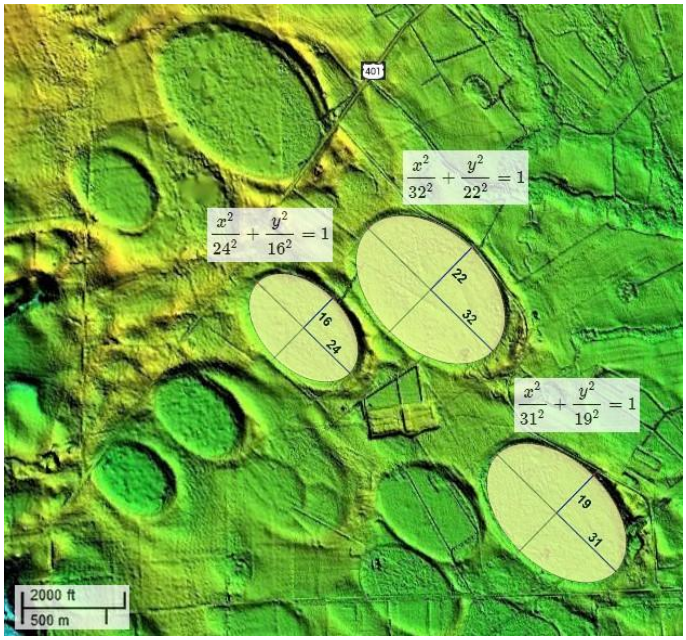
Fitting ellipses to Carolina Bays

Antonio Zamora, July 5, 2022

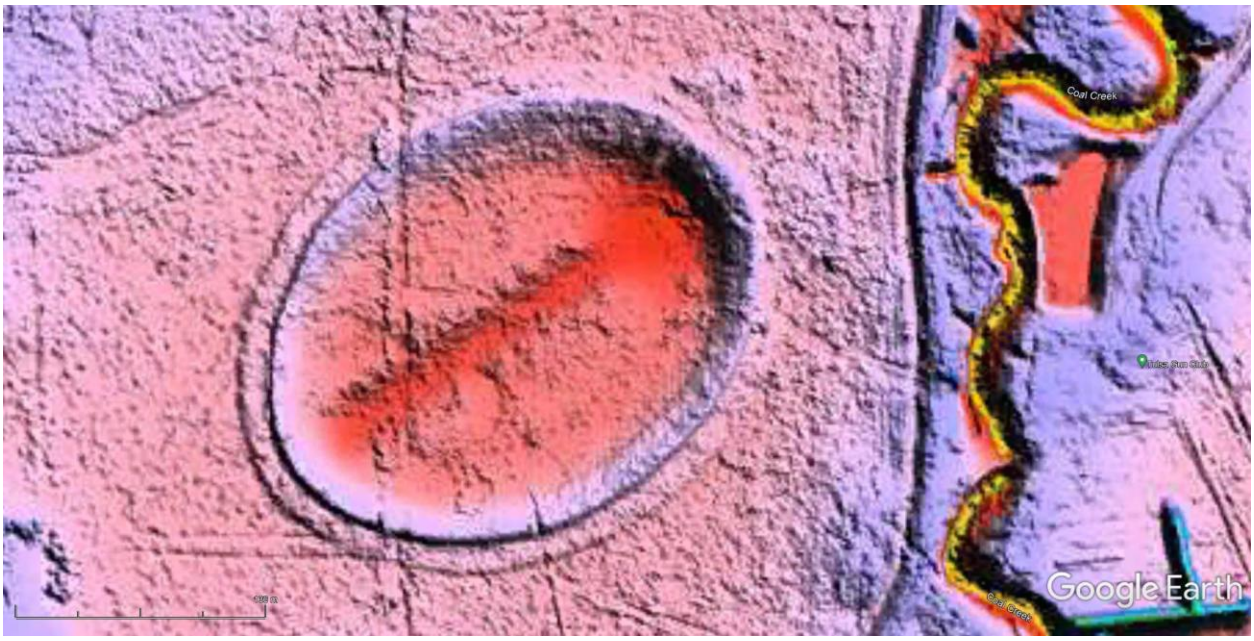
The Carolina Bays are shallow elliptical depressions on unconsolidated soil that originated as penetration funnels from secondary impacts of glacier ice boulders ejected by an extraterrestrial impact on the Laurentide Ice Sheet (Zamora, 2017). Since the Carolina Bays are conic sections, it is often necessary to fit them with ellipses.



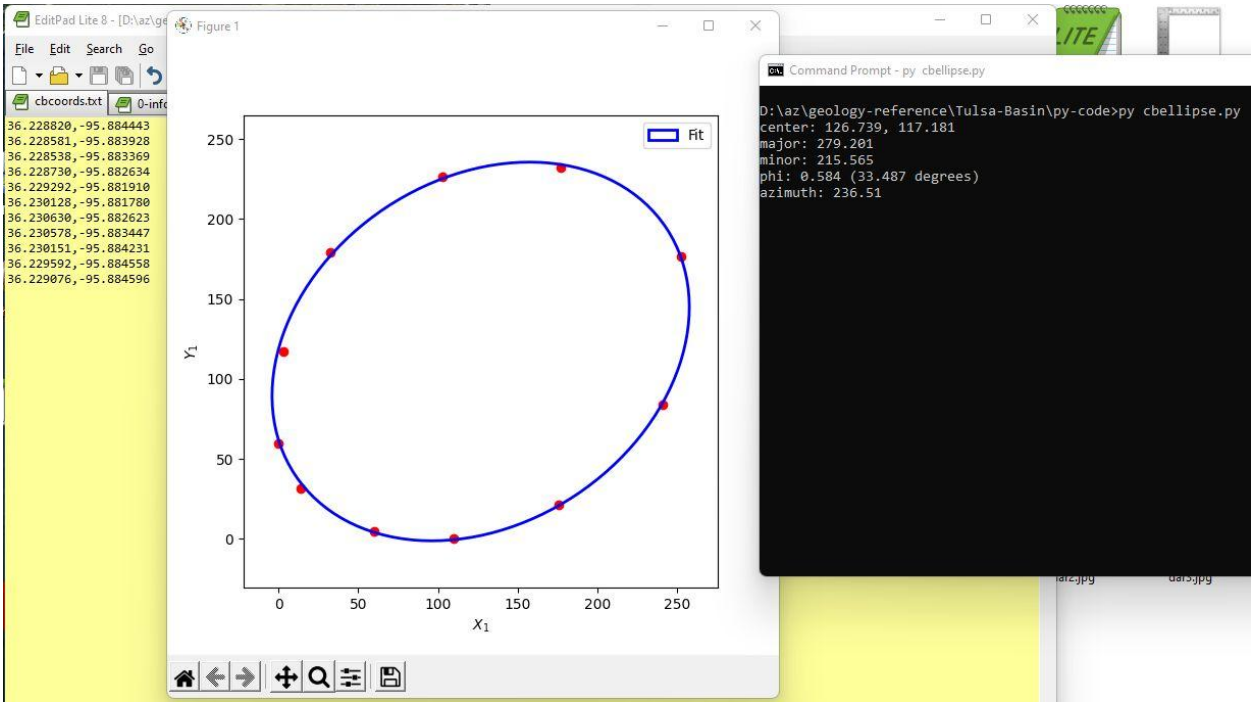
LiDAR image of Carolina Bays near Bowmore, NC
Three bays have been fitted with ellipses.

In many cases, it is possible to fit the ellipses by observation and trial-and-error, but it is desirable to be able to fit the ellipses by identifying points along the perimeter and using a least squares method to fit the ellipses. Ben Hammel and Nick Sullivan-Molina have developed a Python routine for least squares fitting of an ellipse based on a publication of Halir and Flusser. The program **ellipse.py** from the Github package had to be modified to output the coefficients of the polynomial for the ellipse.

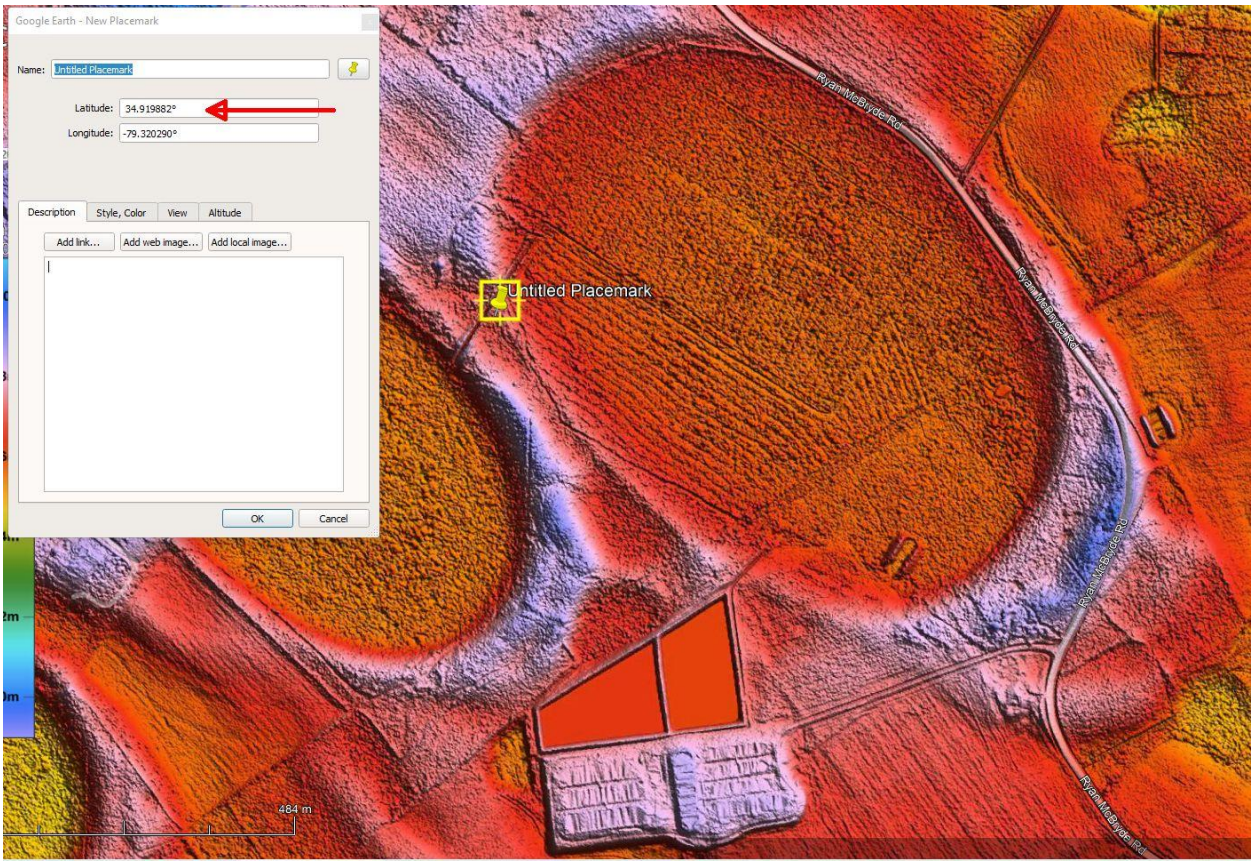
I created a program called **CBellipse.py** to read a file of latitude/longitude pairs as listed in Appendix I. The coordinates were changed to meters relative to the southmost and westmost positions so that the ellipse would be in the first quadrant. The procedure for converting the coordinates to meters recognizes that one degree of latitude is equal to 10,000,000 meters/90 degrees or 111,111 meters/degree. The distance in meters between degrees of longitude depends on the latitude. The program uses the cosine of the minimum latitude in the coordinate pairs and multiplies by 111,111.



LiDAR image of the Tulsa Basin. The coordinates of the rim are given in Sample Input file in Appendix I.

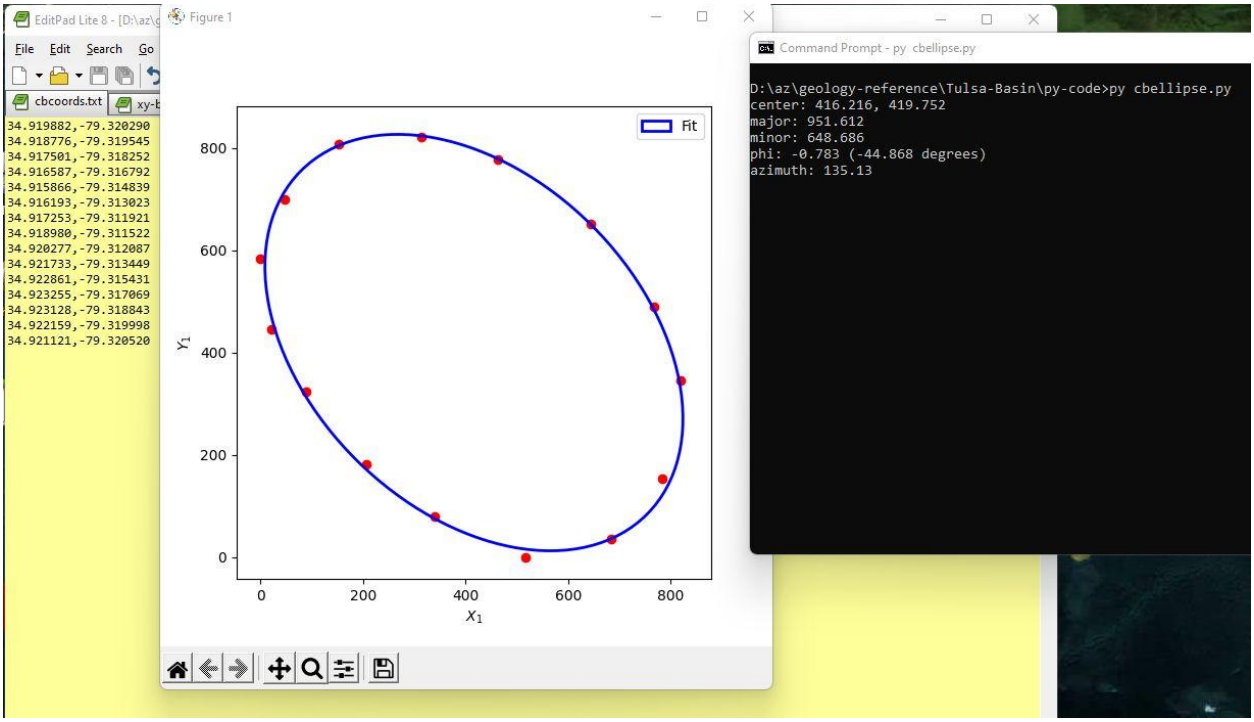


Results of processing the coordinate pairs of the Tulsa Basin



LiDAR image of Carolina Bays near Bowmore, NC.

The LiDAR visualization tool by Michael Davias [4] integrates with Google Earth and makes it possible to position the crosshairs of a pushpin along the rim of the bay in order to display a panel from which the coordinates for the latitude and longitude may be copied.



Results of processing the coordinate pairs of a Carolina Bay near Bowmore, NC

A second program called **driver2d.py** performs the functions of **CBellipse.py**, but it adds the capability of entering x,y coordinates from a digitized image instead of latitude and longitude by specifying *2D as a comment at the beginning of the data file.

References:

1) A. Zamora, A model for the geomorphology of the Carolina Bays, *Geomorphology*, 282, 209–216. (2017), DOI 10.1016/j.geomorph.2017.01.019 https://doi.org/10.1016/j.geomorph.2017.01.019

2) Ben Hammel, & Nick Sullivan-Molina. (2020, March 21). bdhammel/least-squares-ellipse-fitting: v2.0.0 (Version v2.0.0). Zenodo. http://doi.org/10.5281/zenodo.3723294

3) Least Squares fitting of ellipses, python routine
based on the publication Halir, R., Flusser, J.: 'Numerically Stable Direct Least Squares Fitting of Ellipses'
https://github.com/bdhammel/least-squares-ellipse-fitting/tree/v2.0.0

4) Davias, M. Visualization Tool Using Google Earth

http://cbaysurvey.cintos.org/

Appendix I

CBellipse.py

```
# Python program to fit an ellipse to Carolina Bay rim coordinates.
# A file "cbcoords.txt" contains more than six lines, each with the
#   latitude and longitude of a point along the Carolina Bay rim.
# Antonio Zamora July 5, 2022
# 07/07/2022 - Added BOM handling
# 07/11/2022 - Corrected problem with lat./lon.
#            allowed comment lines starting with # or *
#  A title for the graph is specified with a line starting with *T= title
# 07/21/2022 - restricted azimuth calculation for 28>Lat<49 & -105<Lon<-66

import numpy as np
from ellipse import LsqEllipse
import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse

# minlat and maxlon are global variables
minlat = ""
maxlon = ""
calc_azimuth = 0  # =1 to calculate azimuth
title = ""

# * * * * * * *
# Read file with coordinates and return two lists containing the x and y data of the
ellipse
# * * * * * * *
def read_coordinates():
   ellipse_x= []
   ellipse_y = []
   list1= []
   list2 = []
   global minlat
   global maxlon
   global calc_azimuth
   global title

   file1 = open('cbcoords.txt', 'r', encoding='utf-8-sig') # remove BOM
   Lines = file1.readlines()
```

```
    for line in Lines:
      line = line.strip()    # remove trailing spaces and \r
      # print("line:", line)
      if line[0:3] == "*T=" :   # check for title line
        title = line[3:]
      # Skip blank lines and comments starting with * or #
      if not line.startswith('*') and not line.startswith('#') and not line == '':
        line = line.split(",")
        # print("Lat: ", line[0], " Lon: ", line[1])  #print latitude and longitude
        list1.append(line[0])   # Latitude
        list2.append(line[1])   # Longitude
    file1.close()

    # Find minimum X and Y coordinates
    minlat = min(list1)  # minimum latitude (southmost)
    maxlat = max(list1)  # maximum latitude
    maxlon = max(list2)  # maximum longitude (westmost)
    minlon = min(list2)  # minimum longitude
    # print("minLat: ", minlat," maxlat: ",maxlat, " maxLon: ", maxlon, " minLon: ",minlon)
    # 28>Lat<49 & -105<Lon<-66  [coordinates are in the contiguous United States]
    if float(minlat) > 28.0 and float(maxlat) < 49.0 and float(maxlon) > -105. and
float(minlon)< -66.0 :
      calc_azimuth =1
    # print("cos(minlat)=", np.cos(np.radians(float(minlat))) )

    # Convert latitude and longitude to meters relative to minimum coordinates
    # This will place the ellipse in the first quadrant
    # One degree of latitude = 10,000,000 m/90 degrees = 111,111 meters/degree
    for j in list1:
      # print("latitude ", j)
      ellipse_y.append( (float(j) - float(minlat))*111111 ) # Subtract latitude from minlat
and convert to meters
      # print( (float(j) - float(minlat))*111111  )

    # Process Longitude
    # the distance in meters between degrees of longitude depends
    #    on the latitude: cos(minlat)*111111
    for j in list2:
      # print("longitude ", j)
```

```
    ellipse_x.append( (abs(float(maxlon)) -
abs(float(j)))*111111*np.cos(np.radians(float(minlat)) ) )
    # print( (abs(float(maxlon)) - abs(float(j)))*111111*np.cos(np.radians(float(minlat)) )
)

  return [ellipse_x, ellipse_y]

# * * * * * * * *
# Make test ellipse
# * * * * * * * *
def make_test_ellipse(center=[1, 1], width=1, height=.6, phi=3.14/5):
  """Generate Elliptical data with noise

  Parameters
  ----------
  center: list:float
    (<x_location>, <y_location>)
  width: float
    semimajor axis. Horizontal dimension of the ellipse (**)
  height: float
    semiminor axis. Vertical dimension of the ellipse (**)
  phi: float:radians
    tilt of the ellipse, the angle the semimajor axis
    makes with the x-axis

  Returns
  -------
  data:  list:list:float
    list of two lists containing the x and y data of the ellipse.
    of the form [[x1, x2, ..., xi],[y1, y2, ..., yi]]
  """
  t = np.linspace(0, 2*np.pi, 300)
  x_noise, y_noise = np.random.rand(2, len(t))

  ellipse_x = center[0] + width*np.cos(t)*np.cos(phi)-height*np.sin(t)*np.sin(phi) +
x_noise/2.  # noqa: E501
  ellipse_y = center[1] + width*np.cos(t)*np.sin(phi)+height*np.sin(t)*np.cos(phi) +
y_noise/2.  # noqa: E501

  return [ellipse_x, ellipse_y]
```

```
# * * * * * * *
# Main program
# * * * * * * *
if __name__ == '__main__':

   X1, Y1 = read_coordinates()
#    X1, Y1 = make_test_ellipse()

   X = np.array(list(zip(X1, Y1)))
   reg = LsqEllipse().fit(X)
   center, semimajor, semiminor, phi = reg.as_parameters()
   # [to do] get percent error in fit

   # print("minlat=", minlat, " maxlon=", maxlon)
   # [to do] For incomplete ellipse data points, we need to adjust center to get correct
coordinates
   #        Get minimum coordinates of ellipse path (may be negative)
   print("coefficients for ax**2 + 2bxy + cy**2 + 2dx + 2fy + g")
   a, b, c, d, f, g = reg.coeffs()
   print(f'a={a:.3f}, b={b:.3f}, c={c:.3f}, d={d:.3f}, f={f:.3f}, g={g:.3f}')

   print(f'center: {center[0]:.3f}, {center[1]:.3f}')

   # Calculate coordinates for center
   Latitude = float(minlat) + center[1]/111111
   Longitude = float(maxlon) + center[0]/(111111*np.cos(np.radians(float(minlat))))
   print(f'center Lat.,Lon. {Latitude:.6f}, {Longitude:.6f}')

   k1 = semimajor
   k2 = semiminor
   if k1 < k2:  # swap so major axis is larger than minor axis
     j = k2;
     k2 = k1
     k1 = j

   print(f'major: {2*k1:.3f}')
   print(f'minor: {2*k2:.3f}')
   print(f'phi: {phi:.3f} ({np.rad2deg(phi):.3f} degrees)')
   # calculate azimuth
```

```
  if calc_azimuth == 1 :
    if phi < 0:      # phi is negative
      # print("phi < 0")
      azrad = np.pi/2 + abs(phi)  # |phi| + 90 degrees
    elif semimajor < semiminor:     # case when major and minor axes were flipped.
      # print("semimajor < semiminor")
      azrad = np.pi - phi         # 180 - phi (phi>0)
    else:
      # print("semimajor > semiminor")
      azrad = np.pi + np.pi/2 - phi

    # convert radians to degrees
    print(f'azimuth: {np.degrees(azrad):.2f}')
  # end if calc_azimuth == 1

  fig = plt.figure(figsize=(6, 6))
  ax = plt.subplot()
  ax.axis('equal')
  ax.plot(X1, Y1, 'ro', zorder=1)
  ax.set_title(f'{title}\n major axis: {2*k1:.1f} m, minor axis: {2*k2:.1f} m')
  ax.plot(center[0], center[1], 'go', label='center')
  ellipse = Ellipse(
     xy=center, width=2*semimajor, height=2*semiminor, angle=np.rad2deg(phi),
     edgecolor='b', fc='None', lw=2, label='Fit', zorder=2
  )
  ax.add_patch(ellipse)

  plt.xlabel('$X_1$')
  plt.ylabel('$Y_1$')

  plt.legend()
  plt.show()
```

SAMPLE INPUT FILE for the Tulsa basin

```
36.228820,-95.884443
36.228581,-95.883928
36.228538,-95.883369
36.228730,-95.882634
36.229292,-95.881910
36.230128,-95.881780
```

36.230630,-95.882623
36.230578,-95.883447
36.230151,-95.884231
36.229592,-95.884558
36.229076,-95.884596