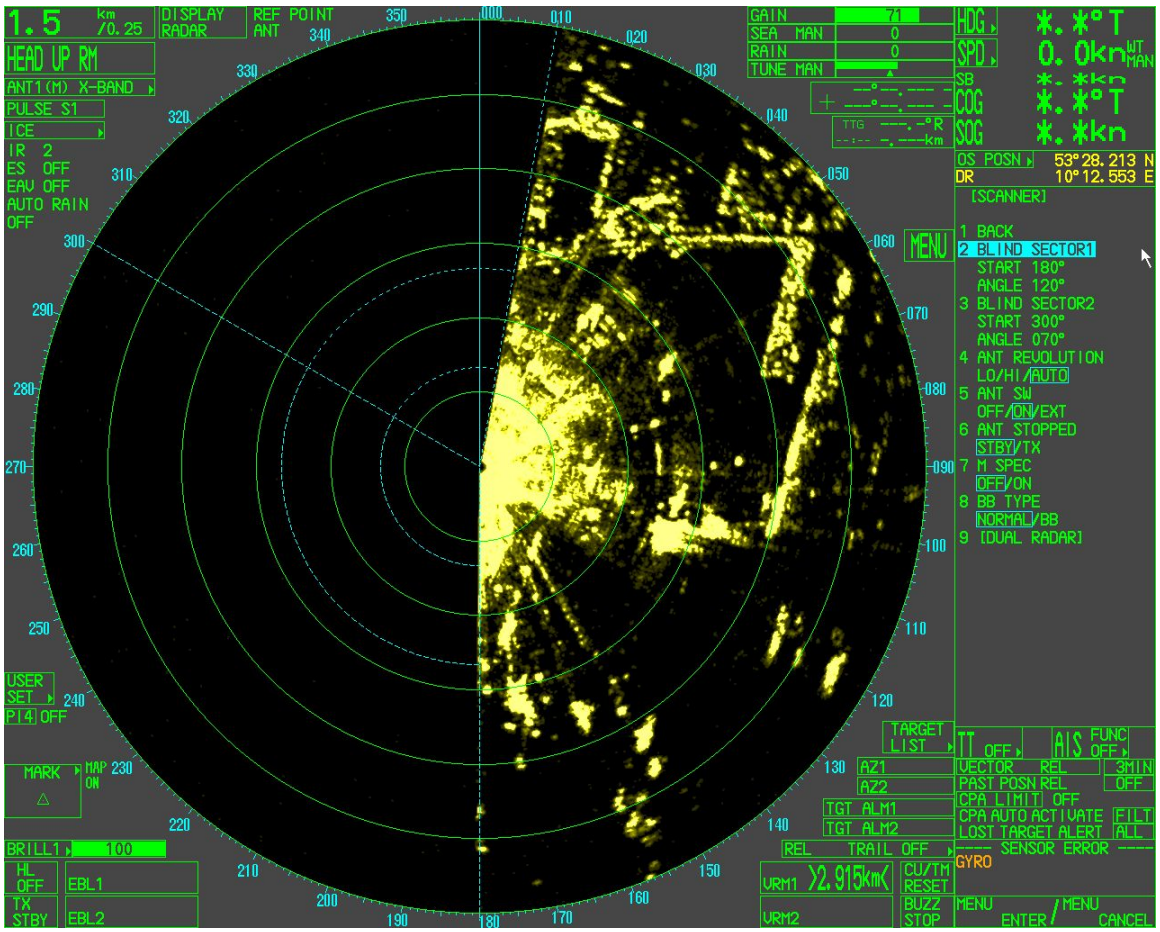
DUY NGUYEN - HAW HAMBURG

RADARBILDER OPTIMIERUNG
FLEDERWIND Projekt



Einleitung

Bei dem Projekt haben wir Radar eingestellt und dadurch möchten wir irgendwie Fledermäuse in der Umgebung zu tracken. Einer von vielen schwierigen Aspekte ist, dass die Radarbilder unheimlich viel Störsignale enthält und man kann mit blöden Augen auch nicht so gut betrachten. Deswegen führe ich diese Radarbilder-Optimierung zu, wird auch als Präprozessor der Radarbilder, damit wir echt bessere Motion-tracking mit Beobachter beim Monitoring bzw. später mit Maschinen tracken können.

Zu dem Code (DelNoi.py *deleteNoise.py)

Nach mehreren Versuchen mit OpenCV, Pillow (PIL - Python Imaging Library) gehe ich davon aus, dass es möglich ist, in Pixels-level die unverbrauchten Signale/Informationen aus den Radarbilder zu löschen und komprimieren. Damit wir bessere die Änderungen aus den Bilder betrachten können und dadurch, mehrere hilfreiche Informationen davon extrahieren dürften (Bewegungen, Objekte, ..)

Dieses Programm hat einige Hauptfunktionen, die nämlich:

- Die Störsignale aus eine Menge von Radarbilder zählen
- Alle Pixels, die als Störsignale entnommen wurden, wird in jedem Radarbild ausgeblendet
- Alle analysierte Bilder werden mit ffmpeg in einem Video zusammengefasst.

Mehr Details

Verwendete Bibliothek:

- time
- os, os.path
- numpy
- PIL (Module Image)

*Wie kann man Bibliothek in Python installieren?:

- Install pip
- pip install <name of bibliothek> *in PyPi Webseite gesuchter Name

Oder in Anaconda kann man auch die Bibliothek importieren.

Lernquellen

Pillow - Python Imaging Library

https://pillow.readthedocs.io/en/stable/handbook/index.html

Und mehrere Stackoverflow Diskussionen.

Google :)

Code delNoi.py

```
1. import time
2. import os, os.path, PIL
3. import numpy
4. from PIL import Image
5.
6. #Start time counter
7. t0 = time.time()
8. print("="*80)
9. print("Welcome to the Noise Reduction Program! All radar images in the Directory
   in the python file are going to be converted!")
10.
11.   #Access all images files in directory
12.   imageDirectory = "Radarbilder" #change here to change the image directory
   (which in the directory of this python file)
13.
14.   #create a list of filenames in imagesDirectory
15.   imagesNameList = [filename for filename in os.listdir(imageDirectory)]
16.   imagesNameList.sort() #sorting namelist a-z
17.
18.   #Assuming all image files in directory have the same size, get dimensions of
   first image
19.   imageModel = Image.open(imageDirectory + "/" + imagesNameList[0])
20.   rows = imageModel.height #number of rows of pixels
21.   columns = imageModel.width   #number of columns of pixels
22.
23.   #Create a 2D array of integer to store the counter of color's appear
24.   #Assume that all elements of matrix has the value 0
25.   counterMatrix = numpy.zeros((rows, columns), dtype = int)
26.
27.   #Change this value to change how much images are analyzed to count the noise
28.   counter = 100
29.   for indexofImage in range(counter): # the first "counter" images is taken to
   analyse the noise (noise appear in the same spot > 1 times in 100 images)
30.       image = Image.open(imageDirectory + "/" + imagesNameList[indexofImage])
31.       loadedPixelMap = image.load() #loading the (R,G,B) array of all pixel
```

```
32.      #for every columns and rows of matrix
33.      for column in range(columns):
34.          for row in range(rows):
35.              if loadedPixelMap[column, row] != (0,0,0): #if pixel is not black
   -> (!signal) -> add one to counterMatrix on the spot of this pixel
36.                  counterMatrix[row, column] = counterMatrix[row, column] + 1
37.
38.   t1 = time.time()
39.   print("="*80)
40.   print("Time required: ",t1 - t0)
41.   print("End of noise counter!")
42.   print()
43.
44.   for indexofImage in range(len(imagesNameList)): #analyze all images in
   folders and delete the unused colors
45.      #Load pixels of an image and go though all pixels
46.      image = Image.open(imageDirectory + "/" + imagesNameList[indexofImage])
47.      loadedPixelMap = image.load()
48.      #image.size[columns, rows]
49.      #delete all not interested colors
50.      for column in range(columns):
51.          for row in range(rows):
52.              if  loadedPixelMap[column, row] == (70,70,70) or \
53.                  loadedPixelMap[column, row] == (35,163,163) or \
54.                  loadedPixelMap[column, row] == (0,255,0) or \
55.                  loadedPixelMap[column, row] == (255,170,0) or \
56.                  loadedPixelMap[column, row] == (0,255,255) or \
57.                  loadedPixelMap[column, row] == (255,255,0) or \
58.                  loadedPixelMap[column, row] == (0,157,157) or \
59.                  loadedPixelMap[column, row] == (255,0,0) or \
60.                  loadedPixelMap[column, row] == (255,255,255):
61.                  loadedPixelMap[column, row] = (0,0,0)
62.      #os.remove(imageDirectory + "/" + imagesNameList[indexofImage])
63.      image.save(imageDirectory + "/" + imagesNameList[indexofImage])
64.
65.   t2 = time.time()
66.   print("="*80)
67.   print("Time required to convert full radar images to yellow signal images:
   ",t2-t1)
68.   print(f"{len(imagesNameList)} images are converted! Congratulations!")
69.   print()
70.
71.   #NOTE not interested color list
72.   # color: (R,G,B)
73.   # lightblue1(coordinates) : (35,163,163)
74.   # lightblue2(coordinates) : (0,255,255)
75.   # darkblue(radar's lines) : (0,157,157)
76.   # green(data numbers) :     (0,255,0)
77.   # trueyellow(numbers) :     (255,255,0)
78.   # orangeyellow(numbers) :   (255,170,0)
```

```
79.   # red(time) :                (255,0,0)
80.   # gray(background) :         (70,70,70)
81.   # white(cursor) :            (255,255,255)
82.   #Read the Matrix and analyse
83.   #Go through all images and delete the Noise
84.   for indexofImage in range(len(imagesNameList)):
85.      image = Image.open(imageDirectory + "/" + imagesNameList[indexofImage])
86.      loadedPixelMap = image.load()
87.      for column in range(columns): #for all columns
88.          for row in range(rows): #for all rows
89.              if counterMatrix[row, column] > 1:
90.                  #loadedPixelMap[column, row] = (122,122,190) #toggle this line
   and comment the next line to see the noise
91.                  loadedPixelMap[column, row] = (0,0,0)
92.      image.save(imageDirectory + "/" + imagesNameList[indexofImage])
93.
94.   t3 = time.time()
95.   print("="*80)
96.   print("Time required to delete noises to all radar images: ",t3 - t2)
97.   print("="*80)
98.   print()
99.   print("="*80)
100.  os.system("ffmpeg -framerate 20 -i " + imageDirectory + "/" +
   imagesNameList[0][:-4] + "%08d.png " + "ffmpeg.mp4") #when bmp images are used,
   change png to bmp with the script changetoBMP.py
101.  print("All the noise reduced radar images is saved in video ffmpeg.mp4 in
   20fps!")
102.
103.  #End time counter
104.  t4 = time.time()
105.  print("="*80)
106.  print()
107.  print("Time required for all processes: ", t4 - t0)
108.  print("Congratulations! Alles Gute!")
```

Anmerkungen:

Es koennte problematisch dabei, wie wir die Radarbilder einstellen. Am Besten waere etwas mit “cap_000001.png" so aehnlich zu benennen. Da wir die Ansatz “imagesNameList[0][:-4]” + “%08d.png” nutzen, was den ersten Name von NameList nehmen, aber die letzte 4 Zeichen werden abgeloescht: “.png” und dann wieder mit “%08d.png” ersetzt. Das ist nur ein Problem, wenn wir dann mit ffmpeg arbeiten, es gibt mehrere alternative Loesung, besonders fuer Windows Maschine, also nur wenn interessiert bitte einfach auf Linux gehen *Ubuntu/Arch/Manjaro,usw... und ueber ffmpeg informieren.