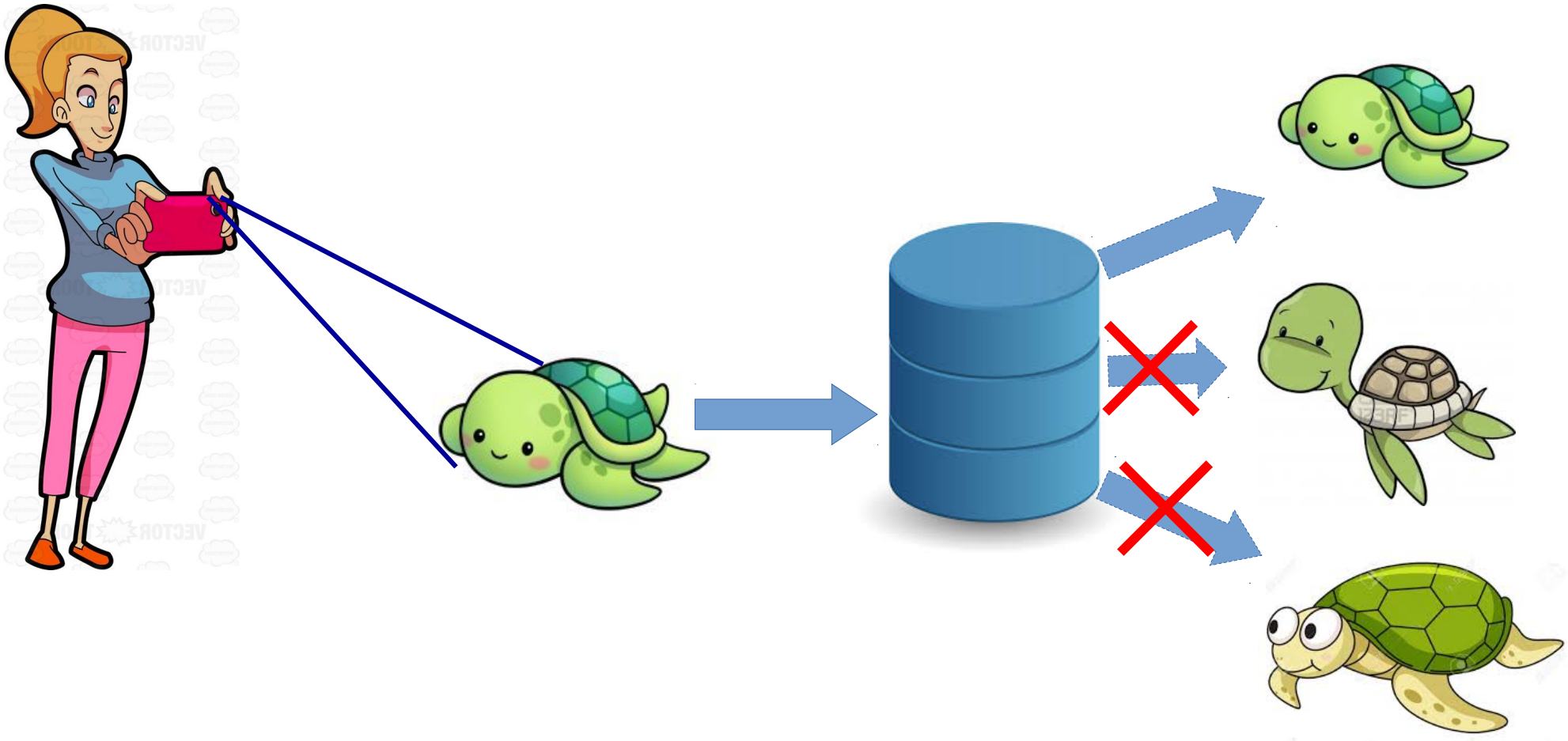
Photographic Identification of Sea Turtles using Python and OpenCV

Natapon Pantuwong

PyCon2018, Bangkok

This slide can be found at **https://github.com/pantuwong/pycon2018.git**

What’s photo identification for sea turtle



Why I come with this system

- I have a friend who works at Sea Turtle Conservation Center
  - Main mission are captive-rearing and release of turtle hatchlings
- He needs to track the released turtles when they are found again
- With statistical data, we can analyze the population of sea turtle and plan how can we protect them

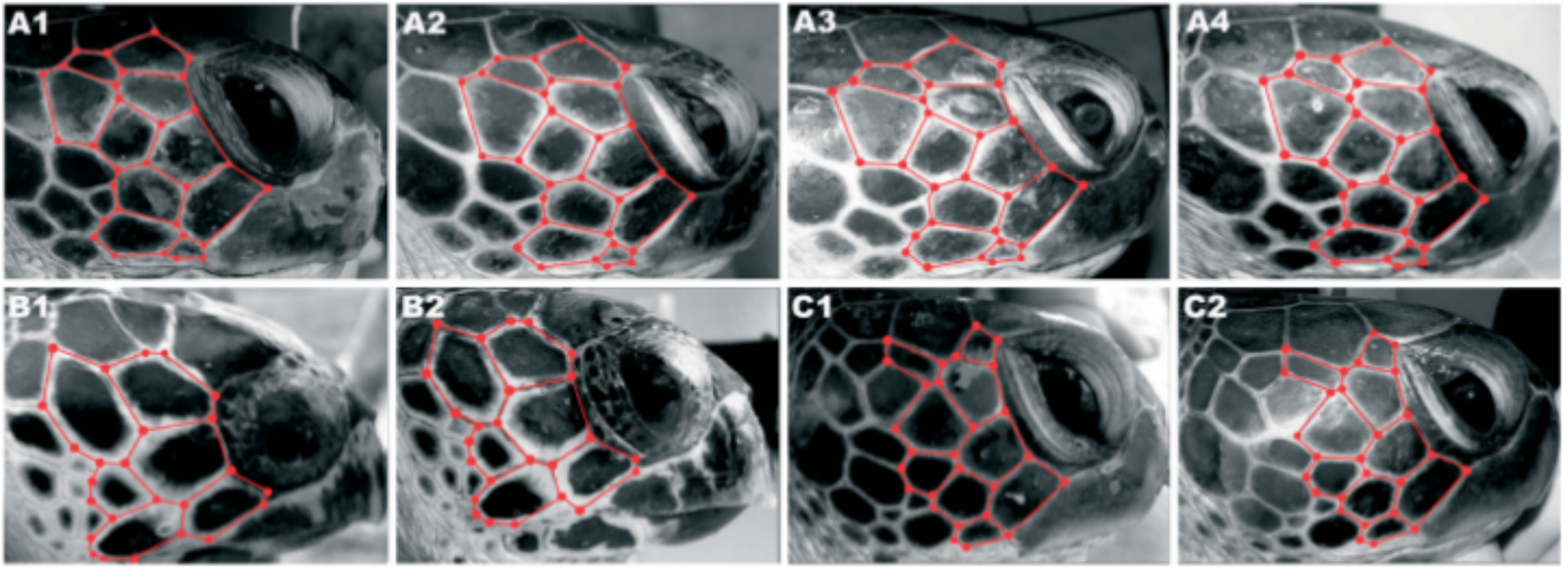
Why I come with this system (con't)

- Normally, my friend uses a tag attached to sea turtle, but it is not the best solution.

- When a turtle go into the ocean, it might be attacked and the tag might be lost
- It is easier, if we can just take its photo, and identify it with our database

How can we identify a sea turtle using photo

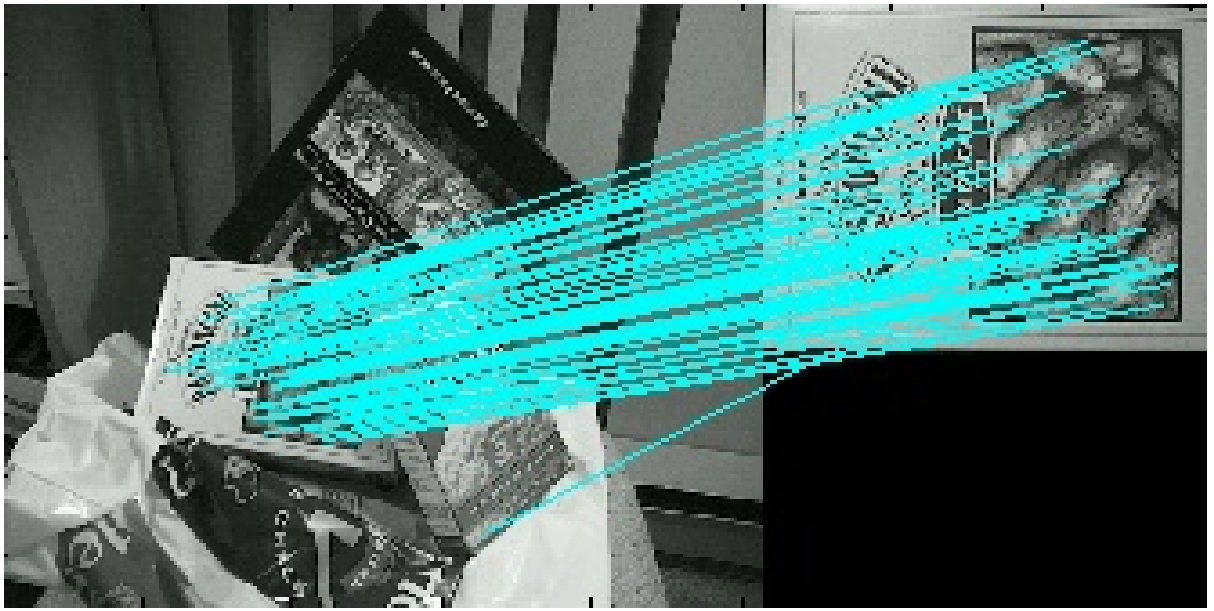
- Sea turtles have a unique pattern of facial scutes



These photos have been taken in different time during 3 years

How can we identify a sea turtle using photo

- So, identifying sea turtle could be solved by using image (of facial scute) matching
- A good algorithm to start is SIFT by Prof.David Lowe



How Python and OpenCV do SIFT

- SIFT feature extraction

  # Initiate SIFT detector

  sift = cv2.SIFT()

  # find the keypoints and descriptors with SIFT

  kp1, des1 = sift.detectAndCompute(img1,None)

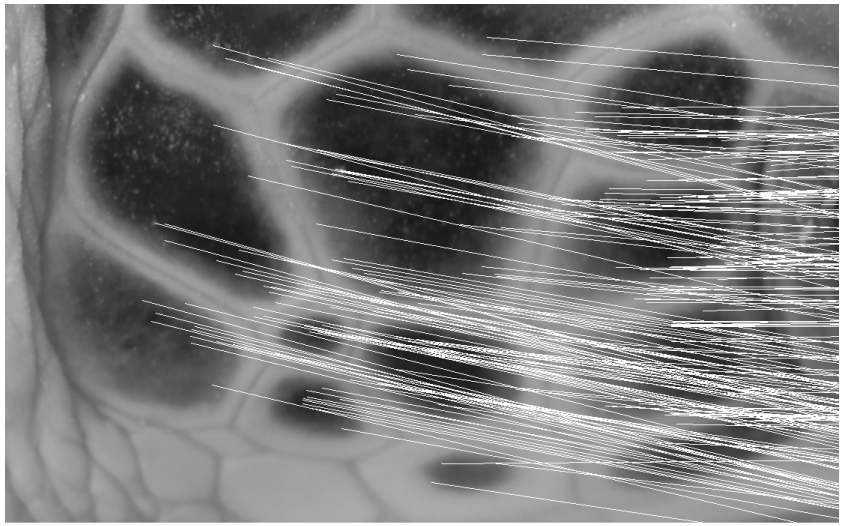
  kp2, des2 = sift.detectAndCompute(img2,None)

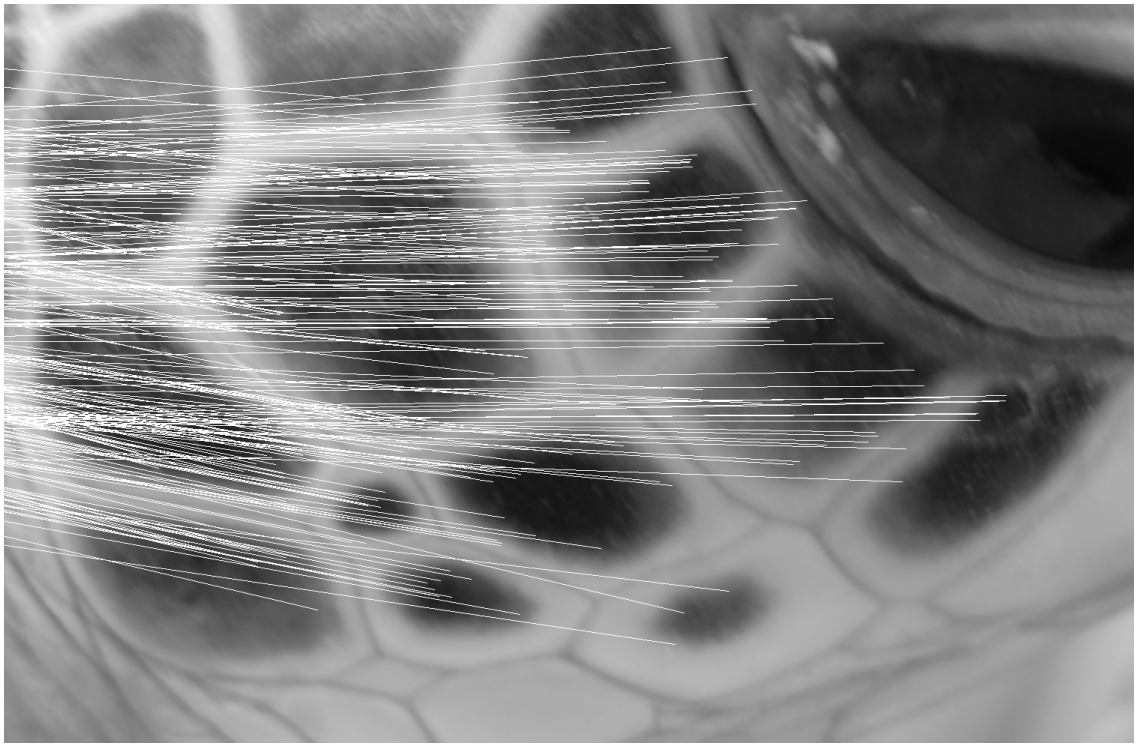
How Python and OpenCV do SIFT

- SIFT feature matching

```
# BFMatcher with default params
bf = cv2.BFMatcher()
matches = bf.knnMatch(des1,des2, k=2)
```

Results





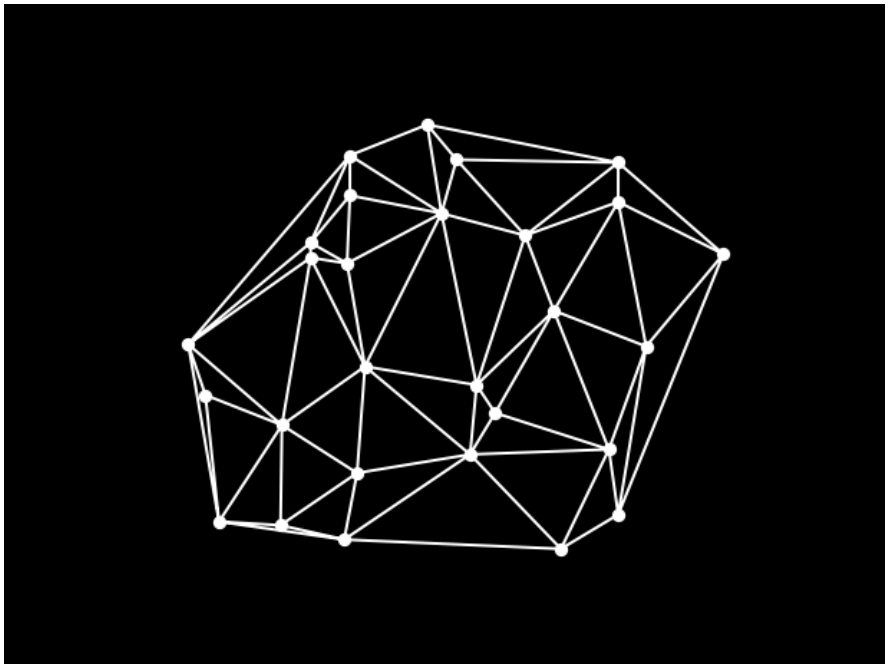
Is it promising method?

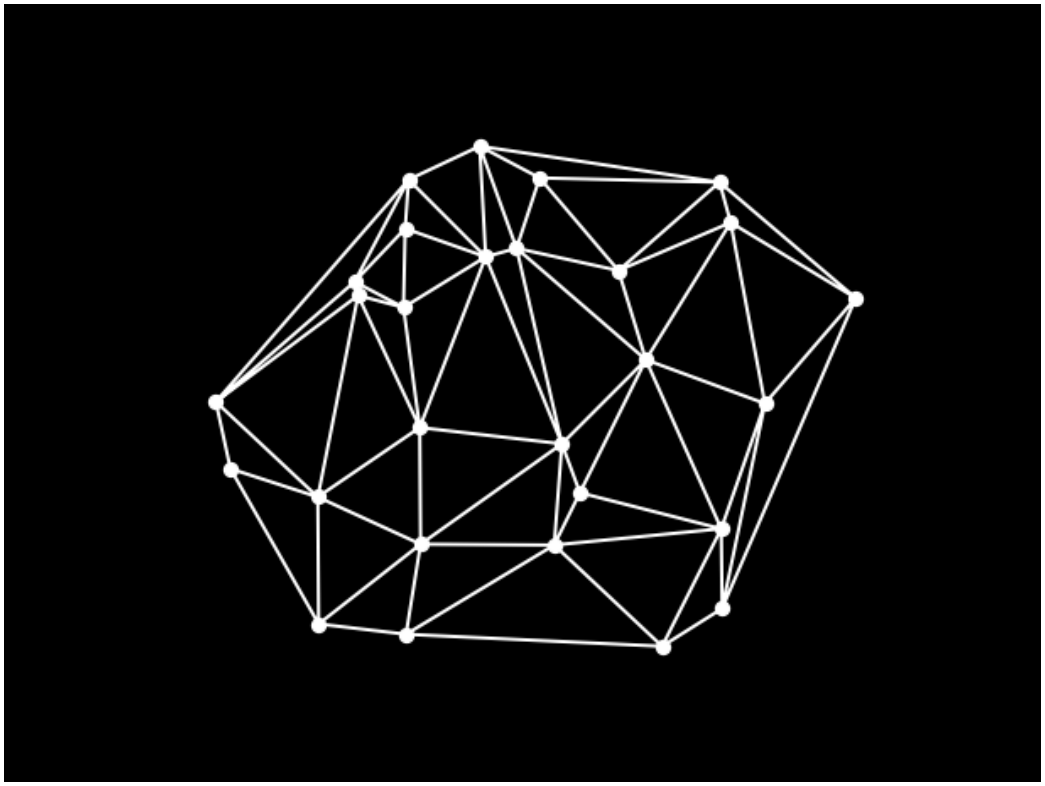
- Unfortunately, it is not a promising method
  - SIFT it not invariant against color and intensity change
  - Scute pattern doesn’t change, but color change

Another idea to solve this problem

- We need only information about scute pattern, not its color or intensity
- Extract scute edge is not an easy problem
- Need user involvement
  - User selects all corners of scutes
  - Create a mesh from all corner positions using Delaunay triangulation
  - Mesh matching by apply SIFT to mesh image

Example





Delaunay Triangulation with Python

- Use Scipy to triangulate

  >>> points = np.array([[0, 0], [0, 1.1], [1, 0], [1, 1]])

  >>> from scipy.spatial import Delaunay

  >>> tri = Delaunay(points)

- Use Matplotlib to plot

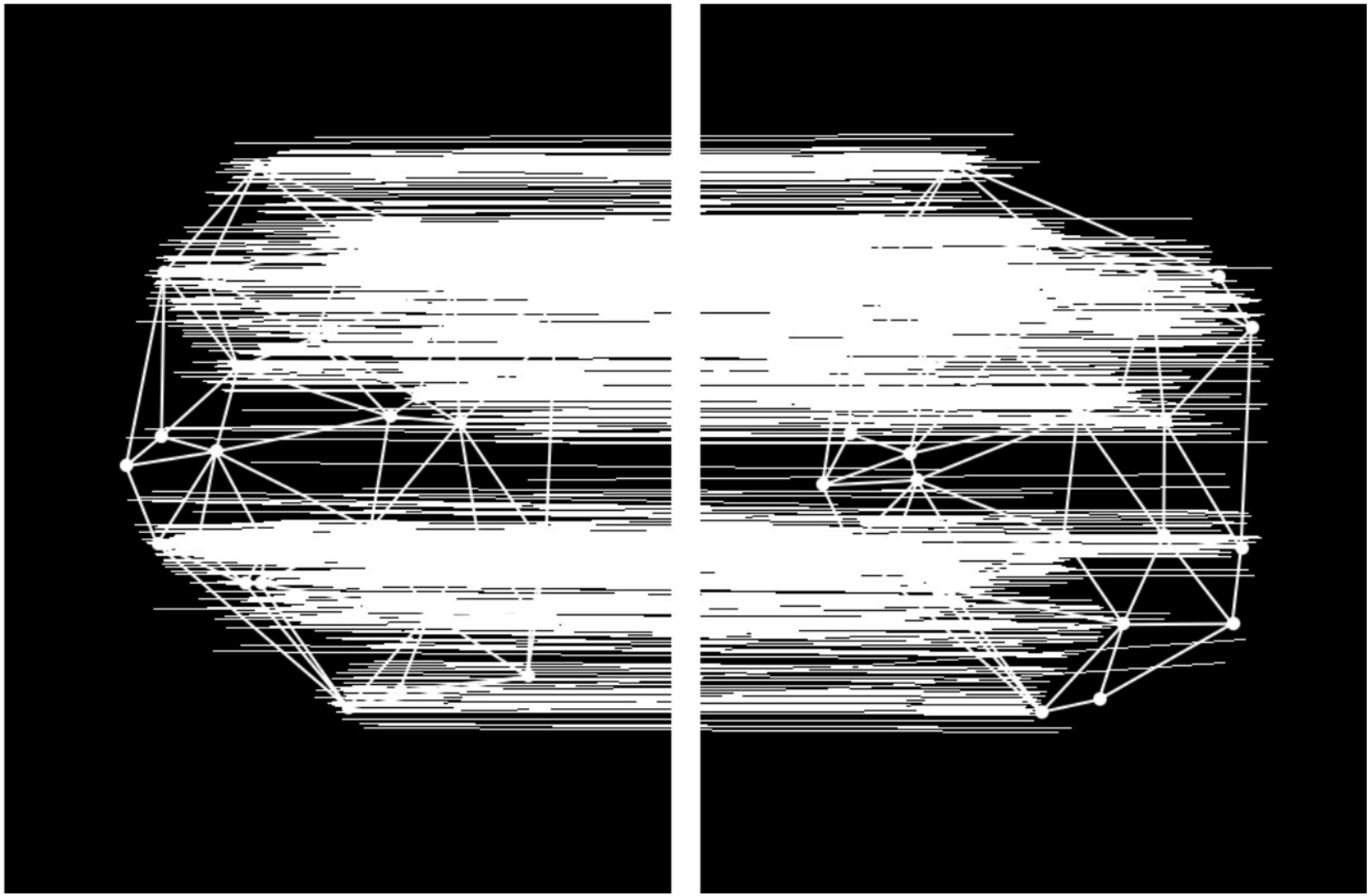
  >>> import matplotlib.pyplot as plt

  >>> plt.triplot(points[:,0], points[:,1], tri.simplices.copy())

  >>> plt.plot(points[:,0], points[:,1], 'o')

  >>> plt.show()

Result



Let's conserve our sea turtles