In [ ]:

```
from IPython.display import Image
img = Image('https://vip.arizona.edu/images/logoviplab.png')
img
```

Out[ ]:

BE 585

2/11/2022

Danielle Tadych

In this exercise:

1. Compute Image pixel size
2. Compute the area of an object(s)

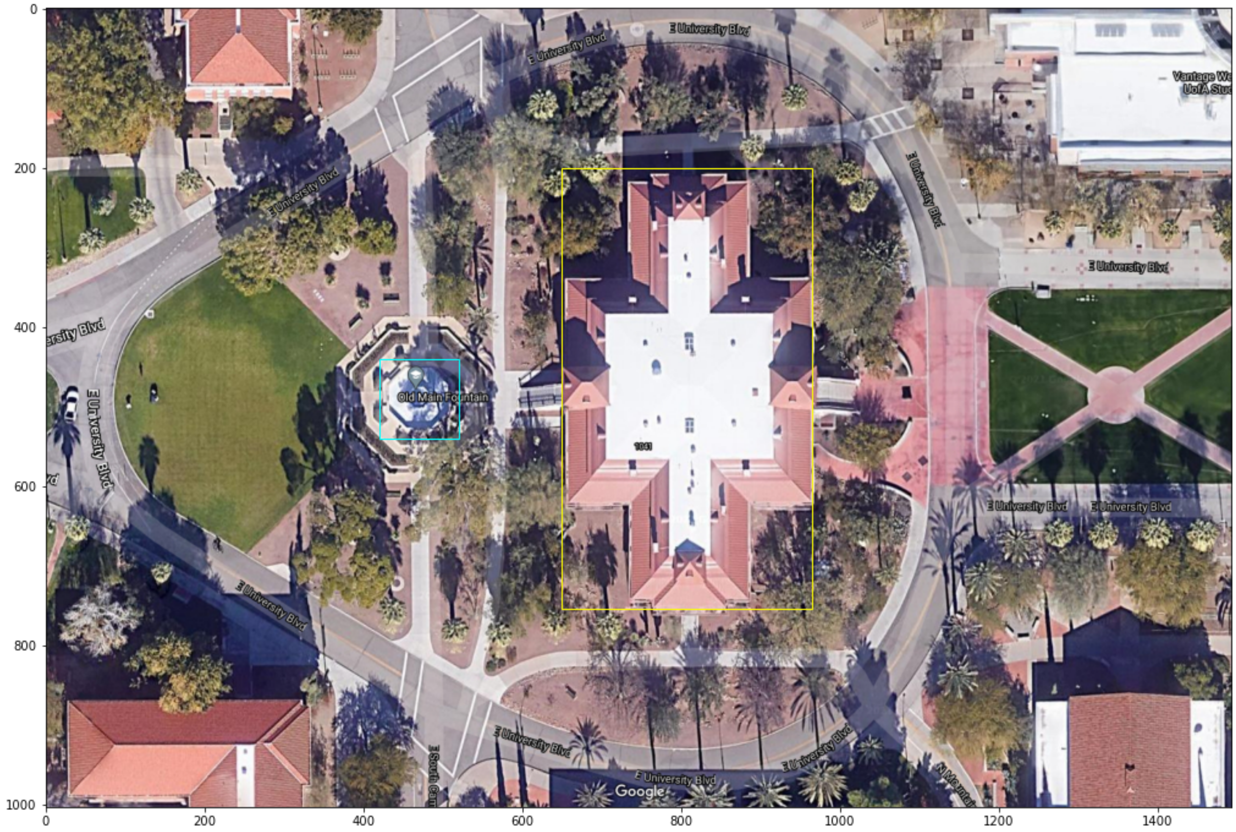
In [1]:

```
# load library modules
import os   # Import the Operating System Library. This is needed to deal with
import math # Import the Math Library to perform some basic math.
import matplotlib.pyplot as plt # Load the graphing/plotting library
import matplotlib.image as mpimg
import matplotlib.patches as patches
import numpy as np
from PIL import Image
```

Pixel size Calculator

In [31]:

```
#Display Old Main and outlne the building with a box

plt.figure(figsize = (20,12))  # Define canvas size to have a lrge figure

# Load the image into an 2D Array/MAtrix
im = Image.open('./Images/UA_Old_Main.jpg')

#Show the image
plt.imshow(im)

#Create a plot for the outline BOX
ax = plt.gca()

#Define the box size by Upper Left Corner X,Y and Extent X,Y
rect_OldMain = patches.Rectangle((650,200),
                 315,
                 555,
                 linewidth=1,
                 edgecolor='yellow',
                 fill = False)

ax.add_patch(rect_OldMain)


#Define another random box. USe this code to define a box aroudn the Fountain
rect_Fountain = patches.Rectangle((420,440),
                 100,
                 100,
                 linewidth=1,
                 edgecolor='cyan',
                 fill = False)
ax.add_patch(rect_Fountain)

plt.show()
```



In [17]:

```
# Use the shape call to find the size of the image
#It will display Numbe of ROWS, PIXELS, Bands
print(np.shape(im))
```

```
(1004, 1493, 3)
```

In [23]:

```
#Declare and define the variables to be used in this exercise.

# From the Image get the following values
# Old Main number of  pixels in the X (columns) direction
OldMain_xpixels= 315 #Change this to the actual number

# Old Main number of rows in the Y (columns) direction
OldMain_ypixels= 555    #Change this to the actual number
```

In [32]:
```
# Real (reference) values
# Calculated with google maps measurement tool
#distance in X direction (in meters)
xdistance= 40    # Get the actual Distance
ydistance= 68    # Get the actual Distance
```

In [33]:
```
# Compute pixel size in both directions (meters)

pixelsize_x= xdistance / OldMain_xpixels
pixelsize_y= ydistance / OldMain_ypixels
```

In [35]:
```
# display input values and  results
print("Measurements of reference:")
print("OldMain in pixels: x =",OldMain_xpixels, ", y =",OldMain_ypixels)
print("OldMain in meters: x =",xdistance,", y =",ydistance)
print("Computed pixel size (in meters):")
print(" size x =",pixelsize_x,", y =",pixelsize_y)
```

```
Measurements of reference:
OldMain in pixels: x = 315 , y = 555
OldMain in meters: x = 40 , y = 68
Computed pixel size (in meters):
 size x = 0.12698412698412698 , y = 0.12252252252252252
```

To Do : Homework

Use the methods above to:

1. Estimate the size of the Fountain (Cyan rectangle) in front of the old main without direct measurement.
2. i.e. by using the pixel size from above and the size of the fountain in the image
3. i.e. confirm with Googel Map measurment tool
4. Show all your steps
5. What would happen if you have selectd different locations/buildings/sizes
6. What does all of this mean?

In [38]:
```
# Estimating the fountain size

f_x = 100 * pixelsize_x
f_y = 100 * pixelsize_y

# Since we calculated above the pixel size to meter ratio, we can convert
area = f_x * f_y
print("Fountain Area (m squared) = ", area)
```

```
Fountain Area (m squared) =  155.58415558415555
```

If I had selected different buildings or location sizes, the conversion of pixel size to meters would have been the same so this would all mean that I could calculate the actual pixel size of any object in that picture.

In [39]:
```
print("End of program.")
```

```
End of program.
```

--- Exercise 2 ---

In this exercise:

1. read data from an Excel file
2. user defined functions
3. Resampling
   A. Loop pixels by row and columns
   B. Get subsets
   C. Get average of subsets
   D. Create a computed image

In [42]:

```
# load library modules
import os # Load the Operating System library to access & manipulate files and
import xlrd # Load the library that reads Exel sheets
import numpy as np # Load the famous NUMPY library the most useful library in
import matplotlib.pyplot as plt # Load the graphing/plotting library
import matplotlib.image as mpimg
import viplab_lib as vip # Load our very own home brewed library it has few add
```

Resampling Function so we can reuse it

In [63]:

```
# Here we will create a function that resamples data from any resolution to any
# This function resamples an image to a user defined space

# In Python we declare then define the function first, then we can refer to it
# This could also be added to a library

# Just use this code try to understyand it will become clearer as we proceed wi

def resample_data(data,n): # When we call the function we pass it the data and
    #get size of input band
    nrowsIN,ncolsIN=data.shape
    #calculate output band size
    nrows=nrowsIN // n
    ncols=ncolsIN // n

    #create empty band
    datares=np.zeros((nrows,ncols))
    for i in range(0,nrows):
        for j in range(0,ncols):

            #calculate row at input band
            rowIN=i*n
            #check for out of boundary row
            if(rowIN<0):
                rowIN=0
            elif (rowIN>nrowsIN-1):
                rowIN=nrowsIN-1

            #calculate col at input band
            colIN=j*n
            #check for out of boundary column
            if(colIN<0):
                colIN=0
            elif (colIN>ncolsIN-1):
                colIN=nrowsIN-1


            #subset and get average
            avgvalue=np.mean(data[rowIN:rowIN+n,colIN:colIN+n])

            #get the integer value of the average
            datares[i,j]= int(avgvalue)
    return datares
```

Open Old Main Image (RGB)

In [64]:

```
#Load the image
img = mpimg.imread('Images/UA_Old_Main.jpg') # Load the whole image into a 3D

# Speate the layer (Using Python slicing on the third dimension)
DataRed = img[:, :, 0]
DataGreen = img[:, :, 1]
DataBlue = img[:, :, 2]
```

Perform the Resampling and display results

In [65]:

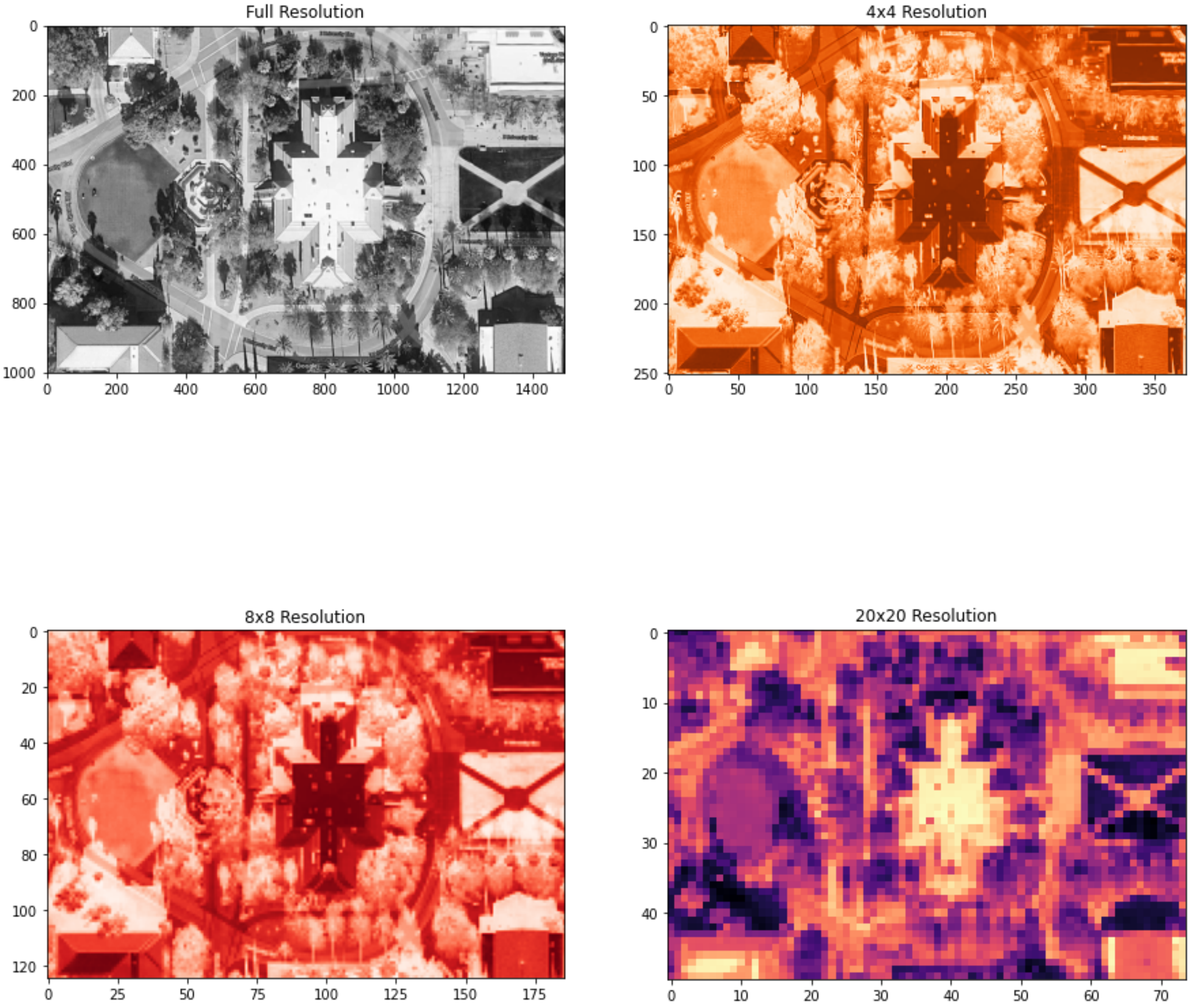
```
# Resample the data
# Resample the original image
# To 4x4

print("Resampling 4x4")
DataRed_Res4=resample_data(DataRed,4)

# Resample 8x8
print("Resampling 8x8")
DataRed_Res8=resample_data(DataRed,8)

# Resample 20x20
print("Resampling 20x20")
DataRed_Res20=resample_data(DataRed,20)
```

```
Resampling 4x4
Resampling 8x8
Resampling 20x20
```

In [66]:

```
# Let's look at the data we've just read, then resame it to different sizes/res

f, Plot_Arr = plt.subplots(2,2,figsize=(15, 15))  # We can create a canvas of

#Pay attention to the way we pass the plot to the grid on the Canvas

Plot_Arr[0,0].imshow(DataRed,cmap='gray')     # Original data full resolution
Plot_Arr[0,0].title.set_text('Full Resolution')  # Assign title to plot


Plot_Arr[0,1].imshow(DataRed_Res4,cmap='Oranges') # Resmapled 4x4 resolution
Plot_Arr[0,1].title.set_text('4x4 Resolution')

Plot_Arr[1,0].imshow(DataRed_Res8,cmap='Reds') # Resmapled 8x8 resolution
Plot_Arr[1,0].title.set_text('8x8 Resolution')
```

```
Plot_Arr[1,1].imshow(DataRed_Res20,cmap='magma') # Resmapled 20x20 resolution
Plot_Arr[1,1].title.set_text('20x20 Resolution')

# Try these color maps [cmaps]
# 'viridis','plasma','inferno','magma', 'cividis',
# 'Greys', 'Purples', 'Blues', 'Greens', 'Oranges', 'Reds','YlOrBr', 'YlOrRd',
# 'OrRd','PuRd','RdPu','BuPu','GnBu', 'PuBu', 'YlGnBu', 'PuBuGn', 'BuGn', 'YlGr
```



In [67]:

```
# Use the shape call to find the size of the image
#It will display Numbe of ROWS, PIXELS, Bands
print(img.shape)
```

```
(1004, 1493, 3)
```

To Do : Homework

Use the methods above to:

1. Modify the code so it resamples all the bands
2. Combine them into a single true color image and display

Resampling Green and Blue (rest of the bands)

```
In [68]: # Resample the data
         # Resample the original image
         # To 4x4

         print("Resampling 4x4")
         DataGreen_Res4=resample_data(DataGreen,4)
         DataBlue_Res4=resample_data(DataBlue,4)

         # Resample 8x8
         print("Resampling 8x8")
         DataGreen_Res8=resample_data(DataGreen,8)
         DataBlue_Res8=resample_data(DataBlue,8)

         # Resample 20x20
         print("Resampling 20x20")
         DataGreen_Res20=resample_data(DataGreen,20)
         DataBlue_Res20=resample_data(DataBlue,20)
```

```
Resampling 4x4
Resampling 8x8
Resampling 20x20
```

```
In [70]: np.shape(DataBlue_Res4)
```

```
Out[70]: (251, 373)
```

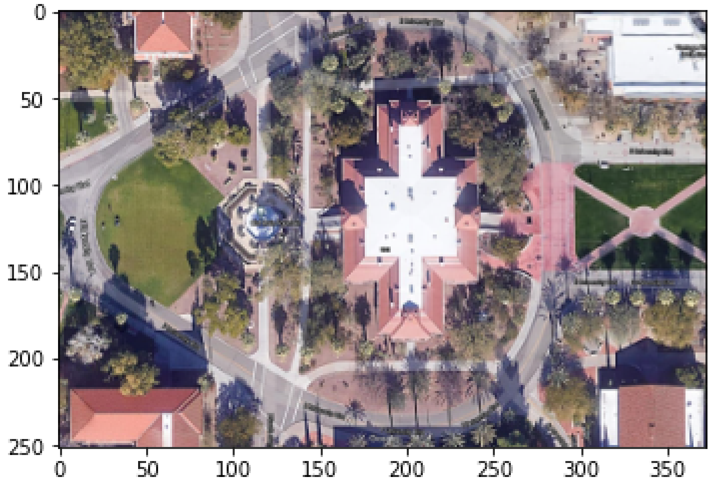
Combining the resampled

```
In [71]: # Needs to be a numpy array to use vip function below
         type(DataGreen_Res4)
```

```
Out[71]: numpy.ndarray
```

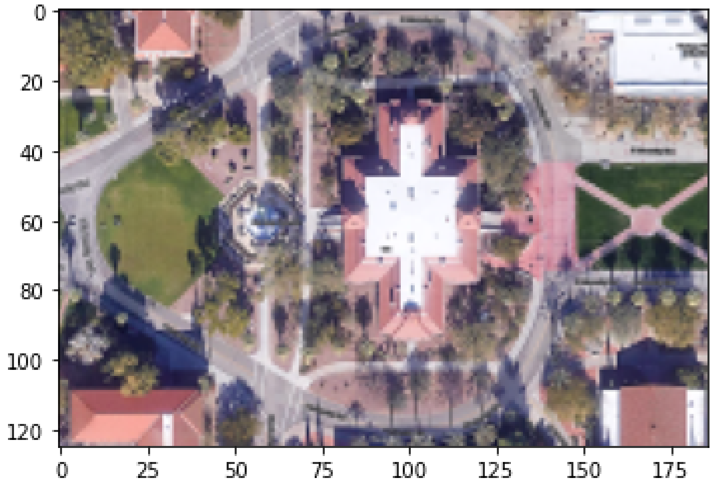
```
In [74]: # Combine all bands, the Red, Green and Blue data into an RGB model for display
         print("4 res image")
         Res4_Image=vip.Image_getRGB(DataRed_Res4,DataGreen_Res4,DataBlue_Res4,1400)
         # Display RGB Image
         plt.figure()
         plt.imshow(Res4_Image)
```

```
4 res image
done
```



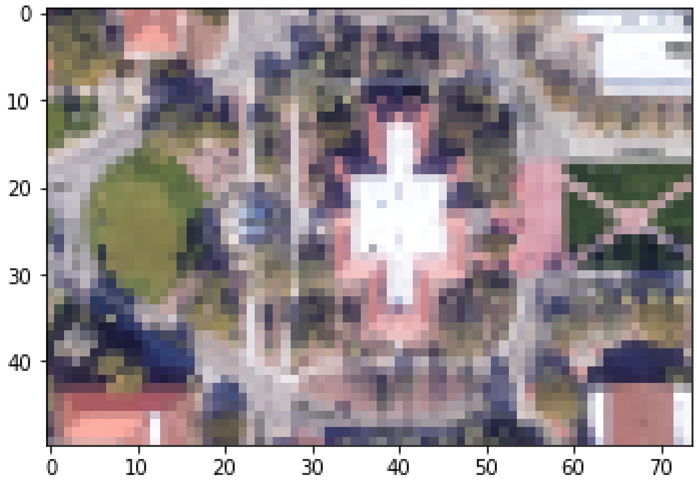
```
In [76]: print("8 res image")
         Res8_Image=vip.Image_getRGB(DataRed_Res8,DataGreen_Res8,DataBlue_Res8,1400)
         # Display RGB Image
         plt.figure()
         plt.imshow(Res8_Image)
```

```
         8 res image
Out[76]: <matplotlib.image.AxesImage at 0x7ffab4f4d790>
```



```
In [77]: print("20 res image")
         Res20_Image=vip.Image_getRGB(DataRed_Res20,DataGreen_Res20,DataBlue_Res20,1400)
         # Display RGB Image
         plt.figure()
         plt.imshow(Res20_Image)
```

```
         20 res image
Out[77]: <matplotlib.image.AxesImage at 0x7ffab2b5b940>
```



```
In [78]: print("End of program.")
```

```
         End of program.
```

```
In [ ]:
```