LCO Synthesis by Sol-Gel Method

General Audience Abstract This summer I had the opportunity to independently work on a research project involving the process development for the synthesis of Lanthanum Cobalt Oxide. This material is of interest because the method by which we are producing it, the Sol-Gel method, potentially provides for a cost-effective way to produce a photocatalytic material. Specifically, Lanthanum Cobalt Oxide is promising in the potential application of water-splitting, in which a photocatalyst is used to drive a hydrogen evolution reaction and produce H2. H2 is a potentially high efficiency source of green energy and water-splitting provides a green method of H2 production. The outcome of this project was largely successful. At the onset of the summer, samples up to date displayed discontinuities and were far to thick for ideal electrical performance. By investigating multiple facets of the synthesis process and using tools such as Raman spectroscopy, X-ray diffraction spectroscopy, and optical imaging I wad able to produce LCO samples with continuous films of thicknesses below 500 nanometers. The biggest lesson I learned from this summer was the importance of doing my research before jumping into my exploration. Many problems faced along the way could have been much more easily resolved if I had taken the time to reference literature instead of attempting to figure out my issues head on in the laboratory. I was also able to use this internship as an opportunity to refine my skills as a programmer. Much of the data processing necessary for this project was done in the Python IDE Spyder. Specifically, I was able to improve my ability to use Python to extract data from large data sets and apply mathematically stipulated algorithms to my data in order to convert to an easily visualizable format and extract important data metrics. Finally, I had the opportunity to meet some great people this summer. My mentor and coworkers were all very helpful and allowed me to have a successful experience at Sandia National Laboratories.

Abstract

The synthesis of Lanthanum Cobalt Oxide (LCO) via Sol-Gel method provides a potentially low-cost method of production for a P-type photocatalytic. LCO was prepped from Lanthanum and Nitrate Precursors in Aqueous solution. After a significant amount of water is evaporated, the solution is deposited onto SiO2 substrate via spin-deposition method. After annealing, the samples produce thin film LCO that display thickness of sub-500 nanometers. The samples material profile is confirmed by both Raman spectroscopy and X-Ray Diffraction spectroscopy (XRD). Additionally, two-point probing tested the conductivity of the samples. The thin film samples display characteristics of a P-type photocatalytic and may potentially be used in the formation of a P-N junction for user in water-splitting applications.

Introduction

The importance of transitioning society's energy dependance from fossil fuels to green energy is of great importance for the longevity of nature and human society at large. One such potential energy source is hydrogen gas, which produces more energy per conversion unit when compared to fossil fuels and produces zero emissions upon combustion. The major drawback of hydrogen as a fuel source is that is very difficult to produce large quantities of it cleanly, and as such we have relied on hydrogen production methods that contribute to the production of greenhouse gasses as a biproduct of the hydrogen production process.

Methodology

Samples were fabricated using the following Procedure. First, stoichiometrically equivalent amounts of Lanthanum Nitrate Hexahydrate and Cobalt Nitrate Hexahydrate were dissolved in aqueous solution. After this Ethylenediaminetetraacetic acid in a 1:1 stoichiometric ratio with the Nitrates is added to the solution and left to chelate at room temperature for 30 minutes. After this, 0.3 grams of Polyethyleneimine (PEI)

was dissolved in aqueous solution. Once fully dissolved, the PEI solution is added to the above solution in a dropwise manner. In doing so, the solution should transition from a light pink color to a darker orange-brown color. The solution should be left stirring at room temperature for 24 hours, making sure for the color to return to a dark red shade before continuing to the next step. Once ready, the solution is heated in an oil bath at 150 degrees Celsius while being stirred. Once the solution reaches a viscosity still characteristic of water but with a deep dark red color the solution is spin deposited onto SiO2 substrate at 6000 rpm. SiO2 substrates were cleaned prior to deposition by being submerged and sonicated in ethanol, water, and acetone sequentially, to be followed by a UV/Ozone cleaning.

Raman

Raman mapping software was produced using Spyder, a Python IDE. I personally developed the software at Sandia for use as a mapping software for a general data type. As it became clear that Raman would be my main analysis tool for the project, I built upon the code to automate the data analysis process for data Raman spectra maps acquired using the Renishaw Raman microscope. The following features were added to streamline this process: Automatic Raman intensity map at user specified wavenumber, point Raman spectrum at user defined coordinate, and prominent peak detection and location. The Raman map is plotted by automatically extracting data from a txt file and using the data to plot an intensity map of only the wavenumber that the user specifies at the beginning of the script. The point Raman spectra is extracted by isolating data from a specific coordinate, pre-defined by the user, and plotting that intensity as a function of wavenumber. Finally, prominent peaks are extracted from the data file by detecting slope behavior of the graph and using a user defined threshold value to distinguish which local maximums constitute a legitimate peak. Once the peak is detected by the software, it will be marked in the Raman spectrum and its exact position in the spectrum is outputted into the python console for the user.

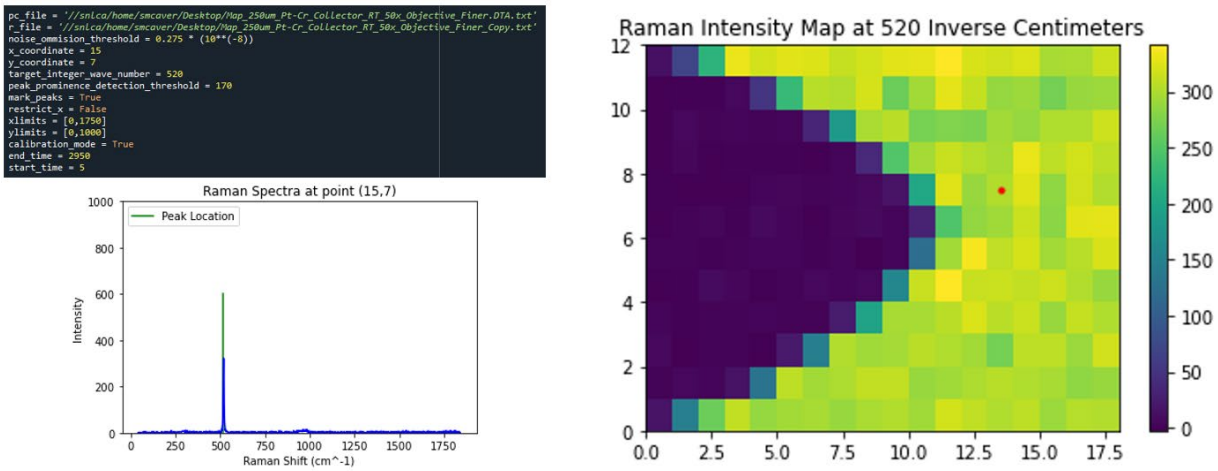


Figure 1: The figures above a typical panel of user defined values (Top Left), a Raman intensity map of 520 inverse centimeters (Right), and the individual Raman spectrum of the individual point 15,7 in the provided Raman map (Bottom Left).

Raman mapping scans were used to determine the material profile of the deposited perovskite film. Raman mapping scans using 532 nm laser revealed that many samples displayed characteristic Raman peaks of both Lanthanum Cobalt Oxide and a secondary phase perovskite, Cobalt Oxide. In order to address the issues, the following approaches were investigated.

In order to ensure that the cobalt and lanthanum precursors were fully reacting to form LCO, the pre-evaporation phase of the synthesis process was extended from its original time of 1 hour into greater time periods of 8 hours and eventually 24 hours.

Optical Imaging

Additionally, the deposited material was not continuous across the substrate. Instead, perovskite platelets formed on the material with variant thickness ranging from 1-10 microns across a single platelet. Platelets had lengths on the order of tens of microns across.

In order prevent the formation of platelets on the substrate and encourage the formation of a continuous thin film, the deposition viscosity was decreased from that of a paste to a more watery viscosity with dark red profile. Additionally, the spin deposition speed was increased from 1500 rpm to 3000 rpm and eventually to 6000 rpm.

XRD

XRD scans show characteristic LCO spectrum with peaks at roughly 32, 40, 47, and 59 degrees. The peaks are broad which may indicate poor crystallinity.

Our Raman analysis shows that the formation of secondary perovskite phases is a prevalent occurrence in many of the samples we fabricated. A general trend can be detected when comparing difference in the precursor formula's evaporation profile and the presence of these secondary phase Raman peaks, which primarily belong to Cobalt Oxide. Namely, when the solution is allowed longer periods of time to stir at room temperature before evaporation begins, higher purity LCO resulted after annealing.

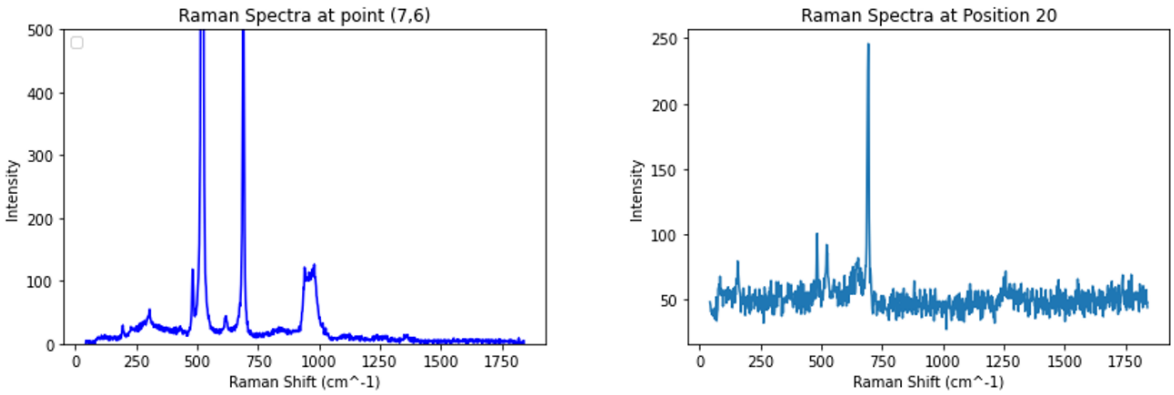


Figure 2: The above figure displays Raman spectra of two different samples that were formulated using different deposition methods. The Raman Spectra on the left displays a sample that was left to stir at room temperature for only 1 Hour before evaporation, while the Raman spectra on the right displays a sample that was allowed to mix for 24 Hours before evaporation. Most importantly, it can be observed that the 1 Hour mix spectrum displays characteristic peaks of secondary phase perovskites at ~200 and ~270 inverse centimeters. In contrast, the 24 Hour mix spectrum is absent of these secondary phase perovskite Raman peaks.

Raman also shows the effect that deposition viscosity has on the determination of film thickness. Initial samples were spin deposited once reaching levels of viscosity similar to a runny paste. These initial

samples had platelets of material with variant thickness. The general dimensions for these platelets were on the order of tens of microns across with thicknesses ranging between 1-10 microns throughout a platelet. It was determined that reducing the viscosity and increasing the spin rate mitigated these problems significantly. Final samples being deposited at speeds up to 6000 rpm with a viscosity similar to that of water displayed continuous films with thicknesses below 500 nanometers.

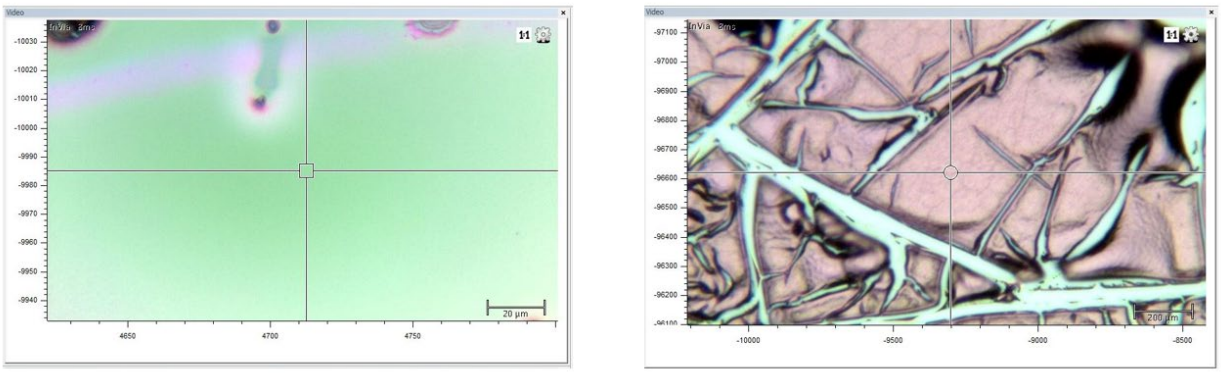


Figure 3: The figure above shows the optical images of two LCO samples in which different levels of viscosity were used when conducting the deposition step of the sample fabrication process. The image on the left shows a thin film of LCO deposited when the sample was at a watery viscosity level, which resulted in an LCO layer less than 500 nanometers in thickness. The image on the right shows a thicker layer of LCO deposited when the sample was at a thicker pasty viscosity level, which resulted in an LCO layer with variant thickness ranging from 1 micron to 10 microns.

Optical imaging shows the correlation between annealment temperature as well as deposition viscosity and film continuity. As previously mentioned, viscosity is not the only factor affecting film continuity. Films deposited at slower spin rates redder films. As a rule of thumb, the thicker the layer of LCO becomes the more it shifts from green to red.

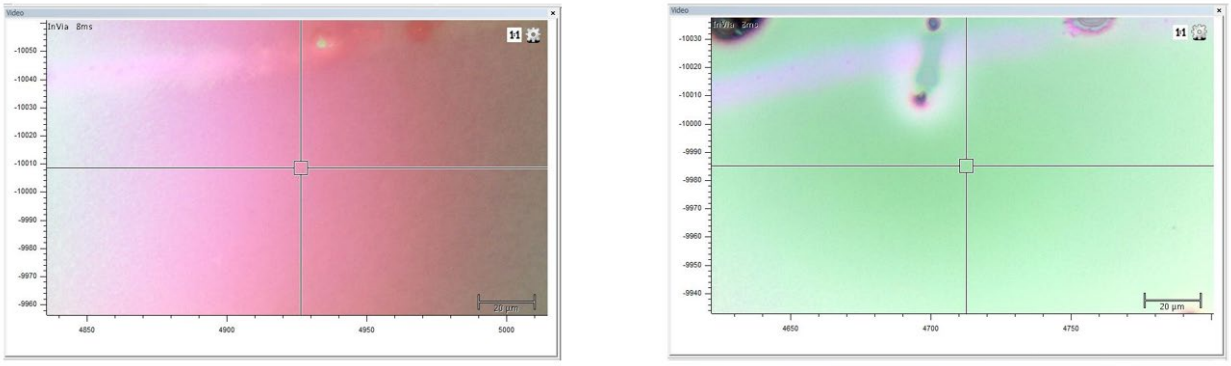


Figure 4: The figure above shows thin film LCO layers deposited at the same viscosity but at different deposition speeds. The image on the left is an optical image of a sample deposited at 1500-rpm, whereas the image on the right shows an LCO sample deposited at 6000-rpm. The 1500-rpm has a thickness of roughly 700-800 nanometers, whereas the 6000-rpm sample has a thickness of less than 500 nanometers.

Conclusion

The progress made this summer for the SULI internship was substantial. Using the previously discussed methods LCO of substantial purity was successfully synthesized and deposited onto SiO2 substrates. By varying formulation techniques and temperature profiles, I was able to raise the purity of the LCO to secondary perovskite phase ratio in my samples. Additionally, in changing the deposition profile by reducing the viscosity of the deposition material and increasing the spin rate of the substrate I produced a continuous thin film of material on the surface of the substrate with levels of thickness reaching below 500 nanometers, a vast improvement from the potentially discontinuous 10-micron thick layers being produced just a few months ago.

Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly-owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

1. Zhang, Y. P., Liu, H. F., Hu, H. L., Xie, R. S., Ma, G. H., Huo, J. C., & Wang, H. B. (2018). Orientation-dependent structural and photocatalytic properties of LaCoO3 epitaxial nano-thin films. *Royal Society open science*, *5*(2), 171376.
2. Rana, A. K., Jeong, M. H., Noh, Y. I., Park, H., Baik, J. M., & Choi, K. J. (2022). Phase-tuned MoS2 and its hybridization with perovskite oxide as bifunctional catalyst: a rationale for highly stable and efficient water splitting. *ACS Applied Materials & Interfaces*, *14*(16), 18248-18260.
3. Malkhandi, S., Yang, B., Manohar, A. K., Manivannan, A., Prakash, G. S., & Narayanan, S. R. (2012). Electrocatalytic properties of nanocrystalline calcium-doped lanthanum cobalt oxide for bifunctional oxygen electrodes. *The journal of physical chemistry letters*, *3*(8), 967-972.
4. Hwang, H. J., Moon, J., Awano, M., & Maeda, K. (2000). Sol–gel route to porous lanthanum cobaltite (LaCoO3) thin films. *Journal of the American Ceramic Society*, *83*(11), 2852-2854.
5. Hadjiev, V. G., Iliev, M. N., & Vergilov, I. V. (1988). The raman spectra of Co3O4. *Journal of Physics C: Solid State Physics*, *21*(7), L1999.

Appendix Program for plotting photocurrent maps correlated to Raman maps.

Created on Wed Jul 31 11:35:49 2024

@author: smcaver
"""
```
import matplotlib.pyplot as plt
import numpy as np
import sys
pc_file = '//snlca/home/smcaver/Desktop/Javier_Data_3/07-24-24_JF-II-231/Difficult_Data_Set/Map_250um_Pt-Cr_Collector_RT_50x_Objective_pc.txt'
r_file = '//snlca/home/smcaver/Desktop/Javier_Data_3/07-24-24_JF-II-231/Difficult_Data_Set/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Copy_raman.txt'
noise_ommision_threshold = 0.275 * (10**(-8))
x_coordinate = 1
y_coordinate = 1
target_integer_wave_number = 520
peak_prominence_detection_threshold = 170
mark_peaks = True
```

```
restrict_x = False
xlimits = [0,1750]
ylimits = [0,1000]
calibration_mode = True
end_time = 1889 # 1889
start_time = 165 # 165
# Extract Raman Data From txt File
r_lines = []
total_r_lines = 0
with open(r_file) as file:
    for file_line in file:
        r_lines.append(file_line)
        total_r_lines += 1
file.close()
x_positions = []
y_positions = []
wave_numbers = []
r_intensities = []
i = 0
for line in r_lines:
    if i == 0:
        i += 1
    else:
        x_positions.append(float(line.split()[0]))
        y_positions.append(float(line.split()[1]))
        wave_numbers.append(float(line.split()[2]))
        r_intensities.append(float(line.split()[3]))
# Determine the Unique X and Y Positions
unique_y_positions = []
unique_y_positions.append(y_positions[0])
latest_y = y_positions[0]
for y_position in y_positions:
    if y_position != latest_y:
        unique_y_positions.append(y_position)
        latest_y = y_position
unique_x_positions = []
unique_x_positions.append(x_positions[0])
latest_x = x_positions[0]
for x_position in x_positions:
    if x_position == unique_x_positions[0] and np.size(unique_x_positions) > 1:
        break
    if x_position != latest_x:
        unique_x_positions.append(x_position)
        latest_x = x_position
```

```
# Extract Photocurrent Data From txt File
pc_lines = []
total_pc_lines = 0
with open(pc_file) as file:
    for file_line in file:
        pc_lines.append(file_line)
        total_pc_lines += 1
file.close()
times = []
currents = []
previous = 1000
tail = True
for line in pc_lines:
    if len(line) >= 15:
        if tail == False:
            times.append(float(line.split()[1]))
            currents.append(float(line.split()[3]))
            if times[-1] >= end_time:
                break
            if line.split()[1].isnumeric() and tail == True and float(line.split()[1]) >= start_time:
            if float(line.split()[3]) > (previous*2):
                times.append(float(line.split()[1]))
                currents.append(float(line.split()[3]))
                tail = False
            previous = float(line.split()[3])
    # Plot Noisy Photocurrent Data to Determine Proper Noise Ommision Threshold
plt.plot(times, currents)
plt.title("Photocurrent Without Noise Ommision")
plt.xlabel("Time (s)")
plt.ylabel("Current (A)")
plt.show()
# Filter Out Noise
points_size = np.size(times)
currents_ave = np.mean(currents)
filtered_currents = []
i = 0
for current in currents:
    if current < noise_ommision_threshold and current >= 0:
        filtered_currents.append(current)
    else:
        filtered_currents.append(currents_ave)


# Plot Filtered Photocurrent Data
```

```
plt.plot(times, filtered_currents)
plt.title("Filtered Photocurrent")
plt.xlabel("Time (s)")
plt.ylabel("Current (A)")
plt.ylim(-max(filtered_currents), 2*max(filtered_currents))
plt.show()
# Extract Target Wave Number Indexes and Corresponding Intensity Values
wave_number_indexes = []
k = 0
wave_number_indexes = []
for wave_number in wave_numbers:
    if np.trunc(wave_number) == float(target_integer_wave_number):
        wave_number_indexes.append(k)
    k += 1
if np.size(wave_number_indexes) != (np.size(unique_x_positions) * np.size(unique_y_positions)):
    print("Target integer wave number of " + str(target_integer_wave_number) + " not found in txt file.
Either change the number slightly or check txt file for valid inputs.")
    print("Generally, valid input values are within 3 cm^-1 of the desired wave number value.")
    print("Suggested input values: " + str(target_integer_wave_number - 2) + ", " +
str(target_integer_wave_number - 1) + ", " + str(target_integer_wave_number + 1) + ", " +
str(target_integer_wave_number + 2))
    sys.exit()
target_intensities = []
for index in wave_number_indexes:
    target_intensities.append(r_intensities[index])
# Extract Photocurrent Values
pc_scans_size = points_size/np.size(unique_y_positions)
point_readings_size = (points_size/np.size(unique_y_positions))/np.size(unique_x_positions)
point_scan = []
local_mins = []
local_maxes = []
# Model 3 - Determining Point Scan Partiton Time Values
previous_current = filtered_currents[0]
partition_time_values = []
temp_array = []
append_temp = False
for (time, filtered_current) in zip(times, filtered_currents):
    if filtered_current <= (0.75 * previous_current):
        append_temp = True
    if append_temp == True:
        temp_array.append(time)
    if filtered_current >= (2 * previous_current) and append_temp == True:
        partition_time_values.append(temp_array[int(np.trunc(np.size(temp_array)/2))] * 10)
        temp_array = []
```

```
        append_temp = False
    previous_current = filtered_current
point_spacings = []
for i in range(1, np.size(partition_time_values)):
    point_spacings.append(partition_time_values[i] - partition_time_values[i-1])
average_point_spacing = np.mean(point_spacings)
for i in range(np.size(partition_time_values)-1):
    if partition_time_values[i+1] - partition_time_values[i] <= (0.6 * average_point_spacing):
        partition_time_values[i] = -1
k = 0
for i in range(np.size(partition_time_values)):
    i -= k
    if partition_time_values[i] <= 0:
        del(partition_time_values[i])
        k += 1
if np.size(partition_time_values) < (np.size(unique_x_positions) * np.size(unique_y_positions)):
    point_spacings = []
    for i in range(1, np.size(partition_time_values)):
        point_spacings.append(partition_time_values[i] - partition_time_values[i-1])
    for i in range(1, np.size(partition_time_values)):
        if (partition_time_values[i] - partition_time_values[i-1]) == max(point_spacings):
            partition_time_values.insert(partition_time_values.index(partition_time_values[i]),
(((partition_time_values[i] - partition_time_values[i-1])/2) + partition_time_values[i-1]))
            del(point_spacings[point_spacings.index(max(point_spacings))])
            if np.size(partition_time_values) == (np.size(unique_x_positions) * np.size(unique_y_positions)):
                break
i = start_time * 10 # pc points
j = 0 # partition index
for filtered_current in filtered_currents:
    if i <= partition_time_values[j]:
        point_scan.append(filtered_current)
    if i > partition_time_values[j]:
        local_maxes.append(max(point_scan))
        local_mins.append(min(point_scan))
        point_scan = [filtered_current]
        j += 1
    i += 1
    if j == np.size(partition_time_values) or i >= end_time * 10:
        break
dark_current = max(local_mins)
photocurrents = []
row = []
j = 0
i = 0
```

```
for local_max in local_maxes:
    if i == 0:
        photocurrents.append(local_max-dark_current)
        j += 1
    if i >= 1:
        row.append(local_max-dark_current)
        j += 1
    if j == np.size(unique_x_positions):
        if np.size(row) == 0:
            i += 1
            j = 0
        if np.size(row) != 0:
            photocurrents = np.vstack([photocurrents, row])
            row = []
            j = 0
        # Matricize Raman Data
r_matrix = []
row = []
for i in range(np.size(target_intensities)):
    if i < np.size(unique_x_positions):
        r_matrix.append(target_intensities[i])
    if i/np.size(unique_x_positions) - np.fix(i/np.size(unique_x_positions)) == 0 and i !=
np.size(unique_x_positions) and i != 0:
        r_matrix = np.vstack([r_matrix,row])
        row = []
    if i == (np.size(target_intensities) - 1):
        row.append(target_intensities[-1])
        r_matrix = np.vstack([r_matrix,row])
    if i >= np.size(unique_x_positions):
        row.append(target_intensities[i])
# Determine Raman Intensities and Local Peak Locations
if x_coordinate > np.size(unique_x_positions):
    print("Requested coordinate does not fit within the bounds of the scan.")
    print("Pick a x_coordinate value that is <= " + str(np.size(unique_x_positions)))
    sys.exit()
if y_coordinate > np.size(unique_y_positions):
    print("Requested coordinate does not fit within the bounds of the scan.")
    print("Pick a y_coordinate value that is <= " + str(np.size(unique_y_positions)))
    sys.exit()
raman_intensities = []
i = 0
raman_range = 0
for intensity in r_intensities:
```

```
    if y_positions[i] == unique_y_positions[y_coordinate-1] and x_positions[i] ==
unique_x_positions[x_coordinate-1]:
        raman_intensities.append(intensity)
        raman_range += 1
    i += 1
local_peaks = []
noise_intensity = np.mean(raman_intensities[0:100])
prior_prominent_intensity = 0
local_peak_found = False
for raman_intensity in  raman_intensities:
    if raman_intensity < prior_prominent_intensity and local_peak_found == False and
prior_prominent_intensity >= (peak_prominence_detection_threshold + noise_intensity):
        local_peaks.append(prior_prominent_intensity)
        local_peak_found = True
    if raman_intensity >= prior_prominent_intensity and local_peak_found == True:
        local_peak_found = False
    prior_prominent_intensity = raman_intensity
local_peak_locations = []
for local_peak in local_peaks:
    index = raman_intensities.index(local_peak)
    local_peak_locations.append(wave_numbers[0:raman_range][index])
# Plot Raman Intensity Map
plt.pcolormesh(r_matrix)
plt.colorbar()
plt.title("Raman Intensity Map at " + str(target_integer_wave_number) + " Inverse Centimeters")
plt.plot(x_coordinate - 0.5,y_coordinate - 0.5,"r.")
plt.show()
# Plot Raman Spectra at User Specified Point
if mark_peaks == True:
    for local_peak_location in local_peak_locations:
        if local_peak_location == local_peak_locations[0]:
            plt.plot(local_peak_location*np.ones(2),[0,(ylimits[1]-ylimits[0])*(0.6)],"green", label = "Peak
Location")
        else:
            plt.plot(local_peak_location*np.ones(2),[0,(ylimits[1]-ylimits[0])*(0.6)],"green")
plt.plot(wave_numbers[0:raman_range],raman_intensities,"blue")
plt.title("Raman Spectra at point (" + str(x_coordinate) + "," + str(y_coordinate) + ")")
plt.xlabel("Raman Shift (cm^-1)")
plt.ylabel("Intensity")
plt.legend(loc = "upper left")
if restrict_x == True:
    plt.xlim(xlimits[0],xlimits[1])
plt.ylim(ylimits[0],ylimits[1])
plt.show()
```

```
print("Peak Locations Detected at " + str(local_peak_locations))
# Plot Photocurrent Intensity Map
if calibration_mode == False:
    plt.pcolormesh(photocurrents)
    plt.colorbar()
    plt.title("Photocurrent Intensity Map (A)")
    plt.show()
# Notes:
# Individual signals with too low of an intensity will have noise that will be near that of the max signal stength or higher, not to exceed user specified noise ommission threshold
# This causes the first column to be noisy and display highly amplified intensity compared to the actual photocurrent value
if calibration_mode == True:
    for i in range(np.size(unique_y_positions)):
        plt.plot(times, filtered_currents, "blue")
        for partition_time_value in partition_time_values:
            plt.plot([partition_time_value/10, partition_time_value/10], [0, max(filtered_currents)], "red")
        plt.xlim((pc_scans_size/10)*i, (pc_scans_size/10)*(i+1))
        plt.title("Graph " + str(i+1))
        plt.show()
# pc scan size method sometimes comes up short during calibration mode plotting
```

LaCoO3 Synthesis

Spencer Caverly

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

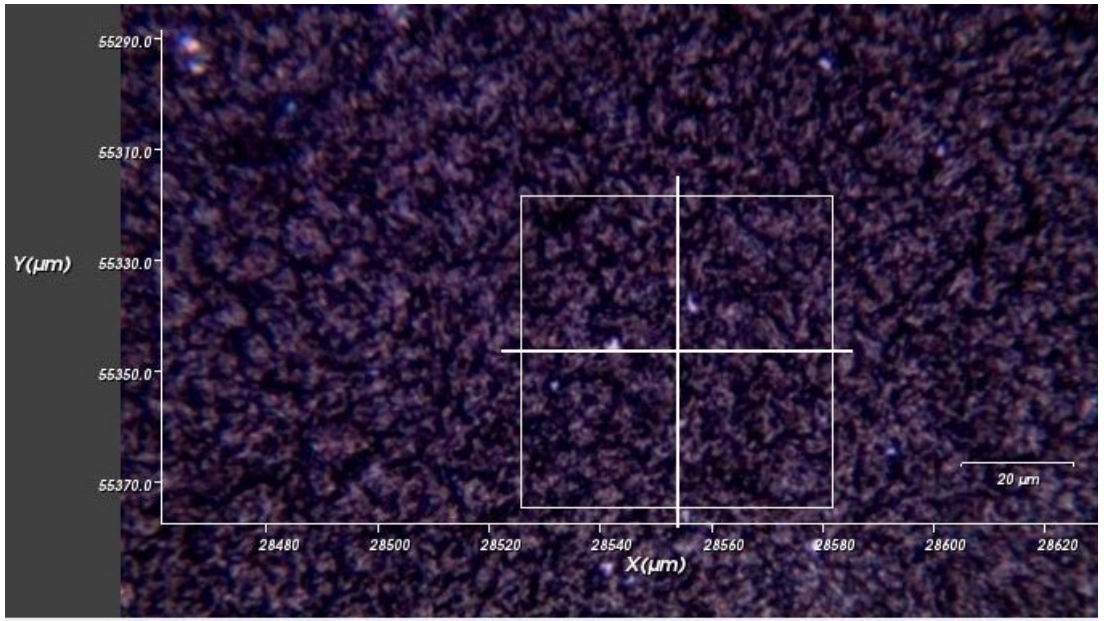
LCO - Motivation

- The goal of this project is to create thin film LCO layers for use as a p-type photovoltaic for a water-splitting Hydrogen Evolution Reaction(HER).
- There is a lot of interest in the utilization of LCO as a photocatalytic because compared to other commonly used materials, they possess a narrower band gap, wider λ range of light absorption, high sunlight utilization rate, low charge recombination rate, and high quantum efficiency
- LCO production via Sol-Gel method provides a potentially cost efficient method for the production of LCO thin film materials

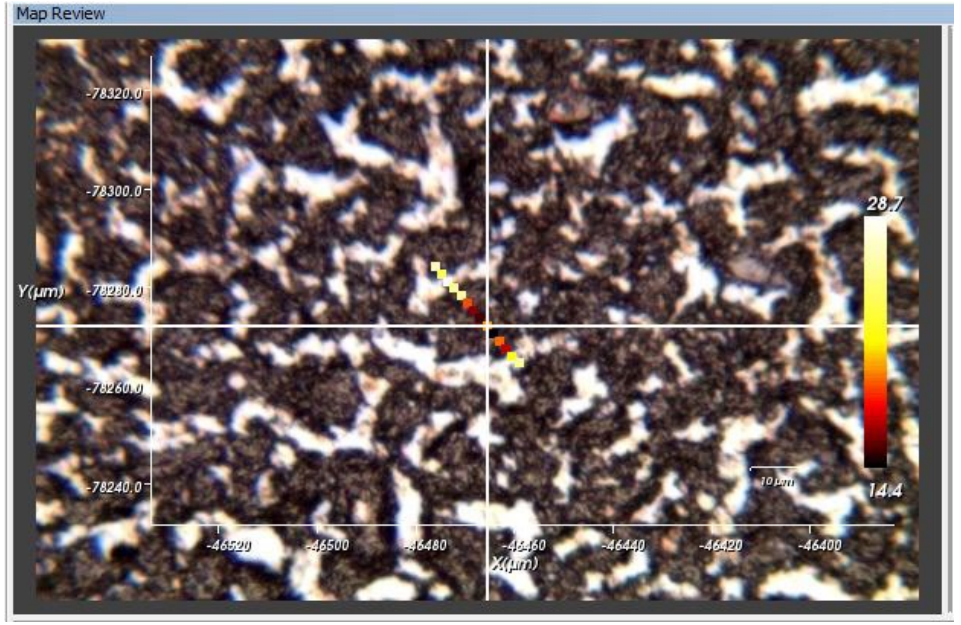
LCO – Methodology and Controls

- This summer I have focused on formulating LCO using Lanthanum Nitrate and Cobalt Nitrate precursors in aqueous solution
- The above solution is allowed to evaporate over an amount of time before being spin deposited onto a substrate and annealed in a tube furnace
- The main factors I controlled for in my synthesis processes were the anneal temperature profile, deposition viscosity, and spin rate.

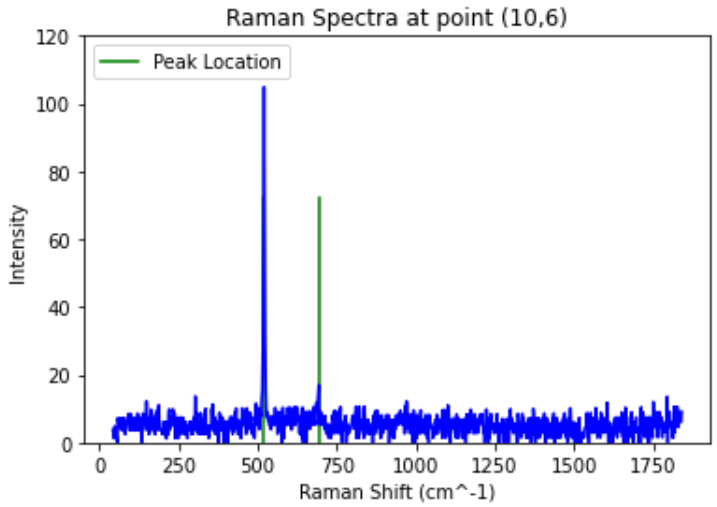
LCO – Temperature Profile

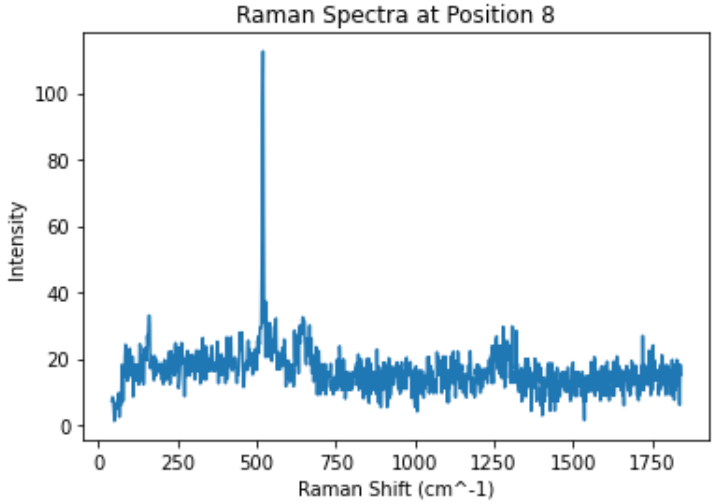


Annealed at 650 °C

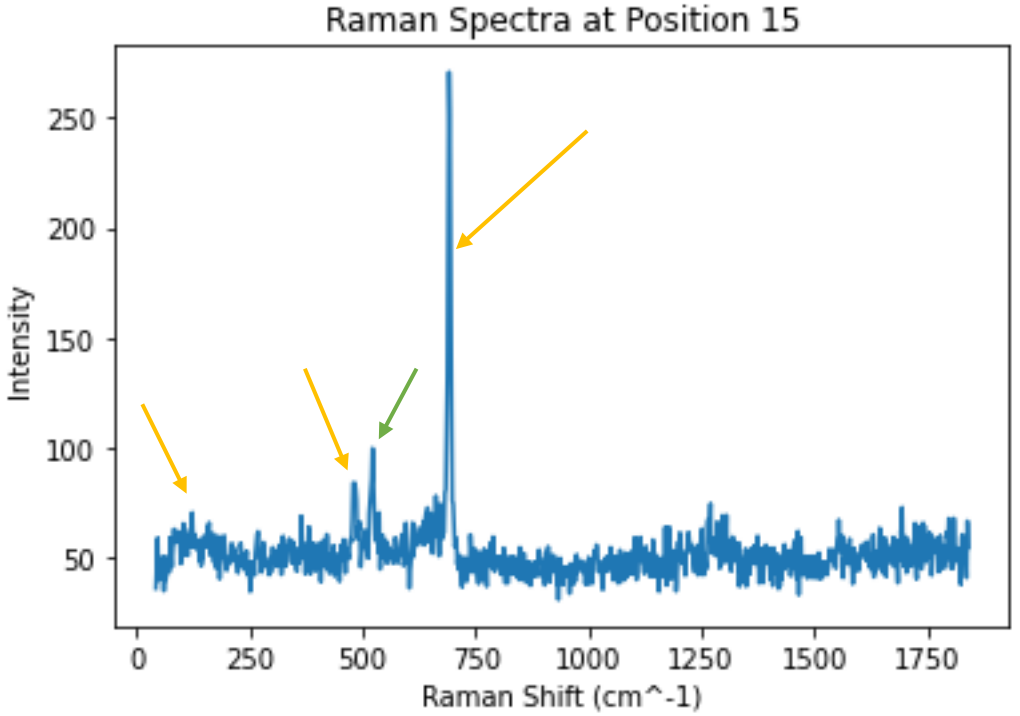


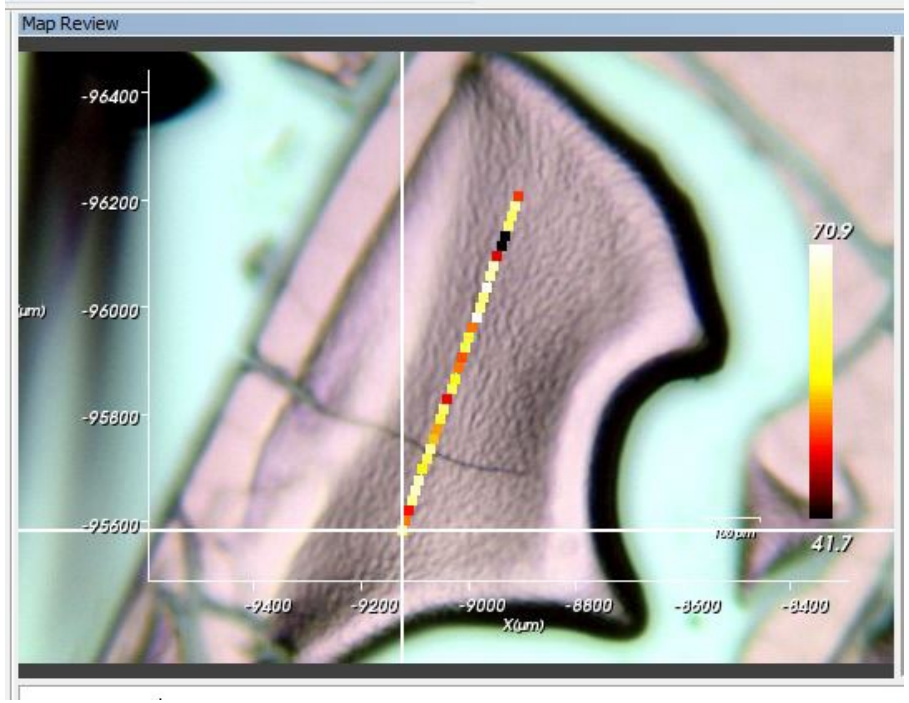
Annealed at 1000 °C





LCO – Formation of LCO “Platelets”

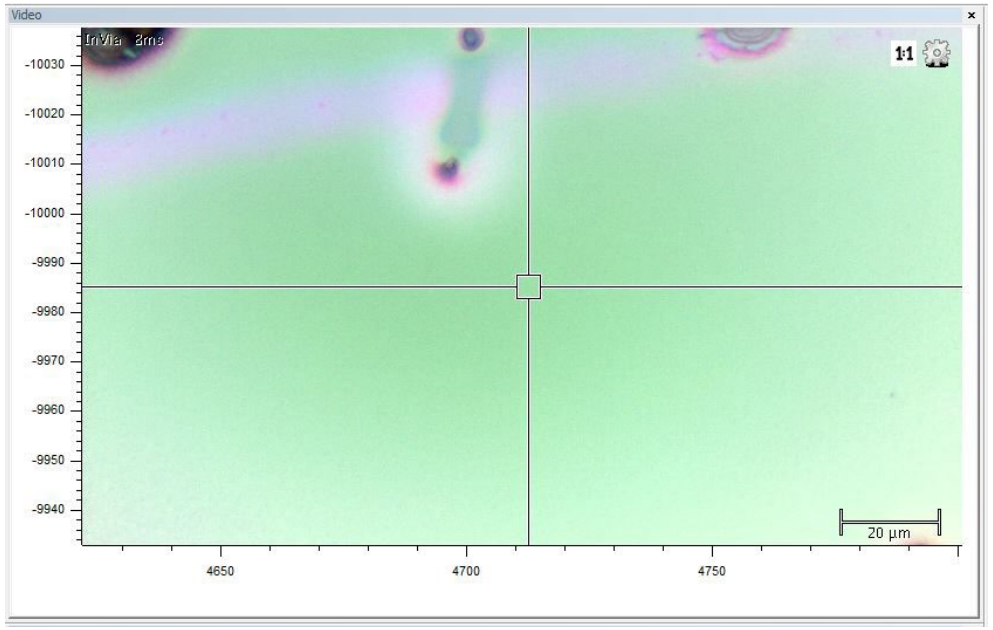




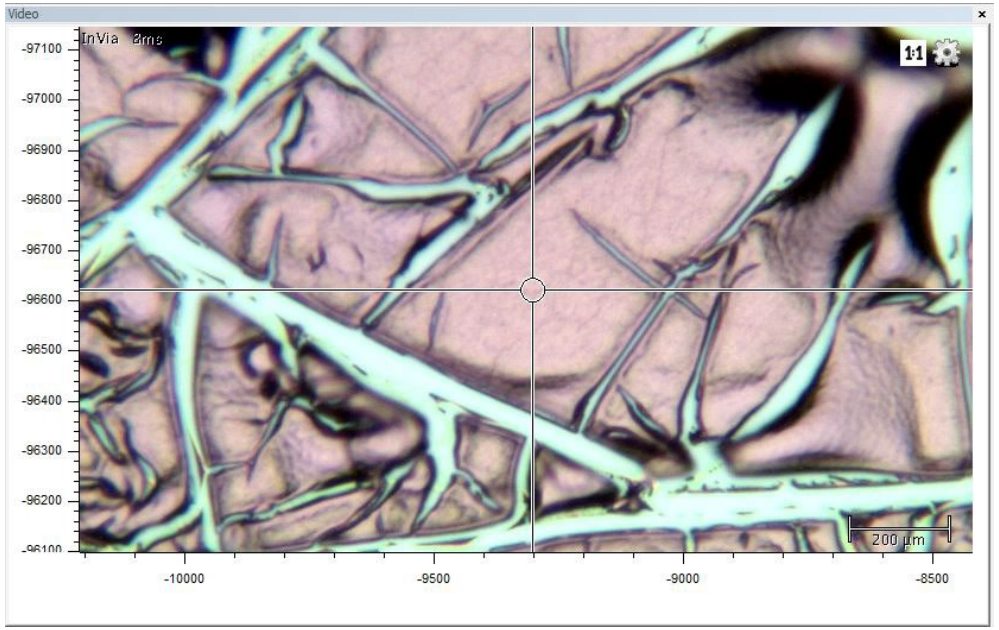
By strictly adhering to a stoichiometric balance in the precursor ratio and by slowly ramping the temperature profile to 900 °C I was able to collect Raman scans that displayed characteristic Raman peaks.

- Deposited LCO breaks apart and forms platelets at higher annealing temperatures
- Additionally, the thickness of each platelet is variant, ranging between 1-10 microns across a given platelet

LCO – Controlling for Deposition Viscosity



Viscosity – Watery with semi-opaque(red) coloring
Thickness – 500 Nanometers (Continuous)



Viscosity – Dark red substance with viscosity similar to a thick syrup
Thickness – 1-10 Microns (Variant across platelet)

Photocurrent Mapping Software

Spencer Caverly

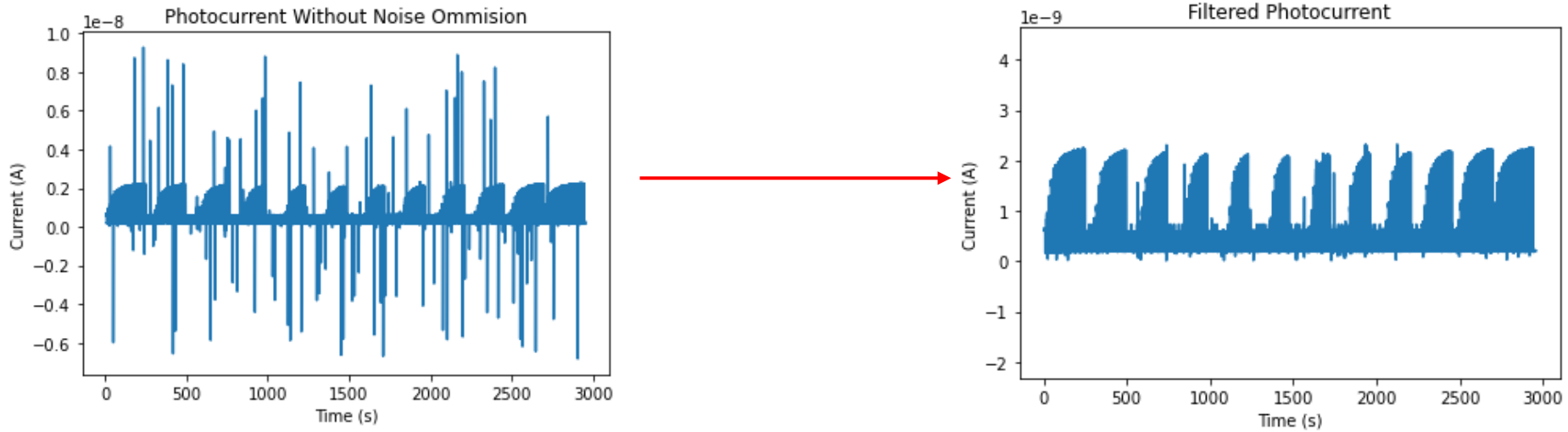
Raman/Photocurrent Mapping Software

- Before making this software, chronoamperometry data would be plotted and photocurrent signals would be visually recorded at each point in a line scan
- Because of the inefficiency of this method, pc data would be limited to low-resolution line scans spanning no more than a dozen data collection points across a sample

PC Noise Omission

- Software allows the user to easily define a noise omission threshold that detects and eliminates noise in the data

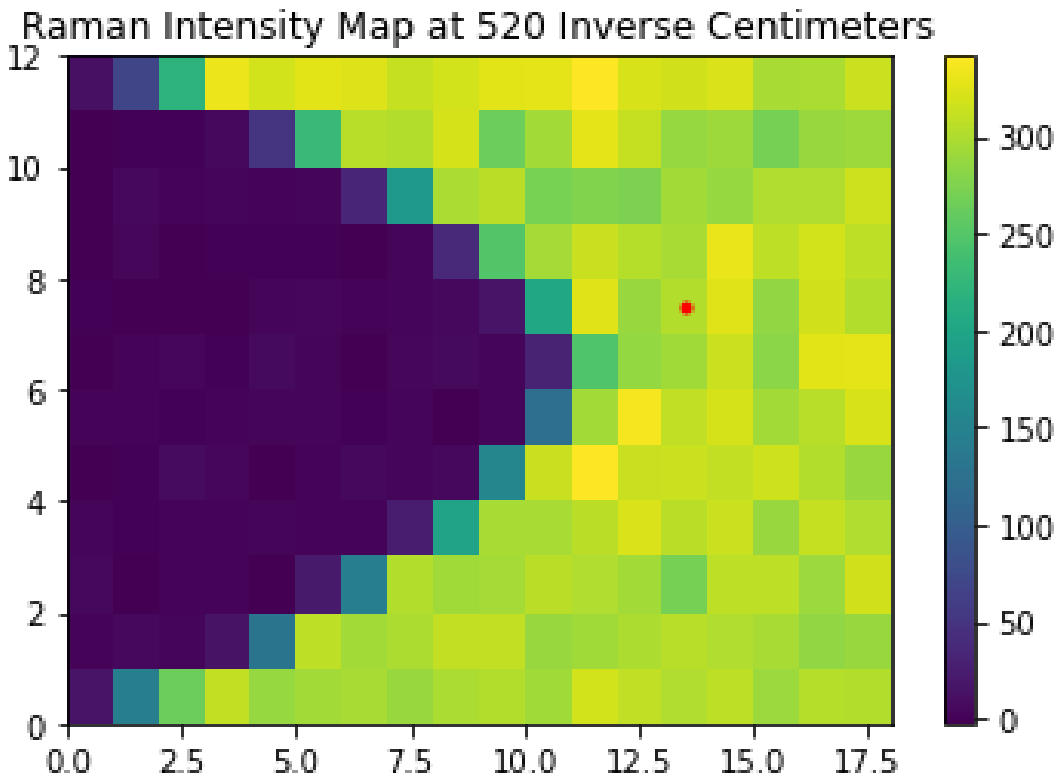
```
pc_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer.DTA.txt'
r_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer_Copy.txt'
noise_ommision_threshold = 0.275 * (10**(-8))
x_coordinate = 15
y_coordinate = 7
target_integer_wave_number = 520
peak_prominence_detection_threshold = 170
mark_peaks = True
restrict_x = False
xlimits = [0,1750]
ylimits = [0,1000]
calibration_mode = True
end_time = 2950
start_time = 5
```



Raman Mapping

- Software produces Raman map for the intensity of the user defined wave number

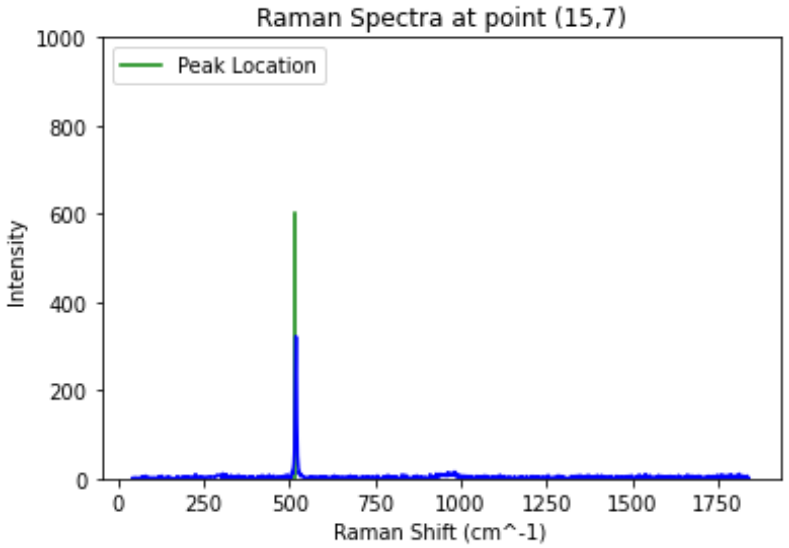
```
pc_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer.DTA.txt'
r_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer_Copy.txt'
noise_ommision_threshold = 0.275 * (10**(-8))
x_coordinate = 15
y_coordinate = 7
target_integer_wave_number = 520
peak_prominence_detection_threshold = 170
mark_peaks = True
restrict_x = False
xlimits = [0,1750]
ylimits = [0,1000]
calibration_mode = True
end_time = 2950
start_time = 5
```

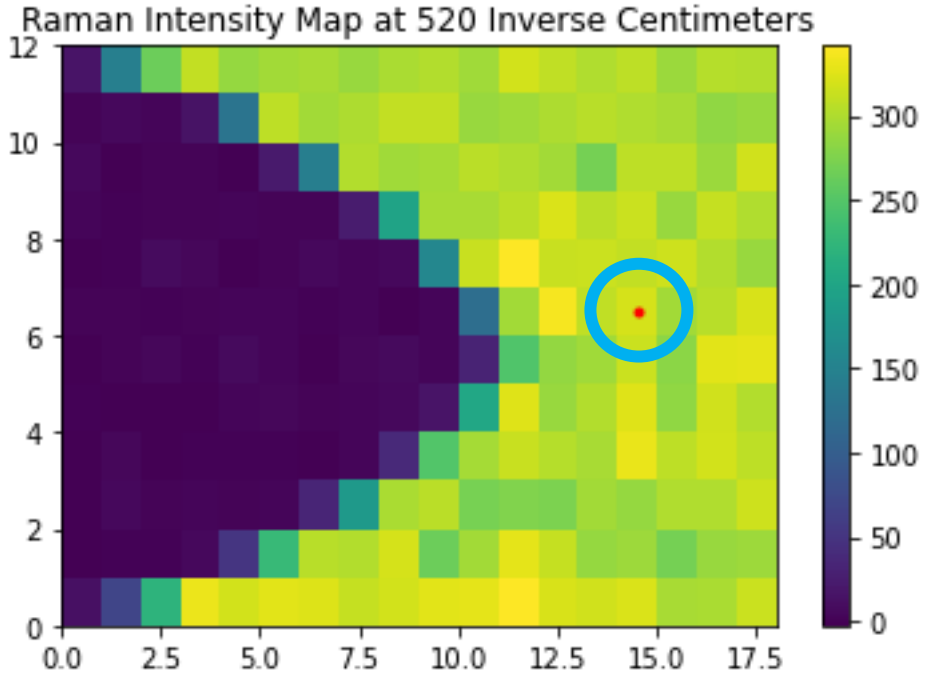


Raman Spectra Analysis

- By indicating a cartesian coordinate on the Raman map the user can inspect the Raman profiles of individual points on the sample

```
pc_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer.DTA.txt'
r_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer_Copy.txt'
oise_ommision_threshold = 0.275 * (10**(-8))
x_coordinate = 15
y_coordinate = 7
arget_integer_wave_number = 520
peak_prominence_detection_threshold = 170
mark_peaks = True
restrict_x = False
xlimits = [0,1750]
ylimits = [0,1000]
calibration_mode = True
end_time = 2950
start_time = 5
```

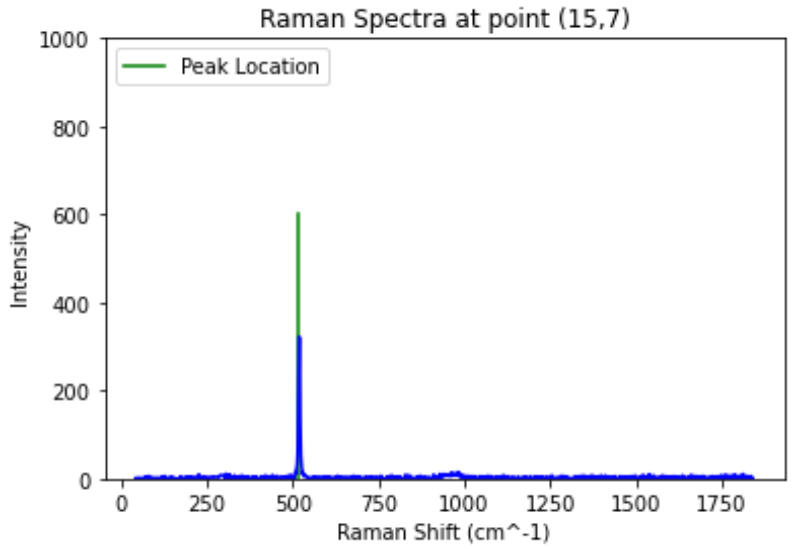




Raman Spectra Analysis

- In addition to displaying the Raman Spectrum, the software will use a user defined threshold value to identify and mark prominent peak locations

```
pc_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer.DTA.txt'
r_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer_Copy.txt'
noise_ommision_threshold = 0.275 * (10**(-8))
x_coordinate = 15
y_coordinate = 7
target_integer_wave_number = 520
peak_prominence_detection_threshold = 170
mark_peaks = True
restrict_x = False
xlimits = [0,1750]
ylimits = [0,1000]
calibration_mode = True
end_time = 2950
start_time = 5
```

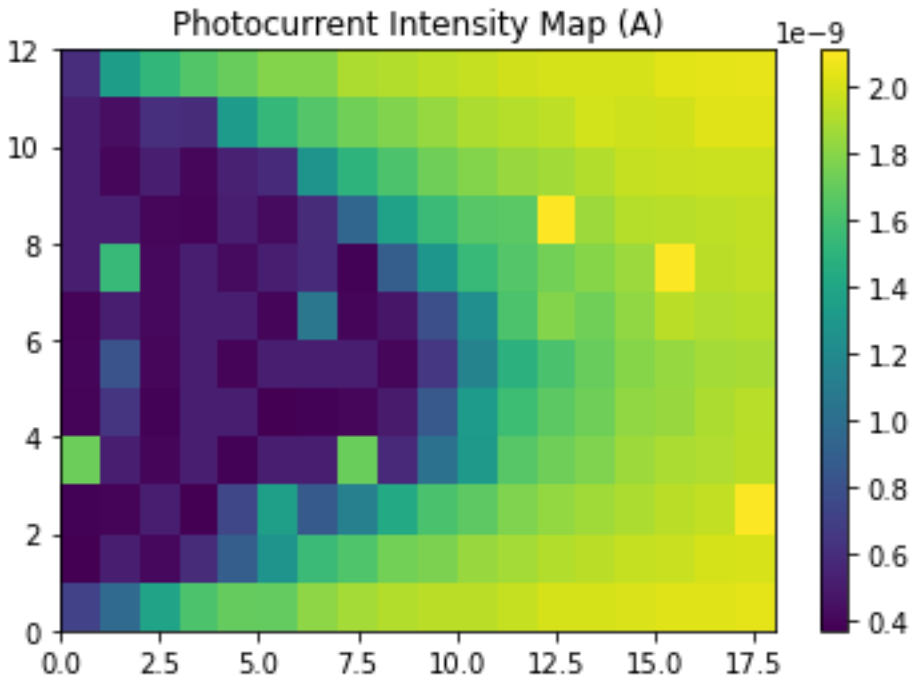



```
In [82]: runfile('//snlca/home/smcaver/Desktop/Photocurrent_Mapping_Irregular_Light_Modulation.py', wdir='//snlca/home/smcaver/
Desktop')
Peak Locations Detected at [518.207031]
```

Photocurrent Mapping

- Software superimposes Photocurrent onto Raman Map to create a Photocurrent Map of the analysis area

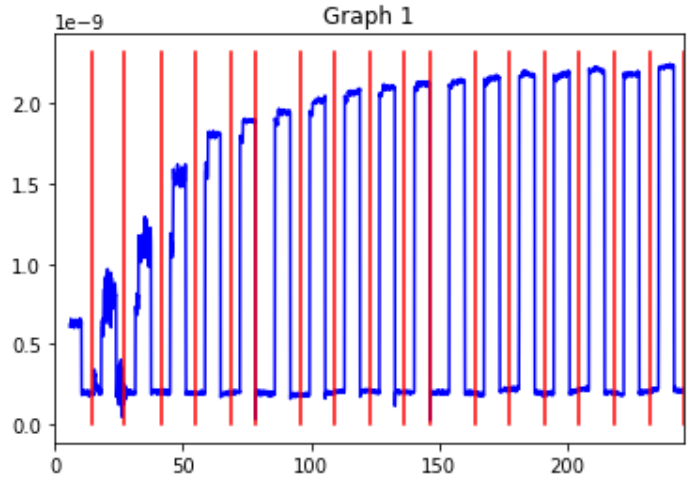
```
pc_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer.DTA.txt'
r_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer_Copy.txt'
noise_ommision_threshold = 0.275 * (10**(-8))
x_coordinate = 15
y_coordinate = 7
target_integer_wave_number = 520
peak_prominence_detection_threshold = 170
mark_peaks = True
restrict_x = False
xlimits = [0,1750]
ylimits = [0,1000]
calibration_mode = True
end_time = 2950
start_time = 5
```

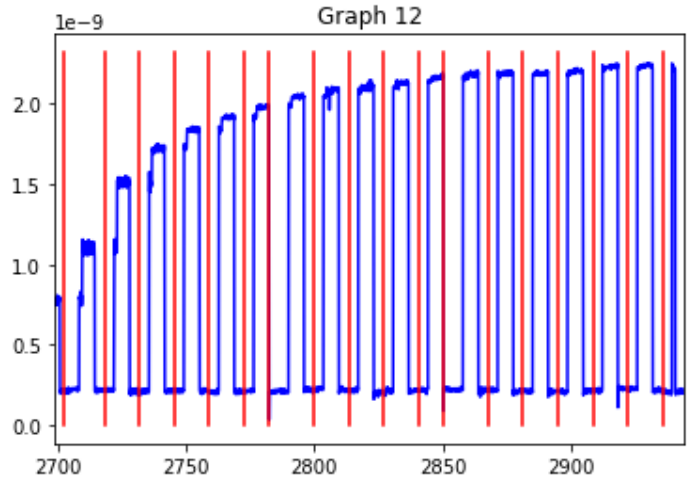


Photocurrent Mapping

- In order to correct for the offset in start and end times between the Gamry Chronoamperometry data and the Renishaw Raman data, a calibration mode can be enabled to ensure the two data sets are compatible by trimming the data using user defined parameters

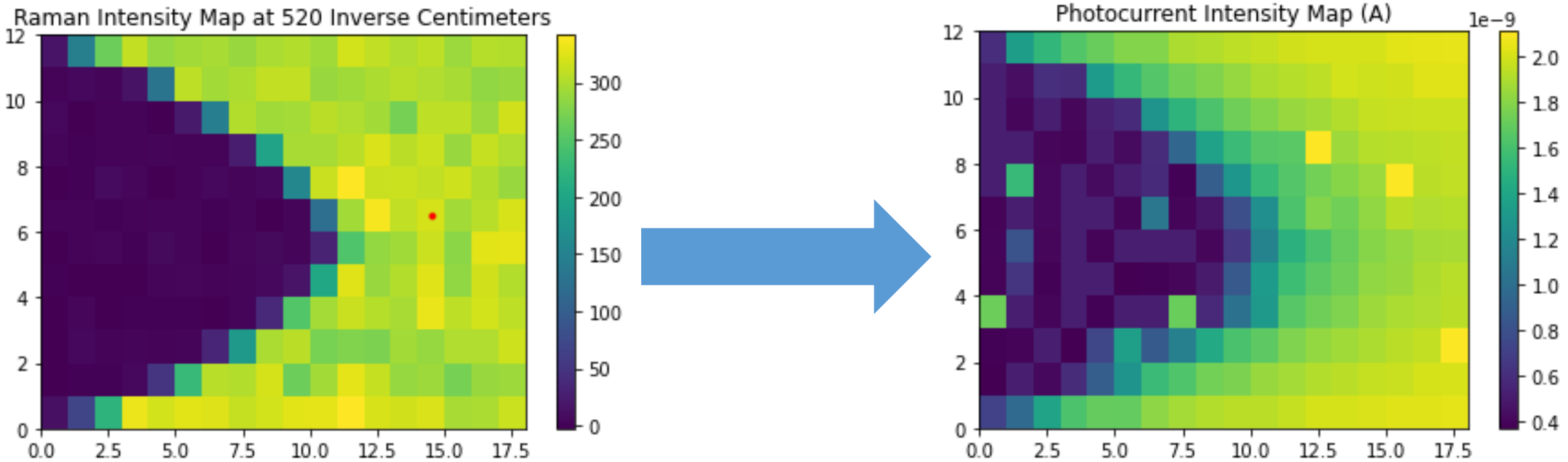
```
pc_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer.DTA.txt'
r_file = '//snlca/home/smcaver/Desktop/Map_250um_Pt-Cr_Collector_RT_50x_Objective_Finer_Copy.txt'
noise_ommision_threshold = 0.275 * (10**(-8))
x_coordinate = 15
y_coordinate = 7
target_integer_wave_number = 520
peak_prominence_detection_threshold = 170
mark_peaks = True
restrict_x = False
xlimits = [0,1750]
ylimits = [0,1000]
calibration_mode = True
end_time = 2950
start_time = 5
```





Raman/Photocurrent Mapping Software

- The development of this software will allow researchers to produce high resolution photocurrent maps of their samples and correlate their superimposed photocurrent data with wave number adjustable Raman maps



Raman/Photocurrent Mapping Software