach. The Hessian matrix or Hessian is a square matrix of second-order partial derivatives $\frac{\partial^2 f}{\partial x_i \partial x_j}(x_1^*, \ldots, x_n^*)$ and is calculated on square pixel patches of the image. The determinant is the scaling factor of each patch. This approach is fastest and detects blobs by finding maximas in this matrix (of the Determinant of the Hessian of the image). Detection speed is independent of the size of blobs as the implementation uses box filters, $\begin{bmatrix} 1 & amp;1 & amp;1 \\ 1 & amp;1 & amp;1 \\ 1 & amp;1 & amp;1 \end{bmatrix}$, instead of Gaussians for the convolution. As a result, small blobs (< 3 pixels) cannot be detected accurately.

[11]:
```
blobs_doh = skimage.feature.blob_doh(image_grayArr, max_sigma=30, threshold=.01)
fig, ax = plt.subplots(figsize = (15, 15))
ax.axis('off')

plt.imshow(image_grayArr, interpolation='nearest')
for blob in blobs_doh:
    y, x, r = blob
    c = plt.Circle((x, y), r, linewidth=2, fill=False)
    ax.add_patch(c)
```

[30]:
```
blobs_doh_o = skimage.feature.blob_doh(image_grayArr_o, max_sigma=30,␣
 ↪threshold=.01)
fig, ax = plt.subplots(figsize = (15, 15))
ax.axis('off')

plt.imshow(image_grayArr_o, interpolation='nearest')
for blob in blobs_doh:
    y, x, r = blob
    c = plt.Circle((x, y), r, linewidth=2, fill=False)
    ax.add_patch(c)
```

Humans possess an incredible ability to identify objects in an image. Segmentation is the process of dividing an image into meaningful regions. All pixels belonging to a region should receive a unique label in an ideal segmentation.

Region Adjacency Graphs (RAGs) are a common data structure for many segmentation algorithms. First, we define regions through the SLIC algorithm that assigns a unique label to each region or a localized cluster of pixels sharing some similar property (e.g., color or grayscale intensity). Then we'll consider each region a node in a graph, and construct a region boundary RAG, where the edge weight between two regions is the average value of the corresponding pixels in edge_map along their shared boundary. Then edges below a specified threshold are removed and a connected component is labeled as one region.

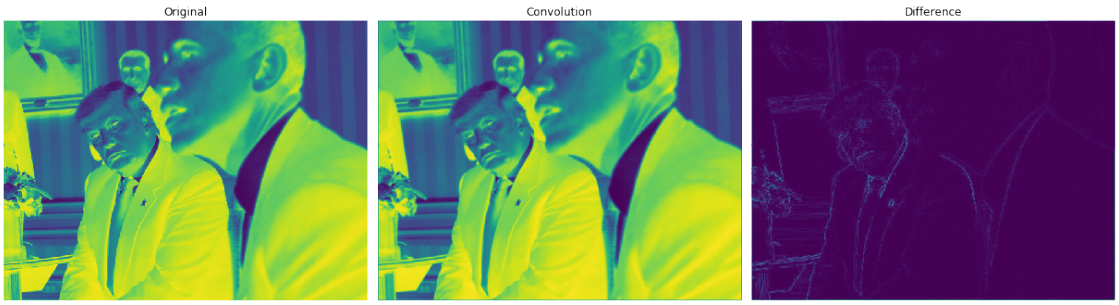
NOTE: A change to networkx means the following code might not work on your laptop.

[12]:
```
def normalizeConvo(a_c):
    s = np.sum(a_c.flatten())
    return a_c / s

def displayConvolution(img, convArray, threshold = None):
    img_array = np.asarray(img).astype('uint8')
```

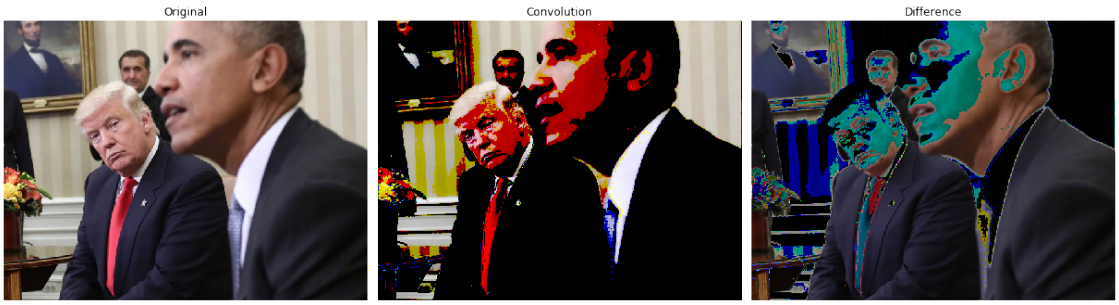
```
    if len(img_array.shape) > 2:
        conv = np.zeros(img_array.shape)
        for i in range(img_array.shape[2]):
            conv[:,:,i] = scipy.ndimage.convolve(img_array[:,:,i],␣
↪normalizeConvo(convArray), mode='constant')
    else:
        conv = scipy.ndimage.convolve(img_array, normalizeConvo(convArray),␣
↪mode='constant')
    conv = conv.astype('uint8')
    if threshold is not None:
        if threshold < 1:
            threshold = threshold * 255
        conv[conv < threshold] = 0

    conv_image = PIL.Image.fromarray(conv)
    fig, (ax1, ax2, ax3) = plt.subplots(ncols=3, figsize = (17, 10))
    #This is to deal with some annoying plt/PIL stuff
    if len(img_array.shape) > 2:
        ax1.imshow(img)
    else:
        ax1.imshow(img_array)
    ax1.axis('off')
    ax1.set_title("Original")
    #return conv_image
    if len(img_array.shape) > 2:
        ax2.imshow(conv_image)
    else:
        ax2.imshow(np.asarray(conv_image))
    ax2.axis('off')
    ax2.set_title("Convolution")

    diff = PIL.ImageChops.difference(conv_image, img)
    if len(img_array.shape) > 2:
        ax3.imshow(diff)
    else:
        ax3.imshow(np.asarray(diff))
    ax3.axis('off')
    ax3.set_title("Difference")
    plt.tight_layout()
    plt.show()
    #Not returning anything to make displaying nicer
```

Here are a fistfull kenels to try. In image processing, kernels (also convolution matrices or masks) are small matrices. They are used for blurring, sharpening, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image by adding each element of the image to its local neighbors, weighted by the kernel, as shown below:

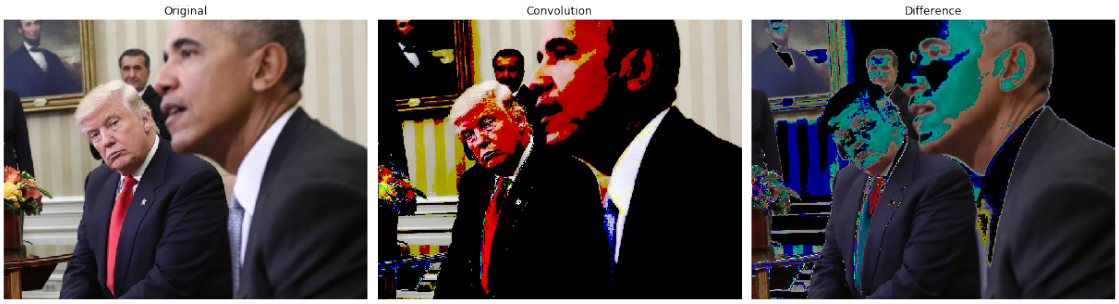
[13]:
```
smoothingKernel = np.array([
    [1, 1, 1],
    [1, 2, 1],
    [1, 1, 1]])
c = displayConvolution(image_gray, smoothingKernel)
```



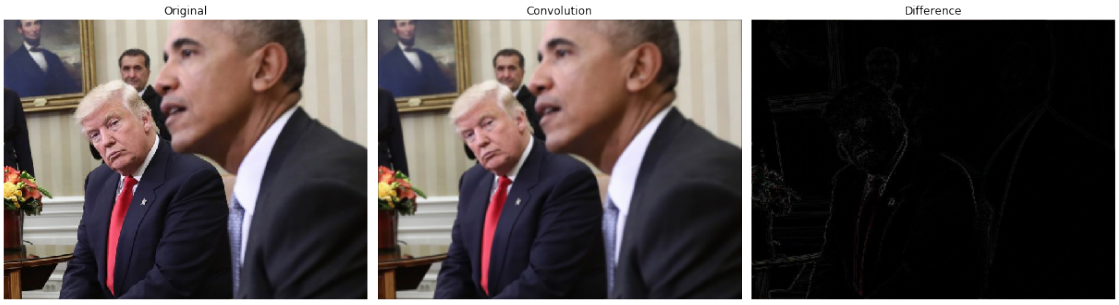
[14]:
```
verticalLineKernel = np.array([
    [1, 0, -1],
    [0, 8, 0],
    [1, 0, -1]])
displayConvolution(image, verticalLineKernel, threshold=.6)
```



[15]:
```
diagonalLineKernel = np.array([
    [-1, 0, -1],
    [0, 8, 0],
    [1, 0, -1]])
displayConvolution(image, diagonalLineKernel, threshold=.6)
```



[16]:
```
blurKernel = np.array([
    [1, 2, 4, 2, 1],
    [2, 4, 8, 4, 2],
    [4, 8, 16, 8, 4],
    [2, 4, 8, 4, 2],
    [1, 2, 4, 2, 1]])
displayConvolution(image, blurKernel)
```



[17]:
```
SharpenKernel = np.array([
    [ 0,  0,  -1,  0,  0],
    [ 0, -1,  -2, -1,  0],
    [-1, -2,  20, -2, -1],
    [ 0, -1,  -2, -1,  0],
    [ 0,  0,  -1,  0,  0]])
displayConvolution(image, SharpenKernel)
```

[18]:
```
differentKernel = np.array([
    [ 1, 0,  0, 0,  -1],
    [ 1, 0,  0, 0,  -1],
    [ 1, 0,  1, 0,  -1],
    [ 1, 0,  0, 0,  -1],
    [ 1, 0,  0, 0,  -1]])
#This looks neat
displayConvolution(image, differentKernel)
```